MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CLOCKING AND SYNCHRONIZATION WITHIN A FAULT-TOLERANT MULTIPROCESSOR

by

Howard R. Krauss

June 1972

Degree of Master of Science

T-564

PREPARED AT

CHARLES STARK DRAPER LABORATORY

CAMBRIDGE, MASSACHUSETTS. 02139

CLOCKING AND SYNCHRONIZATION WITHIN A FAULT-TOLERANT MULTIPROCESSOR

by

Howard R. Krauss

B.E. The Cooper Union
(1971)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1972

Signature of Author Howard R. Krauss
Department of Aeronautics and Astronautics
May 1972

Certified by Albert L. Hopkins Jr.
Thesis Supervisor

Accepted by Judson R. Baron
Chairman, Departmental Committee on Graduate Students

CLOCKING AND SYNCHRONIZATION WITHIN A FAULT-TOLERANT MULTIPROCESSOR

by

Howard R. Krauss

Submitted to the Department of Aeronautics and Astronautics, May, 1972, in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

In this thesis the synchronization requirements of a fault-tolerant multiprocessor are defined and methods of maintenance of synchronism are developed. It is demonstrated that a synchronous fault-tolerant multiprocessor driven by a fault-tolerant clock is more efficient and more easily implemented than is an asynchronous fault-tolerant multiprocessor.

Fault-tolerant clocking has been examined intensively here. From fault-tolerance requirements and the established multiprocessor synchronization requirements, general specifications are developed for a fault-tolerant clock. Two general methods of design have been explored, and it has been concluded that if the clock is to be distributed to many modules, fault-tolerant clocking through the concepts advanced by William Daly and John McKenna of the C.S. Draper Laboratory, is more practical to implement than is fault-tolerant clocking through failure-detection and subsequent clock substitution. Clocks developed by Daly and McKenna have been examined, refined, and revised. It is demonstrated that it is desirable to have available a fault-tolerant clock which runs at 20 MHz, but that such a frequency is not achieveable by a McKenna-type clock (with use of current technology). A method of achieving the use of a relatively slow McKenna-type clock in conjunction with a frequency multiplier is developed. Also, analog phase-locking techniques are shown to be unsuitable for the design of a fault-tolerant clock.

Thesis Supervisor: Albert L. Hopkins, Jr.

Title: Associate Professor of Aeronautics and Astronautics

ACKNOWLEDGEMENTS

The author wishes to express his gratitude, first and foremost, to Dr. Albert L. Hopkins for the guidance and motivation he provided throughout the production of this thesis. Second he is grateful to John McKenna for having reviewed the work on the McKenna Clock and for the suggestions he made. Also, he is indebted to Mary Shamlian for her aid in typing the text of this thesis.

Finally, the author wishes to offer his thanks to Eileen Hack as well as the many other friends and acquaintances for their parts in filling his life.

This report was prepared under DSR Project 55-23890, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through Contract NAS 9-4065.

The publication of this report does not constitute approval by the Charles Stark Draper Laboratory or the National Aeronautics and Space Administration of the findings or the conclusions contained herein. It is published only for the exchange and stimulation of ideas.

TABLE OF CONTENTS

Page

CHAPTER 1

FAULT-TOLERANT MULTIPROCESSING

1.1 Introduction

The concept of a fault-tolerant multiprocessor was developed and explored at the C.S. Draper Laboratory as a method of satisfying future spacecraft guidance requirements. Future space vehicles will require the handling of additional control loops, and as missions become more complex and/or lengthier greater reliability is required. In a proposal to NASA particular emphasis was placed on the application of a fault-tolerant multiprocessor as a space shuttle guidance computer.

1.2 General Characteristics

The essential elements of a multiprocessor are two or more processors capable of simultaneously executing different programs (or the same programs) and a common memory accessible by all processors. This collection of units has a single path for input-output communication. A conceptual diagram is shown in Fig. 1.1.

Because of the parallel operation of the individual processors there is a significant increase in computational capability. This parallelism lends itself very well to the requirement of simultaneous control of many loops. The germ

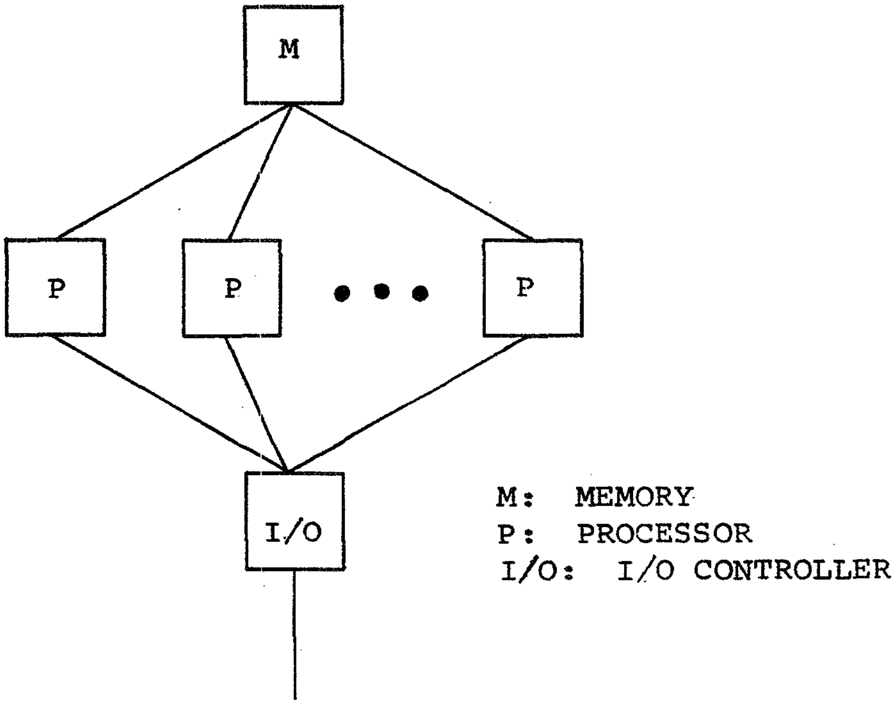


Fig. 1.1 Generalized Multiprocessor

of increased reliability is also inherent within the multiprocessing concept: processors are alike and hence any processor is capable of performing within any control loop at a given time; it is this modularity which is the fundamental idea behind fault-tolerant multiprocessing.

1.2.1 Fault-Tolerance

In general, fault-tolerance is achieved through coded redundancy, replicated redundancy, or a combination of the two. Except in special instances, redundancy by replication is more reliable and more simply implemented (Ref. 1). Within the multiprocessor under study, fault-tolerance is achieved through comparison and/or voting amongst replicated units. Some basic assumptions in the design of this system are: (1) failures are independent of one another; (2) the same error will not be made at the same time by two elements which are in a comparison or voting scheme; and (3) multiple errors will not occur so as to outwit the fault-tolerant scheme (roughly equivalent to saying that errors will be separated by some minimum time). In a system which operates such that failures are independent of one another, the probability of assumptions nos. 2 and 3 being violated is extremely small.

1.3 Particular Configuration

Figure 1.2 is a representation of the data management system recommended for the Space Shuttle. The system is designed to meet a fail operational, fail operational, fail safe (FOFOFS) specification; by this specification it is meant that the system will maintain its performance capabilities after the occurrence of any two failures and, as a result

