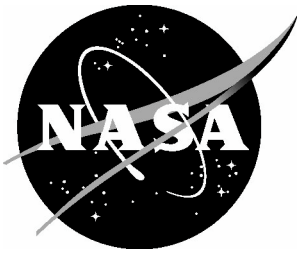
NASA/TM-2005-213934



Design of the Protocol Processor for the ROBUS-2 Communication System

Wilfredo Torres-Pomales, Mahyar Malekpour and Paul S. Miner
Langley Research Center, Hampton, Virginia

November 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

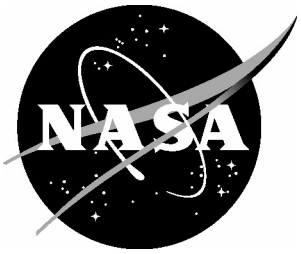
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2005-213934



Design of the Protocol Processor for the ROBUS-2 Communication System

Wilfredo Torres-Pomales, Mahyar Malekpour and Paul S. Miner
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

November 2005

Acknowledgement

This work was supported, in part, by the FAA William J. Hughes Technical Center under interagency agreement DTFA03-96-X90001.

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Abstract

The ROBUS-2 Protocol Processor (RPP) is a custom-designed hardware component implementing the functionality of the ROBUS-2 fault-tolerant communication system. The Reliable Optical Bus (ROBUS) is the core communication system of the Scalable Processor-Independent Design for Enhanced Reliability (SPIDER), a general-purpose fault-tolerant integrated modular architecture currently under development at NASA Langley Research Center. ROBUS is a time-division multiple access (TDMA) broadcast communication system with medium access control by means of time-indexed communication schedule. ROBUS-2 is a developmental version of the ROBUS providing guaranteed fault-tolerant services to the attached processing elements (PEs), in the presence of a bounded number of faults. These services include message broadcast (Byzantine Agreement), dynamic communication schedule update, time reference (clock synchronization), and distributed diagnosis (group membership). ROBUS also features fault-tolerant startup and restart capabilities. ROBUS-2 tolerates internal as well as PE faults, and incorporates a dynamic self-reconfiguration capability driven by the internal diagnostic system. ROBUS consists of RPPs connected to each other by a lower-level physical communication network. The RPP has a pipelined architecture and the design is parameterized in the behavioral and structural domains. The design of the RPP enables the bus to achieve a PE-message throughput that approaches the available bandwidth at the physical layer.

Table of Contents

Notation

The following table lists the symbols used in this document. Many of the symbols have generic descriptions and are reused where applicable. Subscripts are used to distinguish different quantities labelled with the same main symbol.

Symbol	Description
N	• Total number of BIUs in the system • Range: Positive Integers; Unit: None
M	• Total number of RMUs in the system • Range: Positive Integesr; Unit: None
n	• Total number of BIUs trusted by a node • Range: Naturals; Unit: None
m	• Total number of RMUs trusted by a node • Range: Naturals; Unit: None
Ω	• Total number of nodes of the opposite kind • Range: Positive Integers; Unit: None
Θ	• Total number of nodes of the same kind • Range: Positive Integers; Unit: None
ω	• Total number of nodes of the opposite kind trusted by a node • Range: Naturals; Unit: None
θ	• Total number of nodes of the same kind trusted by a node • Range: Naturals; Unit: None
$IMP(x_1, x_2)$	• IMP = Integer Mid-Point function (= Rounded Average) • Integer closest to the mid-point between x_1 and x_2 • Algorithm: Step 1: $x = (x_1 + x_2)/2$ Step 2: IMP = round(x), with $round(x) = \lfloor x \rfloor$ if $x < ½$, or $\lceil x \rceil$ if $x \geq ½$ • Range: Integers; Unit: Same as x1 and x2
L_{LS}	• Link syndrome width; number of syndrome bits for each receiver • Range: Naturals; Unit: None
L_{PF}	• Payload field width for ROBUS messages • Range: Positive Integers; Unit: None
f	• Clock frequency • Range: Reals; Unit: Hertz (Hz = Cycles per second)
τ	• Granularity of a clock (i.e., duration of a clock tick) • Range: Positive Reals; Unit: Second
t	• Real time • Range: Reals; Unit: Nominal Tick
T	• Clock time • Range: Integers; Unit: Local Tick
ρ	• Bound on the drift rate a clock signal generated by a physical oscilaltor • Range: Reals; Unit: None
k	• Frequency division factor (= Frequency step-down factor from physical-oscillator frequency to local-clock frequency) • Range: Positive Integers; Unit: None or Tick/Tick
$c_x(T)$	• Earliest real time at which clock x reaches value T • Range: Reals; Unit: Nominal Tick
π	• Bound on the relative time skew of particular events among specified nodes • Range: Non-negetive Reals; Unit: Tick

Symbol	Description
Π	• Bound on the observed relative time skew of particular events • Range: Positive Integers; Unit: Local Tick
δ	• General real-time delay • Range: Reals; Unit: Nominal Tick
Δ	• General clock-time delay • Range: Integers; Unit: Local Tick
d	• Message delivery delay • Range: Non-negative Reals; Unit: Nominal Tick
v	• Uncertainty in message delivery delay • Range: Non-negative Reals; Unit: Nominal Tick
r	• Message reception delay • Range: Non-negative Reals; Unit: Nominal Tick
e	• Uncertainty in message reception delay (= Network imprecision) • Range: Non-negative Reals; Unit: Nominal Tick
R	• Expected reception delay • Range: Naturals; Unit: Local Tick
μ	• Bound on error of expected reception delay • Range: Reals; Unit: Nominal Tick
W	• Window size • Range: Naturals; Unit: Local Tick
C	• Computation Process delay with respect to the end of the deskew window for time-driven protocols • Range: Naturals; Unit: Local Tick
S	• Send Process delay with respect to a reference time for time-driven protocols • Range: Naturals; Unit: Local Tick
A	• Accept function delay with respect to the time of reception of the selected event for synchronization protocols • Range: Naturals; Unit: Local Tick
B	• Send Process delay with respect to a reference event for synchronization protocols • Range: Naturals; Unit: Local Tick
H	• Reset delay with respect to a reference event • Range: Naturals; Unit: Local Tick
λ	• Data Introduction Interval • Range: Reals; Unit: Nominal Tick
Λ	• Data Introduction Interval • Range: Positive Integers; Unit: Local Tick
K	• Number of messages in a stream • Range: Positive Integers; Unit: None
P (uppercase)	• Nominal resynchronization period • Range: Positive Integers; Unit: Local Tick
p (lowercase)	• Nominal resynchronization period • Range: Positive Reals; Unit: Nominal Tick

1. Introduction

The Reliable Optical Bus (ROBUS) is the core communication system of the Scalable Processor-Independent Design for Enhanced Reliability (SPIDER), a general-purpose fault-tolerant integrated modular architecture (IMA) currently under development at NASA Langley Research Center. The purpose of this effort is to produce a flexible architecture that can be configured to satisfy a wide range of performance and reliability requirements, while preserving a consistent interface to application programs. The architecture is expected to support functions of various criticality levels, including ultra-reliable and safety-critical aircraft functions with hard real-time deadlines.

ROBUS is a time-division multiple access (TDMA) broadcast communication system with medium access control by means of a time-indexed communication schedule. ROBUS-2, a developmental version of ROBUS, provides the following guaranteed fault-tolerant services to the attached processing elements (PEs): message broadcast (Byzantine Agreement), dynamic communication schedule update, time reference (clock synchronization), and distributed diagnosis (group membership). The bus is tolerant to internal as well as PE faults, and incorporates a dynamic self-reconfiguration capability driven by the internal diagnostic system. ROBUS-2 also features fault-tolerant startup and restart capabilities.

ROBUS-2 is intended for laboratory experimentation and demonstration of the capability to reintegrate repaired nodes, dynamically update the communication schedule, and tolerate and recover from correlated transient faults. ROBUS-2 is also intended to demonstrate that the bus is an efficient communication system that can achieve a PE-message throughput that approaches the available bandwidth at the physical communication layer, while preserving the fault-tolerance guarantees. A thorough description of the high-level ROBUS-2 design can be found in [Torres 05].

ROBUS is a distributed system consisting of a set of dedicated ROBUS Protocol Processors (RPP) communicating over a lower-level physical communication network. For ROBUS-2, the RPPs are custom hardware-based components that implement the ROBUS functionality. The RPPs have simple strobe-in/strobe-out synchronous interfaces to the PEs and to the lower-level communication network, and they are capable of processing messages at a rate of one message per clock tick. The RPPs have a large array of error detectors for self-checks, checks of remotes nodes, and checks for the integrity of the bus. The RPPs can also read and diagnose error syndromes generated by the lower-level network.

The RPPs are described in the VHDL (Very High Speed Integrated Circuit Hardware Description Language) language and are implemented on Field-Programmable Gate Arrays (FPGA). The VHDL code is highly parameterized in the behavioral (e.g., time to wait before asserting a signal) and structural (e.g., size of a buffer) domains in order to enable customization of the bus with respect to the targeted application. Most of the parameters must be specified before VHDL synthesis. Parameters that uniquely identify a particular RPP within the system (e.g., the assigned unique identification number) can be specified pre-synthesis or given post-synthesis as an input to the RPP.

This document is organized as follows. First, an overview of the ROBUS-2 conceptual design is presented. Then, the abstract timing models used in the timing analyses are introduced. The timing analyses appear after that. Next, the RPP design requirements are enumerated. This is followed by a description of the RPP. The formulas for the behavioral and structural parameters are given after that, followed by a list of the variables that must be specified for a particular implementation. The appendix has the detailed specification for one of the components of the RPP.

2. Overview of the ROBUS-2 communication system

This section presents a brief description of ROBUS-2, including the structure and operation of the system. ROBUS-2 is intended for implementations with a relatively small number of PEs, say, no more than seven. A more detailed description can be found in [Torres 05].

2.1.1. System structure

Figure 2.1 shows the ROBUS topology. The bus has an active-star architecture with the **Bus Interface Units** (BIUs) serving as the bus access ports and the **Redundancy Management Units** (RMUs) providing connectivity as network hubs. The network between BIUs and RMUs forms a complete bipartite graph in which each node is directly connected to every node of the opposite kind. Only the links shown are available for communication. All the communication links are bidirectional.

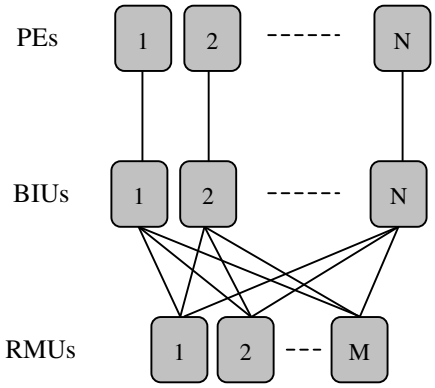


Figure 2.1: ROBUS topology

Figure 2.2 depicts the basic structural components of a ROBUS node. This decomposition applies to BIUs and RMUs. The **Communication Module** handles all the point-to-point communication. The links between BIUs and RMUs can be either one-to-one or one-to-many links, as long as broadcast communication is supported. The nature of the links between BIUs and PEs depends on how they are physically related. If each BIU and its corresponding PE are in physically separate fault-containment regions (FCRs) (see the **Redundancy management** section), then they are interconnected by a one-to-one data communication link. If each PE-BIU pair share an FCR, then some other means of local data exchange can be used.

The **Computation Module**, also known as the ROBUS Protocol Processor (RPP), handles all the ROBUS-specific functions including mode transition logic, low-level protocols, error detection, diagnosis, reconfiguration, and distributed coordination. The main difference between BIUs and RMUs is the functionality of their RPPs.

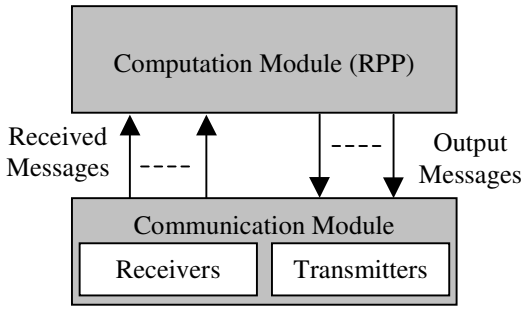


Figure 2.2: Generic node structure for BIUs and RMUs

2.1.2. Distributed coordination

Each ROBUS node is driven by an independent, free-running **physical oscillator**. These oscillators are characterized by a known bound on their drift rates with respect to real time. Each node also has a logical-time clock, referred to as the **local-time clock**, which keeps track of the passage of time as indicated by the physical oscillator. Given an initial precision of synchronization for the local times at any two nodes, the precision can worsen over time at a rate determined by the drift rates of the physical oscillators.

There are two main categories of ROBUS protocols: synchronization and synchronous. The **synchronization protocols** use event-triggered communication and event-processing operations to generate high-precision distributed events that are used to synchronize the local-time clocks. The **synchronous protocols** process information using time-triggered communication and operations. To achieve proper coordinated action in the execution of the synchronous protocols, the local-time clocks of the participating nodes must be synchronized within some known bounded precision.

The ROBUS has two synchronization states: synchronized and unsynchronized. In the **synchronized** state, the precision of synchronization is determined by an internal distributed reference event generated by a clock synchronization protocol. The precision of this event allows the nodes to achieve very tight local-time synchronization. The bus is in the **unsynchronized** state when it transitions to the startup and restart processes. The precision of synchronization in this state is mainly determined by events not directly controlled by the bus. It is assumed that the synchronization precision in this mode has a known bound that can be large relative to the precision in the synchronized state. The bus transitions from the unsynchronized state to the synchronized state after the execution of a synchronization protocol. Because the local times can drift apart, a synchronization protocol must be re-executed at regular intervals to ensure that the local times are kept synchronized. The rate of re-synchronization is constrained by physical parameters of the design (e.g., oscillator drift rates) as well as precision and accuracy goals. The fault-tolerance attribute of the synchronization protocols enables the bus to achieve and maintain synchronization even in the presence of failed nodes.

The execution of synchronous protocols is driven by the local time and a **time-indexed operation schedule**. The low-level distributed protocols specify the node activities by defining the operations, the operation sequencing, the message flow patterns, and the executing nodes for each operation. The timing of the operations is determined using a model of **distributed synchronous composition**. This execution scheme and the high synchronization precision in the synchronized state make the steady-state behavior of the ROBUS highly deterministic as it precisely specifies the timing of all the internal communication between BIUs and RMUs, as well as the communication with the PEs.

2.1.3. Redundancy management

The purpose of redundancy management is to increase the probability of continued service delivery through effective utilization of available resources. The ROBUS is designed to manage its redundant BIU and RMU components independently from the PEs.

Fault containment refers to the isolation of physical faults to prevent their propagation throughout the system. This is achieved by establishing **fault containment regions** (FCR) that ensure a sufficiently high degree of independence with respect to physical faults. Ideally, the FCRs have separate power supplies and are physically and electrically isolated from each other. Communication between FCRs is through carefully specified interfaces that ensure a sufficiently high degree of fault containment. In the ROBUS, each RMU node is contained in its own FCR, and each BIU can be located by itself in a separate FCR, or it can share an FCR with its corresponding PE.

Each BIU and RMU node is an **observer** of every node on the bus. An observed node is referred to as a **defendant**. The diagnostic system of the ROBUS is a distributed system divided into two layers. In the **local layer**, the nodes monitor the communication and independently diagnose each individual node and the bus as a whole. In the **collective layer**, the nodes exchange local diagnostic information to augment their local assessments. Every ROBUS node performs the diagnostic functions of error detection, node assessment, and bus assessment.

Error detection is the foundation of the diagnostic system. The **communication checks** monitor the communication links between the nodes. The **in-line checks** are applied to the received messages and are based on expected timing and content characteristics. The **cross-lane checks** also detect errors in received messages by comparing them against the result of dynamic voting. The **protocol checks** inspect received messages and voting results with respect to expected properties for intermediate and final protocol results. The **self-checks** are performed by a node to monitor its own operation. **PE-error checks** inspect the messages received by the BIUs from their attached PEs. These error checks generate the syndromes from which diagnostic decisions are made.

The diagnostic system assesses each node to determine its suitability to participate in the delivery of services to the PEs. A **trustworthy** node can be relied upon to deliver the expected services. **Untrustworthy** nodes do not behave as expected. A defendant is locally **accused** by an observer when the observer determines that the defendant is untrustworthy, but it is uncertain whether other observers have reached the same conclusion. A defendant is collectively **convicted** when the observers agree that a sufficient number of them consider the defendant untrustworthy. An observer forms a full diagnostic assessment of a defendant based on the local and collective diagnoses.

In the context of the ROBUS, a **clique** is a group of BIUs and RMUs working together in a coordinated way to deliver services to the PEs. A clique is considered trustworthy if its services are in accordance with the specification. The diagnosis of the bus consists of determining if a trustworthy clique is in operation.

The BIU and RMU nodes use the diagnostic assessments to determine the clique membership. A clique is reconfigured by adding or removing nodes from its membership. The purpose of **reconfiguration** is to enhance the ability of a clique to establish and preserve proper service delivery in the presence of untrustworthy nodes. A clique member is allowed to participate in the delivery of services to the PEs and is referred to as a **trusted** node. A node searching for or trying to become part of a clique is called a **recovering** node.

The FCRs ensure that the only error propagation path between nodes is through their interfaces. **Error containment** for the interfaces between BIUs and RMUs is realized by placing barriers at both ends of each interface. The BIUs and RMUs disable their outputs upon detection of a local failure or a bus failure (i.e., fail stop). At the receiving end of the interfaces, the nodes use input-error detection in the form of communication and in-line checks, and dynamic voting to mask undetected errors from trusted sources. The sources whose inputs are considered in a vote are called the **eligible voters**.

2.1.4. Operational modes

Figure 2.3 shows the major mode transitions for BIU and RMU nodes. After a power-on enable, a node goes to the **Self-Test** major mode to perform a local initialization and test its circuitry. The node will remain in this mode indefinitely unless it successfully passes the test. A recovering node enters the **Clique Detection** mode to determine whether there is a clique operating in the Clique Preservation mode. If a clique is found, the recovering node transitions to the **Clique Join** mode, where it demonstrates to the clique members that it is suitable for admission. If a clique is not found, the recovering node transitions to the **Clique Initialization** mode to form a new clique. In the **Clique Preservation** mode, a clique delivers services to the PEs according to the service schedule. At any time, if a node detects a local failure or a bus failure, it transitions back to the Self-Test mode to reinitialize its operation and find other nodes suitable for providing communication services to the PEs.

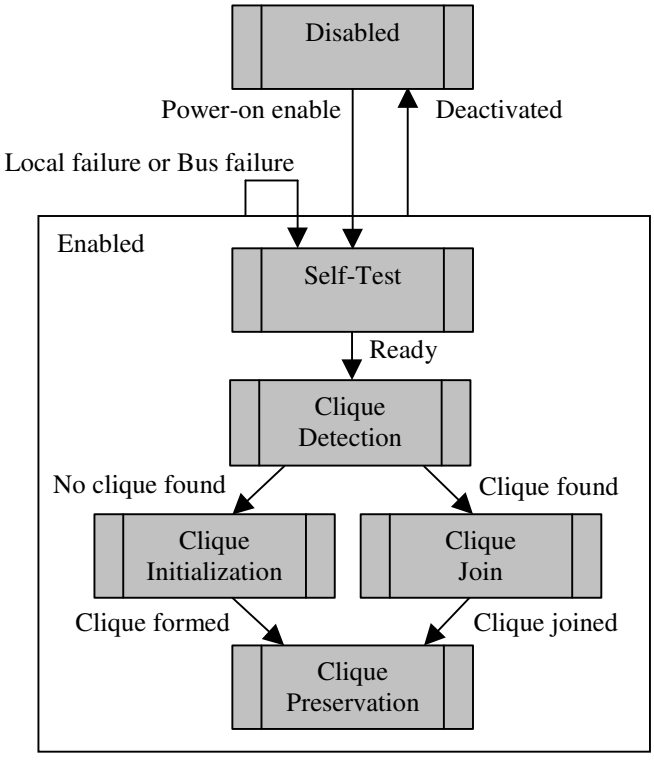


Figure 2.3: Major operational mode transitions for ROBUS nodes

2.1.4.1. Clique Preservation

Figure 2.4 illustrates the minor mode transitions for the Clique Preservation major mode. In the **Schedule Update** mode, a schedule-download protocol is executed to allow the PEs to reprogram the bus

according to their communication needs. During **PE Communication**, first the PE messages are broadcast according to the communication schedule, and then the BIUs and RMUs exchange accumulated accusations against nodes of the opposite kind, which serves to enhance the diagnosis and reconfiguration capabilities of the bus. This is followed by a re-synchronization of the local time in the **Synchronization Preservation** mode and then a reassessment of the clique membership in the **Collective Diagnosis** mode.

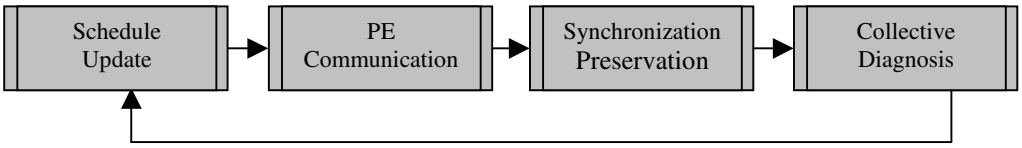


Figure 2.4: Minor mode transitions for Clique Preservation mode

2.1.4.2. Self-Test

Upon entering the Self-Test mode, a node disables its output and performs a reset of its circuitry. This mode serves as a checkpoint in which the nodes are required to exercise and assess the status of their circuitry before attempting to join other nodes on the bus. This mode also provides a safe state to which the ROBUS nodes can go after detecting a failure and before attempting to re-engage.

2.1.4.3. Clique Detection

Figure 2.5 shows the minor mode transitions in the Clique Detection major mode. In **Local Diagnosis Acquisition**, a node uses asynchronous local observations to make a first assessment of the likely members of a clique. In **Synchronization Acquisition**, the node attempts to synchronize to the clique. In **Collective Diagnosis Acquisition**, the node captures the health assessment for each node as determined by the clique during the execution of the distributed diagnosis protocol. If at any time during the Clique Detection mode the node determines that a valid clique is not present, it will exit this mode and attempt to form a new clique. Otherwise, it will assume that a clique exists and will try to join it.

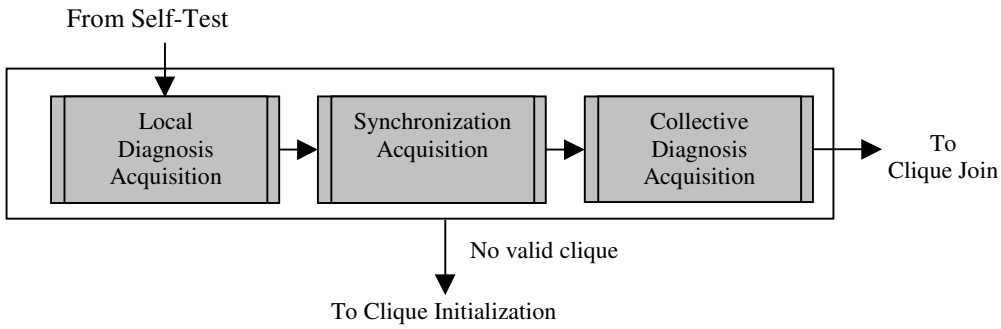


Figure 2.5: Minor modes transitions for Clique Detection mode

2.1.4.4. Clique Join

When a node enters the Clique Join mode, its state is in agreement with the state of the clique. In this mode, the node runs for two diagnostic cycles, essentially trying to demonstrate that it can be trusted.

The existing members of the clique will integrate the node as soon as they confirm that the admission rules have been satisfied.

2.1.4.5. Clique Initialization

Figure 2.6 shows the minor mode transitions for the Clique Initialization major mode. A node transitions to the Clique Initialization major mode to form a new clique. The first minor mode is **Initial Diagnosis**, in which a node identifies other nodes that are also attempting to form a new clique. This is followed by the **Initial Synchronization** and **Collective Diagnosis** minor modes, where the nodes are synchronized and a consistent clique membership is established.

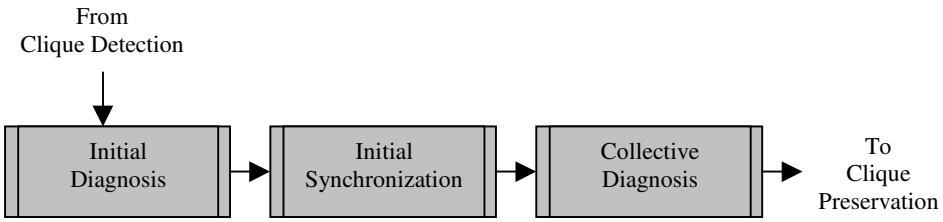


Figure 2.6: Minor modes transitions for Clique Initialization mode

2.1.5. ROBUS Messages

The BIUs, RMUs, and PEs communicate using **ROBUS Messages** (RM). Figure 2.7 illustrates the message format, which consists of a Tag field followed by a Payload field. The Tag field has one of two values: SPECIAL or DATA. The format and content of the Payload field depends on the value of the Tag field and the context in which the message is used.

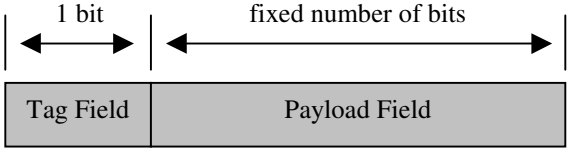


Figure 2.7: ROBUS Message format

A SPECIAL message carries a bit pattern corresponding to one of several labels, including the INIT and ECHO labels used by the synchronization protocols.

DATA messages carry data with a context-specific format. For Collective Diagnosis, the Payload field of each message carries diagnostic information in the form of a Boolean vector. For Schedule Update, the messages carry the number of messages scheduled for a particular PE. For the PE Broadcast protocol in the PE Communication mode, the messages carry information from the PEs with an application-dependent format. The exchange of accusations after the completion of the scheduled PE broadcasts uses the payload format for diagnostic messages.

2.1.6. Point-to-point communication

Figure 2.8 illustrates the composition of the one-way communication path between a BIU and an RMU. The link transmitter and receiver are part of the Communication Module at the source and receiver nodes, respectively. The received messages are stored by the Computation Module at the receiver node until the proper time for processing. This arrangement supports all the modes of point-to-point communication between BIUs and RMUs: synchronous, fixed-delay, and asynchronous-monitoring.

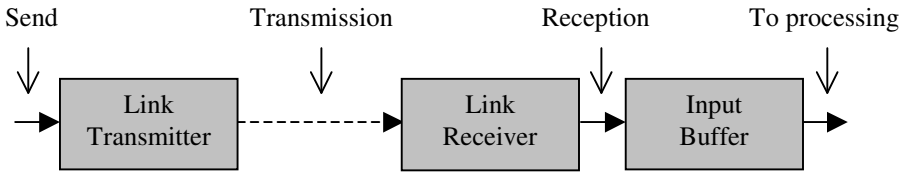


Figure 2.8: Generic point-to-point communication path

Synchronous communication is used with the synchronous protocols. This is a time-triggered communication scheme. Synchronous communication requires that the local-time clocks of the source and receiver nodes be synchronized within some known bounded precision. Figure 2.9 illustrates the main variables. A time T_{REF} is chosen as a reference to coordinate the send and receive actions. A message sent at time T_{SND} with a nominal reception delay R_{PP} is expected to be received at local time $T_{RCV,E}$. Taking into consideration the local-time skew between the source and the receiver, and the uncertainty in the reception delay, a message from a trustworthy source should be received within an expected-reception interval of duration W_{RCV} centered at $T_{RCV,E}$. A message that arrives outside this interval is considered invalid. A valid received message is buffered until the scheduled time for processing T_{PROC}. This buffering corresponds to a deskewing function in which the received message is synchronized to the local time at the receiving node.

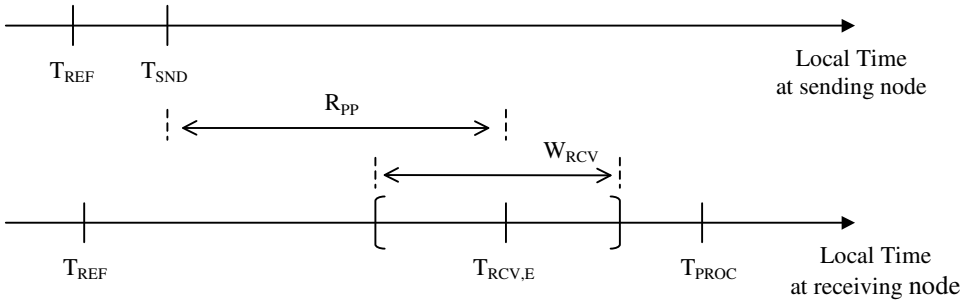


Figure 2.9: Timing for synchronous communication

Fixed-delay communication is used with the synchronization protocols. For this communication mode the information of interest is in the timing of the messages. A transmission is triggered by events at the source. At the receiving end, the message is buffered for a predetermined time duration before processing it. This communication mode is used only with the INIT and ECHO messages of the synchronization protocols. For the Synchronization Preservation protocol executed in the Clique Preservation and Clique Join modes, it is possible to use local events at the nodes to determine a nominal expected time of reception and an expected-reception interval for the synchronization messages.

Asynchronous-monitoring communication is used by a recovering node to observe the activity on the bus before its local time is synchronized. This communication mode does not require coordination

between a source node (which could be a synchronized clique member) and the receiving recovering node. Asynchronous monitoring is made possible by the fact that the BIUs and RMUs broadcast their transmissions to all the nodes of the opposite kind and the point-to-point communication path allows the recovering node to receive messages regardless of its state. The recovering node uses the buffer as a fixed-delay queue, and the messages are processed in the order in which they are received. The delay is not necessarily the same as for the fixed-delay communication mode used with the synchronization protocols.

The PE Interface at the BIUs is designed using a first-in first-out (FIFO) buffer abstraction for input and output. For input, it is assumed that each PE message is available when expected or there is a corresponding error indication. For output, it is always assumed that the message can be output at its scheduled time without having to confirm that the PE is ready to receive it.

2.1.7. Communication patterns

This section presents the patterns of communication for the ROBUS-2 protocols. The description is limited to the sequences of computation processes and message transmissions. The actual computation operations performed by the nodes are not described here. [Torres 05] has a complete description of the protocols.

2.1.7.1. Collective Diagnosis

Figure 2.10 shows the communication pattern for the Collective Diagnosis protocol. The circles represent the processing done by the nodes. Each arrow represents a single-message broadcast transmission from the sources to the receivers. BIUs and RMUs use synchronous communication. The results of the protocol are the convictions against BIUs and RMUs, which are stored locally by BIUs and RMUs, and forwarded to the PEs.

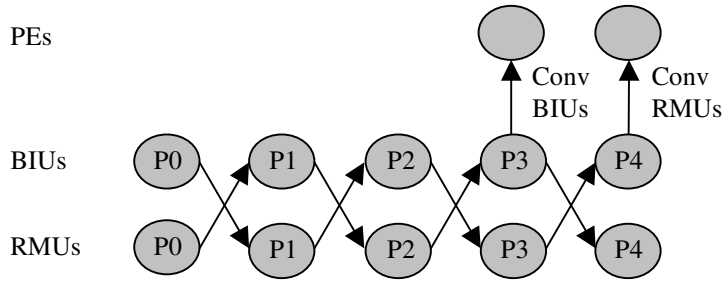


Figure 2.10: Message flow graph for the Collective Diagnosis protocol

2.1.7.2. Schedule Update

Let N denote the number of BIUs, which is assumed to equal the number of PEs connected to the bus. The PEs are identified according to the statically assigned identification numbers which uniquely identify each ROBUS port. The desired schedule is delivered by each PE to its BIU in the form of N consecutive messages with the positions in the sequence corresponding to the identification numbers of the PEs and the payload fields of the messages indicating the desired number of messages to be broadcast. The

interval between the send time of one message and the send time of the next (known as the **data introduction interval** or DII) [De Micheli 94] is constant. The submitted schedule messages are processed using an agreement protocol, called the Schedule Update protocol, to ensure that all the BIU and RMU clique members and the PEs agree on the result for each PE. Figure 2.11 shows the message flow graph for the Schedule Update protocol. BIUs and RMUs use synchronous communication. The protocol is applied independently N times, with each iteration processing the messages delivered by the PEs that indicate the number of messages to be broadcast by a particular PE. The result of each protocol iteration is sent back to the PEs in process P2. After all the messages have been processed, the ROBUS nodes individually assess the resulting schedule. If the new schedule is valid, it is accepted. Otherwise, a default schedule known to all the PE, BIU, and RMU nodes is used.

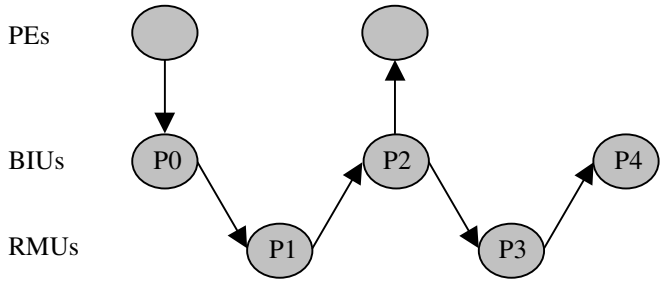


Figure 2.11: Message flow graph for the Schedule Update protocol

2.1.7.3. PE Broadcast

In PE Communication mode, the ROBUS grants bus access to individual PEs according to the communication schedule. An interactive consistency protocol, called the PE Broadcast protocol, is used for each scheduled message to ensure that the PEs receive consistent messages. The bus access pattern is a time-indexed, as-soon-as-possible (ASAP) round-robin sequence. Figure 2.12 provides an example of the access pattern. The PEs access the bus in ascending order according to the port identification numbers. The first scheduled message is sent at some predetermined time. The DII for PE messages is constant. After all the scheduled messages for one PE have been sent, the messages for the next PE are broadcast maintaining the proper DII between messages. If a PE is not scheduled to send messages, then the messages for the next scheduled PE are sent. This continues until all the scheduled messages have been sent.

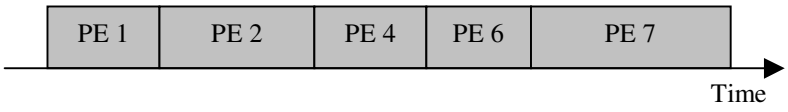


Figure 2.12: Example of an access pattern during the PE Broadcast service

Figure 2.13 shows the message flow pattern for the PE Broadcast protocol. This protocol is used to process each scheduled PE message. Only the scheduled PE and its corresponding BIU are required to send messages. The protocol uses synchronous communication. The result of the protocol is relayed to the PEs.

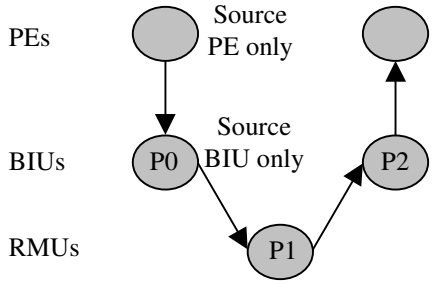


Figure 2.13: Message flow graph for the PE Broadcast protocol

2.1.7.4. Accusation Exchange

The broadcast of PE messages in the PE Communication mode is followed by an exchange of accumulated accusations against nodes of the opposite kind using the Accusation Exchange protocol. Figure 2.14 shows the message flow pattern. This protocol uses synchronous communication.

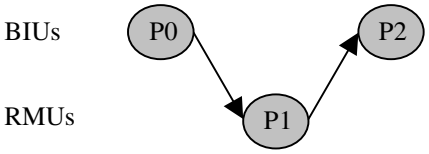


Figure 2.14: Message flow graph for the Accusation Exchange protocol

2.1.7.5. Synchronization Preservation

Figure 2.15 shows the communication pattern for the Synchronization Preservation protocol. The message to be sent by each process is indicated in the figure. Fixed-delay communication is used for all the messages. For this protocol, it is possible to use the time of transmission of a message in one process to determine an expected time of reception in another process. For example, the RMUs can estimate the expected time of reception for process P3 based on the time of transmission in process P1. [Torres 05] describes in detail how to do this.

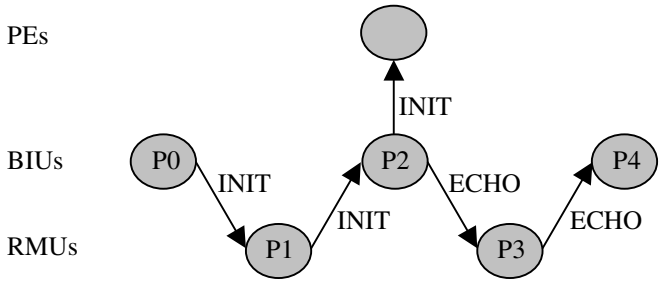


Figure 2.15: Message flow graph for the Synchronization Preservation protocol

2.1.7.6. Local Diagnosis Acquisition

In Local Diagnosis Acquisition, a recovering node monitors the activity on the bus to determine a trusted set of opposite-kind nodes operating in the Clique Preservation mode. The recovering node uses the asynchronous-monitoring communication mode to make its observations. A node in this mode does not transmit messages. Figure 2.16 illustrates the nominal message flow for a recovering RMU in a 3x3 system (i.e., 3 BIUs and 3 RMUs). The PEs and their links are not shown. The solid arrows represent the message flow from the BIUs to the recovering node. The dashed lines represent the bidirectional communication between the BIUs and the other RMUs. The message flow is similar for a recovering BIU.

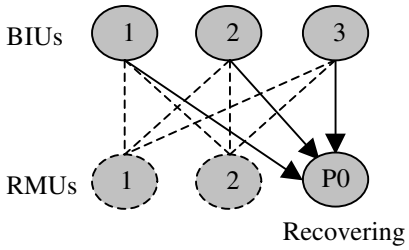


Figure 2.16: Message flow in a 3x3 system with a recovering RMU executing Local Diagnosis Acquisition

2.1.7.7. Synchronization Acquisition

The Synchronization Acquisition mode has two protocols: Frame Synchronization and Synchronization Capture. The Frame Synchronization protocol monitors the activity on the bus essentially to find the gap between consecutive executions of the Synchronization Preservation protocol. Figure 2.16 also applies to this protocol. The recovering node can use fixed-delay or asynchronous-monitoring communication.

The Synchronization Capture protocol is activated by the completion of Frame Synchronization protocol. A recovering node executing Synchronization Capture receives messages only from the opposite kind nodes and does not generate any messages. In that sense, Figure 2.16 also applies to this protocol. Synchronization Capture uses fixed-delay communication applied to ECHO messages from the Synchronization Preservation protocol. Figure 2.17 shows the message flow graph for the Synchronization Preservation protocol expanded to include the Synchronization Capture processes. As shown, a recovering RMU or BIU processes the ECHO messages broadcast between processes P2 and P3, or between P3 and P4, respectively. In addition, a recovering BIU executing process P4C also sends an ECHO message to its attached BIU.

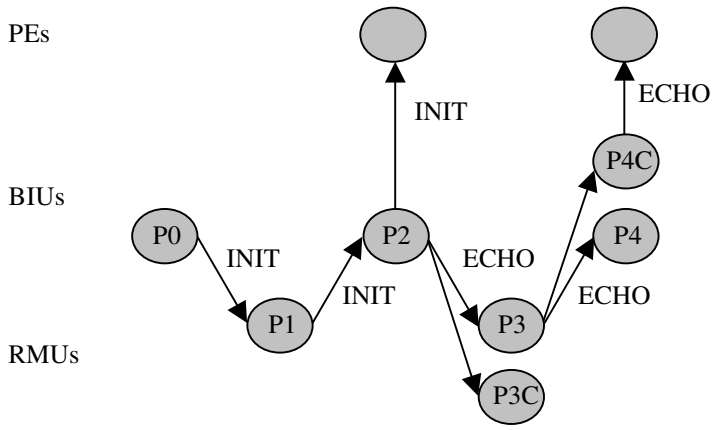


Figure 2.17: Message flow graph for Synchronization Preservation with the Synchronization Capture processes

2.1.7.8. Collective Diagnosis Acquisition

A recovering node executing the Collective Diagnosis Acquisition protocol is assumed to be synchronized to a clique. The processing in this protocol is essentially the same as for the Collective Diagnosis protocol and is executed by a recovering node in parallel with the execution of the Collective Diagnosis protocol by the clique members. Figure 2.10 shows the message flow pattern for the Collective Diagnosis protocol. A recovering node receives messages using the synchronous communication model. In terms of the communication pattern, the main difference between Collective Diagnosis Acquisition and Collective Diagnosis is that a recovering node does not broadcast messages during the Collective Diagnosis Acquisition protocol. In that sense Figure 2.16 also applies to this protocol.

2.1.7.9. Initial Diagnosis

In the Initial Diagnosis minor mode, the nodes execute a synchronous protocol to determine an initial trusted set taking advantage of the known bound on the synchronization precision when operating in the unsynchronized state. Figure 2.18 illustrates the message flow pattern. This protocol uses synchronous communication.

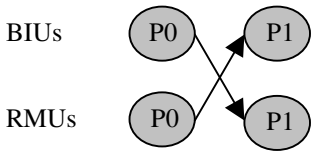


Figure 2.18: Message flow graph for the Initial Diagnosis protocol

2.1.7.10. Initial Synchronization

The Initial Synchronization protocol is similar to the Synchronization Preservation protocol. The differences in processing are the result of the possibly large bound on the relative local-time skew at the beginning of the protocol execution. Figure 2.19 shows the message flow pattern. For this protocol, the BIUs send an ECHO message to the PEs from process P4, instead of an INIT message from process P2. This protocol uses fixed-delay communication.

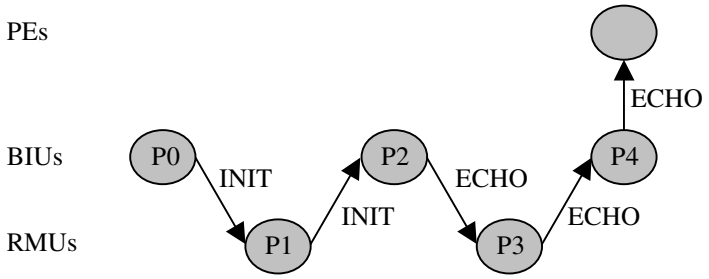


Figure 2.19: Message flow graph for the Initial Synchronization protocol

3. Timing models

This section presents the abstract models used in the genreric timing analysis of ROBUS. RPP design constraints and application-specific timing specifications must be applied to the models in order to determine the behavior for a particular ROBUS implementation. The results of the analysis can then be used to compute the corresponding values for the structural and behavioral parameters of the RPP.

The following terms are used throughout. A **process** is a set of interrelated activities or operations performed to produce a prescribed output or product. An **action** is something that is done to produce a result or effect. An **operation** is an action. An operation is composed of one or more operations, also called **sub-operations**. A **primitive operation** is the simplest type of operation for which further decomposition is not of interest. An **event** is a change in condition or state. An event takes place at an instant of time and does not have duration. **Time** is a nonspatial continuum that is measured in terms of events that succeed one another from past through present to future. A **clock** is a device for measuring time, and it contains a **physical oscillation mechanism** that periodically generates an event called a **tick**. The duration between two consecutive ticks is called the **granularity** of the clock. A **trigger** is an event that causes the activation of some action. An **operation driver** is an entity that causes an operation to advance by triggering the lower level actions. An **event-triggered operation** is an operation triggered by an event. A **time-triggered operation** is an operation triggered by a time event. An **event-driven operation** is an operation in which the sub-operations are triggered by events. A **time-driven operation** is an operation in which the sub-operations are triggered by time events.

The timing models are expressed in terms of parameterized behavior of individual nodes. The behavior of groups of nodes can be expressed in terms of bounds on the values of these behavioral parameters, as well as bounds on the relative local-time skew.

3.1. Physical oscillators and local-time clocks

Each ROBUS node is driven by an independent, free-running physical oscillator (i.e., the phase is not controlled in any way by ROBUS) and a logical-time clock (i.e., a counter) that keeps track of the passage of time as indicated by the oscillator. An oscillator **tick**, also called a **clock tick** or a **system tick**, is the basic unit of time on the bus. In what follows, the term **oscillator clock** denotes the signal generated by the physical oscillator, the **local-time clock** refers to the logical-time clock, and the **local time** refers to the state of the logical-time clock. The process of synchronizing a signal or a message to the transitions of the oscillator clock is referred to as **signal synchronization**. The process of synchronizing a message to the local time is referred to as **deskewing**.

Let f_0 denote the nominal frequency of an oscillator measured in ticks per second or Hertz (Hz). The duration of a tick for an ideal oscillator is exactly $1/f_0$ seconds. An ideal oscillator is said to have zero drift rate with respect to real-time since the oscillator perfectly marks the passage of time with a tick duration of exactly $1/f_0$ seconds. Real oscillators are characterized by non-zero drift rates with respect to real-time. It is assumed that the drift rate of the physical oscillators is bounded by a small constant ρ_0, which is positive, real valued, and unitless. The bound on the drift of the physical oscillators is interpreted as follows. Let $c_x(T)$ denote the earliest real time at which local-time clock x reaches value T. $c_x(T)$ has units of **nominal ticks** (1 nominal tick = $1/f_0$ seconds). T_1 and T_2 denote arbitrary values of the local-time clock with the constraint $T_2 \geq T_1$. Then:

$$(T_2 - T_1)/(1 + \rho_0) \le c_x(T_2) - c_x(T_1) \le (1 + \rho_0)(T_2 - T_1) \quad (3.1)$$

Let τ_0 denote the nominal tick duration measured in seconds (i.e., 1 nominal tick = τ_0 seconds = $1/f_0$ seconds). τ_x denotes the actual tick duration of local-time clock x. The bound on the drift rate of clock x can be expressed as follows:

$$\tau_0/(1 + \rho_0) \le \tau_x \le (1 + \rho_0)\tau_0 \quad (3.2)$$

In other words, the fastest clock has a tick duration of at least $1/(1 + \rho_0)$ nominal ticks, and the slowest clock has a tick duration of at most $(1 + \rho_0)$ nominal ticks. This model accounts for the drift with respect to real time of the physical oscillators and the local-time clocks. The point-to-point communication model accounts for jitter on the output signal of the physical oscillators.

3.2. Drift rate of a clock signal generated by a frequency divider

We want to determine the drift rate for clock signal y generated by a factor-of-k frequency divider driven by clock signal x. k is a positive-integer constant. f_x denotes the actual frequency of clock signal x, τ_x denotes the actual tick duration of clock signal x, $\tau_{x,0}$ denotes the nominal tick duration of clock signal x, ρ_x^* denotes the actual signed drift rate for clock signal x, and ρ_x denotes the absolute value of the actual drift rate (i.e., $\rho_x = |\rho_x^*|$). Here, $\rho_x^* > 0$ corresponds to a slow clock, and $\rho_x^* < 0$ corresponds to a fast clock.

The relation between the actual drift rate and the actual tick duration of x is as follows:

If $\rho_x^* \ge 0$, then $\tau_x = \tau_{x,0}(1 + \rho_x)$. (3.3)

If $\rho_x^* < 0$, then $\tau_x = \tau_{x,0}/(1 + \rho_x)$. (3.4)

Let τ_y denote the actual tick duration and absolute value of the drift rate for clock signal y, and let ρ_y denote the actual absolute value of the drift rate for clock signal y. The actual frequency of the derived clock signal y is $f_y = f_x/k$ and the tick duration is $\tau_y = k\tau_x$. The nominal tick duration of y is $\tau_{y,0} = k\tau_{x,0}$. Using (3.3) and (3.4), we show that a frequency divider preserves the drift rate of the driving clock signal as follows:

If $\rho_x^* \ge 0$, then $\tau_x = \tau_y/k = \tau_{x,0}(1 + \rho_x)$. Thus: $\tau_y = k\tau_{x,0}(1 + \rho_x) = \tau_{y,0}(1 + \rho_x)$. (3.5)

If $\rho_x^* < 0$, then $\tau_x = \tau_y/k = \tau_{x,0}/(1 + \rho_x)$. Thus: $\tau_y = k\tau_{x,0}/(1 + \rho_x) = \tau_{y,0}/(1 + \rho_x)$ (3.6)

Thus, clock signal y has the same drift rate as clock signal x (i.e., $\rho_y = \rho_x$). In addition, the bound on the drift rate for clock x applies to the derived clock y.

From this point on, we use ρ_0 as the bound on the drift rate for all clock signals.

3.3. Paths to Self-Test mode

A ROBUS node enters the Self-Test mode either for startup or for restart. The transition to this mode is triggered by events. For startup, the triggering event is the power-on enable. For restart, the triggering

event is the detection of a failure.

3.3.1. Startup

For startup, all the nodes that will form a clique are enabled within a time interval of known bounded duration. This results in a corresponding bounded relative skew of the local time at power-on enable. It is assumed that the nodes begin the Self-Test mode as soon as they are enabled.

δ_{POE} denotes the actual duration of the time interval within which the nodes are enabled, and $\delta_{POE}|_{max}$ denotes the upper bound on δ_{POE}. δ_{POE} is measured in seconds. π_{POE} denotes the upper bound on the relative time skew at power-on enable. π_{POE} is measured in nominal clock ticks. $\delta_{POE}|_{max}$ and π_{POE} are related as follows:

$$\pi_{POE} = \delta_{POE}|_{max} / \tau_0 \tag{3.7}$$

3.3.2. Restart

A node returns to the Self-Test mode to attempt a restart after a failure condition is detected. This condition can be caused by random hardware failures or by environmental conditions. Random hardware failures generally affect only one node at a time, while environmental conditions have the potential to directly affect all the system nodes simultaneously. After detecting a failure, a node is assumed to go through a process of local recovery to regain control of its local operation, probably involving a local reset, before entering the Self-Test mode. Figure 3.1 shows the sequence of events in the timing model. δ_{FCP} denotes the actual duration of a fault-causing phenomenon measured in nominal clock ticks. $\delta_{FCP}|_{max}$ denotes the maximum value of δ_{FCP}. Δ_{FD} denotes the actual duration of the failure-detection delay measured in local clock ticks. δ_{FD} denotes the actual duration of the failure-detection delay measured in nominal clock ticks. $\delta_{FD}|_{max}$ denotes the maximum value of δ_{FD}. Δ_{LR} denotes the duration of the local recovery process measured in units of local clock ticks.

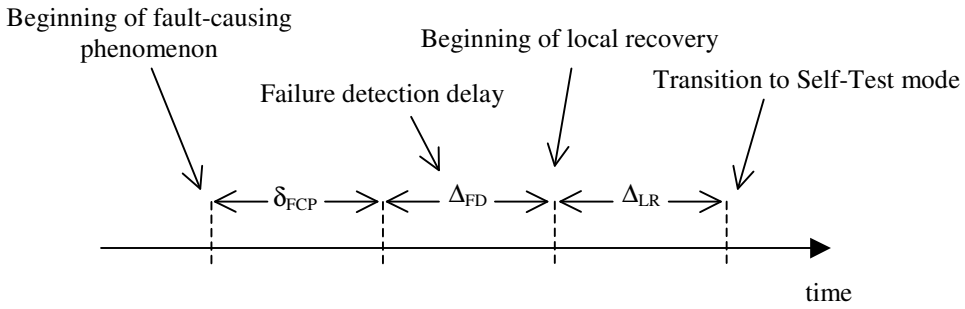


Figure 3.1: Path to Self-Test for a restart

3.4. Self-Test mode

Once in Self-Test mode, the execution can be driven by time or by local events associated with this mode. In general, the duration of the Self-Test mode must satisfy the timing requirements for the transient-fault scenarios expected to be handled successfully by the implementation.

Δ_{STM} denotes the duration of the Self-Test mode for a ROBUS node measured in units of local clock ticks.

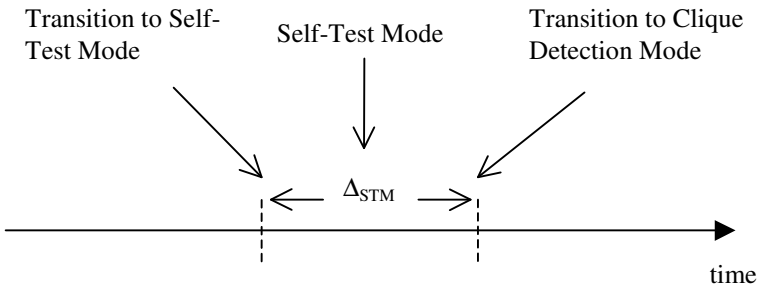


Figure 3.2: Self-Test mode

3.5. Clique Detection mode

The transition to Clique Detection mode is triggered by the end of the Self-Test mode. The Clique Detection mode is composed of three minor modes: Local Diagnosis Acquisition (a.k.a., Preliminary Diagnosis), Synchronization Acquisition, and Collective Diagnosis Acquisition. Local Diagnosis Acquisition is composed of two consecutive observation intervals, each of duration at least as large as a re-synchronization interval. Synchronization Acquisition is composed of the Frame Synchronization protocol and the Synchronization Capture protocol. Collective Diagnosis Acquisition has the same timing characteristics as the Collective Diagnosis protocol in Preservation mode. Figure 3.3 illustrates the elements of the model for this mode.

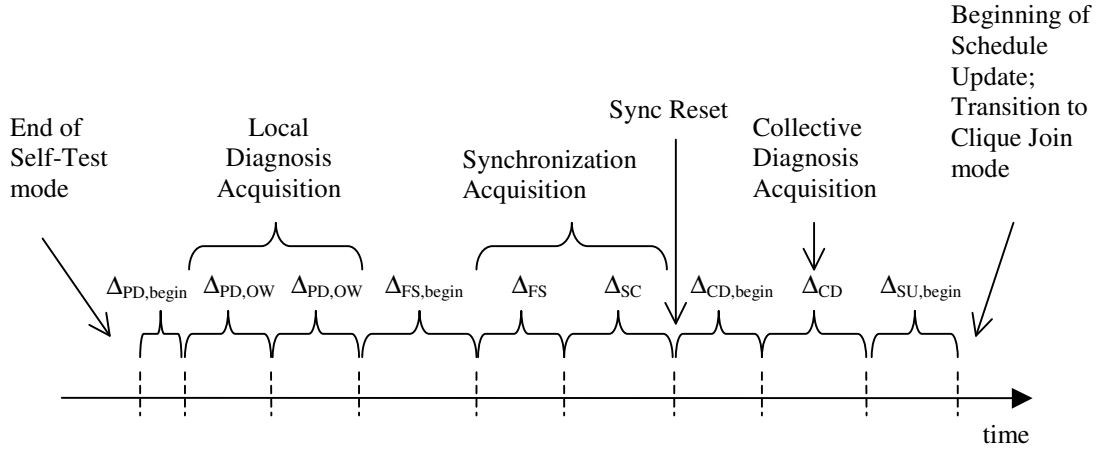


Figure 3.3: Clique Detection mode

$\Delta_{PD,begin}$ denotes the delay from the time a node exits the Self-Test mode until the beginning of the first observation interval during Local Diagnosis Acquisition, measured in local-clock ticks. $\Delta_{PD,OW}$ denotes the duration of the observation intervals (or "windows"), measured in local-clock ticks.

We assume that the Synchronization Capture protocol begins immediately after the Frame Synchronization protocol is complete. $\Delta_{FS,begin}$ denotes the delay from the end of the second observation

window during Local Diagnosis Acquisition to the beginning of the Frame Synchronization protocol during Synchronization Acquisition, measured in local clock ticks. Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in local clock ticks. δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in nominal clock ticks. Δ_{SC} denotes the actual duration of the execution of the Synchronization Capture protocols measured in local clock ticks. δ_{SC} denotes the actual duration of the execution of the Synchronization Capture protocols measured in nominal clock ticks.

Synchronization Acquisition ends with a synchronization reset and the setting of the local clock to 0. From this point on, the local time is synchronized to the clique in Preservation mode. The time delay to begin the Collective Diagnosis Acquisition protocol in Clique Detection mode is the same as the time delay to begin the Collective Diagnosis protocol in Preservation mode. $\Delta_{CD,begin}$ denotes the time from the synchronization reset to the beginning of the Collective Diagnosis protocol, measured in local clock ticks. Δ_{CD} denotes the time to complete the execution of the Collective Diagnosis protocol in local clock ticks.

The transition to the Clique Join mode occurs at the beginning of the execution of the Schedule Update protocol. Before that point, a detected failure attributable to the absence of a clique results in a transition to the Initialization mode. $\Delta_{SU,begin}$ denotes the time from the end of the Collective Diagnosis protocol to the beginning of the Schedule Update protocol, measured in local clock ticks. Δ_{CDM} denotes the actual duration of the Clique Detection mode for a ROBUS node, measured in units of local clock ticks. δ_{CDM} denotes the actual duration of the Clique Detection mode for a ROBUS node, measured in units of nominal clock ticks

It is assumed that a ROBUS node can detect the absence of a valid clique at any time after entering the Local Diagnosis Acquisition windows. After detecting this condition, a node clears its state and transitions to the Clique Initialization mode. $\Delta_{CDM\text{-}CIM}$ denotes the delay to transition to the Initialization mode after detecting the absence of a valid clique, measured in units of local clock ticks.

3.6. Initialization mode

The Initialization mode is composed of the Initial Diagnosis, Initial Synchronization, and Collective Diagnosis protocols. Figure 3.4 illustrates the model for this mode.

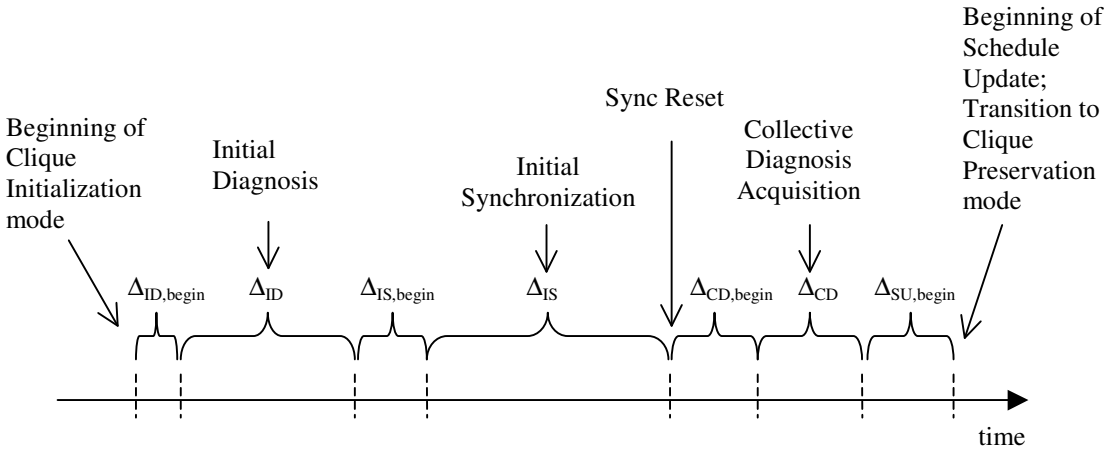


Figure 3.4: Clique Initialization mode

$\Delta_{ID,begin}$ denotes the delay from the time a node enters the Clique Initialization mode until the time it begins execution of Initial Diagnosis, measured in units of local clock ticks. Δ_{ID} denotes the actual duration of the Initial Diagnosis protocol, measured in units of local clock ticks. δ_{ID} denotes the actual duration of the Initial Diagnosis protocol, measured in units of nominal clock ticks. $\Delta_{IS,begin}$ denotes the delay from the time a node completes the Initial Diagnosis protocol until the time it begins execution of Initial Synchronization; measured in units of local clock ticks. Δ_{IS} denotes the actual duration of the Initial Synchronization protocol, measured in units of local clock ticks. δ_{IS} denotes the actual duration of the Initial Synchronization protocol, measured in units of nominal clock ticks.

Initial Synchronization ends with a synchronization reset and the setting of the local clock to 0. From this point on the local time is synchronized to the newly formed clique. The time to begin the Collective Diagnosis protocol in the Clique Initialization mode is the same as the time to begin the Collective Diagnosis protocol in the Preservation mode. The parameters for this execution are presented above in the section covering the Clique Detection mode.

The transition to the Clique Preservation mode occurs at the beginning of execution of the Schedule Update protocol.

3.7. Synchronized time-triggered operation

Once a node has synchronized to a clique, and for as long as it remains synchronized, the execution of protocols is triggered exclusively by the local time. The Synchronization Preservation protocol is driven by events generated during its execution. The other protocols, known as the synchronous protocols, are driven by time. The synchronous protocols are executed in the Collective Diagnosis, Schedule Update, and PE Communication minor modes. Figure 3.5 shows the relevant variables.

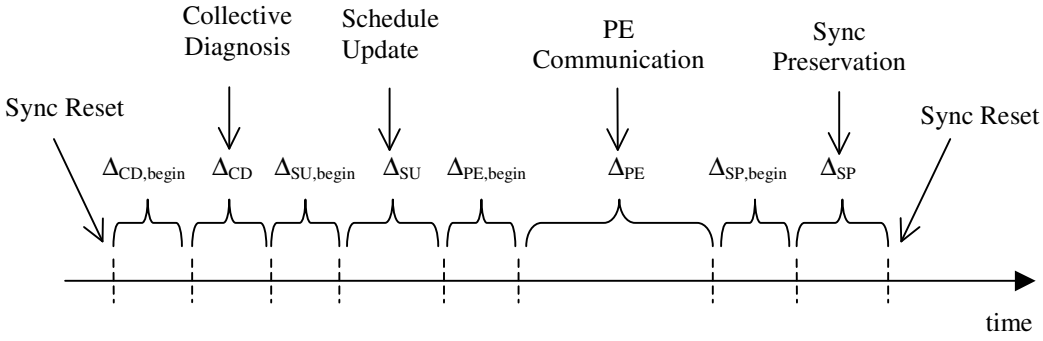


Figure 3.5: Synchronous Operation

Δ_{SU} denotes the actual duration of the Schedule Update protocol measured in local clock ticks. $\Delta_{PE,begin}$ denotes the actual time delay from the end of the Schedule Update protocol to the beginning of PE Communication protocol, measured in local clock ticks. Δ_{PE} denotes the actual duration of the PE Communication mode, measured in local clock ticks. $\Delta_{SP,begin}$ denotes the actual time delay from the end of the PE Communication protocol to the beginning of the Synchronization Preservation protocol, measured in local clock ticks. Δ_{SP} denotes the actual duration of the Synchronization Preservation protocol, measured in local clock ticks.

3.8. Communication Module

The Communication Module of each ROBUS node is composed of send and receive sub-modules. The send sub-module consists of one or more separate transmitters and supports broadcast transmissions. The receive sub-module consists of a separate receiver for each node of the opposite kind.

For ROBUS-2, communication, processing, and diagnosis are based on single-message transactions. The size of a Communication Module transaction is a ROBUS message. The communication of each ROBUS message can be decomposed into multiple physical transmissions. However, the Communication Module should not combine multiple ROBUS messages into a single communication unless the timing of such transaction preserves the current assumptions and specified behavior of the Computation Module.

The transmitters and receivers are expected to be generic components supporting event-triggered communication. For the transmitters, this means that the reading of a new message and the beginning of its transmission process is triggered by a send signal at the transmitter's input interface. Similarly, the receivers should be able to receive new messages whenever they arrive. During normal operation, the only expected communication timing constraint at the input interface of the transmitter is the data introduction interval. Startup and restart timing constraints for the transmitters and receivers should be taken into consideration in the design of the startup and restart behavior for the ROBUS nodes.

The **message delivery delay** is the time elapsed from the instant a transmitter receives a send request to the instant the message is presented at the receiver's output interface in the communication model. It is assumed that the output of the receiver is not edge synchronous with respect to the clock signal at the receiving node (see Section 4).

The **message reception delay** is equal to the message delivery delay plus the additional time delay to synchronize the received message to the local clock signal at the receiving node.

Symbol $d_{PP,l}$ denotes the minimum point-to-point message delivery delay, measured in nominal clock ticks. $d_{PP,h}$ denotes the maximum point-to-point message delivery delay, measured in nominal clock ticks. v_{PP} denotes the delivery precision (i.e., uncertainty in the point-to-point delivery delay), measured in nominal clock ticks. $r_{PP,l}$ denotes the minimum point-to-point message reception delay, measured in nominal clock ticks. $r_{PP,h}$ denotes the maximum point-to-point message reception delay, measured in nominal clock ticks. e_{PP} denotes the reception precision (i.e., uncertainty in the point-to-point reception delay), measured in nominal clock ticks. Λ_{Comm} denotes the minimum data introduction interval for a send port of the Communication Module, measured in local clock ticks.

3.9. Computation Module

The Computation Module is modeled in terms of two components: the Computation Process and the Send Process. The Computation Process handles the processing of received messages according to the requirements of the protocol being executed. The Send Process handles the timing and formatting requirements for the output messages.

3.9.1. Computation Process

The Computation Process has two stages. The **Reception Stage** provides timing adjustment and inline error detection for the received messages. The C**omputation Stage** performs the reduction of the set of received messages into a single result value, and it also handles the cross-lane error detection.

3.9.1.1. Reception Stage

Depending on the protocol being executed, the Reception Stage provides either fixed or variable delays for the input messages. For the synchronization protocols, the Reception Stage has a fixed input-output delay for each input path. This behavior preserves the relative positions of the received synchronization messages, which allows the Accept function in the Computation Stage to properly read the relative skews of the timing events. In this mode, the delay of the Reception Stage depends mainly on the uncertainty in the time of reception and on the time required to process the messages for error detection and diagnosis. For the synchronous protocols, the Reception Stage performs a deskewing function in order to synchronize the received messages to the local time. The deskewing is applied independently to each input path, and the received messages are forwarded to the Computation Stage at a predetermined time. The Reception Stage also inspects the messages for error detection.

For Local Diagnosis Acquisition and Frame Synchronization, the Reception Stage has a fixed input-output delay for each input path.

3.9.1.2. Computation Stage

The timing of the Computation Stage depends on the protocol being executed. For the synchronization protocols, the Accept function produces the output event with a predetermined delay with respect to the time it receives the input event to be selected. For the events not selected, the Accept function appears to have a variable input-output delay. For the synchronous protocols, the Reception Stage forwards all the messages at the same time and the Computation Stage produces its output a fixed delay later. In this case, the input-output delay of the Computation Stage is the same for all the inputs.

The Computation Stage has a fixed input-output delay for Local Diagnosis Acquisition. A worst-case delay is used for Frame Synchronization.

3.9.1.3. Timing model for synchronization protocols

For the synchronization protocols, the Computation Process has a fixed delay from the time when the event to be selected is received to the time when the Accept output is asserted. We combine the Reception Stage and Computation Stage delays into a single parameter. Symbol A denotes the Computation Process delay for synchronization protocols, measured in local clock ticks.

3.9.1.4. Timing model for the synchronous protocols

For the synchronous protocols, the Reception Stage and Computation Stage operations are time triggered. The Reception Stage is allocated a time interval to receive, deskew, and diagnose the input messages. There is a predetermined time, referred to as the expected time of reception, that is defined as

the nominal time at which a message should arrive at the Receive Stage. The messages from good sources should arrive at the Reception Stage within a predetermined time interval (or window) centered at the expected time of reception. This interval is called the **deskewing window**. For expected messages arriving during the deskewing window, the deskewing function forwards the received messages at a predetermined time after the end of the window. The timing of the Computation Process for synchronous protocols is modeled by two delays: the time from the expected time of reception to the closing of the deskewing window, and the time from the closing of the deskewing window to the output of the Computation Stage.

W_{Deskew} denotes the size of the deskewing window measured in local clock ticks. $W_{Deskew,pre}$ denotes the pre-expectation window (i.e., the size of the deskewing window before the expected time of reception measured in local clock ticks). $W_{Deskew,post}$ denotes the post-expectation window (i.e., the size of the deskewing window after the expected time of reception, measured in local clock ticks. C denotes the processing delay from the closing of the deskewing window to the output of the Computation Stage, measured in local clock ticks.

3.9.2. Send Process

The Send Process handles the transmission of messages. Once the process has been triggered either by an event or by time, the internal operation of the process is driven by the time since the trigger.

3.9.2.1. Timing model for the synchronization protocols

For the synchronization protocols, the operation of the Send Process for the first protocol transmission (i.e., the INIT from the BIUs) is triggered by the local time, while the remaining transmissions are triggered by the Accept output events. Symbol B denotes the send delay with respect to the reference event for synchronization protocols; measured in local clock ticks

3.9.2.2. Timing model for the synchronous protocols

For the synchronous protocols, the operation of the Send Process is triggered exclusively by time. Symbol S denotes the send delay with respect to a reference time, measured in local clock ticks

3.9.3. Constraint on the data introduction interval for the Computation Module

The throughput potential of the Computation Module is characterized by its minimum data introduction interval. This parameter characterizes the input rate constraints of the Computation Process and the Send Process. If the individual processes have different constraints, the larger one is taken as the constraint for the Computation Module. Λ_{Comp} denotes the minimum data introduction interval for the Computation Module; measured in local clock ticks.

3.10. PE Interface

The BIUs interact with the PEs using a first-in-first-out (FIFO) interface abstraction for input and output. It is assumed that the BIUs can access this interface for read and write without the need to

directly coordinate their actions with the PEs. For PE-to-BIU transfers, the PEs are responsible for making their data available to the BIUs at the input FIFO at or before the time at which it will be read. For BIU-to-PE transfers, the BIUs simply write the data to the output FIFO as soon as it is ready. The PEs are responsible for ensuring that no data is lost due to a buffer overflow.

The interaction between PEs and BIUs can be coordinated indirectly by proper selection of the ROBUS timing parameters.

4. Point-to-point communication

This section examines the point-to-point communication between ROBUS nodes. The Communication Module of each ROBUS node is composed of transmit and receive sub-modules. The transmit sub-module consists of one or more separate transmitters to support broadcast transmissions. The receive sub-module consists of a separate receiver for each node of the opposite kind. The transmitters and receivers are expected to be generic components supporting event-triggered communication. The granularity of a Communication Module transaction should be a ROBUS Message, since the communication, processing, and diagnosis performed by the ROBUS protocols are based on single-message transactions. For the transmitters, the reading of a new message and the beginning of its transmission process is triggered by a send signal at the transmitter's input interface. Similarly, the receivers should be able to receive new messages whenever they arrive. The only expected communication throughput constraint at the input interface of the transmitters is the minimum data introduction interval (DII), which is the minimum number of clock ticks between consecutive requests to send messages.

The communication system must be able to support the fixed-delay and synchronous communication models. For some receiver designs, the output signals from the receiver are not synchronized to the circuitry-driving signal generated by a local physical oscillator. Therefore, the Computation Module must synchronize each received message with respect to the local oscillator before proceeding with further processing. For the synchronous communication model, the processing of received messages is triggered by the local-time clock. Therefore, a node must be able to buffer received messages until it is time to process them. The timing design of the system must be able to handle the uncertainty in the time of transmission, the transmission delay, and the signal synchronization delay.

In addition, this version of the ROBUS is intended to demonstrate that the bus can achieve a PE-message throughput that approaches the available bandwidth at the physical links. For most transmissions, it is possible to compute a local-time interval during which a receiver should expect to receive the message. For low link data rates, the reception intervals for individual nodes do not overlap and each message can be processed before the next one arrives. For high data rates, the reception intervals of consecutive messages overlap and the processing must be pipelined in order to match the link throughput. This section examines some critical aspects of pipelined communication.

4.1. Synchronization of asynchronous signals

Single-phase edge-triggered flip-flops used as building blocks in traditional synchronous sequential digital circuits have a simple nominal timing behavior: If the signal at the data input is stable within a specified window around the oscillator clock's triggering edge, then the input value will propagate to the output of the flip-flop and stabilize within some guaranteed time. The propagation delay of the flip-flops is the time elapsed from the triggering edge of the oscillator clock until the output is stable. The window around the oscillator clock's triggering edge is characterized by the setup and hold time of the flip-flop. The **setup time** is the minimum time that the input signal must remain stable before the triggering edge of the oscillator clock in order for the output of the flip-flop to meet the nominal propagation delay. The **hold time** is the minimum time that the input signal must remain stable after the triggering edge of the oscillator clock in order for the output of the flip-flop to meet the nominal propagation delay.