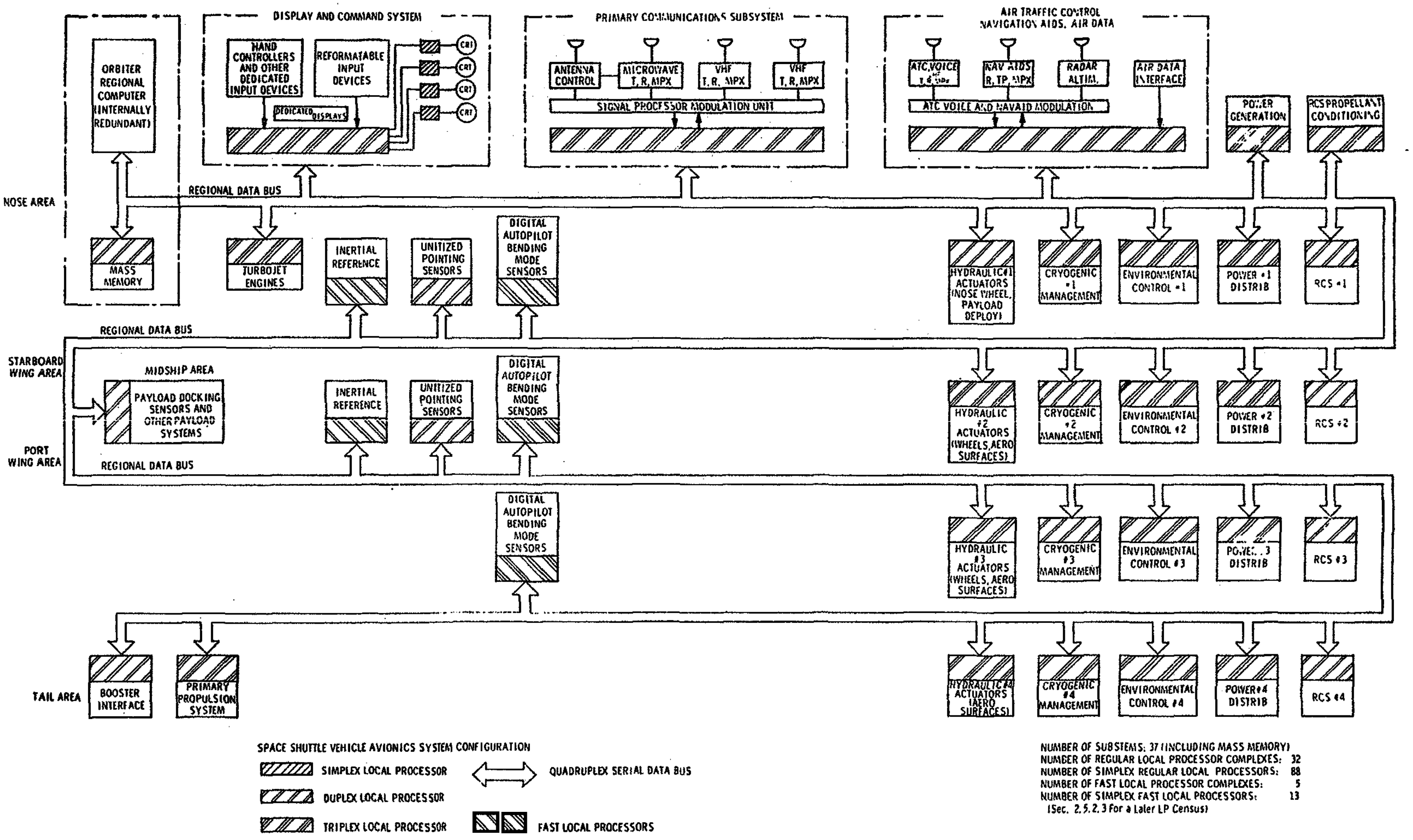

DISPLAY AND COMMAND SYSTEM
PRIMARY COMMUNICATIONS SUBSYSTEM
AIR TRAFFIC CONTROL NAVIGATION AIDS, AIR DATA
ORBITER REGIONAL COMPUTER (INTERNALLY REDUNDANT)
HAND CONTROLLERS AND OTHER DEDICATED INPUT DEVICES
REFORMATABLE INPUT DEVICES
DEDICATED DISPLAYS
CRT
CRT
CRT
CRT
ANTENNA CONTROL
MICROWAVE T, R, MPX
VHF T, R, MPX
VHF T, R, MPX
SIGNAL PROCESSOR MODULATION UNIT
ATC VOICE T & RDR
NAV AIDS R, TP, MPX
RADAR ALTIM.
AIR DATA INTERFACE
ATC VOICE AND NAVAID MODULATION
POWER GENERATION
RCS PROPELLANT CONDITIONING
NOSE AREA
REGIONAL DATA BUS
MASS MEMORY
TURBOJET ENGINES
INERTIAL REFERENCE
UNITIZED POINTING SENSORS
DIGITAL AUTOPILOT BENDING MODE SENSORS
HYDRAULIC #1 ACTUATORS (NOSE WHEEL, PAYLOAD DEPLOY)
CRYOGENIC #1 MANAGEMENT
ENVIRONMENTAL CONTROL #1
POWER #1 DISTRIB
RCS #1
REGIONAL DATA BUS
STARBOARD WING AREA
MIDSHIP AREA
PAYLOAD DOCKING SENSORS AND OTHER PAYLOAD SYSTEMS
INERTIAL REFERENCE
UNITIZED POINTING SENSORS
DIGITAL AUTOPILOT BENDING MODE SENSORS
HYDRAULIC #2 ACTUATORS (WHEELS, AERO SURFACES)
CRYOGENIC #2 MANAGEMENT
ENVIRONMENTAL CONTROL #2
POWER #2 DISTRIB
RCS #2
PORT WING AREA
REGIONAL DATA BUS
DIGITAL AUTOPILOT BENDING MODE SENSORS
HYDRAULIC #3 ACTUATORS (WHEELS, AERO SURFACES)
CRYOGENIC #3 MANAGEMENT
ENVIRONMENTAL CONTROL #3
POWER #3 DISTRIB
RCS #3
TAIL AREA
BOOSTER INTERFACE
PRIMARY PROPULSION SYSTEM
HYDRAULIC #4 ACTUATORS (AERO SURFACES)
CRYOGENIC #4 MANAGEMENT
ENVIRONMENTAL CONTROL #4
POWER #4 DISTRIB
RCS #4
SPACE SHUTTLE VEHICLE AVIONICS SYSTEM CONFIGURATION
SIMPLEX LOCAL PROCESSOR
DUPLEX LOCAL PROCESSOR
TRIPLEX LOCAL PROCESSOR
QUADRUPLEX SERIAL DATA BUS
FAST LOCAL PROCESSORS
NUMBER OF SUBSTEMS: 37 (INCLUDING MASS MEMORY)
NUMBER OF REGULAR LOCAL PROCESSOR COMPLEXES: 32
NUMBER OF SIMPLEX REGULAR LOCAL PROCESSORS: 88
NUMBER OF FAST LOCAL PROCESSOR COMPLEXES: 5
NUMBER OF SIMPLEX FAST LOCAL PROCESSORS: 13
(Sec. 2.5.2.3 For a Later LP Census)

Fig. 1.2 Assumed System Configuration

of a third failure, in the worst case, suffer a graceful degradation to a configuration which can still assure safe control of the vehicle. The system is hierarchical. The many sensors and effectors compose the lowest level. Next up in the hierarchy, the local processors transform between the serial-multiplex format of the data bus and the sensor-effector formats; also they are charged with the function of assuming those burdens which would unnecessarily overload the top level of the system. At the top, the regional computer provides data processing services to the entire system and manages interactions between subsystems.

The use of multiprocessing techniques in the regional computer (RC) serves both to achieve fault-tolerance and to yield a larger throughput than would be possible with a simplex machine. Within the local processor (LP) duplication of processors is used solely as a tool for the achievement of fault-tolerance.

1.3.1 Achievement of Fault-Tolerance Within The Regional Computer

The regional computer multiprocessor configuration recommended by the Draper Lab is shown in Fig. 1.3. Each Central Processing Unit (CPU) consists of two processors and a triplicated scratchpad memory which stores local temporary data and performs input/output (I/O) buffering. The memory, memory bus, and data bus are each redundant. The memory may be accessed by only one CPU at a time, and only one CPU (or LP) may be transmitting on the data bus at any time.

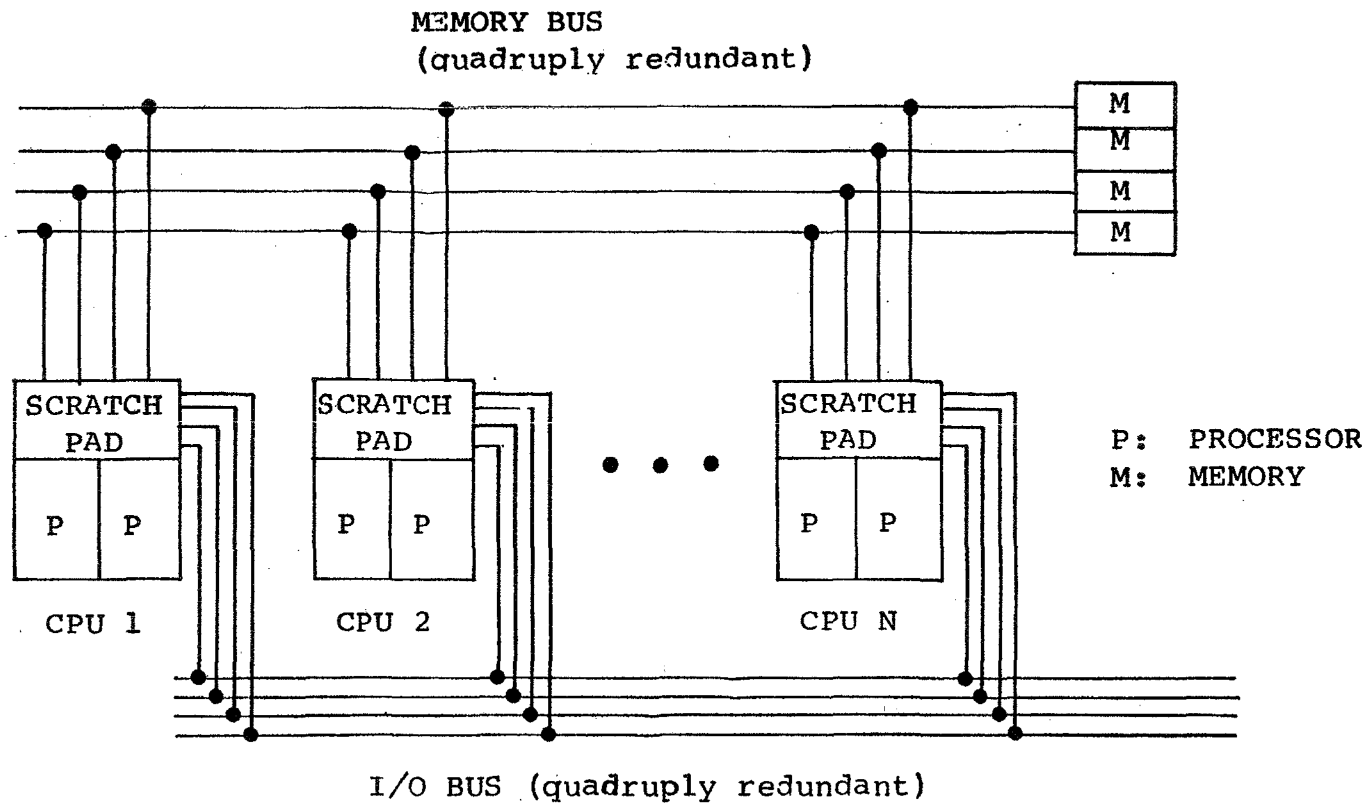


Fig. 1.3 Regional Computer Multiprocessor

CPU error detection is achieved by comparing the outputs of the two processors, which run identical programs. The detection of a CPU error triggers Single Instruction Restart (SIR), which consists of the moving of the scratchpad contents of the "failed" unit into memory and the subsequent loading of this information into the next available "healthy" CPU, whereupon the failed job is resumed. The recovery is transparent to the software.

Fault-tolerance requirements are met by providing sufficient CPUs such that after an established number have failed the remainder can provide the necessary response speed and throughput to meet the system requirements. Thus, there is the advantage of extra processing capability before any CPU fails.

1.3.2 Achievement of Fault-Tolerance Within The Local Processor

Depending upon system requirements a local processor may be simplex, duplex, or triplex. In any case error detection is achieved as in a CPU: each LP unit contains two processors which perform identical operations and compare outputs. Fault-tolerance is not through a restart mechanism, however. In the case of a duplex or triplex LP, local fault-tolerance is achieved by keeping two LP units in synchronism. One feeds the data bus and the other has its output blocked; in the event of a failure of an LP unit, its output is blocked and the other's is enabled. It is because of the differences amongst local processors serving different sensor-effector systems as well as a desire to limit data bus use that this mechanism for achieving fault-tolerance is used rather than an SIR.

CHAPTER 2

SYNCHRONIZATION

2.1 Definitions and Requirements

If fault-tolerance is to be achieved through comparison and/or voting amongst replicated units then there are two requirements of operation which establish a need for synchronization. Obviously in order for the comparison to be effective, corresponding output information from each unit must be compared. Second, in order to assure equality of internal operations, accessed input information must be equal at corresponding program points. It should be noted that although synchronization is required, simultaneous production of corresponding information or simultaneous performance of corresponding internal operations are not required. It is best to discard the notion that two events can be made to occur at the same time; in a real system there must always be finite tolerances in the "simultaneous" initiation of events. It is fortunate that the synchronization requirements do not call for simultaneity, but, as shall be seen in Section 2.2, the impossibility of assuring simultaneity gives rise to difficulties in assuring synchronization.

2.2 Loss of Synchronization

Assuming that several modules have been synchronized, loss of synchronization may be caused by a slivering of pulses.

A sliver can occur when two independent observers strobe a common signal while it is undergoing a transition. Because of gate thresholds and propagation delays, the observers may see different values of the signal. Slivering can cause differences in the sequences of internal operations (leading to uncorrelated outputs) or in the outputs of replicated comparators.

If the input or the event being strobed is phase-locked (i.e., dependent on the same clock) with the strobe, then slivering may be avoided (barring the event of a failure) through good design techniques. If, however, the observed signal is produced asynchronously, or more generally, if the production of the observed signal is uncorrelated with the strobe or the call for data, then anti-sliver circuits must be utilized as necessary.

Current designs for anti-sliver circuits require the presence of a two-phase clock. A two-phase clock may be simply produced as illustrated in Fig. 2.1. The pulses of the two phases are mutually exclusive. The application of this clock to an anti-sliver circuit is illustrated in Fig. 2.2. The event pulse is stored in the first buffer; as illustrated, either the concurrent (as in Case 1) or the next (as in Case 2) phase A pulse will cause the event to be stored in the second buffer, which is strobed, after settling, by a phase B pulse, thereby feeding a healthy signal to both units.

In a fault-tolerant system, when a replicated group of modules is receiving replicated information asynchronously, an anti-sliver unanimity circuit may be used to maintain synchronization. Such a circuit is illustrated in Fig. 2.3. The transmitted information from the A_is to the B_is is sliver-free