The **domain** of an oscillator clock includes all the digital circuitry driven by that signal. A signal is

said to be **synchronous** with respect to a particular oscillator clock if the timing of the signal meets the input setup and hold time constraints of the flip-flops driven by the oscillator clock. A signal that does not meet these constraints is called **asynchronous** with respect to the given oscillator clock. Since the oscillator clocks in the fault containment regions of the ROBUS are independent and the timing of their transitions is not coordinated in any way, any signal crossing from one FCR to another is considered asynchronous when it arrives at the receiving FCR.

Asynchronous signals must be synchronized to the oscillator clock before they can be processed. Various mechanisms can be used to achieve this synchronization. Ultimately, however, consideration must be given to the problem of violations of the setup and hold times of flip-flops reading the signal. A flip-flop sampling an input that is not stable within the setup and hold window can enter a metastable condition in which the output does not settle to a valid logic state within the nominal propagation delay. If not handled properly, this can result in the generation of more asynchronous signals and the propagation of errors throughout the receiving FCR.

The mean time between failure (MTBF) for a flip-flop reading an asynchronous input is (see [XAPP077]):

$$MTBF = e^{-C_2 * t_{MET}} / (2 * C_1 * f_D * f_C) \quad (4.1)$$

where t_{MET} denotes the time available for the metastability to resolve itself (i.e., time allowed by downstream circuitry before reading the output of the flip-flop), f_D denotes the input signal frequency ($2*f_D$ is the input signal event rate), f_C denotes the oscillator clock frequency, C_1 denotes the metastability aperture of the flip-flop (which is related to the width of the window during which an input can cause a metastability condition), and C2 denotes the resolution rate (which is related to the speed with which the metastable condition will be resolved). Constants C1 and C2 are functions of the process technology and flip-flop design. For current technology, the variables of the MTBF can be selected such that the probability of metastability failures is extremely small.

In the following analysis it is assumed that the problem of metastability is properly handled by the implementation of the ROBUS. Unless explicitly stated otherwise, it is assumed that the nodes have ideal signal synchronizers, each consisting of a single flip-flop driven by the oscillator clock. These ideal flip-flops have no metastable states and zero propagation delay. The timing behavior is as follows.

If the input changes before the triggering-edge of the oscillator clock, this latest input value will propagate to the output as soon as the triggering-edge of the oscillator clock arrives. If the input changes at exactly the same time as the triggering-edge of the oscillator clock, the input value will not affect the output until the next triggering-edge of the oscillator clock (assuming that the input remains constant).

4.2. Single-message communication

The communication of a message from a source node to a receiver node is modeled as a four step process: (1) **Send**: The Computation Module of the source node signals the transmitter(s) in the Communication Module that a message is ready for transmission; (2) **Transmission**: The transmitter reads the message and transmits the corresponding signals over the transmission medium; (3) **Delivery**: The link receiver gets the message from the transmission medium and signals the arrival to the signal

synchronizer; (4) **Reception**: The synchronizer signals the arrival of a new message to the Computation Module. Figure 4.1 illustrates the point-to-point communication path. CLK_{Rx} denotes the oscillator clock at the receiving node. The **message delivery delay** is the time elapsed from the instant a transmitter receives a send request until the message is presented at the output interface of the receiver. The **message reception delay** is equal to the message delivery delay plus the additional time delay to synchronize the received message to the oscillator clock at the receiving node.

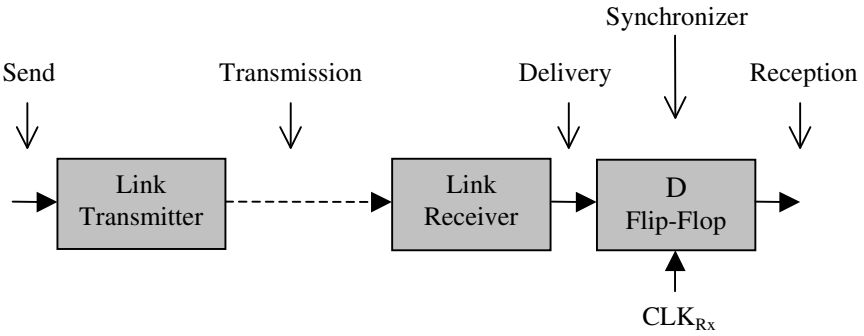


Figure 4.1: Conceptual point-to-point communication path

Symbols $d_{PP,l}$ and $d_{PP,h}$ denote the minimum and maximum point-to-point message delivery delays, respectively, measured in units of nominal clock ticks. v_{PP} denotes the delivery precision (i.e., the uncertainty in the point-to-point delivery delay) measured in units of nominal clock ticks. $r_{PP,l}$ and $r_{PP,h}$ denote the minimum and maximum point-to-point message reception delays, respectively, measured in nominal clock ticks. e_{PP} denotes the reception precision (i.e., the uncertainty in the point-to-point reception delay) measured in nominal clock ticks.

4.2.1. Reception delay

Let T_0 denote the local time at which the source sends the message, and let t_0 denote the corresponding real time. The real-time range of point-to-point message delivery is $[t_0 + d_{PP,l}, t_0 + d_{PP,h}]$. Therefore, the delivery precision is:

$$v_{PP} = d_{PP,h} - d_{PP,l} \tag{4.2}$$

The minimum point-to-point message reception delay happens when the message is sampled by the input synchronizer at exactly the same time it is delivered.

$$r_{PP,l} = d_{PP,l} \tag{4.3}$$

The maximum point-to-point message reception delay happens when the message is sampled by the input synchronizer exactly one tick after it is delivered. The worst case delay occurs when the oscillator clock at the receiving node is slow.

$$r_{PP,h} = d_{PP,h} + (1 + \rho_0) \tag{4.4}$$

The real-time range of reception is $[t_0 + r_{PP,l}, t_0 + r_{PP,h}]$. Therefore, the reception precision is:

$$e_{PP} = r_{PP,h} - r_{PP,l} = [d_{PP,h} + (1 + \rho_0)] - d_{PP,l} = 1 + \rho_0 + v_{PP} \tag{4.5}$$

e_{PP} accounts for time-discretization errors, jitter and drift of the source and recceiver oscillators, as well as differences in point-to-point communication delays due to differences in thie length of communication wires or optical fibers.

Next, we define IMP(x_1, x_2), the Integer Mid-Point value (i.e., Rounded Average), as the integer closest to the mid-point of x_1 and x_2. IMP(x_1, x_2) is computed in two steps:

Step 1: $x = (x_1 + x_2)/2$

Step 2: IMP = round(x), with round(x) = $\lfloor x \rfloor$ if $x < ½$, or $\lceil x \rceil$ if $x \geq ½$

R_{PP} denotes the expected reception delay:

$$R_{PP} = IMP(r_{PP,l}, r_{PP,h}) \tag{4.6}$$

4.2.2. Estimate of the local-time at the source

Let T_{RCV} denote the local time at the receiver when it receives the message. To estimate the local time at the source node, the receiver assumes that the message reception delay is R_{PP} ticks of its oscillator clock. The estimated local time at the source node at the time of reception is:

$$T_{SRC,E} = T_0 + R_{PP} \tag{4.7}$$

The error in the local-time estimate is bounded as follows. T_{RCV} occurs no earlier than $\mu_{PP,l}$ nominal ticks from the actual local time $T_{SRC,E}$ at the source:

$$\mu_{PP,l} = (1 + \rho_0)R_{PP} - r_{PP,l} \tag{4.8}$$

T_{RCV} occurs no later than $\mu_{PP,h}$ nominal ticks from the actual local time $T_{SRC,E}$ at the source:

$$\mu_{PP,h} = r_{PP,h} - R_{PP}/(1 + \rho_0) \tag{4.9}$$

4.2.3. Expected local time of reception

Let $\pi_{PP,SR}$ denote a bound on the relative local-time skew between the source and the receiver nodes. This bound is assumed to hold for the duration of the communication. The expected local time of reception at the receiver is denoted by $T_{RCV,E}$.

$$T_{RCV,E} = T_0 + R_{PP} \tag{4.10}$$

Due to the relative local-time skew and the uncertainty in the message-reception delay, the message will arrive within some local-time interval containing $T_{RCV,E}$. Let $\Delta_{PP,RCV}$ denote the local-time error in T_{RCV}:

$$\Delta_{PP,RCV} = T_{RCV} - T_{RCV,E} \tag{4.11}$$

We want to determine the absolute maximum local-time error in T_{RCV}, denoted by $\Delta_{PP,RCV}|_{abs\text{-}max}$:

$$|T_{RCV} - T_{RCV,E}| \leq \Delta_{PP,RCV}|_{abs\text{-}max} \quad (4.12)$$

The value of $\Delta_{PP,RCV}|_{abs\text{-}max}$ is derived as follows. The bound on the local-time synchronization between the source and the receiver nodes is expressed as:

$$|c_{SRC}(T) - c_{RCV}(T)| \leq \pi_{PP,SR} \quad (4.13)$$

where $c_{SRC}(T)$ and $c_{RCV}(T)$ denote the earliest real times at which the local times at the source and at the receiver, respectively, reach value T. From the previous analysis, it is known that the real-time difference between the time when the source reaches $T_{RCV,E}$ and the time when the message is actually received T_{RCV} is bounded above and below by $\mu_{PP,h}$ and $\mu_{PP,l}$, respectively.

$$c_{SRC}(T_{RCV,E}) - \mu_{PP,l} \leq c_{RCV}(T_{RCV}) \leq c_{SRC}(T_{RCV,E}) + \mu_{PP,h} \quad (4.14)$$

For local time $T_{RCV,E}$, inequality (4.13) can be re-expressed as:

$$c_{SRC}(T_{RCV,E}) - \pi_{PP,SR} \leq c_{RCV}(T_{RCV,E}) \leq c_{SRC}(T_{RCV,E}) + \pi_{PP,SR} \quad (4.15)$$

Combining inequalities (4.14) and (4.15), we get:

$$-\pi_{PP,SR} - \mu_{PP,l} \leq c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E}) \leq \pi_{PP,SR} + \mu_{PP,h} \quad (4.16)$$

So:

$$|c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \leq \max(\pi_{PP,SR} + \mu_{PP,l}\ ,\ \pi_{PP,SR} + \mu_{PP,h}) \quad (4.17)$$

Equivalently:

$$|c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \leq \pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h}) \quad (4.18)$$

Using the constraint that the local clocks are ρ-bounded, the definition of $\Delta_{PP,RCV}$, and the real time duration of $\Delta_{PP,RCV}$ ticks for the fastest allowed clock:

$$|\Delta_{PP,RCV}|/(1 + \rho_0) \leq |c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \quad (4.19)$$

Combining (4.18) and (4.19):

$$|\Delta_{PP,RCV}| \leq (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \quad (4.20)$$

Since $\Delta_{PP,RCV}$ is an integer, we can take the floor in (4.20):

$$|\Delta_{PP,RCV}| \leq \lfloor (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad (4.21)$$

Therefore, the worst-case local-time difference between the actual time of reception T_{RCV} and the expected time of reception $T_{RCV,E}$ is:

$$\Delta_{PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad (4.22)$$

4.3. Coordination for synchronous communication

For the synchronous ROBUS protocols, the scheduling of operations is based on a **distributed synchronous composition** abstract model of the system in which a single oscillator drives a common local-time clock and fixed-delay processes corresponding to the communication and computation operations of the BIUs and RMUs. Communication during time-driven operations is time-triggered. For each transmission, the sources and receivers use a particular local-time value as a distributed reference event to coordinate their actions. Given specific bounds for the reception delay and the relative local-time skew between sources and receivers, it is possible to coordinate the send and receive operations such that the transmitted messages are received within a predetermined local-time range measured at the receivers. The receivers can then apply a deskewing function and forward the received messages for processing at a predetermined local time. By leveraging the previous analysis, it is possible to analyze the source-receiver coordination problem using only global time (i.e., synchronized local time viewed from a global perspective). Figure 4.2 illustrates the relevant timing events. T_{REF} denotes the reference local- time value. T_{SND} is the time at which the message is sent. R_{PP} is the expected reception delay. $T_{RCV,E}$ is the expected time of reception. W_{Deskew} is the size of the deskewing window. $W_{Deskew,pre}$ is the pre-expectation window (i.e., the size of the section of the deskewing window before the expected time of reception). $W_{Deskew,post}$ is the post-expectation window (i.e., the size of the deskewing window after the expected time of reception). $T_{PROC,begin}$ denotes the time for the beginning of message processing.

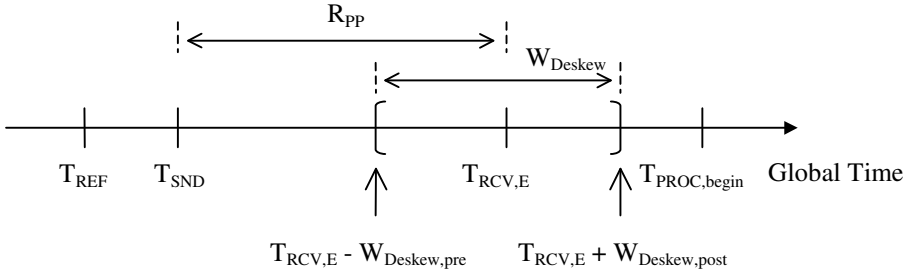


Figure 4.2: Timing events for point-to-point communication

A message from a good source is expected to arrive during the following closed time interval, which includes all triggering edges of the local clock within the expected time range of reception:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max}] \quad (4.23)$$

The deskewing window includes all triggering edges of the clock within the expected time range of reception, a total of $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ edges. The deskewing window is intended to cover the duration of all local clock counts corresponding to the triggering edges of the clock within the expected time range of reception. The local clock counts corresponding to these triggering edges determine a time interval with a duration of $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ ticks. The deskewing window extends for the real-time interval corresponding to the following half-closed local-time interval:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1) \quad (4.24)$$

So:

$$W_{Deskew} = 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (4.25)$$

And:

$$W_{Deskew,pre} = \Delta_{PP,RCV}|_{abs\text{-}max} \quad (4.26)$$

$$W_{Deskew,post} = \Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (4.27)$$

For proper communication, the following constraints must be satisfied:

$$T_{REF} \leq T_{SND} \quad (4.28)$$

$$T_{REF} \leq T_{RCV,E} - W_{Deskew,pre} \quad (4.29)$$

$$T_{RCV,E} = T_{SND} + R_{PP} \quad (4.30)$$

$$T_{RCV,E} + W_{Deskew,post} \leq T_{PROC,begin} \quad (4.31)$$

Relations (4.28) and (4.29) express basic time constraints for the common reference time. Relation (4.30) captures the goal of source-receiver coordination, which is to receive the message at the expected time of reception. Relation (4.31) is only relevant to the composition of operations at the receiving node. Let $\Delta_{REF\text{-}SND}$ denote the delay from T_{REF} to T_{SND} measured in local clock ticks.

$$\Delta_{REF\text{-}SND} = T_{SND} - T_{REF} \quad (4.32)$$

Let $\Delta_{REF\text{-}RCVWND}$ denote the delay from T_{REF} to $T_{RCV,E} - W_{Deskew,pre}$ measured in local clock ticks.

$$\Delta_{REF\text{-}RCVWND} = (T_{RCV,E} - W_{Deskew,pre}) - T_{REF} \quad (4.33)$$

Let $\Delta_{REF\text{-}SND}|_{min}$ and $\Delta_{REF\text{-}RCVWND}|_{min}$ denote the minimum possible values for $\Delta_{REF\text{-}SND}$ and $\Delta_{REF\text{-}RCVWND}$, respectively. Relation (4.30) can be re-expressed as follows:

$$\Delta_{REF\text{-}SND} + R_{PP} = \Delta_{REF\text{-}RCVWND} + W_{Deskew,pre} \quad (4.34)$$

We are interested in finding the values for $\Delta_{REF\text{-}SND}$ and $\Delta_{REF\text{-}RCVWND}$ to achieve the earliest communication satisfying (4.28), (4.29), and (4.30). We consider two cases.

Case 1: $\Delta_{REF\text{-}SND}|_{min} + R_{PP} \geq \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre}$

For this case, the message can be sent as soon as possible, but the window must be delayed to align it with the expected time of reception.

$$\Delta_{REF\text{-}SND} = \Delta_{REF\text{-}SND}|_{min} \quad (4.35)$$

$$\Delta_{REF\text{-}RCVWND} = \Delta_{REF\text{-}SND}|_{min} + R_{PP} - W_{Deskew,pre} \quad (4.36)$$

Case 2: $\Delta_{REF\text{-}SND}|_{min} + R_{PP} < \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre}$

For this case, the window can be opened as soon as possible, but the message must be delayed to achieve proper alignment.

$$\Delta_{REF\text{-}SND} = \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (4.37)$$

$$\Delta_{REF\text{-}RCVWND} = \Delta_{REF\text{-}RCVWND}|_{min} \quad (4.38)$$

4.4. Message streams

Each message in a message stream is processed independently. Let K denote the total number of messages in the stream. i denotes the index for the messages in the stream, with $0 \le i \le K\text{-}1$. $T_{SND,i}$ is the local time at which the source sends the i-th message of the stream. $T_{RCV,E,i}$ is the expected local time of reception for the i-th message of the stream. Λ_{stream} denotes the data introduction interval at the source measured in local clock ticks.

The throughput capacities of the Communication Module and the Computation Module are characterized by their respective minimum data introduction interval [De Micheli 94]. Let Λ_{Comm} and Λ_{Comp} denote the minimum data introduction interval for the Communication Module and the Computation Module, respectively. Λ_{Comm} and Λ_{Comp} are measured in local-clock ticks. For proper processing, Λ_{stream} must be greater than or equal to Λ_{Comm} and Λ_{Comp}.

$$\Lambda_{stream} \ge \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (4.39)$$

4.4.1. Message delivery rate

We would like to compute the number of messages that can be delivered during a particular time interval. We consider intervals during steady state transmission after the leading edge of the stream and before the trailing edge. Because of the drift rate of the clocks, the observed number of delivered messages can vary within a range.

Let W_{RCV} denote the size of the observation window at the receiving node measured in local clock ticks. Q denotes the number of messages delivered during the observation window. λ_{SRC} denotes the data introduction interval measured in nominal clock ticks. w_{RCV} denotes the size of the observation window at the receiving node measured in nominal clock ticks. Let $t_{deliver,i}$ denote the real time at which message i is delivered.

$$t_{deliver,i} = t_{deliver,0} + i\lambda_{SRC} \quad (4.40)$$

Let $t_{obs,l}$ and $t_{obs,h}$ denote the beginning and end times, respectively, for the observation window. The observer records received messages during the closed interval $[t_{obs,l}, t_{obs,h}]$. $t_{obs,l}$ and $t_{obs,h}$ are related by the size of the observation window.

$$t_{obs,h} = t_{obs,l} + w_{RCV} \quad (4.41)$$

The following constraints are applied in order to determine the number of observed messages.

$$t_{deliver,0} < t_{obs,l} \quad (4.42)$$

$$t_{deliver,1} \ge t_{obs,l} \quad (4.43)$$

$$t_{deliver,Q} \le t_{obs,h} \quad (4.44)$$

$$t_{deliver,Q+1} > t_{obs,h} \tag{4.45}$$

For these constraints, a total of Q messages in the index range 1 to Q are delivered within the observation interval. The maximum value of Q is derived as follows. Relation (4.44) can be re-expressed as:

$$t_{deliver,0} + Q\lambda_{SRC} \leq t_{obs,1} + w_{RCV} \tag{4.46}$$

So:

$$Q \leq [(t_{obs,1} - t_{deliver,0}) + w_{RCV}]/\lambda_{SRC} \tag{4.47}$$

The right-hand side reaches its maximum value when $t_{obs,1} - t_{deliver,0} = \lambda_{SRC}$. In that case, $t_{deliver,1} = t_{obs,1}$. So:

$$Q \leq [\lambda_{SRC} + w_{RCV}]/\lambda_{SRC} \tag{4.48}$$

Since Q is an integer, we can take the floor on the right-hand side of the expression. Then:

$$Ql_{max} = \lfloor w_{RCV}/\lambda_{SRC} \rfloor + 1 \tag{4.49}$$

For a fast source clock:

$$\lambda_{SRC,fast} = \Lambda_{stream}/(1 + \rho_0) \tag{4.50}$$

For a slow receiver clock:

$$w_{RCV,slow} = (1 + \rho_0)W_{RCV} \tag{4.51}$$

Therefore, for the maximum value of Q:

$$Ql_{max} = \lfloor (W_{RCV}/\Lambda_{stream})(1 + \rho_0)^2 \rfloor + 1 \tag{4.52}$$

The minimum value of Q is derived as follows. Relation (4.45) can be re-expressed as:

$$t_{deliver,0} + (Q + 1)\lambda_{SRC} > t_{obs,1} + w_{RCV} \tag{4.53}$$

So:

$$Q > [(t_{obs,1} - t_{deliver,0}) + w_{RCV}]/\lambda_{SRC} - 1 \tag{4.54}$$

The right-hand side approaches its minimum value as $t_{obs,1} - t_{deliver,0}$ approaches 0. So:

$$Q > w_{RCV}/\lambda_{SRC} - 1 \tag{4.55}$$

Q is an integer strictly larger than $w_{RCV}/\lambda_{SRC} - 1$. The smallest integer that satisfies this relation is given by:

$$Ql_{min} = \lfloor w_{RCV}/\lambda_{SRC} \rfloor \tag{4.56}$$

For a slow source clock:

$$\lambda_{SRC,slow} = (1 + \rho_0)\Lambda_{stream} \tag{4.57}$$

For a fast receiver clock:

$$w_{RCV,fast} = W_{RCV}/(1 + \rho_0) \tag{4.58}$$

Therefore, for the minimum value of Q:

$$Q|_{min} = \lfloor (W_{RCV}/\Lambda_{stream})/(1 + \rho_0)^2 \rfloor \tag{4.59}$$

4.4.2. Expected local time of reception

The transmission times for the messages are related by the data introduction interval:

$$T_{SND,i} = T_{SND,0} + i\Lambda_{stream} \tag{4.60}$$

At the receiver, the relation among the messages is similar.

$$T_{RCV,E,i} = T_{RCV,E,0} + i\Lambda_{stream} \tag{4.61}$$

Using the analysis for single-message communication:

$$T_{RCV,E,i} = T_{SND,i} + R_{PP} \tag{4.62}$$

Let $T_{RCV,i}$ denote the actual time of reception for the i-th message. From the analysis of single-message communication, $T_{RCV,i}$ and $T_{RCV,E,i}$ are related as follows:

$$|T_{RCV,i} - T_{RCV,E,i}| \leq \Delta_{PP,RCV}|_{abs\text{-}max} \tag{4.63}$$

Re-expressing (4.63):

$$T_{RCV,E,i} - \Delta_{PP,RCV}|_{abs\text{-}max} \leq T_{RCV,i} \leq T_{RCV,E,i} + \Delta_{PP,RCV}|_{abs\text{-}max} \tag{4.64}$$

The stream as a whole should be received within the following local time interval:

$$[T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} , T_{RCV,E,K-1} + \Delta_{PP,RCV}|_{abs\text{-}max}] \tag{4.65}$$

Re-expressing (4.65):

$$[T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} , T_{RCV,E,0} + (K-1)\Lambda_{stream} + \Delta_{PP,RCV}|_{abs\text{-}max}] \tag{4.66}$$

4.4.3. Message reception rate

The Λ_{stream} communication parameter gives the nominal message reception rate for the stream in units of ticks per message. An important consideration for the processing of message streams is the relation between Λ_{stream} and $\Delta_{PP,RCV}|_{abs\text{-}max}$. As presented above, $\Delta_{PP,RCV}|_{abs\text{-}max}$ measures the uncertainty in the time

of reception of each message. In particular, the total uncertainty in the time of reception for a particular message is $2\Delta_{PP,RCV}|_{abs\text{-}max}$ local clock ticks centered around the expected time of reception. A message from a good source can be received at any of the $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ triggering edges of the oscillator clock in the corresponding reception interval. Let Z denote the number of messages from a good source that can be received during a $2\Delta_{PP,RCV}|_{abs\text{-}max}$ interval. Then:

$$Z = \lfloor 2\Delta_{PP,RCV}|_{abs\text{-}max}/\Lambda_{stream} \rfloor + 1 \quad (4.67)$$

4.4.3.1. Non-overlapping reception intervals

If $\Lambda_{stream} > 2\Delta_{PP,RCV}|_{abs\text{-}max}$, the expected reception intervals for consecutive messages do not overlap or even coincide end-to-end (i.e., no shared triggering edges in consecutive expected reception intervals). For this case, Z = 1, which means that the messages of the stream are received as separate communications with no interaction.

4.4.3.2. Overlapping reception intervals

If $\Lambda_{stream} \leq 2\Delta_{PP,RCV}|_{abs\text{-}max}$, the expected reception intervals for consecutive messages overlap or coincide at the ends. For this case, Z > 1, which means that the interaction between the messages must be taken into consideration. This is especially important for the diagnosis of timing errors.

4.4.4. Load size for a message reception buffer

We refer to the number of messages stored in a buffer as the **load** on the buffer. The function of the message receive buffer is to collect the messages received at the Computation Process. For single-message communication, it is expected that the processing of each message will begin at or before the next message is received. The same can occur for a message stream in which the reception intervals for consecutive messages do not overlap. In these cases, the load of the receive buffer is less than or equal to 1. From this point on, we only consider cases in which the processing of individual messages may begin after the reception of subsequent messages in the stream. This includes cases of overlapping and non-overlapping reception intervals.

Let $\Delta_{PROC,begin}$ denote the delay in the beginning of processing of a message with respect to the corresponding expected time of reception. We assume that the interval between the beginning of processing of consecutive messages is the same as the data introduction interval for the message stream, Λ_{stream}. $T_{PROC,i}$ denotes the local time at the beginning of processing for message i.

$$T_{PROC,i} = T_{RCV,E,i} + \Delta_{PROC,begin} \quad (4.68)$$

4.4.4.1. Combined message synchronization and buffering

Figure 4.3 illustrates the interconnection of functions for this case. CLK_{Rx} denotes the oscillator clock at the receiving node. STB_{Rx} denotes the strobe signal indicating that a new message is ready. The Link Receiver transfers the messages to the Receive Buffer as soon as they are ready. The output of the receiver is assumed to be asynchronous with respect to the oscillator clock. The Receive Buffer is an asynchronous FIFO, which means that the push (i.e., write) and pop (i.e., remove) action signals are

synchronous with respect to different clock signals. In effect, in addition to being a buffer, the asynchronous FIFO serves as a signal synchronizer for data crossing from one clock domain to the other. Note that the data is read for computation one tick before it is popped from the receive buffer.

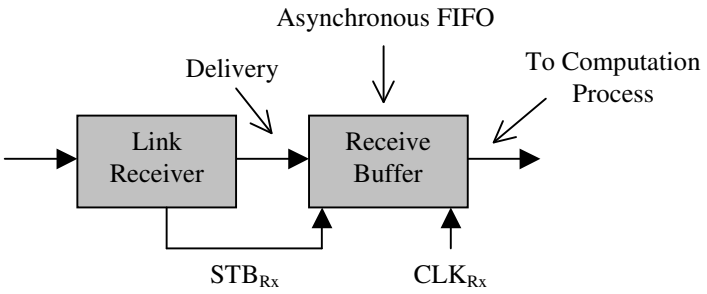


Figure 4.3: Reception using combined message synchronization and buffering

In order to ensure a read-after-write sequence at the Receive Buffer during normal operation, the reading of a particular message by the Computation Process should be triggered after the end of the corresponding reception interval. The following relation must hold in order to satisfy this property.

$$\Delta_{PROC,begin} > \Delta_{PP,RCV}|_{abs\text{-}max} \tag{4.69}$$

Again, note that the pop takes place one tick after the start of processing for each message. $t_{deliver,i}$ denotes the real time at which message i is written to the buffer. A message gets pushed at the same time that it is delivered. With λ_{SRC} denoting the data introduction interval at the source node, $t_{deliver,i}$ is given by the following equation.

$$t_{deliver,i} = t_{deliver,0} + i\lambda_{SRC} \tag{4.70}$$

Let $t_{pop,i}$ denote the real time at which message i is popped from the buffer. The pop times for the Computation Process are given by the following relation, with λ_{RCV} equal to the data introduction interval at the Computation Process.

$$t_{pop,i} = t_{pop,0} + i\lambda_{RCV} \tag{4.71}$$

Let $Q_{deliver}(t)$ denote the number of delivered messages by time t. $Q_{pop}(t)$ denotes the number of popped messages by time t. $Q_{Async\text{-}Buffer}(t)$ denotes the number of messages held by the asynchronous Receive Buffer at time t.

For $Q_{deliver}(t)$:

$$Q_{deliver}(t) = \begin{cases} 0, & \text{for } t < t_{deliver,0} \\ \lfloor [(t - t_{deliver,0})/\lambda_{SRC}] + 1 \rfloor, & \text{for } t_{deliver,0} \le t \le t_{deliver,0} + (K\text{-}1)\lambda_{SRC} \\ K, & \text{for } t > t_{deliver,0} + (K\text{-}1)\lambda_{SRC} \end{cases} \tag{4.72}$$

For $Q_{pop}(t)$:

$$Q_{pop}(t) = \begin{cases} 0, & \text{for } t < t_{pop,0} \\ \lfloor [(t - t_{pop,0})/\lambda_{RCV}] + 1 \rfloor, & \text{for } t_{pop,0} \le t \le t_{pop,0} + (K\text{-}1)\lambda_{RCV} \\ K, & \text{for } t > t_{pop,0} + (K\text{-}1)\lambda_{RCV} \end{cases} \quad (4.73)$$

For $Q_{Asyn\text{-}Buffer}(t)$:

$$Q_{Asyn\text{-}Buffer}(t) = Q_{deliver}(t) - Q_{pop}(t) \quad (4.74)$$

To determine the maximum load for the receive buffer, we consider the case of a fast source clock and a slow receiver clock. Thus:

$$\lambda_{SRC} = \lambda_{SRC,fast} = \Lambda_{stream}/(1 + \rho_0) \quad (4.75)$$

$$\lambda_{RCV} = \lambda_{RCV,slow} = (1 + \rho_0)\Lambda_{stream} \quad (4.76)$$

Assume that the first message is delivered at the earliest possible time. That is:

$$t_{deliver,0} = c_{RCV}(T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max}) \quad (4.77)$$

The time of the first pop action is:

$$\begin{aligned} t_{pop,0} &= c_{RCV}(T_{RCV,E,0} + \Delta_{PROC,begin} + 1) \\ &= t_{deliver,0} + (1 + \rho_0)(\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1) \end{aligned} \quad (4.78)$$

Since the source has a faster clock, the number of buffered messages can increase up to the instant the last message is delivered (i.e., $t = t_{deliver,K\text{-}1} = t_{deliver,0} + (K\text{-}1)\lambda_{SRC}$). Thus, the maximum buffer load is given by $Q_{Async\text{-}Buffer}$ evaluated at $t_{deliver,K\text{-}1}$.

$$\begin{aligned} Q_{Asyn\text{-}Buffer}(t)|_{max} &= Q_{Asyn\text{-}Buffer}(t_{deliver,K\text{-}1}) \\ &= K - \lfloor [(K\text{-}1)\lambda_{SRC,fast} - (1 + \rho_0)(\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1)]/\lambda_{RCV,slow} + 1 \rfloor \\ &= K - \lfloor (K\text{-}1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1)/\Lambda_{stream} + 1 \rfloor \end{aligned} \quad (4.79)$$

4.4.4.2. Separate message synchronization and buffering

Figure 4.4 illustrates the interconnection of functions for this case. The receiver is assumed to hold the message until it is processed by the synchronizer. For this synchronization mechanism, the input rate must be slower than the local clock frequency to ensure at least one triggering edge of the oscillator clock per delivered message. Thus, Λ_{stream} must be must at least 2 (i.e., $\Lambda_{stream} \ge 2$).

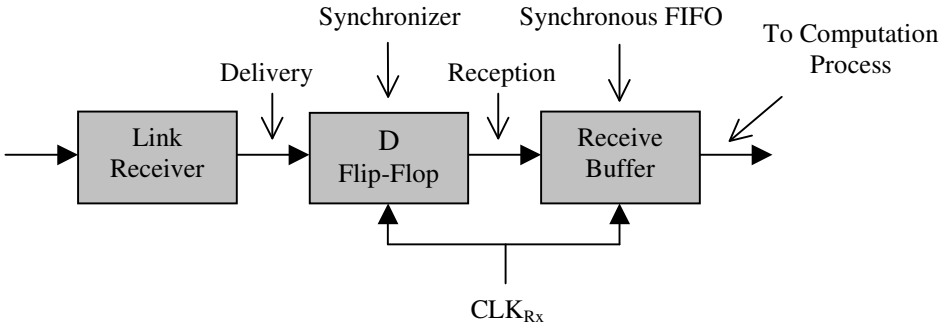


Figure 4.4: Reception using separate message synchronization and buffering

This configuration differs from the one using the asynchronous FIFO in that the synchronization is performed by a dedicated synchronizer. At a minimum, this element introduces a one-tick delay in the transfer of messages from the Link Receiver to the Receive Buffer. The worst-case delay in storing the message in the buffer is two oscillator clock ticks. Therefore, compared to the timing of the circuit with the asynchronous FIFO, the writing of streamed messages to the synchronous FIFO buffer begins at least 1 local tick later and can end up to 2 oscillator clock ticks later.

To determine the maximum load for the receive buffer, we consider the case of a fast source clock and a slow receiver clock. The maximum load is assessed at the earliest time at which the last message of the input stream can be written to the buffer. Let $t_{write,0}$ denote the earliest real time at which a received message is written to the buffer.

$$t_{write,0} = c_{RCV}(T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} + 1) = t_{deliver,0} + (1 + \rho_0) \tag{4.80}$$

The delivery time for the last message is:

$$t_{deliver,K-1} = t_{deliver,0} + (K-1)\lambda_{SRC,fast} \tag{4.81}$$

After delivery, the message must be synchronized and written to the buffer. In the fastest case, the delivered message is immediately read by the synchronizer and presented to the buffer for loading, which will then occur 1 tick later.

$$t_{write,K-1} = t_{deliver,K-1} + (1 + \rho_0) = t_{deliver,0} + (K-1)\lambda_{SRC,fast} + (1 + \rho_0) \tag{4.82}$$

Let $Q_{Sync\text{-}Buffer}(t)$ denote the number of messages held by the synchronous receive buffer at time t. The maximum load is given by:

$$\begin{aligned} Q_{Sync\text{-}Buffer}(t)|_{max} &= Q_{Sync\text{-}Buffer}(t_{write,K-1}) \\ &= K - Q_{pop}(t_{write,K-1}) \\ &= K - \lfloor [(t_{write,K-1} - t_{pop,0})/\lambda_{RCV,slow}] + 1 \rfloor \\ &= K - \lfloor (K-1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin})/\Lambda_{stream} + 1 \rfloor \end{aligned} \tag{4.83}$$

5. Clock synchronization protocols

This section examines the timing aspects for the local-time synchronization scheme. The diagnostic system works in close coordination with the clock synchronization system to determine the status of the bus and to specify the nodes eligible to participate in clock synchronization operations. That aspect of the ROBUS is outside the scope of this section. The analysis presented here uses the fundamental fault-tolerance concepts presented in Appendix A of [Torres 05] and the point-to-point communication concepts presented in Section 4 of this document.

5.1. Clock synchronization system

The allowed range for the tick duration τ_x of an oscillator is determined by the nominal tick duration τ_0 and the bound on the drift rate ρ_0.

$$\tau_0/(1 + \rho_0) \leq \tau_x \leq (1 + \rho_0)\tau_0 \qquad (5.1)$$

That is, an actual oscillator has a tick duration between $1/(1 + \rho_0)$ and $(1 + \rho_0)$ nominal ticks.

The local-time clock of a node is essentially a counter driven by the local physical oscillator. The local time is equal to the state of the counter. Resetting the counter sets the local time to 0. The clock synchronization system enables the nodes to use the local time as a reference for the coordination of distributed operations. A basic requirement for proper distributed coordination is that the relative clock skews remain within known bounds. The **relative skew** between two clocks is the real time elapsed from the instant one clock makes a particular state transition (i.e., the count reaches a particular value) until the other clock makes the same transition. In general, the relative skew between two events is equal to the real time elapsed between the occurrence of the events. Bounded relative skew is achieved by the generation and preservation of approximate real-time agreement on the transitions of the local-time clock. The synchronization protocols deliver high-precision distributed events used as references to reset the local-time clocks. The state of a local-time clock indicates the time elapsed since the last synchronization-reset event. The bound on the relative skew between synchronized clocks is tightest at the time of the synchronization reset. After the reset, the local times can drift apart from each other and from real time at rates determined by the drift rates of the oscillators. The clocks are resynchronized at regular time intervals in order to ensure that the relative skews remain within known bounds.

Figure 5.1 illustrates the conceptual mode transitions for the clock synchronization system. Normally there is a clique executing the Synchronization Preservation (SP) protocol to ensure that their relative local-time skews remain within known bounds. Nodes in this mode are said to be in a **synchronized state**. A goal of every node is to reach and remain in this state. In the context of the synchronization system, nodes operating in a mode other than Synchronization Preservation are referred to as **recovering** nodes. After a power-on enable or the detection of a failure, a node examines the activity on the bus. If a clique is found, the recovering node transitions to Synchronization Acquisition (SA) mode in order to synchronize its local time to the time of the clique. If a clique is not found, the recovering node transitions to the Initial Synchronization (IS) mode. After achieving synchronization, the recovering node transitions to Synchronization Preservation mode.

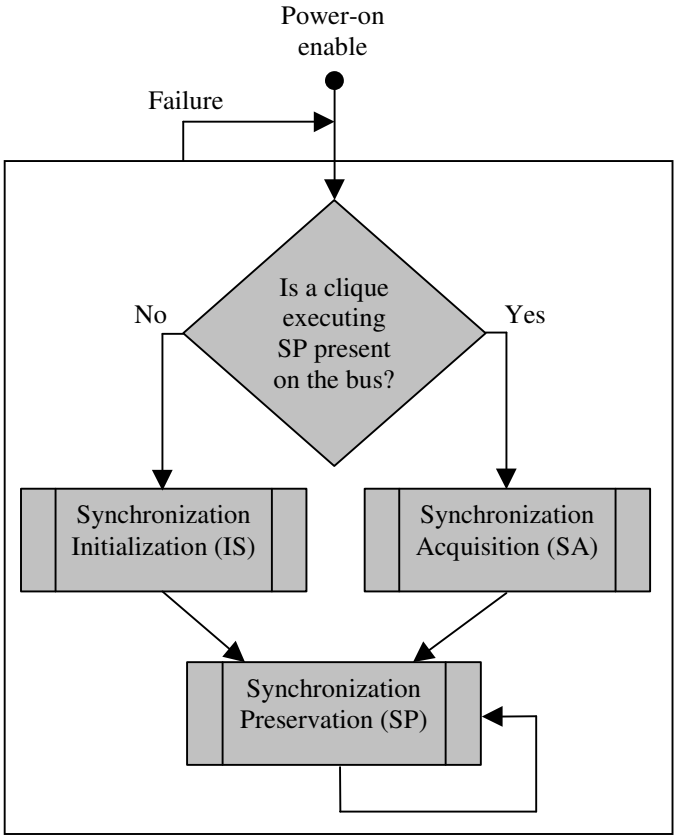


Figure 5.1: Conceptual mode transitions for the clock synchronization system

At the time of entry into the Synchronization Acquisition mode, the recovering node is in an **asynchronous state** in which there is no significant relation between its local time and the local time of the clique. The recovering node uses an Accept function to capture the synchronization events in the agreement propagation phase of the Synchronization Preservation protocol. This requires that the Accept function only receive synchronization messages from the same execution of the protocol. This is accomplished by enabling the Accept function after a frame synchronization step in which the gap between executions of the Synchronization Preservation protocol is found.

In general, a group of nodes enters the Initial Synchronization mode within a time interval of known bounded duration. When a recovering node enters this mode, it expects that there is at least one node of the opposite kind that also makes the transition within the bounded time interval. This interval duration is in effect a bound on the relative local-time skew for the initializing nodes. Before the execution of the synchronization protocol, these nodes are said to be in an **unsynchronized state** since the initial skew bound can be relatively large compared to the skew after the execution of the protocol.

Figure 5.2 illustrates how the mode transitions are related in time. A group of nodes enters Initial Synchronization with a large bound on the relative skew, denoted by π_{IS}. At the end of the protocol execution, the local time is set to 0 with the bound on the relative skew reduced to the level required for normal operation, denoted by π_{SP}. At local time T_{SP}, the Synchronization Preservation protocol is executed to ensure that the skew remains within the expected bound. This cyclic operation continues until a failure occurs or the system is shut down. A recovering node in Synchronization Acquisition

trying to synchronize to the clique executes the Frame Synchronization (FS) protocol followed by the Synchronization Capture (SC) protocol. The duration of the Frame Synchronization protocol execution depends on factors like the total number of nodes of the opposite kind, the number of untrustworthy nodes of the opposite kind active on the bus, the bound on the relative local-time skew of the nodes, and the position of the start of the protocol relative to local time of the clique nodes. Synchronization Capture is enabled immediately after the execution of Frame Synchronization is complete. The relative skew achieved by Synchronization Acquisition is within the bounds of the skew for normal operation.

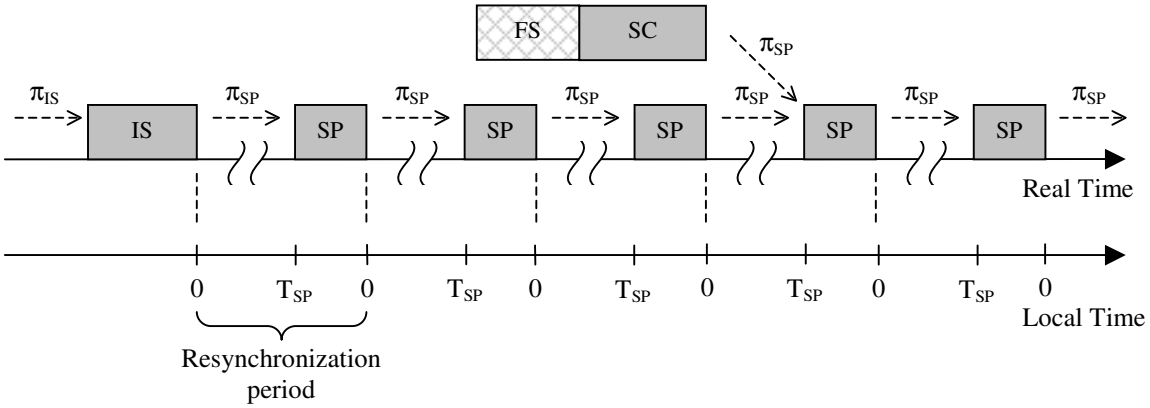


Figure 5.2: Timing of mode transitions for the clock synchronization system

The Initial Synchronization, Synchronization Preservation, and Synchronization Capture protocols are based on the same theory of distributed computation using Accept functions to process timing events. Figure 5.3 illustrates the message flow graph examined in this section. This graph includes all the processes and messages required for the three protocols.

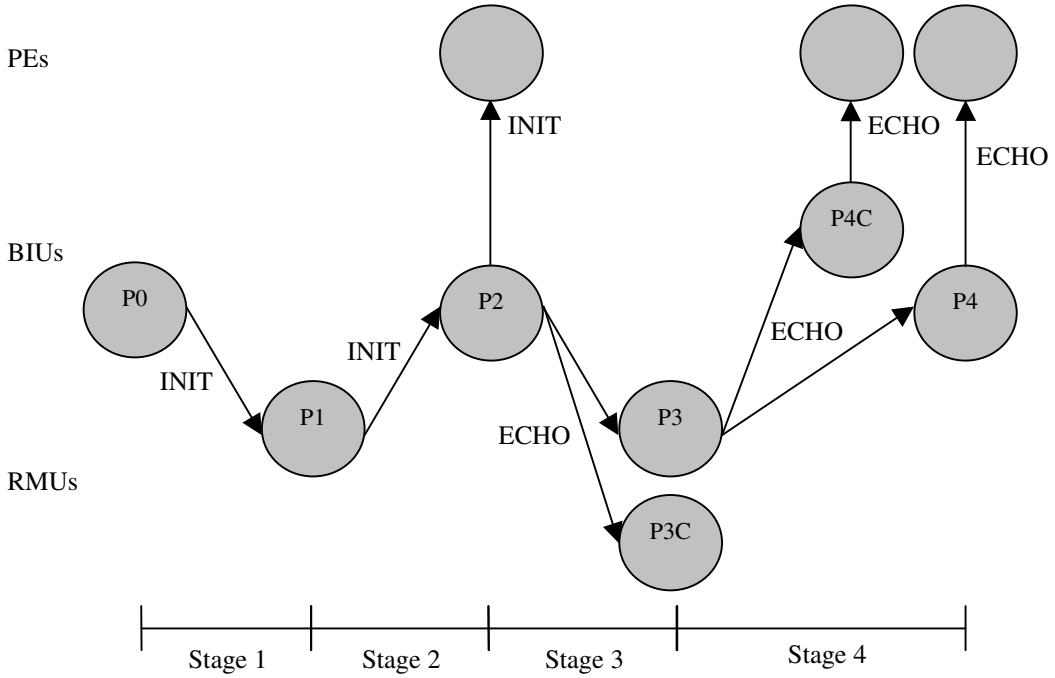


Figure 5.3: Combined message flow graph for the analysis of the synchronization protocols

5.2. Stage 1: P0 to P1

Figure 5.4 illustrates the detailed message flow graph for stage 1 in a 3x3 system (i.e., 3 BIUs and 3 RMUs).

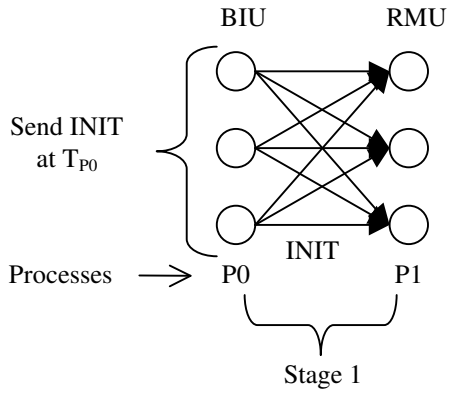


Figure 5.4: Detailed message flow graph for stage 1 in a 3x3 system

5.2.1. Expected time of reception for process P1

The analysis for point-to-point communication presented in the Section 4 of this document can be leveraged for the problem of determining the local time range of reception of INIT messages in process P1. This is covered in Section 5.9.2.1 for the Initial Synchronization and Synchronization Preservation protocols.

5.2.2. Bound on the observed relative skew of received messages for process P1

Let $\Pi_{P1,RCV}$ denote the bound on the relative skew observed in process P1 for the received messages from process P0 at trustworthy BIUs. $\Pi_{P1,RCV}$ is measured in local clock ticks. $\Pi_{P1,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

Let T_{P0} denotes the local time at which a BIU node sends the INIT message in process P0 (i.e., the local time when the source's Computation Module signals the Communication Module to send the INIT message). $t_{P0,l}$ and $t_{P0,h}$ denote the earliest and latest real times, respectively, at which the trustworthy BIU nodes send INIT in process P0. Let π_{P0} denote the bound on the relative local-time skew for the trustworthy BIUs. π_{P0} is assumed to apply for the duration of the protocol execution. π_{P0} also bounds the precision with which the trustworthy BIU nodes send the INIT messages.

$$\pi_{P0} = t_{P0,h} - t_{P0,l} \tag{5.2}$$

Let $t_{P1,RCV,l}$ and $t_{P1,RCV,h}$ denote the earliest and latest real times, respectively, at which an INIT message from a trustworthy BIU node can be received in process P1 at the trustworthy RMUs.

$$t_{P1,RCV,l} = t_{P0,l} + r_{PP,l} \tag{5.3}$$

$$t_{P1,RCV,h} = t_{P0,h} + r_{PP,h} \tag{5.4}$$

Let $T_{P1,RCV,l}$ and $T_{P1,RCV,h}$ denote the earliest and latest local times, respectively, at which a node in process P1 can receive messages from process P0 at trustworthy BIU nodes. $\Delta_{P1,RCV}$ denotes the measured skew between the earliest and latest received messages from trustworthy BIU nodes (i.e., $\Delta_{P1,RCV} = T_{P1,RCV,h} - T_{P1,RCV,l}$). We need to determine the maximum value of $\Delta_{P1,RCV}$. Using (5.3), (5.4), and the local-clock function :

$$c_{RCV}(T_{P1,RCV,h}) - c_{RCV}(T_{P1,RCV,l}) \leq t_{P1,RCV,h} - t_{P1,RCV,l} \tag{5.5}$$

From the constraint that the drift rate of the local clocks be ρ_0-bounded and the definition of $\Delta_{P1,RCV}$:

$$\Delta_{P1,RCV}/(1 + \rho_0) \leq c_{RCV}(T_{P1,RCV,h}) - c_{RCV}(T_{P1,RCV,l}) \tag{5.6}$$

Combining (5.5) and (5.6), and using the fact that $\Delta_{P1,RCV}$ is an integer:

$$\Delta_{P1,RCV} \leq \lfloor (1 + \rho_0)(t_{P1,RCV,h} - t_{P1,RCV,l}) \rfloor \tag{5.7}$$

$\Pi_{P1,RCV}$ is given by the maximum value of $\Delta_{P1,RCV}$:

$$\Pi_{P1,RCV} = \Delta_{P1,RCV}|_{max} = \lfloor (1 + \rho_0)(t_{P1,RCV,h} - t_{P1,RCV,l}) \rfloor = \lfloor (1 + \rho_0)(\pi_{P0} + e_{PP}) \rfloor \tag{5.8}$$

5.2.3. Relative skew of the Accept outputs for process P1

Let A_{P1} denote the delay (in local-clock ticks) of the Computation Process in process P1 measured from the local time of reception of the selected message until the Accept output is asserted. $t_{P1,A,l}$ and $t_{P1,A,h}$ denote the earliest and latest real times, respectively, at which an Accept output in process P1 at the trustworthy RMUs can be asserted.

$$t_{P1,A,l} = t_{P0,l} + r_{PP,l} + A_{P1}/(1 + \rho_0) \tag{5.9}$$

$$t_{P1,A,h} = t_{P0,h} + r_{PP,h} + (1 + \rho_0)A_{P1} \tag{5.10}$$

Therefore, the Accept functions of the trustworthy RMU nodes assert their outputs during a real-time interval with the following duration:

$$\begin{aligned} t_{P1,A,h} - t_{P1,A,l} &= [\pi_{P0} + r_{PP,h} + (1 + \rho_0)A_{P1}] - [r_{PP,l} + A_{P1}/(1 + \rho_0)] \\ &= \pi_{P0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \end{aligned} \tag{5.11}$$

Let AEV_P1 denote the set of asymmetric BIU eligible voters in process P1 at a trustworthy RMU node. |AEV_P1| denotes the cardinality of AEV_P1. $\pi_{P1,A}$ denotes the bound on the real-time relative skew of the Accept outputs in process P1 at the trustworthy RMUs. If |AEV_P1| = 0 for each trustworthy RMU node, they essentially accept on the same message.

$$\pi_{P1,A}|_{|AEV\_P1| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \tag{5.12}$$

If $|AEV\_P1| \neq 0$ for some trustworthy RMU, all we know with certainty is that the RMU nodes accept on a message from a trustworthy BIU node or a message from an untrustworthy BIU node flanked by messages from trustworthy BIU nodes.

$$\pi_{P1,A}|_{|AEV\_P1|\neq 0} = \pi_{P0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \quad (5.13)$$

From this point on, unless otherwise stated:

$$\pi_{P1,A} = \pi_{P1,A}|_{max} = \pi_{P1,A}|_{|AEV\_P1|\neq 0} \quad (5.14)$$

5.3. Stage 2: P1 to P2

Figure 5.5 illustrates the detailed message flow graph for stage 1 and 2 in a 3x3 system.

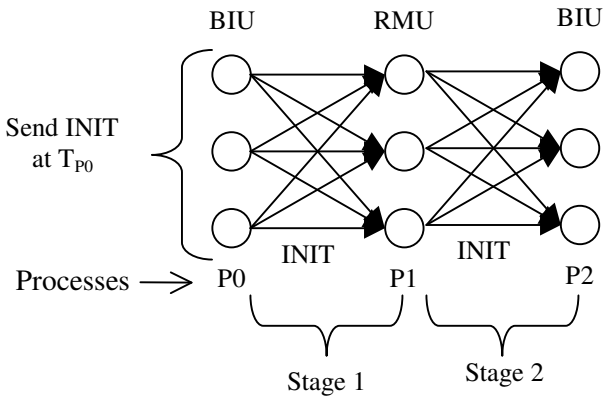


Figure 5.5: Detailed message flow graph for stages 1 and 2 in a 3x3 system

5.3.1. Effective reception delay for process P2

Let B_{P0} denote the send delay for process P0. We want to compute the effective reception delay for process P2. In general, this delay is measured from the time of some local event to the time of reception. We use T_{P0}, the local time of transmission of the INIT message in process P0, as the local reference event to measure the reception delay from process P0 to process P2. Note that instead of the start of the protocol, we choose the send time for process P0 as the reference time to measure the reception delay. This approach enables the analysis of the synchronization protocols independently of B_{P0}. B_{P0} is computed based on a single-stage point-to-point synchronous communication model (see Section 5.9.2.1).

We need to determine the earliest and latest real times of reception for process P2. Let B_{P1} denote the send delay for process P1. $t_{P2,RCV,l}$ and $t_{P2,RCV,h}$ denote the earliest and latest real times, respectively, at which an INIT message from a trustworthy RMU node can be received in process P2 at the trustworthy BIUs.

$$t_{P2,RCV,l} = t_{P1,A,l} + B_{P1}/(1 + \rho_0) + r_{PP,l} = t_{P0,l} + 2r_{PP,l} + (A_{P1} + B_{P1})/(1 + \rho_0) \quad (5.15)$$

$$t_{P2,RCV,h} = t_{P1,A,h} + (1 + \rho_0)B_{P1} + r_{PP,h} = t_{P0,l} + \pi_{P0} + 2r_{PP,h} + (1 + \rho_0)(A_{P1} + B_{P1}) \quad (5.16)$$

$r_{P0\text{-}P2,l}$ denotes the minimum effective message-reception delay for INIT messages in process P2 and is measured from the latest time at which the trustworthy BIU nodes can send INIT to the earliest time at which the BIU nodes can receive INIT messages from the trustworthy RMU nodes.

$$r_{P0\text{-}P2,l} = t_{P2,RCV,l} - t_{P0,h} = 2r_{PP,l} + (A_{P1} + B_{P1})/(1 + \rho_0) - \pi_{P0} \quad (5.17)$$

$r_{P0\text{-}P2,h}$ denotes the maximum effective message-reception delay for INIT messages in process P2 and is measured from the earliest time at which the trustworthy BIU nodes can send INIT to the latest time at which the BIU nodes can receive INIT messages from the trustworthy RMU nodes.

$$r_{P0\text{-}P2,h} = t_{P2,RCV,h} - t_{P0,l} = \pi_{P0} + 2r_{PP,h} + (1 + \rho_0)(A_{P1} + B_{P1}) \quad (5.18)$$

The expected reception delay for process P2 is:

$$R_{P0\text{-}P2} = IMP(r_{P0\text{-}P2,l}\ ,\ r_{P0\text{-}P2,h}) \quad (5.19)$$

The total effective uncertainty in the real time of reception of the INIT messages in process P2 is:

$$r_{P0\text{-}P2,h} - r_{P0\text{-}P2,l} = 2\pi_{P0} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1}) \quad (5.20)$$

5.3.2. Expected time of reception for process P2

The BIU nodes expect to receive INIT messages at local time $T_{P2,RCV,E}$.

$$T_{P2,RCV,E} = T_{P0} + R_{P0\text{-}P2} \quad (5.21)$$

The real-time error for $T_{P2,RCV,E}$ is bounded as follows. A BIU node will receive an INIT message from a trustworthy RMU node no earlier than $\mu_{P0\text{-}P2,l}$ nominal ticks from $T_{P2,RCV,E}$.

$$\mu_{P0\text{-}P2,l} = (1 + \rho_0)R_{P0\text{-}P2} - r_{P0\text{-}P2,l} \quad (5.22)$$

A BIU node will receive an INIT message from a trustworthy RMU node no later than $\mu_{P0\text{-}P2,h}$ nominal ticks from $T_{P2,RCV,E}$.

$$\mu_{P0\text{-}P2,h} = r_{P0\text{-}P2,h} - R_{P0\text{-}P2}/(1 + \rho_0) \quad (5.23)$$

Let $T_{P2,RCV}$ denote the actual local time at a BIU node when an INIT message from a trustworthy RMU node is received. In addition, let $\Delta_{P2,RCV}$ denote the local-time error in $T_{P2,RCV}$.

$$\Delta_{P2,RCV} = T_{P2,RCV} - T_{P2,RCV,E} \quad (5.24)$$

We want to determine a bound for the local-time error in the actual time of reception in process P2, denoted by $\Delta_{P2,RCV}|_{max}$.

$$|T_{P2,RCV} - T_{P2,RCV,E}| \leq \Delta_{P2,RCV}|_{max} \quad (5.25)$$

$\Delta_{P2,RCV}|_{max}$ is derived as follows. We know that the difference between the expected and the actual time of reception at a BIU node for INIT messages from the trustworthy RMU nodes is bounded by $\mu_{P0\text{-}P2,l}$ and $\mu_{P0\text{-}P2,h}$, such that:

$$c_{RCV}(T_{P2,RCV,E}) - \mu_{P0\text{-}P2,l} \leq c_{RCV}(T_{P2,RCV}) \leq c_{RCV}(T_{P2,RCV,E}) + \mu_{P0\text{-}P2,h} \quad (5.26)$$

So:

$$|c_{RCV}(T_{P2,RCV}) - c_{RCV}(T_{P2,RCV,E})| \leq \max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \tag{5.27}$$

From the constraint that the local clocks be ρ-bounded and (5.24):

$$|\Delta_{P2,RCV}|/(1 + \rho_0) \leq |c_{RCV}(T_{P2,RCV}) - c_{RCV}(T_{P2,RCV,E})| \tag{5.28}$$

Combining (5.27) and (5.28):

$$|\Delta_{P2,RCV}| \leq (1 + \rho_0)\max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \tag{5.29}$$

Since $\Delta_{P2,RCV}$ is an integer:

$$|\Delta_{P2,RCV}| \leq \lfloor (1 + \rho_0)\max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \rfloor \tag{5.30}$$

Therefore:

$$\Delta_{P2,RCV}|_{max} = \lfloor (1 + \rho_0)\ \max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \rfloor \tag{5.31}$$

5.3.3. Bound on the observed relative skew of received messages for process P2

Let $\Pi_{P2,RCV}$ denote the bound on the relative skew observed in process P2 for the received messages from process P1 at trustworthy RMUs. $\Pi_{P2,RCV}$ is measured in local clock ticks. $\Pi_{P2,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P1 at the trustworthy RMU nodes. The bound on the relative skew of the Accept outputs in process P1 is $\pi_{P1,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs to the time of reception in process P2 is $e_{PP} + [(1+\rho_0) - 1/(1+\rho_0)]B_{P1}$.

$$\Pi_{P2,RCV} = \lfloor (1 + \rho_0)(t_{P2,RCV,h} - t_{P2,RCV,l}) \rfloor$$

$$= \lfloor (1 + \rho_0)\{\pi_{P1,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P1}\} \rfloor$$

$$\Pi_{P2,RCV} = \lfloor (1 + \rho_0)\{\pi_{P0} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1})\} \rfloor \tag{5.32}$$

5.3.4. Relative skew of the Accept outputs for process P2

Let AEV_P2 denote the set of asymmetric RMU eligible voters in process P2 at a trustworthy BIU node. Let $\pi_{P2,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P2 at trustworthy BIUs. A_{P2} denotes the delay (in local-clock ticks) of the Computation Process in process P2 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If |AEV_P1| = 0 for each trustworthy RMU node, the trustworthy BIU nodes may have asymmetric RMU nodes in their sets of eligible voters for process P2 (i.e., |AEV_P2| ≠ 0 for some trustworthy BIUs). In this case, the trustworthy BIU nodes accept within the time range delimited by messages from trustworthy RMU nodes.

$$\pi_{P2,A}|_{|AEV\_P2| \neq 0} = \pi_{P1,A}|_{|AEV\_P1| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P1} + A_{P2})$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2}) \quad (5.33)$$

If |AEV_P1| ≠ 0 for some trustworthy RMU nodes, the trustworthy BIU nodes do not have asymmetric RMU nodes in their sets of eligible voters for process P2 (i.e., |AEV_P2| = 0 at each trustworthy BIU). In this case, the BIU nodes essentially accept on the same message.

$$\pi_{P2,A}|_{|AEV\_P2| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P2} \quad (5.34)$$

From this point on, unless otherwise stated:

$$\pi_{P2,A} = \pi_{P2,A}|_{max} = \pi_{P2,A}|_{|AEV\_P2| \neq 0} \quad (5.35)$$

5.4. Stage 3: P2 to P3

Figure 5.6 illustrates the detailed message flow graph up to stage 3 for a 3x3 system.

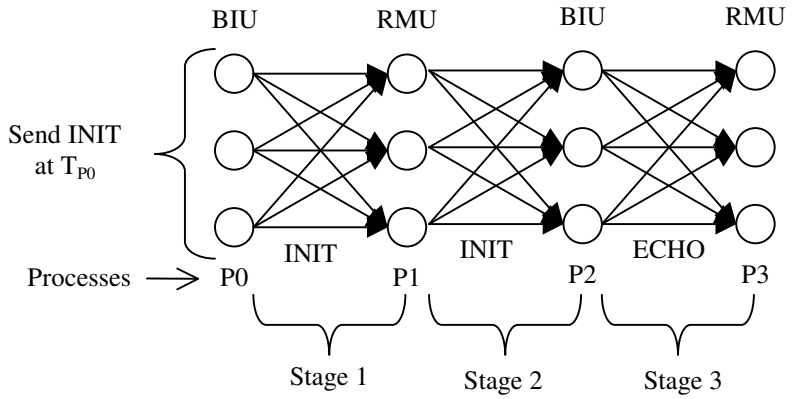


Figure 5.6: Detailed message flow graph for stages 1 through 3 in a 3x3 system

5.4.1. Effective reception delay for process P3

We need to determine the earliest and latest real times of reception of ECHO messages in process P3. Let $T_{P1,A,i}$ denote the local time at RMU node i when it asserts the output of its Accept function in process P1. Let $t_{P1,A,l}$ and $t_{P1,A,h}$ denote the earliest and latest real times, respectively, at which the trustworthy RMUs can assert the Accept outputs in process P1.

$$\pi_{P1,A} = t_{P1,A,h} - t_{P1,A,l} \quad (5.36)$$

Let B_{P2} denote the send delay for process P2. $t_{P3,RCV,l}$ and $t_{P3,RCV,h}$ denote the earliest and latest real times, respectively, at which ECHO messages from trustworthy BIU nodes can be received by an RMU node in process P3.

$$t_{P3,RCV,l} = t_{P1,A,l} + 2r_{PP,l} + (B_{P1} + A_{P2} + B_{P2})/(1 + \rho_0) \quad (5.37)$$

$$t_{P3,RCV,h} = t_{P1,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P1} + A_{P2} + B_{P2}) \quad (5.38)$$

$r_{P1\text{-}P3,l}$ denotes the minimum effective message-reception delay for ECHO messages in process P3 and is measured from the latest time at which trustworthy RMU nodes can assert their Accept(INIT) output to the earliest time at which the RMU nodes receive ECHO messages from the trustworthy BIU nodes.

$$r_{P1\text{-}P3,l} = t_{P3,RCV,l} - t_{P1,A,h} = 2r_{PP,l} + (B_{P1} + A_{P2} + B_{P2})/(1 + \rho_0) - \pi_{P1,A} \quad (5.39)$$

$r_{P1\text{-}P3,h}$ denotes the maximum effective message-reception delay for ECHO messages in process P3 and is measured from the earliest time at which the trustworthy RMU nodes can assert its Accept(INIT) output to the latest time at which the RMU nodes can receive ECHO messages from the trustworthy BIU nodes.

$$r_{P1\text{-}P3,h} = t_{P3,RCV,h} - t_{P1,A,l} = \pi_{P1,A} + 2r_{PP,h} + (1 + \rho_0)(B_{P1} + A_{P2} + B_{P2}) \quad (5.40)$$

The expected reception delay for process P3 is:

$$R_{P1\text{-}P3} = IMP(r_{P1\text{-}P3,l}\ ,\ r_{P1\text{-}P3,h}) \quad (5.41)$$

The effective uncertainty in the real time of reception of the ECHO messages in process P3 is:

$$r_{P1\text{-}P3,h} - r_{P1\text{-}P3,l} = 2\pi_{P1,A} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P1} + A_{P2} + B_{P2}) \quad (5.42)$$

5.4.2. Expected time of reception for process P3

RMU node i expects to receive ECHO messages at local time $T_{P3,RCV,E,i}$.

$$T_{P3,RCV,E,i} = T_{P1,A,i} + R_{P1\text{-}P3} \quad (5.43)$$

The real-time error for $T_{P3,RCV,E,i}$ is bounded as follows. RMU node i will receive an ECHO message from a trustworthy BIU node no earlier than $\mu_{P1\text{-}P3,l}$ nominal ticks from $T_{P3,RCV,E,i}$.

$$\mu_{P1\text{-}P3,l} = (1 + \rho_0)R_{P1\text{-}P3} - r_{P1\text{-}P3,l} \quad (5.44)$$

RMU node i will receive an ECHO message from a trustworthy BIU node no later than $\mu_{P1\text{-}P3,h}$ nominal ticks from $T_{P3,RCV,E,i}$.

$$\mu_{P1\text{-}P3,h} = r_{P1\text{-}P3,h} - R_{P1\text{-}P3}/(1 + \rho_0) \quad (5.45)$$

We want to determine the maximum local-time error for the actual time of reception at the RMU nodes, denoted by $\Delta_{P3,RCV}|_{max}$.

$$|T_{P3,RCV} - T_{P3,RCV,E}| \leq \Delta_{P3,RCV}|_{max} \quad (5.46)$$

Following the analysis for process P2:

$$\Delta_{P3,RCV}|_{max} = \lfloor (1 + \rho_0)\max(\mu_{P1\text{-}P3,l}\ ,\ \mu_{P1\text{-}P3,h}) \rfloor \quad (5.47)$$

5.4.3. Bound on the observed relative skew of received messages for process P3

Let $\Pi_{P3,RCV}$ denote the bound on the relative skew observed in process P3 for the received messages

from trustworthy sources in process P2. $\Pi_{P3,RCV}$ is measured in local clock ticks. $\Pi_{P3,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P2 at the trustworthy BIU nodes. The bound on the relative skew of the Accept outputs in process P2 is $\pi_{P2,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs in process P2 to the time of reception in process P3 is $e_{PP} + [(1+\rho_0) - 1/(1+\rho_0)]B_{P2}$.

$$\Pi_{P3,RCV} = \lfloor (1 + \rho_0)(t_{P3,RCV,h} - t_{P3,RCV,l}) \rfloor$$

$$= \lfloor (1 + \rho_0)\{\pi_{P2,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P2}\} \rfloor$$

$$= \lfloor (1 + \rho_0)\{3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2})\} \rfloor \quad (5.48)$$

5.4.4. Relative skew of the Accept outputs for process P3

Let AEV_P3 denote the set of asymmetric BIU eligible voters in process P3 at a trustworthy RMU node. Let $\pi_{P3,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P3 at the trustworthy RMUs. A_{P3} denotes the delay (in local-clock ticks) of the Computation Process in process P3 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If $|AEV\_P2| = 0$ for each trustworthy BIU node, the trustworthy RMU nodes may have asymmetric BIU nodes in their sets of eligible voters for process P3 (i.e., $|AEV\_P3| \neq 0$ for some trustworthy RMU nodes). In this case, the trustworthy RMU nodes accept within the time range delimited by messages from trustworthy BIU nodes.

$$\pi_{P3,A}|_{|AEV\_P3| \neq 0} = \pi_{P2,A}|_{|AEV\_P2| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3})$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3}) \quad (5.49)$$

If $|AEV\_P2| \neq 0$ for some trustworthy BIU nodes, the trustworthy RMU nodes do not have asymmetric BIU nodes in their sets of eligible voters for process P3 (i.e., $|AEV\_P3| = 0$ for each trustworthy RMU node). In this case, the RMU nodes essentially accept on the same message.

$$\pi_{P3,A}|_{|AEV\_P3| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P3} \quad (5.50)$$

From this point on, unless otherwise stated:

$$\pi_{P3,A} = \pi_{P3,A}|_{max} = \pi_{P3,A}|_{|AEV\_P3| \neq 0} \quad (5.51)$$

5.5. Stage 4: P3 to P4

Figure 5.7 illustrates the detailed message flow graph up to stage 4 for a 3x3 system.