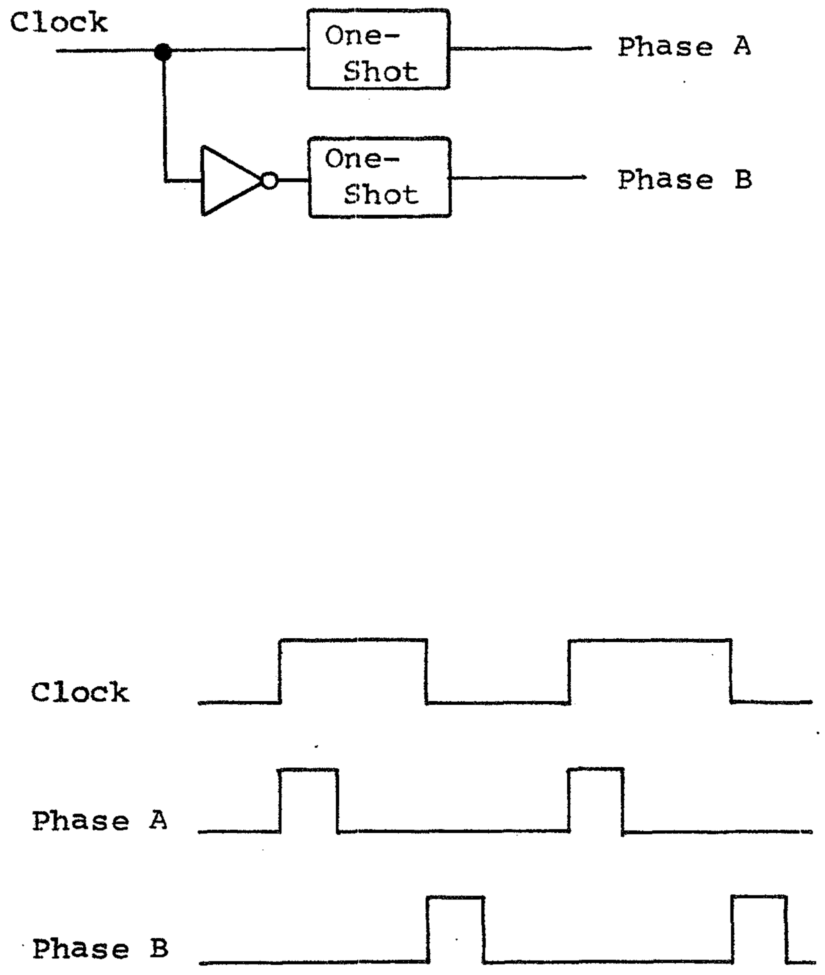


Fig. 2.1 Two-Phase Clock

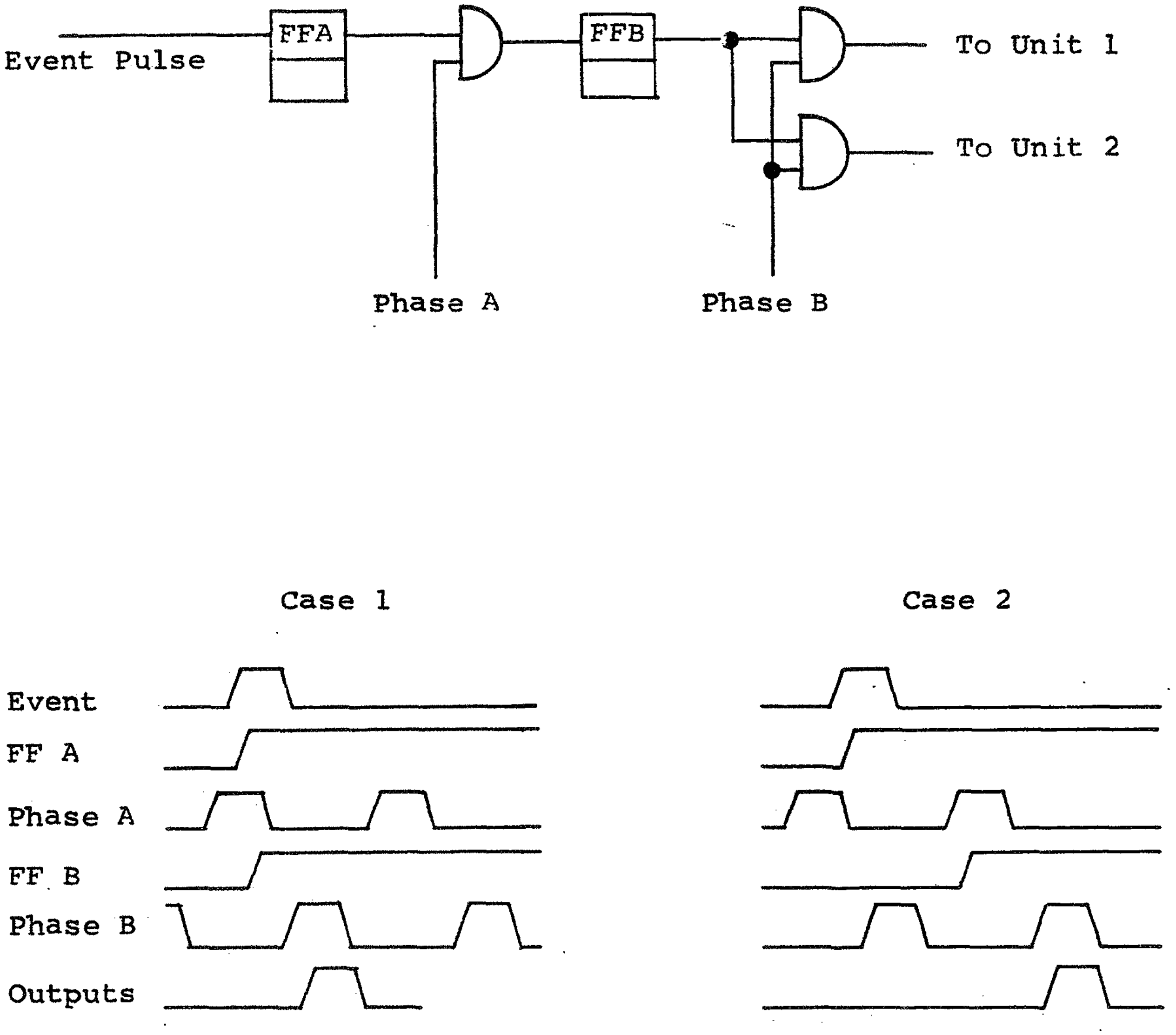


Fig. 2.2 Anti-Sliver Circuit

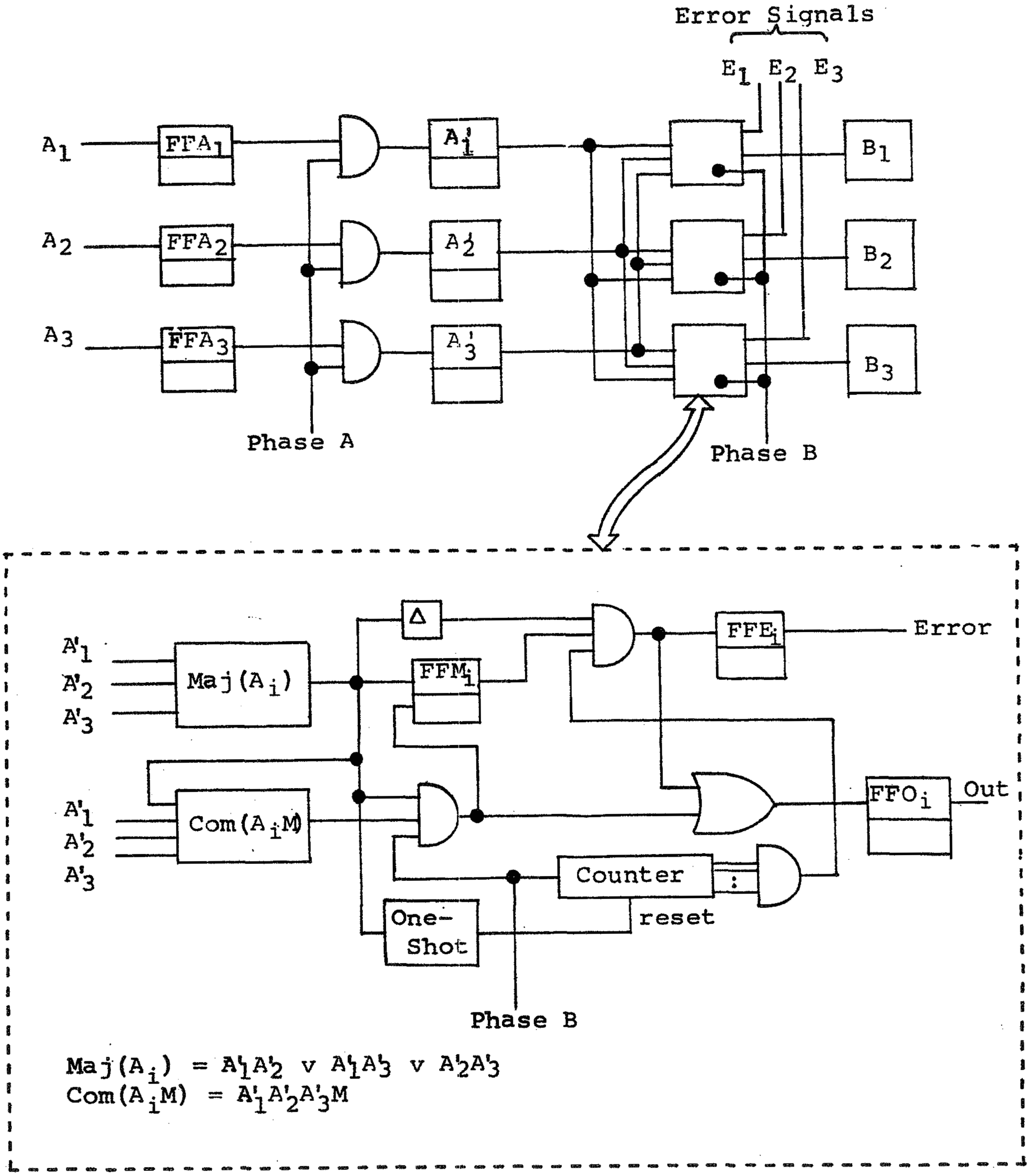


Fig. 2.3 Anti-Sliver Unanimity Circuit

and all the B_is receive synchronized information from all the A_is at the same time so that an accurate vote may be taken, and so that the B_is remain in synchronism. Each unanimity circuit waits long enough to accumulate all the A_is but not so long as to impair operation due to a failure of one of the A_is.

The timing chart in Fig. 2.4 illustrates a possible sequence of events for the case of no failure. The event to be transmitted to the B_is is $A_i \rightarrow 1$. The example shown in Fig. 2.4 indicates that the events, $A_i \rightarrow 1$, occurs first in A_2, second in A_1, and last in A_3. Each event, $A_i \rightarrow 1$, is held in the corresponding flip-flop, FFA_i. Concurrent with the first Phase-A pulse shown, only FFA_1 and FFA_2 are at logic-level-1; the flip-flops A_1' and A_2' are set during the first Phase-A pulse. The setting of the flip-flop A_3' occurs during the second Phase-A pulse. As can be seen in Fig. 2.3, each flip-flop A_i' is fed to each of the majority and comparator elements; the output of $Maj(A_i)$ is logic-1 when a majority of the inputs is at logic-1, and the output of $Comp(A_iM)$ is logic-1 only when all the inputs are at logic-1. In this example, $Maj(A_i) \rightarrow 1$ shortly after the first Phase-A pulse, while $Comp(A_iM)$ does not go to logic-1 until the occurrence of the second Phase-A pulse. The occurrence of $Maj(A_i) \rightarrow 1$: causes the counter of Phase-B pulses to be reset to zero, causes the flip-flop FFM_i to be set at logic-1, and also feeds the delay element Δ. The occurrence of $Comp(A_iM) \rightarrow 1$: causes FFM_i to be reset (here, before the counter reaches an all-1's state, preventing the propagation of an error signal), and causes an output signal to be propagated to all B_is.

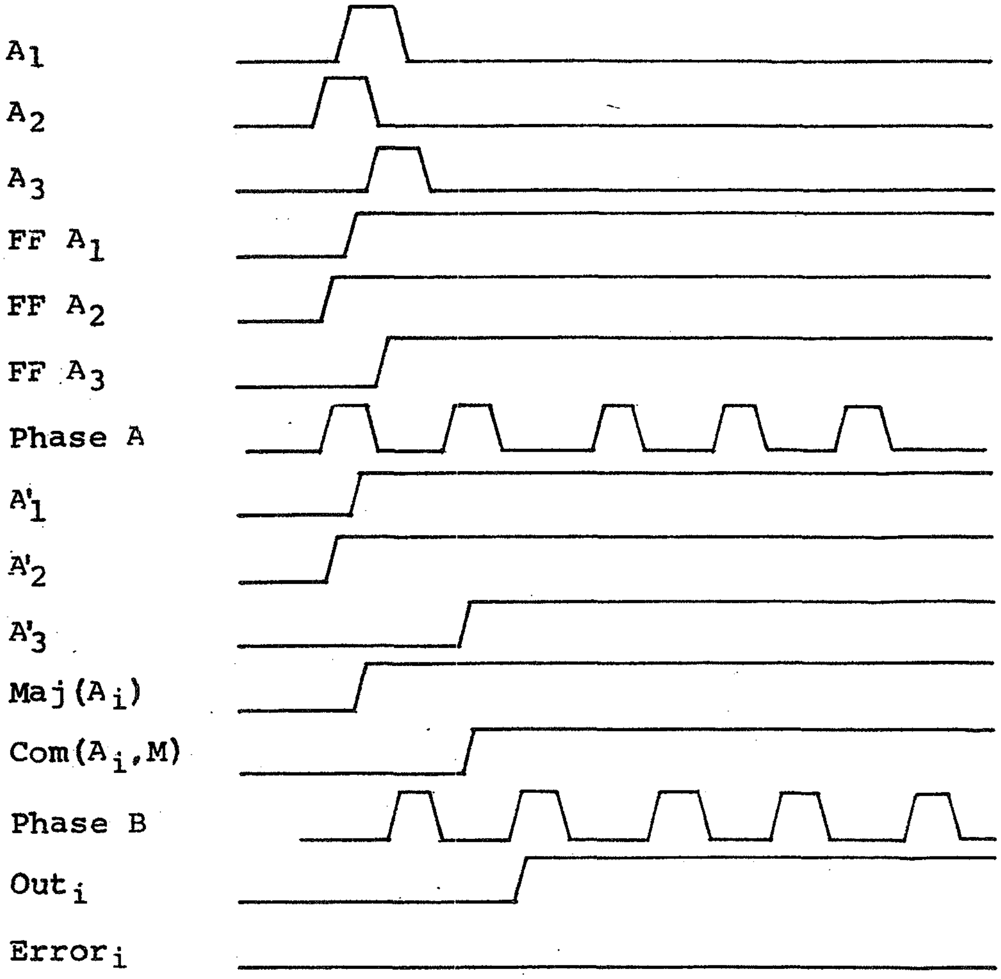


Fig. 2.4 Response of an anti-sliver unanimity circuit in the event of no failure

Fig. 2.5 illustrates a possible sequence for the case of a failure of one of the A_is. Here A_3 is described as having failed to produce a bit of information (pulse). A_1 is shown to precede A_2 such that the events are passed on to the majority and comparator elements separated by one Phase-A period. When flip-flop A_2' goes to logic-1, A_1' has already been set, and the majority elements go to logic-1, the comparator elements, however, remain at logic-0, as flip-flop A_3' was not set. Again, as a result of $Maj(A_i) \rightarrow 1$, the counter is reset, FFM_i is set, and the delay element is fed. It should be noted that the time-delay element, Δ, is used in order to prevent the false indication of an error when an all-1's state is indicated by the counter just prior to reset. The delay time required is dependent on clock frequency and propagation delay between a reset command and a response (assuming the counter was in an all-1's state) at the input of the error-indicating AND gate. In the example shown in Fig. 2.5 it is assumed that a 2-bit counter is used; hence, shortly after the occurrence of the third Phase-B pulse, succeeding the transition $Maj(A_i) \rightarrow 1$, each line indicates both an output signal and an error.