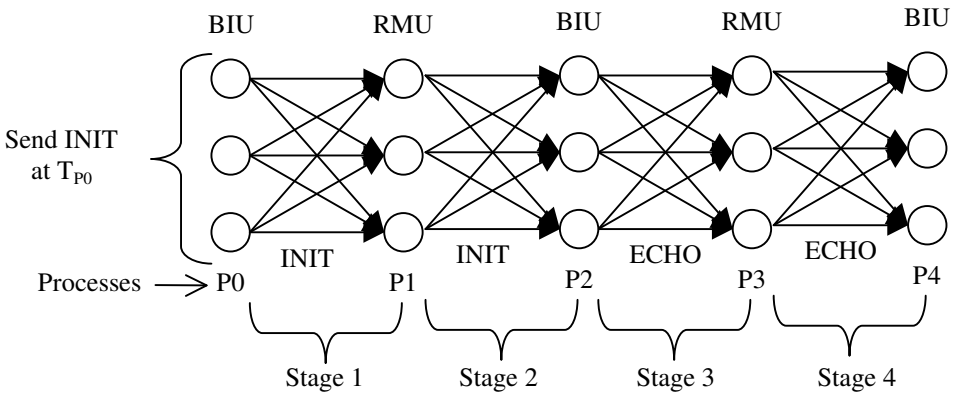


Figure 5.7: Detailed message flow graph for stages 1 through 4 in a 3x3 system

5.5.1. Effective reception delay for process P4

We need to determine the earliest and latest real time of reception of ECHO messages from trustworthy RMU nodes by the BIU nodes in process P4. Let $T_{P2,A,j}$ denote the local time at which trustworthy BIU node j asserts the output of its Accept function for process P2. $t_{P2,A,l}$ and $t_{P2,A,h}$ denote the earliest and latest real times, respectively, at which the trustworthy BIUs can assert their Accept outputs in process P2.

$$\pi_{P2,A} = t_{P2,A,h} - t_{P2,A,l} \tag{5.52}$$

Let B_{P3} denote the send delay for process P3. $t_{P4,RCV,l}$ denotes the earliest real time at which ECHO messages from trustworthy RMU nodes can be received by a BIU node in process P4.

$$t_{P4,RCV,l} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3})/(1 + \rho_0) \tag{5.53}$$

$t_{P4,RCV,h}$ denotes the latest real time at which ECHO messages from trustworthy RMU nodes can be received by a BIU node in process P4.

$$t_{P4,RCV,h} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3}) \tag{5.54}$$

$r_{P2\text{-}P4,l}$ denotes the minimum effective message-reception delay for ECHO messages in process P4 and is measured from the latest time at which the trustworthy BIU nodes can assert their Accept(ECHO) outputs to the earliest time at which the BIU nodes can receive ECHO messages from the trustworthy RMU nodes.

$$r_{P2\text{-}P4,l} = t_{P4,RCV,l} - t_{P2,A,h} = 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3})/(1 + \rho_0) - \pi_{P2,A} \tag{5.55}$$

$r_{P2\text{-}P4,h}$ denotes the maximum effective message-reception delay for ECHO messages in process P4 and is measured from the earliest time at which the trustworthy BIU nodes assert their Accept(ECHO) outputs to the latest time at which the BIU nodes can receive ECHO messages from the trustworthy RMU nodes.

$$r_{P2\text{-}P4,h} = t_{P4,RCV,h} - t_{P2,A,l} = \pi_{P2,A} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3}) \tag{5.56}$$

The expected reception delay for process P4 is:

$$R_{P2\text{-}P4} = IMP(r_{P2\text{-}P4,l} , r_{P2\text{-}P4,h}) \quad (5.57)$$

The effective uncertainty in the real time of reception of the ECHO messages in process P4 is:

$$r_{P2\text{-}P4,h} - r_{P2\text{-}P4,l} = 2\pi_{P2,A} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3} + B_{P3}) \quad (5.58)$$

5.5.2. Expected time of reception for process P4

BIU node j expect to receive ECHO messages at local time $T_{P4,RCV,E,j}$:

$$T_{P4,RCV,E,j} = T_{P2,A,j} + R_{P2\text{-}P4} \quad (5.59)$$

The real-time error for $T_{P4,RCV,E,j}$ is bounded as follows. BIU node j will receive an ECHO message from a trustworthy RMU node no earlier than $\mu_{P2\text{-}P4,l}$ nominal ticks from $T_{P4,RCV,E,j}$:

$$\mu_{P2\text{-}P4,l} = (1 + \rho_0)R_{P2\text{-}P4} - r_{P2\text{-}P4,l} \quad (5.60)$$

BIU node j will receive an ECHO message from a trustworthy RMU node no later than $\mu_{P2\text{-}P4,h}$ nominal ticks from $T_{P4,RCV,E,j}$:

$$\mu_{P2\text{-}P4,h} = r_{P2\text{-}P4,h} - R_{P2\text{-}P4}/(1 + \rho_0) \quad (5.61)$$

We want to determine the maximum local-time error for the actual time of reception at the BIU nodes in process P4, denoted by $\Delta_{P4,RCV}|_{max}$.

$$|T_{P4,RCV} - T_{P4,RCV,E}| \leq \Delta_{P4,RCV}|_{max} \quad (5.62)$$

Following the analysis for process P2:

$$\Delta_{P4,RCV}|_{max} = \lfloor (1 + \rho_0)\max(\mu_{P2\text{-}P4,l} , \mu_{P2\text{-}P4,h}) \rfloor \quad (5.63)$$

5.5.3. Bound on the observed relative skew of received messages for process P4

Let $\Pi_{P4,RCV}$ denote the bound on the relative skew observed in process P4 for the received messages from process P3 at trustworthy RMUs. $\Pi_{P4,RCV}$ is measured in local clock ticks. $\Pi_{P4,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P3 at trustworthy RMU nodes. The bound on the relative skew of the Accept outputs in process P3 at the trustworthy RMUs is $\pi_{P3,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs in process P3 to the time of reception in process P4 is $e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P3}$.

$$\Pi_{P4,RCV} = \lfloor (1 + \rho_0)(t_{P4,RCV,h} - t_{P4,RCV,l}) \rfloor$$

$$= \lfloor (1 + \rho_0)\{\pi_{P3,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P3}\} \rfloor$$

$$= \lfloor (1 + \rho_0)\{3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3})\} \rfloor \quad (5.64)$$

5.5.4. Relative skew of the Accept outputs for process P4

Let AEV_P4 denote the set of asymmetric RMU eligible voters in process P4 at a trustworthy BIU node. Let $\pi_{P4,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P4 at the trustworthy BIUs. A_{P4} denotes the delay (in local-clock ticks) of the Computation Process in process P4 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If |AEV_P3| = 0 for each trustworthy RMU node, the trustworthy BIU nodes may have asymmetric RMU nodes in their sets of eligible voters for process P4 (i.e., |AEV_P4| ≠ 0 for some trustworthy BIU nodes). In this case, the BIU nodes accept within the time range delimited by messages from trustworthy RMU nodes.

$$\pi_{P4,A}|_{|AEV\_P4| \neq 0} = \pi_{P3,A}|_{|AEV\_P3| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P3} + A_{P4})$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P3} + B_{P3} + A_{P4}) \quad (5.65)$$

If |AEV_P3| ≠ 0 for some trustworthy RMU nodes, the trustworthy BIU nodes do not have asymmetric RMU nodes in their sets of eligible voters for process P4 (i.e., |AEV_P4| = 0 for each trustworthy BIU node). In this case, the BIU nodes essentially accept on the same message.

$$\pi_{P4,A}|_{|AEV\_P4| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P4} \quad (5.66)$$

From this point on, unless otherwise stated:

$$\pi_{P4,A} = \pi_{P4,A}|_{max} = \pi_{P4,A}|_{|AEV\_P4| \neq 0} \quad (5.67)$$

5.6. Synchronization capture

Figure 5.8 illustrates the detailed message flow graph for the synchronization-capture stages in a 3x3 system. The nodes executing processes P3C and P4C are called **recovering** nodes.

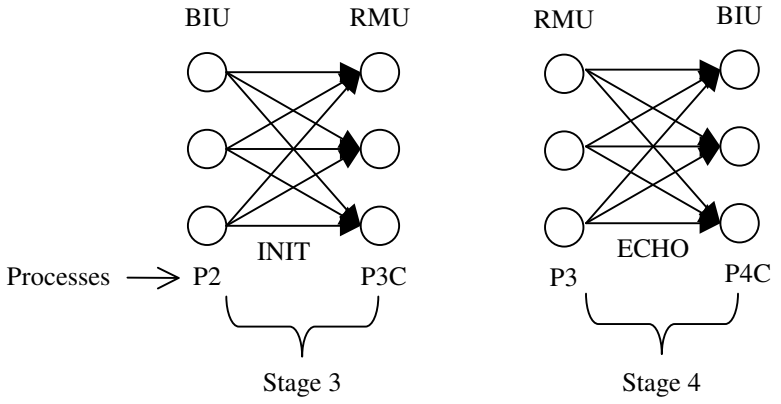


Figure 5.8: Detailed message flow graph for synchronization-capture stages in a 3x3 system

5.6.1. Bound on the observed relative skew of received messages for process P3C

Let $\Pi_{P3C,RCV}$ denote the bound on the relative skew observed in process P3C for the received messages from process P2 at trustworthy BIUs. $\Pi_{P3C,RCV}$ is measured in local clock ticks. $\Pi_{P3C,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The nodes in process P3C receive ECHO messages from process P2 at trustworthy BIUs in the same real time range as nodes executing process P3. Therefore:

$$\Pi_{P3C,RCV} = \Pi_{P3,RCV} \tag{5.68}$$

5.6.2. Relative skew of the Accept outputs for process P3C

Recovering RMU nodes synchronize using the ECHO messages from process P2 of the Synchronization Preservation protocol. Because the recovering nodes may have asymmetric faulty nodes in their sets of eligible voters, all we know is that they will accept within the time range delimited by ECHO messages from trustworthy BIU nodes. Let $\pi_{P3C,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P3C at the good recovering RMUs. Let A_{P3C} denote the delay (in local-clock ticks) of the Computation Process in process P3C measured from the local time of reception of the selected message to the local time when the Accept output is asserted. We assume that the delay of the Computation Process in process P3C is the same as in process P3.

$$A_{P3C} = A_{P3} \tag{5.69}$$

The worst-case real-time relative skew occurs when the trustworthy BIU nodes and the recovering RMU nodes simultaneously have asymmetric nodes in their sets of eligible voters. For that case:

$$\begin{aligned}\pi_{P3C,A} &= \pi_{P2,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3C}) \\ &= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3})\end{aligned} \tag{5.70}$$

5.6.3. Bound on the observed relative skew of received messages for process P4C

Let $\Pi_{P4C,RCV}$ denote the bound on the maximum relative skew observed in process P4C for the received messages from process P3 at trustworthy RMUs. $\Pi_{P4C,RCV}$ is measured in local clock ticks. $\Pi_{P4C,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The nodes executing process P4C receive ECHO messages from process P3 at trustworthy RMUs in the same time range as nodes executing process P4. Therefore:

$$\Pi_{P4C,RCV} = \Pi_{P4,RCV} \tag{5.71}$$

5.6.4. Relative skew of the Accept outputs for process P4C

Recovering BIU nodes synchronize using the ECHO messages from process P3 of the

Synchronization Preservation protocol. Because the recovering BIUs may have asymmetric faulty nodes in their sets of eligible voters, all we know is that they will accept within the time range delimited by ECHO messages from trustworthy RMU nodes. Let $\pi_{P4C,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P4C at the good recovering BIUs. Let A_{P4C} denote the delay (in local-clock ticks) of the Computation Process in process P4C measured from the local time of reception of the selected message to the local time when the Accept output is asserted. We assume that the delay of the Computation Process in process P4C is the same as in process P4.

$$A_{P4C} = A_{P4} \tag{5.72}$$

The worst-case real-time relative skew occurs when the trustworthy RMU nodes and the recovering BIU nodes simultaneously have asymmetric nodes in their sets of eligible voters. For that case:

$$\begin{aligned}\pi_{P4C,A} &= \pi_{P3,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P3} + A_{P4C}) \\ &= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4})\end{aligned} \tag{5.73}$$

5.7. Resetting the local time

5.7.1. Relative skew of the local-time reset for process P4

Let $T_{P4,A,j}$ denote the local time of the Accept output in process P4 at trustworthy BIU j. H_{P4} denotes the synchronization-reset delay applied by the BIU nodes resetting with respect to the Accept output in process P4. $T_{P4,H,j}$ denotes the local time at which the next cycle begins for BIU node j synchronizing with respect to process P4.

$$T_{P4,H,j} = T_{P4,A,j} + H_{P4} \tag{5.74}$$

$\pi_{P4,H}$ denotes the bound on the relative skew of the local-time reset for BIU nodes synchronizing with respect to process P4. Then:

$$\begin{aligned}\pi_{P4,H} &= \pi_{P4,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P4} \\ &= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P3} + B_{P3} + A_{P4} + H_{P4})\end{aligned} \tag{5.75}$$

5.7.2. Relative skew of the local-time reset for process P4C

Let $T_{P4C,A,j}$ denote the local time of the Accept output in process P4C at a good recovering BIU j. H_{P4C} denotes the synchronization-reset delay applied by the nodes resetting with respect to the Accept output in process P4C at the good recovering BIUs. $T_{P4C,H,j}$ denotes the local time at which the next cycle begins for BIU node j synchronizing with respect to process P4C. The BIU nodes executing process P4C apply the same synchronization-reset delay as the nodes executing process P4.

$$H_{P4C} = H_{P4} \tag{5.76}$$

So:

$$T_{P4C,H,j} = T_{P4C,A,j} + H_{P4C} = T_{P4C,A,j} + H_{P4} \quad (5.77)$$

The bound on the relative skew of the Accept output for process P4C at the good recovering BIUs is given by $\pi_{P4C,A}$. $\pi_{P4C,H}$ denotes the bound on the relative skew of the local-time reset for good recovering BIU nodes synchronizing with respect to process P4C. Then:

$$\pi_{P4C,H} = \pi_{P4C,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P4}$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad (5.78)$$

5.7.3. Reset delay for process P3

Let $T_{P3,A,i}$ denote the local time of the Accept output in process P3 at trustworthy RMU i. H_{P3} denotes the synchronization-reset delay applied by the nodes resetting with respect to the Accept output in process P3. $T_{P3,H,i}$ denotes the local time at which the next cycle begins for RMU i synchronizing with respect to process P3.

$$T_{P3,H,i} = T_{P3,A,i} + H_{P3} \quad (5.79)$$

H_{P3} is the expected delay from the time when the RMU nodes in process P3 assert their Accept output until the BIU nodes synchronizing with respect to process P4 reset their local-time clocks. The bound on the relative skew of the Accept outputs in process P3 at the trustworthy RMUs is given by $\pi_{P3,A}$. $t_{P3,A,l}$ and $t_{P3,A,h}$ denote the earliest and latest real times, respectively, at which the Accept outputs can be asserted in process P3 at the trustworthy RMUs. So:

$$\pi_{P3,A} = t_{P3,A,h} - t_{P3,A,l} \quad (5.80)$$

$t_{P4,H,l}|_{P3,A}$ and $t_{P4,H,h}|_{P3,A}$ denote the earliest and latest real times, respectively, at which a trustworthy BIU node synchronizing with respect to process P4 can reset its local-time clock. $t_{P4,H,l}|_{P3,A}$ and $t_{P4,H,h}|_{P3,A}$ are measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4,H,l}|_{P3,A} = t_{P3,A,l} + r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (5.81)$$

$$t_{P4,H,h}|_{P3,A} = t_{P3,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P3} + A_{P4} + H_{P4}) \quad (5.82)$$

Let $h_{P3,l}$ denote the minimum effective delay from the time the Accept output in process P3 at trustworthy RMUs is asserted to the time a trustworthy BIU node resets its local-time clock with respect to process P4.

$$h_{P3,l} = t_{P4,H,l}|_{P3,A} - t_{P3,A,h}$$

$$= r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) - \pi_{P3,A} \quad (5.83)$$

$h_{P3,h}$ denotes the maximum effective delay from the time the Accept output in process P3 at trustworthy RMUs is asserted to the time a trustworthy BIU node resets its local-time clock with respect to process P4.

$$h_{P3,h} = t_{P4,H,h}|_{P3,A} - t_{P3,A,l}$$

$$= \pi_{P3,A} + r_{PP,h} + (B_{P3} + A_{P4} + H_{P4})(1 + \rho_0) \quad (5.84)$$

H_{P3} is given by:

$$H_{P3} = IMP(h_{P3,l}, h_{P3,h}) \quad (5.85)$$

The real-time error for $T_{P3,H,i}$ is bounded as follows. A trustworthy BIU node can reset its local-time clock with respect to process P4 no earlier than $\mu_{P3,H,l}$ nominal ticks from local time $T_{P3,H,i}$ at a trustworthy RMU node synchronizing with respect to process P3.

$$\mu_{P3,H,l} = (1 + \rho_0)H_{P3} - h_{P3,l} \quad (5.86)$$

A trustworthy BIU node can reset its local-time clock with respect to process P4 no later than $\mu_{P3,H,h}$ nominal ticks from local time $T_{P3,H,i}$ at a trustworthy RMU node synchronizing with respect to process P3.

$$\mu_{P3,H,h} = h_{P3,h} - H_{P3}/(1 + \rho_0) \quad (5.87)$$

Note that this analysis also applies to the real-time error for $T_{P3,H,i}$ with respect to the local-time reset of nodes synchronizing in process P4C.

5.7.4. Relative skew of the local-time reset between processes P3, and P4 or P4C

Let $\pi_{P3\text{-}P4,H}$ denote the bound on the relative skew of the local-time reset between RMU nodes synchronizing with respect to process P3 and BIU nodes synchronizing with respect to process P4.

$$\pi_{P3\text{-}P4,H} = \max(\mu_{P3,H,l}, \mu_{P3,H,h}) \quad (5.88)$$

$\pi_{P3\text{-}P4C,H}$ denotes the bound on the relative skew of the local-time reset between RMU nodes synchronizing with respect to process P3 and BIU nodes synchronizing with respect to process P4C. $\pi_{P3\text{-}P4,H}$ also applies here.

$$\pi_{P3\text{-}P4C,H} = \pi_{P3\text{-}P4,H} \quad (5.89)$$

5.7.5. Relative skew of the local-time reset for process P3

The bound on the relative skew of the Accept outputs in process P3 at trustworthy RMUs is given by $\pi_{P3,A}$. $\pi_{P3,H}$ denotes the bound on the relative skew of the local-time reset for trustworthy RMU nodes resetting with respect to the Accept output in process P3.

$$\pi_{P3,H} = \pi_{P3,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P3}$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + H_{P3}) \quad (5.90)$$

5.7.6. Relative skew of the local-time reset for process P3C

Let $T_{P3C,A,i}$ denote the local time of the Accept output in process P3C at good recovering RMU i. H_{P3C} denotes the synchronization-reset delay applied by the RMU nodes resetting with respect to the Accept

output in process P3C. $T_{P3C,H,i}$ denotes the local time at which the next cycle begins for RMU node i synchronizing with respect to process P3C. The RMU nodes executing process P3C apply the same synchronization-reset delay as the RMU nodes executing process P3.

$$H_{P3C} = H_{P3} \tag{5.91}$$

So:

$$T_{P3C,H,i} = T_{P3C,A,i} + H_{P3C} = T_{P3C,A,i} + H_{P3} \tag{5.92}$$

The bound on the relative skew for the Accept outputs in process P3C at the good recovering RMUs is given by $\pi_{P3C,A}$. $\pi_{P3C,H}$ denotes the bound on the relative skew of the local-time reset for the nodes synchronizing with respect to the Accept output in process P3C.

$$\pi_{P3C,H} = \pi_{P3C,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P3}$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \tag{5.93}$$

5.7.7. Reset delay for process P2

Let $T_{P2,A,k}$ denote the local time of the Accept output in process P2 at trustworthy BIU k. H_{P2} denotes the synchronization-reset delay applied by the BIU nodes resetting with respect to the Accept output in process P2. $T_{P2,H,k}$ denotes the local time at which the next cycle begins for BIU node k synchronizing with respect to process P2.

$$T_{P2,H,k} = T_{P2,A,k} + H_{P2} \tag{5.94}$$

H_{P2} is the expected delay from the time when the BIU nodes executing process P2 assert their Accept outputs until the RMU nodes synchronizing with respect to process P3 reset their local-time clocks. $t_{P3,H,l}|_{P2,A}$ denotes the earliest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3,H,l}|_{P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \tag{5.95}$$

Let $t_{P3,H,h}|_{P2,A}$ denote the latest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3,H,h}|_{P2,A} = t_{P2,A,h} + r_{PP,h} + (B_{P2} + A_{P3} + H_{P3})(1 + \rho_0) \tag{5.96}$$

Let $h_{P2,l}$ denote the minimum effective delay from the time a trustworthy BIU node in process P2 asserts its Accept output until a trustworthy RMU node in process P3 resets its local-time clock.

$$h_{P2,l} = t_{P3,H,l}|_{P2,A} - t_{P2,A,h}$$

$$= r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) - \pi_{P2,A} \tag{5.97}$$

Let $h_{P2,h}$ denote the maximum effective delay from the time a trustworthy BIU node in process P2 asserts its Accept output until a trustworthy RMU node in process P3 resets its local-time clock.

$$h_{P2,h} = t_{P3,H,h}|_{P2,A} - t_{P2,A,l}$$

$$= \pi_{P2,A} + r_{PP,h} + (B_{P2} + A_{P3} + H_{P3})(1 + \rho_0) \quad (5.98)$$

H_{P2} is given by:

$$H_{P2} = IMP(h_{P2,l} , h_{P2,h}) \quad (5.99)$$

The real-time error for $T_{P2,H,k}$ is bounded as follows. A trustworthy RMU node in process P3 can reset its local-time clock no earlier than $\mu_{P2,H,l}$ nominal ticks from local time $T_{P2,H,k}$ at a BIU node synchronizing with respect to process P2.

$$\mu_{P2,H,l} = (1 + \rho_0)H_{P2} - h_{P2,l} \quad (5.100)$$

A trustworthy RMU node in process P3 can reset its local-time clock no later than $\mu_{P2,H,h}$ nominal ticks from local time $T_{P2,H,k}$ at a BIU node synchronizing with respect to process P2.

$$\mu_{P2,H,h} = h_{P2,h} - H_{P2}/(1 + \rho_0) \quad (5.101)$$

Note that this analysis also applies to the real-time error for $T_{P2,H,k}$ with respect to the local-time reset of nodes synchronizing with respect to process P3C.

5.7.8. Relative skew of the local-time reset between processes P2, and P3 or P3C

Let $\pi_{P2\text{-}P3,H}$ denote the bound on the relative skew of the local-time reset between trustworthy BIU nodes synchronizing with respect to process P2 and trustworthy RMU nodes synchronizing with respect to process P3.

$$\pi_{P2\text{-}P3,H} = \max(\mu_{P2,H,l} , \mu_{P2,H,h}) \quad (5.102)$$

$\pi_{P2\text{-}P3C,H}$ denotes the bound on the relative skew of the local-time reset between trustworthy BIU nodes synchronized with respect to process P2 and good recovering RMU nodes synchronized with respect to process P3C. $\pi_{P2\text{-}P3,H}$ also applies here.

$$\pi_{P2\text{-}P3C,H} = \pi_{P2\text{-}P3,H} \quad (5.103)$$

5.7.9. Relative skew of the local-time reset for process P2

The bound on the relative skew of the Accept outputs in process P2 at trustworthy BIUs is given by $\pi_{P2,A}$. $\pi_{P2,H}$ denotes the bound on the relative skew of the local-time reset for trustworthy BIU nodes synchronizing with respect to process P2. Then:

$$\pi_{P2,H} = \pi_{P2,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P2}$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + H_{P2}) \quad (5.104)$$

5.7.10. Relative skew of the local-time reset for a set including processes P2 and P3C

Let $t_{P3C,H,l}|_{P2,A}$ denote the earliest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at trustworthy BIUs.

$$t_{P3C,H,l}|_{P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad (5.105)$$

$t_{P3C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at trustworthy BIUs.

$$t_{P3C,H,h}|_{P2,A} = t_{P2,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + H_{P3}) \quad (5.106)$$

$t_{P2,H,l}$ denotes the earliest real time at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{P2,H,l} = t_{P2,A,l} + H_{P2}/(1 + \rho_0) \quad (5.107)$$

$t_{P2,H,h}$ denotes the latest real time at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{P2,H,h} = t_{P2,A,h} + H_{P2}(1 + \rho_0) \quad (5.108)$$

$\pi_{P2+P3C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all the trustworthy or good recovering nodes synchronizing with respect to process P2 or P3C.

$$\pi_{P2+P3C,H} = \max(|\, t_{P3C,H,h}|_{P2,A} - t_{P2,H,l}\,|, |\, t_{P2,H,h} - t_{P3C,H,l}|_{P2,A}\,|, \pi_{P2,H}, \pi_{P3C,H}) \quad (5.109)$$

5.7.11. Relative skew of the local-time reset for a set including processes P2 and P3

Let $\pi_{P2+P3,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy BIU nodes synchronizing with respect to process P2 or P3. With respect to process P2, the Accept outputs in process P3 at the trustworthy RMUs and the Accept outputs in process P3C at the good recovering RMUs can be asserted during the same real time interval. In the presence of asymmetric faulty BIU nodes, we know that the time interval of the Accept outputs in process P3 at the trustworthy RMUs is contained within the time interval of the Accept outputs in process P3C at the good recovering RMUs. Therefore:

$$\pi_{P2+P3,H} \leq \pi_{P2+P3C,H} \quad (5.110)$$

5.7.12. Relative skew of the local-time reset for a set including processes P2 and P4C

Let $t_{P4C,H,l}|_{P2,A}$ denote the earliest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,l}|_{P2,A} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (5.111)$$

$t_{P4C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,h}|_{P2,A} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad (5.112)$$

$\pi_{P2+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all BIU nodes synchronizing with respect to process P2 or P4C.

$$\pi_{P2+P4C,H} = \max(| t_{P4C,H,h}|_{P2,A} - t_{P2,H,l} |, | t_{P2,H,h} - t_{P4C,H,l}|_{P2,A} |, \pi_{P2,H}, \pi_{P4C,H}) \quad (5.113)$$

5.7.13. Relative skew of the local-time reset for a set including processes P2 and P4

Let $\pi_{P2+P4,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy BIU nodes synchronizing with respect to process P2 or P4. With respect to process P2, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. In the presence of asymmetric faulty RMU nodes, we know that the time interval of the Accept outputs in process P4 at the trustworthy BIUs is contained within the time interval of the Accept outputs in process P4C at the good recovering BIUs. Therefore:

$$\pi_{P2+P4,H} \leq \pi_{P2+P4C,H} \quad (5.114)$$

5.7.14. Relative skew of the local-time reset for a set including processes P3 or P3C

Let $\pi_{P3+P3C,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering RMU nodes synchronizing with respect to process P3 or P3C. With respect to process P2, the Accept outputs in process P3 at the trustworthy RMUs and the Accept outputs in process P3C at the good recovering RMUs can be asserted during the same real time interval. This interval is determined by the time range during which the trustworthy BIU nodes executing process P2 send ECHO. In the presence of asymmetric faulty BIU nodes, the good recovering RMU nodes executing process P3C may not be able to synchronize any better than the duration of this time range.

$$\pi_{P3+P3C,H} = \max(\pi_{P3,H}, \pi_{P3C,H}) = \pi_{P3C,H} \quad (5.115)$$

5.7.15. Relative skew of the local-time reset for a set including processes P3 and P4C

Let $t_{P4C,H,l}|_{P3,A}$ denote the earliest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4C,H,l}|_{P3,A} = t_{P3,A,l} + r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (5.116)$$

$t_{P4C,H,h}|_{P3,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4C,H,h}|_{P3,A} = t_{P3,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P3} + A_{P4} + H_{P4}) \quad (5.117)$$

$t_{P3,H,l}|_{P3,A}$ denotes the earliest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P3,H,l}|_{P3,A} = t_{P3,A,l} + H_{P3}/(1 + \rho_0) \quad (5.118)$$

$t_{P3,H,h}|_{P3,A}$ denotes the latest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P3,H,h}|_{P3,A} = t_{P3,A,h} + H_{P3}(1 + \rho_0) \quad (5.119)$$

$\pi_{P3+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering nodes synchronizing with respect to process P3 or P4C.

$$\pi_{P3+P4C,H} = \max(| t_{P4C,H,h}|_{P3,A} - t_{P3,H,l}|_{P3,A} |, | t_{P3,H,h}|_{P3,A} - t_{P4C,H,l}|_{P3,A} |, \pi_{P3,H}, \pi_{P4C,H}) \quad (5.120)$$

5.7.16. Relative skew of the local-time reset for a set including processes P3 and P4

Let $\pi_{P3+P4,H}$ denote the bound on the relative skew of the local-time reset for a node set including all BIU nodes synchronizing with respect to process P3 or P4. With respect to process P3, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. In the presence of asymmetric faulty RMU nodes, we know that the time interval of the Accept outputs in process P4 at the trustworthy BIU nodes is contained within the time interval of the Accept outputs in process P4C at the good recovering BIU nodes. Therefore:

$$\pi_{P3+P4,H} \leq \pi_{P3+P4C,H} \quad (5.121)$$

5.7.17. Relative skew of the local-time reset for a set including processes P3C and P4C

Let $t_{P3C,H,l}|_{P2,A}$ denote the earliest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3C,H,l}|_{P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad (5.122)$$

$t_{P3C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering RMU node synchronizing with respect to process P3C resets its local-time clock, measured with respect to the Accept outputs in process P2.

$$t_{P3C,H,h}|_{P2,A} = t_{P2,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + H_{P3}) \quad (5.123)$$

$t_{P4C,H,l}|_{P2,A}$ denotes the earliest real time at which a good BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,l}|_{P2,A} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (5.124)$$

$t_{P4C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,h}|_{P2,A} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad (5.125)$$

$\pi_{P3C+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all good recovering nodes synchronizing with respect to process P3C or P4C.

$$\pi_{P3C+P4C,H} = \max(|\, t_{P4C,H,h}|_{P2,A} - t_{P3C,H,l}|_{P2,A}\,|, |\, t_{P3C,H,h}|_{P2,A} - t_{P4C,H,l}|_{P2,A}\,|, \pi_{P3C,H}, \pi_{P4C,H}) \quad (5.126)$$

5.7.18. Relative skew of the local-time reset for a set including processes P4 and P4C

Let $\pi_{P4+P4C,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering BIUs synchronizing with respect to process P4 or P4C. With respect to process P3, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. This time range is determined by the time range during which the trustworthy RMU nodes executing process P3 send their ECHO messages. In the presence of asymmetric faulty RMU nodes, the good recovering BIUs executing process P4C may not able to synchronize any better than the duration of this time range.

$$\pi_{P4+P4C,H} = \max(\pi_{P4,H}\,, \pi_{P4C,H}) = \pi_{P4C,H} \quad (5.127)$$

5.7.19. Relative skew of the local-time reset for a set including all the synchronizing nodes

Let $\pi_{ALL,H}$ denote the upper bound on the relative skew of the local-time reset for all the trustworthy or good recovering nodes executing the synchronization protocol. The following relations allow us to reduce the number of relative skews that must be considered:

$$\pi_{P2+P3C,H} \geq \pi_{P2,H} \text{ and } \pi_{P2+P3C,H} \geq \pi_{P3C,H} \quad (5.128)$$

$$\pi_{P2+P3C,H} \geq \pi_{P2+P3,H} \quad (5.129)$$

$$\pi_{P2+P4C,H} \geq \pi_{P2,H} \text{ and } \pi_{P2+P4C,H} \geq \pi_{P4C,H} \quad (5.130)$$

$$\pi_{P2+P4C,H} \geq \pi_{P2+P4,H} \quad (5.131)$$

$$\pi_{P3+P4C,H} \geq \pi_{P3,H} \text{ and } \pi_{P3+P4C,H} \geq \pi_{P4C,H} \quad (5.132)$$

$$\pi_{P3+P4C,H} \geq \pi_{P3+P4,H} \quad (5.133)$$

$$\pi_{P3C+P4C,H} \geq \pi_{P3C,H} \text{ and } \pi_{P3C+P4C,H} \geq \pi_{P4C,H} \quad (5.134)$$

So:

$$\pi_{ALL,H} = \max(\pi_{P2+P3C,H}, \pi_{P2+P4C,H}, \pi_{P3+P4C,H}, \pi_{P3C+P4C,H}) \quad (5.135)$$

5.8. Relative local-time skews for source-receiver pairs

5.8.1. Duration of the synchronization protocol execution

From global perspective, the execution of the synchronization protocol ends when all the trustworthy and good recovering nodes have reset their local-time clocks. $t_{sync,l}$ and $t_{sync,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node begins to execute the synchronization protocol. π_{P0} denotes the bound on the relative local-time skew for the trustworthy BIU nodes executing process P0.

$$\pi_{P0} = t_{sync,h} - t_{sync,l} \tag{5.136}$$

$t_{sync,P2,H,l}$ and $t_{sync,P2,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{sync,P2,H,l} = t_{sync,l} + 2r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2})/(1 + \rho_0) \tag{5.137}$$

$$t_{sync,P2,H,h} = t_{sync,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2}) \tag{5.138}$$

$t_{sync,P3,H,l}$ and $t_{sync,P3,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock.

$$t_{sync,P3,H,l} = t_{sync,l} + 3r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \tag{5.139}$$

$$t_{sync,P3,H,h} = t_{sync,h} + 3r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \tag{5.140}$$

$t_{sync,P4,H,l}$ and $t_{sync,P4,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node synchronizing with respect to process P4 can reset its local-time clock.

$$t_{sync,P4,H,l} = t_{sync,l} + 4r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \tag{5.141}$$

$$t_{sync,P4,H,h} = t_{sync,h} + 4r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \tag{5.142}$$

$t_{sync,P3C,H,l}$ and $t_{sync,P3C,H,h}$ denote the earliest and latest times, respectively, at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock.

$$t_{sync,P3C,H,l} = t_{sync,P3,H,l} \tag{5.143}$$

$$t_{sync,P3C,H,h} = t_{sync,P3,H,h} \tag{5.144}$$

$t_{sync,P4C,H,l}$ and $t_{sync,P4C,H,h}$ denote the earliest and latest times, respectively, at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock.

$$t_{sync,P4C,H,l} = t_{sync,P4,H,l} \tag{5.145}$$

$$t_{sync,P4C,H,h} = t_{sync,P4,H,h} \quad (5.146)$$

π_{ALL} denotes the bound on the relative local-time skew for all the nodes participating in the execution of the synchronization protocol. The calculation of π_{ALL} does not include the nodes executing the synchronization-capture processes. $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$ denote lower and upper bounds, respectively, on the real-time duration of the execution of the synchronization protocol for the trustworthy nodes. $\delta_{sync}|_{min}$ is measured from the latest time at which a trustworthy node begins to execute the protocol to the earliest time at which a trustworthy node resets its local-time clock. We choose the following value for $\delta_{sync}|_{min}$:

$$\delta_{sync}|_{min} = -(\pi_{ALL} - \pi_{P0}) + \min[(t_{sync,P2,H,l} - t_{sync,h}), (t_{sync,P3,H,l} - t_{sync,h}), (t_{sync,P4,H,l} - t_{sync,h})] \quad (5.147)$$

$\delta_{sync}|_{max}$ is measured from the earliest time at which a trustworthy node begins to execute the protocol to the latest time at which a trustworthy node resets its local-time clock. We choose the following value for $\delta_{sync}|_{max}$:

$$\delta_{sync}|_{max} = (\pi_{ALL} - \pi_{P0}) + \max[(t_{sync,P2,H,h} - t_{sync,l}), (t_{sync,P3,H,h} - t_{sync,l}), (t_{sync,P4,H,h} - t_{sync,l})] \quad (5.148)$$

We define the following variables in order to simplify these expressions for $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$.

$$\Delta_{sync,P2,H,l} = 2r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2})/(1 + \rho_0) \quad (5.149)$$

$$\Delta_{sync,P3,H,l} = 3r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad (5.150)$$

$$\Delta_{sync,P4,H,l} = 4r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (5.151)$$

$$\Delta_{sync,P2,H,h} = 2r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2}) \quad (5.152)$$

$$\Delta_{sync,P3,H,h} = 3r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \quad (5.153)$$

$$\Delta_{sync,P4,H,h} = 4r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad (5.154)$$

Then:

$$\delta_{sync}|_{min} = -\pi_{ALL} + \min(\Delta_{sync,P2,H,l}, \Delta_{sync,P3,H,l}, \Delta_{sync,P4,H,l}) \quad (5.155)$$

$$\delta_{sync}|_{max} = \pi_{ALL} + \max(\Delta_{sync,P2,H,h}, \Delta_{sync,P3,H,h}, \Delta_{sync,P4,H,h}) \quad (5.156)$$

5.8.2. Bounds on the resynchronization period

Let $\delta_{SP}|_{min}$ and $\delta_{SP}|_{max}$ denote the values of $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$, respectively, for the Synchronization Preservation protocol. T_{SP} denotes the scheduled local time to begin the execution of the Synchronization Preservation protocol. p_{min} denotes a lower bound on the real-time duration of a synchronization cycle. p_{min} is measured from the time of the synchronization reset in one cycle to the time of the synchronization reset in the next.

$$p_{min} = T_{SP}/(1 + \rho_0) + \delta_{SP}|_{min} \quad (5.157)$$

p_{max} denotes an upper bound for the real-time duration of a synchronization cycle. p_{max} is measured from the time of the synchronization reset in one cycle to the time of the synchronization reset in the next.

$$p_{max} = (1 + \rho_0)T_{SP} + \delta_{SP}|_{max} \tag{5.158}$$

P denotes the nominal resynchronization period for the analysis of relative skews. P is measured in units of local-clock ticks. We want a count of P local-clock ticks to be at least as large as the maximum duration of a synchronization cycle measured in nominal ticks. This constraint is captured by the following expression:

$$P/(1 + \rho_0) \geq p_{max} \tag{5.159}$$

So:

$$P \geq (1 + \rho_0)p_{max} \tag{5.160}$$

We choose P to be the smallest integer that satisfies the previous inequality.

$$P = \lceil (1 + \rho_0)p_{max} \rceil \tag{5.161}$$

5.8.3. Relative skew between P2-synchronized BIUs and P3- or P3C-synchronized RMUs

Let $\pi_{P2\text{-}P3}$ denote the bound on the relative local-time skew during the synchronization cycle for trustworthy BIU nodes synchronized with respect to process P2 and trustworthy RMU nodes synchronized with respect to process P3.

$$\pi_{P2\text{-}P3} = \pi_{P2\text{-}P3,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \tag{5.162}$$

$\pi_{P2\text{-}P3C}$ denotes the bound on the relative local-time skew during the synchronization cycle for trustworthy BIU nodes synchronized with respect to process P2 and good recovering RMU nodes synchronized with respect to process P3C. $\pi_{P2\text{-}P3}$ also applies here.

$$\pi_{P2\text{-}P3C} = \pi_{P2\text{-}P3} \tag{5.163}$$

5.8.4. Relative skew between P3-synchronized RMUs and P4- or P4C-synchronized BIUs

Let $\pi_{P3\text{-}P4}$ denote the bound on the relative local-time skew during the synchronization cycle for trustworthy RMU nodes synchronized with respect to process P3 and trustworthy BIU nodes synchronized with respect to process P4. Then:

$$\pi_{P3\text{-}P4} = \pi_{P3\text{-}P4,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \tag{5.164}$$

$\pi_{P3\text{-}P4C}$ denotes the bound on the relative local-time skew during the synchronization cycle for trustworthy RMU nodes synchronized with respect to process P3 and good recovering BIU nodes synchronized with respect to process P4C. $\pi_{P3\text{-}P4}$ also applies here.

$$\pi_{P3\text{-}P4C} = \pi_{P3\text{-}P4} \tag{5.165}$$

5.8.5. Bound on the relative local-time skew for all the nodes executing the synchronization protocol

$\pi_{SP,ALL}$ denotes the value of π_{ALL} for Synchronization Preservation.

$$\pi_{SP,ALL} = \pi_{ALL,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \quad (5.166)$$

5.8.6. Generic relative local-time skew between sources and receivers for synchronous communication

For synchronized operations, we would like to use a single value of the relative local-time skew between sources and receivers for all point-to-point communication. $\pi_{PP,SR}$ denotes the common bound on the relative local-time skew between sources and receivers for synchronized communication. From the preceding analysis, there are only two particular source-receiver cases that need to be considered to determine a common skew bound: the skew between P2-synchronized nodes and P3-synchronized nodes (i.e., $\pi_{P2\text{-}P3}$), and the skew between P3-synchronized nodes and P4-synchronized nodes (i.e., $\pi_{P3\text{-}P4}$). We choose $\pi_{PP,SR}$ to be the largest of the two.

$$\pi_{PP,SR} = \max(\pi_{P2\text{-}P3}, \pi_{P3\text{-}P4}) \quad (5.167)$$

5.9. Specifying the Computation Process and Send Process delays

A goal of this ROBUS version is to achieve nearly the same tightness for the relative local-time skew when executing the Synchronization Preservation, Initial Synchronization, and Synchronization Capture protocols.

The Synchronization Preservation and Initial Synchronization protocols can be decomposed into two major phases: agreement generation and agreement propagation. The agreement generation phase includes the first two stages of the protocol from the Send Process in P0 to the Computation Process in P2. In this phase, the relative skew goes from a bounded initial value denoted by π_{P0} to a relative skew of the Accept outputs denoted by $\pi_{P2,A}$, which is independent of π_{P0} but dependent on the process delays. The agreement propagation phase includes the last two stages of the protocol from the Send Process in P2 to the Computation Process in P4, including the Computation Processes in P3C and P4C for the Synchronization Capture protocol. The synchronization-reset delays are applied with respect to the Accept outputs in processes P2, P3, P3C, P4, and P4C. The process delays for this second phase of the protocol are important determinants of the final relative local-time skew.

The approach taken to determine the process delays for the synchronization protocols in this version of the ROBUS is as follows. Since we expect the value of π_{P0} to be different for the Synchronization Preservation and the Initial Synchronization protocols, we specify the Send Process delay for process P0 (i.e., B_{P0}) independently for each protocol according to the particular timing requirements of the protocol. To ensure that all the versions of the protocol achieve approximately the same relative skew, we compute one set of Computation Process and Send Process delays for the synchronization processes from P1 on. These delays must be used by all the synchronization protocols.

An additional consideration is the constraint on the minimum data-introduction interval (DII) for the send port of the Communication Module, Λ_{Comm}. This constraint applies to the BIUs and the RMUs, and

is satisfied by adding functional requirements to the Send Processes at the BIUs and the RMUs. The details are described next.

5.9.1. Computation Process delays

The Computation Process delay is decomposed into two parts: the reception delay in the Reception Process and the computation delay in the Accept Process. For the case of the Synchronization Preservation protocol, the reception delay is the delay allocated to ensure that all valid messages are received before the computation begins. This delay is similar to the deskewing window applied for the synchronous point-to-point communication. A fundamental difference between the reception delay for the synchronization protocols and the deskewing window for the synchronous protocols is that in the synchronization protocols the relative time spacing between received messages is preserved when forwarding the messages to the computation, while in the synchronous protocols the messages are accumulated and forwarded at the same time.

To specify the reception delay, we consider the timing of reception in the Synchronization Preservation protocol. The timing of reception in the Initial Synchronization protocol is not considered because in that protocol the uncertainty in the time of reception can be extremely large, especially for processes P1 and P2, which would result in very large delays for the protocol. Having a quick execution is very important for the Synchronization Preservation protocol since the duration of the protocol determines how much time is available to execute the synchronous protocols for a given resynchronization period.

Let $A_{IS,P1}$, $A_{IS,P2}$, $A_{IS,P3}$, and $A_{IS,P4}$ denote the Computation Process delays for processes P1, P2, P3, and P4 of the Initial Synchronization protocol, respectively. $A_{SP,P1}$, $A_{SP,P2}$, $A_{SP,P3}$, and $A_{SP,P4}$ denote the Computation Process delays for processes P1, P2, P3, and P4 of the Synchronization Preservation protocol, respectively. $A_{SC,P3C}$ and $A_{SC,P4C}$ denote the Computation Process delays for processes P3C and P4C of the Synchronization Capture protocol, respectively. All the synchronization protocols have the same Computation Process delays.

$$A_{P1} = A_{IS,P1} = A_{SP,P1} \tag{5.168}$$

$$A_{P2} = A_{IS,P2} = A_{SP,P2} \tag{5.169}$$

$$A_{P3} = A_{IS,P3} = A_{SP,P3} = A_{SP,P3C} \tag{5.170}$$

$$A_{P4} = A_{IS,P4} = A_{SP,P4} = A_{SP,P4C} \tag{5.171}$$

For process P1 of the Synchronization Preservation protocol, the expected time range of reception is as follows (This interval includes all the clock edges at which valid messages can arrive.):

$$[T_{SP,P1,RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max},\ T_{SP,P1,RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max}\ ] \tag{5.172}$$

$W_{SP,P1}$ denotes the reception delay applied in process P1. For the Synchronization Preservation protocol, this delay must be large enough to ensure that the Accept Process receives the messages after the clock edges during which valid messages are expected to arrive.

$$W_{SP,P1} = 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1 \tag{5.173}$$

$W_{SP,P2}$, $W_{SP,P3}$, and $W_{SP,P4}$ are similarly defined. Let $\Delta_{SP,P2,RCV}|_{max}$, $\Delta_{SP,P3,RCV}|_{max}$, and $\Delta_{SP,P4,RCV}|_{max}$ denote the maximum valid local time error for the time of reception of synchronization messages in processes P2, P3, and P4 of the Synchronization Preservation protocol. These variables correspond to $\Delta_{P2,RCV}|_{max}$, $\Delta_{P3,RCV}|_{max}$, and $\Delta_{P4,RCV}|_{max}$ evaluated for the case of the Synchronization Preservation protocol. Then:

$$W_{SP,P2} = 2\Delta_{SP,P2,RCV}|_{max} + 1 \tag{5.174}$$

$$W_{SP,P3} = 2\Delta_{SP,P3,RCV}|_{max} + 1 \tag{5.175}$$

$$W_{SP,P4} = 2\Delta_{SP,P4,RCV}|_{max} + 1 \tag{5.176}$$

Notice that for process P1 the expected time of reception is exactly $\Delta_{SP,P1,RCV}|_{max}$ ticks from the left edge of the reception interval in that process. Similar observations apply to processes P2 through P4. The computation delay is measured from the time the message to be selected is presented to the Accept Process until the Accept output is asserted. $C_{SP,P1}$, $C_{SP,P2}$, $C_{SP,P3}$, and $C_{SP,P4}$ denote the Accept Process delays for processes P1, P2, P3, and P4, respectively, of the Synchronization Preservation protocol. These delays also apply to the Initial Synchronization and Synchronization Capture protocols. Then:

$$A_{P1} = W_{SP,P1} + C_{SP,P1} \tag{5.177}$$

$$A_{P2} = W_{SP,P2} + C_{SP,P2} \tag{5.178}$$

$$A_{P3} = W_{SP,P3} + C_{SP,P3} \tag{5.179}$$

$$A_{P4} = W_{SP,P4} + C_{SP,P4} \tag{5.180}$$

5.9.2. Send Process delays

The Send Process delays must be set to ensure proper inter-process communication. The Send Process delay for process P0 does not need to be the same for Initial Synchronization and Synchronization Preservation. The specification of that value for each protocol is presented below. For all the other Send Processes, we specify the delays based on two factors. First, we would like to specify the process delays based on the execution of the Synchronization Preservation protocol. The timing of execution of the Initial Synchronization protocol is not preferred because in that protocol the uncertainty in the time of reception can be extremely large, which would result in extremely large process delays for the protocol. The second factor when specifying the Send Process delays is the need to satisfy the minimum data-introduction-interval constraint for the send port of the Communication Module, Λ_{Comm}, which must be satisfied at the BIUs and the RMUs.

For the execution of the Synchronization Preservation protocol, the main concern in specifying the Send Process delays is ensuring proper coordination between the send and receive operations. In particular, the specification of the send delay must take into consideration the expected reception delay, the minimum delays in opening the input windows, and the size of the input windows. This is not a consideration in the Initial Synchronization protocol since in that case all the Computation Processes are enabled at the beginning of the execution of the protocol.

For the Synchronization Preservation protocol, we must ensure that the time separation between the sending of INIT and ECHO messages satisfies the Λ_{Comm} constraint. The preferred method to satisfy this

constraint in the Synchronization Preservation protocol is to increase the Send Process delays for the INIT messages in processes P0 and P1 and/or the ECHO messages in processes P2 and P3 until sufficient separation between them is ensured.

For the Initial Synchronization protocol, the problem is more complicated. Because the initial relative local-time skew can be much larger than the Computation Process and Send Process delays, there is no way to meet the Λ_{Comm} constraint by simply changing the process delays while still achieving the other design goals. The preferred solution for this case is to add functionality to the Send Processes at the BIUs and the RMUs to force a minimum separation between INIT and ECHO messages. However, the buffering of synchronization messages for a bounded but unspecified amount of time at a Send Process is an undesired solution because it would result in an increase in the bound on the relative local-time skew achieved by the protocol. Instead, the solution is based on the observation that for the Initial Synchronization protocol, once the Computation Process of a node has performed the computation that triggers the sending of an ECHO message (i.e., Accept(INIT) in process P2 at the BIUs, and Accept(ECHO) in process P3 at the RMUs), there is no need for the node to send an INIT message. To understand this, notice that the synchronization protocol achieves synchronization in process P2, and this is then propagated to processes P3 and P4 using ECHO messages. For the Initial Synchronization protocol, RMUs and BIUs reset their local times with respect to the Accept(ECHO) outputs in processes P3 and P4, respectively. Therefore, the fact that an ECHO message is going to be sent means that whatever critical timing information was going to be provided by processes P0 and P1, it has already been received. Therefore, the INIT messages are redundant from that point on. So, for Initial Synchronization, to meet the minimum data-introduction-interval constraint, the Send Process must have the following features:

- The sending of an INIT message must be blocked if the message has not been sent by the time the Accept output that triggers the sending of an ECHO message is asserted.

- The send delay for ECHO messages must be greater than or equal to Λ_{Comm} - 1.

The first functional requirement removes redundant INIT messages. The second requirement ensures that, if an INIT message is sent at or before the tick at which the Accept output that triggers the sending of an ECHO message is asserted, then the ECHO message will be sent at least Λ_{Comm} ticks after the INIT message.

Let $B_{IS,P0}$, $B_{IS,P1}$, $B_{IS,P2}$, and $B_{IS,P3}$ denote the Send Process delays for processes P0, P1, P2, and P3 of the Initial Synchronization protocol, respectively. $B_{SP,P0}$, $B_{SP,P1}$, $B_{SP,P2}$, and $B_{SP,P3}$ denote the Computation Process delay for processes P0, P1, P2, and P3 of the Synchronization Preservation protocol, respectively. For processes P1, P2, and P3:

$$B_{P1} = B_{IS,P1} = B_{SP,P1} \tag{5.181}$$

$$B_{P2} = B_{IS,P2} = B_{SP,P2} \tag{5.182}$$

$$B_{P3} = B_{IS,P3} = B_{SP,P3} \tag{5.183}$$

$B_{IS,P0}$ and $B_{SP,P0}$ are specified separately for Initial Synchronization and Synchronization Preservation.

5.9.2.1. Send delay for process P0

5.9.2.1.1. Synchronization Preservation

The Synchronization Preservation protocol is a time-triggered, event-driven protocol. The communication between processes P0 and P1 follows a time-triggered pattern similar to the point-to-point communication of the synchronous protocols. After that operation, the rest of the Synchronization Preservation protocol proceeds driven by communication and processing events. T_{SP} denotes the local-time trigger for the execution of the Synchronization Preservation protocol. $B_{SP,P0}$ denotes the send delay for process P0 of the Synchronization Preservation protocol. $B_{SP,P0}|_{min}$ denotes the minimum send delay for process P0. $B_{SP,P0}|_{min}$ is assumed to be the time needed to prepare the message for transmission. $B_{SP,P0} - B_{SP,P0}|_{min}$ is additional delay added to align the send and receive operations. $\Delta_{SP,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the reception window in process P1. $\Delta_{SP,P1,RCVWND}|_{min}$ is the minimum value of $\Delta_{SP,P1,RCVWND}$. R_{PP} denotes the expected point-to-point reception delay. $W_{SP,P1}$ is the size of the reception window. $W_{SP,P1,pre}$ is the pre-expectation window (i.e., the size of the section of the reception window before the expected time of reception). Considering the analysis for point-to-point communication presented in a previous section, $W_{SP,P1,pre}$ corresponds to $W_{Deskew,pre}$. So:

$$W_{SP,P1,pre} = \Delta_{PP,RCV}|_{abs\text{-}max.} \tag{5.184}$$

$T_{SP,P0,SND}$ denotes the send time for process P0. $T_{SP,P0,SND}$ corresponds to T_{P0} in the general analysis of the clock synchronization protocols. $T_{SP,P1,RCV,E}$ denotes the expected time of reception for process P1. $T_{SP,P0\text{-}P1,REF}$ denotes the reference time for the transmission between P0 and P1.

$$T_{SP,P0\text{-}P1,REF} = T_{SP} \tag{5.185}$$

Two cases must be considered.

Case 1: $B_{SP,P0}|_{min} + R_{PP} \geq \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre}$

For this case:

$$B_{SP,P0} = B_{SP,P0}|_{min} \tag{5.186}$$

$$\Delta_{SP,P1,RCVWND} = B_{SP,P0}|_{min} + R_{PP} - W_{SP,P1,pre} \tag{5.187}$$

So:

$$T_{SP,P0,SND} = T_{SP,P0\text{-}P1,REF} + B_{SP,P0} = T_{SP} + B_{SP,P0}|_{min} \tag{5.188}$$

And:

$$T_{SP,P1,RCV,E} = T_{SP,P0,SND} + R_{PP} = T_{SP} + B_{SP,P0}|_{min} + R_{PP} \tag{5.189}$$

Case 2: $B_{SP,P0}|_{min} + R_{PP} < \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre}$

For this case:

$$B_{SP,P0} = \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} - R_{PP} \quad (5.190)$$

$$\Delta_{SP,P1,RCVWND} = \Delta_{SP,P1,RCVWND}|_{min} \quad (5.191)$$

So:

$$T_{SP,P0,SND} = T_{SP,P0\text{-}P1,REF} + B_{SP,P0} = T_{SP} + \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} - R_{PP} \quad (5.192)$$

And:

$$T_{SP,P1,RCV,E} = T_{SP,P0,SND} + R_{PP} = T_{SP} + \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} \quad (5.193)$$

5.9.2.1.2. Initial Synchronization

Let π_{IS} denote the bound on the relative local-time skew considering BIUs and RMUs during the execution of the Initial Synchronization protocol, measured in nominal clock ticks. T_{IS} denotes the local time triggering the execution of the Initial Synchronization protocol. The timing of the first-stage communication can be analyzed similarly to the point-to-point communication for synchronous protocols.

$T_{IS,P0\text{-}P1,REF}$ denotes the reference time for the communication between processes P0 and P1. $T_{IS,P0,SND}$ denotes the local time at which process P0 sends the message. $T_{IS,P0,SND}$ corresponds to T_{P0} in the general analysis of the clock synchronization protocols. $T_{IS,P1,RCV,E}$ denotes the expected time of reception in process P1. $B_{IS,P0}$ denotes the Send Process delay for process P0. $B_{IS,P0}|_{min}$ denotes the minimum send delay for process P0. $\Delta_{IS,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the reception window in process P1. $W_{IS,P1}$ denotes the size of the reception window in process P1. $W_{IS,P1,pre}$ denotes the pre-expectation window in process P1 (i.e., the size of the section of the reception window before the expected time of reception). We use T_{IS} as the reference time for the communication between processes P0 and P1.

$$T_{IS,P0\text{-}P1,REF} = T_{IS} \quad (5.194)$$

We use the analysis for point-to-point communication to determine $W_{IS,P1,pre}$. To determine $W_{IS,P1}$, we need the maximum error in the expected time of reception for the Initial Synchronization protocol messages, $\Delta_{IS,PP,RCV}|_{abs\text{-}max}$.

$$\Delta_{IS,PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{IS} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad (5.195)$$

$\mu_{PP,l}$ and $\mu_{PP,h}$ are given in the Section 4. So, for the reception window:

$$W_{IS,P1} = 2\Delta_{IS,PP,RCV}|_{abs\text{-}max} + 1 \quad (5.196)$$

$$W_{IS,P1,pre} = \Delta_{IS,PP,RCV}|_{abs\text{-}max} \quad (5.197)$$

$B_{IS,P0}|_{min}$ is assumed to be the time needed to prepare the message for transmission.

We expect the upper bound on the relative local-time skew during the execution of the first stage of the Initial Synchronization protocol to be much larger than any minimum timing constraints associated with the process of communication. Based on this, we assume that the following condition holds for the communication between processes P0 and P1.

$$B_{IS,P0}|_{min} + R_{PP} < \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} \quad (5.198)$$

For this case:

$$B_{IS,P0} = \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} - R_{PP} \quad (5.199)$$

$$\Delta_{IS,P1,RCVWND} = \Delta_{IS,P1,RCVWND}|_{min} \quad (5.200)$$

So:

$$T_{IS,P0,SND} = T_{IS,P0\text{-}P1,REF} + B_{IS,P0} = T_{IS} + \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} - R_{PP} \quad (5.201)$$

And:

$$T_{IS,P1,RCV,E} = T_{IS,P0,SND} + R_{PP} = T_{IS} + \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} \quad (5.202)$$

5.9.2.2. Send delay for process P1

B_{P1} is specified based on timing considerations for Synchronization Preservation. $B_{P1}|_{min}$ is determined by the implementation. Process P1 sends INIT to process P2. However, the reference event used to coordinate the communication between processes P1 and P2 is the trigger time for the transmission of the message in process P0. Let $T_{SP,P0\text{-}P2,REF}$ denote this reference.

$$T_{SP,P0\text{-}P2,REF} = T_{SP} + B_{SP,P0} \quad (5.203)$$

$R_{SP,P0\text{-}P2}$ denotes the expected reception delay for process P2 of the Synchronization Preservation protocol. $R_{SP,P0\text{-}P2}$ is measured from the send time in process P0 to the expected time of reception in process P2. $\Delta_{SP,P2,RCVWND}$ denotes the delay from the reference time to the opening of the input window in process P2. $\Delta_{SP,P2,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P0\text{-}P2} = \Delta_{SP,P2,RCVWND} + \Delta_{SP,P2,RCV}|_{max} \quad (5.204)$$

Here, both $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ are functions of B_{P1}, and $\Delta_{SP,P2,RCVWND}$ can be made larger than $\Delta_{SP,P2,RCVWND}|_{min}$. Solving this equation for B_{P1} is not trivial. However, note that $R_{SP,P0\text{-}P2}$ varies one-to-one with respect to B_{P1}, while $\Delta_{SP,P2,RCV}|_{max}$ changes by approximately $2\rho_0 B_{P1}$ for each unit step in B_{P1}. This observation allows us to use the following algorithm to determine B_{P1}. The notation $R_{SP,P0\text{-}P2}(B_{P1})$ and $\Delta_{SP,P2,RCV}|_{max}(B_{P1})$ highlights the dependence of $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ on B_{P1}.

1. $B_{P1} = B_{P1}|_{min}$
2. while $[R_{SP,P0\text{-}P2}(B_{P1}) < \Delta_{SP,P2,RCV}|_{max}(B_{P1}) + \Delta_{SP,P2,RCVWND}|_{min}]$
3. $\{B_{P1} = B_{P1} + 1\}$

4. Results:
5. B_{P1}
6. $R_{SP,P0\text{-}P2} = R_{SP,P0\text{-}P2}(B_{P1})$
7. $\Delta_{SP,P2,RCV}|_{max} = \Delta_{SP,P2,RCV}|_{max}(B_{P1})$
8. $\Delta_{SP,P2,RCVWND} = R_{SP,P0\text{-}P2} - \Delta_{SP,P2,RCV}|_{max}$

5.9.2.3. Send delay for process P2

B_{P2} is specified based on timing considerations for Synchronization Preservation and the minimum data-introduction-interval constraint of the Communication Module. $B_{P2}|_{min}$ is determined by the implementation. $T_{SP,P1\text{-}P3,REF}$ denotes the reference time for the communication message propagation from P1 to P3. The event used to coordinate this communication is the time of the Accept output in process P1, denoted by $T_{SP,P1,A}$ for the Synchronization Preservation protocol.

$$T_{SP,P1\text{-}P3,REF} = T_{SP,P1,A} \tag{5.205}$$

$R_{SP,P1\text{-}P3}$ denotes the expected reception delay for process P3 of the Synchronization Preservation protocol. $R_{SP,P1\text{-}P3}$ is measured from the time of Accept output in process P1 to the expected time of reception in process P3. $\Delta_{SP,P3,RCVWND}$ denotes the delay from the reference time to the opening of the input window in process P3. $\Delta_{SP,P3,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P1\text{-}P3} = \Delta_{SP,P3,RCVWND} + \Delta_{SP,P3,RCV}|_{max} \tag{5.206}$$

As for the case of B_{P1}, solving this equation for B_{P2} is non-trivial. Therefore, we use here the same algorithm used to solve for B_{P1}. An additional constraint is that B_{P2} must be greater than or equal to Λ_{Comm} - 1.

1. $B_{P2} = B_{P2}|_{min}$
2. if ($B_{P2} < \Lambda_{Comm} - 1$), then $B_{P2} = \Lambda_{Comm} - 1$.
2. while $[R_{SP,P1\text{-}P3}(B_{P2}) < \Delta_{SP,P3,RCV}|_{max}(B_{P2}) + \Delta_{SP,P3,RCVWND}|_{min}]$
3. $\{B_{P2} = B_{P2} + 1\}$

4. Results:
5. B_{P2}
6. $R_{SP,P1\text{-}P3} = R_{SP,P1\text{-}P3}(B_{P2})$
7. $\Delta_{SP,P3,RCV}|_{max} = \Delta_{SP,P3,RCV}|_{max}(B_{P2})$
8. $\Delta_{SP,P3,RCVWND} = R_{SP,P1\text{-}P3} - \Delta_{SP,P3,RCV}|_{max}$

5.9.2.4. Send delay for process P3

B_{P3} is specified based on timing considerations for Synchronization Preservation and the minimum data-introduction-interval constraint of the Communication Module. $B_{P3}|_{min}$ is determined by the implementation. $T_{SP,P2\text{-}P4,REF}$ denotes the reference time for the communication message propagation from P2 to P4. The event used to coordinate this communication is the time of the Accept output in process P1, denoted by $T_{SP,P2,A}$ for the Synchronization Preservation protocol.

$$T_{SP,P2\text{-}P4,REF} = T_{SP,P2,A} \tag{5.207}$$

$R_{SP,P2\text{-}P4}$ denotes the expected reception delay for process P4 of the Synchronization Preservation protocol. $R_{SP,P2\text{-}P4}$ is measured from the time of Accept in process P2 to the expected time of reception in process P4. Let $\Delta_{SP,P4,RCVWND}$ denote the delay from the reference time to the opening of the input window in process P4. $\Delta_{SP,P4,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P2\text{-}P4} = \Delta_{SP,P4,RCVWND} + \Delta_{SP,P4,RCV}|_{max} \quad (5.208)$$

As for the case of B_{P2}, solving this equation for B_{P3} is non-trivial. Therefore, we use here the same algorithm used to solve for B_{P2}. An additional constraint is that B_{P3} must be greater than or equal to $\Lambda_{Comm} - 1$.

1. $B_{P3} = B_{P3}|_{min}$
2. if ($B_{P3} < \Lambda_{Comm} - 1$), then $B_{P3} = \Lambda_{Comm} - 1$.
2. while $[R_{SP,P2\text{-}P4}(B_{P3}) < \Delta_{SP,P4,RCV}|_{max}(B_{P3}) + \Delta_{SP,P4,RCVWND}|_{min}]$
3. $\{B_{P3} = B_{P3} + 1\}$

4. Results:
5. B_{P3}
6. $R_{SP,P2\text{-}P4} = R_{SP,P2\text{-}P4}(B_{P3})$
7. $\Delta_{SP,P4,RCV}|_{max} = \Delta_{SP,P4,RCV}|_{max}(B_{P3})$
8. $\Delta_{SP,P4,RCVWND} = R_{SP,P2\text{-}P4} - \Delta_{SP,P4,RCV}|_{max}$

5.10. Additional considerations

5.10.1. Frame Synchronization

The Frame Synchronization protocol is executed by recovering nodes in the Synchronization Acquisition mode. The end of the Frame Synchronization protocol triggers the execution of the P3C or P4C synchronization-capture processes. An assumption for the protocol is the existence of a single valid clique in Preservation mode. The protocol monitors the ECHO messages from the trusted nodes identified during Local Diagnosis Acquisition. Achieving frame synchronization is equivalent to finding the time gap between consecutive executions of the Synchronization Preservation protocol. The Frame Synchronization protocol consists of searching for a time interval during which the clique is not sending ECHO messages. Finding such interval indicates that the clique is in between computations of clock adjustments, and thus it is an appropriate time to start the execution of the Synchronization Capture protocol. The Frame Synchronization protocol can achieve synchronization even if, for the node executing the protocol, it is not true that a majority of the eligible sources of the opposite kind is trustworthy. The time interval measured by the gap timer, called the **frame synchronization gap**, corresponds to the maximum observed relative skew between received ECHO messages from trustworthy nodes. The analysis presented here applies to BIUs and RMUs.

Let $\Delta_{FS,GAP}$ denote the duration of the frame synchronization gap, measured in local clock ticks. $\Delta_{FS,GAP}|_{RMU}$ and $\Delta_{FS,GAP}|_{BIU}$ correspond to $\Delta_{FS,GAP}$ for RMUs and BIUs, respectively.

$$\Delta_{FS,GAP}|_{RMU} = \Pi_{SP,P3C,RCV} \quad (5.209)$$

$$\Delta_{FS,GAP}|_{BIU} = \Pi_{SP,P4C,RCV} \quad (5.210)$$

We choose $\Delta_{FS,GAP}|_{max}$ to be the largest value of $\Delta_{FS,GAP}$.

$$\Delta_{FS,GAP}|_{max} \geq \max(\Pi_{SP,P3C,RCV}, \Pi_{SP,P4C,RCV}) \quad (5.211)$$

We are interested in the worst-case duration of the Frame Synchronization protocol. Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in local clock ticks. To determine the maximum duration of the Frame Synchronization protocol, we need to consider the possible patterns of interruption of the interval timer.

N denotes the total number of BIU nodes, and M denotes the total number of RMUs nodes. Let ω denote the number of eligible sources of the opposite kind.

$$\omega|_{max} = \max(N, M) \tag{5.212}$$

Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol, measured in local-clock ticks. An assumption for the Frame Synchronization protocol is that, during its execution, it will encounter at most one execution of the Synchronization Preservation protocol. Therefore, a source is allowed to interrupt the interval timer at most once during the execution of the protocol. In the worst-case, interruptions from eligible sources can consume up to $\omega|_{max} * \Delta_{FS,GAP}|_{max}$ local ticks in failed attempts to find a quiet frame synchronization gap (i.e., an interval with no gap timer interruptions). Adding an additional $\Delta_{FS,GAP}$ for the last interval, for which interruptions would not be allowed, then:

$$\Delta_{FS}|_{max} = (\omega|_{max} + 1) * \Delta_{FS,GAP}|_{max} \tag{5.213}$$

δ_{FS} denotes the worst-case duration of the Frame Synchronization protocol measured in nominal clock ticks.

$$\delta_{FS}|_{max} = (1 + \rho_0)\Delta_{FS}|_{max} \tag{5.214}$$

The assumption that, during its execution, the Frame Synchronization protocol will encounter at most one execution of the Synchronization Preservation protocol imposes the following constraint on the minimum duration of the resynchronization period p_{min}.

$$p_{min} \geq \delta_{FS}|_{max} \tag{5.215}$$

5.10.2. Executing Synchronization Preservation after Synchonization Acquisition

Recovering BIUs synchronize to an existing clique by executing the Synchronization Capture protocol and synchronizing with respect to process P4C. After synchronizing, the recovering BIUs behave synchronously just like the existing BIU members of the clique. In the first execution of the Synchronization Preservation protocol, the recovering BIUs synchronize with respect to process P2. There are two important differences between the existing trustworthy BIU members of a clique and good recovering BIUs during this execution of the Synchronization Preservation protocol.

The first difference is that good recovering BIUs do not transmit synchronization messages. Even if they transmitted messages, the existing members of the clique would not include those messages in the computation of the protocol. Therefore, for the existing trustworthy clique members, the result of the synchronization protocol does not depend on the performance of recovering BIUs.

The second (and more important) difference is that the good recovering BIUs are not necessarily synchronized to the existing trustworthy BIU clique members as tightly as the existing trustworthy BIU clique members are synchronized to each other. This difference is significant in the execution of process

P2 and in the definitions of the expected reception delay $R_{SP,P0\text{-}P2}$ and the worst-case local-time difference between the actual time of reception and the expect time of reception $\Delta_{SP,P2,RCV}|_{max}$. As presented previously, the definition of these parameters uses the relative local-time skew of the transmitting BIUs in process P0 (i.e., π_{P0}). Because good recovering BIUs are not included in the definition of π_{P0}, their effective reception delay and its the worst-case error can be different than for the trustworthy BIU clique members. To correct this problem, the relative local-time skew used to compute $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ must include the good recovering BIUs.

$\pi_{SP,P0}|_{P2,RCV}$ denotes the value of π_{P0} used to compute $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ for process P2 of the Synchronization Preservation protocol.

$$\pi_{SP,P0}|_{P2,RCV} = \pi_{P2+P4C,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \tag{5.216}$$

$\pi_{SP,P0}$ denotes the value of π_{P0} for the Synchronization Preservation protocol. Except for the case above, $\pi_{SP,P0}$ is:

$$\pi_{SP,P0} = \pi_{P2,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \tag{5.217}$$

5.10.3. Time service accuracy for the Synchronization Preservation protocol

The PEs receive periodic time updates from the BIUs in the form of INIT messages. These messages are triggered by the output of the Accept(INIT) functions in process P2 of the Synchronization Preservation protocol. The accuracy of the time service is defined here as the maximum error in the expect period between Accept(INIT) outputs in consecutive executions of the Synchronization Preservation protocol.

Consider two consecutive executions of the Synchronization Preservation protocol, denoted by SP1 and SP2. $\pi_{P2,A}$ denotes the bound on the real-time relative skew of the Accept outputs in process P2 at the trustworthy BIU nodes. $\pi_{P2,A}$ applies to SP1 and SP2. Let $t_{P2,A,l}|_{SP1}$ and $t_{P2,A,h}|_{SP1}$ denote the bounds on the earliest and latest real times, respectively, at which the trustworthy BIU nodes synchronizing with respect to process P2 of SP1 assert the output of their Accept(INIT) functions. Thus:

$$\pi_{P2,A} = t_{P2,A,h}|_{SP1} - t_{P2,A,l}|_{SP1} \tag{5.218}$$

Let $t_{P2,A,l}|_{SP2}$ and $t_{P2,A,h}|_{SP2}$ denote the bounds on the earliest and latest real times, respectively, at which the Accept outputs are asserted in process P2 at the trustworthy BIUs for SP2. The relations between $t_{P2,A,h}|_{SP1}$ and $t_{P2,A,l}|_{SP1}$, and $t_{P2,A,h}|_{SP2} - t_{P2,A,l}|_{SP2}$ are constrained by the drift rate of the local-time clocks and the validity interval for the Accept outputs in process P2 of SP2.

$$t_{P2,A,l}|_{SP2} = t_{P2,A,l}|_{SP1} + 2r_{PP,l} + (H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2})/(1 + \rho_0) \tag{5.219}$$

$$t_{P2,A,h}|_{SP2} = t_{P2,A,h}|_{SP1} + 2r_{PP,h} + (1 + \rho_0)(H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) \tag{5.220}$$

$P_{SVC}|_{min}$ and $P_{SVC}|_{max}$ denote the minimum and maximum intervals, respectively, between time updates for the time-reference service, measured in units of nominal clock ticks.

$$P_{SVC}|_{min} = t_{P2,A,l}|_{SP2} - t_{P2,A,h}|_{SP1}$$

$$= 2r_{PP,l} + (H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2})/(1 + \rho_0) - \pi_{P2,A} \quad (5.221)$$

$P_{SVC}|_{max} = t_{P2,A,h}|_{SP2} - t_{P2,A,l}|_{SP1}$

$$= 2r_{PP,h} + (1 + \rho_0)(H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) + \pi_{P2,A} \quad (5.222)$$

P_{SVC} denotes the expected period between time updates for the time-reference service, measured in units of nominal clock ticks.

$$P_{SVC} = (P_{SVC}|_{min} + P_{SVC}|_{max})/2 \quad (5.223)$$

Let α denote the accuracy of P_{SVC}.

$\alpha = (P_{SVC}|_{max} - P_{SVC}|_{min})/2$

$$= \pi_{P2,A} + e_{PP} + (1/2)[(1 + \rho_0) - 1/(1 + \rho_0)](H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) \quad (5.224)$$

Substituting for $\pi_{P2,A}$:

$$\alpha = 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)][(3/2)(A_{P1} + B_{P1} + A_{P2}) + (1/2)(H_{P2} + T_{SP} + B_{P0})] \quad (5.225)$$

6. Clique Preservation mode

This section examines the protocols executed in the Clique Preservation mode. The timing analysis also applies to the Clique Join mode.