2.3 A Synchronized System With Unsynchronized Elements

Before considering a multi-layered hierarchial system, it is wise to look at a simple model of this problem. Consider two modules each with one input line and one output line, each receiving and transmitting data serially. At an arbitrary time ($t = 0$), there is no signal on any line and the internal states of the modules (processors) are equivalent. For all time, $t > 0$, equivalent input data is received in serial bytes

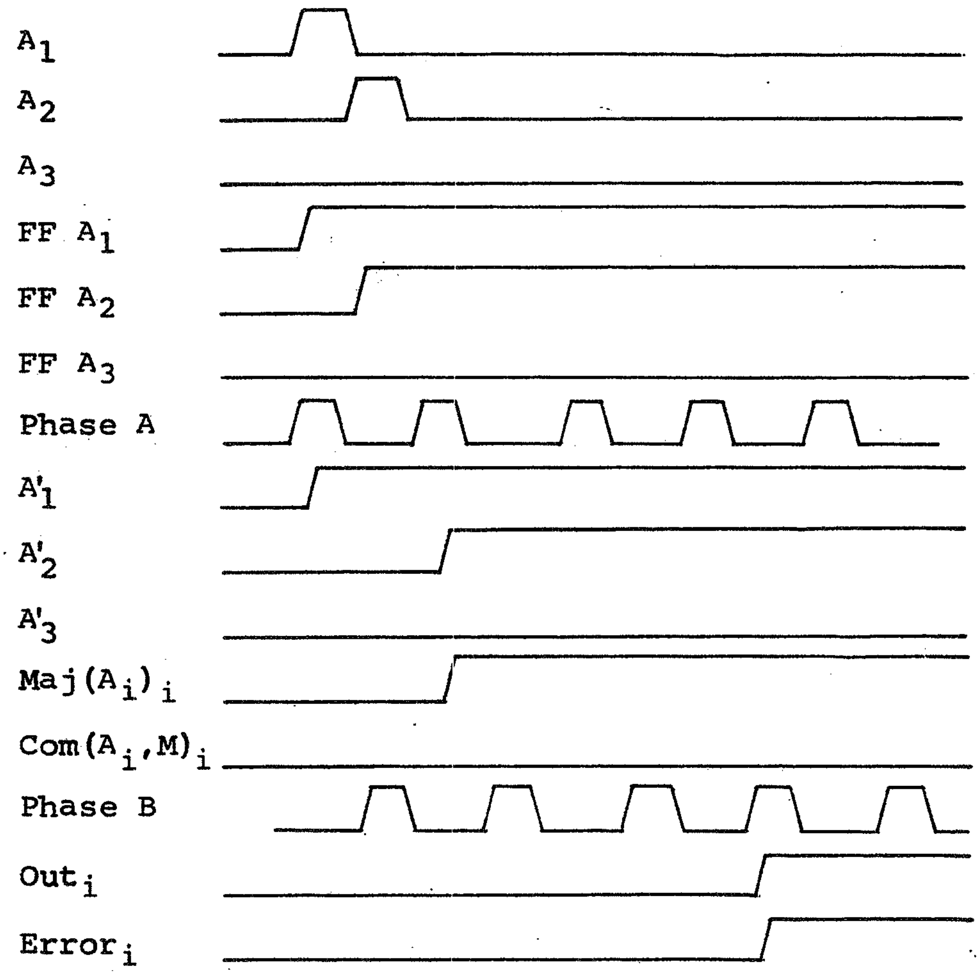


Fig. 2.5 Response of an anti-sliver unanimity circuit in the event of a failure of one of the A_is

by both modules (not simultaneously); it is desired that the modules perform identical operations with identical internal and input data, and that corresponding bits of outputted data be recognized in order to facilitate comparison. In this analysis it is not necessary to consider whether the processors are clock-driven (synchronous) or asynchronous machines; in either case, corresponding produced data bits are separated in time.

Not only may input data be received at different times by the modules, but more significantly it may be received at different points in the program being run by the two modules. In the worst case such a condition may cause calculations to be made with different numbers, or a branch to occur in one processor but not in the other.

Two theorems, which taken together contend that two independent processors (or, in general, modules) may be synchronized, are stated and proved below.

Theorem 1. Two independent modules can be made to perform identical operations with identical internal and input data.

Proof. Assume there is an interfacing unit associated with each module's input. The interfacing units may communicate with each other as well as with their associated modules - see Fig. 2.6. As in the case of the modules, the structure and operation of the interfaces are identical with each other. Assume that incoming bytes of information are buffered in corresponding registers within each interface. Let each register have two extra bits (n extra bits - in the case of n parallel modules) above the number used to store input data;

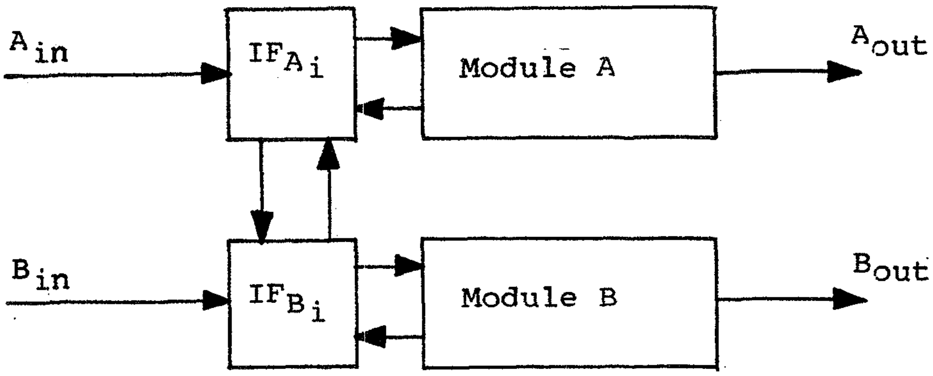


Fig. 2.6 Synchronization of Input Information

these bits are used as check bits: a 1 in, say, the left bit will be taken to mean that the input byte to "me" has been stored in this register, and a 1 in the other bit will mean that the corresponding input byte to the other module has been stored in the corresponding register. Hence each module may be aware if corresponding information is available to both. The precise nature of the structure, possible microprogram, or requirements for fault-tolerance, of the interface units need not be considered presently. Rather the purpose of this discussion is to determine, first, if a system meeting the requirements (stated earlier in this section) is possible. Through use of these check bits by the program the modules may be kept in synchronization. Note that it is important that updating of memory associated with a module by input data be controlled by the program so as to assure equality of available information at corresponding program points. Anti-slivering circuits will not be needed between interface and module as the structure of the programming will exclude slivering difficulties.

Theorem 2. Corresponding bits of output data produced by two independent modules can be recognized.

Proof. For purposes of buffering of information and comparison allow an interface unit to be associated with each module's output. To allow comparison these interfaces have communication with each other. See Fig. 2.7. At an arbitrary t = 0, the registers of the output buffers are clear and no output bytes of information are stored in the registers of the interface. When corresponding registers have been written into, the contents are compared by comparators within both

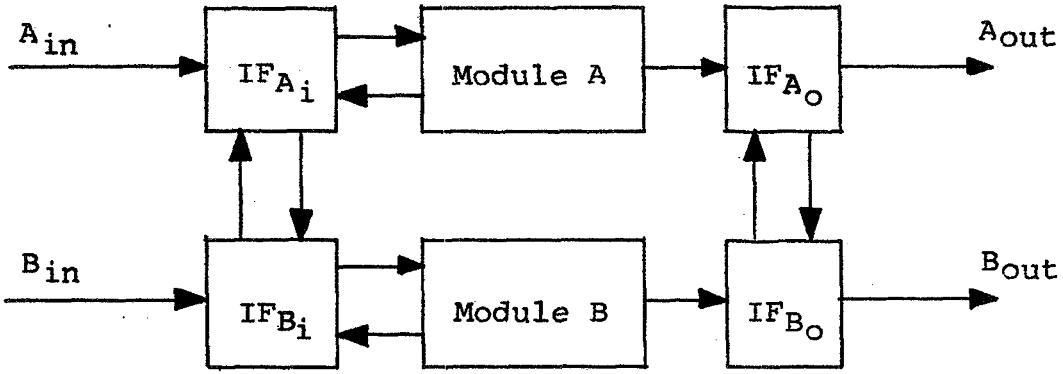


Fig. 2.7 Synchronization of Asynchronous Modules

interface units. If there is no error the data is released to output and the registers read are cleared.