Figure 6.1 illustrates the message exchange pattern between a BIU and its attached PE during the Clique Join and the Clique Preservation modes. The mode and identification messages are sent to the PE between the Self-Diagnosis and Schedule Update services. During Schedule Update, the BIU reads the schedule submitted by its PE and sends to the PE the results determined by the bus for each PE. This is followed by a single message with the assessment of the new schedule. This consists of a SPECIAL message with the payload set to VALID_SCHEDULE, ZERO_SCHEDULE (i.e., the schedule is valid and equal to zero for all the PEs), or INVALID_SCHEDULE. If the schedule is invalid, the ROBUS will automatically switch to a default schedule. During the PE Broadcast service, a BIU reads the scheduled messages from its PE and outputs to the PE the result for all the scheduled messages. A broadcast result equal to PE_ERROR indicates that there was an error at the source PE. If the bus determines that the BIU of a source PE is not operating properly, then the result of the broadcast will be SOURCE_ERROR or NO_MAJORITY. A result of NO_MAJORITY indicates that the RMUs received different messages from the source BIU. If the assessment of the schedule was ZERO_SCHEDULE, the PE Broadcast service is not executed and the ROBUS simply waits until it is time to execute the Time Reference service. During the Time Reference service, a BIU outputs a SPECIAL message with the payload set to INIT. The sending of this message is triggered by the reference event that the BIU will use to reset its local-time clock. (For Initial Synchronization and Synchronization Acquisition, the payload is set to ECHO to explicitly indicate that a different protocol event is used as a reference to reset the local-time clock.) During the Self-Diagnosis service, the output of a BIU consists of two messages containing the diagnostic results for the BIUs and the RMUs. These are the last messages of the cycle. The next messages are the mode and identification messages for the next cycle.

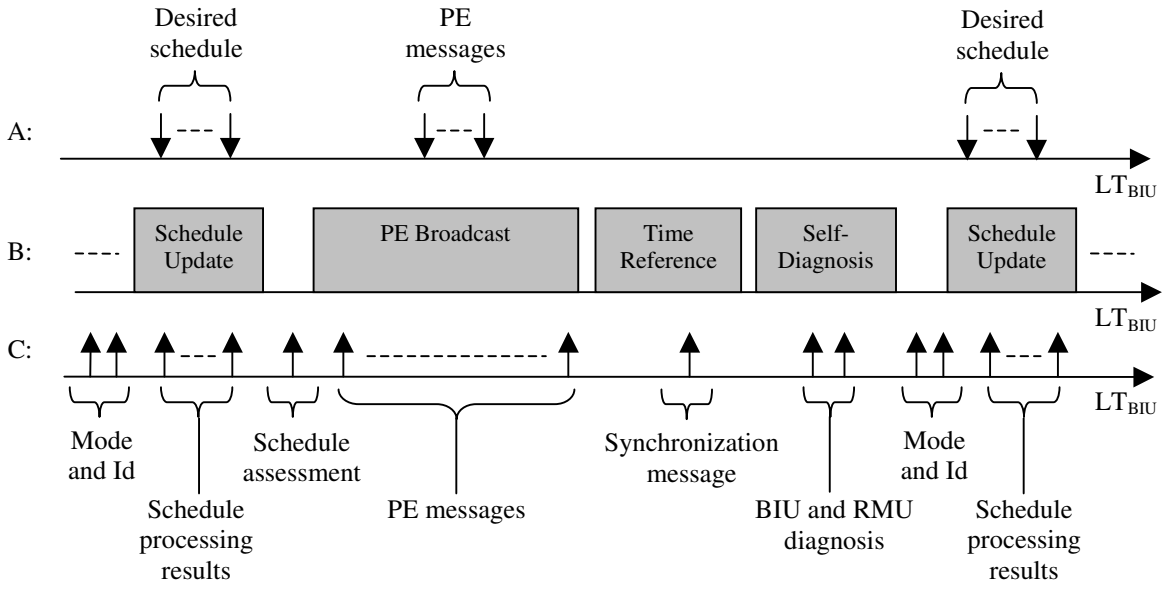


Figure 6.1: Message exchange pattern between a BIU and its attached PE in Clique Preservation mode

A: PE-to-BIU messages, B: Bus services, C: BIU-to-PE messages

6.1. Collective Diagnosis Protocol

Figure 6.2 shows the message flow graph for the Collective Diagnosis protocol. For this protocol, BIUs and RMUs have the same timing characteristics. The analysis presented here does not refer to the kind of the node sending or receiving messages for any of the protocol processes.

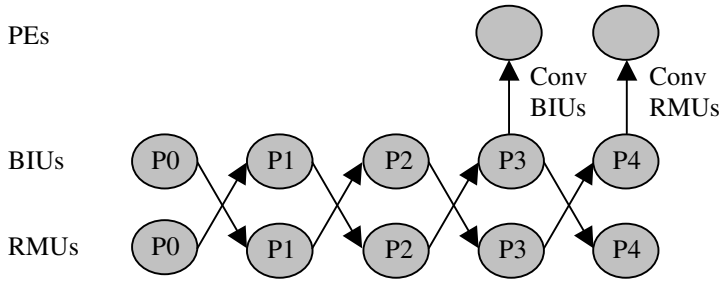


Figure 6.2: Message flow graph for the Collective Diagnosis protocol

6.1.1. Stage 1: P0 to P1

T_{CD} denotes the local-time trigger for the execution of the Collective Diagnosis protocol, R_{PP} denotes the point-to-point expected reception delay, $S_{CD,P0}$ denotes the Send Process delay for process P0, $S_{CD,P0}|_{min}$ denotes the minimum send delay for process P0, $\Delta_{CD,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the reception window in process P1; $\Delta_{CD,P1,RCVWND}|_{min}$ is the minimum value. $T_{CD,P0,SND}$ denotes the send time for process P0, and $T_{CD,P1,RCV,E}$ denotes the expected time of reception for process P1.

6.1.1.1. Communication between processes P0 and P1

Let $T_{CD,P0\text{-}P1,REF}$ denote the reference time for the transmission between P0 and P1.

$$T_{CD,P0\text{-}P1,REF} = T_{CD} \tag{6.1}$$

Case 1: $S_{CD,P0}|_{min} + R_{PP} \geq \Delta_{CD,P1,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P0} = S_{CD,P0}|_{min} \tag{6.2}$$

$$\Delta_{CD,P1,RCVWND} = S_{CD,P0}|_{min} + R_{PP} - W_{Deskew,pre} \tag{6.3}$$

So:

$$T_{CD,P0,SND} = T_{CD,P0\text{-}P1,REF} + S_{CD,P0}$$

$$= T_{CD} + S_{CD,P0}|_{min} \tag{6.4}$$

And :

$$T_{CD,P1,RCV,E} = T_{CD,P0,SND} + R_{PP}$$

$$= T_{CD} + S_{CD,P0}|_{min} + R_{PP} \quad (6.5)$$

Case 2: $S_{CD,P0}|_{min} + R_{PP} < \Delta_{CD,P1,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P0} = \Delta_{CD,P1,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.6)$$

$$\Delta_{CD,P1,RCVWND} = \Delta_{CD,P1,RCVWND}|_{min} \quad (6.7)$$

So:

$$T_{CD,P0,SND} = T_{CD,P0\text{-}P1,REF} + S_{CD,P0}$$

$$= T_{CD} + \Delta_{CD,P1,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.8)$$

And:

$$T_{CD,P1,RCV,E} = T_{CD,P0,SND} + R_{PP}$$

$$= T_{CD} + \Delta_{CD,P1,RCVWND}|_{min} + W_{Deskew,pre} \quad (6.9)$$

6.1.1.2. Computation in process P1

$C_{CD,P1}$ denotes the computation delay in process P1. The computation delay is measured from the end of the deskewing window. $C_{CD,P1}$ is assumed to be a constant independent of the computation input or the system state. $T_{CD,P1,C}$ denotes the local time at which the Computation Process outputs the result for process P1.

$$T_{CD,P1,C} = T_{CD,P1,RCV,E} + W_{Deskew,post} + C_{CD,P1} \quad (6.10)$$

6.1.2. Stage 2: P1 to P2

6.1.2.1. Communication between processes P1 and P2

Let $T_{CD,P1\text{-}P2,REF}$ denote the reference time for the transmission between P1 and P2. We choose the end of the deskewing window in process P1 as the reference time.

$$T_{CD,P1\text{-}P2,REF} = T_{CD,P1,RCV,E} + W_{Deskew,post} \quad (6.11)$$

$S_{CD,P1}$ denotes the Send Process delay for process P1, $\Delta_{CD,P2,RCVWND}$ denotes the delay in opening the reception window in process P2, $T_{CD,P1,SND}$ denotes the send time for process P1, and $T_{CD,P2,RCV,E}$ denotes the expected time of reception for process P2.

We assume that the constraint on the minimum data introduction interval for the Communication

Module is satisfied by the minimum delay between the send times for processes P0 and P1. That is, we assume that the following is true:

$$R_{PP} + W_{Deskew,post} + C_{CD,P1} + S_{CD,P1}|_{min} \geq \Lambda_{Comm} \quad (6.12)$$

Case 1: $C_{CD,P1} + S_{CD,P1}|_{min} + R_{PP} \geq \Delta_{CD,P2,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P1} = S_{CD,P1}|_{min} \quad (6.13)$$

$$\Delta_{CD,P2,RCVWND} = C_{CD,P1} + S_{CD,P1}|_{min} + R_{PP} - W_{Deskew,pre} \quad (6.14)$$

So:

$$T_{CD,P1,SND} = T_{CD,P1\text{-}P2,REF} + C_{CD,P1} + S_{CD,P1}$$

$$= T_{CD,P1,RCV,E} + W_{Deskew,post} + C_{CD,P1} + S_{CD,P1}|_{min} \quad (6.15)$$

And:

$$T_{CD,P2,RCV,E} = T_{CD,P1,SND} + R_{PP}$$

$$= T_{CD,P1,RCV,E} + W_{Deskew,post} + C_{CD,P1} + S_{CD,P1}|_{min} + R_{PP} \quad (6.16)$$

Case 2: $C_{CD,P1} + S_{CD,P1}|_{min} + R_{PP} < \Delta_{CD,P2,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P1} = \Delta_{CD,P2,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} - C_{CD,P1} \quad (6.17)$$

$$\Delta_{CD,P2,RCVWND} = \Delta_{CD,P2,RCVWND}|_{min} \quad (6.18)$$

So:

$$T_{CD,P1,SND} = T_{CD,P1\text{-}P2,REF} + C_{CD,P1} + S_{CD,P1}$$

$$= T_{CD,P1,RCV,E} + W_{Deskew,post} + \Delta_{CD,P2,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.19)$$

And:

$$T_{CD,P2,RCV,E} = T_{CD,P1,SND} + R_{PP}$$

$$= T_{CD,P1,RCV,E} + W_{Deskew,post} + \Delta_{CD,P2,RCVWND}|_{min} + W_{Deskew,pre} \quad (6.20)$$

6.1.2.2. Computation in process P2

$C_{CD,P2}$ denotes the computation delay in process P2. The computation delay is measured from the end of the deskewing window. $C_{CD,P2}$ is assumed to be a constant independent of the computation input and

the system state. $T_{CD,P2,C}$ denotes the local time at which the Computation Process outputs the result for process P2.

$$T_{CD,P2,C} = T_{CD,P2,RCV,E} + W_{Deskew,post} + C_{CD,P2} \quad (6.21)$$

6.1.3. Stage 3: P2 to P3

6.1.3.1. Communication between processes P2 and P3

Let $T_{CD,P2\text{-}P3,REF}$ denote the reference time for the transmission between P2 and P3. We choose the end of the deskewing window in process P2 as the reference time.

$$T_{CD,P2\text{-}P3,REF} = T_{CD,P2,RCV,E} + W_{Deskew,post} \quad (6.22)$$

$S_{CD,P2}$ denotes the Send Process delay for process P2, $\Delta_{CD,P3,RCVWND}$ denotes the delay in opening the reception window in process P3, $T_{CD,P2,SND}$ denotes the send time for process P2, and $T_{CD,P3,RCV,E}$ denotes the expected time of reception for process P3.

We assume that the constraint on the minimum data introduction interval for the Communication Module is satisfied by the minimum delay between the send times for processes P1 and P2. That is, we assume that the following is true:

$$R_{PP} + W_{Deskew,post} + C_{CD,P2} + S_{CD,P2}|_{min} \geq \Lambda_{Comm} \quad (6.23)$$

Case 1: $C_{CD,P2} + S_{CD,P2}|_{min} + R_{PP} \geq \Delta_{CD,P3,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P2} = S_{CD,P2}|_{min} \quad (6.24)$$

$$\Delta_{CD,P3,RCVWND} = C_{CD,P2} + S_{CD,P2}|_{min} + R_{PP} - W_{Deskew,pre} \quad (6.25)$$

So:

$$\begin{aligned} T_{CD,P2,SND} &= T_{CD,P2\text{-}P3,REF} + C_{CD,P2} + S_{CD,P2} \\ &= T_{CD,P2,RCV,E} + W_{Deskew,post} + C_{CD,P2} + S_{CD,P2}|_{min} \end{aligned} \quad (6.26)$$

And:

$$\begin{aligned} T_{CD,P3,RCV,E} &= T_{CD,P2,SND} + R_{PP} \\ &= T_{CD,P2,RCV,E} + W_{Deskew,post} + C_{CD,P2} + S_{CD,P2}|_{min} + R_{PP} \end{aligned} \quad (6.27)$$

Case 2: $C_{CD,P2} + S_{CD,P2}|_{min} + R_{PP} < \Delta_{CD,P3,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P2} = \Delta_{CD,P3,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} - C_{CD,P2} \quad (6.28)$$

$$\Delta_{CD,P3,RCVWND} = \Delta_{CD,P3,RCVWND}|_{min} \quad (6.29)$$

So:

$$T_{CD,P2,SND} = T_{CD,P2\text{-}P3,REF} + C_{CD,P2} + S_{CD,P2}$$

$$= T_{CD,P2,RCV,E} + W_{Deskew,post} + \Delta_{CD,P3,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.30)$$

And:

$$T_{CD,P3,RCV,E} = T_{CD,P2,SND} + R_{PP}$$

$$= T_{CD,P2,RCV,E} + W_{Deskew,post} + \Delta_{CD,P3,RCVWND}|_{min} + W_{Deskew,pre} \quad (6.31)$$

6.1.3.2. Computation in process P3

Let $C_{CD,P3}$ denote the computation delay in process P3. The computation delay is measured from the end of the deskewing window. $C_{CD,P3}$ is assumed to be a constant independent of the computation input and the system state. $T_{CD,P3,C}$ denotes the local-time at which the Computation Process outputs the result for process P3.

$$T_{CD,P3,C} = T_{CD,P3,RCV,E} + W_{Deskew,post} + C_{CD,P3} \quad (6.32)$$

6.1.4. Stage 4: P3 to P4

6.1.4.1. Communication between processes P3 and P4

Let $T_{CD,P3\text{-}P4,REF}$ denote the reference time for the transmission between P3 and P4. We choose the end of the deskewing window in process P3 as the reference time.

$$T_{CD,P3\text{-}P4,REF} = T_{CD,P3,RCV,E} + W_{Deskew,post} \quad (6.33)$$

$S_{CD,P3}$ denotes the Send Process delay for process P3, $\Delta_{CD,P4,RCVWND}$ denotes the delay in opening the reception window in process P4, $T_{CD,P3,SND}$ denotes the send time for process P3, and $T_{CD,P4,RCV,E}$ denotes the expected time of reception for process P4.

We assume that the constraint on the minimum data introduction interval for the Communication Module is satisfied by the minimum delay between the send times for processes P2 and P3. That is, we assume that the following is true:

$$R_{PP} + W_{Deskew,post} + C_{CD,P3} + S_{CD,P3}|_{min} \geq \Lambda_{Comm} \quad (6.34)$$

Case 1: $C_{CD,P3} + S_{CD,P3}|_{min} + R_{PP} \geq \Delta_{CD,P4,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P3} = S_{CD,P3}|_{min} \tag{6.35}$$

$$\Delta_{CD,P4,RCVWND} = C_{CD,P3} + S_{CD,P3}|_{min} + R_{PP} - W_{Deskew,pre} \tag{6.36}$$

So:

$$T_{CD,P3,SND} = T_{CD,P3\text{-}P4,REF} + C_{CD,P3} + S_{CD,P3}$$

$$= T_{CD,P2,RCV,E} + W_{Deskew,post} + C_{CD,P3} + S_{CD,P3}|_{min} \tag{6.37}$$

And:

$$T_{CD,P4,RCV,E} = T_{CD,P3,SND} + R_{PP}$$

$$= T_{CD,P3,RCV,E} + W_{Deskew,post} + C_{CD,P3} + S_{CD,P3}|_{min} + R_{PP} \tag{6.38}$$

Case 2: $C_{CD,P3} + S_{CD,P3}|_{min} + R_{PP} < \Delta_{CD,P4,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{CD,P3} = \Delta_{CD,P4,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} - C_{CD,P3} \tag{6.39}$$

$$\Delta_{CD,P4,RCVWND} = \Delta_{CD,P4,RCVWND}|_{min} \tag{6.40}$$

So:

$$T_{CD,P3,SND} = T_{CD,P3\text{-}P4,REF} + C_{CD,P3} + S_{CD,P3}$$

$$= T_{CD,P3,RCV,E} + W_{Deskew,post} + \Delta_{CD,P4,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \tag{6.41}$$

And:

$$T_{CD,P4,RCV,E} = T_{CD,P3,SND} + R_{PP}$$

$$= T_{CD,P3,RCV,E} + W_{Deskew,post} + \Delta_{CD,P4,RCVWND}|_{min} + W_{Deskew,pre} \tag{6.42}$$

6.1.4.2. Computation in process P4

Let $C_{CD,P4}$ denote the computation delay in process P4. The computation delay is measured from the end of the deskewing window. $C_{CD,P4}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{CD,P4,C}$ denote the local time at which the Computation Process outputs the result for process P4.

$$T_{CD,P4,C} = T_{CD,P4,RCV,E} + W_{Deskew,post} + C_{CD,P4} \tag{6.43}$$

6.1.5. Duration of the protocol

Let $\Delta_{CD,P4,C\text{-}END}$ denote the delay in process P4 from the end of computation to the end of the Collective

Diagnosis protocol. $\Delta_{CD,P4,C\text{-}END}$ is assumed to be a constant independent of the system state. Let $T_{CD,END}$ denote the local-time at which the execution of the Collective Diagnosis protocol ends.

$$T_{CD,END} = T_{CD,P4,C} + \Delta_{CD,P4,C\text{-}END} \quad (6.44)$$

Let Δ_{CD} denote the duration of the execution of the Collective Diagnosis protocol.

$$\Delta_{CD} = T_{CD,END} - T_{CD} \quad (6.45)$$

6.2. Schedule Update Protocol

Figure 6.3 shows the message flow graph for the Schedule Update protocol.

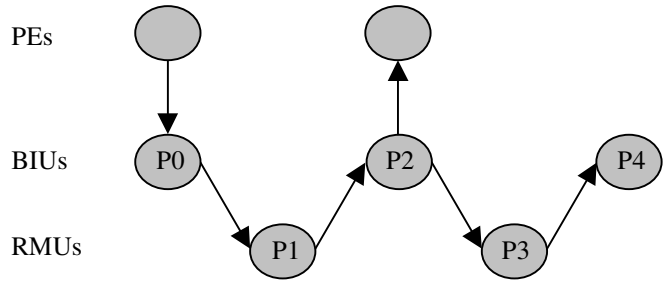


Figure 6.3: Message flow graph for the Schedule Update protocol

6.2.1. Stage 1: P0 to P1

Let T_{SU} denote the local-time trigger for the execution of the Schedule Update protocol. The length of the message stream is equal to the number of BIUs, denoted by N. Let i denote the index for the messages in the stream.

$$0 \le i \le N - 1 \quad (6.46)$$

Let Λ_{SU} denote the data introduction rate for the stream. Λ_{SU} must be greater than or equal to the minimum data introduction interval for the Communication and Computation Modules.

$$\Lambda_{SU} \ge \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (6.47)$$

Let $S_{SU,P0}$ denote the Send Process delay for process P0. $S_{SU,P0}$ is assumed to apply to all the messages in the stream. Let $\Delta_{SU,P1,RCVWND}$ denote the delay from the communication reference time to the opening of the reception window in process P1. $\Delta_{SU,P1,RCVWND}$ is assumed to apply to all the messages in the stream.

$T_{SU,P0,SND,i}$ denotes the send time for the i-th message in process P0 and $T_{SU,P1,RCV,E,i}$ denotes the expected time of reception for the i-th message in process P1. $T_{SU,P0\text{-}P1,REF,i}$ denotes the reference time for the transmission of the i-th message between P0 and P1.

$$T_{SU,P0\text{-}P1,REF,i} = T_{SU} + i\Lambda_{SU} \quad (6.48)$$

For the version of the RPP covered in this document, the RPP sends mode and node identification

messages to the PE triggered by the beginning of the Schedule Update protocol.

The RPP sends the current-mode message to the PE $\Delta_{SU,SND\text{-}MODE}$ ticks after the beginning of the protocol. $\Delta_{SU,SND\text{-}MODE\text{-}READ\text{-}PE}$ denotes the delay from the time the mode message is sent to the PE to the time at which the first schedule message is read from the PE. The RPP is designed to read each PE message 1 tick before it is to be sent. The absolute minimum value of $S_{SU,P0}$, denoted by $S_{SU,P0}|_{min}$, and the delay to read the first PE message constrain the actual minimum value of $S_{SU,P0}$ to the following effective minimum value.

$$S_{SU,P0}|_{min,eff} = \max(S_{SU,P0}|_{min} , \Delta_{SU,SND\text{-}MODE} + \Delta_{SU,SND\text{-}MODE\text{-}READ\text{-}PE}|_{min} + 1) \quad (6.49)$$

6.2.1.1. Communication between processes P0 and P1

Two cases must be considered.

Case 1: $S_{SU,P0}|_{min,eff} + R_{PP} \geq \Delta_{SU,P1,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{SU,P0} = S_{SU,P0}|_{min,eff} \quad (6.50)$$

$$\Delta_{SU,P1,RCVWND} = S_{SU,P0}|_{min,eff} + R_{PP} - W_{Deskew,pre} \quad (6.51)$$

So:

$$T_{SU,P0,SND,i} = T_{SU,P0\text{-}P1,REF,i} + S_{SU,P0}$$

$$= T_{SU} + i\Lambda_{SU} + S_{SU,P0}|_{min,eff} \quad (6.52)$$

And:

$$T_{SU,P1,RCV,E,i} = T_{SU,P0,SND,i} + R_{PP}$$

$$= T_{SU} + i\Lambda_{SU} + S_{SU,P0}|_{min,eff} + R_{PP} \quad (6.53)$$

Case 2: $S_{SU,P0}|_{min,eff} + R_{PP} < \Delta_{SU,P1,RCVWND}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{SU,P0} = \Delta_{SU,P1,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.54)$$

$$\Delta_{SU,P1,RCVWND} = \Delta_{SU,P1,RCVWND}|_{min} \quad (6.55)$$

So:

$$T_{SU,P0,SND,i} = T_{SU,P0\text{-}P1,REF,i} + S_{SU,P0}$$

$$= T_{SU} + i\Lambda_{SU} + \Delta_{SU,P1,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.56)$$

And:

$$T_{SU,P1,RCV,E,i} = T_{SU,P0,SND,i} + R_{PP}$$

$$= T_{SU} + i\Lambda_{SU} + \Delta_{SU,P1,RCVWND}|_{min} + W_{Deskew,pre} \quad (6.57)$$

6.2.1.2. Computation in process P1

Let $C_{SU,P1}$ denote the computation delay in process P1. The computation delay is measured from the end of the deskewing window. $C_{SU,P1}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{SU,P1,C,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P1.

$$T_{SU,P1,C,i} = T_{SU,P1,RCV,E,i} + W_{Deskew,post} + C_{SU,P1} \quad (6.58)$$

6.2.2. Stage 2: P1 to P2

6.2.2.1. Communication between processes P1 and P2

Let $T_{SU,P1\text{-}P2,REF,i}$ denote the reference time for the i-th transmission between P1 and P2. We choose the reference times for the transmissions between P1 and P2 to be the same as for the transmissions between P0 and P1.

$$T_{SU,P1\text{-}P2,REF,i} = T_{SU,P0\text{-}P1,REF,i} = T_{SU} + i\Lambda_{SU} \quad (6.59)$$

Let $\Delta_{SU,P1,REF\text{-}SND}$ denote the delay from the reference time to the send time for the transmissions from process P1 to process P2. $S_{SU,P1}$ denotes the Send Process delay for process P1 measured from $T_{SU,P1,C,i}$ to the corresponding send time, $\Delta_{SU,P2,RCVWND}$ denotes the delay to open the reception window in process P2 measured from the communication reference time to the opening of the input window, $T_{SU,P1,SND,i}$ denotes the send time for the i-th message in process P1, and $T_{SU,P2,RCV,E,i}$ denotes the expected time of reception for the i-th message in process P2.

Since process P2 starts receiving messages after process P1 does, it is expected that process P2 will have ample time to open its input windows before the messages start arriving. Therefore, we only need to consider the case in which process P1 sends the messages as soon as possible.

$\Delta_{SU,P1,REF\text{-}SND}$ is measured from the reference times for the transmissions between P1 and P2. As such, in this case, $\Delta_{SU,P1,REF\text{-}SND}$ includes the delay from the reference time until the time at which the Computation Process in P1 outputs the results. The actual Send Process delay is only $S_{SU,P1}|_{min}$.

So:

$$\Delta_{SU,P1,REF\text{-}SND} = S_{SU,P0} + R_{PP} + W_{Deskew,post} + C_{SU,P1} + S_{SU,P1}|_{min} \quad (6.60)$$

$$S_{SU,P1} = S_{SU,P1}|_{min} \quad (6.61)$$

And:

$$\Delta_{SU,P2,RCVWND} = \Delta_{SU,P1,RCVWND} + W_{Deskew} + C_{SU,P1} + S_{SU,P1}|_{min} + R_{PP} - W_{Deskew,pre} \quad (6.62)$$

$$= \Delta_{SU,P1,RCVWND} + W_{Deskew,post} + C_{SU,P1} + S_{SU,P1}|_{min} + R_{PP} \quad (6.63)$$

In addition:

$$T_{SU,P1,SND,i} = T_{SU,P1,C,i} + S_{SU,P1}$$

$$= T_{SU,P1,RCV,E,i} + W_{Deskew,post} + C_{SU,P1} + S_{SU,P1}|_{min} \quad (6.64)$$

And:

$$T_{SU,P2,RCV,E,i} = T_{SU,P1,SND,i} + R_{PP}$$

$$= T_{SU,P1,RCV,E,i} + W_{Deskew,post} + C_{SU,P1} + S_{SU,P1}|_{min} + R_{PP} \quad (6.65)$$

6.2.2.2. Computation in process P2

Let $C_{SU,P2}$ denote the computation delay in process P2. The computation delay is measured from the end of the deskewing window. $C_{SU,P2}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{SU,P2,C,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P2.

$$T_{SU,P2,C,i} = T_{SU,P2,RCV,E,i} + W_{Deskew,post} + C_{SU,P2} \quad (6.66)$$

6.2.3. Stage 3: P2 to P3

6.2.3.1. Communication between processes P2 and P3

The communication pattern of the Schedule Update protocol consists of two passes through the system. Conceptually, we think of the protocol as processing two streams. The first stream is processed from P0 to P2, and the second stream is processed from P2 to P4.

Let $\Delta_{SU,STREAM,P0\text{-}P2}|_{min}$ denote the desired minimum separation between the two streams. This separation constraint is measured from the last message of the first stream to the first message of the second stream, and it is assumed to apply at the inputs and outputs of the BIUs and the RMUs. $\Delta_{SU,STREAM,P0\text{-}P2}|_{BIU}$ and $\Delta_{SU,STREAM,P0\text{-}P2}|_{RMU}$ denote the actual separation between the streams at the BIUs in process P2 and at the RMUs in process P3, respectively.

$\Delta_{SU,STREAM,P0\text{-}P2}|_{min}$ must be greater than or equal to the stream's data introduction interval.

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{min} \geq \Lambda_{SU} \quad (6.67)$$

At P3, we want the expected reception intervals for the first and the second stream to be separated by at least one tick. This is captured by the following constraint.

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{min} \geq W_{Deskew} + 1 \quad (6.68)$$

The effective allowed minimum separation between the streams is:

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff} = \max(\Delta_{SU,STREAM,P0\text{-}P2}|_{min}\ ,\ \Lambda_{SU}\ ,\ W_{Deskew} + 1) \tag{6.69}$$

Let $T_{SU,P2\text{-}P3,REF,i}$ denote the reference time for the i-th transmission between P2 and P3. We choose these reference times to be the equal to the times at which the Computation Process of P2 outputs the corresponding results.

$$T_{SU,P2\text{-}P3,REF,i} = T_{SU,P2,C,i} \tag{6.70}$$

$S_{SU,P2}$ denotes the Send Process delay for process P2, $T_{SU,P2,SND,i}$ denotes the send time for the i-th message in process P2, and $T_{SU,P3,RCV,E,i}$ denotes the expected time of reception for the i-th message in process P3.

$T_{SU,P2,SND,i}$ must be chosen to ensure that the constraint on the minimum stream separation is satisfied. Two cases must be considered.

Case 1: $T_{SU,P2,C,0} + S_{SU,P2}|_{min} < T_{SU,P0,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}$

For this case, the results of the computation in P2 must be buffered until they can be sent with the proper separation from the first stream. So:

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{BIU} = \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff} \tag{6.71}$$

Let $\Delta_{SU,P2,BUF}$ denote the buffering delay for the stream at P2.

$$\Delta_{SU,P2,BUF} = (T_{SU,P0,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}) - (T_{SU,P2,C,0} + S_{SU,P2}|_{min}) \tag{6.72}$$

The send delay for process P2 is:

$$S_{SU,P2} = S_{SU,P2}|_{min} + \Delta_{SU,P2,BUF} \tag{6.73}$$

So:

$$\begin{aligned} T_{SU,P2,SND,i} &= T_{SU,P2,C,i} + S_{SU,P2} \\ &= T_{SU,P2,RCV,E,i} + W_{Deskew,post} + C_{SU,P2} + S_{SU,P2} \end{aligned} \tag{6.74}$$

And:

$$\begin{aligned} T_{SU,P3,RCV,E,i} &= T_{SU,P2,SND,i} + R_{PP} \\ &= T_{SU,P2,RCV,E,i} + W_{Deskew,post} + C_{SU,P2} + S_{SU,P2} + R_{PP} \end{aligned} \tag{6.75}$$

Case 2: $T_{SU,P2,C,0} + S_{SU,P2}|_{min} \geq T_{SU,P0,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}$

For this case, the results of the computation can be sent as soon as possible. The send delay is the minimum value.

$$S_{SU,P2} = S_{SU,P2}|_{min} \tag{6.76}$$

So:

$$T_{SU,P2,SND,i} = T_{SU,P2,C,i} + S_{SU,P2}$$

$$= T_{SU,P2,RCV,E,i} + W_{Deskew,post} + C_{SU,P2} + S_{SU,P2}|_{min} \tag{6.77}$$

And:

$$T_{SU,P3,RCV,E,i} = T_{SU,P2,SND,i} + R_{PP}$$

$$= T_{SU,P2,RCV,E,i} + W_{Deskew,post} + C_{SU,P2} + S_{SU,P2}|_{min} + R_{PP} \tag{6.78}$$

Therefore, the actual separation between the streams at the BIUs is:

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{BIU} = T_{SU,P2,SND,0} - T_{SU,P0,SND,N-1} \tag{6.79}$$

$$= T_{SU,P2,SND,0} - [T_{SU,P0,SND,0} + (N-1)\Lambda_{SU}] \tag{6.80}$$

$$= 2R_{PP} + 2W_{Deskew,post} + (C_{SU,P1} + C_{SU,P2}) + (S_{SU,P1} + S_{SU,P2}) - (N-1)\Lambda_{SU} \tag{6.81}$$

6.2.3.2. Computation in process P3

Let $C_{SU,P3}$ denote the computation delay in process P3. The computation delay is measured from the end of the deskewing window. $C_{SU,P3}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{SU,P3,C,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P3.

$$T_{SU,P3,C,i} = T_{SU,P3,RCV,E,i} + W_{Deskew,post} + C_{SU,P3} \tag{6.82}$$

6.2.4. Stage 4: P3 to P4

6.2.4.1. Communication between processes P3 and P4

$\Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}$ also applies to the communication between P3 and P4. Let $S_{SU,P3}$ denote the Send Process delay for process P3. $S_{SU,P3}$ is measured from the time the Computation Process outputs a message until the time the message is sent. Let $T_{SU,P3\text{-}P4,REF,i}$ denote the reference time for the i-th transmission between P3 and P4. We choose these reference times to be the equal to the times at which the Computation Process outputs the corresponding results.

$$T_{SU,P3\text{-}P4,REF,i} = T_{SU,P3,C,i} \tag{6.83}$$

$T_{SU,P3,SND,i}$ denotes the send time for the i-th message in process P3, and $T_{SU,P3,RCV,E,i}$ denotes the expected time of reception for the i-th message in process P3. $T_{SU,P3,SND,i}$ must be chosen to ensure that the constraint on the minimum stream separation is satisfied. Two cases must be considered.

Case 1: $T_{SU,P3,C,0} + S_{SU,P3}|_{min} < T_{SU,P1,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}$

For this case, the results of the computation in P3 must be buffered until they can be sent with the proper separation from the first stream. So:

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{RMU} = \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff} \quad (6.84)$$

Let $\Delta_{SU,P3,BUF}$ denote the buffering delay for the stream at P3.

$$\Delta_{SU,P3,BUF} = (T_{SU,P1,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}) - (T_{SU,P3,C,0} + S_{SU,P3}|_{min}) \quad (6.85)$$

The send delay for process P3 is:

$$S_{SU,P3} = S_{SU,P3}|_{min} + \Delta_{SU,P3,BUF} \quad (6.86)$$

So:

$$T_{SU,P3,SND,i} = T_{SU,P3,C,i} + S_{SU,P3}$$

$$= T_{SU,P3,RCV,E,i} + W_{Deskew,post} + C_{SU,P3} + S_{SU,P3} \quad (6.87)$$

And:

$$T_{SU,P4,RCV,E,i} = T_{SU,P3,SND,i} + R_{PP}$$

$$= T_{SU,P3,RCV,E,i} + W_{Deskew,post} + C_{SU,P3} + S_{SU,P3} + R_{PP} \quad (6.88)$$

Case 2: $T_{SU,P3,C,0} + S_{SU,P3}|_{min} \geq T_{SU,P1,SND,N-1} + \Delta_{SU,STREAM,P0\text{-}P2}|_{min,eff}$

For this case, the results of the computation can be sent as soon as possible. The send delay is the minimum value.

$$S_{SU,P3} = S_{SU,P3}|_{min} \quad (6.89)$$

So:

$$T_{SU,P3,SND,i} = T_{SU,P3,C,i} + S_{SU,P3}$$

$$= T_{SU,P3,RCV,E,i} + W_{Deskew,post} + C_{SU,P3} + S_{SU,P3}|_{min} \quad (6.90)$$

And:

$$T_{SU,P4,RCV,E,i} = T_{SU,P3,SND,i} + R_{PP}$$

$$= T_{SU,P3,RCV,E,i} + W_{Deskew,post} + C_{SU,P3} + S_{SU,P3}|_{min} + R_{PP} \quad (6.91)$$

Therefore:

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{RMU} = T_{SU,P3,SND,0} - T_{SU,P1,SND,N-1} \quad (6.92)$$

$$= T_{SU,P3,SND,0} - [T_{SU,P1,SND,0} + (N-1)\Lambda_{SU}] \quad (6.93)$$

$$= 2R_{PP} + 2W_{Deskew,post} + (C_{SU,P2} + C_{SU,P3}) + (S_{SU,P2} + S_{SU,P3}) - (N-1)\Lambda_{SU} \quad (6.94)$$

6.2.4.2. Computation in process P4

Let $C_{SU,P4}$ denote the computation delay in process P4. The computation delay is measured from the end of the deskewing window. $C_{SU,P4}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{SU,P4,C,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P4.

$$T_{SU,P4,C,i} = T_{SU,P4,RCV,E,i} + W_{Deskew,post} + C_{SU,P4} \quad (6.95)$$

6.2.5. Duration of the protocol

Let $\Delta_{SU,P4,C\text{-}END}$ denote the delay in process P4 from the end of the computation for the last message to the end of the Schedule Update protocol. $\Delta_{SU,P4,C\text{-}END}$ is assumed to be a constant independent of the system state. Let $T_{SU,END}$ denote the local-time at which the execution of the Schedule Update protocol ends.

$$T_{SU,END} = T_{SU,P4,C,N-1} + \Delta_{SU,P4,C\text{-}END} \quad (6.96)$$

Let Δ_{SU} denote the duration of the Schedule Update protocol execution.

$$\Delta_{SU} = T_{SU,END} - T_{SU} \quad (6.97)$$

6.3. PE Communication and Accusation Exchange Protocols

Figures 6.4 and 6.5 show the message flow graphs for the PE Broadcast and Accusation Exchange protocols, respectively. For this RPP, these protocols are implemented as a single protocol composed of two phases. In the first phase, the scheduled PE messages are processed as a stream. In the second phase, the protocol adds a message containing accusations against nodes of the opposite kind. The processing of the scheduled PE messages is examined next. The analysis of the accusations messages is presented after that.

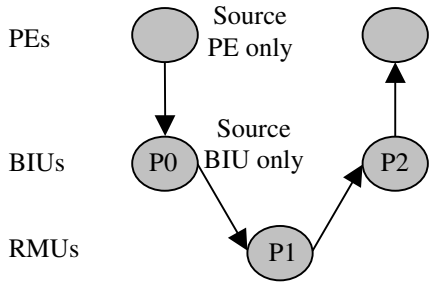


Figure 6.4: Message flow graph for the PE Broadcast protocol

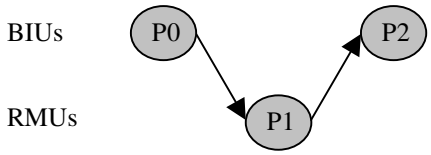


Figure 6.5: Message flow graph for the Accusation Exchange protocol

6.3.1. Scheduled PE messages in stage 1: P0 to P1

Let T_{PE} denote the local-time trigger for the transition to PE Communication mode. $K_{PE,sched}$[1] denotes the number of scheduled PE messages. Let $\Lambda_{PE,sched}$ denote the data introduction interval for the stream of scheduled PE messages. $\Lambda_{PE,sched}$ must satisfy the constraints on the minimum data introduction intervals for the Communication and Computation modules.

$$\Lambda_{PE,sched} \geq \max(\Lambda_{Comm}, \Lambda_{Comp}) \tag{6.98}$$

Let i denote the index for the stream of scheduled PE messages.

$$0 \leq i \leq K_{PE,sched} - 1 \tag{6.99}$$

Let $S_{PE,P0,sched}$ denote the Send Process delay for scheduled PE messages in process P0. We assume that $S_{PE,P0,sched}$ is a constant independent of process input. Let $\Delta_{PE,P1,RCVWND,sched}$ denote the delay from the communication reference time to the opening of the reception window for scheduled PE messages in process P1. We assume that $\Delta_{PE,P1,RCVWND,sched}$ is a constant. $T_{PE,P0,SND,sched,i}$ denotes the send time for the i-th scheduled PE message in process P0, and $T_{PE,P1,RCV,E,sched,i}$ denotes the expected time of reception for the i-th scheduled PE message in process P1. Let $T_{PE,P0\text{-}P1,REF,sched,i}$ denote the reference time for the transmission of the i-th scheduled PE message between P0 and P1.

$$T_{PE,P0\text{-}P1,REF,sched,i} = T_{PE} + i\Lambda_{PE,sched} \tag{6.100}$$

For the version of the RPP covered in this document, the RPP sends the schedule-assessment message to the PE triggered by the beginning of the PE Communication minor mode. The RPP sends the message to the PE $\Delta_{PE,SND\text{-}SA}$ ticks after the beginning of the protocol at time T_{PE}. $\Delta_{PE,SND\text{-}SA\text{-}READ\text{-}PE}$ denotes the delay from the time the schedule assessment message is sent to the PE to the time at which the first PE message is read from a PE. The RPP is designed to read each PE message 1 tick before it is to be sent. The absolute minimum value of $S_{PE,P0,sched}$, denoted by $S_{PE,P0,sched}|_{min}$, and the delay to read the first PE message constrain the actual minimum value of $S_{PE,P0,sched}$ to the following effective minimum value.

$$S_{PE,P0,sched}|_{min,eff} = \max(S_{PE,P0,sched}|_{min}\ ,\ \Delta_{PE,SND\text{-}SA} + \Delta_{PE,SND\text{-}SA\text{-}READ\text{-}PE}|_{min}\ + 1) \tag{6.101}$$

6.3.1.1. Communication between processes P0 and P1

Two cases must be considered.

Case 1: $S_{PE,P0,sched}|_{min,eff} + R_{PP} \geq \Delta_{PE,P1,RCVWND,sched}|_{min} + W_{Deskew,pre}$

[1] $K_{PE,sched}$ is denoted simply as K when used as a subscript.

For this case:

$$S_{PE,P0,sched} = S_{PE,P0,sched}|_{min,eff} \quad (6.102)$$

$$\Delta_{PE,P1,RCVWND,sched} = S_{PE,P0,sched}|_{min,eff} + R_{PP} - W_{Deskew,pre} \quad (6.103)$$

So:

$$T_{PE,P0,SND,sched,i} = T_{PE,P0\text{-}P1,REF,sched,i} + S_{PE,P0,sched}|_{min,eff}$$

$$= T_{PE} + i\Lambda_{PE,sched} + S_{PE,P0,sched}|_{min,eff} \quad (6.104)$$

And:

$$T_{PE,P1,RCV,E,sched,i} = T_{PE,P0,SND,sched,i} + R_{PP}$$

$$= T_{PE} + i\Lambda_{PE,sched} + S_{PE,P0,sched}|_{min,eff} + R_{PP} \quad (6.105)$$

Case 2: $S_{PE,P0,sched}|_{min,eff} + R_{PP} < \Delta_{PE,P1,RCVWND,sched}|_{min} + W_{Deskew,pre}$

For this case:

$$S_{PE,P0,sched} = \Delta_{PE,P1,RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad (6.106)$$

$$\Delta_{PE,P1,RCVWND,sched} = \Delta_{PE,P1,RCVWND,sched}|_{min} \quad (6.107)$$

So:

$$T_{PE,P0,SND,sched,i} = T_{PE,P0\text{-}P1,REF,sched,i} + S_{PE,P0,sched} \quad (6.108)$$

$$= T_{PE} + i\Lambda_{PE,sched} + \Delta_{PE,P1,RCVWND,sched}|_{min} + W_{Deskew,pre} - R_{PP}$$

And:

$$T_{PE,P1,RCV,E,sched,i} = T_{PE,P0,SND,sched,i} + R_{PP}$$

$$= T_{PE} + i\Lambda_{PE,sched} + \Delta_{PE,P1,RCVWND,sched}|_{min} + W_{Deskew,pre} \quad (6.109)$$

6.3.1.2. Computation in process P1

Let $C_{PE,P1,sched}$ denote the computation delay for scheduled PE messages in process P1 . The computation delay is measured from the end of the deskewing window for each message. $C_{PE,P1,sched}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{PE,P1,C,sched,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P1.

$$T_{PE,P1,C,sched,i} = T_{SU,P1,RCV,E,sched,i} + W_{Deskew,post} + C_{PE,P1,sched} \quad (6.110)$$

6.3.2. Scheduled PE messages in stage 2: P1 to P2

6.3.2.1. Communication between processes P1 and P2

Let $T_{SU,P1\text{-}P2,REF,sched,i}$ denote the reference time for the i-th transmission between processes P1 and P2. We choose the reference times for the transmissions between P1 and P2 to be the same as for the transmissions between P0 and P1.

$$T_{PE,P1\text{-}P2,REF,sched,i} = T_{PE,P0\text{-}P1,REF,sched,i} = T_{PE} + i\Lambda_{PE,sched} \tag{6.111}$$

$\Delta_{PE,P1,REF\text{-}SND,sched}$ denotes the delay from the reference time to the send time for the transmissions of scheduled PE messages from process P1 to process P2, $S_{PE,P1,sched}$ denotes the Send Process delay for process P1 measured from $T_{SU,P1,C,i}$ to the corresponding send time, $\Delta_{PE,P2,RCVWND,sched}$ denotes the delay in the opening of the reception window in process P2 measured from the communication reference time to the opening of the input window, $T_{PE,P1,SND,sched,i}$ denotes the send time for the i-th message in process P1, and $T_{PE,P2,RCV,E,sched,i}$ denotes the expected time of reception for the i-th message in process P2. Since process P2 starts receiving messages after process P1 does, it is expected that process P2 will have ample time to open its input windows before the messages start arriving. Therefore, we only consider the case in which process P1 sends the messages as soon as possible.

$\Delta_{PE,P1,REF\text{-}SND,sched}$ is measured from the reference times for the transmissions between P1 and P2 of scheduled PE messages. As such, in this case, $\Delta_{PE,P1,REF\text{-}SND,sched}$ includes the delay from the reference time to the time at which the Computation Process in P1 outputs the results. The actual Send Process delay is only $S_{PE,P1,sched}|_{min}$. So:

$$\Delta_{PE,P1,REF\text{-}SND,sched} = S_{PE,P0,sched} + R_{PP} + W_{Deskew,post} + C_{PE,P1,sched} + S_{PE,P1,sched}|_{min} \tag{6.112}$$

And:

$$S_{PE,P1,sched} = S_{PE,P1,sched}|_{min} \tag{6.113}$$

Also:

$$\Delta_{PE,P2,RCVWND,sched} = \Delta_{PE,P1,RCVWND,sched} + W_{Deskew} + C_{PE,P1,sched} + S_{PE,P1,sched}|_{min} + R_{PP} - W_{Deskew,pre} \tag{6.114}$$

$$= \Delta_{PE,P1,RCVWND,sched} + W_{Deskew,post} + C_{PE,P1,sched} + S_{PE,P1,sched}|_{min} + R_{PP} \tag{6.115}$$

And:

$$T_{PE,P1,SND,sched,i} = T_{PE,P1,C,sched,i} + S_{PE,P1,sched}|_{min} \tag{6.116}$$

$$= T_{PE,P1,RCV,E,sched,i} + W_{Deskew,post} + C_{PE,P1,sched} + S_{PE,P1,sched}|_{min} \tag{6.117}$$

And:

$$T_{PE,P2,RCV,E,sched,i} = T_{PE,P1,SND,sched,i} + R_{PP} \tag{6.118}$$

$$= T_{SU,P1,RCV,E,sched,i} + W_{Deskew,post} + C_{PE,P1,sched} + S_{PE,P1,sched}|_{min} + R_{PP} \tag{6.119}$$

6.3.2.2. Computation in process P2

Let $C_{PE,P2,sched}$ denote the computation delay in process P2. The computation delay is measured from the end of the deskewing window for each message. $C_{PE,P2,sched}$ is assumed to be a constant independent of the computation input and the system state. Let $T_{PE,P2,C,sched,i}$ denote the local-time at which the Computation Process outputs the i-th result for process P2.

$$T_{PE,P2,C,sched,i} = T_{PE,P2,RCV,E,sched,i} + W_{Deskew,post} + C_{PE,P2,sched} \quad (6.120)$$

6.3.3. Accusations message in stage 1: P0 to P1

The timing of the accusations message can be analyzed as if it were part of a separate stream being processed after the stream of PE messages. The analysis presented here follows the analysis used for the Schedule Update protocol.

6.3.3.1. Communication between processes P0 and P1

Let $\Delta_{PE,sched\text{-}acc}|_{min}$ denote the desired minimum separation between the stream of PE messages and the accusations message. This separation constraint is measured from the last message of the PE message stream to the accusations message, and it is assumed to apply at the inputs and outputs of the BIUs and the RMUs. We assume that $\Delta_{PE,sched\text{-}acc}|_{min}$ is greater than or equal to the data introduction interval of the PE message stream.

$$\Delta_{PE,sched\text{-}acc}|_{min} \geq \Lambda_{PE,sched} \quad (6.121)$$

Let $T_{PE,P0,acc}$ denote the time at which the accusations are available for transmission in process P0. It is assumed that $T_{PE,P0,acc}$ occurs after the reference time for transmission of the last PE message.

$$T_{PE,P0,acc} > T_{PE} + (K_{PE,sched} - 1)\Lambda_{PE,sched} \quad (6.122)$$

Let $\Delta_{PE,P0,acc\_ready}$ denote the delay from the reference time for transmission of the last PE message to the time at which the accusations are available for transmission in process P0.

$$\Delta_{PE,P0,acc\_ready} = T_{PE,P0,acc} - [T_{PE} + (K_{PE,sched} - 1)\Lambda_{PE,sched}] \quad (6.123)$$

$S_{PE,P0,acc}$ denotes the Send Process delay for the accusations message in process P0, $T_{PE,P0,SND,acc}$ denotes the send time for the accusations message in process P0, $T_{PE,P1,RCV,E,acc}$ denotes the expected time of reception for the accusations message in process P1, $T_{PE,P0,SND,sched,K-1}$ denotes the send time for the last scheduled PE message. $T_{PE,P0,SND,acc}$ must be chosen to ensure that the constraint on the minimum stream separation is satisfied. Two cases must be considered.

Case 1: $T_{PE,P0,acc} + S_{PE,P0,acc}|_{min} < T_{PE,P0,SND,sched,K-1} + \Delta_{PE,sched\text{-}acc}|_{min}$

For this case, the accusations message must be buffered until it can be sent with the proper separation from the PE message stream. Let $\Delta_{PE,P0,BUF,acc}$ denote the buffering delay for the accusations at P0.

$$\Delta_{PE,P0,BUF,acc} = (T_{PE,P0,SND,sched,K-1} + \Delta_{PE,sched\text{-}acc}|_{min}) - (T_{PE,P0,acc} + S_{PE,P0,acc}|_{min}) \quad (6.124)$$

The send delay for the accusations message in process P0 is:

$$S_{PE,P0,acc} = S_{PE,P0,acc}|_{min} + \Delta_{PE,P0,BUF,acc} \quad (6.125)$$

So:

$$T_{PE,P0,SND,acc} = T_{PE,P0,acc} + S_{PE,P0,acc} \quad (6.126)$$

And:

$$T_{PE,P1,RCV,E,acc} = T_{PE,P0,SND,acc} + R_{PP}$$

$$= T_{PE,P0,acc} + S_{PE,P0,acc} + R_{PP} \quad (6.127)$$

Case 2: $T_{PE,P0,acc} + S_{PE,P0,acc}|_{min} \geq T_{PE,P0,SND,sched,K-1} + \Delta_{PE,sched-acc}|_{min}$

For this case, the accusations message can be sent as soon as possible. The send delay is the minimum value.

$$S_{PE,P0,acc} = S_{PE,P0,acc}|_{min} \quad (6.128)$$

So:

$$T_{PE,P0,SND,acc} = T_{PE,P0,acc} + S_{PE,P0,acc}$$

$$= T_{PE,P0,acc} + S_{PE,P0,acc}|_{min} \quad (6.129)$$

And:

$$T_{PE,P1,RCV,E,acc} = T_{PE,P0,SND,acc} + R_{PP}$$

$$= T_{PE,P0,acc} + S_{PE,P0,acc}|_{min} + R_{PP} \quad (6.130)$$

6.3.3.2. Computation in process P1

Let $C_{PE,P1,acc}$ denote the computation delay in process P1 for the accusations message. The computation delay is measured from the end of the deskewing window. Let $T_{PE,P1,C,acc}$ denote the local-time at which the computation process outputs the result.

$$T_{PE,P1,C,acc} = T_{PE,P1,RCV,E,acc} + W_{Deskew,post} + C_{PE,P1,acc} \quad (6.131)$$

6.3.4. Accusations message in stage 2: P1 to P2

Process P1 generates a set of accusations that are not necessarily related to the accusations received from P0. The processing of these accusations is analyzed independently. However, it is assumed that the constraint on the minimum separation between the stream of PE messages and the accusations message also applies here.

6.3.4.1. Communication between processes P1 and P2

Let $T_{PE,P1,acc}$ denote the time at which the accusations are available for transmission in process P1. $T_{PE,P1,C,sched,K-1}$ denotes the local time at which the Computation Process outputs the result for the last scheduled PE message. It is assumed that $T_{PE,P1,acc}$ occurs after the time at which the Computation Process outputs the result for the last PE message.

$$T_{PE,P1,acc} > T_{PE,P1,C,sched,K-1} \tag{6.132}$$

Let $\Delta_{PE,P1,acc\_ready}$ denote the delay from the time at which the Computation Process outputs the result for the last PE message to the time at which the accusations are available for transmission in process P1.

$$\Delta_{PE,P1,acc\_ready} = T_{PE,P1,acc} - T_{PE,P1,C,sched,K-1} \tag{6.133}$$

$S_{PE,P1,acc}$ denotes the Send Process delay for the accusations message in process P1, $T_{PE,P1,SND,acc}$ denotes the send time for the accusations message in process P1, $T_{PE,P2,RCV,E,acc}$ denotes the expected time of reception for the accusations message in process P2, and $T_{PE,P1,SND,sched,K-1}$ denotes the sent time for the last scheduled PE message. $T_{PE,P1,SND,acc}$ must be chosen to ensure that the constraint on the minimum stream separation is satisfied. Two cases must be considered.

Case 1: $T_{PE,P1,acc} + S_{PE,P1,acc}|_{min} < T_{PE,P1,SND,sched,K-1} + \Delta_{PE,sched-acc}|_{min}$

For this case, the accusations message must be buffered until it can be sent with the proper separation from the PE message stream. Let $\Delta_{PE,P1,BUF,acc}$ denote the buffering delay for the accusations at P1.

$$\Delta_{PE,P1,BUF,acc} = (T_{PE,P1,SND,sched,K-1} + \Delta_{PE,sched-acc}|_{min}) - (T_{PE,P1,acc} + S_{PE,P1,acc}|_{min}) \tag{6.134}$$

The send delay for accusations in process P1 is:

$$S_{PE,P1,acc} = S_{PE,P1,acc}|_{min} + \Delta_{PE,P1,BUF,acc} \tag{6.135}$$

So:

$$T_{PE,P1,SND,acc} = T_{PE,P1,acc} + S_{PE,P1,acc} \tag{6.136}$$

And:

$$\begin{aligned} T_{PE,P2,RCV,E,acc} &= T_{PE,P1,SND,acc} + R_{PP} \\ &= T_{PE,P1,acc} + S_{PE,P1,acc} + R_{PP} \end{aligned} \tag{6.137}$$

Case 2: $T_{PE,P1,acc} + S_{PE,P1,acc}|_{min} \geq T_{PE,P1,SND,sched,K-1} + \Delta_{PE,sched-acc}|_{min}$

For this case, the accusations message can be sent as soon as possible. The send delay is the minimum value.

$$S_{PE,P1,acc} = S_{PE,P1,acc}|_{min} \tag{6.138}$$

So:

$$T_{PE,P1,SND,acc} = T_{PE,P1,acc} + S_{PE,P1,acc}$$

$$= T_{PE,P1,acc} + S_{PE,P1,acc}|_{min} \quad (6.139)$$

And:

$$T_{PE,P2,RCV,E,acc} = T_{PE,P1,SND,acc} + R_{PP}$$

$$= T_{PE,P1,acc} + S_{PE,P1,acc}|_{min} + R_{PP} \quad (6.140)$$

6.3.4.2. Computation in process P2

Let $C_{PE,P2,acc}$ denote the computation delay in process P2 for the accusations message. The computation delay is measured from the end of the deskewing window. Let $T_{PE,P2,C,acc}$ denote the local-time at which the Computation Process outputs the result.

$$T_{PE,P2,C,acc} = T_{PE,P2,RCV,E,acc} + W_{Deskew,post} + C_{PE,P2,acc} \quad (6.141)$$

6.3.5. Duration of the protocol

Let $\Delta_{PE,P2,acc,C\text{-}END}$ denote the delay in process P2 from the end of the computation to the end of the PE Communication mode protocols. $\Delta_{PE,P2,acc,C\text{-}END}$ is assumed to be a constant independent of the system state.

Let $T_{PE,END}$ denote the local-time at which the execution of the PE Communication protocols ends.

$$T_{PE,END} = T_{PE,P2,C,acc} + \Delta_{PE,P2,acc,C\text{-}END} \quad (6.142)$$

Let Δ_{PE} denote the duration of the PE Communication protocol.

$$\Delta_{PE} = T_{PE,END} - T_{PE} \quad (6.143)$$

6.3.6. Bound on the number of PE messages

The execution of the PE Communication protocol must end at or before the time at which the execution of the Synchronization Preservation protocol is scheduled to begin. Let T_{SP} denote the local-time trigger for the execution of the Synchronization Preservation protocol. If no additional time is needed from the end of the PE Communication to the beginning of the Synchronization Preservation protocol, the following relation expresses the constraint on the duration of the PE Communication protocol.

$$T_{PE,END}|_{max} = T_{SP} \quad (6.144)$$

So:

$$\Delta_{PE}\,|_{max} = T_{SP} - T_{PE} \quad (6.145)$$

To compute the maximum number of PE messages that can be processed, we need to determine the time consumed in processing the accusation messages. $K_{PE,sched}|_{max}$ denotes the maximum number of scheduled PE messages that can be processed, and $\Delta_{PE,acc}$ denotes the time delay from the reference time for the transmission of the last scheduled PE message from P0 to P1 to the end of the protocol.

$$\Delta_{PE,acc} = T_{PE,END} - [T_{PE} + (K_{PE,sched} - 1)\Lambda_{PE,sched}]$$

$$= S_{PE,P0,sched} + R_{PP} + W_{Deskew,post} + C_{SU,P1,sched}$$

$$+ \Delta_{PE,P1,acc\_ready} + S_{PE,P1,acc} + R_{PP} + W_{Deskew,post} + C_{PE,P2,acc} + \Delta_{PE,P2,acc,C\text{-}END}$$

$$= (S_{PE,P0,sched} + S_{PE,P1,acc}) + 2(R_{PP} + W_{Deskew,post})$$

$$+ (C_{SU,P1,sched} + C_{PE,P2,acc}) + (\Delta_{PE,P1,acc\_ready} + \Delta_{PE,P2,acc,C\text{-}END}) \quad (6.146)$$

So:

$$K_{PE,sched}|_{max} = \lfloor (T_{SP} - T_{PE} - \Delta_{PE,acc})/\Lambda_{PE,sched} \rfloor + 1 \quad (6.147)$$

6.3.7. PE message latency

Let $\Delta_{PE,latency}$ denote the latency for PE messages. For this analysis, we define the PE message latency as the time from the reference time for the transmission of the i-th scheduled PE message by process P0 until the completion of the corresponding computation in process P2. So:

$$\Delta_{PE,latency} = T_{PE,P2,C,sched,i} - T_{PE,P0\text{-}P1,REF,sched,i} \quad (6.148)$$

$$= (S_{PE,P0,sched} + S_{PE,P1,sched}|_{min}) + 2(R_{PP} + W_{Deskew,post}) + (C_{PE,P1,sched} + C_{PE,P2,sched}) \quad (6.149)$$

6.4. Synchronization Preservation Protocol

Section 5 of this document examines the timing aspects of the Synchronization Preservation protocol.

6.5. Miscellaneous considerations

6.5.1. Time gap between Sync Reset and Collective Diagnosis

$\Delta_{CD,begin}$ denotes the time from the synchronization reset to the beginning of the Collective Diagnosis protocol, measured in local clock ticks.

$$\Delta_{CD,begin} = T_{CD} \quad (6.150)$$

If needed, $\Delta_{CD,begin}$ can be chosen to ensure that the delay from the send time of the ECHO message in Initial Synchronization or Synchronization Preservation to the send time of the first message in Collective Diagnosis satisfies the minimum data introduction interval for the Communication Module. At the RMUs, which are the last to send ECHO messages, this constraint corresponds to:

$$T_{CD,P0,\,SND} - (T_{P3,A} + B_{P3}) \geq \Lambda_{Comm} \quad (6.151)$$

Or:

$$H_{P3} - B_{P3} + T_{CD} + S_{CD,P0} \geq \Lambda_{Comm} \quad (6.152)$$

6.5.2. Time gap between Collective Diagnosis and Schedule Update

$\Delta_{SU,begin}$ denotes the time from the end of the Collective Diagnosis protocol to the beginning of the Schedule Update protocol, measured in local clock ticks.

$$\Delta_{SU,begin} = T_{SU} - T_{CD} - \Delta_{CD} \quad (6.153)$$

If needed, $\Delta_{SU,begin}$ can be chosen to ensure that the delay from the send time of the last Collective Diagnosis message to the send time of the first Schedule Update message satisfies the minimum data introduction interval for the Communication Module. At the BIUs, this constraint corresponds to:

$$T_{SU,P0,SND,0} - T_{CD,P3,SND} \geq \Lambda_{Comm} \quad (6.154)$$

At the RMUs, the constraint is:

$$T_{SU,P1,SND,0} - T_{CD,P3,SND} \geq \Lambda_{Comm} \quad (6.155)$$

6.5.3. Time gap between Schedule Update and PE Communication

$\Delta_{PE,begin}$ denotes the actual time delay from the end of the Schedule Update protocol to the beginning of the PE Communication protocol, measured in local clock ticks.

$$\Delta_{PE,begin} = T_{PE} - T_{SU} - \Delta_{SU} \quad (6.156)$$

If needed, $\Delta_{PE,begin}$ can be chosen to ensure that the delay from the send time of the last Schedule Update message to the send time of the first PE Communication message satisfies the minimum data introduction interval for the Communication Module. At the BIUs, this constraint corresponds to:

$$T_{PE,P0,SND,0} - T_{SU,P2,SND,N-1} \geq \Lambda_{Comm} \quad (6.157)$$

At the RMUs, this constraint is:

$$T_{PE,P1,SND,0} - T_{SU,P3,SND,N-1} \geq \Lambda_{Comm} \quad (6.158)$$

6.5.4. Time gap between PE Communication and Synchronization Preservation

$\Delta_{SP,begin}$ denotes the actual time delay from the end of the PE Communication protocol to the beginning of the Synchronization Preservation protocol, measured in local clock ticks.

$$\Delta_{SP,begin} = T_{SP} - T_{PE} - \Delta_{PE} \quad (6.159)$$

If needed, $\Delta_{SP,begin}$ can be chosen to ensure that the delay from the send time of the last PE Communication message to the send time of the first Synchronization Preservation message satisfies the minimum data introduction interval for the Communication Module. At the BIUs, this constraint corresponds to:

$$(T_{SP} + B_{SP,P0}) - T_{PE,P2,SND,N-1} \geq \Lambda_{Comm} \quad (6.160)$$

At the RMUs, this constraint is:

$$(T_{PE,P1,RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max} + A_{P1} + B_{P1}) - T_{PE,P3,SND,N-1} \geq \Lambda_{Comm} \quad (6.161)$$

7. Self-Test mode

This section examines the timing aspects of the Self-Test mode. The main objective is to determine a bound on the relative time skew at the end of this mode.

7.1. Bound on the relative time skew at the beginning of the Self-Test mode

An RPP enters the Self-Test mode for startup after receiving a power-on enable, or for restart after the detection of a local failure or a failure of the clique.

7.1.1. Power-one enable

It is assumed that the nodes enter the Self-Test mode immediately after power-on enable. δ_{POE} denotes the actual duration of the time interval within which the nodes are enabled measured in units of seconds. $\delta_{POE}|_{max}$ denotes the upper bound for δ_{POE}. π_{POE} denotes the upper bound on the relative time skew at power-on enable, measured in nominal clock ticks. τ_0 denotes the nominal duration of a clock tick measured in seconds. δ_{POE} and π_{POE} are related as follows:

$$\pi_{POE} = \delta_{POE}|_{max}/\tau_0 \tag{7.1}$$

7.1.2. Local failure or bus failure

δ_{FCP} denotes the actual duration of a fault-causing phenomenon measured in units of seconds. We assume that the duration of the fault-causing phenomenon as experienced by individual nodes can be effectively 0. ($\delta_{FCP} = 0$ means that the phenomenon has a negligibly small duration, not that the phenomenon has no effect.)

$$\delta_{FCP}|_{min} = 0 \tag{7.2}$$

$\delta_{FCP}|_{max}$ depends on the characteristics of the fault-causing phenomenon for which the design is targeted.

Δ_{FD} denotes the actual duration of the failure-detection delay measured in local clock ticks. We assume that it is possible for a node to detect a failure condition immediately. $\Delta_{FD}|_{max}$ is implementation-dependent. δ_{FD} denotes the actual duration of the failure-detection delay measured in nominal clock ticks.

$$\delta_{FD}|_{min} = 0 \tag{7.3}$$

$$\delta_{FD}|_{max} = (1 + \rho_0)\Delta_{FD}|_{max} \tag{7.4}$$

Let $t_{FCP,0}$ denote the time at which the fault-causing phenomenon begins. Let $t_{restart,l}$ and $t_{restart,h}$ denote the earliest and latest times, respectively, at which nodes affected by the fault-causing phenomenon enter the Self-Test mode.

$$t_{restart,l} = t_{FCP,0} + \delta_{FCP}|_{min} + \delta_{FD}|_{min}$$

$$= t_{FCP,0} \tag{7.5}$$

And:

$$t_{restart,h} = t_{FCP,0} + \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{7.6}$$

Let $\pi_{restart}$ denote the upper bound on the relative time skew when entering the Self-Test mode for restart, measured in nominal clock ticks.

$$\pi_{restart} = t_{restart,h} - t_{restart,l}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{7.7}$$

7.2. Duration of the Self-Test mode

The **Local Upset Abatement Delay** (LUAD) for a transient-fault scenario is defined as the delay from the time the fault-causing phenomenon reaches a node until the node has regained control of its local operation. Local regaining of control is assumed to occur after the node has detected the failure condition, at which time the node disables its broadcast outputs and transitions to the Self-Test mode. In the Self-Test mode, the node first performs a full local reset and then begins the execution of the self-test procedure. This local reset activity should cover the Communication and Computation modules. The duration of the reset is implementation-dependent. Let δ_{LUAD} denote the actual duration of the Local Upset Abatement Delay, measured in units of nominal clock ticks.

$$\delta_{LUAD}|_{max} = t_{restart,h} - t_{FCP,0}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{7.8}$$

The **Observed Upset Abatement Delay** (OUAD) for a transient-fault scenario is defined as the delay from the time the fault-causing phenomenon begins until the affected nodes can be consistently recognized by all direct observers as being untrustworthy. Note that this delay is defined with respect to the effects perceived by the observing nodes. Thus, the message reception delay must be taken into consideration. Let δ_{OUAD} denote the actual duration of the Observed Upset Abatement Delay, measured in units of nominal clock ticks.

$$\delta_{OUAD}|_{max} = \delta_{LUAD}|_{max} + r_{PP,h}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} + r_{PP,h} \tag{7.9}$$

Δ_{STM} denotes the duration of the Self-Test mode for a ROBUS node, measured in units of local clock ticks. Δ_{STM} is assumed constant. The duration of the Self-Test mode must satisfy the timing requirements for the expected transient-fault scenarios. To increase the probability that a restarting node does not trust an affected node, we require that the restarting nodes exit the Self-Test mode only after the latest time at which affected nodes can be incorrectly diagnosed as trustworthy.

$$t_{restart,l} + \Delta_{STM}/(1 + \rho_0) \geq t_{restart,h} + r_{PP,h}$$

$$\Delta_{STM}/(1 + \rho_0) \geq \pi_{restart} + r_{PP,h}$$

$$\Delta_{STM}/(1 + \rho_0) \geq \delta_{OUAD}|_{max} \quad (7.10)$$

In terms of local clock ticks, the above inequality corresponds to the following constraint:

$$\Delta_{STM} \geq \lceil (1 + \rho_0)\delta_{OUAD}|_{max} \rceil \quad (7.11)$$

7.3. Bound on the relative local-time skew at the end of the Self-Test mode

π_{STM} denotes the upper bound on the relative time skew at the end of the Self-Test mode, measured in nominal clock ticks.

$$\pi_{STM} = \max(\pi_{POE}, \pi_{restart}) + [(1 + \rho_0) - 1/(1 + \rho_0)]\Delta_{STM} \quad (7.12)$$

8. Clique Detection mode

The Clique Detection mode is composed of three main processes: Local Diagnosis Acquisition, Synchronization Acquisition, and Collective Diagnosis Acquisition.

8.1. Local Diagnosis Acquisition

Local Diagnosis Acquisition (a.k.a., Preliminary Diagnosis) is composed of two consecutive observation intervals, each with a duration at least as large as a resynchronization interval. $\Delta_{PD,begin}$ denotes the delay from the time a node exits the Self-Test mode until the beginning of the first observation interval, measured in local-clock ticks. The value of $\Delta_{PD,begin}$ is determined by the implementation and assumed constant.

8.1.1. Bound on the duration of an observation phase

It is assumed that, at the earliest, a node in Local Diagnosis Acquisition can detect the absence of a valid clique as soon as it enters the observation phase. $\Delta_{PD,OW}$ denotes the duration of the observation intervals (or "windows"), measured in local-clock ticks. The value of $\Delta_{PD,OW}$ is determined by the implementation and assumed constant. $\Delta_{PD,OW}$ should be large enough to cover the duration of a resynchronization cycle measured in local-clock ticks.

$$\Delta_{PD,OW} \geq P \tag{8.1}$$

P is the nominal resynchronization period given in Section 5 of this document.

8.1.2. Bound on the duration of Local Diagnosis Acquisition

Let δ_{PD} denote the actual duration of Local Diagnosis Acquisition measured in nominal clock ticks.

$$\delta_{PD}|_{min} = \Delta_{PD,begin}/(1+\rho_0) \tag{8.2}$$

$$\delta_{PD}|_{max} = (1+\rho_0)(\Delta_{PD,begin} + 2\Delta_{PD,OW}) \tag{8.3}$$

8.2. Synchronization Acquisition

Synchronization Acquisition is composed of the Frame Synchronization and Synchronization Capture protocols. Synchronization Acquisition ends with the synchronization reset, at which point the local time is set to 0.

8.2.1. Frame Synchronization

It is assumed that a node can detect the absence of a valid clique at any time during Synchronization Acquisition. $\Delta_{FS,begin}$ denotes the delay from the end of the second observation window during Local

Diagnosis Acquisition to the beginning of the Frame Synchronization protocol during Synchronization Acquisition, measured in local clock ticks. $\Delta_{FS,begin}$ is implementation-dependent and assumed constant.

Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in local clock ticks. $\Delta_{FS}|_{max}$ is given in Section 5 of this document. δ_{FS} denotes the actual duration of the Frame Synchronization protocol measured in nominal clock ticks.

$$\delta_{FS}|_{max} = (1 + \rho_0)\Delta_{FS}|_{max} \tag{8.4}$$

8.2.2. Synchronization Capture

We assume that the Synchronization Capture protocol begins immediately after the Frame Synchronization protocol is complete. We want to determine a bound on the duration of Synchronization Capture protocol.

δ_{SC} denotes the actual duration of the execution of the Synchronization Capture protocol measured in nominal clock ticks. The execution of the protocol may begin shortly after the ECHO messages are transmitted by the clique during the execution of the Synchronization Preservation protocol. In that case, the end of the Synchronization Capture protocol would occur after the reset is applied during the next execution of the Synchronization Preservation protocol. To specify a bound for the duration of the Synchronization Capture protocol, we consider an interval containing two consecutive executions of the Synchronization Preservation protocol. $\delta_{SP}|_{max}$ denotes the upper bound on the real-time duration of the execution of the Synchronization Preservation protocol. $\delta_{SP}|_{max}$ is given in Section 5 of this document. T_{SP} denotes the scheduled local time at which the execution of the Synchronization Preservation protocol begins.

$$\begin{aligned}\delta_{SC}|_{max} &= p_{max} + \delta_{SP}|_{max} \\ &= (1 + \rho_0)T_{SP} + 2\delta_{SP}|_{max}\end{aligned} \tag{8.5}$$

8.2.3. Bound on the duration of Synchronization Acquisition

Let δ_{SA} denote the actual duration of the execution of Synchronization Acquisition measured in nominal clock ticks.

$$\delta_{SA}|_{max} = (1+\rho_0)\Delta_{FS,begin} + \delta_{FS}|_{max} + \delta_{SC}|_{max} \tag{8.6}$$

Δ_{SA} denotes the actual duration of Synchronization Acquisition measured in local clock ticks. We want to ensure that a count of $\Delta_{SA}|_{max}$ local ticks takes no fewer than $\delta_{SA}|_{max}$ nominal ticks.

$$\Delta_{SA}|_{max}/(1+\rho_0) \geq \delta_{SA}|_{max} \tag{8.7}$$

We choose the minimum value that satisfies that constraint.

$$\Delta_{SA}|_{max} = \lceil (1+\rho_0)\delta_{SA}|_{max} \rceil \tag{8.8}$$

8.3. Bound on the duration of the Clique Detection mode

Synchronization Acquisition ends with the synchronization reset, at which point the local time is set to 0. From that point on, the local time should be synchronized to the clique in Preservation mode. The delays to begin and complete the Collective Diagnosis Acquisition protocol in the Clique Detection mode are the same as for the Collective Diagnosis protocol in Clique Preservation mode. $\Delta_{CD,begin}$ denotes the time from the synchronization reset to the beginning of the Collective Diagnosis protocol, measured in local clock ticks. Δ_{CD} denotes the time to complete the execution of the Collective Diagnosis protocol in local clock ticks. The transition to the Clique Join mode occurs at the beginning of execution of the Schedule Update protocol. Before that point, a detected failure attributable to the absence of a clique results in a transition to the Clique Initialization mode. $\Delta_{SU,begin}$ denotes the time from the end of the Collective Diagnosis protocol to the beginning of the Schedule Update protocol, measured in local clock ticks. $\Delta_{CD,begin}$, Δ_{CD}, and $\Delta_{SU,begin}$ are implementation-dependent and determined by the time-indexed operation schedule specifying the timing for bus activities.

After detecting the absence of a valid clique, a node clears its state and transitions to the Clique Initialization mode. $\Delta_{CDM\text{-}CIM}$ denotes the delay to transition to the Clique Initialization mode after detecting the absence of a valid clique, measured in units of local clock ticks. $\Delta_{CDM\text{-}CIM}$ is implementation-dependent and assumed constant. δ_{CDM} denotes the actual duration of the Clique Detection mode for a ROBUS node, measured in units of nominal clock ticks.

$$\delta_{CDM}|_{min} = \delta_{PD}|_{min} + \Delta_{CDM\text{-}CIM}/(1 + \rho_0) \tag{8.9}$$

$$\delta_{CDM}|_{max} = \delta_{PD}|_{max} + \delta_{SA}|_{max} + [(1+\rho_0)\,(\Delta_{CD,begin} + \Delta_{CD} + \Delta_{SU,begin} + \Delta_{CDM\text{-}CIM})] \tag{8.10}$$

9. Clique Initialization mode

This section examines timing aspects related to the Clique Initialization mode. Only Initial Diagnosis and Initial Synchronization are discussed. The operation during Collective Diagnosis is the same as during Clique Preservation mode, which is discussed in Section 6.

9.1. Bound on the relative time skew at the beginning of the Clique Initialization mode

$\pi_{CIM,BEGIN}$ denotes the upper bound on the relative time skew at the beginning of the Clique Initialization mode, measured in nominal clock ticks.

$$\pi_{CIM,BEGIN} = \pi_{STM} + (\delta_{CDM}|_{max} - \delta_{CDM}|_{min}) \quad (9.1)$$

π_{STM} is defined in Section 7, and $\delta_{CDM}|_{max}$ and $\delta_{CDM}|_{min}$ are defined in Section 8 of this document.

9.2. Initial Diagnosis

To simplify the presentation, we would like to compute a single upper bound for the relative local-time skew during the execution of the Initial Diagnosis and Initial Synchronization protocols. Let π_{ID+IS} denote that bound, measured in nominal clock ticks.

Figure 9.1 shows the message flow graph for Initial Diagnosis. BIUs and RMUs are assumed to have the same timing characteristics. The analysis presented here does not refer to the kind of the node sending or receiving messages for any of the protocol processes.

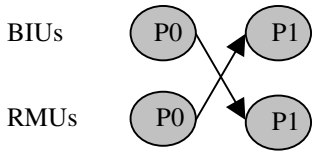


Figure 9.1: Message flow graph for the Initial Diagnosis protocol

$\Delta_{ID,begin}$ denotes the delay from the time a node enters the Clique Initialization mode until the time it begins the execution of the Initial Diagnosis protocol, measured in units of local clock ticks. The value of $\Delta_{ID,begin}$ is determined by the implementation and assumed constant.

9.2.1. Communication between processes P0 and P1

The following variables are defined: T_{ID} denotes the local time triggering the execution of the Initial Diagnosis protocol; $T_{ID,P0\text{-}P1,REF}$ denotes the reference time for the communication between processes P0 and P1; $T_{ID,P0,SND}$ denotes the time at which process P0 sends the message; $T_{ID,P1,RCV,E}$ denotes the expected time of reception in process P1; $S_{ID,P0}$ denotes the Send Process delay for process P0; $\Delta_{ID,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the reception window in process P1; R_{PP} denotes the nominal point-to-point reception delay; $W_{ID,Deskew}$ denotes the size

of the deskewing window in process P1; $W_{ID,Deskew,pre}$ denotes the pre-expectation window in process P1 (i.e., the size of the section of the deskewing window before the expected time of reception); $W_{ID,Deskew,post}$ denotes the post-expectation window in process P1 (i.e., the size of the section of the deskewing window after the expected time of reception); $\Delta_{ID,PP,RCV}|_{abs\text{-}max}$ denotes the absolute value of the maximum error in the actual time of reception in process P1 for a good source-receiver pair; $C_{ID,P1}$ denotes the computation delay in process P1 (The computation delay is measured from the end of the deskewing window. $C_{CD,P1}$ is assumed constant.); $\Delta_{ID,P1,C\text{-}END}$ denotes the delay in process P1 from the end of the computation to the end of the execution of the Initial Diagnosis protocol ($\Delta_{ID,P1,C\text{-}END}$ is assumed constant.); and Δ_{ID} denotes the duration of the execution of the Initial Diagnosis protocol.

T_{ID} is the reference time for the communication between processes P0 and P1. Given that the local time is reset at the start of the Clique Initialization mode, then:

$$T_{ID,P0\text{-}P1,REF} = T_{ID} = \Delta_{ID,begin} \tag{9.2}$$

To determine $W_{ID,Deskew}$, we need the maximum error in the expected time of reception for the Initial Diagnosis protocol messages, $\Delta_{ID,PP,RCV}|_{abs\text{-}max}$. Based on the analysis presented for point-to-point communication:

$$\Delta_{ID,PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \tag{9.3}$$

$\mu_{PP,l}$ and $\mu_{PP,h}$ are given in Section 4 of this document. For the deskewing window:

$$W_{ID,Deskew} = 2\Delta_{ID,PP,RCV}|_{abs\text{-}max} + 1 \tag{9.4}$$

$$W_{ID,Deskew,pre} = \Delta_{ID,PP,RCV}|_{abs\text{-}max} \tag{9.5}$$

$$W_{ID,Deskew,post} = \Delta_{ID,PP,RCV}|_{abs\text{-}max} + 1 \tag{9.6}$$

We expect the upper bound on the relative local-time skew during the execution of the protocol to be much larger than any minimum timing constraints associated with the process of communication. Based on this, we assume that the following condition holds for the communication between processes P0 and P1.

$$S_{ID,P0}|_{min} + R_{PP} < \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} \tag{9.7}$$

For this case:

$$S_{ID,P0} = \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} - R_{PP} \tag{9.8}$$

And:

$$\Delta_{ID,P1,RCVWND} = \Delta_{ID,P1,RCVWND}|_{min} \tag{9.9}$$

So:

$$\begin{aligned} T_{ID,P0,SND} &= T_{ID,P0\text{-}P1,REF} + S_{ID,P0} \\ &= T_{ID} + \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} - R_{PP} \end{aligned} \tag{9.10}$$

And:

$$T_{ID,P1,RCV,E} = T_{ID,P0,SND} + R_{PP}$$

$$= T_{ID} + \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} \quad (9.11)$$

9.2.2. Bound on the duration of the Initial Diagnosis protocol

Let $T_{ID,P1,C}$ denote the local-time at which the Computation Process outputs the result for process P1.

$$T_{ID,P1,C} = T_{ID,P1,RCV,E} + W_{ID,Deskew,post} + C_{ID,P1} \quad (9.12)$$

Let $T_{ID,P1,END}$ denote the local-time at which the execution of the Initial Diagnosis protocol ends.

$$T_{ID,P1,END} = T_{ID,P1,C} + \Delta_{ID,P1,C\text{-}END} \quad (9.13)$$

The duration of the execution of the Initial Diagnosis protocol is:

$$\Delta_{ID} = T_{ID,P1,END} - T_{ID}$$

$$= \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END} \quad (9.14)$$

9.3. Initial Synchronization

Let T_{IS} denote the local time triggering the execution of the Initial Synchronization protocol. $\Delta_{IS,begin}$ denotes the delay from the end of Initial Diagnosis to the beginning of Initial Synchronization, measured in units of local clock ticks. The value of $\Delta_{IS,begin}$ is determined by the implementation and assumed constant.

$$\Delta_{IS,begin} = T_{IS} - T_{ID,P1,END} \quad (9.15)$$

9.3.1. Bound on the relative skew at the beginning of the Initial Synchronization protocol

Let $\pi_{IS,BEGIN}$ denote the upper bound on the relative local-time skew at the beginning of the Initial Synchronization protocol, measured in nominal clock ticks.

$$\pi_{IS,BEGIN} = \pi_{CIM,BEGIN} + [(1 + \rho_0) - 1/(1 + \rho_0)](\Delta_{ID,begin} + \Delta_{ID} + \Delta_{IS,begin}) \quad (9.16)$$

9.3.2. Communication between processes P0 and P1

This is discussed in Section 5 of this document. There, π_{IS} denotes the bound on the relative skew during the execution of the Initial Synchronization protocol. Thus:

$$\pi_{IS} = \pi_{ID+IS} \quad (9.17)$$

9.3.3. Bound on the duration of the Initial Synchronization protocol

$\delta_{IS}|_{max}$ denotes the upper bound on the real-time duration of the execution of the Initial Synchronization protocol measured from the earliest time at which a node begins executing the protocol to the latest time at which a node applies the synchronization reset. $\delta_{IS,sync}|_{max}$, $\Delta_{IS,P2,H,h}$, $\Delta_{IS,P3,H,h}$, and $\Delta_{IS,P4,H,h}$ are given by $\delta_{sync}|_{max}$, $\Delta_{sync,P2,H,h}$, $\Delta_{sync,P3,H,h}$, and $\Delta_{sync,P4,H,h}$ in Section 5 of this document with B_{P0} replaced by $B_{IS,P0}$.

$$\delta_{IS}|_{max} = \pi_{IS,BEGIN} + \max(\Delta_{IS,P2,H,h} , \Delta_{IS,P3,H,h} , \Delta_{IS,P4,H,h}) \tag{9.18}$$

Let $\Delta_{IS}|_{max}$ denotes the upper bound on the duration of the execution of the Initial Synchronization protocol measured in local clock ticks. We want the fastest count of $\Delta_{IS}|_{max}$ ticks to be greater than or equal to $\delta_{IS}|_{max}$.

$$\Delta_{IS}|_{max}/(1 + \rho_0) \geq \delta_{IS}|_{max} \tag{9.19}$$

We choose the following value for $\Delta_{IS}|_{max}$.

$$\Delta_{IS}|_{max} = \lceil (1 + \rho_0)\delta_{IS}|_{max} \rceil \tag{9.20}$$

9.4. Bound on the relative skew during Initial Diagnosis and Initial Synchronization

The bound on the relative local-time skew during Initial Diagnosis and Initial Synchronization is:

$$\pi_{ID+IS} = \pi_{IS,BEGIN} + [(1 + \rho_0) - 1/(1 + \rho_0)]\delta_{IS}|_{max} \tag{9.21}$$

The following variables are defined in order to simplify the expressions presented below.

$$X_{IS,H,h} = \max(\Delta_{IS,P2,H,h}, \Delta_{IS,P3,H,h}, \Delta_{IS,P4,H,h}) \tag{9.22}$$

$$\sigma_0 = [(1 + \rho_0) - 1/(1 + \rho_0)] \tag{9.23}$$

Then:

$$\begin{aligned}
\pi_{ID+IS} &= \pi_{IS,BEGIN} + \sigma_0(\pi_{IS,BEGIN} + X_{IS,H,h}) \\
&= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)\pi_{IS,BEGIN} \\
&= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{ID} + \Delta_{IS,begin})] \\
&= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})] + \sigma_0(1 + \sigma_0)\Delta_{ID}
\end{aligned} \tag{9.24}$$

The following inequality holds for $W_{ID,Deskew}$:

$$W_{ID,Deskew} \leq 2(1 + \rho_0)[\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})] + 1 \tag{9.25}$$

Applying this inequality to Δ_{ID}, then:

$$\pi_{ID+IS} \leq \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})]$$

$$+ \sigma_0(1 + \sigma_0)\{\Delta_{ID,P1,RCVWND}|_{min} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END}$$

$$+ [2(1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})) + 1]\} \quad (9.26)$$

Again, the definition of the following variable simplifies the presentation.

$$Y = \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})]$$

$$+ \sigma_0(1 + \sigma_0)(\Delta_{ID,P1,RCVWND}|_{min} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END}) \quad (9.27)$$

So:

$$\pi_{ID+IS} \leq Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})) + 1]$$

$$\pi_{ID+IS} \leq \{Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)\max(\mu_{PP,l}, \mu_{PP,h}) + 1]\}/\{1 - 2\sigma_0(1 + \sigma_0)(1 + \rho_0)\} \quad (9.28)$$

We choose the right side of this expression as the value for π_{ID+IS}.

$$\pi_{ID+IS} = \{Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)\max(\mu_{PP,l}, \mu_{PP,h}) + 1]\}/\{1 - 2\sigma_0(1 + \sigma_0)(1 + \rho_0)\} \quad (9.29)$$

10. RPP requirements

ROBUS-2 is a developmental version of ROBUS intended for laboratory experimentation and capability demonstrations. The RPP is the component that realizes the characteristic functionality of ROBUS. The ROBUS-2 RPP shall implement the behavior summarized in Section 2 of this document and described in detail in [Torres 05]. The design of the RPP shall comply with the following additional requirements.

- The Self-Test major mode will not include an actual self-test of the RPP or the FCR. The main reason for this requirement is to simplify the design of the RPP. The exclusion of self tests increases the likelihood that a faulty node will exit the Self-Test mode and eventually send bad messages to trustworthy nodes. Given the fault tolerance capabilities of ROBUS, the resulting increase in the probability of system failure should be small. That is an acceptable trade-off for this version of the RPP.

- Input-error monitoring shall be performed concurrently with the execution of the Frame Synchronization, Synchronization Capture, and Initial Synchronization protocols. This monitoring shall be independently performed for each opposite-kind node. Error detection shall be based on the validity of the received message sequence in the context of the protocols being executed. The detection of an error for a particular input source shall result in an immediate accusation.

- The error syndromes generated by the Frame Synchronization protocol (see [Torres 05]) shall be considered in the determination of the set of eligible voters for the Synchronization Capture protocol. This requirement is meant to ensure that the Synchronization Capture protocol is started with the latest available diagnostic information.

- An error check shall be implemented to corroborate the validity of the eligible voters latched at the beginning of the execution of the Synchronization Capture and Initial Synchronization protocols. These protocols are defined with the assumption that a majority of the latched eligible voters are trustworthy. This check shall consist of a comparison of the number of eligible voters at the beginning of the protocol execution against the number of eligible voters at the end. A failure shall be declared if a majority of the initial eligible voters are not in the final set of eligible voters. The determination of the final set of eligible voters shall include the error syndromes generated by the concurrent input-error monitors and the cross-lane checks described in the definition of the protocols (see [Torres 05]).

- A timeout check shall be performed for the Synchronization Acquisition and Initial Synchronization protocols. These protocols depend on receiving the expected synchronization messages from a majority of the eligible voters in order to complete their execution. A timeout error shall result in the immediate termination of the protocol execution and the declaration of a failure.

- A timeout check shall be performed for the resynchronization period. The synchronization period is the time interval between consecutive synchronization-reset events at a BIU or RMU node generated by synchronization protocols. Three sequences of synchronization protocols define resynchronization intervals: from Synchronization Capture to Synchronization Preservation, from Initial Synchronization to Synchronization Preservation, and from one execution of Synchronization Preservation to the next.

- The RPP shall be capable of processing Schedule Update messages with a data-introduction interval greater than or equal to one local oscillator clock tick.

- For the PE Broadcast protocol, all the BIUs shall broadcast a message for each execution of the protocol. The scheduled BIU shall broadcast a message from its attached PE, and all other BIUs shall broadcast a diagnostic message corresponding to the current local accusations against RMU nodes.

- The RPP shall be capable of processing PE Broadcast messages with a data-introduction interval greater than or equal to one local oscillator clock tick.

- For BIUs and RMUs the time interval between the last scheduled PE Broadcast message and the message for the Accusation Exchange protocol shall be the same as the data-introduction interval for the PE Broadcast messages. So, if there are K_{sched} scheduled PE messages, each BIU and RMU is expected to transmit $K_{sched} + 1$ messages with a constant DII during the PE Communication mode.

- The RPP shall be able to behave as either a BIU or an RMU. The desired node kind shall be selectable at the RPP external interface.

- The RPP external interface shall include input reset and enable control signals, as well as an output signal indicating when the RPP has detected a local or bus failure. This feature enables the implementation of a coordinated reset for all the circuitry within a particular FCR.

- The node identification number shall be selectable at the RPP external interface.

- The RPP shall be described in the VHDL hardware description language.

- The RPP VHDL description shall be highly parameterized in the behavioral and structural domains. This feature enables the reuse of the RPP description for a wide range of ROBUS-2 system characteristics.

- The synthesized RPP shall fit on a small to medium size FPGA (Field-Programmable Gate Array).

11. RPP design description

This section describes the design of the ROBUS-2 protocol processor, including insight into the synthesis process, the top-level design, and the design of the RPP sub-units.

11.1. High-level design concepts

This section presents some of the ideas that influenced the design of the RPP.

11.1.1. Functional partitions

The organization of the RPP is determined by the partitioning of functions along several lines. The first is a separation of operational and diagnostic functions. As illustrated in Figure 11.1, the RPP is composed of two separate but coordinated systems. The **operational system** handles the communication and computation activities required by the distributed protocols, in addition to all the processing associated with the mode logic, local time, and PE communication schedule. The **diagnostic system** monitors the operational system and provides timely information for reconfiguration and error containment. To support the fault-tolerance mechanisms, the diagnostic processes must work in close coordination with the operational system processes. The basic functions of the diagnostic system are to detect errors during the execution of the protocols, assess the status of individual nodes, and assess the status of the bus.

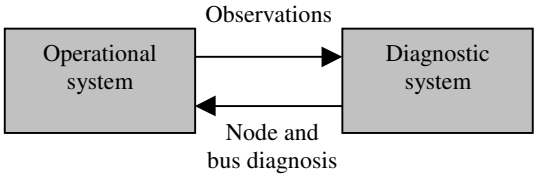


Figure 11.1: RPP functional partition: operations and diagnostics

The RPP is further partitioned into reception and transmission functions, as illustrated in Figure 11.2. The ROBUS protocols require that the nodes be able to simultaneously send and receive messages. The data handled by the nodes includes node diagnoses, local-time synchronization events, PE-communication schedule messages, and PE messages. For all the protocols, the reception of messages is followed by a computation function determined by the protocol processes being executed. The computation results can be passed forward for transmission or stored locally, depending on the protocol being executed. The reception and transmission functions must be coordinated with respect to timing events or the local time, depending on whether a synchronization protocol or a synchronous protocol is being executed.

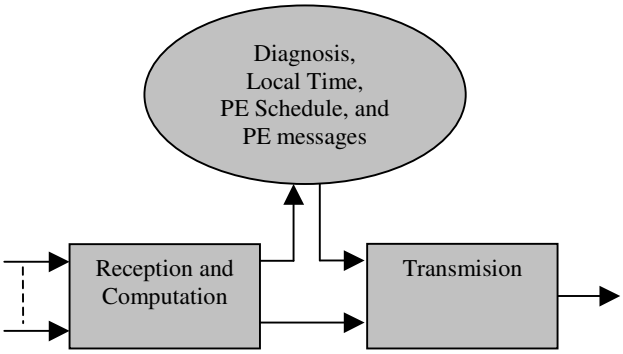


Figure 11.2: RPP functional partition: reception and transmission

Most of the ROBUS protocols exhibit a high degree of symmetry in the behavior of BIUs and RMUs. One of the main distinguishing factors between BIUs and RMUs is that the BIUs have external connections to the PEs as well as to the RMUs, while the RMUs communicate only with the BIUs. Since the same RPP design must work as either a BIU or an RMU, the RPP must include functionality to communicate with the PEs. As shown in Figure 11.3, the RPP is partitioned into bus functions and PE interfacing functions. The bus-functions section handles all the functionality for communication between BIUs and RMUs, and its implementation exploits the inherent symmetry of the protocols. The PE interface is active only when the RPP performs as a BIU.

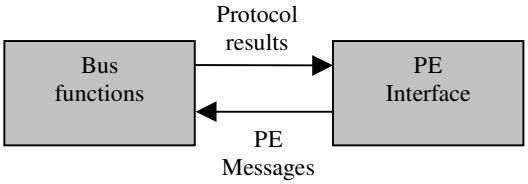


Figure 11.3: RPP functional partition: bus functions and PE interface

The activities of the bus are organized in a hierarchy with the following levels: **major mode**, **minor mode**, **protocol**, **(protocol) process**, and **(process) step**. The major and minor modes are high-level, node management states that specify how a node is supposed to interact with the other nodes on the bus. The transitions for major and minor modes depend on the local time, computed synchronization events, and the diagnostic assessment of the local node and the clique. For a given value of the major and minor modes, a node must execute a particular protocol or sequence of protocols. The implementation of the protocols requires intricate signaling sequences to control the operation of the functional units that perform the actual data processing. As illustrated in Figure 11.4, the RPP control functions are partitioned into two groups. The mode control section handles the high-level control functions. The protocol control section manages the execution of the protocols. The command from the mode control section includes all the high-level state information needed to fully specify the protocol to be executed and the diagnostic operations to be performed.

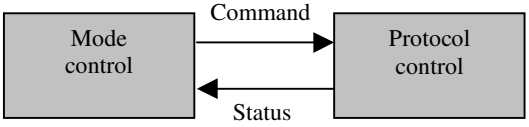


Figure 11.4: RPP functional partition: mode and protocol control

For most protocols executed in more than one major mode, the required processing is the same or differs only slightly. Many of these major-mode-dependent variations affect the diagnostic system, not the operational system. In addition, examination of the required processing for the major and minor mode sequences reveals that they can be reduced to a simple protocol execution pattern, as shown in Figure 11.5. This cyclic pattern with two entry paths, one to merge with an existing clique and the other to form a new clique, is the result of the required service delivery sequence and the chosen diagnostic policy for ROBUS-2. The transitions to the Collective Diagnosis protocol occur after the synchronization of the local time. Once inside the loop, the execution of protocols is triggered by the local time. The blocks in Figure 11.5 correspond to the redefined minor modes as actually implemented on the RPP. The protocol section of the RPP executes all the processing activities corresponding to a minor mode before being ready to receive the next command. In addition to the minor mode, the command from the mode control section must include all other mode-relevant information needed to properly execute the protocols.

11.1.2. Distributed pipelining

According to the requirements, the RPP must be capable of processing messages with a DII of one or greater (i.e., a maximum message throughput of one message per local clock tick) for the Schedule Update and PE Communication modes. The actual DII is a behavioral parameter specified before synthesis. The selection of the DII depends on factors like the performance of the PE-BIU links, the throughput capacity of the communication links between BIUs and RMUs, the bit error rate of the links, and desired timing-error coverage at the receiving end of the links. Notice that as the DII is decreased, the reception intervals over which messages are expected to arrive will get closer and eventually overlap. The input-timing checks and their effectiveness are necessarily impacted by an overlap in the reception intervals. This must be taken into consideration in the design of the RPP.

The Schedule Update, PE Broadcast, and Accusation Exchange protocols are synchronous protocols. The timing of execution of the synchronous protocols is specified by a time-indexed operation schedule. The scheduling of operations is based on a **distributed synchronous composition** abstract model of the system in which a single oscillator drives a common local-time clock and the fixed-delay processes corresponding to the communication and computation operations of the BIUs and RMUs (see [Torres 05]).

For this version of the RPP, the local-time clock and the message processing logic are driven by the same physical oscillator. To meet the DII requirement, the processing is pipelined with a stage delay of one clock tick. Because of the deskewing function required for the synchronous communication, the actual input-to-output processing delay for any particular message depends on its time of arrival. However, the delay from the time a message is read from the input buffer until its processing is complete is fixed and determined only by the number of pipeline stages. The reading of messages from the input buffers is a time-triggered operation.

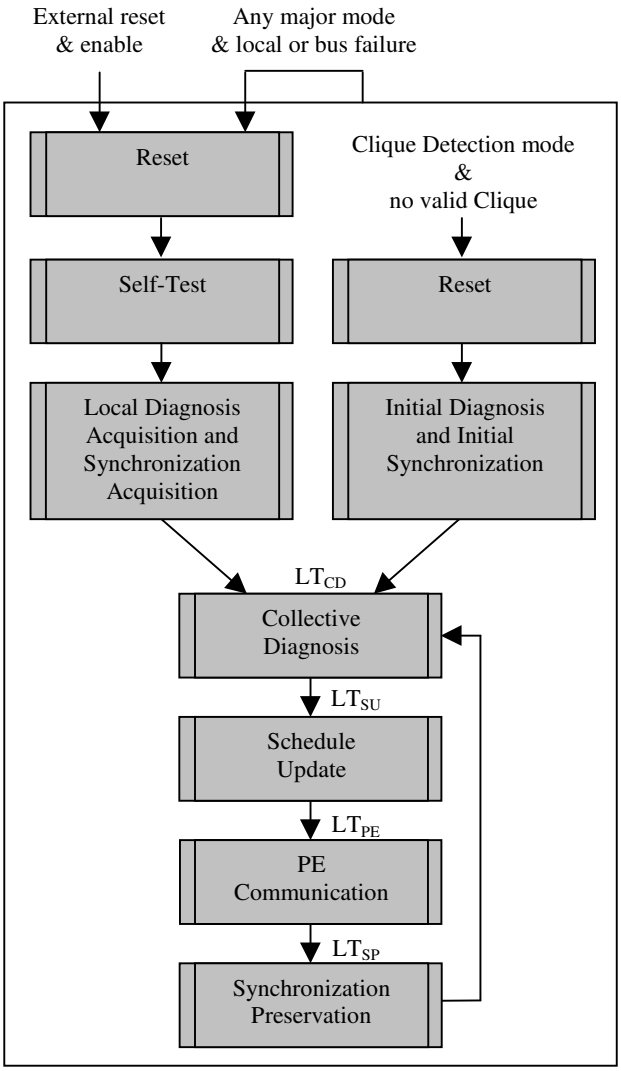


Figure 11.5: Minor-mode command transitions

11.2. RPP top-level design

This section describes the top-level design of the RPP.

11.2.1. Block diagram

Figure 11.6 shows the block diagram for the RPP. The **Mode Control Unit** (MCU) handles the high-level control functions and the interaction with other control logic within the same FCR. The **Input Unit** (IU) handles the reception of messages, implements the Frame Synchronization protocol, and controls the timing of data introduction for the computation pipeline. The **Input Diagnostics Unit** (IDU) checks the content of the received messages and performs the asynchronous monitoring function for the Clique Detection and Clique Initialization modes. The **Route and Vote Unit** (RVU) performs the actual computation, including routing scheduled messages for the PE Broadcast protocol and dynamic voting for other protocols. The **Node Diagnostics Unit** (NDU) performs the diagnostic assessment of nodes based

on error syndromes generated by the IU, IDU, and RVU. The **Output Unit** (OU) handles the transmission of messages. The **Status Monitoring Unit** (SMU) assesses the status of the local node and the bus. The **PE Input Unit** (PE_IU) and the **PE Output Unit** (PE_OU) handle the communication with the PE.

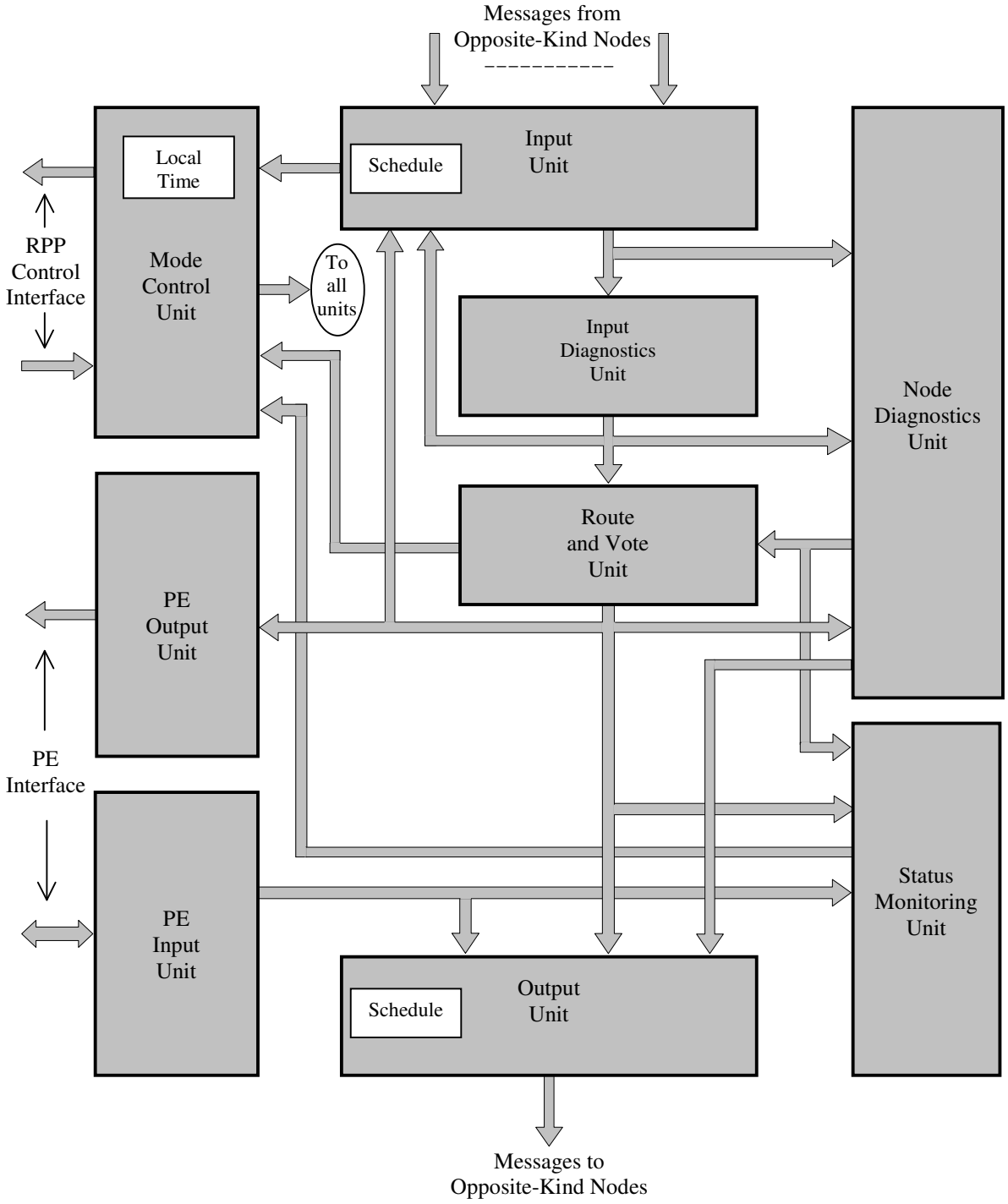


Figure 11.6: RPP block diagram

The MCU, IU, and OU modules have time-driven control functions triggered by the local time or by computed synchronization events. The IU and OU include independent schedule processing functions that store the computed schedule and perform the assessment. The IU reports the result of the schedule assessment to the MCU.

11.2.2. Interface

Figure 11.7 shows the VHDL entity declaration for the top level module of the ROBUS Protocol Processor. Signal **CLK_i** is the oscillator-clock signal. The "External inputs" signals correspond to the RPP Control Interface. Signal **Reset_In_i** forces a synchronous reset when set TRUE. **Failure_In_i** is coordinated with Reset_In_i and is set TRUE to indicate that the reset is due to an FCR failure. Signal **Enable_In_i** can be used to hold the RPP in an idle state after Reset_In_i is asserted. When Enable_In_i is asserted, the RPP becomes operational and Enable_In_i is ignored until the next time Reset_In_i is asserted. Signals **Node_Kind_In_i** and **Node_Id_In_i** specify the kind (i.e., RMU or BIU) and identification number. **Failure_Out_o** is asserted when the RPP detects an internal fault or a failure of the bus.

Signal **ROBUS_Input_Vect_In_i** is a VHDL record array with one record for each node of the opposite kind. Each record has three fields: Strobe, Message, and Synd. Field **Strobe** is asserted every time a new message is received. Field **Message** has the content of the received message. Field **Synd** is a vector carrying the error bits generated by the corresponding receiver.

Signal **ROBUS_Output_Out_o** is a record composed of field **Strobe** (which is asserted when a new message is being sent) and field **Message** (which has the content of the message).

The PE Interface has two parts. The interface for PE-to-RPP data flow is composed of two signals: PE_Input_i and PE_Read_Out_o. Signal **PE_Read_out_o** is aserted when the RPP is reading the next PE message. Signal **PE_Input_i** is a record with fields **Message** (the message from the PE) and **Synd** (a Boolean signal which is asserted when an error has been detected in the communication from the PE). The interface for RPP-to-PE data flow has only signal **PE_Output_o**, which is a record with fields **Message** (the content of the message sent to the PE) and **Strobe** (asserted when a new message is ready).

```
entity RPP_Unit is
   port ( CLK_i                    : in  bit ;

          -- External inputs
          Reset_In_i               : in  boolean ;
          Failure_In_i             : in  boolean ;
          Enable_In_i              : in  boolean ;
          Node_Id_In_i             : in  Node_Id_Typ ;
          Node_Kind_In_i           : in  Node_Kind_Typ ;
          Failure_Out_o            : out boolean ;

          -- Inputs from other ROBUS nodes
          ROBUS_Input_Vect_In_i    : in  ROBUS_Input_Vect_Typ ;

          -- Outputs to other ROBUS nodes
          ROBUS_Output_Out_o       : out ROBUS_Output_Typ ;

          -- Inputs from PE
          -- External inputs from the PE
          PE_Input_i               : in  PE_Input_Typ ;
          PE_Read_Out_o            : out boolean ;

          -- Outputs to PE
          PE_Output_o              : out PE_Output_Typ

        ) ;

end RPP_Unit ;
```

Figure 11.7: VHDL entity declaration for the top-level RPP description

11.3. Mode Control Unit

The MCU performs the following functions: handles the interaction with other control logic within the same FCR; stores the assigned node kind and node identification number; implements the mode transition logic (i.e., major mode, diagnostic cycle, and minor mode transitions); controls the enabling of the send output to opposite-kind nodes; and implements the local time function.

11.3.1. Block diagram

Figure 11.8 shows the block diagram for the MCU. The Controller is a 12-state Mealy machine (FSM) implementing the logic in Figure 11.5. Additional state is stored in the Major Mode, Diagnostic Cycle, and Output Status registers. The Local Time timer triggers the issuing of commands after synchronization is achieved. The Timer block is used to measure the time during the Self-Test mode Δ_{STM} (see Section 7). The Node Id and Node Kind are loaded directly from the RPP Control Interface and forwarded to the other RPP sub-units as part of the set of command signals.

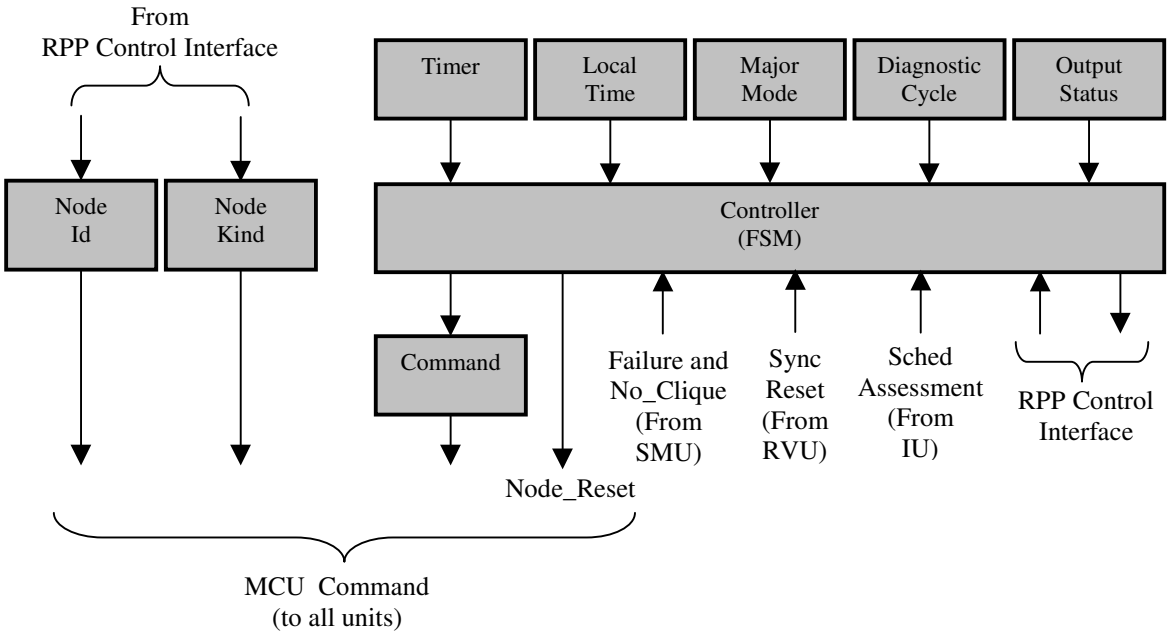


Figure 11.8: MCU block diagram

11.3.2. Interface

Figure 11.9 shows the VHDL entity declaration for the MCU. The "External inputs" and "External outputs" signals correspond to the RPP Control Interface as described previously for the top-level design. The signals from the Input Unit indicate whether the currently loaded PE communication schedule is valid or not, and whether the available schedule (loaded or default) is zero (i.e., no scheduled messages). Signal **RVU_Sync_Reset_i** indicates when it is time to reset the Local Time and start a new synchronization cycle. Signal **SMU_Failure_i** indicates when a local failure or a clique failure has been detected. **SMU_No_Clique_i** is asserted when no valid clique is detected. The outputs to the other RPP units include signals **Node_Kind_o** and **Node_Id_o**, which are held constant during the operation of the RPP. Signal **MCU_Node_Reset_o** commands an immediate reset all the RPP units. **MCU_Comand_o** is a record signal with fields **Major_Mode** (current major mode), **Diagnostic_Cycle**, **Minor_Mode**, **Schedule_States** (valid, zero, or invalid), and **Output_Enable**. Signal **MCU_Ready_o** is asserted to indicate that a new MCU command is being issued.

```
entity Mode_Control_Unit is
   port ( CLK_i                 : in  bit ;

          -- External inputs
          Reset_In_i            : in  boolean ;
          Failure_In_i          : in  boolean ;
          Enable_In_i           : in  boolean ;
          Node_Id_In_i          : in  Node_Id_Typ ;
          Node_Kind_In_i        : in  Node_Kind_Typ ;

          -- External outputs
          Failure_Out_o         : out boolean ;

          -- From Input Unit
          IU_Zero_Schedule_i    : in  boolean ;
          IU_Invalid_Schedule_i : in  boolean ;

          -- From Route and Vote Unit
          RVU_Sync_Reset_i      : in  boolean ;

          -- From Status Monitoring Unit
          SMU_Failure_i         : in  boolean ;
          SMU_No_Clique_i       : in  boolean ;

          -- Output to other units
          Node_Id_o             : out Node_Id_Typ ;
          Node_Kind_o           : out Node_Kind_typ ;
          MCU_Node_Reset_o      : out boolean ;
          MCU_Command_o         : out MCU_Cmnd_Typ ;
          MCU_Ready_o           : out boolean
        ) ;

end Mode_Control_Unit ;
```

Figure 11.9: VHDL entity declaration for the Mode Control Unit

11.4. Schedule Processor

The functions assigned to the Schedule Processor include storing the schedule computed by the Schedule Update protocol, assessing the loaded schedule, and reading and interpreting the current schedule. If the currently loaded schedule is invalid, the Schedule Processor uses the default schedule. Since the RPP has separate reception and transmission processes (embodied by the IU and OU, respectively), each one is given a separate Schedule Processor.

11.4.1. Block diagram

Figure 11.10 shows the block diagram for the Schedule Processor. The Controller is a 7-state Mealy machine. The output from the RVU is stored in the Schedule Memory, which consists of a FIFO (first-in-first-out) buffer. The output from the Schedule Memory includes the source Id and number of scheduled messages for each scheduled PE, in addition to the total number of scheduled sources. The Schedule

Assessment module determines the validity of the loaded schedule by comparing the number of scheduled messages against the maximum allowed and by detecting a PE_ERROR result for any of the sources. If the schedule is invalid, the content of the Default Schedule is used. The Message Counter counts the number of messages processed for the current source and signals the controller when the last message has been reached. The Source Counter is used as a source Id generator during schedule load, and as a source counter during schedule execution.

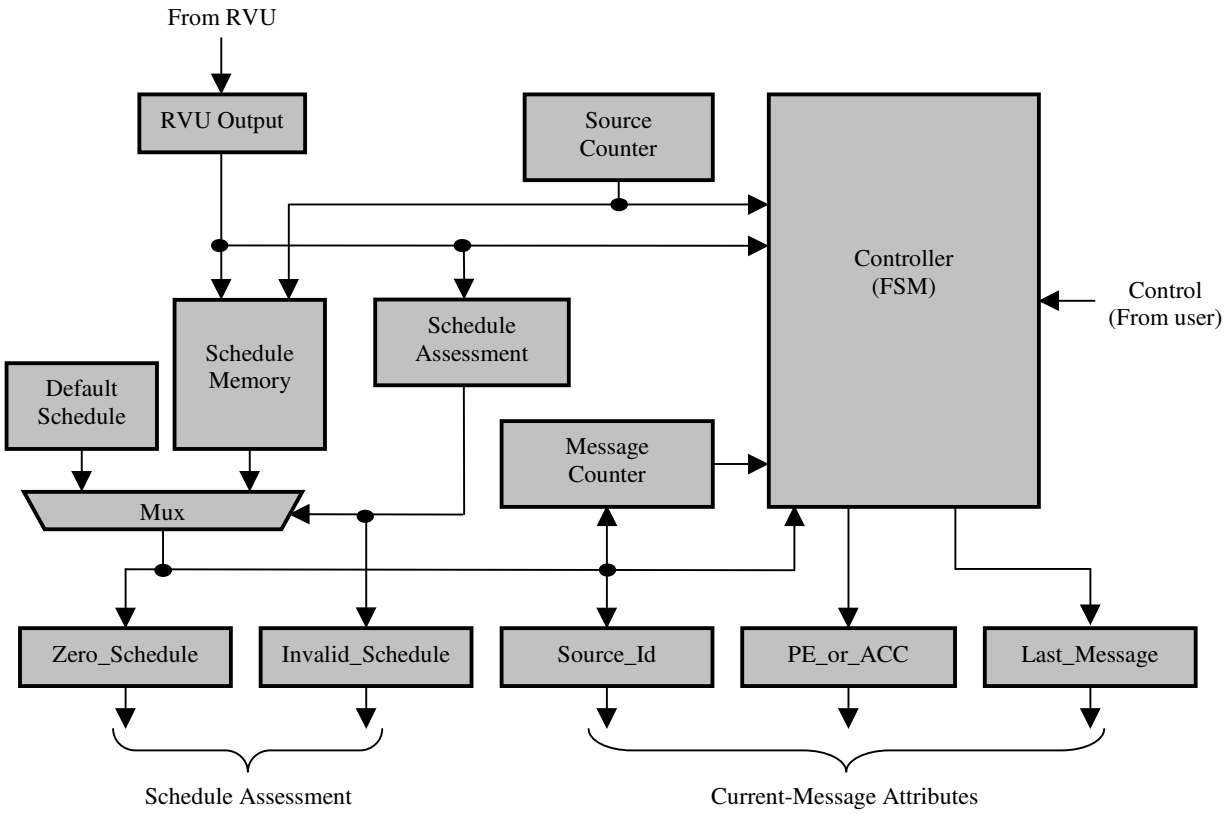


Figure 11.10: Block diagram for the Schedule Processor

11.4.2. Interface

Figure 11.11 shows the VHDL entity declaration for the Schedule Processor. The user of the Processor is either the IU or the OU. The **Reset_i** forces an immediate synchronous reset. **SCH_Proc_Cmnd_i** is used to command loading or execution of a schedule. **SCH_Ready_i** is asserted when the attributes for the next message are needed. The signals from the RVU include **RVU_Transfrm_Result_i** (the result stream from the Schedule Update protocol), **RVU_Ready** (asserted when a new schedule result is available), and **RVU_Last_Msg_i** (asserted when the last result has been reached). Outputs **SCH_Zero_o** and **SCH_Invalid_o** indicate the assessment result for the currently loaded schedule. **SCH_Source_Id_o** indicates the Id for the current source, **SCH_PE_or_ACC_o** is set to PE when the current message is a PE message or to ACC when processing Accusation Exchange protocol messages, and **SCH_Last_Msg_o** is asserted when the last message of the PE Communication mode has been reached.

```
entity Schedule_Processor is
   port ( CLK_i                 : in  bit ;

          -- From the user of this unit
          Reset_i               : in  boolean ;
          SCH_Proc_Cmnd_i       : in  Sched_Proc_Cmnd ;
          SCH_Ready_i           : in  Boolean ;

          -- From Route and Vote Unit
          RVU_Transfrm_Result_i : in  ROBUS_Msg_Typ ;
          RVU_Ready_i           : in  Boolean ;
          RVU_Last_Msg_i        : in  boolean ;

          -- Outputs
          SCH_Source_Id_o       : out Node_Id_Typ ;
          SCH_Last_Msg_o        : out boolean ;
          SCH_PE_or_ACC_o       : out PE_or_ACC_Typ ;
          SCH_Zero_o            : out boolean ;
          SCH_Invalid_o         : out boolean
        ) ;

end Schedule_Processor ;
```

Figure 11.11: VHDL entity declaration for the Schedule Processor

11.5. Input Unit

The Input Unit performs three main functions: message reception, frame synchronization, and control of data introduction for the computation pipeline. The message reception modes include synchronous, fixed-delay, and asynchronous-monitoring. The IU performs error detection for each of these modes. Appendix A has the detailed IU specification.

11.5.1. Block diagram

Figure 11.12 shows the block diagram for the Input Unit. The Controller is a 35-state Mealy machine that uses the MCU commands, RVU sync events, and the output of the Frame Synchronizer as timing references to control the execution of the modules within the IU. Some Pipeline Control signals are generated by the Controller and the rest are taken directly from the Schedule Processor. There is one Message Receiver for each opposite-kind node. These modules are used for the three reception modes. For synchronous reception, the Input Buffer behaves as a FIFO buffer with data being buffered upon arrival and read from the buffer at predetermined values of the local times. For fixed-delay reception, the sync pulses for INIT and ECHO messages are generated a fixed delay after their reception. For asynchronous reception, the Input Buffer performs as a 1-tick delay register. The Error Detector is active for all modes of reception. The error checks include link error (received from the communication receivers), non-arrival of an expected message, reception of more messages than expected, input reception rate too high, and reception of an unexpected message. The Frame Synchronizer implements the frame synchronization protocol described in Appendix A.

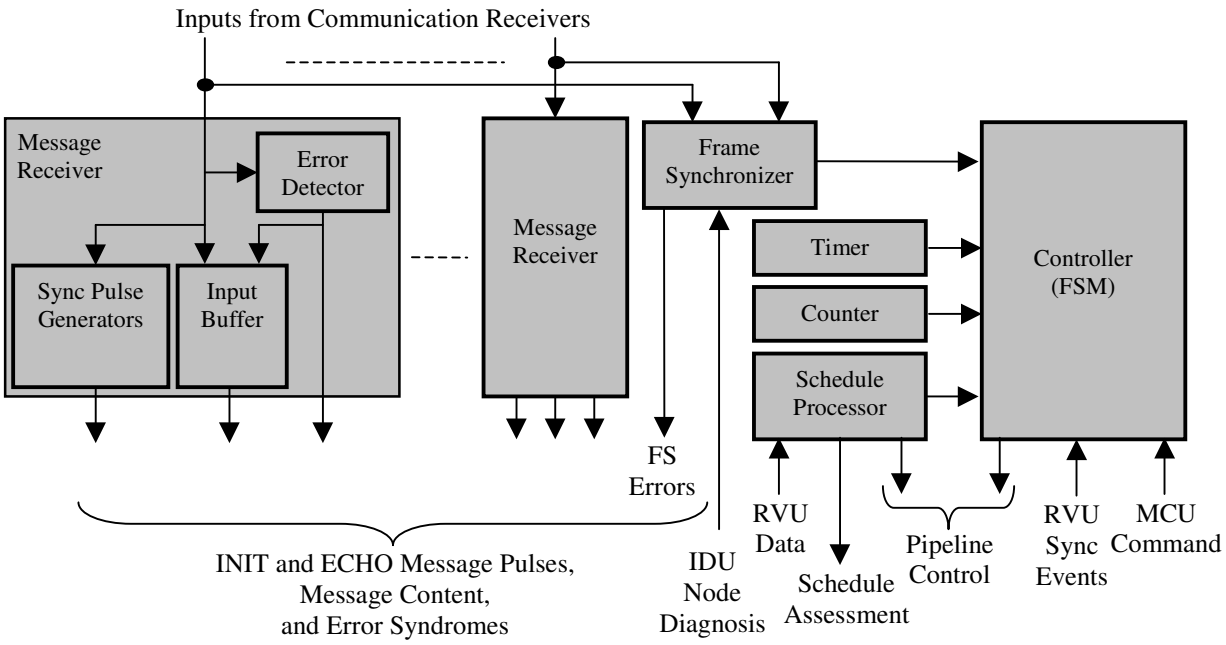


Figure 11.12: Block diagram for the Input Unit

11.5.2. Interface

Figure 11.13 shows the VHDL entity declaration for the Input Unit. The inputs from the MCU are sent directly to the IU controller. Signal **ROBUS_Input_Vect_i** is a vector of records, with one record for each node of the opposite kind. The fields of these records include **Message** (the message content), **Synd** (the vector of link-error syndromes generated by the corresponding communication receiver), and **Strobe** (which is asserted when a new message is received). Signals **IDU_Invalid_Input_i** and **IDU_Ready_i** specify the initial set of eligible participants for the Frame Synchronization protocol. The inputs from the RVU include the sync event signals **RVU_Accept_Init_i** and **RVU_Accept_Echo_i** (each one generated by the corresponding Accept function), in addition to **RVU_Transfrm_Result_i**, **RVU_Ready_i**, and **RVU_Last_Msg_i**, which are used to load the results of the Schedule Update protocol.

The outputs from the IU can be divided conceptually into several groups. Signals **IU_Init_o** and **IU_Echo_o** represent synchronization events received in the form INT and ECHO messages. These events are converted into pulses, which are then used by the RVU to compute the Accept(INIT) and Accept(ECHO) functions. Signal **IU_Input_Msg_o** is a vector containing the received message content from each opposite kind node. The pipeline control signals include **IU_Input_Strobe_o** (for each input port, the corresponding vector element is asserted one tick after a message is received), **IU_Ready_o** (asserted at critical timing points in the processing of synchronization protocols, and when it is time to process a synchronous-protocol message), **IU_Receiving_o** (asserted when the IU is expecting to receive messages), **IU_PE_or_ACC_o** (taken from the Schedule Processor), **IU_Source_Id_o** (taken from the Schedule Processor), and **IU_Last_Msg_o** (asserted when the last message in a stream is being processed during Schedule Update and PE Communication). Signals **IU_Synd_o** and **IU_Imm_Synd_o** are of type record and carry the error syndromes. **IU_Invalid_Schedule_o** and **IU_Zero_Schedule_o** are the assessment results generated by the Schedule Processor.

```
entity Input_Unit is
   port ( CLK_i                 : in  bit ;

          -- From Mode Control Unit
          Node_Id_i             : in  Node_Id_Typ ;
          Node_Kind_i           : in  Node_Kind_typ ;
          MCU_Node_Reset_i      : in  boolean ;
          MCU_Command_i         : in  MCU_Cmnd_Typ ;
          MCU_Ready_i           : in  boolean ;

          -- From other ROBUS nodes
          ROBUS_Input_Vect_i    : in  ROBUS_Input_Vect_Typ ;

          -- From Input Diagnostics Unit
          IDU_Invalid_Input_i   : in  OK_Bool_Vect_Typ ;
          IDU_Ready_i           : in  boolean ;

          -- From Route and Vote Unit
          RVU_Accept_Init_i     : in  boolean ;
          RVU_Accept_Echo_i     : in  boolean ;
          RVU_Transfrm_Result_i : in  ROBUS_Msg_Typ ;
          RVU_Ready_i           : in  boolean ;
          RVU_Last_Msg_i        : in  boolean ;

          -- Output to other units
          IU_Init_o             : out OK_Sync_Pls_2D_Vect_Typ ;
          IU_Echo_o             : out OK_Sync_Pls_2D_Vect_Typ ;
          IU_Input_Msg_o        : out OK_Msg_Vect_Typ ;
          IU_Input_Strobe_o     : out OK_Bool_Vect_Typ ;
          IU_Ready_o            : out boolean ;
          IU_Receiving_o        : out boolean ;
          IU_PE_or_ACC_o        : out PE_or_ACC_Typ ;
          IU_Source_Id_o        : out Source_Id_Typ ;
          IU_Last_Msg_o         : out boolean ;
          IU_Synd_o             : out IU_Synd_Typ ;
          IU_Immd_Synd_o        : out IU_Imm_Synd_Typ ;
          IU_Zero_Schedule_o    : out boolean ;
          IU_Invalid_Schedule_o : out boolean
        ) ;

end Input_Unit ;
```

Figure 11.13: VHDL entity declaration for the Input Unit

11.6. Input Diagnostics Unit

The main purpose of the IDU is to identify invalid inputs. The IDU combines the results of its own error checks with the error syndromes from the Input Unit to detect invalid inputs. When the RPP is unsynchronized and using asynchronous-monitoring communication during Clique Detection and Clique Initialization modes, the IDU performs monitoring and error detection for each opposite kind node. The monitoring functions include rate checks on the reception of ECHO messages during Local Diagnosis Acquisition in the Clique Detection mode, and message sequence checks during Clique Detection and

Clique Initialization modes. After the RPP synchronizes to the clique, the IDU performs error detection based on the content of received messages.

11.6.1. Block diagram

Figure 11.14 shows the block diagram for the IDU. The Controller is an 18-state Mealy machine. Each input lane is processed separately. For asynchronous monitoring, the rate and sequence monitoring results are combined with the IU reception error syndromes and Frame Synchronization protocol errors to determine validity. The Rate Monitor includes a 9-state Mealy machine and a counter. The Sequence Monitor includes a 16-state Mealy machine and a counter. For synchronized operations, validity is based on content error checks and IU reception error syndromes.

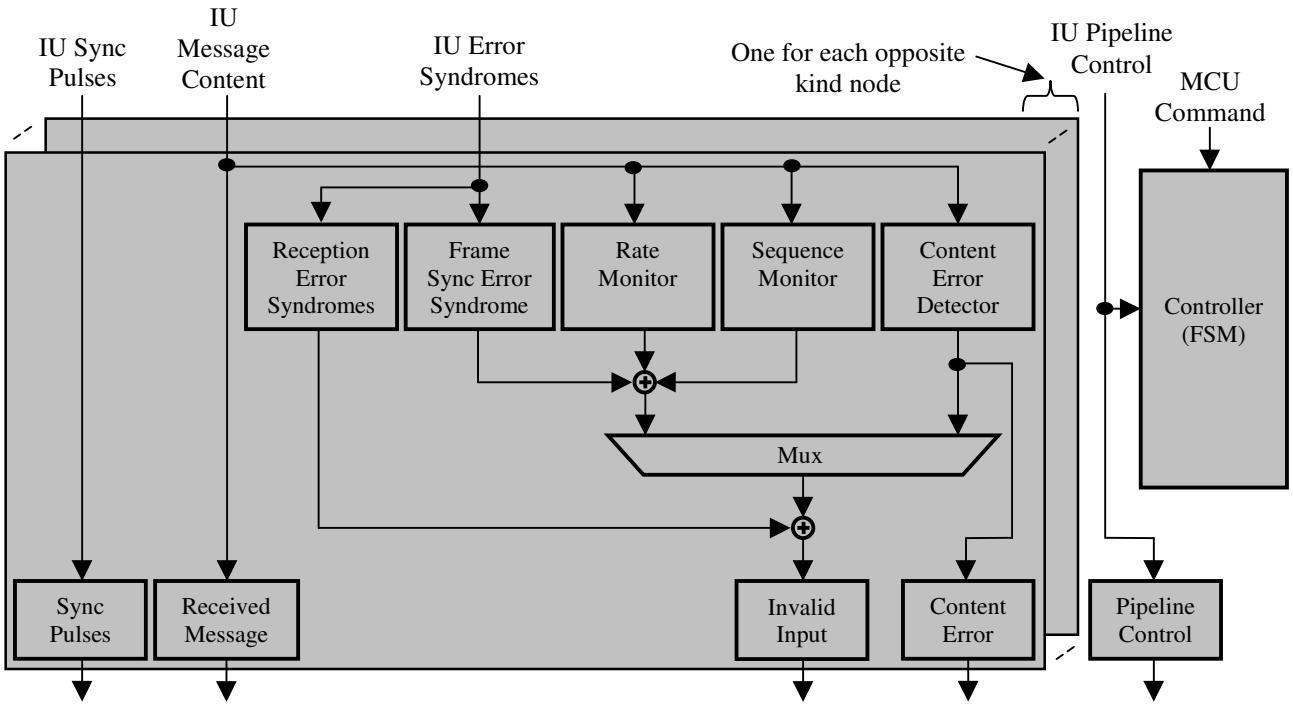


Figure 11.14: Block diagram for the Input Diagnostics Unit

11.6.2. Interface

Figure 11.15 is the VHDL entity declaration for the IDU. The inputs include the clock **CLK_i**, the MCU command, and the IU outputs. The IDU outputs include the sync pulses **IDU_Init_o** and **IDU_Echo_o**; the received messages **IDU_Input_Msg_o**; and the pipeline control signals **IDU_Ready_o**, **IDU_PE_or_ACC_o**, **IDU_Source_Id_o**, and **IDU_Last_Msg_o**. Signal **IDU_Invalid_Input_o** specifies the valid inputs, and **IDU_Synd_o** is the content error syndromes.

```
entity Input_Diagnostics_Unit is
   port ( CLK_i                 : in  bit ;

          -- From Mode Control Unit
          Node_Id_i             : in  Node_Id_Typ ;
          Node_Kind_i           : in  Node_Kind_typ ;
          MCU_Node_Reset_i      : in  boolean ;
          MCU_Command_i         : in  MCU_Cmnd_Typ ;
          MCU_Ready_i           : in  boolean ;

          -- From Input Unit
          IU_Init_i             : in  OK_Sync_Pls_2D_Vect_Typ ;
          IU_Echo_i             : in  OK_Sync_Pls_2D_Vect_Typ ;
          IU_Input_Msg_i        : in  OK_Msg_Vect_Typ ;
          IU_Input_Strobe_i     : in  OK_Bool_Vect_Typ ;
          IU_Ready_i            : in  boolean ;
          IU_Receiving_i        : in  boolean ;
          IU_PE_or_ACC_i        : in  PE_or_ACC_Typ ;
          IU_Source_Id_i        : in  Source_Id_Typ ;
          IU_Last_Msg_i         : in  boolean ;
          IU_Synd_i             : in  IU_Synd_Typ ;
          IU_Immd_Synd_i        : in  IU_Imm_Synd_Typ ;

          -- Output to other units
          IDU_Init_o            : out OK_Sync_Pls_2D_Vect_Typ ;
          IDU_Echo_o            : out OK_Sync_Pls_2D_Vect_Typ ;
          IDU_Input_Msg_o       : out OK_Msg_Vect_Typ ;
          IDU_Ready_o           : out boolean ;
          IDU_PE_or_ACC_o       : out PE_or_ACC_Typ ;
          IDU_Source_Id_o       : out Source_Id_Typ ;
          IDU_Last_Msg_o        : out boolean ;
          IDU_Invalid_Input_o   : out OK_Bool_Vect_Typ ;
          IDU_Synd_o            : out IDU_Synd_Typ
        ) ;

end Input_Diagnostics_Unit ;
```

Figure 11.15: VHDL entity declaration for the Input Diagnostics Unit

11.7. Route and Vote Unit

The Route and Vote Unit is where the main computation takes place. The Input Eligible Voters for the dynamic voting functions are computed by the RVU using the Invalid Inputs computed by the IDU and the Accusations and Convictions provided by the Node Diagnostics Unit. For the synchronization protocols, the RVU computes Accept(INIT) and Accept(ECHO) functions. For the Schedule Update, PE Broadcast, and Collective Diagnosis protocols the RVU computes exact-match majority word vote functions (i.e., the unit of data is the full content of a message). For the PE Broadcast protocol, the RVU also performs a routing function that selects an input to be forwarded to the output. For the Collective Diagnosis and Accusation Exchange protocols, the RVU computes message-wide Boolean majority bit vote functions (i.e., each bit location is voted independently). The RVU performs error checks for each dynamic voting function, as well as for the Input Eligible Voters function.

11.7.1. Block diagram

Figure 11.16 shows the block diagram for the RVU. The Controller is a 24-state Mealy machine. The Timer is used to measure the synchronization-reset delay for the synchronization protocols. The Input Eligible Voters block computes the eligible voters (IEV) for all the dynamic voting functions. This module also performs checks for the conditions of no-eligible-voters and invalid-initial-eligible-voters for the Synchronization Capture and Initial Synchronization protocols, as stated in the RPP requirements. The Accept functions use the widths of the sync pulses to check for no-majority conditions among its IEVs and for determining disagreement with the result of the function. The Content Transformation block performs transformations required by the protocols beyond the basic dynamic voting and routing operations.

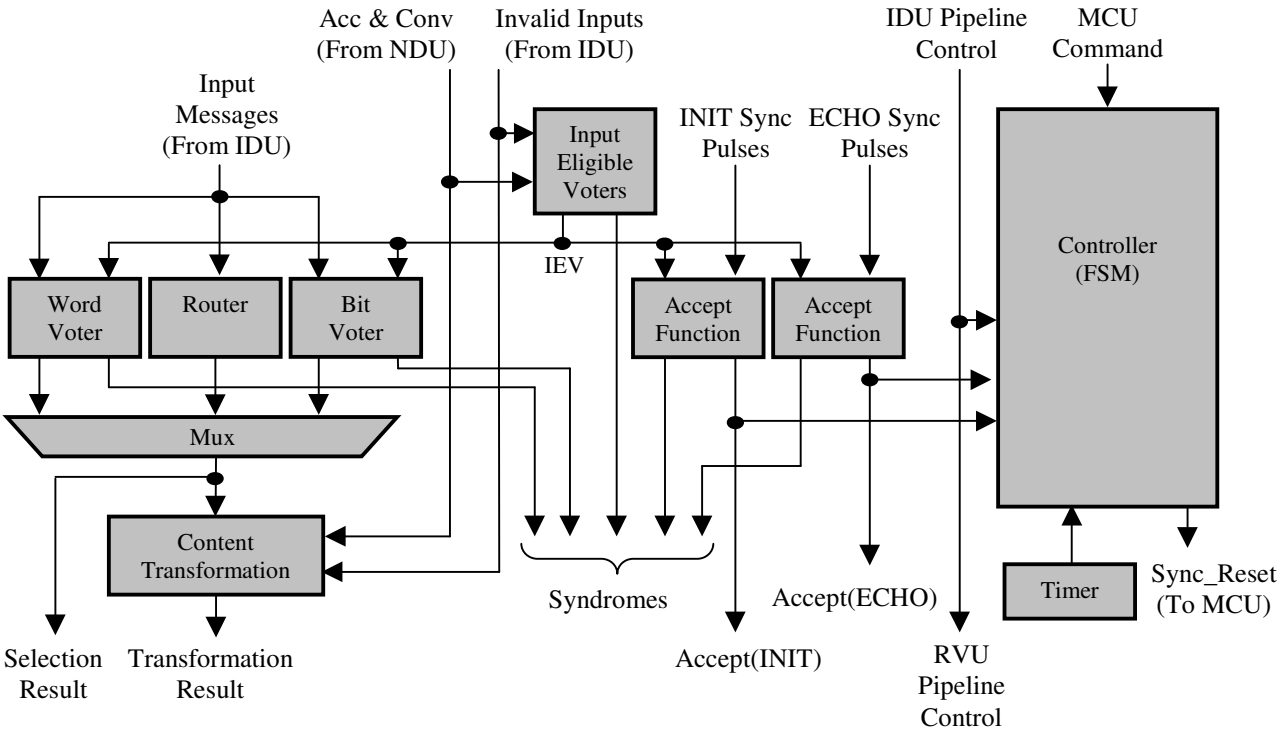


Figure 11.16: Block diagram for the Route and Vote Unit

11.7.2. Interface

Figure 11.17 shows the VHDL entity declaration for the RVU. The inputs from the MCU are sent directly to the RVU controller. The IDU inputs are used as described above for the block diagram. The NDU inputs include the accusations and convictions against the nodes of the same and opposite kinds. The RVU outputs include synchronization events (**RVU_Accept_Init_o**, **RVU_Accept_Init_o**, and **RVU_Sync_Reset_o**), content results (**RVU_Transfrm_Result_o** and **RVU_Select_Result_o**), pipeline control outputs (**RVU_Ready_o**, **RVU_PE_or_ACC_o**, **RVU_Source_Id_o**, and **RVU_Last_Msg_o**), and the error syndromes (**RVU_Synd_o**). The RVU syndromes are disagreement with the result of Accept(INIT), disagreement with the result of Accept(ECHO), disagreement with the results of the bit vote, disagreement with the result of the word vote, no-majority for Accept(INIT), no-majority for Accept(ECHO), no-majority for the word vote, no eligible voters, and invalid eligible voters.

```
entity Route_and_Vote_Unit is
   port ( CLK_i                  : in  bit ;

          -- From Mode Control Unit
          Node_Id_i              : in  Node_Id_Typ ;
          Node_Kind_i            : in  Node_Kind_typ ;
          MCU_Node_Reset_i       : in  boolean ;
          MCU_Command_i          : in  MCU_Cmnd_Typ ;
          MCU_Ready_i            : in  boolean ;

          -- From Input Diagnostics Unit
          IDU_Init_i             : in  OK_Sync_Pls_2D_Vect_Typ ;
          IDU_Echo_i             : in  OK_Sync_Pls_2D_Vect_Typ ;
          IDU_Input_Msg_i        : in  OK_Msg_Vect_Typ ;
          IDU_Ready_i            : in  boolean ;
          IDU_PE_or_ACC_i        : in  PE_or_ACC_Typ ;
          IDU_Source_Id_i        : in  Source_Id_Typ ;
          IDU_Last_Msg_i         : in  boolean ;
          IDU_Invalid_Input_i    : in  OK_Bool_Vect_Typ ;

          -- From Node Diagnostics Unit
          NDU_Acc_and_Conv_i     : in  Acc_and_Conv_Typ ;

          -- Output to other units
          RVU_Accept_Init_o      : out boolean ;
          RVU_Accept_Echo_o      : out boolean ;
          RVU_Sync_Reset_o       : out boolean ;
          RVU_Transfrm_Result_o  : out ROBUS_Msg_Typ ;
          RVU_Ready_o            : out boolean ;
          RVU_PE_or_ACC_o        : out PE_or_ACC_Typ ;
          RVU_Source_Id_o        : out Source_Id_Typ ;
          RVU_Last_Msg_o         : out boolean ;
          RVU_Select_Result_o    : out ROBUS_Msg_Typ ;
          RVU_Synd_o             : out RVU_Synd_Typ
        ) ;

end Route_and_Vote_Unit ;
```

Figure 11.17: VHDL entity declaration for the Route and Vote Unit

11.8. Node Diagnostics Unit

The main purpose of the Node Diagnostics Unit is to diagnose all the nodes in the system using the syndromes from the IU, IDU, and RVU. The diagnostics rules are described in detail in [Torres 05]. The NDU generates suspicions and accusations against nodes of the same and opposite kinds. It also store the convictions results computed during the execution of the Collective Diagnosis and Collective Diagnosis acquisition protocols.

11.8.1. Block diagram

Figure 11.18 shows the block diagram for the NDU. The Controller is a 26-state Mealy machine. The Suspicion Generators generates suspicions based on the syndromes of disagreement with the result of the

word vote and the bit vote. The Controller specifies when these syndromes are valid for the generation of suspicions. The generated suspicions are stored in the Suspicions Matrix, a 2-dimensional Boolean array with one row per node of the same kind and one column per node of the opposite kind. The accumulated suspicions are reduced after the execution of the Accusation Exchange protocol by performing row-wise and column-wise dynamic bit vote operations for every row and every column. The eligible voters are based on the accusations and convictions held by the NDU at the time of the vote.

During asynchronous-monitoring, the IDU performs the diagnosis of opposite-kind nodes. At the end of that communication mode, the diagnostics results are loaded directly from the IDU Invalid_Inputs signal as the current opposite-kind accusations.

The Temporary Accusations Register is used during the execution of the Collective Diagnosis protocol to hold the accusations for the previous diagnostic cycle and enable the accumulation of new accusations for the next diagnostic cycle while the convictions are being computed.