In order to determine how much buffering space is required consideration must be given to such things as: peak rate of output production, maximum time separation in production of corresponding information; and rate of comparsion and clearing of registers within the interface units. If we define the following quantities:

p: peak rate of production (in bits/sec),

n: length of one register (in bits),

t_c: time required for comparison and subsequent outputting and clearing of register (in sec.),

t_s: maximum time separation in production of corresponding bits, and

x: the number of registers required for one output interfacing buffer,

then if $t_c < \frac{\text{n bits}}{\text{p bits/sec}}$:

$$x = \frac{1}{n}(pt_s + 2n)$$

If the nature of the processor is such that it produces information in serial bytes, then the peak rate of production of bits, averaged over several bytes, will be less than p. If we define:

a: peak time averaged rate of production (in bits/sec.), and

t_a: averaging time,

then if $\frac{n \text{ bits}}{p \text{ bits/sec}} < t_c < \frac{n \text{ bits}}{a \text{ bits/sec}}$:

$$x = \frac{at_a}{n} - \frac{1}{t_c} (\frac{at_a}{p}) + \frac{pt_s}{n} + 2$$

In the worst case $t_c = \frac{n \text{ bits}}{a \text{ bits/sec.}}$, then:

$$x = \frac{at_a}{n} (1 - \frac{a}{p}) + \frac{pt_s}{n} + 2$$

A numerical example would be helpful; consider:

$p = 10^7$ bits/sec,

$n = 16$ bits,

$t_s = 10^{-3}$ sec,

$a = 10^6$ bits/sec,

$t_a = 1$ sec,

then, if $t_c < \frac{n}{p}$:

$$x = \frac{pt_s}{n} + 2 = 627 \text{ registers;}$$

but if $\frac{n}{p} < t_c < \frac{n}{a}$:

$$x = \frac{at_a}{n} (1 - \frac{a}{p}) + \frac{pt_s}{n} + 2 = 56{,}877.$$

It can be seen that if the time required to ready a full register for the next load, t_c, is greater than the time it takes the processor to fill a register, $\frac{n}{p}$, then the buffering requirements are large. If t_c is larger than $\frac{n}{a}$, then the buffer must have an infinite number of registers in order to assure successful operation of the interfacing system.

The elements of design of this elementary system may be applied to a multi-layered hierarchical system. The primary drawback is in the requirements which must be imposed on the software of each processor in order to maintain synchronism of operations amongst replicated units. Different software "tricks" will be required for different input information usage; any requirement imposed on software for purposes of maintaining synchronism will serve to decrease processor speed and hence overall system speed. It is generally poor design procedure to depend on software improvisation for system operation.

2.4 System Synchronization Through Use Of A Common Clock

Consideration is now given to a system in which all units are synchronous machines and one clock is used for the driving of all units. Replicated units which are driven by the same clock may be defined to be in synchronism in the case of units which, for purposes of fault tolerance, run the same program and receive the same data (the input being controlled by the same clock), the initiation of each corresponding microprogram step as well as the receipt of corresponding bits of information occur concurrently (plus or minus some small tolerance); such units are said to be in tight synchronism.

First consider the same elementary problem explored in Section 2.3: the synchronization of two processing units. Even though corresponding input information and corresponding program steps are synchronized by the same clock, slivering may allow one unit to recognize an input a microstep before the other. However, to maintain tight synchronism anti-sliver circuits are not necessary; two-phase clocking is sufficient to avoid slivering: one phase (A) is used for receiving and transmitting of information and the other phase (B) for driving the processor. Output bits produced by phase A are buffered and then compared and transmitted by phase B.

In a multi-level hierarchical system, such as the C.S. Draper Laboratory Space Shuttle Guidance Computer proposal, this method of synchronization should be adequate for the entire system. However, information transmitted from sensor to local processor is not likely to be synchronized with the system clock; for such an interface anti-sliver circuits (or anti-sliver unanimity circuits where called for by fault tolerance requirements) can provide the necessary synchronization of receipt of information by local processors.

2.5 Conclusions

At first glance the system described in Section 2.4 is quite simple and desirable, especially in light of the alternative (Section 2.3). The difficulty in the design of a system synchronized through use of a common clock lies in the design of the clock. Such a clock must meet the fault-tolerant specifications both in its internal structure and in its distribution around the system; this is no easy task. Nevertheless it is felt that it is much more desirable to add to the complexity of hardware design by calling for a fault-tolerant clock than it is to suffer the pains of dependency on software improvisation required in an unsynchronized system. It should also be noted that although a synchronous process is generally slower than an asynchronous processor, an asynchronous fault-tolerant multiprocessor, due to increased software requirements and necessary stop and wait periods, would probably be slower than a fault-tolerant multiprocessor driven by a fault-tolerant clock.

CHAPTER 3

CLOCKING

3.1 Specifications of a Fault-Tolerant Clock

As stated in Section 2.5, if a common clock is to be used to drive the system, it must meet the fault-tolerance specifications of the overall system. Whether the system specification be fail operational or fail safe, the clock specification must be fail operational; the clock is as fundamental to the system as the power supply. In the case of a fault-tolerant clock designed to drive a Space Shuttle guidance computer, the clock would need to be able to perform after the occurrence of any combination of three independent failures.

Of prime importance in the design is that the synchronized state of the system is affected neither by any mode of failure of the clock nor by the method of recovery from the failed state (i.e., the synchronization of the system must be transparent to clock failures).

For purposes of design and discussion, the distribution of the clock to all parts of the system will be considered as a part of the clock design; this seems logical, as different concepts of fault-tolerant clocking may conceivably warrant different methods of distribution. It should be realized, however, that one of the keys to a good design will be minimization of the number of wires required for distribution. In a system such as that required for the Space Shuttle, distances between modules may be on the order of 100 feet; in such a geographically distributed system wiring may assume a large share of the cost and complexity.

As a fault-tolerant computing system may have on the order of hundreds of modules, it is desired to minimize the logic required within each to convert the distributed clock information into the one train of clock pulses which is used for driving the module. Any failure of this logic will be considered as a failure of the entire module, which will be detected by comparison of outputs amongst replicated units.

3.2 General Methods of Design

Two general design approaches come to mind: (1) use a single clock in conjunction with a single-wire bus for distribution until the occurrence of a failure in the oscillator or in the distribution, at which time another clock and its associated bus are brought into action; this principle is illustrated in Fig. 3.1; the enable circuits permit only one clock to be distributed at a time; $(\text{Enable})_n$ passes clock n if and only if failure detectors 1 through n-1 indicate failure (initially clock 1 is distributed); (2) use a group of mutually synchronized oscillators which can tolerate the required number of failures and still have several "good" outputs; see Fig. 3.2.

3.3 Fault-Tolerant Clocking Through Failure-Detection and Subsequent Clock Substitution

In this section, through logical development, an exploration is made of the feasibility of a clocking system which achieves fault-tolerance through failure-detection and subsequent

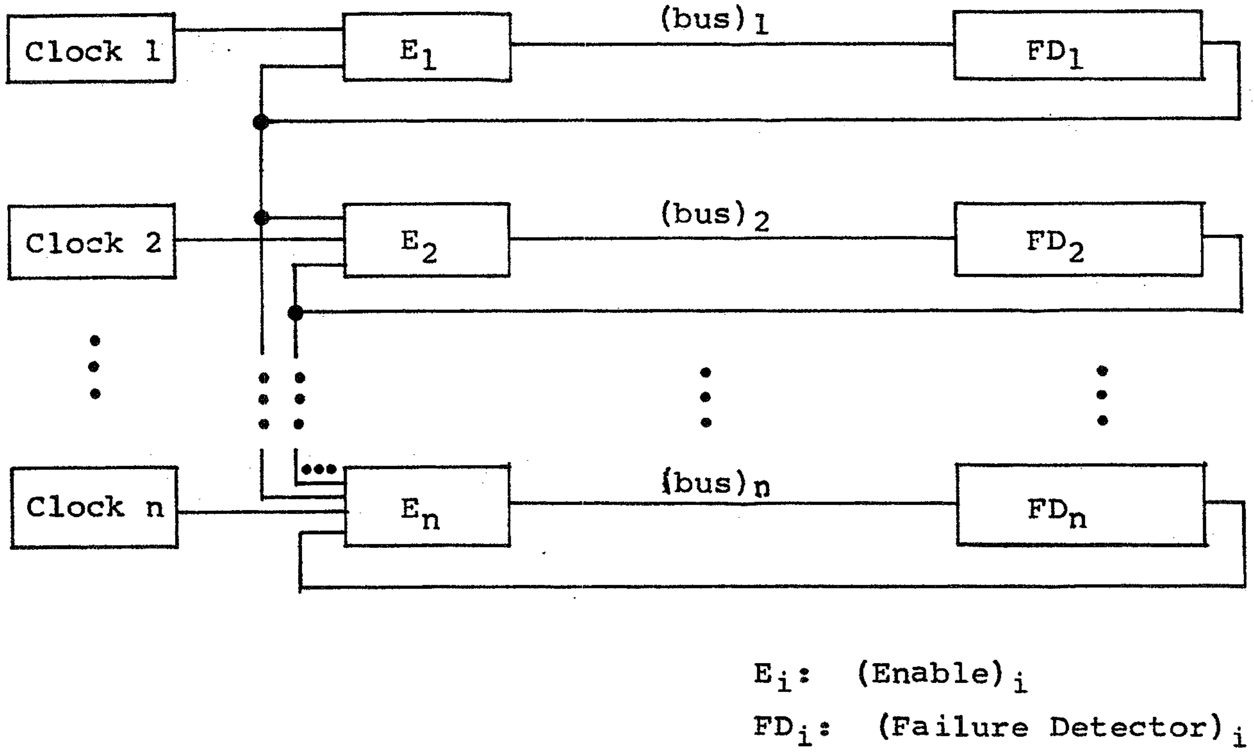


Fig. 3.1 Fault-Tolerant Clocking through Failure-Detection and Clock Substitution