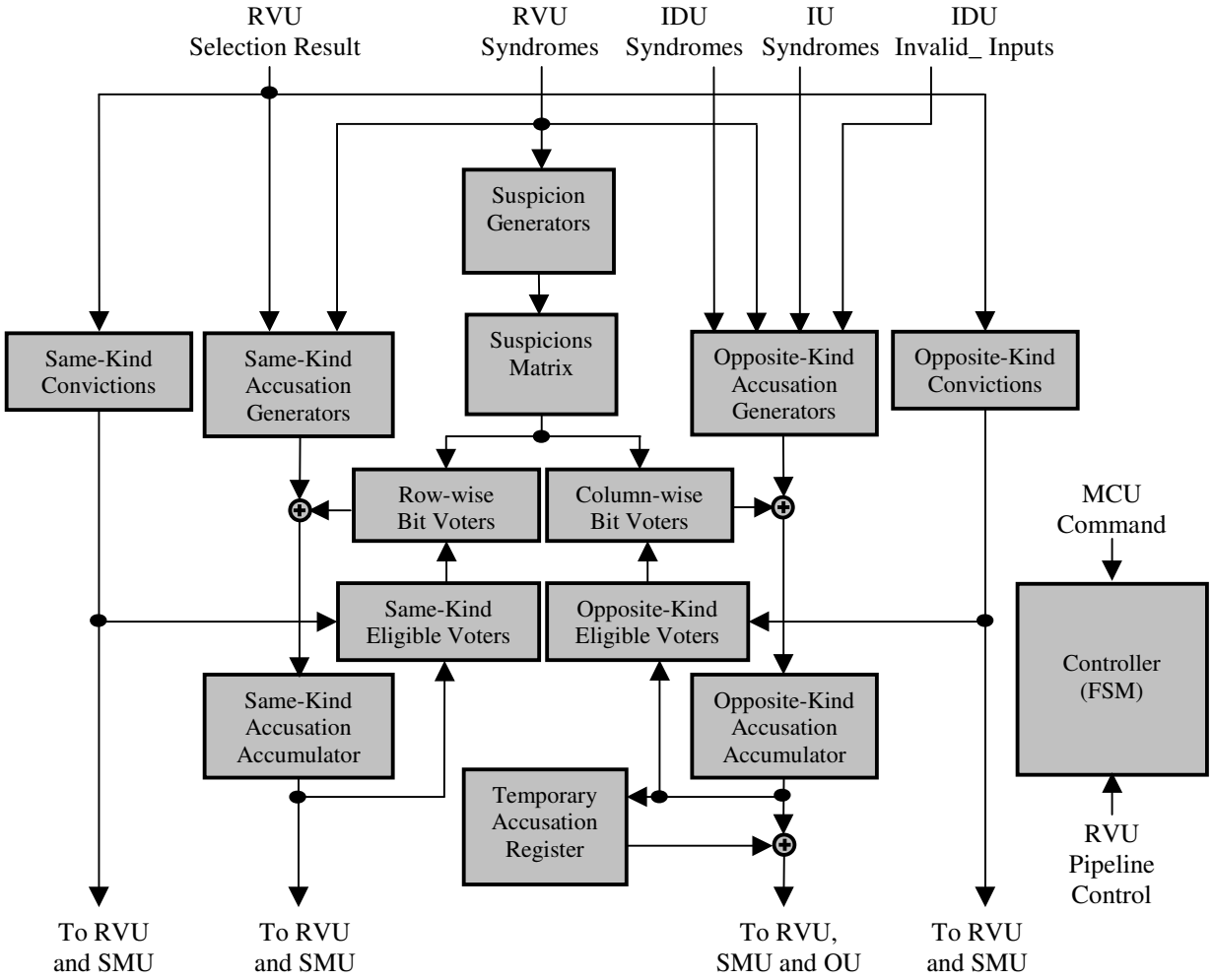


Figure 11.18: Block diagram for the Node Diagnostics Unit

11.8.2. Interface

Figure 11.19 shows the VHDL entity declaration for the NDU. The inputs consist of the MCU command, the IU syndromes with the relevant timing signals, the IDU syndromes and Invalid_Input signal with the timing signals, and RVU syndromes and Selection Result output with the pipeline control signals. The outputs include accusations and convictions as a record data type, and the accusation sent to the Output Unit as a ROBUS Message.

```
entity Node_Diagnostics_Unit is
   port ( CLK_i                 : in  bit ;

          -- From Mode Control Unit
          Node_Id_i             : in  Node_Id_Typ ;
          Node_Kind_i           : in  Node_Kind_typ ;
          MCU_Node_Reset_i      : in  boolean ;
          MCU_Command_i         : in  MCU_Cmnd_Typ ;
          MCU_Ready_i           : in  boolean ;

          -- From Input Unit
          IU_Input_Strobe_i     : in  OK_Bool_Vect_Typ ;
          IU_Ready_i            : in  boolean ;
          IU_Receiving_i        : in  boolean ;
          IU_Synd_i             : in  IU_Synd_Typ ;
          IU_Immd_Synd_i        : in  IU_Imm_Synd_Typ ;

          -- From Input Diagnostics Unit
          IDU_Ready_i           : in  boolean ;
          IDU_Invalid_Input_i   : in  OK_Bool_Vect_Typ ;
          IDU_Synd_i            : in  IDU_Synd_Typ ;

          -- From Route and Vote Unit
          RVU_Ready_i           : in  boolean ;
          RVU_PE_or_ACC_i       : in  PE_or_ACC_Typ ;
          RVU_Source_Id_i       : in  Source_Id_Typ ;
          RVU_Last_Msg_i        : in  boolean ;
          RVU_Select_Result_i   : in  ROBUS_Msg_Typ ;
          RVU_Synd_i            : in  RVU_Synd_Typ ;

          -- Output to other units
          NDU_Acc_and_Conv_o    : out Acc_and_Conv_Typ ;
          NDU_Diag_Msg_o        : out ROBUS_Msg_Typ
        ) ;

end Node_Diagnostics_Unit ;
```

Figure 11.19: VHDL entity declaration for the Node Diagnostics Unit

11.9. Status Monitoring Unit

The purpose of the SMU is to detect a local failure or a clique failure, and to detect when a clique is not present on the bus. This assessment is done based on the accusations and convictions from the NDU,

and the protocol results from the RVU. The SMU also includes a timeout check for the Synchronization Acquisition and Initial Synchronization protocols, as well as for the resynchronization interval after synchronization is achieved.

11.9.1. Block diagram

Figure 11.20 shows the block diagram for the SMU. The Controller is a 23-state Mealy machine. The Sync_Reset signal from the RVU is the timing reference for the timeout checks. The RVU syndromes, including no-majority for Acept(INIT), no-majority for Accept(ECHO), no-majority for word vote, no eligible voters, and invalid eligible voters, are monitored by the controller to detect protocols failures. In additions, checks are performed on the RVU Selection Result output. These check include a comparison of the message sent by a source BIU against the corresponding RVU result during the PE Broadcast protocol, comparing the results of processes P2 and P4 of the Collective Diagnosis protocol, and checking the results of the Collective Diagnosis protocol for all-convicted conditions or a conviction-against-local-node result. The Status Monitoring block combines the NDU diagnosis, timeout checks, and protocol failure indicators to detect failure conditions and the absence of a valid clique.

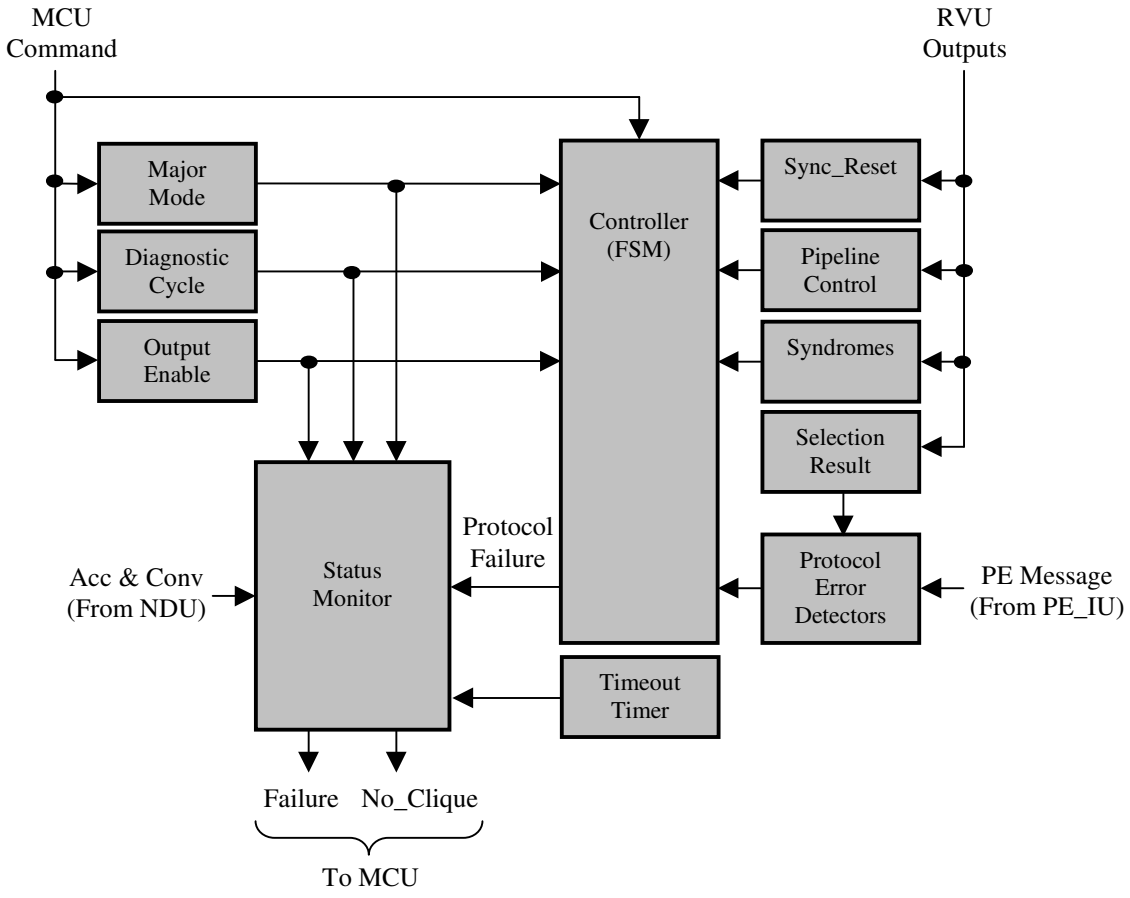


Figure 11.20: Block diagram for the Status Monitoring Unit

11.9.2. Interface

Figure 11.21 shows the VHDL entity declaration for the SMU. Signal **OU_Read_i** from the Output Unit is asserted when a PE message is read from the PE Input Unit (PE_IU).

```
entity Status_Monitoring_Unit is
   port ( CLK_i                  : in  bit ;

          -- From Mode Control Unit
          Node_Id_i              : in  Node_Id_Typ ;
          Node_Kind_i            : in  Node_Kind_typ ;
          MCU_Node_Reset_i       : in  boolean ;
          MCU_Command_i          : in  MCU_Cmnd_Typ ;
          MCU_Ready_i            : in  boolean ;

          -- From Route and Vote Unit
          RVU_Sync_Reset_i       : in  boolean ;
          RVU_Ready_i            : in  boolean ;
          RVU_PE_or_ACC_i        : in  PE_or_ACC_Typ ;
          RVU_Source_Id_i        : in  Source_Id_Typ ;
          RVU_Last_Msg_i         : in  boolean ;
          RVU_Select_Result_i    : in  ROBUS_Msg_Typ ;
          RVU_Synd_i             : in  RVU_Synd_Typ ;

          -- From Node Diagnostics Unit
          NDU_Acc_and_Conv_i     : in  Acc_and_Conv_Typ ;

          -- From PE Input Unit
          PEIU_Msg_i             : in  ROBUS_Msg_Typ ;

          -- From Output Unit
          OU_Read_i              : in  Boolean ;

          -- Output to other units
          SMU_Failure_o          : out boolean ;
          SMU_No_Clique_o        : out boolean
        ) ;

end Status_Monitoring_Unit ;
```

Figure 11.21: VHDL entity declaration for the Status Monitoring Unit

11.10. Output Unit

The Output Unit is responsible for sending messages triggered by the local time or RVU-computed synchronization events. The messages are either generated internally (INIT and ECHO for synchronization protocols, and INITIALIZATION for Initial Diagnosis), or read from the PE Input Unit, the RVU, or the NDU.

11.10.1. Block diagram

Figure 11.22 shows the block diagram for the OU. OK stands for "Opposite Kind" and RM for "ROBUS Message". The Controller is a 26-state Mealy machine. The RVU transformation results are buffered until it is time to send them. OU_Read indicates when the OU is reading a PE Message from the PE_IU. The OU decides when to assert the Output Strobe signal based on the protocol being executed without considering the Major Mode or Diagnostic Cycle. It is the responsibility of the MCU to specify when to enable the sending of messages.

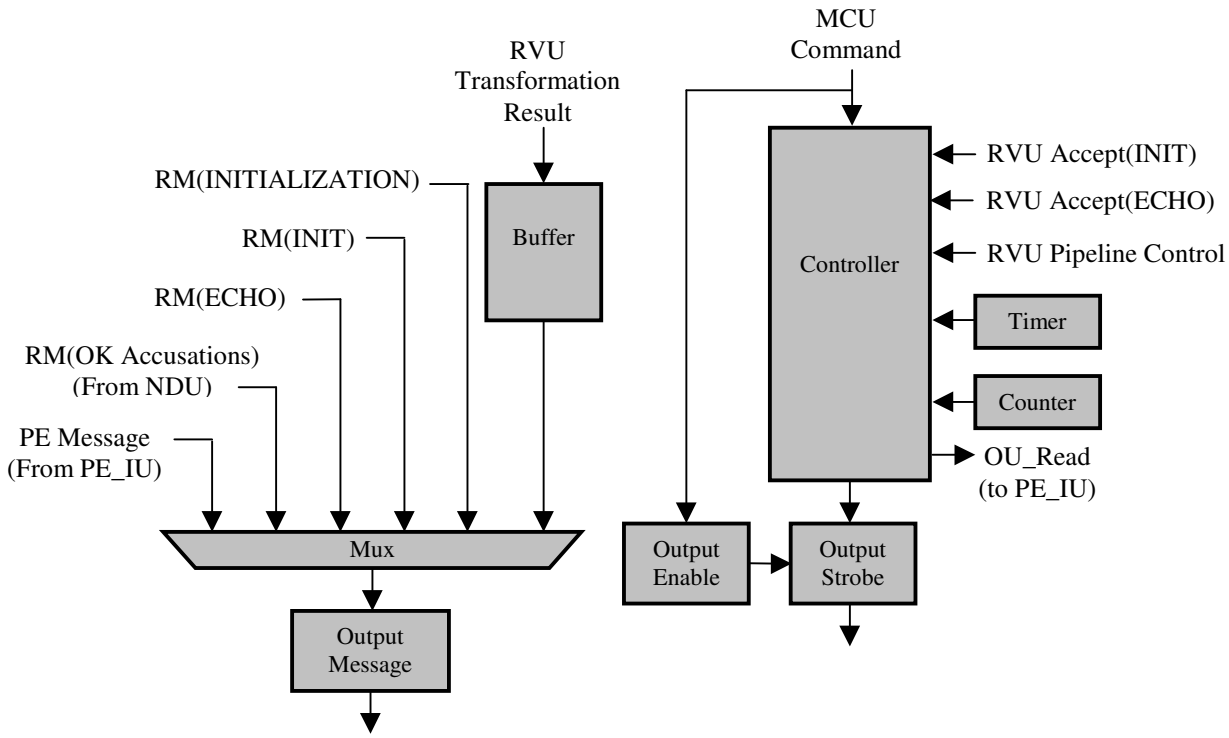


Figure 11.22: Block diagram for the Output Unit

11.10.2. Interface

Figure 11.23 shows the VHDL entity declaration for the OU. Signal **ROBUS_Output_o** is of type record with field equal to Message and Strobe.

```
entity Output_Unit is
   port ( CLK_i                  : in  bit ;

          -- From Mode Control Unit
          Node_Id_i              : in  Node_Id_Typ ;
          Node_Kind_i            : in  Node_Kind_typ ;
          MCU_Node_Reset_i       : in  boolean ;
          MCU_Command_i          : in  MCU_Cmnd_Typ ;
          MCU_Ready_i            : in  boolean ;

          -- From Route and Vote Unit
          RVU_Accept_Init_i      : in  Boolean ;
          RVU_Accept_Echo_i      : in  Boolean ;
          RVU_Transfrm_Result_i  : in  ROBUS_Msg_Typ ;
          RVU_Ready_i            : in  Boolean ;
          RVU_Last_Msg_i         : in  boolean ;

          -- From Node Diagnostice Unit
          NDU_Diag_Msg_i         : in  ROBUS_Msg_Typ ;

          -- Internal outputs
          OU_Read_o              : out Boolean ;

          -- From PE Input Unit
          PEIU_Msg_i             : in  ROBUS_Msg_Typ ;

          -- External/ROBUS outputs
          ROBUS_Output_o         : out ROBUS_Output_Typ
        ) ;

end Output_Unit ;
```

Figure 11.23: VHDL entity declaration for the Output Unit

11.11. PE Input Unit

The PE Input Unit (PE_IU) is the interface between the PE and the RPP Output Unit. The RPP is designed with a FIFO abstraction for the PE input interface. The PE messages are presumed to be available whenever the RPP decided to read a message. The PE is responsible for reporting to the RPP when there is a problem such that a valid message is not available.

11.11.1. Block diagram

Figure 11.24 shows the block diagram for the PE_IU. The data from the PE is carried as the Payload of a DATA-tagged ROBUS Message. If an error is signaled by the PE, then a PE_ERROR message is used. The OU_Read signal is sent directly to the PE side of the interface.

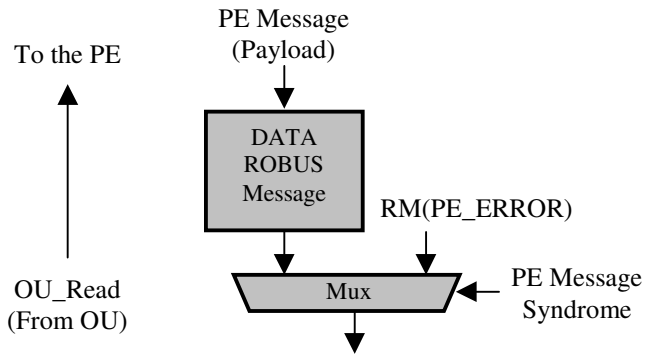


Figure 11.24: Block Diagram for the PE Input Unit

11.11.2. Interface

Figure 11.25 shows the VHDL entity declaration for the PE_IU. The MCU command is not used. **PEIU_Msg_o** is the PE message sent to the OU. **PE_Input_i** is a record with fields **Message** (the PE payload) and **Synd** (asserted when a valid payload is not available). **PEIU_Read_Out_o** is connected directly to **OU_Read_i**.

```
entity PE_Input_Unit is
   port ( CLK_i                 : in  bit ;

          -- From Mode Control Unit
          Node_Id_i             : in  Node_Id_Typ ;
          Node_Kind_i           : in  Node_Kind_typ ;
          MCU_Node_Reset_i      : in  boolean ;
          MCU_Command_i         : in  MCU_Cmnd_Typ ;
          MCU_Ready_i           : in  boolean ;

          -- From Output Unit
          OU_Read_i             : in  boolean ;

          -- Internal outputs
          PEIU_Msg_o            : out ROBUS_Msg_Typ ;

          -- External inputs from the PE
          PE_Input_i            : in  PE_Input_Typ ;

          -- External outputs to the PE
          PEIU_Read_Out_o       : out Boolean
        ) ;

end PE_Input_Unit ;
```

Figure 11.25: VHDL entity declaration for the PE Input Unit

11.12. PE Output Unit

The PE Output Unit sends messages to the PE. The PE_OU messages include Collective Diagnosis

convictions, Schedule Update results, PE Broadcast results, and INIT and ECHO synchronization messages. In addition, the PE_OU sends the assessment result for the current PE communication schedule, the current major mode, and the node Id number.

11.12.1. Block diagram

Figure 11.26 shows the block diagram for the PE_OU. The Controller is a 13-state Mealy machine. The timing of the Output Strobe signal is based on the timing of the MCU Command, RVU Accept(INIT), RVU_Accept(ECHO), and RVU Pipeline Control signals. The Major Mode and Node Id read from the MCU are used as payload field contents for ROBUS Messages sent to the PE. Other ROBUS Messages generated within the PE_OU include VALID_SCHEDULE, INVALID_SCHEDULE, ZERO_SCHEDULE, INIT, and ECHO SPECIAL-tagged ROBUS Messages. RVU Transformation Results are sent to the PE without modification.

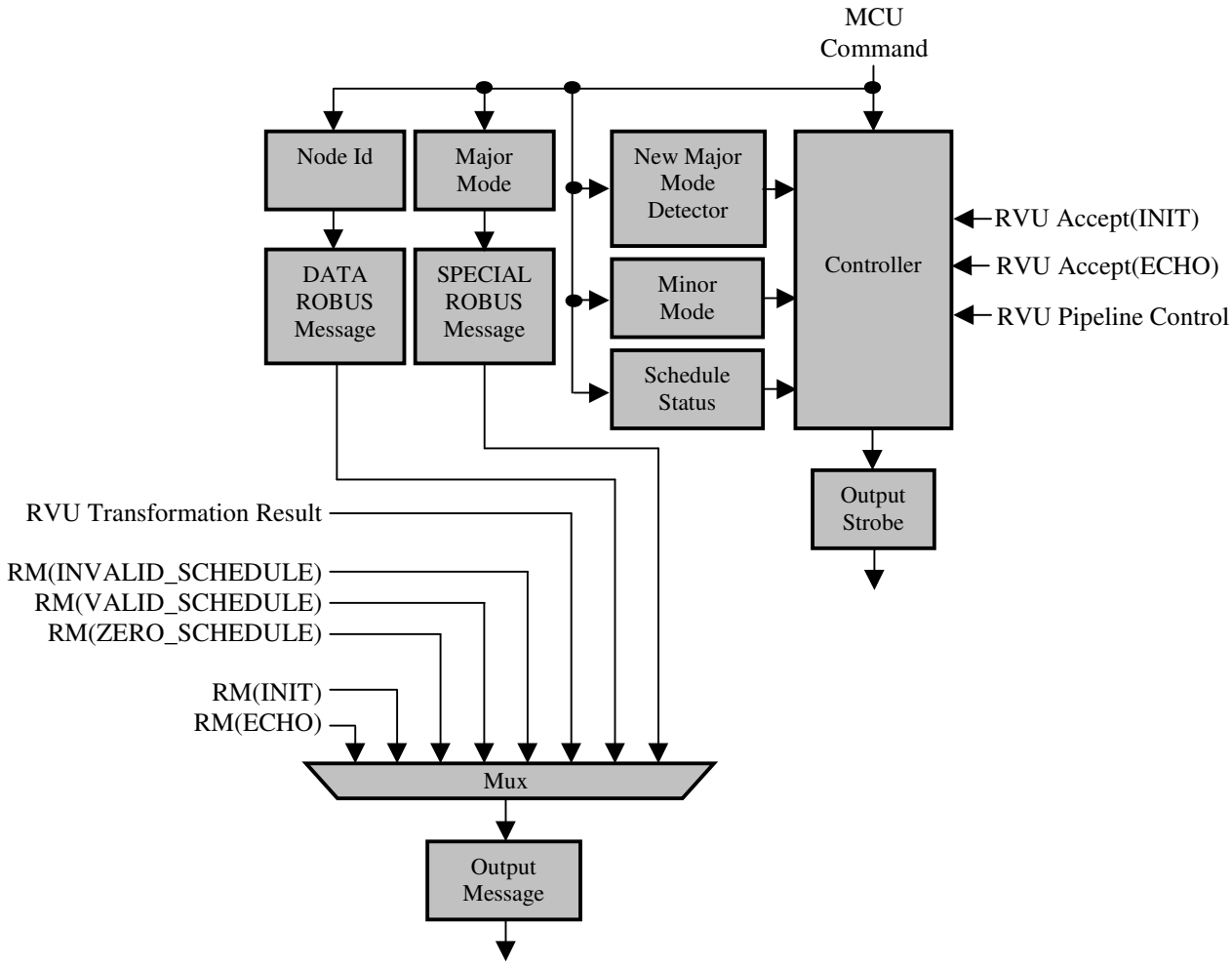


Figure 11.26: Block Diagram for the PE Output Unit

11.12.2. Interface

Figure 11.27 shows the VHDL entity declaration for the PE Output Unit. Signal **PE_Output_o** is of record type with fields Message and Strobe.

```
entity PE_Output_Unit is
   port ( CLK_i                  : in  bit ;

          -- From Mode Control Unit
          Node_Id_i              : in  Node_Id_Typ ;
          Node_Kind_i            : in  Node_Kind_typ ;
          MCU_Node_Reset_i       : in  boolean ;
          MCU_Command_i          : in  MCU_Cmnd_Typ ;
          MCU_Ready_i            : in  boolean ;

          -- From Route and Vote Unit
          RVU_Accept_Init_i      : in  boolean ;
          RVU_Accept_Echo_i      : in  boolean ;
          RVU_Transfrm_Result_i  : in  ROBUS_Msg_Typ ;
          RVU_Ready_i            : in  boolean ;
          RVU_PE_or_ACC_i        : in  PE_or_ACC_Typ ;
          RVU_Source_Id_i        : in  Source_Id_Typ ;
          RVU_Last_Msg_i         : in  boolean ;

          -- External outputs
          PE_Output_o            : out PE_Output_Typ
        ) ;

end PE_Output_Unit ;
```

Figure 11.27: VHDL entity declaration for the PE Output Unit

12. Behavioral parameters

This section presents the formulas for the behavioral parameters in the VHDL description of the RPP. The parameters are grouped according to the relevant RPP unit .

12.1. Mode Control Unit (MCU)

12.1.1. Min_Cmd_DII

Min_Cmd_DII denotes the minimum allowed separation between MCU commands. The PE Output Unit requires a minimum of 3 ticks between commands.

$$\text{Min\_Cmd\_DII} = 3 \tag{12.1}$$

The current design of the RPP imposes the following constraint.

$$\text{Min\_Cmd\_DII} \geq 3 \tag{12.2}$$

12.1.2. ST_Xtra_Dly

ST_Xtra_Dly denotes the extra delay required to meet the minimum-duration constraint for the Self-Test mode. In addition to the self-test delay, the MCU consumes 2 ticks to command a node reset and set the major mode variable to Self-Test.

$$\text{ST\_Xtra\_Dly} = \Delta_{STM} - \text{Min\_Cmd\_DII} - 2 \tag{12.3}$$

Note that for the current version of the MCU controller can cause the actual Self-Test delay may exceed Δ_{STM} by Min_Cmd_DII - 1 ticks. The current design of the RPP imposes the following constraint.

$$\text{ST\_Xtra\_Dly} \geq 1 \tag{12.4}$$

12.1.3. ID_Dly

ID_Dly denotes the delay in starting the Initial Diagnosis after entering the Clique Initialization mode.

$$\text{ID\_Dly} = \Delta_{ID,begin} \tag{12.5}$$

The current design of the RPP imposes the following constraint.

$$\text{ID\_Dly} \geq 1 \tag{12.6}$$

12.1.4. LT_CD

LT_CD denotes the local time at which to begin the execution of the Collective Diagnosis protocol.

$$LT\_CD = T_{CD} \tag{12.7}$$

The current design of the RPP imposes the following constraint.

$$LT\_CD \geq 1 \tag{12.8}$$

12.1.5. LT_SU

LT_SU denotes the local time at which to begin the execution of the Schedule Update protocol.

$$LT\_SU = T_{SU} \tag{12.9}$$

The current design of the RPP imposes the following constraint.

$$LT\_SU \geq 4 \tag{12.10}$$

12.1.6. LT_PE

LT_PE denotes the local time at which to begin the execution of the PE Communication protocols.

$$LT\_PE = T_{PE} \tag{12.11}$$

The current design of the RPP imposes the following constraint.

$$LT\_PE \geq 7 \tag{12.12}$$

12.1.7. LT_SP

LT_SP denotes the local time at which to begin the execution of the Synchronization Preservation protocol.

$$LT\_SP = T_{SP} \tag{12.13}$$

The current design of the RPP imposes the following constraint.

$$LT\_SP \geq 10 \tag{12.14}$$

12.2. Schedule Processor

12.2.1. Max_Num_PE_Msg

Max_Num_PE_Msg denotes the maximum number of PE messages that can be processed during the execution of the PE Communication protocols.

$$\text{Max\_Num\_PE\_Msg} = K_{PE,sched}|_{max} \tag{12.15}$$

The current design of the RPP imposes the following constraint.

$$\text{Max\_Num\_PE\_Msg} \geq 0 \tag{12.16}$$

12.2.2. Dflt_Num_PE_Msg

Dflt_Num_PE_Msg denotes the default number of messages per PE for the default schedule. The value of this parameter is application-dependent. The current design of the RPP imposes the following constraints.

$$\text{Dflt\_Num\_PE\_Msg} \geq 0 \tag{12.17}$$

$$N*\text{Dflt\_Num\_PE\_Msg} \leq K_{PE,sched}|_{max} \tag{12.18}$$

12.3. Input Unit (IU)

12.3.1. PD_Rdy1_Dly

PD_Rdy1_Dly denotes the expected duration of the first observation window during Local Diagnosis Acquisition (a.k.a., Preliminary Diagnosis).

$$\text{PD\_Rdy1\_Dly} = \Delta_{PD,OW} \tag{12.19}$$

The current design of the RPP imposes the following constraint.

$$\text{PD\_Rdy1\_Dly} \geq 1 \tag{12. (20)}$$

12.3.2. PD_Rdy2_Dly

PD_Rdy2_Dly denotes the expected duration of the second observation window during Preliminary Diagnosis.

$$\text{PD\_Rdy2\_Dly} = \Delta_{PD,OW} \tag{12.21}$$

The current design of the RPP imposes the following constraint.

$$PD\_Rdy2\_Dly \geq 1 \quad (12.22)$$

12.3.3. ID_Wnd_Dly

ID_Wnd_Dly denotes the delay to open the reception window in process P1 of the Initial Diagnosis protocol. This delay is measured from the clock edge immediately following the issue of the command by the MCU.

$$ID\_Wnd\_Dly = \Delta_{ID,P1,RCVWND} - 1 \quad (12.23)$$

The current design of the RPP imposes the following constraint.

$$ID\_Wnd\_Dly \geq 0 \quad (12.24)$$

12.3.4. ID_Wnd_Sz

IS_Wnd_Sz denotes the size of the reception window in process P1 of the Initial Diagnosis protocol.

$$ID\_Wnd\_Sz = W_{ID,Deskew} \quad (12.25)$$

The current design of the RPP imposes the following constraint.

$$ID\_Wnd\_Sz \geq 1 \quad (12.26)$$

12.3.5. IS_Wnd_Dly

IS_Wnd_Dly denotes the delay to open the reception window for the Initial Synchronization protocol. This delay is measured from the clock edge at which the reception window of the Initial Diagnosis protocol closes to the clock edge at which the reception window of the Initial Synchronization protocol opens.

$$IS\_Wnd\_Dly = C_{ID,P1} + \Delta_{ID,P1,C\text{-}END} + \Delta_{IS,begin} + \Delta_{IS,P1,RCVWND} \quad (12.27)$$

The current design of the RPP imposes the following constraint.

$$IS\_Wnd\_Dly \geq 1 \quad (12.28)$$

12.3.6. SP_INIT_Wnd_Dly(Node_Kind)

SP_INIT_Wnd_Dly denotes the delay to open the reception window for receiving INIT messages during the execution of the Synchronization Preservation protocol. This delay is measured from the clock edge immediately following the edge at which the MCU issues the command to the clock edge at which the reception window opens.

12.3.6.1. RMU

The RMUs receive INIT messages in process P1.

$$SP\_INIT\_Wnd\_Dly(RMU) = B_{SP,P0} + R_{PP} - W_{Deskew,pre} - 1 \quad (12.29)$$

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Wnd\_Dly(RMU) \geq 0 \quad (12.30)$$

12.3.6.2. BIU

The BIUs receive INIT messages in process P2.

$$SP\_INIT\_Wnd\_Dly(BIU) = B_{SP,P0} + R_{SP,P0\text{-}P2} - \Delta_{SP,P2,RCV}|_{max} - 1 \quad (12.31)$$

Note that the reception variables for process P2 are computed using $\pi_{P0}|_{P2,RCV}$ in consideration of the joining BIUs that synchronize using the Synchronization Capture protocol during the previous execution of the Synchronization Preservation protocol (see Section 5.10.2).

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Wnd\_Dly(BIU) \geq 0 \quad (12.32)$$

12.3.7. SP_INIT_Wnd_Sz(Node_Kind)

SP_INIT_Wnd_Sz denotes the size of the reception window for receiving INIT messages during the execution of the Synchronization Preservation protocol.

12.3.7.1. RMU

For the RMUs in process P1:

$$SP\_INIT\_Wnd\_Sz(RMU) = W_{Deskew} \quad (12.33)$$

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Wnd\_Sz(RMU) \geq 1 \quad (12.34)$$

12.3.7.2. BIU

For the BIUs in process P2:

$$SP\_INIT\_Wnd\_Sz(BIU) = 2\Delta_{SP,P2,RCV}|_{max} + 1 \quad (12.35)$$

Note that the reception variables for process P2 are computed using $\pi_{SP,P0}|_{P2,RCV}$ in consideration of the

joining BIUs that synchronize using the Synchronization Capture protocol during the previous execution of the Synchronization Preservation protocol (see Section 5.10.2).

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Wnd\_Sz(BIU) \geq 1 \tag{12.36}$$

12.3.8. SP_ECHO_Wnd_Dly(Node_Kind)

SP_ECHO_Wnd_Dly denotes the delay to open the reception window for receiving ECHO messages during the execution of the Synchronization Preservation protocol. This delay is measured from the clock edge immediately following the edge at which the Accept(INIT) output is asserted to the clock edge at which the reception window opens.

12.3.8.1. RMU

The RMUs receive ECHO messages in process P3.

$$SP\_ECHO\_Wnd\_Dly(RMU) = R_{SP,P1\text{-}P3} - \Delta_{SP,P3,RCV}|_{max} - 1 \tag{12.37}$$

The current design of the RPP imposes the following constraint.

$$SP\_ECHO\_Wnd\_Dly(RMU) \geq 0 \tag{12.38}$$

12.3.8.2. BIU

The BIUs receive ECHO messages in process P4.

$$SP\_ECHO\_Wnd\_Dly(BIU) = R_{SP,P2\text{-}P4} - \Delta_{SP,P4,RCV}|_{max} - 1 \tag{12.39}$$

The current design of the RPP imposes the following constraint.

$$SP\_ECHO\_Wnd\_Dly(BIU) \geq 0 \tag{12.40}$$

12.3.9. SP_ECHO_Wnd_Sz(Node_Kind)

SP_ECHO_Wnd_Sz denotes the size of the reception window for receiving ECHO messages during the execution of the Synchronization Preservation protocol.

12.3.9.1. RMU

For the RMUs in process P3:

$$SP\_ECHO\_Wnd\_Sz(RMU) = 2\Delta_{SP,P3,RCV}|_{max} + 1 \tag{12.41}$$

The current design of the RPP imposes the following constraint.

$$SP\_ECHO\_Wnd\_Sz(RMU) \geq 1 \tag{12.42}$$

12.3.9.2. BIU

For the BIUs in process P4:

$$SP\_ECHO\_Wnd\_Sz(BIU) = 2\Delta_{SP,P4,RCV}|_{max} + 1 \tag{12.43}$$

The current design of the RPP imposes the following constraint.

$$SP\_ECHO\_Wnd\_Sz(BIU) \geq 1 \tag{12.44}$$

12.3.10. INIT_Pls_Dly(Node_Kind)

INIT_Pls_Dly denotes the delay from the clock edge at which an INIT message is received to the clock edge at which the corresponding event is presented at the output of the Input Unit. These INIT signals are generated by the Input Unit's Sync Delay subunits, whose outputs are directly connected to the outputs of the Input Unit. For proper operation, the INIT outputs should be asserted at least one tick after the READY output is asserted during Synchronization Preservation. The READY output is asserted SP_INIT_Wnd_Sz ticks after the opening of the reception window.

12.3.10.1. RMU

For the RMUs in process P1:

$$INIT\_Pls\_Dly(RMU) = SP\_INIT\_Wnd\_Sz(RMU) + 1 \tag{12.45}$$

The current design of the RPP imposes the following constraint.

$$INIT\_Pls\_Dly(RMU) \geq 1 \tag{12.46}$$

12.3.10.2. BIU

For the BIUs in process P2:

$$INIT\_Pls\_Dly(BIU) = SP\_INIT\_Wnd\_Sz(BIU) + 1 \tag{12.47}$$

The current design of the RPP imposes the following constraint.

$$INIT\_Pls\_Dly(BIU) \geq 1 \tag{12.48}$$

12.3.11. INIT_Skw(Node_Kind)

INIT_Skw denotes the maximum valid observed skew between INIT messages from trustworthy nodes during the Synchronization Preservation protocol.

12.3.11.1. RMU

For RMUs in process P1:

$$\text{INIT\_Skw(RMU)} = \Pi_{SP,P1,RCV} \tag{12.49}$$

The current design of the RPP imposes the following constraint.

$$\text{INIT\_Skw(RMU)} \geq 1 \tag{12.50}$$

12.3.11.2. BIU

For BIUs in process P2:

$$\text{INIT\_Skw(BIU)} = \Pi_{SP,P2,RCV} \tag{12.51}$$

The current design of the RPP imposes the following constraint.

$$\text{INIT\_Skw(BIU)} \geq 1 \tag{12.52}$$

12.3.12. ECHO_Pls_Dly(Node_Kind)

ECHO_Pls_Dly denotes the delay from the clock edge at which an ECHO message is received to the clock edge at which the corresponding event is presented at the output of the Input Unit. These ECHO signals are generated by the Input Unit's Sync Delay subunits, whose outputs are directly connected to the outputs out the Input Unit. For proper operation, the ECHO outputs should be asserted at least one tick after the READY output is asserted during Synchronization Preservation. The READY output is asserted SP_ECHO_Wnd_Sz ticks after the opening of the reception window.

12.3.12.1. RMU

For the RMUs in process P1:

$$\text{ECHO\_Pls\_Dly(RMU)} = \text{SP\_ECHO\_Wnd\_Sz(RMU)} + 1 \tag{12.53}$$

The current design of the RPP imposes the following constraint.

$$\text{ECHO\_Pls\_Dly(RMU)} \geq 1 \tag{12.54}$$

12.3.12.2. BIU

For the BIUs in process P2:

$$ECHO\_Pls\_Dly(BIU) = SP\_ECHO\_Wnd\_Sz(BIU) + 1 \quad (12.55)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Pls\_Dly(BIU) \geq 1 \quad (12.56)$$

12.3.13. ECHO_Skw(Node_Kind)

ECHO_Skw denotes the maximum valid observed skew between ECHO messages from trustworthy nodes during the Synchronization Preservation, Initial Synchronization, and Synchronization Capture protocols.

12.3.13.1. RMU

For RMUs in process P3 or P3C:

$$ECHO\_Skw(RMU) = \Pi_{SP,P3,RCV} \quad (12.57)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Skw(RMU) \geq 1 \quad (12.58)$$

12.3.13.2. BIU

For BIUs in process P4 or P4C:

$$ECHO\_Skw(BIU) = \Pi_{SP,P4,RCV} \quad (12.59)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Skw(BIU) \geq 1 \quad (12.60)$$

12.3.14. CD_Wnd1_Dly

CD_Wnd1_Dly denotes the delay to open the reception window for receiving messages in process P1 during the execution of Collective Diagnosis or Collective Diagnosis Acquisition. This delay is measured from the clock edge immediately following the edge at which the MCU issues the command to the clock edge at which the reception window opens.

$$CD\_Wnd1\_Dly = \Delta_{CD,P1,RCVWND} - 1 \quad (12.61)$$

The current design of the RPP imposes the following constraint.

$$CD\_Wnd1\_Dly \geq 0 \quad (12.62)$$

12.3.15. CD_Wnd2_Dly

CD_Wnd2_Dly denotes the delay to open the reception window for receiving messages in processes P2, P3, and P4 during the execution of Collective Diagnosis or Collective Diagnosis Acquisition. This delay is measured from the clock edge at which the previous receive window is closed until the edge at which the next window opens.

$$CD\_Wnd2\_Dly = \Delta_{CD,P2,RCVWND} \quad (12.63)$$

The current design of the RPP imposes the following constraint.

$$CD\_Wnd2\_Dly \geq 1 \quad (12.64)$$

12.3.16. SU_P1_Wnd1_Dly(Node_Kind)

SU_P1_Wnd1_Dly denotes the delay to open the reception window for receiving the first set of messages during the execution of the Schedule Update protocol. This delay is measured from the clock edge immediately following the edge at which the MCU issues the command to the clock edge at which the reception window opens.

12.3.16.1. RMU

For RMUs in process P1:

$$SU\_P1\_Wnd1\_Dly(RMU) = \Delta_{SU,P1,RCVWND} - 1 \quad (12.65)$$

The current design of the RPP imposes the following constraint.

$$SU\_P1\_Wnd1\_Dly(RMU) \geq 0 \quad (12.66)$$

12.3.16.2. BIU

For BIUs in process P2:

$$SU\_P1\_Wnd1\_Dly(BIU) = \Delta_{SU,P2,RCVWND} - 1 \quad (12.67)$$

The current design of the RPP imposes the following constraint.

$$SU\_P1\_Wnd1\_Dly(BIU) \geq 0 \quad (12.68)$$

12.3.17. SU_P2_Wnd1_Dly(Node_Kind)

SU_P2_Wnd1_Dly denotes the delay to open the reception window for receiving the second message

stream in the execution of the Schedule Update protocol. This delay is measured from the clock edge at which the receive window is closed for the last message of the first stream to the clock edge at which the receive window is opened for the first message of the second stream.

12.3.17.1. RMU

For RMUs in process P3:

$$SU\_P2\_Wnd1\_Dly(RMU) = (T_{SU,P3,RCV,E,0} - W_{Deskew,pre}) - (T_{SU,P1,RCV,E,N-1} + W_{Deskew,post}) \quad (12.69)$$

The current design of the RPP imposes the following constraint.

$$SU\_P2\_Wnd1\_Dly(RMU) \geq 1 \quad (12.70)$$

12.3.17.2. BIU

For BIUs in process P4:

$$SU\_P2\_Wnd1\_Dly(BIU) = (T_{SU,P4,RCV,E,0} - W_{Deskew,pre}) - (T_{SU,P2,RCV,E,N-1} + W_{Deskew,post}) \quad (12.71)$$

The current design of the RPP imposes the following constraint.

$$SU\_P2\_Wnd1\_Dly(BIU) \geq 1 \quad (12.72)$$

12.3.18. SU_Wnd2_Dly

SU_Wnd2_Dly denotes the time gap between consecutive reception windows of a message stream, and it applies to the first and second message streams during the execution of the Schedule Update protocol. This delay is measured from the clock edge at which the reception window of a message ends to the edge at which the next reception window of the stream begins. This delay applies equally to BIUs and RMUs in processes P1, P2, P3, and P4. If the reception windows overlap, this variable is set to 0.

$$SU\_Wnd2\_Dly = \begin{cases} \Lambda_{SU} - W_{Deskew}, & \text{for } \Lambda_{SU} - W_{Deskew} > 0 \\ 0, & \text{for } \Lambda_{SU} - W_{Deskew} \leq 0 \end{cases} \quad (12.73)$$

The current design of the RPP imposes the following constraint.

$$SU\_Wnd2\_Dly \geq 0 \quad (12.74)$$

12.3.19. SU_DII

SU_DII denotes the data introduction interval for the streams of the Schedule Update protocol.

$$SU\_DII = \Lambda_{SU} \quad (12.75)$$

The current design of the RPP imposes the following constraint.

$$SU\_DII \geq 1 \tag{12.76}$$

12.3.20. SU_Max_Buff_Cnt

SU_Max_Buff_Cnt denotes the maximum input buffer load for normal operation when the reception windows overlap during the execution of the Schedule Update protocol. To specify the value of this variable we leverage the timing analysis for point-to-point communication of message streams. The current version of the RPP uses synchronous FIFOs as input buffers. Based on the analysis in Section 4 of this document:

$$SU\_Max\_Buff\_Cnt = K - \lfloor (K-1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin})/\Lambda_{stream} + 1 \rfloor \tag{12.77}$$

For:

$K = N$,

$\Delta_{PROC,begin} = \Delta_{PP,RCV}|_{abs\text{-}max} + 1$, and

$\Lambda_{stream} = \Lambda_{SU}$,

then:

$$SU\_Max\_Buff\_Cnt = N - \lfloor (N-1)/(1 + \rho_0)^2 - W_{Deskew}/\Lambda_{SU} + 1 \rfloor \tag{12.78}$$

The current design of the RPP imposes the following constraint.

$$SU\_Max\_Buff\_Cnt \geq 1 \tag{12.79}$$

12.3.21. PE_Wnd1_Dly(Node_Kind)

PE_Wnd1_Dly denotes the delay to open the reception window for receiving the first message during the execution of the PE Communication protocols. This delay is measured from the clock edge immediately following the edge at which the MCU issues the command to the clock edge at which the reception window opens.

12.3.21.1. RMU

For RMUs in process P1:

$$PE\_Wnd1\_Dly(RMU) = \Delta_{PE,P1,RCVWND,sched} - 1 \tag{12.80}$$

The current design of the RPP imposes the following constraint.

$$PE\_Wnd1\_Dly(RMU) \geq 0 \tag{12.81}$$

12.3.21.2. BIU

For BIUs in process P2:

$$PE\_Wnd1\_Dly(BIU) = \Delta_{PE,P2,RCVWND,sched} - 1 \quad (12.82)$$

The current design of the RPP imposes the following constraint.

$$PE\_Wnd1\_Dly(BIU) \geq 0 \quad (12.83)$$

12.3.22. PE_Wnd2_Dly

PE_Wnd2_Dly denotes the time gap between consecutive reception windows of the message stream during the execution of the PE Communication protocols. This delay is measured from the clock edge at which the previous receive window is closed to the edge at which the next window opens. This delay applies equally to BIUs and RMUs. If the windows overlap, this variable should be set to 0.

$$PE\_Wnd2\_Dly = \begin{cases} \Lambda_{PE} - W_{Deskew}, & \text{for } \Lambda_{PE,sched} - W_{Deskew} > 0 \\ 0, & \text{for } \Lambda_{PE,sched} - W_{Deskew} \leq 0 \end{cases} \quad (12.84)$$

The current design of the RPP imposes the following constraint.

$$PE\_Wnd2\_Dly \geq 0 \quad (12.85)$$

12.3.23. PE_DII

PE_DII denotes the data introduction interval for the PE Communication protocol.

$$PE\_DII = \Lambda_{PE,sched} \quad (12.86)$$

The current design of the RPP imposes the following constraint.

$$PE\_DII \geq 1 \quad (12.87)$$

12.3.24. PE_Max_Buff_Cnt

PE_Max_Buff_Cnt denotes maximum input buffer load for normal operation when the reception windows overlap during the execution of the PE Communication protocols. To specify the value of this variable we leverage the timing analysis for point-to-point communication of message streams. The current version of the RPP uses synchronous FIFOs as input buffers.

$$PE\_Max\_Buff\_Cnt = K - \lfloor (K-1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin})/\Lambda_{stream} + 1 \rfloor \quad (12.88)$$

For:

$$K = K_{PE,sched}|_{max} + 1,$$

$\Delta_{PROC,begin} = \Delta_{PP,RCV}|_{abs\text{-}max} + 1$, and

$\Lambda_{stream} = \Lambda_{PE,sched}$,

then:

$$PE\_Max\_Buff\_Cnt = K_{PE,sched}|_{max} - \lfloor (K_{PE,sched}|_{max} - 1)/(1 + \rho_0)^2 - W_{Deskew}/\Lambda_{PE,sched} + 1 \rfloor \quad (12.89)$$

Since the schedule can be dynamically updated, the actual number of messages in the stream (denoted generically by K) can vary from cycle to cycle. Here, we use the largest allowed stream during PE Communication as the reference to compute PE_Max_Buff_Cnt.

The value $K_{PE,sched}|_{max} + 1$ includes the accusations message sent after the PE messages. (Note that this assumes that the accusations message will be sent with the same DII as the rest of the stream.)

The current design of the RPP imposes the following constraint.

$$PE\_Max\_Buff\_Cnt \geq 1 \quad (12.90)$$

12.3.25. Syncns_Wnd_Sz

Syncns_Wnd_Sz denotes the synchronous reception window size for the time-driven protocols.

$$Syncns\_Wnd\_Sz = W_{Deskew} \quad (12.91)$$

The current design of the RPP imposes the following constraint.

$$Syncns\_Wnd\_Sz \geq 1 \quad (12.92)$$

12.3.26. Frm_Sync_Gap(Node_Kind)

Frm_Sync_Gap denotes the time between valid ECHO messages that a node must search for in order to achieve frame synchronization during the execution of the Synchronization Acquisition protocol. This time is equal to the bound on the observed relative skew of received ECHO messages from trustworthy nodes.

12.3.26.1. RMU

For RMUs in process P3C:

$$Frm\_Sync\_Gap(RMU) = \Pi_{SP,P3C,RCV} \quad (12.93)$$

The current design of the RPP imposes the following constraint.

$$Frm\_Sync\_Gap(RMU) \geq 1 \quad (12.94)$$

12.3.26.2. BIU

For BIUs in process P4C:

$$\text{Frm\_Sync\_Gap(BIU)} = \Pi_{SP,P4C,RCV} \tag{12.95}$$

The current design of the RPP imposes the following constraint.

$$\text{Frm\_Sync\_Gap(BIU)} \geq 1 \tag{12.96}$$

12.4. Input Diagnostics Unit (IDU)

12.4.1. PD_ECHO_Cnt1_Lo

PD_ECHO_Cnt1_Lo denotes the minimum number of ECHO messages that is expected from a node of the opposite kind during the first observation phase of Preliminary Diagnosis (a.k.a., Local Diagnosis Acquisition).

$$\text{PD\_ECHO\_Cnt1\_Lo} = \lfloor (\Delta_{PD,OW} - 1)/[(1 + \rho_0)p_{max}] \rfloor \tag{12.97}$$

The current design of the RPP imposes the following constraint.

$$\text{PD\_ECHO\_Cnt1\_Lo} \geq 0 \tag{12.98}$$

12.4.2. PD_ECHO_Cnt1_Hi

PD_ECHO_Cnt1_Hi denotes the maximum number of ECHO messages that is expected from a node of the opposite kind during the first observation phase of Preliminary Diagnosis.

$$\text{PD\_ECHO\_Cnt1\_Hi} = \lfloor (1 + \rho_0)(\Delta_{PD,OW} - 1)/p_{min} \rfloor + 1 \tag{12.99}$$

The current design of the RPP imposes the following constraint.

$$\text{PD\_ECHO\_Cnt1\_Hi} \geq 0 \tag{12.100}$$

12.4.3. PD_ECHO_Cnt2_Lo

PD_ECHO_Cnt2_Lo denotes the minimum total number of ECHO messages that is expected from a node of the opposite kind by the end of the second observation phase during Preliminary Diagnosis.

$$\text{PD\_ECHO\_Cnt2\_Lo} = \lfloor (2\Delta_{PD,OW} - 1)/[(1 + \rho_0)p_{max}] \rfloor \tag{12.101}$$

The current design of the RPP imposes the following constraint.

$$\text{PD\_ECHO\_Cnt2\_Lo} \geq 0 \tag{12.102}$$

12.4.4. PD_ECHO_Cnt2_Hi

PD_ECHO_Cnt2_Hi denotes the minimum total number of ECHO messages that is expected from a node of the opposite kind by the end of the second observation phase during Preliminary Diagnosis.

$$\text{PD\_ECHO\_Cnt2\_Hi} = \lfloor (1 + \rho_0)(2\Delta_{PD,OW} - 1)/p_{min} \rfloor + 1 \quad (12.103)$$

The current design of the RPP imposes the following constraint.

$$\text{PD\_ECHO\_Cnt2\_Hi} \geq 0 \quad (12.104)$$

12.5. Route-and-Vote Unit (RVU)

12.5.1. INIT_Skw(Node_Kind)

INIT_Skw denotes the maximum valid observed skew between INIT messages from trustworthy nodes during the Synchronization Preservation protocol. This parameter is also used in the Input Unit.

12.5.1.1. RMU

For RMUs in process P1:

$$\text{INIT\_Skw(RMU)} = \Pi_{SP,P1,RCV} \quad (12.105)$$

The current design of the RPP imposes the following constraint.

$$\text{INIT\_Skw(RMU)} \geq 1 \quad (12.106)$$

12.5.1.2. BIU

For BIUs in process P2:

$$\text{INIT\_Skw(BIU)} = \Pi_{SP,P2,RCV} \quad (12.107)$$

The current design of the RPP imposes the following constraint.

$$\text{INIT\_Skw(BIU)} \geq 1 \quad (12.108)$$

12.5.2. ECHO_Skw(Node_Kind)

ECHO_Skw denotes the maximum valid observed skew between ECHO messages from trustworthy nodes during the Synchronization Preservation, Initial Synchronization, and Synchronization Capture protocols. This parameter is also used in the Input Unit.

12.5.2.1. RMU

For RMUs in process P3 or P3C:

$$ECHO\_Skw(RMU) = \Pi_{SP,P3,RCV} \quad (12.109)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Skw(RMU) \geq 1 \quad (12.110)$$

12.5.2.2. BIU

For BIUs in process P4 or P4C:

$$ECHO\_Skw(BIU) = \Pi_{SP,P4,RCV} \quad (12.111)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Skw(BIU) \geq 1 \quad (12.112)$$

12.5.3. SCIS_Sync_Rst_Dly(Node_Kind)

SCIS_Sync_Rst_Dly denotes the sync reset delay for the Synchronization Capture and Initial Synchronization protocols. This delay is measured from the time an Accept(ECHO) output is asserted until the Sync_Reset signal is asserted by the Route-and-Vote Unit. The Local Time is set to 0 at the next clock edge.

12.5.3.1. RMU

For RMUs in process P3 or P3C:

$$SCIS\_Sync\_Rst\_Dly(RMU) = H_{P3} - 1 \quad (12.113)$$

The current design of the RPP imposes the following constraint.

$$SCIS\_Sync\_Rst\_Dly(RMU) \geq 1 \quad (12.114)$$

12.5.3.2. BIU

For BIUs in process P4 or P4C:

$$SCIS\_Sync\_Rst\_Dly(BIU) = H_{P4} - 1 \quad (12.115)$$

The current design of the RPP imposes the following constraint.

$$SCIS\_Sync\_Rst\_Dly(BIU) \geq 1 \quad (12.116)$$

12.5.4. SP_Sync_Rst_Dly(Node_Kind)

SCIS_Sync_Rst_Dly denotes the sync reset delay for the Synchronization Preservation protocol. This delay is measured from the time an Accept output is asserted until the Sync_Reset signal is asserted by the Route-and-Vote Unit. The Local Time is set to 0 at the next clock edge.

12.5.4.1. RMU

For Synchronization Preservation, the RMUs synchronize their Local Time with respect to the output of Accept(ECHO) in process P3 or P3C.

$$SP\_Sync\_Rst\_Dly(RMU) = H_{P3} - 1 \quad (12.117)$$

The current design of the RPP imposes the following constraint.

$$SP\_Sync\_Rst\_Dly(RMU) \geq 1 \quad (12.118)$$

12.5.4.2. BIU

For Synchronization Preservation, the RMUs synchronize their Local Time with respect to the output of Accept(INIT) in process P2.

$$SP\_Sync\_Rst\_Dly(BIU) = H_{P2} - 1 \quad (12.119)$$

The current design of the RPP imposes the following constraint.

$$SP\_Sync\_Rst\_Dly(BIU) \geq 1 \quad (12.120)$$

12.6. Status Monitoring Unit (SMU)

12.6.1. SA_Timeout

SA_Timeout denotes the timeout delay for the Synchronization Acquisition sequence, which includes the Frame Synchronization and Synchronization Capture protocols.

$$SA\_Timeout = \Delta_{SA}|_{max} \quad (12.121)$$

The current design of the RPP imposes the following constraint.

$$SA\_Timeout \geq 1 \quad (12.122)$$

12.6.2. IS_Timeout

IS_Timeout denotes the timeout delay for the Initial Synchronization protocol. For the current version of the ROBUS protocol processor, this is measured from the clock edge at which the Computation Process outputs the result for Initial Diagnosis to the synchronization reset during Initial Synchronization.

$$IS\_Timeout = \Delta_{ID,P1,C\text{-}END} + \Delta_{IS,begin} + \Delta_{IS}|_{max} \quad (12.123)$$

The current design of the RPP imposes the following constraint.

$$IS\_Timeout \geq 1 \quad (12.124)$$

12.6.3. SP_Timeout

SP_Timeout denotes the timeout delay for the resynchronization interval.

$$SP\_Timeout = P \quad (12.125)$$

The current design of the RPP imposes the following constraint.

$$SP\_Timeout \geq 1 \quad (12.126)$$

12.7. Output Unit (OU)

To determine the parameter values for the Output Unit, we must keep under consideration that there is a one-tick delay from the strobe generated by the OU controller to the strobe sent to the Communication Module.

12.7.1. ID_Snd_Dly

ID_Snd_Dly denotes the send delay for Initial Diagnosis. This delay is measured from the clock edge at which the MCU issues the command to one tick before the message is sent.

$$ID\_Snd\_Dly = S_{ID,P0} - 1 \quad (12.127)$$

The current design of the RPP imposes the following constraint.

$$ID\_Snd\_Dly \geq 0 \quad (12.128)$$

12.7.2. IS_INIT_Snd_Dly(Node_Kind)

IS_INIT_Snd_Dly denotes the send delay for the INIT message during Initial Synchronization.

12.7.2.1. RMU

For RMUs in process P1, this delay is measured from the time the output of Accept(INIT) is asserted to one tick before the OU controller asserts its strobe.

$$IS\_INIT\_Snd\_Dly(RMU) = B_{P1} - 1 \tag{12.129}$$

The current design of the RPP imposes the following constraint.

$$IS\_INIT\_Snd\_Dly(RMU) \geq 0 \tag{12.130}$$

12.7.2.2. BIU

For BIUs in process P0, this delay is measured from the clock edge at which the controller generates the strobe for the Initial Diagnosis message to the clock edge at which the controller generates the strobe for the INIT message.

$$IS\_INIT\_Snd\_Dly(BIU) = (T_{IS,P0,SND} - 1) - (T_{ID,P0,SND} - 1) \tag{12.131}$$

$$= R_{PP} + W_{ID,Deskew,post} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END} + \Delta_{IS,begin} + B_{IS,P0} \tag{12.132}$$

The current design of the RPP imposes the following constraint.

$$IS\_INIT\_Snd\_Dly(BIU) \geq 1 \tag{12.133}$$

12.7.3. SP_INIT_Snd_Dly(Node_Kind)

SP_INIT_Snd_Dly denotes the send delay for the INIT messages during the execution of the Synchronization Preservation protocol.

12.7.3.1. RMU

For RMUs in process P1, this delay is measured from the time the Accept(INIT) is asserted to one tick before the message is sent.

$$SP\_INIT\_Snd\_Dly(RMU) = B_{P1} - 1 \tag{12.134}$$

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Snd\_Dly(RMU) \geq 0 \tag{12.135}$$

12.7.3.2. BIU

For BIUs in process P0, this delay is measured from the time at which the MCU issues the command to one tick before the time at which the INIT message is sent.

$$SP\_INIT\_Snd\_Dly(BIU) = B_{SP,P0} - 1 \quad (12.136)$$

The current design of the RPP imposes the following constraint.

$$SP\_INIT\_Snd\_Dly(BIU) \geq 0 \quad (12.137)$$

12.7.4. ECHO_Snd_Dly(Node_Kind)

ECHO_Snd_Dly denotes the send delay for the ECHO messages during the execution of the synchronization protocols.

12.7.4.1. RMU

For RMUs in process P3, this delay is measured from the time the output of Accept(ECHO) is asserted to one tick before the message is sent.

$$ECHO\_Snd\_Dly(RMU) = B_{P3} - 1 \quad (12.138)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Snd\_Dly(RMU) \geq 0 \quad (12.139)$$

12.7.4.2. BIU

For BIUs in process P2, this delay is measured from the time the output of Accept(INIT) is asserted to one tick before the time at which the ECHO message is sent.

$$ECHO\_Snd\_Dly(BIU) = B_{P2} - 1 \quad (12.140)$$

The current design of the RPP imposes the following constraint.

$$ECHO\_Snd\_Dly(BIU) \geq 0 \quad (12.141)$$

12.7.5. CD_Snd1_Dly

CD_Snd1_Dly denotes the send delay for the first message during the execution of Collective Diagnosis and Collective Diagnosis Acquisition. This delay is measured from the clock edge at which the MCU issues the command to one tick before the clock edge at which the message is sent.

$$CD\_Snd1\_Dly = S_{CD,P0} - 1 \quad (12.142)$$

The current design of the RPP imposes the following constraint.

$$CD\_Snd1\_Dly \geq 1 \quad (12.143)$$

12.7.6. CD_Snd2_Dly

CD_Snd2_Dly denotes the send delay for the second, third, and fourth messages during the execution of Collective Diagnosis and Collective Diagnosis Acquisition. This delay is measured from the time the controller generates the strobe for a message to one tick before the next message is to be sent.

$$\text{CD\_Snd2\_Dly} = (T_{CD,P1,SND} - 1) - (T_{CD,P0,SND} - 1)$$

$$= R_{PP} + W_{Deskew,post} + C_{CD,P1} + S_{CD,P2} \quad (12.144)$$

The current design of the RPP imposes the following constraint.

$$\text{CD\_Snd2\_Dly} \geq 1 \quad (12.145)$$

12.7.7. SU_P1_Snd1_Dly(Node_Kind)

SU_P1_Snd1_Dly denotes the send delay for the first message of the first pass in the Schedule Update protocol. This delay is measured from the clock edge at which the MCU command is issued to one tick before the clock edge at which the message is sent.

12.7.7.1. RMU

For RMUs in process P1:

$$\text{SU\_P1\_Snd1\_Dly(RMU)} = T_{SU,P1,SND,0} - T_{SU} - 1 \quad (12.146)$$

The current design of the RPP imposes the following constraint.

$$\text{SU\_P1\_Snd1\_Dly(RMU)} \geq 1 \quad (12.147)$$

12.7.7.2. BIU

For BIUs in process P0:

$$\text{SU\_P1\_Snd1\_Dly(BIU)} = S_{SU,P0} - 1 \quad (12.148)$$

The current design of the RPP imposes the following constraint.

$$\text{SU\_P1\_Snd1\_Dly(BIU)} \geq 0 \quad (12.149)$$

12.7.8. SU_P2_Snd1_Dly(Node_Kind)

SU_P2_Snd1_Dly denotes the send delay for the first message of the second stream in the Schedule Update protocol. This delay is measured from the clock edge at which the controller generates the strobe for the last message of the first stream to the clock edge at which the controller generates the strobe for the first message of the second stream.

For the current design of the ROBUS protocol processor, the Output Unit includes a buffer to store the messages of the second stream. At a minimum, this buffer introduces a one-tick delay to the processing of each message. This extra delay can be assigned to either the computation delays for processes P2 and P3 (i.e., $C_{SU,P2}$ and $C_{SU,P3}$) or the minimum send delay for processes P2 and P3 (i.e., $S_{SU,P2}|_{min}$ and $S_{SU,P3}|_{min}$).

12.7.8.1. RMU

For RMUs in process P3:

$$SU\_P2\_Snd1\_Dly(RMU) = \Delta_{SU,STREAM,P0\text{-}P2}|_{RMU} \tag{12.150}$$

The current design of the RPP imposes the following constraint.

$$SU\_P2\_Snd1\_Dly(RMU) \geq 1 \tag{12.151}$$

12.7.8.2. BIU

For BIUs in process P2:

$$SU\_P2\_Snd1\_Dly(BIU) = \Delta_{SU,STREAM,P0\text{-}P2}|_{BIU} \tag{12.152}$$

The current design of the RPP imposes the following constraint.

$$SU\_P2\_Snd1\_Dly(BIU) \geq 1 \tag{12.153}$$

12.7.9. SU_DII

SU_DII denotes the data introduction interval for the Schedule Update protocol. This is the same parameter used in the Input Unit. This parameters is also used in the Input Unit.

$$SU\_DII = \Lambda_{SU} \tag{12.154}$$

The current design of the RPP imposes the following constraint.

$$SU\_DII \geq 1 \tag{12.155}$$

12.7.10. PE_Snd1_Dly(Node_Kind)

PE_Snd1_Dly denotes the send delay for the first message in PE Communication. This delay is measured from the clock edge at which the command is issued to one tick before the clock edge at which the message is sent.

12.7.10.1. RMU

For RMUs in process P1:

$$PE\_Snd1\_Dly(RMU) = T_{SU,P1,SND,sched,0} - T_{PE} - 1 \quad (12.156)$$

The current design of the RPP imposes the following constraint.

$$PE\_Snd1\_Dly(RMU) \geq 1 \quad (12.157)$$

12.7.10.2. BIU

For BIUs in process P0:

$$PE\_Snd1\_Dly(BIU) = S_{PE,P0,sched} - 1 \quad (12.158)$$

The current design of the RPP imposes the following constraint.

$$PE\_Snd1\_Dly(BIU) \geq 1 \quad (12.159)$$

12.7.11. PE_DII

PE_DII denotes the data introduction interval for PE Communication. This parameter is also used in the Input Unit.

$$PE\_DII = \Lambda_{PE,sched} \quad (12.160)$$

The current design of the RPP imposes the following constraint.

$$PE\_DII \geq 1 \quad (12.161)$$

13. Structural parameters

This section presents the formulas for the structural parameters in the VHDL description of the RPP. The parameters are grouped according to the corresponding RPP unit. Interface-level structural parameters are also defined.

13.1. Interface-level structural parameters

13.1.1. N

N denotes the number of BIUs in the system. The RPP can operate in a system with one or more BIUs.

$$N \geq 1 \tag{13.1}$$

13.1.2. M

M denotes the number of RMUs in the system. The RPP can operate in a system with one or more RMUs.

$$M \geq 1 \tag{13.2}$$

13.1.3. Num_Lnk_Synd

Num_Lnk_Synd (= L_{LS}) denotes the number of syndrome inputs provided by the Communication Module for each receiver port. The current description of the RPP assumes that there is at least one link syndrome for each input port.

$$Num\_Lnk\_Synd \geq 1 \tag{13.3}$$

For an implementation that does not have link syndromes, Num_Lnk_Synd should be set to one and then the input-syndrome signals should be set to the de-asserted value.

13.1.4. Payload_Width

Payload_Width (= L_{PF}) denotes the width of the Payload field for the ROBUS messages. This width must be large enough to satisfy for the following constraints: fit the Payload width requirement for PE messages; fit the width required to send Collective Diagnosis messages; fit the width required for the BIUs to send their Node_Id to the PEs; fit Max_Num_PE_Msg expressed in binary code; and fit a different binary value for each SPECIAL ROBUS message.

Let Max_PE_Payload_Width denote the Payload width requirement for PE messages. The width requirement for Collective Diagnosis messages is max(N, M). The width required for BIU node Ids is $\lceil \log_2(N) \rceil$. (Note that $N > \lceil \log_2(N) \rceil$.) The minimum width requirement to fit the maximum number of PE

messages is $\lceil \log_2(\text{Max\_Num\_PE\_Msg}) \rceil$. We would like Max_Payload to be greater then or equal to Max_Num_PE_Msg in order to be able to allocate all the available bandwidth to a single PE.

Let Num_Special_Msg denote the number of SPECIAL ROBUS messages defined. The minimum width required to assign a different binary value to each SPECIAL message is $\lceil \log_2(\text{Num\_Special\_Msg}) \rceil$. Then:

$$\text{Payload\_Width} \geq \max(\text{Max\_PE\_Payload\_Width}, N, M, \lceil \log_2(\text{Max\_Num\_PE\_Msg} + 1) \rceil, \lceil \log_2(\text{Num\_Special\_Msg}) \rceil) \quad (13.4)$$

13.1.5. Max_Payload

Max_Payload denotes the maximum value of the Payload field in binary code.

$$\text{Max\_Payload} = 2^{\text{Payload\_Width}} - 1 \quad (13.5)$$

13.2. Mode Control Unit (MCU)

13.2.1. Max_Diag_Cyc

Max_Diag_Cyc denotes the maximum value of the Diagnostic Cycle counter. For the current version of the ROBUS protocols, a node in Clique Join mode goes through two Diagnostic Cycles before being allowed to join a clique. The Diagnostic Cycle counter counts up starting from 0. The value of this parameter must satisfy the following constraint.

$$\text{Max\_Diag\_Cyc} \geq 1 \quad (13.6)$$

13.2.2. Max_MCU_LT

Max_MCU_LT denotes the maximum value of the Local Time counter. The Local Time counter must be able to count at least up to the maximum duration of a resynchronization cycle.

$$\text{Max\_MCU\_LT} \geq P \quad (13.7)$$

13.2.3. Max_MCU_Tmr

Max_MCU_Tmr denotes the maximum value of the general-purpose timer in the MCU. The maximum count must be greater than or equal to the maximum loaded value.

$$\text{Max\_MCU\_Tmr} \geq \max(\text{Min\_Cmd\_DII}, \text{ST\_Xtra\_Dly}, \text{ID\_Dly}) \quad (13.8)$$

13.3. Schedule Processor

13.3.1. Max_PE_Msg_Cnt

Max_PE_Msg_Cnt denotes the maximum value of the counter for the total number of scheduled PE messages. For this counter, we want to prevent an overflow condition, which could cause a missed detection of an excessive number of scheduled messages.

Since the error detector has a latching output, we choose Max_PE_Msg_Cnt to be greater than or equal to Max_Payload. This ensures that the counter cannot overflow at the same time that its count exceeds Max_Num_PE_Msg.

$$\text{Max\_PE\_Msg\_Cnt} \geq 2*\text{Max\_Payload} \tag{13.9}$$

13.3.2. Sched_FIFO_Depth

Sched_FIFO_Depth denotes the depth of the schedule memory. Since a schedule consists of N messages, the FIFO must be at least N deep.

$$\text{Sched\_FIFO\_Depth} \geq N \tag{13.10}$$

13.4. Input Unit (IU)

13.4.1. Input_FIFO_Depth

Input_FIFO_Depth denotes the depth of the input FIFO at each input port. This buffer must be large enough to hold at least one message more than the maximum loads for the Schedule Update and PE Communication protocols. The extra message is required to ensure that if an overload occurs, then the error is recorded in the buffer at least once.

$$\text{Input\_FIFO\_Depth} \geq \max(\text{SU\_Max\_Buff\_Cnt}, \text{PE\_Max\_Buff\_Cnt}) + 1 \tag{13.11}$$

13.4.2. Max_Frm_Sync_Tmr

Max_Frm_Sync_Tmr denotes the maximum value of the timer in the Frame Synchronizer. This timer is used to measure the time between ECHO messages.

$$\text{Max\_Frm\_Sync\_Tmr} \geq \max(\text{Frm\_Sync\_Gap(RMU)}, \text{Frm\_Sync\_Gap(BIU)}) \tag{13.12}$$

13.4.3. Max_Sync_Dly_Tmr

Max_Sync_Dly_Tmr denotes the maximum value for the timers in the INIT and ECHO Delay units. The same structural constraint is used since the same component is instantiated for processing INITs and ECHOs. The same timer is used to implement the pulse delay and the pulse width.

Max_Sync_Dly_Tmr ≥ max(INIT_Pls_Dly(RMU), INIT_Pls_Dly(BIU),

ECHO_Pls_Dly(RMU), ECHO_Pls_Dly(BIU),

INIT_Skw(RMU) + 1, INIT_Skw(BIU) + 1,

ECHO_Skw(RMU) + 1, ECHO_Skw(BIU) + 1) (13.13)

13.4.4. Max_IU_Cntrlr_Tmr

Max_IU_Cntrlr_Tmr denotes the maximum value for the general-purpose timer in the IU controller. This timer is used to control the timing of the reception windows and the Ready signal.

Max_IU_Cntrlr_Tmr ≥ max(PD_Rdy1_Dly, PD_Rdy2_Dly, ID_Wnd_Dly,

ID_Wnd_Sz, IS_Wnd_Dly,

SP_INIT_Wnd_Dly(RMU), SP_INIT_Wnd_Dly(BIU),

SP_INIT_Wnd_Sz(RMU), SP_INIT_Wnd_Sz(BIU),

SP_ECHO_Wnd_Dly(RMU), SP_ECHO_Wnd_Dly(BIU),

SP_ECHO_Wnd_Sz(RMU), SP_ECHO_Wnd_Sz(BIU),

CD_Wnd1_Dly, CD_Wnd2_Dly,

SU_P1_Wnd1_Dly(RMU), SU_P1_Wnd1_Dly(BIU),

SU_P2_Wnd1_Dly(RMU), SU_P2_Wnd1_Dly(BIU),

SU_Wnd2_Dly, SU_DII,

PE_Wnd1_Dly(RMU), PE_Wnd1_Dly(BIU),

PE_Wnd2_Dly, PE_DII, Syncns_Wnd_Sz) (13.14)

13.4.5. Max_IU_Cntrlr_Cntr

Max_IU_Cntrlr_Cntr denotes the maximum value of the general-purpose counter in the IU controller. This counter is used to count the messages in a stream during the Schedule Update protocol.

Max_IU_Cntrlr_Cntr ≥ N (13.15)

13.5. Input Diagnostics Unit (IDU)

13.5.1. Max_Rate_Mon_Cntr

Max_Rate_Mon_Cntr denotes the maximum value of the Rate Monitor counter. This counter is used to count the number of received ECHO messages during both Preliminary Diagnosis windows. The maximum value of the counter must be greater than or equal to the maximum number of expected messages. The counter has an overflow output to detect the arrival of more ECHO messages than its capacity.

$$\text{Max\_Rate\_Mon\_Cntr} \geq \text{PE\_ECHO\_Cnt2\_Hi} \tag{13.16}$$

13.5.2. Max_Seq_Mon_Cntr

Max_Seq_Mon_Cntr denotes the maximum value of the Sequence Monitor counter. This counter is used to count the actual number of received messages out of the total number of expected messages during each of the Collective Diagnosis and Schedule Update protocols.

$$\text{Max\_Seq\_Mon\_Cntr} \geq \max(4, N) \tag{13.17}$$

13.6. Route-and-Vote Unit (RVU)

13.6.1. Max_RVU_Acpt_Tmr

Max_RVU_Acpt_Tmr denotes the maximum value of the timer in the Accept function module. This module is instantiated for the Accept(INIT) and Accept(ECHO) functions. The timer is used to check the relative skew of the received messages with respect to the selected message.

$$\text{Max\_RVU\_Acpt\_Tmr} \geq \max(\text{INIT\_Skw(RMU)}, \text{INIT\_Skw(BIU)}, \text{ECHO\_Skw(RMU)}, \text{ECHO\_Skw(BIU)}) \tag{13.18}$$

13.6.2. Max_RVU_Cntrlr_Tmr

Max_RVU_Cntrlr_Tmr denotes the maximum value of the timer used by the RVU controller. This timer is used to implement the synchronization-reset delays for Synchronization Capture, Initial Synchronization, and Synchronization Preservation.

$$\text{Max\_RVU\_Cntrlr\_Tmr} \geq \max(\text{SCIS\_Sync\_Rst\_Dly(RMU)}, \text{SCIS\_Sync\_Rst\_Dly(BIU)}, \text{SP\_Sync\_Rst\_Dly(RMU)}, \text{SP\_Sync\_Rst\_Dly(BIU)}) \tag{13.19}$$

13.7. Status Monitoring Unit (SMU)

13.7.1. Max_Timeout

Max_Timeout denotes the maximum value of the timeout timer.

$$\text{Max\_Timeout} \geq \max(\text{SA\_Timeout}, \text{IS\_Timeout}, \text{SP\_Timeout}) \quad (13.20)$$

13.7.2. Sent_FIFO_Depth

Sent_FIFO_Depth denotes the depth of the SMU FIFO. This FIFO is used to buffer sent PE messages by a source BIU until the corresponding result of the PE Broadcast protocol is computed.

Sent_FIFO_Depth ≥

$$\lfloor [2(R_{PP} + W_{Deskew,post}) + S_{PE,P1,sched} + (C_{PE,P1,sched} + C_{PE,P2,sched}) + 1]/\Lambda_{PE} \rfloor + 1 \quad (13.21)$$

13.8. Output Unit (OU)

13.8.1. Output_FIFO_Depth

Output_FIFO_Depth denotes the depth of the FIFO buffer in the Output Unit. This FIFO is used to buffer RVU messages during Collective Diagnosis processes P1, P2, and P3; for Schedule Update processes P1, P2, and P3; and during process P1 of PE Broadcast and Accusation Exchange. For the Schedule Update protocol, this buffer must hold at most all the messages in a stream: N. For Collective Diagnosis, the buffer must hold at most 1 message. For PE Communication, the load is determined by the DII and the delay between the time when the RVU outputs the message and the time when the OU reads the message.

$$\text{Output\_FIFO\_Depth} \geq \max(N, \text{ceil}((S_{PE,P1,sched} - 1)/\Lambda_{PE,sched}), \text{ceil}((S_{PE,P1,acc} - 1)/\Lambda_{PE,sched})) \quad (13.22)$$

13.8.2. Max_OU_Cntrlr_Tmr

Max_OU_Cntrlr_Tmr denotes the maximum value of the timer in the OU controller. This timer is used to control the timing of output messages.

Max_OU_Cntrlr_Tmr ≥ max(ID_Snd_Dly, IS_INIT_Snd_Dly(RMU), IS_INIT_Snd_Dly(BIU),

SP_INIT_Snd_Dly(RMU), SP_INIT_Snd_Dly(BIU),

ECHO_Snd_Dly(RMU), ECHO_Snd_Dly(BIU),

CD_Snd1_Dly, CD_Snd2_Dly,

SU_P1_Snd1_Dly(RMU), SU_P1_Snd1_Dly(BIU),

SU_P2_Snd1_Dly(RMU), SU_P2_Snd1_Dly(BIU), SU_DII,

PE_Snd1_Dly(RMU), PE_Snd1_Dly(BIU), PE_DII) (13.23)

13.8.3. Max_OU_Cntrlr_Cntr

Max_OU_Cntrl_Cntr denotes the maximum value of the counter in the OU controller. This counter is used to count the number of messages in a stream during the Schedule Update protocol.

Max_OU_Cntrlr_Cntr ≥ N (13.24)

14. Specifying a particular solution

This section lists the variables of the generic model that must be specified in order to compute the RPP behavioral and structural parameters for a particular implementation.

14.1. Platform Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	N	Unitless	Integer	≥ 1	≥ 1
2	M	Unitless	Integer	≥ 1	≥ 1
3	f_0	Hertz	Real	> 0	Arbitrary
4	ρ_0	Unitless	Real	≥ 0	Arbitrary
5	$d_{PP,l}$	Nominal clock ticks	Real	≥ 0	Arbitrary
6	$d_{PP,h}$	Nominal clock ticks	Real	≥ 0	Arbitrary
7	Λ_{Comm}	Local clock ticks	Integer	≥ 1	Arbitrary
8	δ_{POE}	Seconds	Real	≥ 0	Arbitrary
9	L_{LS}	Unitless	Integer	≥ 0	≥ 1
10	$L_{PF\|PE}$	Bits	Integer	≥ 1	≥ 1

Notes:

- **Items 1 and 2:** Number of BIUs and RMUs, respectively.
- **Item 3:** The generic model imposes no significant restrictions on the values of f_0. The valid value range for f_0 is determined by the implementation after place-and-route for the target technology.
- **Item 4:** $\rho_0 = 0$ corresponds to a perfect clock oscillator with no drift with respect to real time.
- **Items 5 and 6:** For some systems, $d_{PP,l}$ and $d_{PP,h}$ could be negligibly small, and thus the analysis should handle the case in which they are set to 0.
- **Item 8:** δ_{POE} could be negligibly small for some systems.
- **Item 9:** Number of syndromes for each receiver.
- **Item 10:** Required Payload field width for PE messages.

14.2. Environmental Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	δ_{FCP}	Nominal clock ticks	Real	≥ 0	Arbitrary

Notes:

- **Item 1:** δ_{FCP} could be negligibly small for some systems.

14.3. Operational-Delay Constraints

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	$\Delta_{ID,begin}$	Local clock ticks	Integer	≥ 0	≥ 2
2	$S_{ID,P0\|min}$	Local clock ticks	Integer	≥ 0	= 1

Item Number	Variable	Units	Type	Value Range	RPP Constraint
3	$\Delta_{ID,P1,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
4	$C_{ID,P1}$	Local clock ticks	Integer	≥ 0	= 1
5	$\Delta_{ID,P1,C\text{-}END}$	Local clock ticks	Integer	≥ 0	= 2
6	$\Delta_{IS,begin}$	Local clock ticks	Integer	≥ 0	≥ 0
7	$B_{IS,P0}\|_{min}$	Local clock ticks	Integer	≥ 0	= 0
8	$\Delta_{IS,P1,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 0
9	$B_{SP,P0}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
10	$B_{P1}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
11	$B_{P2}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
12	$B_{P3}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
13	$\Delta_{SP,P1,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
14	$\Delta_{SP,P2,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
15	$\Delta_{SP,P3,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
16	$\Delta_{SP,P4,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
17	$C_{SP,P1}$	Local clock ticks	Integer	≥ 0	≥ 4
18	$C_{SP,P2}$	Local clock ticks	Integer	≥ 0	≥ 4
19	$C_{SP,P3}$	Local clock ticks	Integer	≥ 0	≥ 4
20	$C_{SP,P4}$	Local clock ticks	Integer	≥ 0	≥ 4
21	$S_{CD,P0}\|_{min}$	Local clock ticks	Integer	≥ 0	= 2
22	$\Delta_{CD,P1,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
23	$C_{CD,P1}$	Local clock ticks	Integer	≥ 0	= 1
24	$S_{CD,P1}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
25	$\Delta_{CD,P2,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
26	$C_{CD,P2}$	Local clock ticks	Integer	≥ 0	= 1
27	$S_{CD,P2}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
28	$\Delta_{CD,P3,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
29	$C_{CD,P3}$	Local clock ticks	Integer	≥ 0	= 1
30	$S_{CD,P3}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
31	$\Delta_{CD,P4,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
32	$C_{CD,P4}$	Local clock ticks	Integer	≥ 0	= 1
33	$\Delta_{CD,P4,C\text{-}END}$	Local clock ticks	Integer	≥ 0	= 3
34	$\Delta_{SU,SND\text{-}MODE}$	Local clock ticks	Integer	≥ 0	= 1
35	$S_{SU,P0}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
36	$\Delta_{SU,P1,RCVWND}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
37	$C_{SU,P1}$	Local clock ticks	Integer	≥ 0	= 1
38	$S_{SU,P1}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
39	$C_{SU,P2}$	Local clock ticks	Integer	≥ 0	= 2
40	$S_{SU,P2}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
41	$C_{SU,P3}$	Local clock ticks	Integer	≥ 0	= 2
42	$S_{SU,P3}\|_{min}$	Local clock ticks	Integer	≥ 0	= 3
43	$C_{SU,P4}$	Local clock ticks	Integer	≥ 0	= 1
44	$\Delta_{SU,P4,C\text{-}END}$	Local clock ticks	Integer	≥ 0	= 5
45	$\Delta_{PE,SND\text{-}SA}$	Local clock ticks	Integer	≥ 0	= 2
46	$S_{PE,P0,sched}\|_{min}$	Local clock ticks	Integer	≥ 0	= 2
47	$\Delta_{PE,P1,RCVWND,sched}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
48	$C_{PE,P1,sched}$	Local clock ticks	Integer	≥ 0	= 1
49	$S_{PE,P1,sched}\|_{min}$	Local clock ticks	Integer	≥ 0	= 4
50	$C_{PE,P2,sched}$	Local clock ticks	Integer	≥ 0	= 1

Item Number	Variable	Units	Type	Value Range	RPP Constraint
51	$\Delta_{PE,P0,acc\_ready}$	Local clock ticks	Integer	≥ 1	= 1
52	$S_{PE,P0,acc}\|_{min}$	Local clock ticks	Integer	≥ 0	= 1
53	$C_{PE,P1,acc}$	Local clock ticks	Integer	≥ 0	= 1
54	$\Delta_{PE,P1,acc\_ready}$	Local clock ticks	Integer	≥ 1	= 1
55	$S_{PE,P1,acc}\|_{min}$	Local clock ticks	Integer	≥ 0	= 4
56	$C_{PE,P2,acc}$	Local clock ticks	Integer	≥ 0	= 1
57	$\Delta_{PE,P2,acc,C\text{-}END}$	Local clock ticks	Integer	≥ 0	= 3
58	Λ_{Comp}	Local clock ticks	Integer	≥ 1	= 1
59	$\Delta_{PD,begin}$	Local clock ticks	Integer	≥ 0	= 1
60	$\Delta_{FS,begin}$	Local clock ticks	Integer	≥ 0	= 0
61	$\Delta_{CDM\text{-}CIM}$	Local clock ticks	Integer	≥ 0	= 0
62	H_{P4}	Local clock ticks	Integer	≥ 0	≥ 4

Notes:

- **Items 1:** The RPP sub-units are ready to receive a command 2 ticks after the MCU asserts the Node_Reset signal.
- **Item 2:** $S_{ID,P0}|_{min}$ is determined by the minimum delay for the OU to assert the output strobe signal after the MCU issues the command.
- **Item 3:** $\Delta_{ID,P1,RCVWND}|_{min}$ is determined by the minimum delay for the IU to open an reception window after the MCU issues the command.
- **Item 4:** $C_{ID,P1}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 5:** $\Delta_{ID,P1,C\text{-}END}$ is taken as the delay from the IDU output until the MCU reads the failure signal from the SMU, which includes an assessment of the NDU output.
- **Item 6:** There is no need to delay the start of Initial Synchronization after Initial Diagnosis is complete.
- **Item 7:** The OUs at the BIUs are ready to send Initial Synchronization messages right after sending the Initial Diagnosis message. Therefore, in effect, the OU is ready to send immediately at the start of Initial Synchronization.
- **Item 8:** The IU can open the Initial Synchronization window just one tick after closing the window for Initial Diagnosis. Therefore, in effect, the IU is ready to open a window immediately at the start of Initial Synchronization.
- **Item 9:** $B_{SP,P0}|_{min}$ is determined by the minimum delay for the OU to set the output strobe signal after the MCU issues the command.
- **Items 10, 11, and 12:** $B_{P1}|_{min}$, $B_{P1}|_{min}$, and $B_{P1}|_{min}$ are taken as the minimum required delays to ensure the diagnosis is complete before the OU sends the message: 1 tick for the NDU, plus 1 tick for the SMU and MCU, plus 1 tick for the OU.
- **Item 13 and 14:** $\Delta_{SP,P1,RCVWND}|_{min}$ and $\Delta_{SP,P2,RCVWND}|_{min}$ are determined by the minimum delay for the IU to open a window after the MCU issues the command.
- **Items 15 and 16:** $\Delta_{SP,P3,RCVWND}|_{min}$ and $\Delta_{SP,P4,RCVWND}|_{min}$ are determined by the minimum delay for the IU to open a window after the corresponding Accept output is asserted.
- **Items 17, 18, 19, and 20:** $C_{SP,P1}$, $C_{SP,P2}$, $C_{SP,P3}$, and $C_{SP,P4}$ are determined by the minimum delay to generate an Accept output after the closing of an reception window.
- **Item 21:** $S_{CD,P0}|_{min}$ is determined by the minimum delay for the OU to set the output strobe signal after the MCU issues the command.
- **Item 22:** $\Delta_{SP,P1,RCVWND}|_{min}$ is determined by the minimum delay for the IU to open a window after the MCU issues the command.
- **Item 23:** $C_{CD,P1}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 24, 27, and 30:** $S_{CD,P1}|_{min}$, $S_{CD,P2}|_{min}$, and $S_{CD,P3}|_{min}$ are taken as the minimum required delays to ensure the diagnosis is complete before the OU sends the message: 1 tick for the NDU, plus 1 tick for the SMU and MCU, plus 1 tick for the OU.

- **Items 25, 28, and 31:** $\Delta_{CD,P2,RCVWND}|_{min}$, $\Delta_{CD,P3,RCVWND}|_{min}$, and $\Delta_{CD,P4,RCVWND}|_{min}$ correspond to the minimum delays for the IU to open a window after the closing of another.
- **Items 26, 29, and 32:** $C_{CD,P2}$, $C_{CD,P3}$, and $C_{CD,P4}$ are taken as the delays from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 33:** $\Delta_{CD,P4,C\text{-}END}$ is taken as the delay from the IDU output until the MCU reads the failure signal from the SMU, which includes an assessment of the NDU output.
- **Item 34:** This is the delay to send the mode message to the PE after the MCU issues the command to start the execution of the Schedule Update protocol. The PE_OU takes 1 tick to do this.
- **Item 35:** $S_{SU,P0}|_{min}$ is determined by the minimum delay for the OU to set the output strobe signal after the MCU issues the command.
- **Item 36:** For RPP2, $\Delta_{SU,P1,RCVWND}|_{min}$ is determined by the minimum delay for the IU to open a window after the MCU issues the command.
- **Item 37:** $C_{SU,P1}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 38:** $S_{SU,P1}|_{min}$ is taken as the minimum required delay to ensure the diagnosis is complete before the OU sends the message: 1 tick for the NDU, plus 1 tick for the SMU and MCU, plus 1 tick for the OU.
- **Items 39 and 41:** $C_{SU,P2}$ and $C_{SU,P3}$ are taken as the minimum delays from the closing of the IU reception window until the RVU result is available at the output of the OU message buffer.
- **Items 40 and 42:** $S_{SU,P2}|_{min}$ and $S_{SU,P3}|_{min}$ are taken as the minimum required delays to ensure the diagnosis is complete before the OU sends the message: 1 tick for the NDU, plus 1 tick for the SMU and MCU, plus 1 tick for the OU.
- **Item 43:** $C_{SU,P4}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 44:** $\Delta_{SU,P4,C\text{-}END}$ is taken as the delay from the IDU output until it is safe to issue the next command. Since the command by the MCU to execute the PE Communication protocol is triggered by "LT = T_{PE} - 1", the T_{PE} must set such that T_{PE} - 1 coincides or occurs later than the earliest time at which the schedule assessment is complete. The Invalid_Schedule signal from the Schedule Processor is available 3 ticks after the IDU output is ready. The RVU output is delayed by 1 tick at the Schedule processor. Then, there is a 1 tick delay to update the Schedule_Overload signal. There is 1 tick delay to update the internal Invalid_Schedule. Finally, there is a 1-tick delay to update the output Invalid_Schedule signal. The trigger to issue the MCU command can occur on the same clock edge the Invalid_Schedule is updated. $\Delta_{SU,P4,C\text{-}END} = 5$ gives the earliest time at which the PE Communication command can be asserted.
- **Item 45:** This is the delay to send the schedule assessment (SA) message to the PE after the MCU issues the command to start the execution of the PE Communication protocol. The PE_OU takes 2 tick to do this.
- **Item 46:** $S_{PE,P0,sched}|_{min}$ is determined by the minimum delay for the OU to set the output strobe signal after the MCU issues the command. The OU allows an extra tick to enable the Schedule Processor to retrieve the schedule information for the first message.
- **Item 47:** $\Delta_{PE,P1,RCVWND,sched}|_{min}$ is determined by the minimum delay for the IU to open a window after the MCU issues the command.
- **Items 48 and 50:** $C_{PE,P1,sched}$ and $C_{PE,P2,sched}$ are taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 49:** This is set equal to $S_{PE,P1,acc}|_{min}$.
- **Item 51:** During PE Communication, the OU at each BIU sends an accusations message whenever it is not its turn to send PE messages. When the time comes to send an accusations message at the end of PE Communication, the message is assumed to be ready just one tick after the last PE message is read.
- **Item 52:** $S_{PE,P0,acc}|_{min}$ is determined by the minimum delay for the OU to set the output strobe signal after the accusations message becomes available.
- **Item 53:** $C_{PE,P1,acc}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 54:** When the time comes to send an accusations message at the end of PE Communication, the message is assumed to be ready just one tick after the last PE message is processed.
- **Item 55:** $S_{PE,P1,acc}|_{min}$ is taken as the minimum required delays to ensure the diagnosis, including processing of the suspicions matrix, is complete before the OU sends the message: 2 ticks for the NDU to process the syndromes including the suspicions, plus 1 tick for the SMU and MCU, plus 1 tick for OU. The OU processes

the accusations message with the same timing as the PE messages. For the OU, the only difference in execution is that the message is read from the NDU instead of the RVU buffer.

- **Item 56:** $C_{PE,P2,acc}$ is taken as the delay from the closing of the IU reception window until the IDU outputs the corresponding results.
- **Item 57:** $\Delta_{PE,P2,acc,C\text{-}END}$ is taken as the delay from the IDU output until the MCU reads the failure signal from the SMU, which includes an assessment of the NDU output. The delay at the NDU includes the reduction of the suspicions matrix.
- **Item 58:** All the operations can be performed with a data introduction interval of 1, except for the accusations message during PE Communication, which requires an extra tick at the NDU in order to reduce the suspicions matrix. This special case is the reason why the accusations message is sent only once.
- **Item 59:** The Preliminary Diagnosis observation window is opened one tick after the MCU issues the command.
- **Item 60:** Frame Synchronization begins immediately after the end of the second observation window in Preliminary Diagnosis.
- **Items 61:** The MCU switches from Clique Detection mode to Clique Initialization mode in the clock edge immediately after a failure or no-clique condition is detected.
- **Item 62:** The minimum value to the synchronization reset delay is determined by the time to diagnose the Input Eligible Voters and complete the diagnosis after the Accept(ECHO) in Initial Synchronization and Synchronization Capture. From the Accept(ECHO), it takes 1 tick for the IU to close the reception window, 1 tick for the IDU to output the final input diagnosis, 1 tick for the NDU to update the accusations, and 1 tick for the SMU to read the accusations and update the failure signal. The synchronization-reset signal can be asserted concurrently with this update to the SMU failure signal.

14.4. Preservation Mode Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	Λ_{SU}	Local clock ticks	Integer	≥ 1	≥ 1
2	$\Delta_{SU,STREAM,P0\text{-}P2\|min}$	Local clock ticks	Integer	≥ 1	≥ 1
3	$\Lambda_{PE,sched}$	Local clock ticks	Integer	≥ 1	≥ 1
4	$\Delta_{PE,sched\text{-}acc\|min}$	Local clock ticks	Integer	≥ 1	$= \Lambda_{PE,sched}$
5	T_{CD}	Local clock ticks	Integer	≥ 0	≥ 1
6	T_{SU}	Local clock ticks	Integer	≥ 0	≥ 4
7	T_{PE}	Local clock ticks	Integer	≥ 0	≥ 7
8	T_{SP}	Local clock ticks	Integer	≥ 0	≥ 10
9	Dflt_Num_PE_Msg	Unitless	Integer	≥ 0	≥ 1
10	$\Delta_{SU,SND\text{-}MODE\text{-}READ\text{-}PE\|min}$	Local clock ticks	Integer	≥ -1	≥ -1
11	$\Delta_{PE,SND\text{-}SA\text{-}READ\text{-}PE\|min}$	Local clock ticks	Integer	≥ -1	≥ -1

Notes:

- **Items 1, 2, and 3:** The RPP is designed to handle a data introduction interval of 1 tick.
- **Item 4:** The accusations message at the end of PE Communication is sent with the same data introduction interval as the scheduled PE messages.
- **Item 5:** The RPP is ready to begin the Collective Diagnosis protocol as soon as the local time is reset. Since the MCU uses a "LT = T_{CD} - 1" comparison to trigger the prototocol and LT >= 0, T_{CD} - 1 must be non-negative.
- **Items 6, 7, and 8:** The MCU must wait at least 3 ticks between commands in order to accommodate the timing constraints at the PE_OU.
- **Item 9:** Number of messages per PE for the default schedule.
- **Item 10:** This is the desired delay from sending of the mode message to the PE to reading of the first PE message during Schedule Update. The RPP can read the first message when the MCU issues the command, but the PE_OU needs 1 tick to send the mode message out.

- **Item 11:** This is the desired delay from sending of the mode message to the PE to reading of the first PE message during Schedule Update. The RPP can read the first message 1 tick after the MCU issues the command, but the PE_OU needs 2 ticks to send the mode message out.

14.5. Failure-Recovery Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	$\Delta_{FD}\|_{max}$	Local clock ticks	Integer	≥ 1	≥ 1
2	Δ_{LR}	Local clock ticks	Integer	≥ 0	$= 0$
3	Δ_{STM}	Local clock ticks	Integer	≥ 1	≥ 8

Notes:
- **Item 1:** $\Delta_{FD}|_{max}$ is the delay to enter the local recovery. Although the SMU can assert the failure signal immediately, the MCU requires 1 tick to read the signal and switch to a node reset state.
- **Item 2:** The local recovery operation is part of the Self-Test mode
- **Item 3:** The Self-Test mode consists of a 1-tick node-reset state, a 1-tick idle state, at least 3 ticks for Self_Test commands, and 3 more ticks for the Reset command.

14.6. Clique Detection Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	$\Delta_{PD,OW}$	Local clock ticks	Integer	≥ 1	≥ 1

Notes:
- **Item 1:** Each Preliminary Diagnosis observation window must have a duration of at least one tick.

14.7. Miscellaneous Specifications

Item Number	Variable	Units	Type	Value Range	RPP Constraint
1	Num_SPECIAL_Msg	Unitless	Integer	≥ 0	$= 13$

Notes:
- **Item 1:** The number of SPECIAL messages is known and constant.

14.8. Additional Constraints

The previous tables present the basic value constraints for each variable according to its function in the generic timing model. In addition to this, some variables are constrained by relations to other variables. These additional constraints are presented next.

During Schedule Update and PE Communication, the data introduction rate must not exceed the

capability of the Communication and Computation Modules.

$$\Lambda_{SU} \geq \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (14.1)$$

$$\Delta_{SU,STREAM,P0\text{-}P2}|_{min} \geq \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (14.2)$$

$$\Lambda_{PE,sched} \geq \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (14.3)$$

$$\Delta_{PE,sched\text{-}acc}|_{min} \geq \max(\Lambda_{Comm}, \Lambda_{Comp}) \quad (14.4)$$

The protocol time triggers must not coincide with other scheduled events. This is expressed in terms of constraints on the time gaps between protocols, which indirectly constrain the valid values of T_{CD}, T_{SU}, T_{PE}, and T_{SP}.

$$\Delta_{CD,begin} \geq 0 \quad (14.5)$$

$$\Delta_{SU,begin} \geq 0 \quad (14.6)$$

$$\Delta_{PE,begin} \geq 0 \quad (14.7)$$

$$\Delta_{SP,begin} \geq 0 \quad 14.\ (8)$$

The duration of a resynchronization interval must be at least as large as the maximum duration of the Frame Synchronization protocol. This is expressed in terms of a constraint on p_{min}, which indirectly constrains the value of $T_{SP.}$

$$p_{min} \geq \delta_{FS}|_{max} \quad (14.9)$$

The duration of the Self-Test mode must meet the restart timing constraints.

$$\Delta_{STM} \geq \lceil (1 + \rho_0)\delta_{OUAD}|_{max} \rceil - \Delta_{LR} \quad (14.10)$$

The duration of each observation window during Local Diagnosis Acquisition (a.k.a., Preliminary Diagnosis) must be at least as large as the worst case duration of a resynchronization interval.

$$\Delta_{PD,OW} \geq P \quad (14.11)$$

15. Concluding remarks

ROBUS is the central feature of SPIDER, a scalable family of modular avionics architectures that will support a range of application and reliability demands. ROBUS-2 is a developmental version of ROBUS intended for laboratory experimentation and demonstrations of capabilities. The ROBUS Protocol Processor is a custom-designed hardware component that implements the ROBUS functionality. The RPP presented in this report implements the full functionality of ROBUS-2 as described in [Torres 05]. In addition, the RPP design has the following features.

- Full complement of error checks as described in [Torres 05] for RPP and bus failure detection.
- Node kind and node Id can be specified pre-synthesis or at run-time.
- Parameterized and synthesizable VHDL description with 73 behavioral and 24 structural parameters.
- Maximum throughput of one ROBUS Message per clock tick for the Schedule Update and PE Communication protocols. At full speed, the bus can achieve over 90% processing efficiency for PE messages, measured as the maximum of number of PE message over the total number of clock ticks in a cycle. The overhead is due to the processing of the Collective Diagnosis, Schedule Update, and Synchronization Preservation protocols, in addition to the end-to-end bus latency.
- Preliminary synthesis and place-and-route (PAR) results for a Xilinx Virtex-2 FPGA [XILINX 05] indicate that the RPP can run at up to 50 MHz. For a 3-BIU by 3-RMU system with 17-bit ROBUS Messages (1 tag plus 16 payload), the RPP can process messages at the equivalent throughput rate of 850 Mbps (millions of bits per seconds). For an physical implementation, the actual throughput will be limited by the lower-level communication network and the interface between the communication module and the RPP.

The full set of timing equations presented in this report has been implemented in a MATLAB script. The process of building a particular version of the RPP consists of the following steps: (1) Determine the value for the variables listed in Section 14 of this report; (2) Run the MATLAB scripts to compute the VHDL behavioral and structural parameters; (3) Set the parameter values in the appropriate VHDL package file; and (4) Run synthesis and PAR for the target technology.

Appendix A. Detailed specification for the RPP Input Unit

The following are the requirements for the RPP Input Unit (IU). These requirements are intended to be used to design of a synthesizable VHDL description. Abstract data types of scalar (e.g., Boolean, Natural, enumerated) and composite (e.g., vector) forms are used to capture the requirements. These types are built-in or easily expressed in the VHDL language. Lower-level circuit design matters is assumed to be handled appropriately to ensure compliance with the stated requirements. The target technology are field-programmable gate arrays (FPGA) running at a minimum clock frequency of 5 MHz, which should be easily achievable with current technology. The maximum allowed clock frequency is determined by the implementation. The required parameters, both structural (e.g., number of input channels) and behavioral (e.g., time to wait before asserting a signal), are introduced as needed.

The Input Unit (IU) requirements are presented as follows. First, the high-level tasks allocated to the IU are listed for each ROBUS protocol. Then, the basic IU functions are described conceptually. Next, the IU interface is described in detail. This is followed by a detailed description of the reception modes for nominal, error-free cases. The generation of error syndromes is addressed after that. Then, the required response for each MCU command is presented, including the functions to be used and timing descriptions. The next section lists all the predefined structural and behavioral parameters relevant to the design of the IU. Depending on the actual design of the IU, additional structural parameters may be identified. The last section presents additional synthesis-related requirements and various miscellaneous remarks.

A.1. ROBUS tasks allocated to the Input Unit

Table A.1 lists the tasks allocated to the Input Unit.

A.2. Basic IU functions

The RPP Input Unit serves as an interface between the Communication Module receivers and the computation elements of the RPP. The structural parameter Num_OK specifies the number of receivers connected to the IU. For this version of the RPP, it is assumed that the signals from the receivers are synchronous with respect to the clock signal generated by the physical oscillator, which is denoted by CLK. The input buffering function is implemented using synchronous first-in first-out (FIFO) buffers. If the outputs of the link receiver are not synchronous with respect to the oscillator clock signal, then a signal synchronizer must be used. Figure A.1 illustrates this configuration. Note that the logic before the input buffer is considered part of the Communication Module.

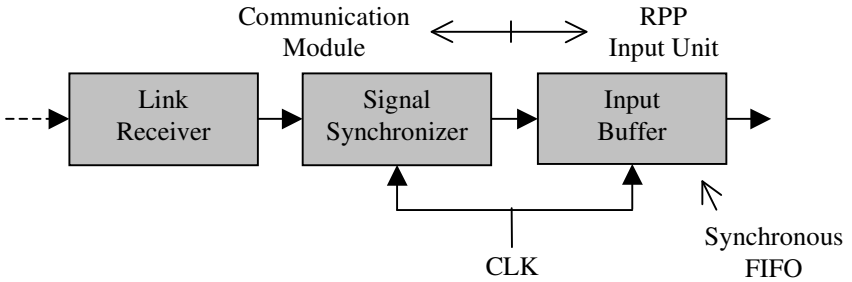


Figure A.1: Reception path for ROBUS nodes with a link receiver with asynchronous outputs

Table A.1: Tasks allocated to the Input Unit

Major Mode	Minor Mode	Protocol	Process and Node Kind	IU Tasks
Clique Preservation	Collective Diagnosis	Collective Diagnosis	P0: BIU & RMU	No applicable tasks
			P1: BIU & RMU	Receive 1 synchronous-communication message from each opposite kind node; The execution of this process does not overlap with process P2.
			P2: BIU & RMU	Receive 1 synchronous-communication message from each opposite kind node; The execution of this process does not overlap with process P3.
			P3: BIU & RMU	Receive 1 synchronous-communication message from each opposite kind node; The execution of this process does not overlap with process P4.
			P4: BIU & RMU	Receive 1 synchronous-communication message from each opposite kind node
	Schedule Update	Schedule Update	P0: BIU	No applicable tasks
			P1: RMU	Receive N synchronous-communication messages from each opposite kind node; Constant DII between messages; The execution of this process does not overlap with process P3.
			P2: BIU	Receive N synchronous-communication messages from each opposite kind node; Constant DII between messages; The execution of this process does not overlap with process P4.
			P3: RMU	Receive N synchronous-communication messages from each opposite kind node; Constant DII between messages
				Load the protocol results and assess the new schedule
			P4: BIU	Receive N synchronous-communication messages from each opposite kind node; Constant DII between messages
				Load the protocol results and assess the new schedule
	PE Communication	PE Broadcast	P0: BIU	No applicable tasks
			P1: RMU	Receive K_{sched} synchronous-communication messages from each opposite kind node; Constant DII between messages
			P2: BIU	Receive K_{sched} synchronous-communication messages from each opposite kind node; Constant DII between messages
		Accusation Exchange	P0: BIU	No applicable tasks
			P1: RMU	Receive 1 synchronous-communication messages from each opposite kind node; Same DII as for PE Broadcast
			P2: BIU	Receive 1 synchronous-communication messages from each opposite kind node; Same DII as for PE Broadcast
	Synchronization Preservation	Synchronization Preservation	P0: BIU	No applicable tasks
			P1: RMU	Receive 1 INIT fixed-delay communication message from each opposite kind node; The execution of this process does not overlap with process P3.

Major Mode	Minor Mode	Protocol	Process and Node Kind	IU Tasks
			P2: BIU	Receive 1 INIT fixed-delay communication message from each opposite kind node; The execution of this process does not overlap with process P4.
			P3: RMU	Receive 1 ECHO fixed-delay communication message from each opposite kind node
			P4: BIU	Receive 1 ECHO fixed-delay communication message from each opposite kind node
Clique Join	Collective Diagnosis	Collective Diagnosis	P0 - P4	Same as for Clique Preservation major mode
	Schedule Update	Schedule Update	P0 - P4	Same as for Clique Preservation major mode
	PE Communication	PE Communication	P0 - P2	Same as for Clique Preservation major mode
	Synchronization Preservation	Synchronization Preservation	P0 - P2	Same as for Clique Preservation major mode
Clique Initialization	Initial Diagnosis and Initial Synchronization	Initial Diagnosis	P0: BIU & RMU	No applicable tasks
			P1: BIU & RMU	Receive 1 synchronous-communication message from each opposite kind node
		Initial Synchronization	P0: BIU	No applicable tasks
			P1: RMU	Receive 1 INIT fixed-delay communication message from each opposite kind node; The execution of this process is concurrent with the execution of process P3.
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol.
			P2: BIU	Receive 1 INIT fixed-delay communication message from each opposite kind node; The execution of this process is concurrent with the execution of process P4.
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol.
			P3: RMU	Receive 1 ECHO fixed-delay communication message from each opposite kind node; The execution of this process is concurrent with the execution of process P1.
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol.
			P4: BIU	Receive 1 ECHO fixed-delay communication message from each opposite kind node; The execution of this process is concurrent with the execution of process P2.
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol

Major Mode	Minor Mode	Protocol	Process and Node Kind	IU Tasks
	Collective Diagnosis	Collective Diagnosis	P0 - P4: BIU & RMU	Same as for Clique Preservation major mode
Clique Detection	Local Diagnosis Acquisition and Synchronization Acquisition	Local Diagnosis Acquisition	P0: BIU & RMU	Receive an indefinite number of messages in asynchronous-monitoring communication mode during two consecutive intervals of equal and predetermined duration.
		Frame Synchronization	P0: BIU & RMU	Execute the Frame Synchronization protocol
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol
		Synchronization Capture	P0: BIU & RMU	Receive 1 ECHO fixed-delay communication message from each opposite kind node
				Receive an indefinite number of messages in asynchronous-monitoring communication mode for the duration of the protocol
	Collective Diagnosis Acquisition	Collective Diagnosis Acquisition	P0 - P4: BIU & RMU	Same as Collective Diagnosis for the Clique Preservation major mode
Self-Test	Self-Test	Self-Test	P0: BIU & RMU	No applicable tasks
	Reset	Reset	P0: BIU & RMU	Reset the Input Unit

The IU handles all timing aspects related to the reception of messages. The IU implements the reception functions to support the point-to-point communication modes: synchronous, fixed-delay, and asynchronous-monitoring. The IU implements the frame-synchronization function, which is essentially a computation-activation function based on the timing of received synchronization messages. The IU also implements the schedule processing function, which is responsible for storing the PE communication schedule, performing an assessment to determine its validity, and converting the schedule into signals suitable for control of the computation pipeline. The IU location at the top of the RPP data-processing path makes it the ideal module to control the timing of the computation pipeline. The timing of the IU control outputs is referenced to the local time, events generated within the RPP, or received events. The IU is also responsible for reading the communication error signals from the link receivers, performing in-line timing-error checks on received messages, and generating error syndromes for diagnostic processing within the RPP.

A.2.1. Synchronous reception

This function is presented mainly from the perspective of a particular point-to-point communication channel. However, the description applies to every IU reception channel.

Synchronous communication is used with the synchronous protocols: Collective Diagnosis, Schedule Update, PE Broadcast, Accusation Exchange, and Initial Diagnosis. For this point-to-point communication mode, the local-time clocks of the source and receiver are assumed to be synchronized within a particular precision bound. There is also a nominal reception delay, denoted by R_{PP}, with a known error bound. Figure A.2 illustrates the relevant timing events for synchronous reception. T_{REF} is a local-time reference selected to coordinate the message transmission and reception actions. A message sent at local time T_{SND} is nominally expected to arrive R_{PP} ticks later at local time $T_{RCV,E}$. With the known synchronization precision bound and the known error in R_{PP}, it is possible to compute the maximum error for $T_{RCV,E}$, denoted $\Delta_{PP,RCV}|_{abs\text{-}max}$, for a message sent by a properly operating source. Thus, the RPP at the receiving node expects to receive the synchronous message at any of the triggering edges of the local-time clock in the interval:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max},\ T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max}] \qquad (A.1)$$

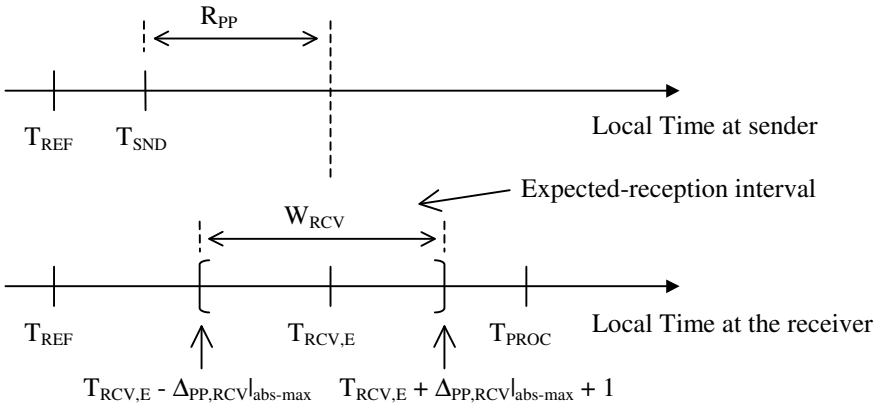


Figure A.2: Timeline for synchronous reception

This interval corresponds to the triggering edges in the local-time interval:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1) \quad (A.2)$$

This expected-reception interval is also called the **reception window**. W_{RCV} denotes the duration of the reception window:

$$W_{RCV} = 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (A.3)$$

The send and receive actions must be coordinated to ensure that the left edge of the reception window does not occur before T_{REF}, i.e., $T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max} \geq T_{REF}$. In addition, the minimum delay required by the source to send the message and the minimum delay required by the receiver to open the reception window, both measured with respect to T_{REF}, must be taken into consideration to determine $T_{RCV,E}$. During the reception window, the receiver expects to receive exactly one message from the source. The received message is held by the input buffer until the scheduled time for processing, denoted by T_{PROC}, which must satisfy the following constraint.

$$T_{PROC} \geq T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (A.4)$$

In order to reduce the PE-to-PE latency of the bus, the processing of synchronous messages will always be scheduled to begin immediately after the closing of its corresponding expected-reception interval. That is:

$$T_{PROC} = T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (A.5)$$

The IU must be able to handle synchronous message streams consisting of a known number of messages with a nominal spacing between them. Consider a generic synchronous-message stream. K_{stream} denotes the total number of messages in the stream. Λ_{stream} denotes the nominal data-introduction interval (DII) (i.e., the inverse of the nominal message rate) for the stream, measured in units of local clock ticks. The spacing between the expected-reception intervals for the messages is a function of $\Delta_{PP,RCV}|_{abs\text{-}max}$ and Λ_{stream}. For a given $\Delta_{PP,RCV}|_{abs\text{-}max}$, as Λ_{stream} is reduced, the intervals get closer until they eventually abut and then overlap. This scenario of overlapping reception windows is handled by defining an overall reception window that applies to the entire synchronous message stream.

$T_{RCV,E,i}$ denotes the expected time of reception for stream message i, with $0 \leq i \leq K_{stream} - 1$. The expected-reception interval for the i-th message is:

$$[T_{RCV,E,0} + i\Lambda_{stream} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E,0} + i\Lambda_{stream} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1) \quad (A.6)$$

The scheduled time to begin processing message i, denoted by $T_{PROC,i}$, is given by:

$$T_{PROC,i} = T_{RCV,E,0} + i\Lambda_{stream} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad (A.7)$$

For $\Lambda_{stream} \leq 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$, the expected-reception intervals are considered to form a single overall window that applies to the entire stream. The reception window for this case is:

$$[T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E,0} + (K_{stream} - 1)\Lambda_{stream} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1) \quad (A.8)$$

The minimum required size for the input buffer (i.e., the minimum number of messages that it must be

able to hold) is a function of $\Delta_{PP,RCV}|_{abs\text{-}max}$ and Λ_{stream}. This minimum buffer size, or depth, is an IU structural constraint.

A.2.2. Fixed-delay reception

This function is presented mainly from the perspective of a particular point-to-point communication path. The description applies to every IU reception channel.

Fixed-delay communication is exclusively used with the INIT and ECHO messages of the synchronization protocols: Synchronization Preservation, Synchronization Capture, and Initial Synchronization. Because the information of main interest is in the time of reception of the messages, the design of the IU must ensure that its outputs properly preserve this information.

Each received synchronization message is characterized by two items of information: the time of reception and the content of the message. Correspondingly, two types of IU outputs are used with fixed-delay reception: timing pulses and message content. A timing pulse is generated a fixed delay after the reception of an expected synchronization message. The message content is stored in the input buffer upon reception and forwarded for processing some time later.

Two sub-modes of fixed-delay reception are defined based on whether the reception window has a predetermined duration or not. **Fixed-delay reception with a predetermined reception window size** is used with the Synchronization Preservation protocol, for which it is possible to define a reception window for each protocol stage. For this reception mode, the generation of the timing pulses and the output of the message content are coordinated with respect to the reception window. The reception window can be used as a reference for timing-error detection similarly to the way done for single-message synchronous reception. **Fixed-delay reception without a predetermined reception window size** is used with the Synchronization Capture and Initial Synchronization protocols. For these protocols, there are no dedicated synchronization-message reception windows, and there is no coordination between the generation of timing pulses and the output of message content.

Figure A.3 illustrates relevant events for fixed-delay reception with a reception window of predetermined size. The reference time for the reception window, denoted by T_{REF}, is based either on the local time or on locally generated synchronization events, depending on the protocol process. The error detection and voting functions for synchronization messages in the Route-and-Vote Unit (RVU) require that the IU generate two pulses for each input channel. The pulses are referred to as the **short pulse** and the **long pulse**. Both pulses are delayed by a predetermined amount with respect to the time of reception of the synchronization message during the reception window. The delay and duration of the pulses are determined by IU behavioral parameters. The pulses are generated only if a message with the expected content (which can be either INIT or ECHO, depending on the protocol processes being executed) is received during the reception window. The pulses are delayed with respect to the first message received with the expected content. At most one pair of short and long pulses is generated for each reception window. For each reception window, the IU performs empty, overload, and overrun error checks. The content of the received message is held in the input buffer until the time for processing, which is at the end of the reception window.

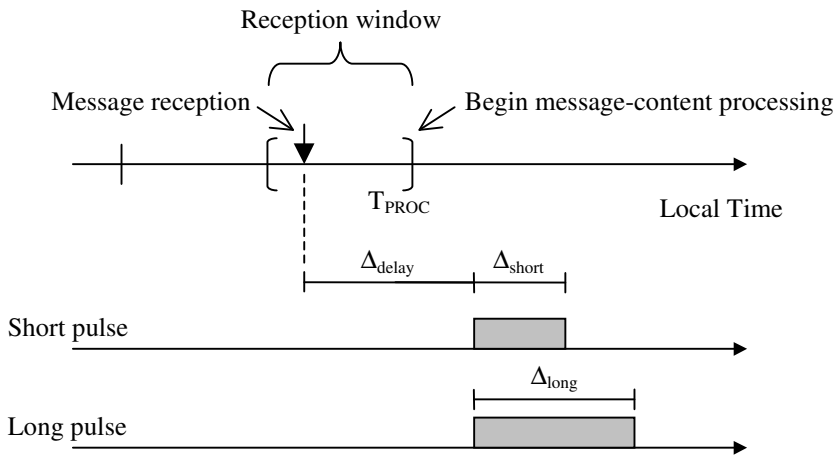


Figure A.3: Timing pulses for a synchronization message with a reception window of predetermined size

For fixed-delay reception without a predetermined reception window size, the short and long pulses are triggered by the reception of an expected synchronization message, and the delay and duration of the pulses are the same as for the case of reception with dedicated reception windows. Because the Initial Synchronization protocol does not preclude the possibility of receiving valid and nearly simultaneous INIT and ECHO messages, the IU must have separate outputs for INIT and ECHO pulses. Once the process of generating the INIT pulses is triggered by the reception of a valid message, any new received INIT message is simply ignored. A similar constraint applies to ECHO messages. For this fixed-delay reception case, the content of received synchronization messages is handled using the asynchronous-monitoring reception.

A.2.3. Asynchronous-monitoring reception

Asynchronous monitoring is performed during Local Diagnosis Acquisition and Synchronization Acquisition in the Clique Detection major mode, and during Initial Synchronization in the Clique Initialization major mode. The reception intervals during which asynchronous monitoring is performed are called **observation windows** (or intervals). During an observation window, each received message is stored for one clock tick in the corresponding IU input buffer and then it is forwarded to the IDU, where the actual input-error monitoring takes place. In the Clique Detection mode, the observation window is open upon transitioning to Local Diagnosis Acquisition, and it remains open until the RVU computes the reference synchronization event during Synchronization Capture. At that point, the IU closes the observation window and the IDU generates its final monitoring results. In the Clique Initialization mode, the observation window is open at the start of the Initial Synchronization protocol and it is closed when the RVU computes the final synchronization reference event.

A.2.4. Frame synchronization

The Frame Synchronization function finds the gap between consecutive executions of the Synchronization Preservation protocol and triggers the activation of the Synchronization Capture protocol. Figure A.4 presents the description of the process.

1. Wait until the IDU readies the set of initial eligible sources.

2. Load the initial eligible sources and start the gap timer.

3. Loop:
  3.1. If an error is detected, remove the corresponding source from the eligible sources.
    3.1.1. Communication checks for each opposite-kind source:
      3.1.1.1. Expecting no link errors
    3.1.2. In-line checks for each opposite-kind source:
      3.1.2.1. At most one ECHO message expected during the execution of this loop.

  3.2. If an ECHO message is received from an eligible source, then restart the gap timer. Else, if gap timer has expired, then assert Done and exit.

Figure A.4: Frame Synchronization process

Major steps 1, 2, and 3 describe separate activities performed on a clock-tick basis. During step 1, the protocol waits until the Input Diagnostics Unit (IDU) determines the initial eligible voters based on the observations during Local Diagnosis Acquisition. During step 2, the protocol loads the eligible sources and starts the gap timer. The "gap timer" is a time-out timer with a predetermined time-out threshold. Step 3 is executed continuously from the second tick on until the process completes and the Done signal is asserted. Steps 3.1 and 3.2 are executed sequentially every tick. In step 3.1, the eligible sources are updated based on the listed error checks. The communication checks are performed by the link receivers of the Communication Module. The definition of these checks is determined by the design of the communication links. The in-line check detects an error when a total of two or more ECHO messages are received from any opposite-kind source during the execution of the loop. In step 3.2, the gap timer is restarted every time an ECHO message is received from an eligible source. Note that if an ECHO message is received from a currently eligible source but an error is simultaneously detected for that source, then the source is declared ineligible and the message is not allowed to trigger a restart of the gap timer. If the gap timer expires and no new ECHO messages are received from eligible sources, then the Done signal is asserted. The expiration of the gap timer triggers the activation of the IU fixed-delay reception mode to handle synchronization messages for the Synchronization Capture protocol. The errors detected during the execution of the Frame Synchronization process become an error-syndrome output by the IU upon completion of the process.

A.2.5. Schedule processing

The communication schedule specifies the number of messages to be transmitted by each PE during the PE Communication mode. The IU schedule processing function consists of three elements: load the results of the schedule update, assess the new schedule, and process the active schedule to control the computation pipeline during the PE Communication mode.

Let N denote the number of BIUs, which is assumed to equal the number of PEs attached to the bus. The Schedule Update protocol is applied N times in order to determine the number of messages to be broadcast by each PE. For each PE, the result of the Schedule Update protocol can be a DATA message with the number of scheduled messages in the payload field, or a PE_ERROR message indicating that the protocol was unable to determine a valid number of messages. The IU loads these results from the RVU

as a series of N consecutive messages sorted by PE identification number in ascending order.

The assessment of the schedule update produces one of three results: invalid, valid, or zero. A schedule is **invalid** if the result of the Schedule Update protocol is PE_ERROR for any PE, or if the total number of scheduled messages exceeds the maximum number of PE messages that the ROBUS can process during the PE Communication mode. A schedule is **valid** if it is not invalid. A **zero** schedule is a special case of a valid schedule in which the number of scheduled messages is zero for every PE. The IU must report to the MCU the result of the schedule assessment.

The activity during the next PE Communication mode depends on the result of the schedule update assessment. If the result is valid, the new schedule is used. If the result is zero, there will be no bus activity. If the result is invalid, a default schedule is used. The default schedule is constant during run-time and is known to all the ROBUS nodes and the PEs. For this version of the ROBUS, the default schedule allocates the same number of transmissions for each PE.

A.2.6. Pipeline control

The IU controls the transfer of data to the computation pipeline. The timing of the IU control outputs is referenced to the local time, events generated within the RPP, or received events. For diagnostic purposes, the IU must also indicate when it has opened a reception or observation window. For synchronous reception, the IU must signal when it is time to begin processing the received messages. The IU must signal when it has enabled fixed-delay reception for synchronization messages. For asynchronous-monitoring, the IDU needs to know when each new message is received.

In addition to timing control, for some protocols the IU must also provide information about the nature of the data. For example, if a message stream is being received, the IU must signal when the end of the stream has been reached. During PE Communication mode, the IU must indicate which PE is the source for each scheduled message, and when the stream is switching from PE Broadcast messages to Accusation Exchange messages.

A.2.7. Error-syndrome generation

The IU generates six types of error syndromes based on the error checks performed. The IU monitors the error signals from the link receivers and generates corresponding syndromes. The IU generates error syndromes based on the checks performed during the execution of the Frame Synchronization protocol. The IU performs empty, overload, and overrun checks and generates corresponding syndromes when receiving messages during windows with predetermined size. Between reception windows, when no messages are expected, the IU monitors the inputs for the arrival of unexpected messages and generates corresponding error syndromes.

A.3. Interface

Figure A.5 shows the signal groups for the Input Unit interface.

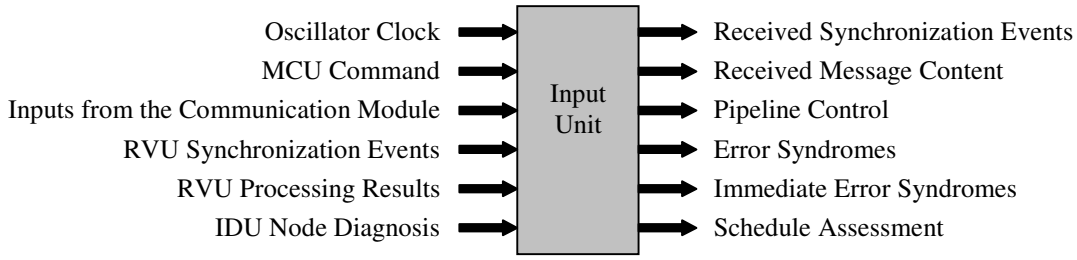


Figure A.5: Input Unit interface: signal groups

Table A.2 describes the interface signals for the Input Unit. The "Direction" column identifies the signal as an input (i.e., generated by another unit) or an output (i.e., generated by the IU). The "Type" column specifies the structure and value set for each signal. The basic scalar types are the following.

- **Bit**: value set {0, 1}
- **Boolean**: value set {TRUE, FALSE}
- **Natural**: value set {non-negative integers in the range indicated in the table }
- **Enumerated**: value set {as listed in the table}

The composite types are the following.

- **Boolean Vector** (BV): a one-dimensional array of Boolean elements addressed by a Natural-type index in the specified range. Type BV(a .. b) corresponds to a Boolean Vector with index in the Natural value range a to b.
- **Boolean Vector of Vectors** (BVV): a two-dimensional array of Boolean elements addressed by row and column Natural-type indexes. Type BVV(a .. b)(c .. d) corresponds to a Boolean Vector of Vectors with row index in the Natural value range a to b and column index in the Natural value range c to d.
- **ROBUS Message** (RM): a two-element record: (Tag, Payload). See XXX for a detailed description of value sets for the Tag and Payload fields.
- **ROBUS Message Vector** (RMV): a one-dimensional array of ROBUS Message elements addressed by a Natural-type index in the specified range. Type RMV(a .. b) corresponds to a ROBUS Message Vector with index in the Natural value range a to b.

The function max(a,b) returns the largest of the Natural-valued variables a and b. In addition, the following symbols are used: N = number of BIUs; Ω = number of nodes of the opposite kind; and L_{LS} = number of syndrome bits from each ROBUS link receiver.

There is no required default value for some IU output signals. This means that an implementation does not need to have a particular default value for those signals. If having a default value is desired, then any arbitrary value within the value set of the signal may be chosen. Arbitrary default values are indicated in Table A.2 as "don't care".

Table A.2: Input-Unit Interface

Signal Group	Signal Name	Direction	Type	Description
Oscillator Clock	CLK	Input	Bit	• Signal generated by the local physical oscillator • This signal marks the passage of time and drives the RPP. • Active (or triggering) transition: $0 \rightarrow 1$ • No assumption shall be made about the duty cycle of this signal.
MCU Command	MCU_Node_Reset	Input	Boolean	• Commands an immediate return to the default state • Note that the assertion of this signal does not imply that the entire RPP, including the MCU, is being reset. • Asserted positive (i.e., Reset on TRUE)
	MCU_Node_Kind	Input	Enumerated: (BIU, RMU)	• Specified RPP node kind • Static value: Latched by the MCU during the time interval between the last RPP reset and the first MCU_Ready, and remains constant until the RPP resets again.
	MCU_Node_Id	Input	Natural: 1 .. max(N,M)	• Specified node identification number • Static value: Latched by the MCU during the time interval between the last RPP reset and the first MCU_Ready, and remains constant until the RPP resets again.
	MCU_Major_Mode	Input	Enumerated: (Self_Test, Clique_Detection, Clique_Initialization, Clique_Join, Clique_Preservation)	• Current RPP major mode • The value of this signal is valid only when MCU_Ready is asserted.
	MCU_Diagnostic_Cycle	Input	Natural: 0 .. 1	• Current RPP diagnostic cycle (0 = first, 1 = second) • The value of this signal is valid only in the Clique Join major mode when MCU_Ready is asserted.
	MCU_Minor_Mode	Input	Enumerated: (Reset, Self_Test, PD_Sync_Capture, ID_Initial_Sync, Collective_Diagnosis, Schedule_Update, PE_Communication, Sync_Preservation)	• Current RPP minor mode • PD_Sync_Capture = from Local Diagnosis Acquisition to Synchronization Capture • ID_Initial_Sync = from Initial Diagnosis to Initial Synchronization • Sync_Preservation = Synchronization Preservation • The value of this signal is valid only when MCU_Ready is asserted.

Signal Group	Signal Name	Direction	Type	Description
	MCU_Schedule_Status	Input	Enumerated: (Valid, Zero, Invalid)	• Assessment result for latest schedule update • The value of this signal is valid only when MCU_Ready is asserted.
	MCU_Output_Enable	Input	Boolean	• Enable control for the RPP broadcast output (i.e., output to opposite-kind nodes) • Asserted positive (i.e., TRUE when the output is enabled) • The value of this signal is valid only when MCU_Ready is asserted.
	MCU_Ready	Input	Boolean	• Indicates when the MCU is issuing a new mode command • Asserted positive (i.e., TRUE when a new command is available)
Inputs from the Communication Module	CM_Message_In	Input	RMV(1 .. Ω)	• Message content output by the link receivers • Each vector element is independent from the others • The value of signal CM_Message_In(i) for input channel i is valid only when CM_Strobe_In(i) is asserted.
	CM_Syndrome_In	Input	BVV(1 .. Ω)(1 .. L_{LS})	• Error syndromes generated by the link receivers • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of signal CM_Syndrome_In (i) for input channel i is valid only when CM_Strobe_In(i) is asserted.
	CM_Strobe_In	Input	BV(1 .. Ω)	• Indicates when a link receiver has a new message or is reporting the detection of an error • When signal CM_Strobe_In(i) is asserted, the IU always reads both CM_Message_In(i) and CM_Syndrome_In(i). • Each vector element is independent from the rest • Asserted positive (i.e., TRUE when a new input is available) • The value of this signal is always valid, except when MCU_Node_Reset is asserted.
RVU Synchronization Events	RVU_Accept_INIT	Input	Boolean	• Output of the Accept(INIT) function • Asserted positive (i.e., TRUE when Accept is triggered)
	RVU_Accept_ECHO	Input	Boolean	• Output of the Accept(ECHO) function • Asserted positive (i.e., TRUE when Accept is triggered)
RVU Processing Results	RVU_Transform_Result	Input	RM	• Result of processing the content of received messages • The value of this signal is valid only when RVU_Ready is asserted.

Signal Group	Signal Name	Direction	Type	Description
	RVU_Ready	Input	Boolean	• Indicates when the RVU has a new content result • Asserted positive (i.e., TRUE when a new result is available)
	RVU_Last_Message	Input	Boolean	• Indicates when a result is the last in a sequence • Asserted positive (i.e., TRUE when a result is the last one) • The value of this signal is valid only when RVU_Ready is asserted.
IDU Node Diagnosis	IDU_Invalid_Input	Input	BV(1 .. Ω)	• Result of error detection performed on the received messages • Each vector element is independent from the others • Asserted positive (i.e., TRUE when the corresponding input channel is invalid or untrustworthy) • The value of this signal is valid only when IDU_Ready is asserted.
	IDU_Ready	Input	Boolean	• Indicates when the IDU has a new result • Asserted positive (i.e., TRUE when a new result is available)
Received Synchronization Events	IU_Init	Output	BV(1 .. Ω)(0 .. 1)	• Timing pulses for received INIT synchronization messages • Each row is independent from the others • For input channel i, IU_Init(i)(0) = short pulse and IU_Init(i)(1) = long pulse • Asserted positive (i.e., TRUE during a pulse) • The value of this signal is valid only during the execution of the synchronization protocols. • Default value: Don't Care
	IU_Echo	Output	BV(1 .. Ω)(0 .. 1)	• Timing pulses for received ECHO synchronization messages • Each row is independent from the others • For input channel i, IU_Init(i)(0) = short pulse and IU_Init(i)(1) = long pulse • Asserted positive (i.e., TRUE during a pulse) • The value of this signal is valid only during the execution of the synchronization protocols. • Default value: Don't Care

Signal Group	Signal Name	Direction	Type	Description
Received Message Content	IU_Message_Out	Output	RM(1 .. Ω)	• Received message content • Each vector element is independent from the others • Depending on the reception mode, the value of this signal is valid only when the corresponding field of IU_Strobe_Out is asserted or when IU_Ready is asserted. • Default value: Don't Care
Pipeline Control	IU_Strobe_Out	Output	BV(1 .. Ω)	• Indicates when a new message has been received • Each vector element is independent from the others • Asserted positive (i.e., TRUE when a new message is received) • IU_Strobe_Out(i) is always asserted one tick after IU_Strobe_In(i) is asserted, except when MCU_Node_Reset is aserted or a Minor_Mode = Reset command is issued. • Default value: FALSE
	IU_Ready	Output	Boolean	• Indicates when the IU has completed a current activity or reached an important intermediate point • Asserted positive (i.e., TRUE when a new result is available) • The value of this signal is always valid, except when MCU_Node_Reset is aserted. • Default value: FALSE
	IU_Receiving	Output	Boolean	• Indicates when the IU is expecting to receive messages • Asserted positive (i.e., TRUE when receiving) • The value of this signal is always valid, except when MCU_Node_Reset is aserted. • Default value: FALSE
	IU_PE_or_Acc	Output	Enumerated: (PE, ACC)	• Indicates the nature of a message during PE Communication mode • The value of this signal is valid only when IU_Ready is asserted during PE Communication. • Default value: Don't Care
	IU_Source_Id	Output	Natural: 1 .. N	• Node ID of the current PE message during PE Communication mode • The value of this signal is valid only when IU_Ready is asserted during PE Communication. • Default value: Don't Care

Signal Group	Signal Name	Direction	Type	Description
	IU_Last_Message	Output	Boolean	• Indicates that the last message in a stream has been reached during Schedule Update and PE Communication mode • Asserted positive (i.e., TRUE when the last message has been reached) • The value of this signal is valid only when IU_Ready is asserted during Schedule Update and PE Communication. • Default value: Don't Care
Error Syndromes	IU_Unexpected_Message	Output	BV(1 .. Ω)	• Indicates that an unexpected message has been received • Each vector element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of signal IU_Unexpected_Message (i) is valid only when IU_Receiving is not asserted and IU_Strobe_Out(i) is asserted. • Default value: Don't Care
	IU_Link_Error	Output	BVV(1 .. Ω)(1 .. L_{LS})	• Indicates that a link error has been reported by the Communication Module receivers • The output of this syndrome is coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of signal IU_Link_Error(i) is valid only when IU_Receiving is asserted and, depending on the reception mode, when IU_Strobe_Out(i) or when IU_Ready is asserted. • Default value: Don't Care
	IU_Empty_Buffer	Output	BV(1 .. Ω)	• Indicates that the input buffer is unexpectedly empty • The output of this syndrome is coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of this signal is valid only when IU_Receiving is asserted and IU_Ready is asserted. • Default value: Don't Care

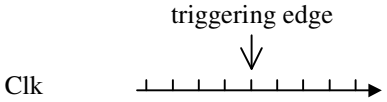
Signal Group	Signal Name	Direction	Type	Description
	IU_Input_Overrun	Output	BV(1 .. Ω)	• Indicates that messages remain in an input buffer after the closing of a reception window • The output of this syndrome is coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of this signal is valid only when IU_Receiving is asserted and IU_Ready is asserted. • Default value: Don't Care
	IU_Buffer_Overload	Output	BV(1 .. Ω)	• Indicates that the input buffer is holding more messages than expected (or equivalently, that the rate of message reception is larger than expected) • The output of this syndrome is coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of this signal is valid only when IU_Receiving is asserted and IU_Ready is asserted. • Default value: Don't Care
	IU_Frame_Sync_Error	Output	BV(1 .. Ω)	• Indicates that an error was detected during Frame Synchronization • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of this signal is valid only when IU_Ready is asserted to signal the completion of the Frame Synchronization protocol. • Default value: Don't Care
Immediate Error Syndromes	IU_Imm_Link_Error	Output	BVV(1 .. Ω)(1 .. L_{LS})	• Indicates that a link error has been reported by the Communication Module receivers • The output of this syndrome is not coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of signal IU_Imm_Link_Error(i) is valid when IU_Strobe_Out(i) is asserted. • Default value: Don't Care

Signal Group	Signal Name	Direction	Type	Description
	IU_Imm_Buffer_Overload	Output	BV(1 .. Ω)	• Indicates that a link error has been reported by the Communication Module receivers • The output of this syndrome is not coordinated with the output of message content. • Each element is independent from the others • Asserted positive (i.e., TRUE when an error is detected) • The value of signal IU_Imm_Link_Error(i) is valid when IU_Strobe_Out(i) is asserted. • Default value: Don't Care
Schedule Assessment	IU_Zero_Schedule	Output	Boolean	• Indicates when the number of scheduled messages for the lastest schedule update is zero. • Asserted positive (i.e., TRUE when there are no scheduled messages) • The value of this signal is arbitrary during Schedule Update until the assessment of the new schedule is complete. At all other times, the value is constant and valid. • Default value: Don't Care
	IU_Invalid_Schedule	Output	Boolean	• Indicates when the lastest schedule update is invalid. • Asserted positive (i.e., TRUE when the downloaded schedule is invalid) • The value of this signal is arbitrary during Schedule Update until the assessment of the new schedule is complete. At all other times, the value is constant and valid. • Default value: TRUE

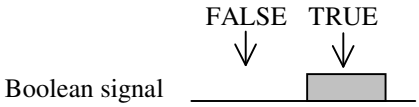
A.4. Message reception

This section provides a detailed description of the reception modes for nominal, error-free cases. The following basic conventions are adopted in the timing diagrams. Other diagrammatic conventions are explained as needed.

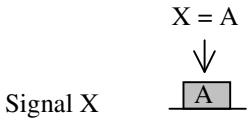
- The Clk signal is represented by tick marks on a time axis indicating the timing of the triggering edges (i.e., $0 \rightarrow 1$ transitions).



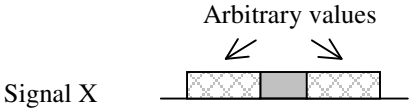
- For scalar Boolean variables, a value of TRUE is represented by an unlabeled gray box and a value of FALSE by an empty section.



- For scalar and vector variables, a labeled gray box is used to indicate that the signal has a certain value, which may or may not be known a priori but is not arbitrary.



- A crosshatched section in the waveform of a particular signal indicates that the signal may have an arbitrary value (i.e., a "don't care") during that section of the waveform.



A.4.1. Synchronous reception

Synchronous reception is characterized by the use of time-referenced expected-reception intervals. The Input Unit handles this reception mode using a uniform processing model whereby messages are always part of a stream of known size, which can be greater than or equal to one. Each synchronous message has a corresponding expected-reception interval of predetermined duration. The size of the message reception interval for the Initial Diagnosis protocol is specified by parameter ID_Wnd_Sz. For the other synchronous protocols, the size of the reception interval for each expected message is given by parameter Syncns_Wnd_Sz.

Figure A.6 shows an example of the timing for synchronous reception of a one-message stream. For illustration purposes only, the duration of an expected-reception interval for a synchronous message is denoted by W_{RCV}, which is measured in units of local-clock ticks. $W_{RCV} = 5$ is an arbitrary value chosen

for illustrative purposes and is not meant to represent a typical value. A single generic input channel i is shown. The IU expects to receive a message from each opposite-kind source during the reception interval. Only messages that arrive during the reception interval are accepted. IU_Receiving is set TRUE to indicate that the IU is expecting to receive a message. Note that IU_Receiving is delayed by one clock tick with respect to the expected-reception interval. The assertion of IU_Ready indicates that the end of the reception interval has been reached and the result is ready for processing. The IU_Strobe_Out(i) is always asserted one tick after a message is received.

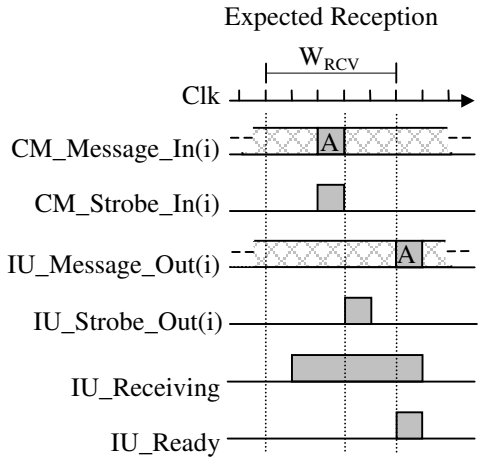


Figure A.6: Synchronous reception of a one-message stream with $W_{RCV} = 5$

There are two reception cases for multi-message streams. In the simplest case, the messages have expected-reception windows that do not overlap or coincide end-to-end. Figure A.7 shows an example of reception for a three-message stream (i.e., K = 3) with $W_{RCV} = 5$ and DII = 7. As can be seen, there is a separate reception window for each message and the reception activity for each is the same as for the case of reception of a one-message stream. Note that IU_Strobe_Out(i) is asserted one tick after a message is received, and IU_Ready indicates when the result is ready for processing.

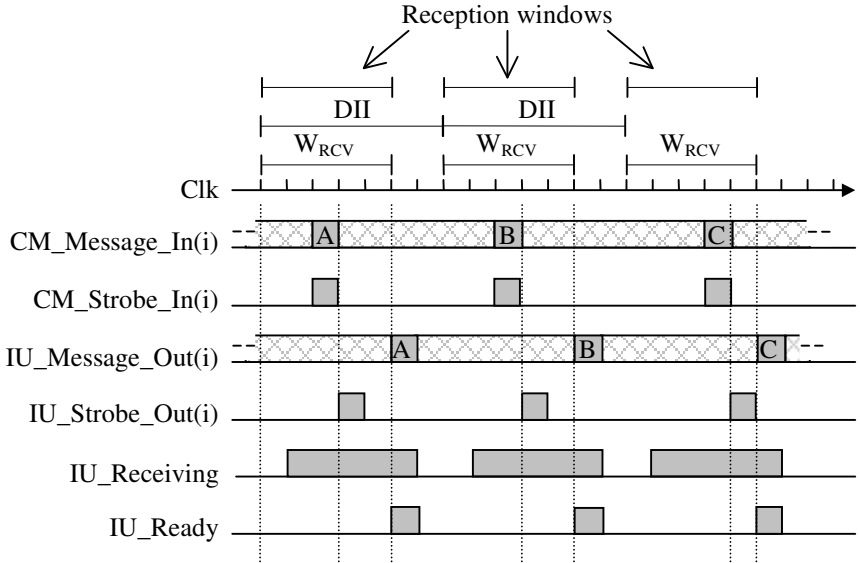


Figure A.7: Synchronous reception of a multi-message stream with separate reception windows ($K = 3$, $W_{RCV} = 5$, DII = 7)

The second case of multi-message stream reception involves expected-reception intervals that overlap or coincide at the ends. For this case, the messages are received with a single continuous reception window. All the messages are expected to arrive within this window with an average DII approximately equal to the nominal value. Figure A.8 shows an example of reception for a stream of three messages (i.e., K = 3) with W_{RCV} = 5 and DII = 3. The IU outputs the messages at the times corresponding to the end of their respective expected-reception intervals, similarly to the way it is done for the other synchronous reception cases.

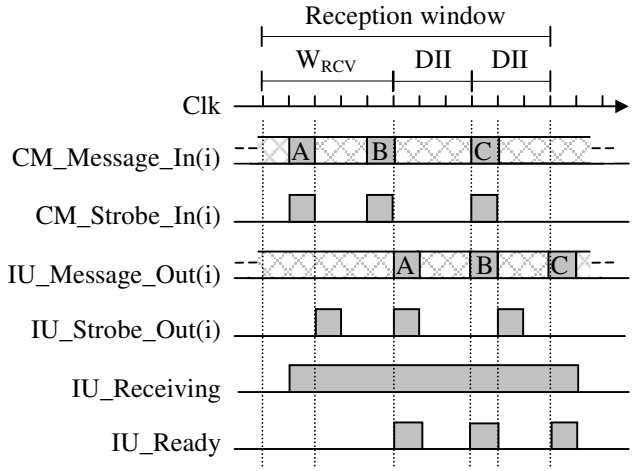


Figure A.8: Synchronous reception of a multi-message stream with a continuous reception window (K = 3, W_{RCV} = 5, DII = 3)

A.4.2. Asynchronous-monitoring reception

In this reception mode, the IU receives messages without knowing in advance exactly when the observation window will end. Signal IU_Receiving remains high during the observation window. The IU closes the observation window upon detection of a certain expected event generated locally within the RPP. The IU is required to output every received message with exactly one tick delay from the time of reception. Figure A.9 shows a reception example for an arbitrary input pattern.