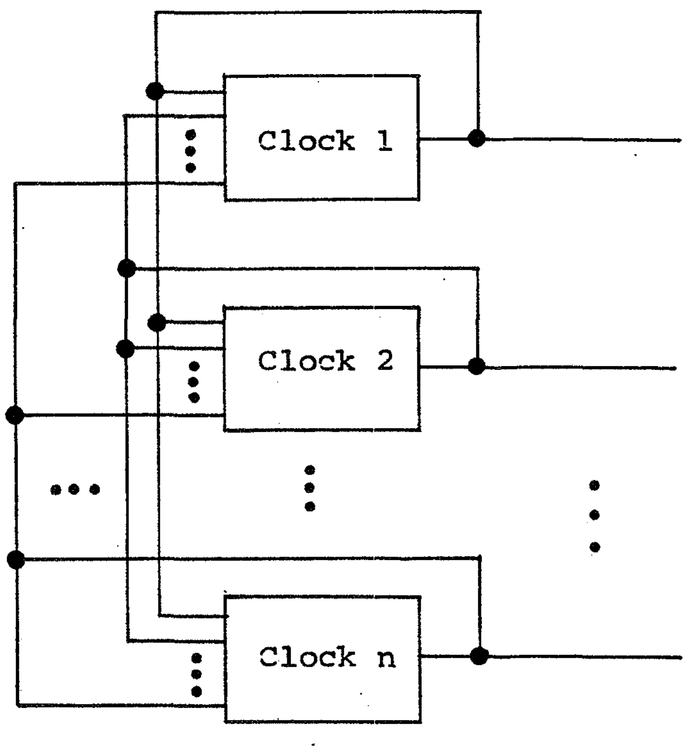


Fig. 3.2 Fault-Tolerant Clocking through Synchronization of Oscillators

clock substitution. Consider Fig. 3.3; the clock system and the clock bus are to be designed to be fault-tolerant; neither the connection between module and bus, nor the module transducer need be fault-tolerant, as a failure there may be considered to be a failure of the associated module. The design of the module transducer and its connections to the bus, however, is an integral part of the design of the clocking system; the module transducer converts the information on the bus into a single continuous clock waveform and needs to be designed such that the outputs of all module transducers are in synchronism. It will be seen that some elements of the clocking system need to be external to the module, while others need to be associated with the module.

In order to simplify the feasibility study, system design for single-fault-tolerance will be explored first. Figure 3.4 is a general description of a single-fault-tolerant clock. In order to assure that each module utilizes the same clock, failure detection should be external to the module.

The most obvious difficulty in designing the failure-detection and reconfiguration scheme is maintenance of synchronization through the failure and reconfiguration process. In order to prevent the failed clock from feeding the data management system, the clock waveform must be tested for failure before it is used; but in order to detect failures in distribution, the clock waveform must be tested after distribution. it would appear that each module transducer must be designed to "hold" (delay) use of the clock waveform until it is sure that a failure has not occurred. When a failure is detected each transducer holds its output at, say, logic-level-0, until after the clock system has been reconfigured, and a "good" clock

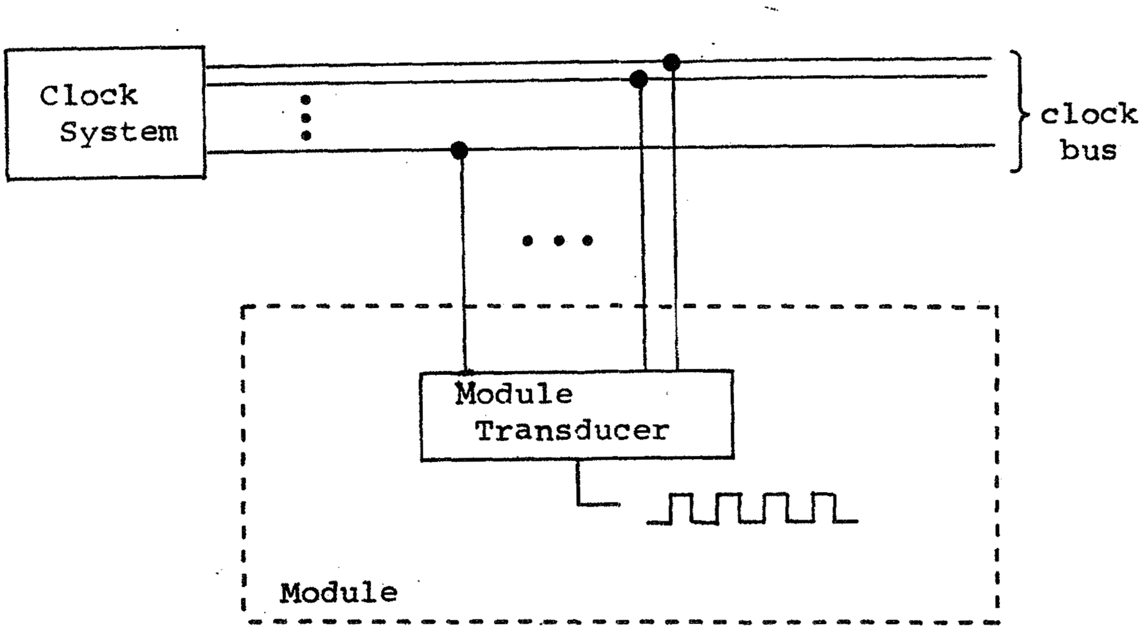


Fig. 3.3 General Fault-Tolerant Clocking Concept

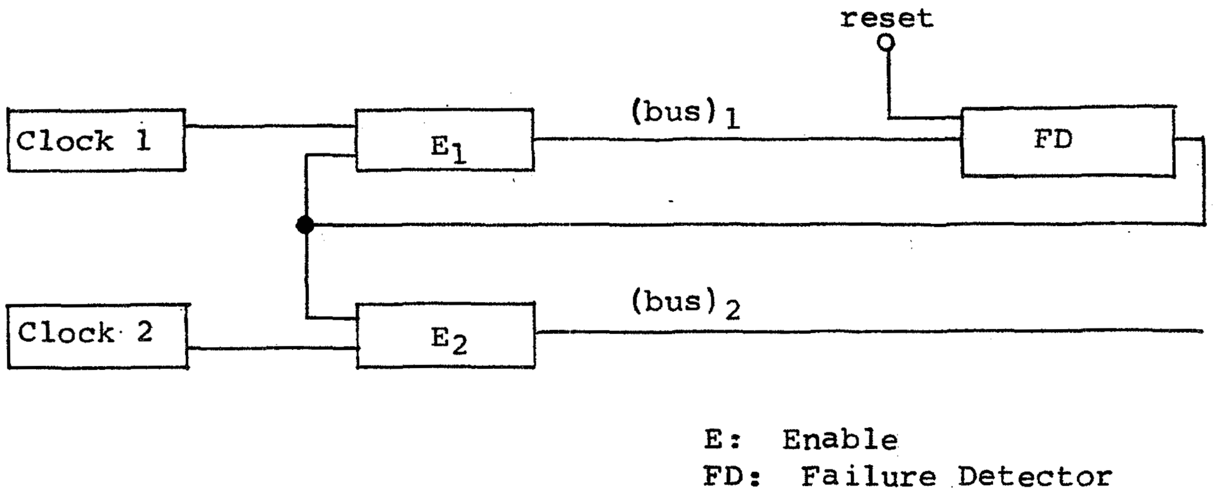


Fig. 3.4 Single-Fault-Tolerant Concept

waveform is available, at which time each transducer passes the good clock waveform. The difficulty now is in assuring that synchronization is maintained through the local transducer process of: output-hold-output.

If one of the enable circuits fails such as to produce a random output, the module transducer is required to choose the "good" waveform, if single-fault-tolerance is to exist. The amount of circuitry required for the transducer to choose the "good" waveform can be reduced if the state of the failure detector is made available to the transducers (via a failure detector bus); if this is done, the enable circuits shown in Fig. 3.4 become superfluous. Figure 3.4 may be revised as illustrated in Fig. 3.5. The failure detector may be simply implemented as illustrated in Fig. 3.6. The circuit is designed to allow a tolerance on clock pulse width and separation between pulses; if the tolerance is violated the output of the failure-detector goes to, and is held at, logic-1; the reset capability is provided for initializing the clocking system. The pulse widths of the one-shot outputs determine the tolerance; it is a straightforward procedure to determine the necessary one-shot timing, given: clock frequency, duty-cycle, and allowed variations in both, as well as data concerning tolerances, of the propagation delays and one-shot pulse widths, associated with the failure-detector components. The retriggerable one-shot should have an output pulse width of approximately twice the period of the clock; it assures the detection of a failure to logic-level-0 or logic-level-1.