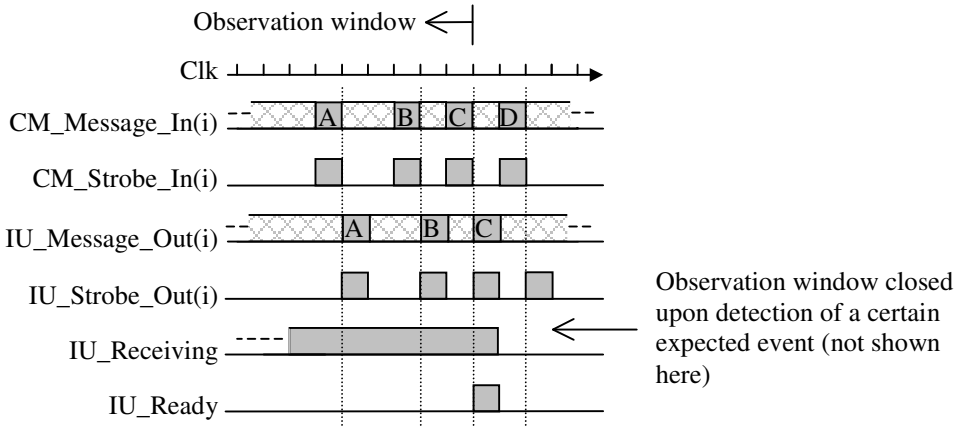


Figure A.9: Asynchronous-monitoring reception

A.4.3. Fixed-delay reception

Fixed-delay reception is used only with the INIT and ECHO messages used in the synchronization protocols. For this reception mode, the IU outputs the timing of reception separately from the content of the received messages. There are two cases of fixed-delay reception, depending on whether a predetermined expected-reception interval is defined or not.

For fixed-delay reception with an expected-reception interval of predetermined size, the location of the intervals is referenced to the local time or to local events generated within the RPP during the execution of the Synchronization Preservation protocol. During a particular reception interval, only a single synchronization message with a specific content is expected from each input channel. The size of the reception intervals varies depending on the synchronization protocol process being executed. The size of the reception windows is specified by the behavioral parameters SP_INIT_Wnd_Sz(MCU_Node_Kind) for INIT messages and SP_ECHO_Wnd_Sz(MCU_Node_Kind) for ECHO messages. Note that these parameters are a function of the node kind, which is specified by signal MCU_Node_Kind. Figure A.10 shows an example of fixed-delay reception with an expected-reception interval of predetermined size. $W_{RCV} = 5$ is an arbitrary value chosen for illustrative purposes only. The shown delay and the duration of the short and long pulses are also arbitrary. For this particular example, only one INIT message is expected during the reception window. The IU_Init(i) pulses are only triggered by the reception of the first INIT message. Any additional messages received from opposite-kind source i during the reception window do not trigger additional pulses. If no INIT messages are received during the window, then the pulses are not generated. Notice that the content of the message is output after the end of the reception interval, similarly to the way it is done for synchronous reception.

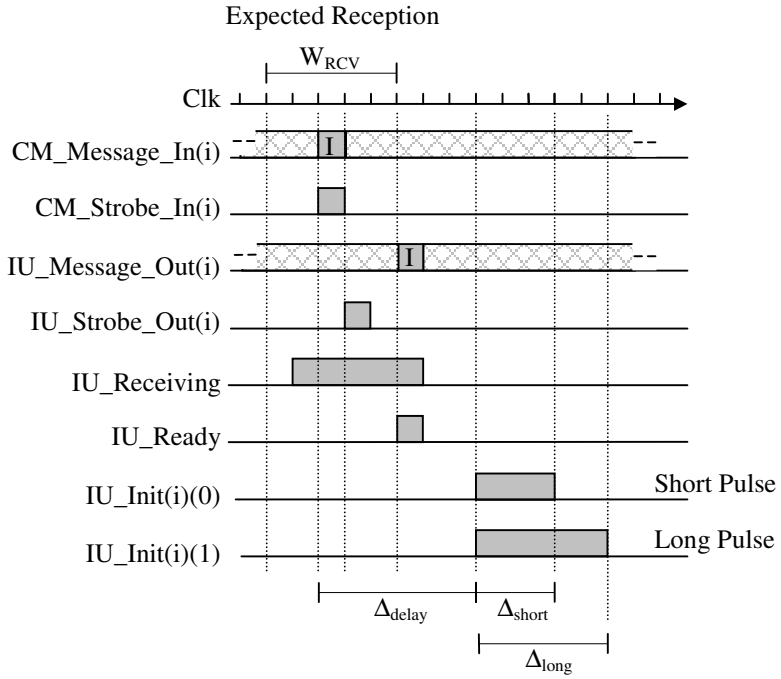


Figure A.10: Fixed-delay reception with a predetermined reception window size ($W_{RCV} = 5$)

When fixed-delay reception without a predetermined expected-reception interval is used, the IU is receives messages without knowing in advance exactly when the reception or observation window will end. At a specific point in time, the IU enables a pulse-generation function to start monitoring the inputs

for INIT or ECHO messages. The short and long pulses are triggered by the reception of synchronization messages. The content of the messages is handled according to the asynchronous-monitoring reception rules. Figure A.11 shows an example for an ECHO message. The pulse-generation function generates one set of pulses when the expected message is received (in this example, an ECHO message). After that, no additional pulses are generated. If the expected message is not received, then the pulses are not generated.

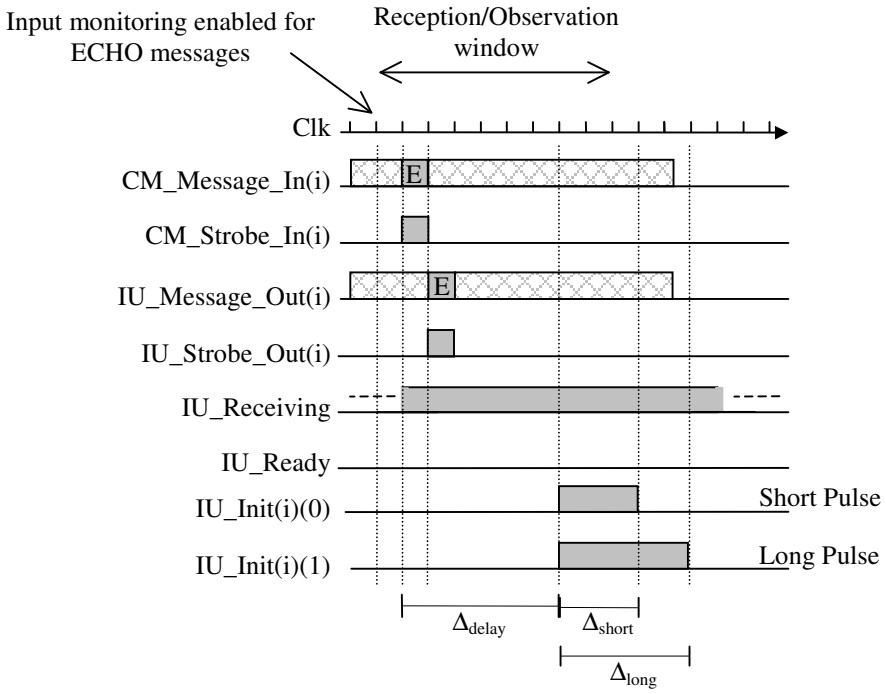


Figure A.11: Fixed-delay reception without a predetermined reception window size

The delay and duration of the pulses are independent of the protocol being executed. For INIT messages, the pulses are generated based on the IU behavioral parameters INIT_Pls_Dly(MCU_Node_Kind) and INIT_Skw(MCU_Node_Kind), both of which are functions of the local node kind.

- $\Delta_{delay}|_{INIT}$ = INIT_Pls_Dly(MCU_Node_Kind) (A.9)
- $\Delta_{short}|_{INIT}$ = INIT_Skw(MCU_Node_Kind) + 1 (A.10)
- $\Delta_{long}|_{INIT}$ = 2*INIT_Skw(MCU_Node_Kind) + 1 (A.11)

The pulses for ECHO messages are generated based on the IU parameters ECHO_Pls_Dly(MCU_Node_Kind) and ECHO_Skw(MCU_Node_Kind).

- $\Delta_{delay}|_{ECHO}$ = ECHO_Pls_Dly(MCU_Node_Kind) (A.12)
- $\Delta_{short}|_{ECHO}$ = ECHO_Skw(MCU_Node_Kind) + 1 (A.13)
- $\Delta_{long}|_{ECHO}$ = 2*ECHO_Skw(MCU_Node_Kind) + 1 (A.14)

A.5. Error-syndrome generation

This section describes the generation of the error syndromes.

A.5.1. IU_Unexpected_Message

An unexpected-message error syndrome is asserted for a particular input channel whenever a message is received while the IU is not expecting to receive messages (i.e., when there is no active reception or observation window). This type of error is signaled at the outputs of the IU one tick after the message is received. Figure A.12 shows various reception cases relative to reception or observation windows. The value of syndrome IU_Unexpected_Message(i) is valid when IU_Receiving is not asserted and IU_Strobe_Out(i) is asserted. For these conditions, IU_Unexpected_Message(i) must not be asserted unless an error has been detected.

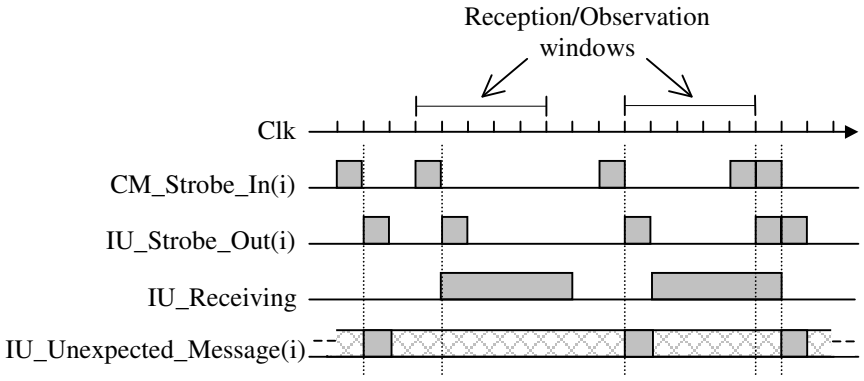


Figure A.12: Generation of the unexpected-message syndrome

A.5.2. IU_Empty_Buffer

For a particular input channel, an empty-buffer error occurs when it is time for the IU to output a message and the input buffer is empty. This type of error can only happen during synchronous reception or fixed-delay reception with a predetermined reception window size. When receiving in asynchronous-monitoring mode, this error cannot happen because messages are always output one tick after they are received. Two cases need to be considered based on the number of expected messages during a reception window: single-message reception and multi-message reception. The value of syndrome IU_Empty_Buffer(i) is valid when IU_Receiving is asserted and either IU_Strobe_Out(i) is asserted for asynchronous-monitoring reception or IU_Ready is asserted for the other reception modes.

Figure A.13 shows an example for the case of a reception window with only one expected message. Here, IU_Empty_Buffer(i) is asserted at the end of the reception window because no messages were received from channel i during the window. Note that IU_Empty_Buffer(i) is asserted at the time to output the expected message as indicated by the IU_Ready signal. There is no required specific default value for the elements of the IU_Message_Out signal. This example applies to the reception of single-message synchronous streams, multi-message synchronous streams with separate reception windows, and fixed-delay reception with reception windows of predetermined sizes.

Figure A.14 shows two examples for the case of reception windows with multiple expected messages.

These examples apply only to synchronous reception. Two streams are expected, each one with three messages. The expected-reception interval for each individual message has a duration of five ticks (i.e., $W_{RCV} = 5$), and the streams have a nominal DII of three ticks. For the first reception window, the second message from input channel i is not received, and thus the IU outputs an arbitrary value and asserts the empty-buffer syndrome. A late reception of the second message would have also resulted in the same error, although the final output message would have been B instead of C. In addition, if C had been received before the end of the second expected-reception interval, it would have been output at the time for message B as the second received message. For the second window, since the third message is missing, the IU asserts the empty-buffer syndrome with the last IU_Ready.

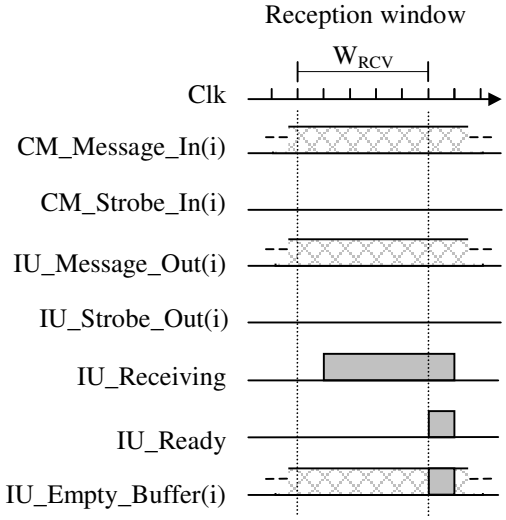


Figure A.13: Generation of the empty-buffer syndrome for a single-message reception window

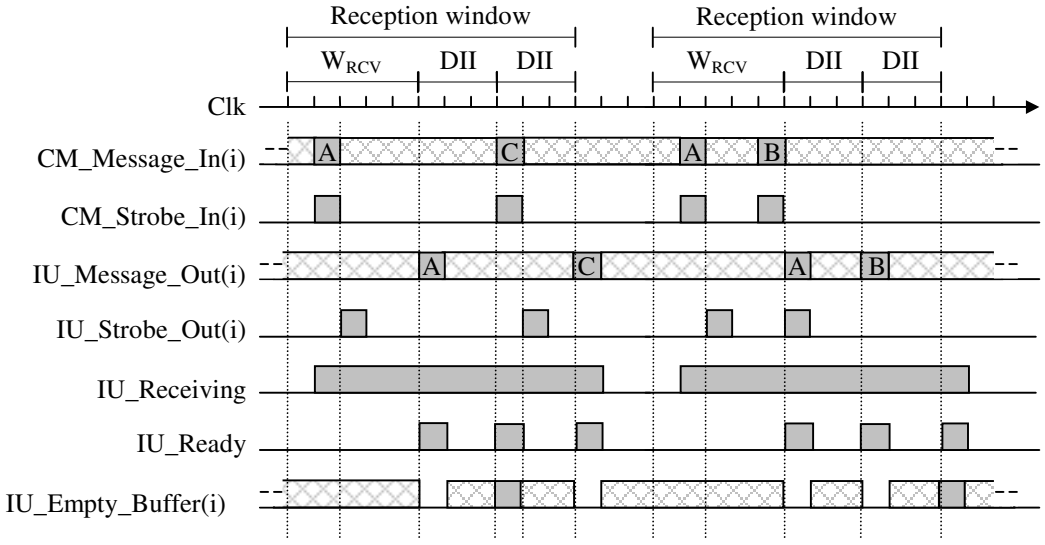


Figure A.14: Generation of the empty-buffer syndrome for multi-message reception windows

A.5.3. IU_Input_Overrun

This error check is meant to detect when more messages than expected are received during a reception window. For a particular input channel, an input-overrun error occurs if messages remain in the input buffer at the end of a reception window when it is time to process the last expected message. This error is reported only at the end of a window. Similarly to the empty-buffer error, this type of error can only happen during synchronous reception or fixed-delay reception with a predetermined reception window size. When receiving in asynchronous-monitoring mode, this error cannot happen because there is no predetermined number of expected messages and all the received messages are output one tick after they are received. The value of syndrome IU_Input_Overrun(i) is valid when IU_Receiving is asserted and either IU_Strobe_Out(i) is asserted for asynchronous-monitoring reception or IU_Ready is asserted for the other reception modes.

Figure A.15 shows an example for the case of a reception window with only one expected message. IU_Input_Overrun(i) is asserted at the end of the reception window because more than one message was received from channel i during the window. This example applies to the reception of a single-message synchronous stream, a multi-message synchronous stream with separate reception windows, and fixed-delay reception with a reception window of predetermined size.

Figure A.16 shows a example for the case of a reception window with multiple expected messages. A three-message stream is expected. IU_Input_Overrun(i) is asserted at the end of the reception window because some received messages will remain in the input buffer.

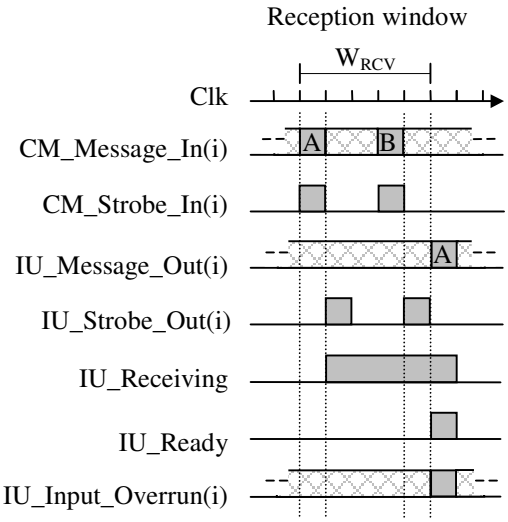


Figure A.15: Generation of the input-overrun syndrome for a single-message reception window

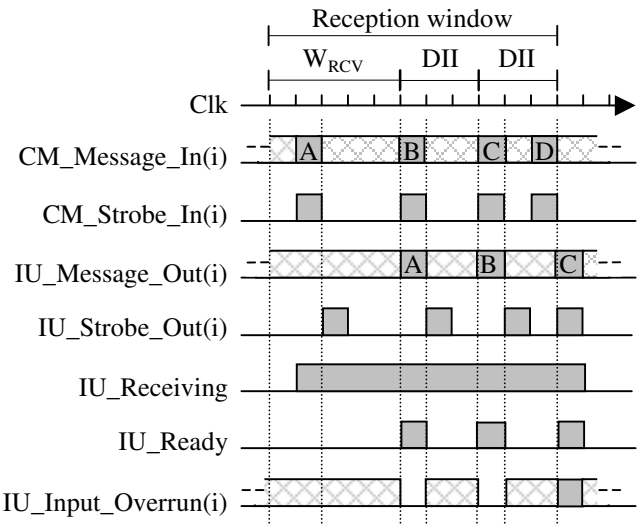


Figure A.16: Generation of the input-overrun syndrome for multi-message reception windows

A.5.4. IU_Link_Error and IU_Imm_Link_Error

The link-error syndromes are based on communication checks performed by the Communication Module (CM) receivers. Each receiver generates a vector of error syndromes the size of which is specified by the structural parameter Num_Link_Synd. Each link receiver asserts its corresponding CM_Strobe_In signal whenever it has a new message. In addition, CM_Strobe_In may also be asserted when a link error is detected. The IU is designed based on a simple transaction model by which the assertion of a CM_Strobe_In input indicates that a new message and error syndrome set is available from the corresponding link receiver. The IU is required to read the syndromes and appropriately forward them to its outputs.

IU signals IU_Link_Error and IU_Imm_Link_Error are generated based on the error syndromes from the link receivers. Whenever a new transaction is received on input channel i, IU_Strobe_Out(i) is asserted one tick later and the syndromes are output on signal IU_Imm_Link_Error(i). The timing of IU_Imm_Link_Error is independent of reception or observation windows. On the other hand, the generation of signal IU_Link_Error is tied to these windows. For asynchronous-monitoring reception, the messages and syndromes for a particular channel are presented at the IU_Message_Out(i) and IU_Link_Error(i) outputs one tick after reception. For synchronous reception and for fixed-delay reception with a predetermined reception window size, the syndromes are output on the IU_Link_Error(i) signal at the same time the corresponding message content is output. Signal IU_Link_Error(i) is not asserted for receptions outside of reception or observation windows. The value of syndrome IU_Link_Error(i) is considered valid when IU_Receiving is asserted and either IU_Strobe_Out(i) is asserted for asynchronous-monitoring reception or IU_Ready is asserted for the other reception modes

Figure A.17 shows an example of link-error syndrome generation for asynchronous-monitoring reception. The syndrome bits for input channel i are represented by characters X, Y, and Z. Note that IU_Link_Error(i) is arbitrary for messages received outside of the observation window.

Figure A.18 shows an example for synchronous reception of a three-message stream. Note that the timing to output the link syndromes on the IU_Link_Error(i) signal matches the timing to output CM_Message_In(i) on signal IU_Message_out(i). For this particular example in which there is an input-

overrun error, the content and syndromes for the extra message do not reach the IU_Message_Out(i) and IU_Link_Error(i) outputs, respectively.

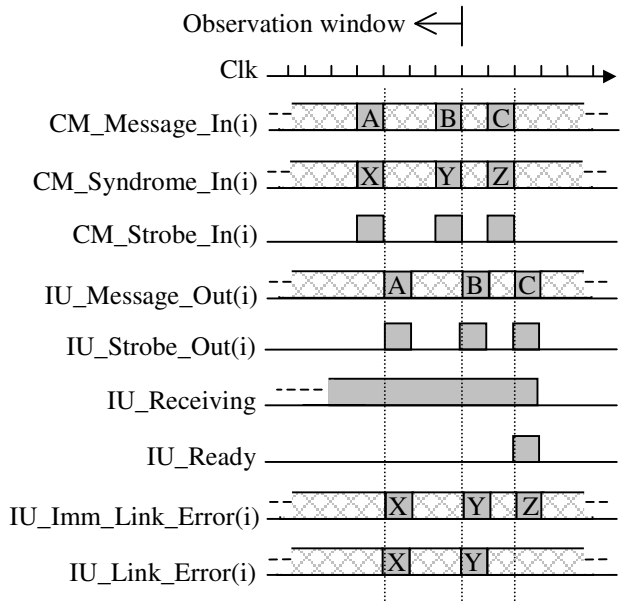


Figure A.17: Generation of the link-error syndromes for an observation window with asynchronous-monitoring reception

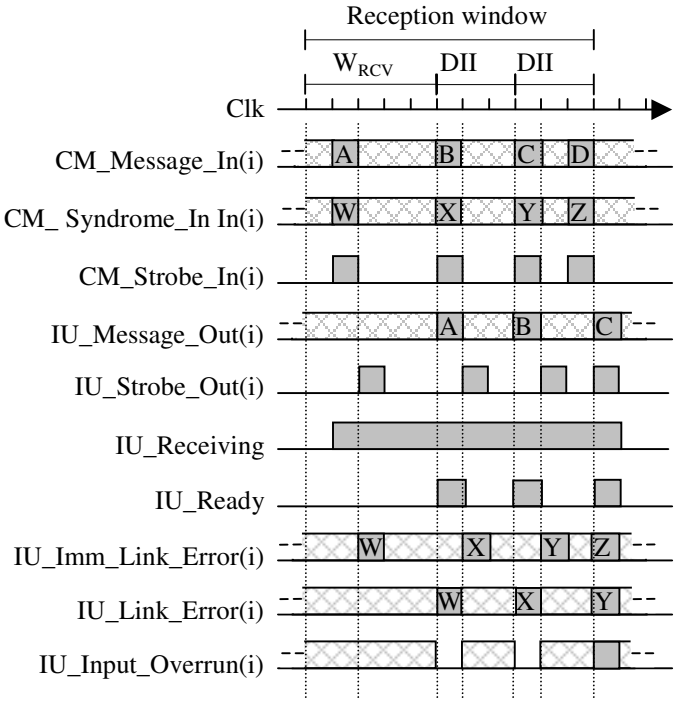


Figure A.18: Generation of the link-error syndromes for a synchronous reception windows

A.5.5. IU_Buffer_Overload and IU_Imm_Buffer_Overload

The purpose of the buffer-overload error check is to detect when the reception rate for a multi-message synchronous stream with a continuous reception window is higher than expected. It is possible to compute a maximum expected input-buffer load (i.e., the number of stored messages) for the reception of a stream using known bounds on the drift rates of the physical clocks, the nominal DII of the stream, and the delay in processing the first message of the stream at the receiving node. A buffer-overload error occurs on an input channel when a message is received such that the buffer load will exceed the expected maximum. The error is assigned to the message that triggers the overload, not the next message to be output by the IU.

A buffer-overload error cannot occur during an observation window with asynchronous-monitoring reception because, irrespective of the actual message reception rate, the buffer never holds one than one message, since all the received messages are output one tick after they are received. For synchronous reception and fixed-delay reception with a predetermined window size, signal IU_Imm_Buffer_Overload(i) for input channel i is asserted one tick after an overload-triggering message is received. The value of signal IU_Imm_Buffer_Overload(i) is valid whenever IU_Receiving and IU_Strobe_Out(i) are asserted, and it is arbitrary for all other conditions. The timing to output the syndrome on signal IU_Buffer_Overload(i) output matches the timing to output the received message content on signal IU_Message_Out(i). Thus, if an overload is detected, IU_Buffer_Overload(i) is asserted when the offending received message reaches the output. For reception windows with a single expected message, an overload occurs when more than one message is received, but IU_Buffer_Overload(i) is not asserted because the extra messages do not reach the output. This last type of error is also an input-overrun error, and thus the IU_Input_Overrun(i) signal will be asserted at the time to output the expected message. The value of signal IU_Input_Overrun(i) is valid when IU_receiving and IU_Ready are asserted, and it is arbitrary for all other conditions.

The structural parameter Input_FIFO_Depth specifies the minimum size for each of the input buffers. The actual buffer size is determined by the IU design and implementation. During a reception window, the messages stored in an input buffer must not be overwritten. If an input buffer becomes full during a reception window, the required IU response is to not load (i.e., to dump) any additional messages received on the corresponding input channel until there is space available in the buffer. The only exception is when there is a simultaneous reading of the buffer; in that case, the required response is to load the received message. Dumped messages are not reflected in the IU_Message_Out, IU_Link_Error and IU_Buffer_Overload signals. The reception of the dumped messages is always reflected in the corresponding fields of the signals IU_Strobe_Out, IU_Imm_Link_Error and IU_Imm_Buffer_Overload since there is no memory storage restriction associated with these signals. Note that a buffer overflow condition can occur for every reception mode except when asynchronous-monitoring is active.

Figure A.19 shows as example of the generation of input-overload syndromes for the reception of two synchronous-message streams. Three messages are expect during each window, and the maximum expected load for the input buffer is two for both cases. For the first window, three messages are received within the expected-reception interval for the first message. Message C triggers the buffer overload. This is reported one tick later on signal IU_Imm_Buffer_Overload(i) and at the output time of the third message on signal IU_Buffer_Overload(i). For the second window, four messages are received and message D triggers the overload. This is reported one tick later on signal IU_Imm_Buffer_Overload(i), but it is never reported on signal IU_Buffer_Overload(i) because the D message does not reach the output. Nevertheless, IU_Buffer_Overrun(i) is asserted in this case.

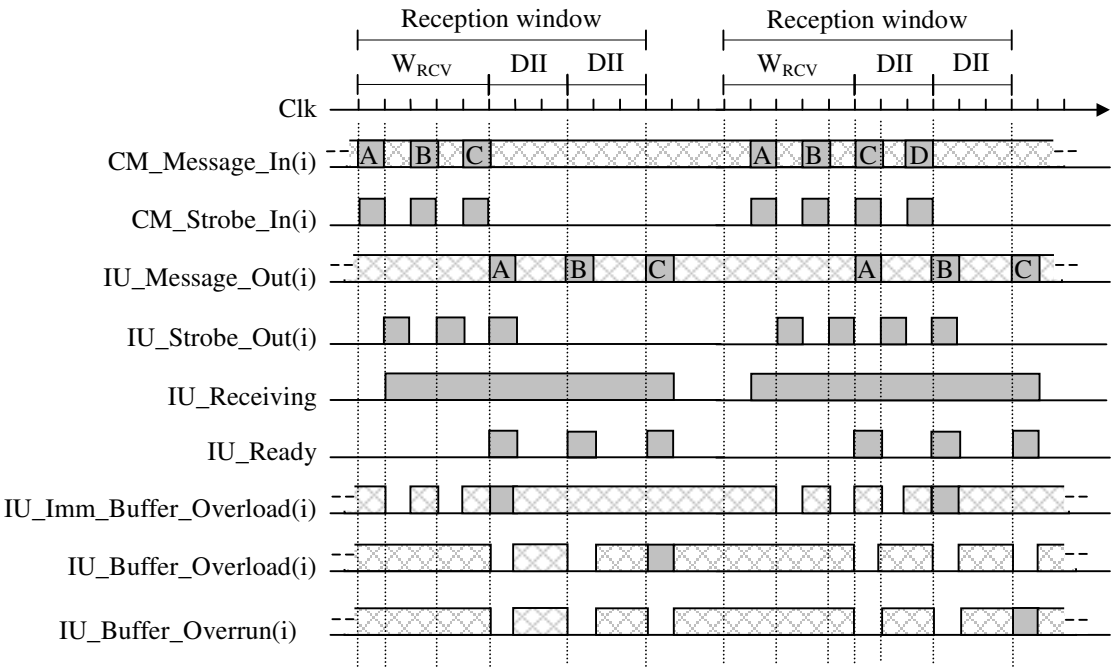


Figure A.19: Generation of the input-overload syndromes for synchronous reception

A.5.6. IU_Frame_Sync_Error

The errors detected during the execution of the Frame Synchronization process are reported when the process completes. Figure A.20 illustrates this. The set of syndrome bits is represented by character Z. For this example, the gap timer is set to expire after six ticks. The IU_Frame_Sync_Error syndromes are valid only when the Frame Synchronization process is complete, which is signaled by the third assertion of IU_Ready in the PD_Sync_Cap minor mode. The value of IU_Frame_Sync_Error is arbitrary at all other times.

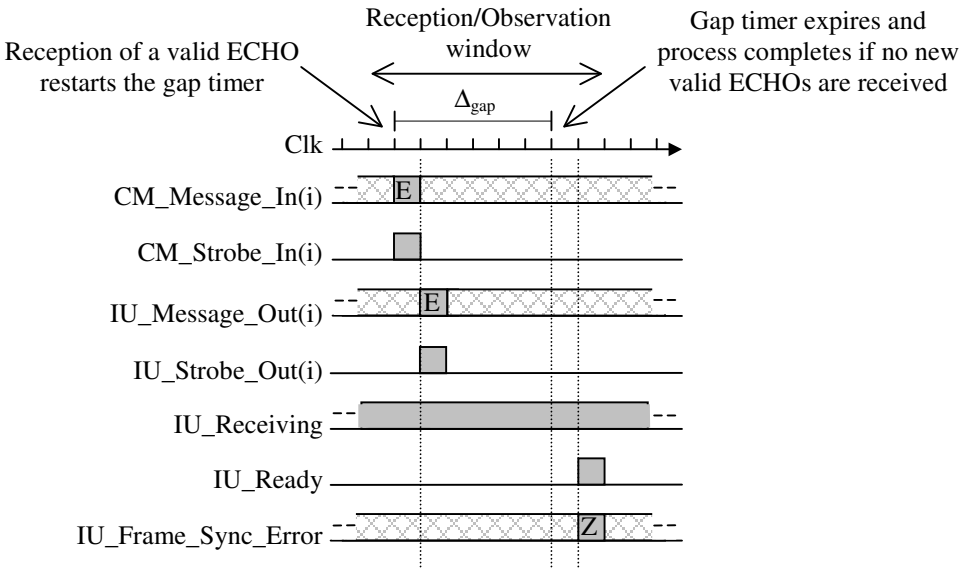


Figure A.20: Generation of the Frame Synchronization syndromes

A.6. Response to MCU commands

The IU reacts to nine different MCU commands. A node-reset command, indicated by asserting the MCU_Node_Reset signal, requires the IU to immediately return to a default state independently of the current activity or the value of other MCU command signals. This command may be issued at any time and the IU must respond to it. The other commands relevant to the behavior of the IU are indicated by the command signal MCU_Minor_Mode. The MCU will issue these commands only when the IU is idle and ready to receive a command. The IU design is required to execute each MCU command independently, and no assumption should be made about the sequence in which the commands are issued.

Note that the IU is required to begin every reception or observation window with empty input buffers.

A.6.1. Node_Reset

The IU is required to return to its default state at most two ticks after the MCU_Node_Reset signal is asserted. The IU response includes setting its output signals to their default values as stated in Table A.2 and invalidating the current PE communication schedule. In its default state, the IU is idle and waiting for a new command from the MCU.

Figure A.21 shows an example of a node-reset command. Note that IU_Strobe_Out must be returned to its default value and held there while MCU_Node_Reset is asserted, and it is allowed to operate normally afterwards.

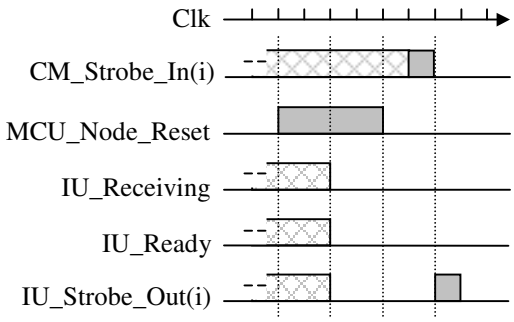


Figure A.21: Response to MCU command: MCU_Node_Reset

A.6.2. Minor_Mode = Reset

The response of the IU to a Reset minor mode command is similar to the response to a node-reset command, except that the IU must be in its default state one tick after the command is issued. Figure A.22 shows an example of the required response. The "Reset" value of signal MCU_Minor_Mode is represented here by the letter X.

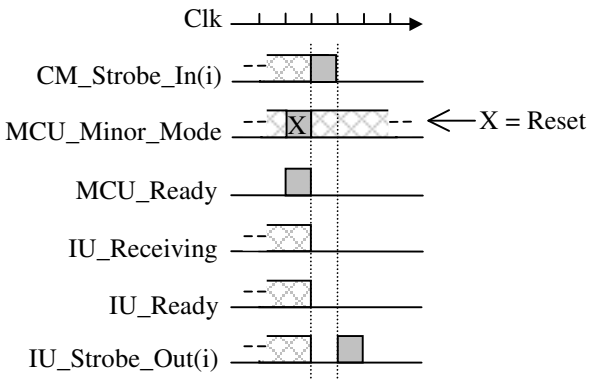


Figure A.22: Response to MCU command: Minor_Mode = Reset

A.6.3. Minor_Mode = Collective_Diagnosis

For Collective Diagnosis the IU expects to receive messages during four separate synchronous-reception windows. For each reception window, only one message is expected from each opposite-kind node. The size of each reception window is given by parameter Syncns_Wnd_Sz. The delay from one tick after MCU_Ready is asserted to the beginning of the first window is given by CD_Wnd1_Dly. The time interval between windows is given by CD_Wnd2_Dly. Figure A.23 illustrates the IU response for CD_Wnd1_Dly = 3, CD_Wnd2_Dly = 2, and Syncns_Wnd_Sz = 4. The letter "X" on the MCU_Minor_Mode waveform represents the "Collective_Diagnosis" value. The label RxWnd stands for "Reception Window". During the execution of this minor mode, the values of signals IU_PE_or_Acc, IU_Source_Id, and IU_Last_Message are arbitrary.

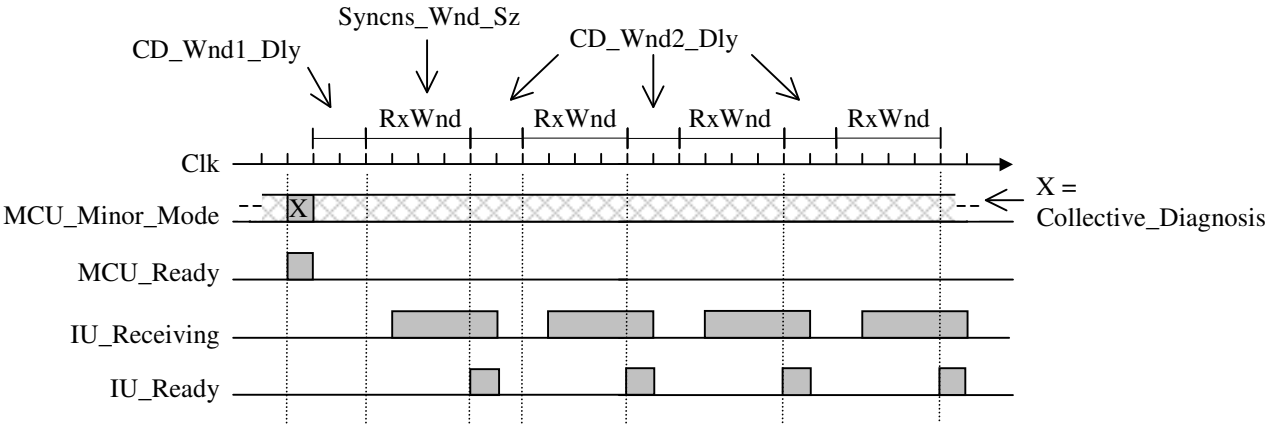


Figure A.23: Response to MCU command: Minor_Mode = Collective_Diagnosis

A.6.4. Minor_Mode = Schedule_Update

For Schedule Update the IU expects to receive two synchronous-message streams from each opposite-kind node. The number of messages in each stream is equal to the number of BIUs, which is denoted by N and given by structural parameter Num_BIU. Each stream is received with separate reception windows for each message or with a continuous reception window for the full stream. Parameter SU_Wnd2_Dly specifies the duration of the time interval between expected-reception intervals for each message of a stream. If SU_Wnd2_Dly = 0, then each stream is received with a continuous reception window. Parameter SU_P1_Wnd1_Dly(MCU_Node_Kind), which is a function of the local node kind, specifies the delay to the expected-reception interval for the first message of the first stream. Parameter SU_P2_Wnd1_Dly(MCU_Node_Kind) specifies the duration of the time interval between the last window for the first stream and the first window for the second stream. SU_DII specifies the DII for both streams. Syncns_Wnd_Sz specifies the duration of the expected-reception interval for each message. Parameter SU_Max_Buff_Cnt specifies the maximum number of messages that an input buffer is expected to hold during a reception window, separate or continuous, under normal operation.

In addition to receiving the message streams, during the execution of this command the IU receives the new schedule from the Route-and-Vote Unit (RVU) in the form of N consecutive messages corresponding to the processing results for the messages of the second stream. No particular time delay should be assumed between IU_Ready and RVU_Ready. After receiving the RVU results, the IU must assess the new schedule and update the schedule assessment outputs. Parameter Max_Num_PE_Msg specifies the maximum number of PE messages that can be scheduled. If the new schedule is invalid, a default schedule is used in which each PE is allocated Dflt_Num_PE_Msg messages.

Figures A.24 illustrates the IU response for a system with 3 BIUs and the following reception parameter values: SU_P1_Wnd1_Dly(MCU_Node_Kind) = 1, SU_P2_Wnd1_Dly(MCU_Node_Kind) = 2, Syncns_Wnd_Sz = 3, SU_Wnd2_Dly = 1, and SU_DII = 4. The letter “X” on the MCU_Minor_Mode signal waveform represents the “Schedule_Update” value of that signal. Note that IU_Last_Message is active in the execution. However, the values of signals IU_PE_or_Acc and IU_Source_Id are arbitrary. Since separate reception windows are used for each message, SU_Max_Buff_Cnt = 1 in each case. The new schedule is represented by values D, E, and F on signal RVU_Transform_Result. The worst-case delay to generate the schedule assessment outputs measured with respect to the last assertion of RVU_Ready is determined by the IU design. The values of IU_Zero_Schedule and IU_Invalid_Schedule

must be held constant until the Schedule_Update command is issued again.

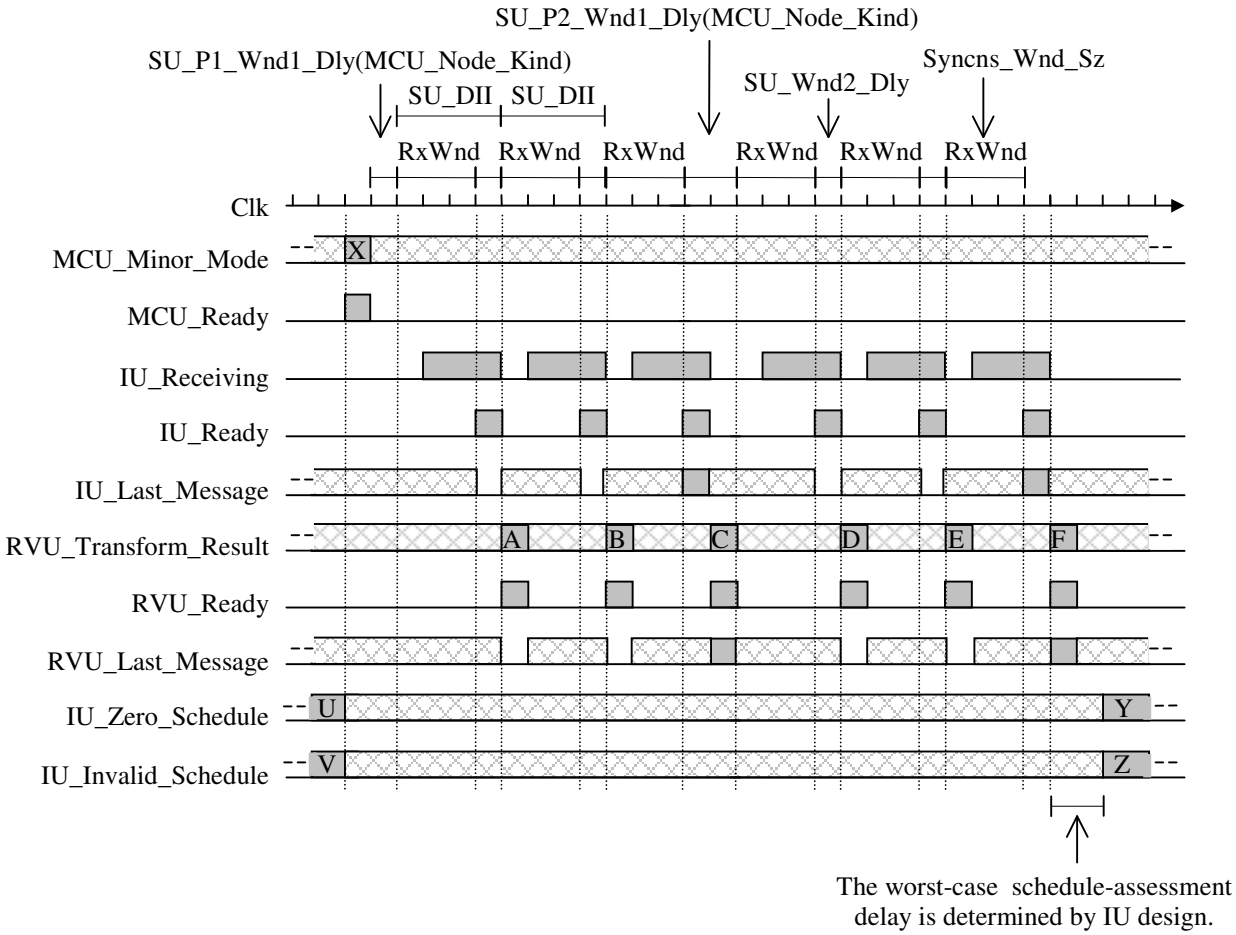


Figure A.24: Response to MCU command: Minor_Mode = Schedule_Update (separate reception windows)

Figures A.25 illustrates the response for continuous reception windows with SU_Wnd2_Dly = 0, and SU_DII = 2. The other parameters have the same values as in Figure A.24.

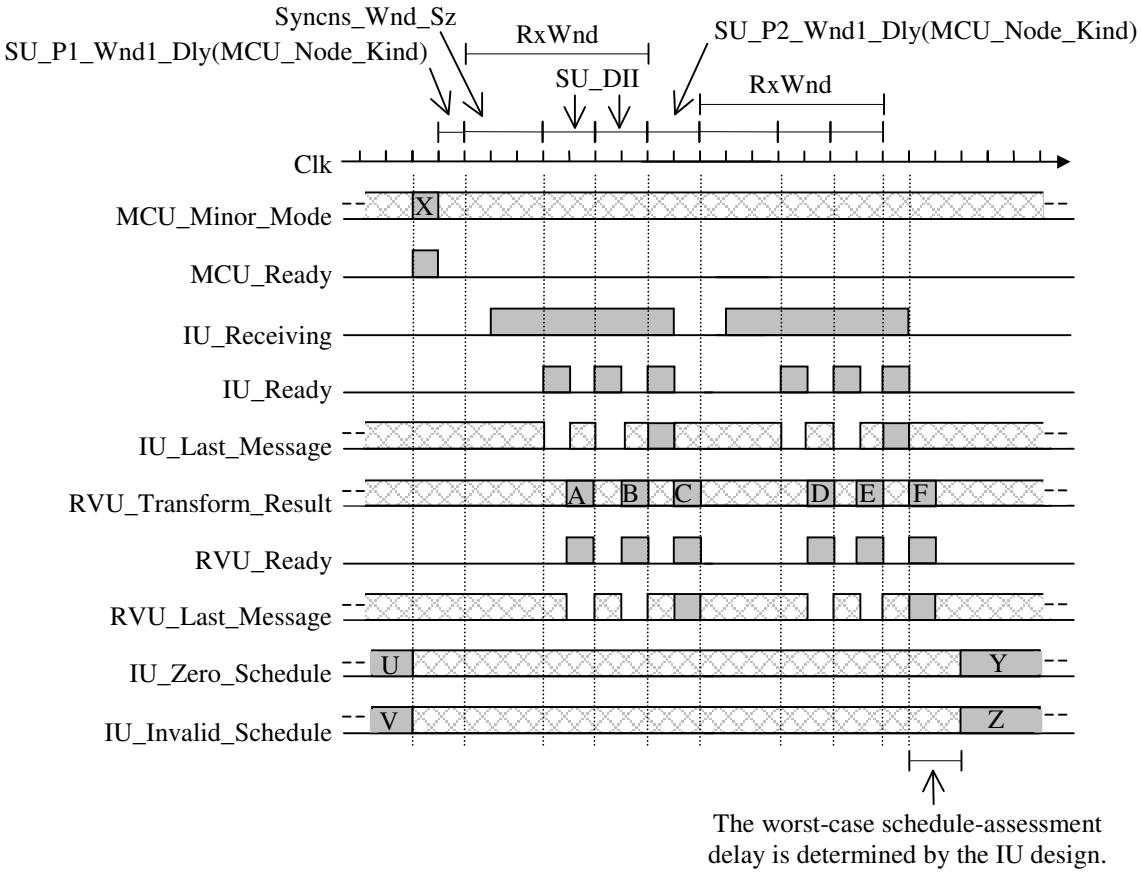


Figure A.25: Response to MCU command: Minor_Mode = Schedule_Update (continuous reception windows)

A.6.5. Minor_Mode = PE_Communication

Irrespective of the schedule assessment results, the MCU will always issue a command to transition to the PE Communication mode. The response of the IU depends on the value of signal MCU_Schedule_Status. If MCU_Schedule_Status = Zero, the IU ignores the command and waits for the next one. If MCU_Schedule_Status = Valid, the latest loaded schedule is executed. If MCU_Schedule_Status = Invalid, the default schedule is executed.

For MCU_Schedule_Status ≠ Zero, the IU expects to receive from each opposite-kind source a synchronous-message stream composed of the scheduled PE messages followed by a message for the Accusation Exchange protocol. Parameter PE_Wnd2_Dly specifies the duration of the time interval between the expected-reception intervals for each message. If PE_Wnd2_Dly = 0, then the stream is received with a continuous reception window. Parameter PE_Wnd1_Dly(MCU_Node_Kind) specifies the delay from one tick after the command is issued until the time at which the expected-reception interval for the first message begins. Syncns_Wnd_Sz specifies the duration of the expected-reception interval for each message. PE_DII specifies the data-introduction interval, which applies to all the messages in the stream, including the Accusation Exchange protocol message at the trailing end. Parameter PE_Max_Buff_Cnt specifies the maximum expected number of messages stored in each input buffer during the reception windows.

Figure A.26 illustrates the IU response for a system with 4 BIUs and schedule {(1, 2),(3, 1),(4, 2)},

where for each (a,b) pair "a" specifies the PE source identification number and "b" specifies the number of scheduled messages. The parameter values are as follows: PE_Wnd1_Dly(MCU_Node_Kind) = 1, PE_Wnd2_Dly = 1, Syncns_Wnd_Sz = 3, and PE_DII = 4. Since the messages are received with separate reception windows, PE_Max_Buff_Cnt = 1 for each window.

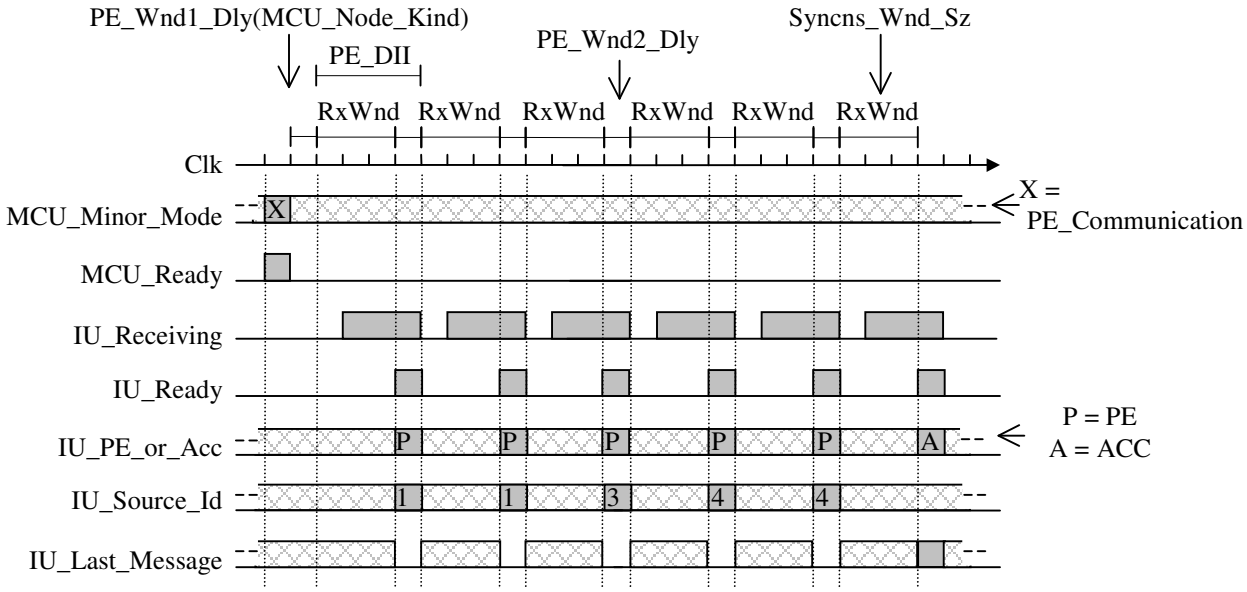


Figure A.26: Response to MCU command: Minor_Mode = PE_Communication (separate reception windows)

Figure A.27 illustrates the IU response with a continuous reception window. For this case, PE_Wnd2_Dly = 0, PE_DII = 2, and PE_Max_Buff_Cnt has a value larger than 1.

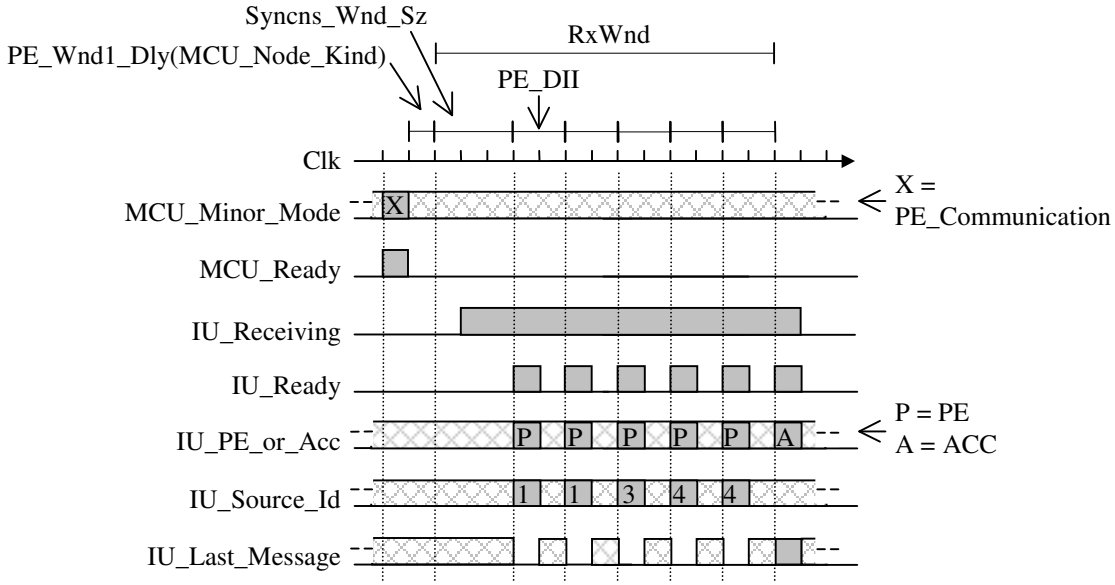


Figure A.27: Response to MCU command: Minor_Mode = PE_Communication (continuous reception window)

A.6.6. Minor_Mode = Sync_Preservation

For this mode the IU expects to receive INIT and ECHO synchronization messages from each opposite-kind source in two separate reception windows. Fixed-delay reception is used in each case. Each window is characterized by delay and size parameters. Figure A.28 illustrates the IU response. For the INIT window, parameter SP_INIT_Wnd_Dly(MCU_Node_Kind) specifies the delay to the expected-reception interval with respect to the time one tick after the MCU command is issued, and parameter SP_INIT_Wnd_Sz(MCU_Node_Kind) specifies the duration of the reception interval. For the ECHO window, parameter SP_ECHO_Wnd_Dly(MCU_Node_Kind) specifies the delay to the expected-reception interval with respect to the time one tick after the RVU_Accept_INIT is asserted, and parameter SP_ECHO_Wnd_Sz(MCU_Node_Kind) specifies the duration of the reception interval. Note that the RVU_Accept_INIT is monitored only after the IU_Ready signal is asserted.

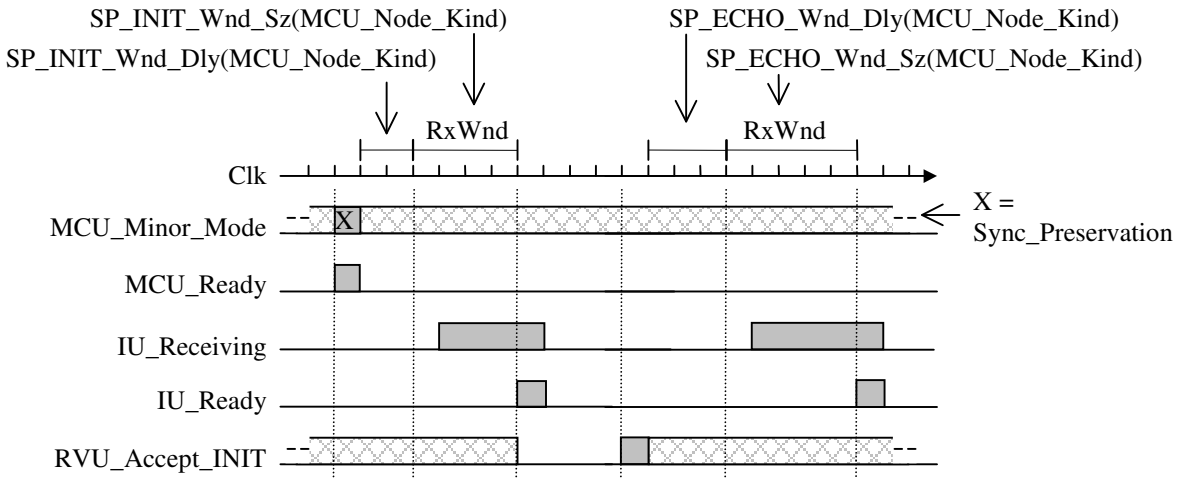


Figure A.28: Response to MCU command: Minor_Mode = Sync_Preservation

A.6.7. Minor_Mode = Self_Test

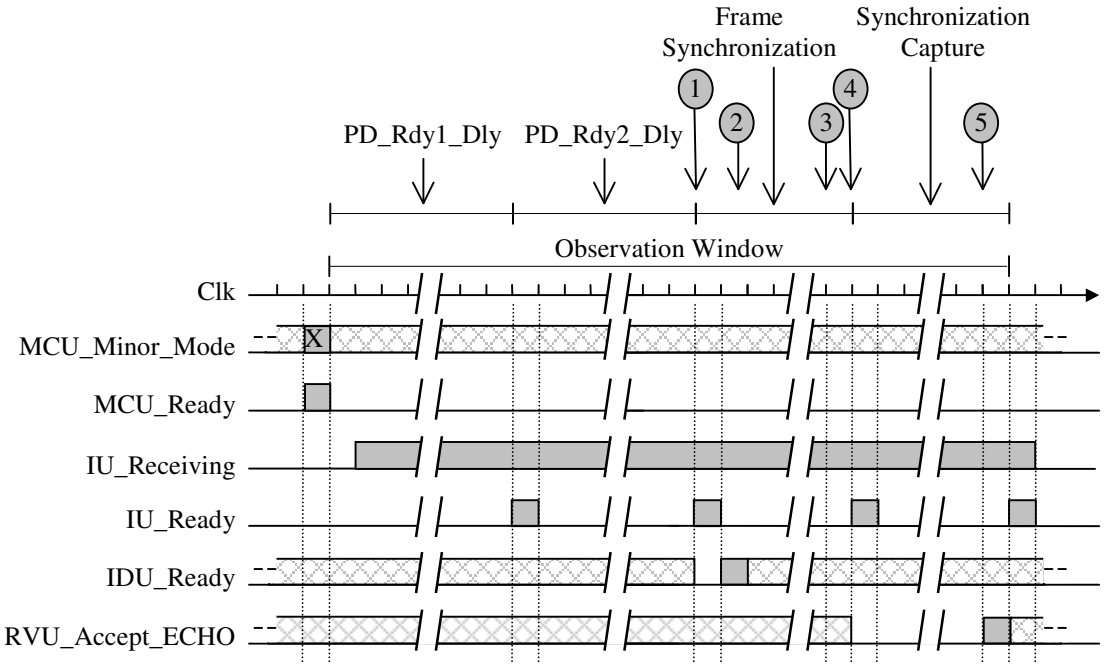
This version of the RPP does not require the implementation of a self-test. When the MCU issues this command, the IU simply ignores it and waits for the next command.

A.6.8. Minor_Mode = PD_Sync_Capture

This command corresponds to the Local Diagnosis Acquisition and Synchronization Acquisition modes executed consecutively. Figure A.29 illustrates the IU response for this command. The IU supports two activity threads in this mode. The first activity thread consists of four steps: measuring two consecutive time intervals, executing the Frame Synchronization protocol, and then enabling fixed-delay reception for ECHO messages. This first thread defines a continuous observation window composed of four intervals corresponding to the four steps. The second activity thread runs in parallel with the first one and consists of performing asynchronous-monitoring reception for each input channel during the full duration of the observation window defined by the first thread.

Starting one tick after the MCU command is issued, the IU measures two consecutive observation

intervals, the first one of duration PD_Rdy1_Dly and the second one of duration PD_Rdy2_Dly. IU_Ready signals the completion of the intervals. The Frame Synchronization (FS) process is enabled upon completion of the second interval. When the FS process is complete, the Synchronization Capture interval begins, during which fixed-delay reception for ECHO messages is performed. The observation window is closed when the RVU_Accept_ECHO signal is asserted.



1. FS process enabled
2. When IDU_Ready is asserted, FS process loads the initial eligible sources and enables the gap timer
3. Gap timer expires and FS process completes if no new valid ECHOs are received
4. Fixed-delay reception for ECHO messages enabled
5. RVU_Accept_ECHO triggers end of observation window

Figure A.29: Response to MCU command: Minor_Mode = PD_Sync_Capture

A.6.9. Minor_Mode = ID_Initial_Sync

This command triggers the execution of Initial Diagnosis followed by Initial Synchronization. Figure A.30 illustrates the IU response. For Initial Diagnosis, the IU performs synchronous reception with one expected message from each opposite-kind source. Parameter ID_Wnd_Dly specifies the delay from one tick after the MCU command is issued until the beginning of the expected-reception interval. Parameter ID_Wnd_Sz specifies the duration of the reception window. After the closing of the Initial Diagnosis window, the IU waits for IS_Wnd_Dly to enable fixed-delay reception for the INIT and ECHO messages of the Initial Synchronization protocol. Simultaneously, the IU performs asynchronous-monitoring reception. The observation window closes when RVU_Accept_ECHO is asserted.

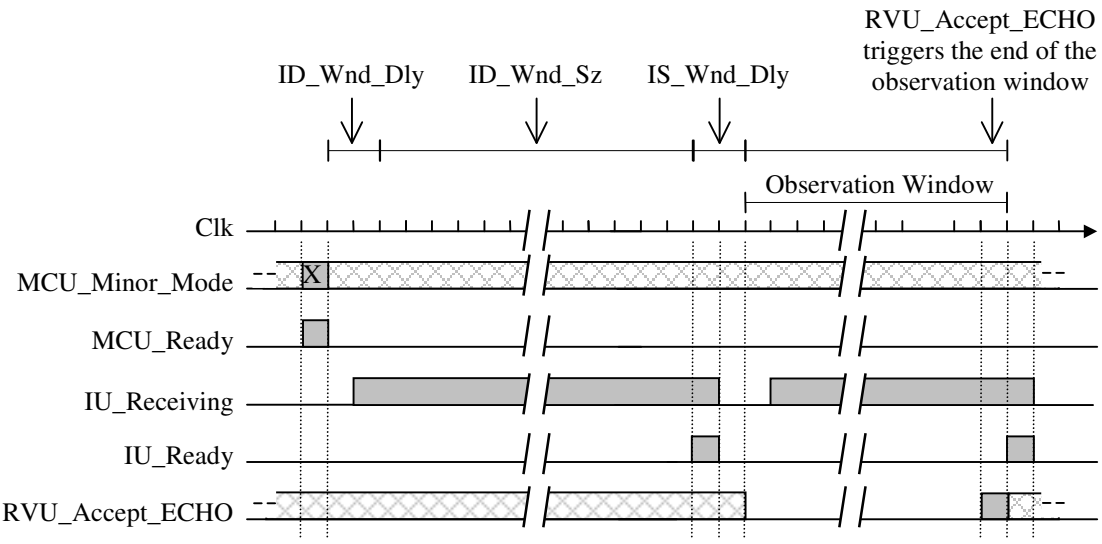


Figure A.30: Response to MCU command: Minor_Mode = ID_Initial_Sync

A.7. Required parameters

This section lists the required structural and behavioral parameters for the Input Unit. All the parameters are of type Natural. The values of these parameters are set prior to simulation and synthesis, and they remain fixed afterwards.

Additional parameters may be identified during the IU design process.

A.7.1. Structural parameters

Table A.3: Required structural parameters

Description	Parameter Label	Allowed Value Range	Typical Value
Number of BIUs (= N)	Num_BIU	≥ 1	3
Number of opposite-kind nodes (= Ω)	Num_OK	≥ 1	3
Number of syndromes from each link receiver (= L_{LS})	Num_Link_Synd	≥ 1	1
Minimum size of the input buffers	Input_FIFO_Depth	≥ 1	9

A.7.2. Behavioral parameters

Table A.4 lists the IU behavioral parameters. Some parameters are a function of the node kind specified by the MCU_Node_Kind signal. All the behavioral parameters have units of local clock ticks.

Table A.4: Required behavioral parameters

Description	Parameter Label	Allowed Value Range	Typical Value
First IU_Ready delay for Local Diagnosis Acquisition	PD_Rdy1_Dly	≥ 1	5001
Second IU_Ready delay for Local Diagnosis Acquisition	PD_Rdy2_Dly	≥ 1	5001
Window delay for Initial Diagnosis	ID_Wnd_Dly	≥ 0	0
Window size for Initial Diagnosis	ID_Wnd_Sz	≥ 1	50000
Window delay for Initial Synchronization	IS_Wnd_Dly	≥ 1	3
INIT window delay for Synchronization Preservation	SP_INIT_Wnd_Dly(Node_Kind)	≥ 0	RMU: 3 BIU: 20
INIT window size for Synchronization Preservation	SP_INIT_Wnd_Sz(Node_Kind)	≥ 1	RMU: 7 BIU: 9
ECHO window delay for Synchronization Preservation	SP_ECHO_Wnd_Dly(Node_Kind)	≥ 0	RMU: 22 BIU: 23
ECHO window size for Synchronization Preservation	SP_ECHO_Wnd_Sz(Node_Kind)	≥ 1	RMU: 9 BIU: 7
Delay for INIT pulses	INIT_Pls_Dly(Node_Kind)	≥ 1	RMU: 8 BIU: 10
Width parameter for INIT pulses	INIT_Skw(Node_Kind)	≥ 1	RMU: 3 BIU: 4
Delay for ECHO pulses	ECHO_Pls_Dly(Node_Kind)	≥ 1	RMU: 10 BIU: 8
Width parameter for ECHO pulses	ECHO_Skw(Node_Kind)	≥ 1	RMU: 3 BIU: 3
First window delay for Collective Diagnosis	CD_Wnd1_Dly	≥ 0	4
Delay between the windows for Collective Diagnosis	CD_Wnd2_Dly	≥ 1	5
First window delay for the first message stream in Schedule Update	SU_P1_Wnd1_Dly(Node_Kind)	≥ 0	RMU: 3 BIU: 15
First window delay for the second message stream in Schedule Update	SU_P2_Wnd1_Dly(Node_Kind)	≥ 1	RMU: 15 BIU: 16
Delay between expected-reception intervals for the first and second message streams in Schedule Update	SU_Wnd2_Dly	≥ 0	0
Data introduction interval for the first and second message streams in Schedule Update	SU_DII	≥ 1	2
Maximum expected input buffer load for the first and second message stream in Schedule Update	SU_Max_Buff_Cnt	≥ 1	4
Maximum number of PE messages that can be processed during PE Communication minor mode	Max_Num_PE_Msg	≥ 1	4645
Number of PE messages per PE for the default communication schedule	Dflt_Num_PE_Msg	≥ 1	50
First window delay for the message stream in PE Communication	PE_Wnd1_Dly(Node_Kind)	≥ 0	RMU: 4 BIU: 16

Description	Parameter Label	Allowed Value Range	Typical Value
Delay between expected-reception intervals for the message stream in PE Communication	PE_Wnd2_Dly	≥ 0	0
Data introduction interval for the message stream in PE Communication	PE_DII	≥ 1	2
Maximum expected input buffer load for the message stream in PE Communication	PE_Max_Buff_Cnt	≥ 1	4
Size of expected-reception interval for synchronous messages (Does not apply to Initial Diagnosis messages)	Syncns_Wnd_Sz	≥ 1	7
Gap timer timeout for Frame Synchronization	Frm_Sync_Gap(Node_Kind)	≥ 1	RMU: 3 BIU: 3

A.8. Additional remarks

This section presents various miscellaneous requirements and remarks not presented elsewhere in the document.

A.8.1. Required registered outputs

The IU is the first processing stage of the RPP for received messages. Its outputs are read directly by the MCU, IDU, and NDU units. To prevent excessive CLK-to-IU-output latency, which could have a strong negative impact on the overall processing performance of the RPP, the following IU outputs are required to be registered (i.e., to be the direct output of a flip-flop, register, or memory element):

- IU_Message_Out
- IU_Strobe_Out
- IU_Receiving
- IU_Ready
- IU_Source_Id
- IU_PE_or_ACC
- IU_Last_Message
- IU_Unexpected_Message
- IU_Link_Error
- IU_Imm_Link_Error
- IU_Empty_Buffer
- IU_Buffer_Overload
- IU_Imm_Buffer_Overload
- IU_Input_Overrun

A.8.2. Relevant parameter relations

The following relations are satisfied for the current design of the RPP.

- INIT_Pls_Dly(Node_Kind) > SP_INIT_Wnd_Sz(Node_Kind) (A.15)
- ECHO_Pls_Dly(Node_Kind) > SP_ECHO_Wnd_Sz(Node_Kind) (A.16)
- INIT_Skw(Node_Kind) ≤ SP_INIT_Wnd_Sz(Node_Kind) (A.17)
- ECHO_Skw(Node_Kind) ≤ SP_ECHO_Wnd_Sz(Node_Kind) (A.18)
- Frm_Sync_Gap(Node_Kind) = ECHO_Skw(Node_Kind) (A.19)
- SU_DII ≥ 2 (A.20)
- PE_DII ≥ 2 (A.21)
- SU_Max_Buff_Cnt = 1 for SU_Wnd2_Dly ≥ 1 (A.22)
- PE_Max_Buff_Cnt = 1 for PE_Wnd2_Dly ≥ 1 (A.23)
- SU_Max_Buff_Cnt < Syncns_Wnd_Sz (A.24)
- PE_Max_Buff_Cnt < Syncns_Wnd_Sz (A.25)
- Input_FIFO_Depth ≥ max(SU_Max_Buff_Cnt, PE_Max_Buff_Cnt) + 1 (A.26)

References

[De Micheli 94] De Micheli, Giovanni: *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[Torres 05] Torres-Pomales, Wilfredo; Malekpour, Mahyar; and Miner, Paul S.: *ROBUS-2: A Fault-Tolerant Broadcast Communication System*. NASA TM-2005-213540, 2005.

[XAPP077] Xilinx Application Note XAPP077: *Metastability Considerations*. version 1.0, Xilinx Corporation, January 1997.

[XILINX 05] *Virtex-II Platform FPGAs: Complete Data Sheet*. Xilinx Corporation, March 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 11 - 2005	Technical Memorandum	

4. TITLE AND SUBTITLE

Design of the Protocol Processor for the ROBUS-2 Communication System

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Torres-Pomales, Wilfredo; Malekpour, Mahyar R.; and Miner, Paul S.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

23-063-30-RF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-19188

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/TM-2005-213934

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 62
Availability: NASA CASI (301) 621-0390

13. SUPPLEMENTARY NOTES

An electronic version can be found at http://ntrs.nasa.gov

14. ABSTRACT

The ROBUS-2 Protocol Processor (RPP) is a custom-designed hardware component implementing the functionality of the ROBUS-2 fault-tolerant communication system. The Reliable Optical Bus (ROBUS) is the core communication system of the Scalable Processor-Independent Design for Enhanced Reliability (SPIDER), a general-purpose fault tolerant integrated modular architecture currently under development at NASA Langley Research Center. ROBUS is a time-division multiple access (TDMA) broadcast communication system with medium access control by means of time-indexed communication schedule. ROBUS-2 is a developmental version of the ROBUS providing guaranteed fault-tolerant services to the attached processing elements (PEs), in the presence of a bounded number of faults. These services include message broadcast (Byzantine Agreement), dynamic communication schedule update, time reference (clock synchronization), and distributed diagnosis (group membership). ROBUS also features fault-tolerant startup and restart capabilities. ROBUS-2 tolerates internal as well as PE faults, and incorporates a dynamic self-reconfiguration capability driven by the internal diagnostic system. ROBUS consists of RPPs connected to each other by a lower-level physical communication network. The RPP has a pipelined architecture and the design is parameterized in the behavioral and structural domains. The design of the RPP enables the bus to achieve a PE-message throughput that approaches the available bandwidth at the physical layer.

15. SUBJECT TERMS

Fault tolerance; Broadcast communication; Integrated Modular Architecture; Avionics buses

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	252	(301) 621-0390