Figure 3.7 is the design of a module transducer which may be used in conjunction with the clock system of Figures 3.5 and 3.6. Δ_1 is the delay associated with each module transducer;

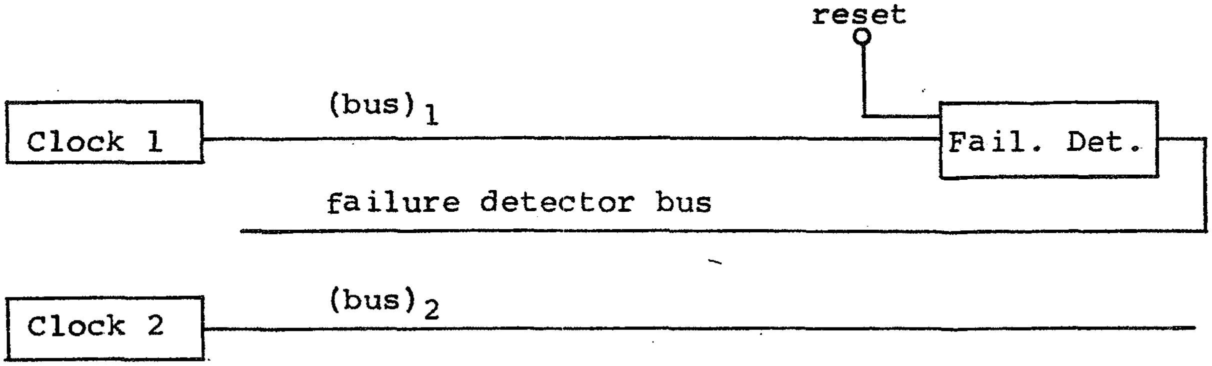


Fig. 3.5 Revised Single-Fault-Tolerant Concept

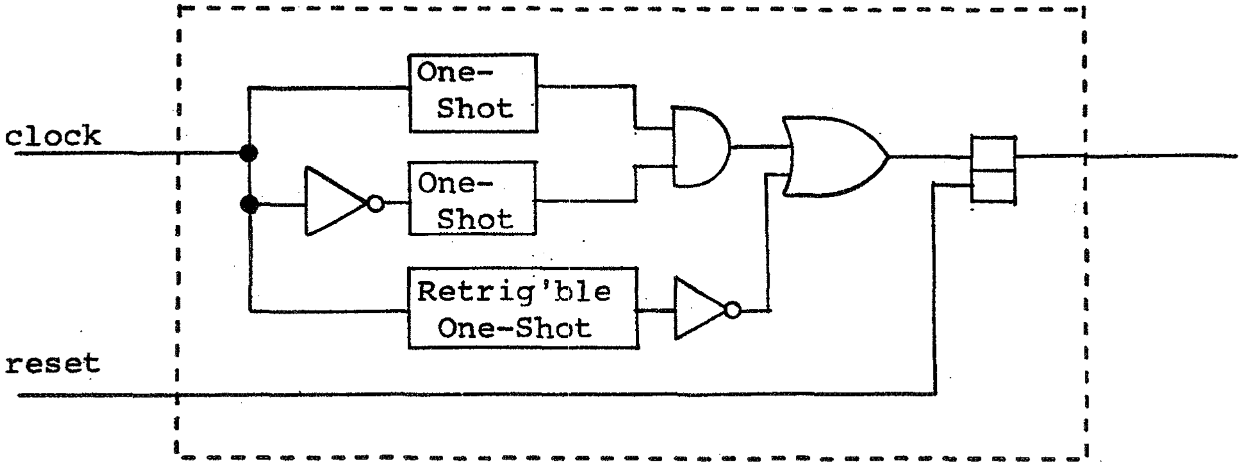


Fig. 3.6 Failure Detector

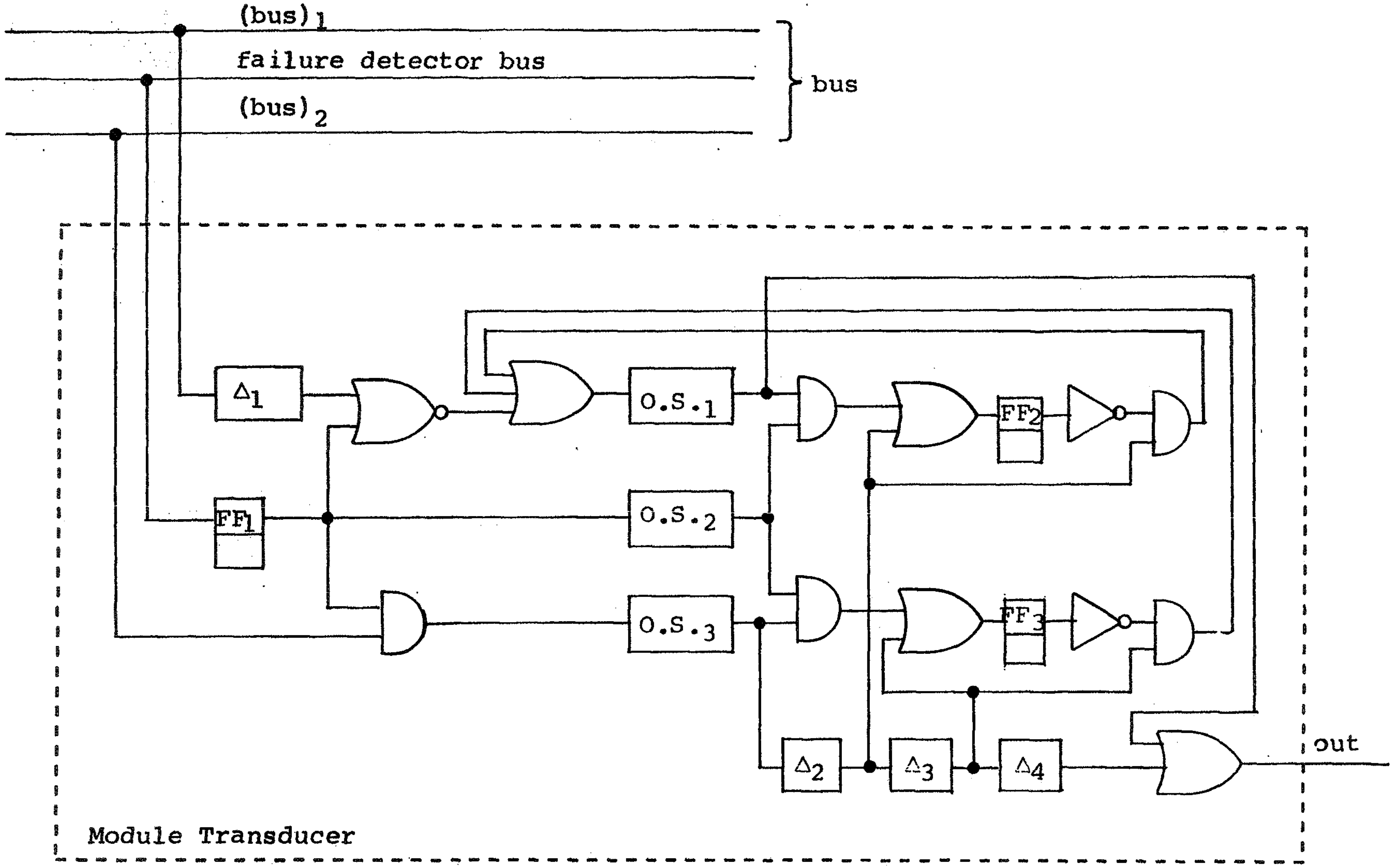


Fig. 3.7 Module Transducer for Single-Fault-Tolerant Clocking through Failure-Detection and Subsequent Clock Substitution

it allows the prevention of the propagation of a failed clock. In a system state prior to clock failure, the waveform of clock 1 is passed through delay, Δ_1, and one-shot, o.s.$_1$, and then to the output. If a failure is indicated, $FF_1 \rightarrow 1$, causing the termination of the distribution of clock 1 and the subsequent distribution of clock 2 to each module. Additional circuitry is provided in each transducer for maintenance of synchronization during and after the switching period. Slivering within the transducer in effecting the cut-off of clock 1 and the cut-in of clock 2 may, in some cases, yield an extra pulse associated with clock 1 or an extra pulse associated with clock 2; thus after the switching has taken place the total number of clock pulses supplied to each module may differ by one or two. If an "extra" pulse of clock 1 is propagated, FF_2 is set, and if an "extra" pulse of clock 2 is propagated, FF_3 is set. In those transducers in which FF_2 or FF_3 was not set during the switching process, one extra pulse for each unset flip-flop is inserted between the end of the clock 1 waveform and the beginning of the clock 2 waveform, thereby maintaining synchronism. Requirements are established for values of Δ_1, Δ_2, Δ_3, Δ_4, and the pulse widths of the one-shots, as well as their tolerances; the requirements are imposed by the system parameters (e.g., clock frequency), as well as by the required method of operation.

It is believed that the module transducer shown in Fig. 3.7 is an example of a minimally complex (or nearly so) transducer required for fault-tolerant clocking through failure detection and subsequent clock substitution. The necessity of requiring such complex operations to be performed on the module level, rather than the external clock system level, has been justified in the development of this section. Because of the module-level complexity here, it is felt that designs such as described in later sections of this thesis, are much more desirable; hence a detailed analysis of the module transducer illustrated in Fig. 3.7 is not presented.

For n-fault-tolerance, the system becomes more complicated on all levels, and, of course, it is the increased complication on the module level which is particularly undesirable. It is concluded that fault-tolerant clocking through failure detection and subsequent clock substitution is possible, but is extremely costly in hardware implementation.

3.4 The McKenna Clock

3.4.1 First Concept

In August, 1971, William Daly and John F. McKenna, in a C.S. Draper Laboratory memo (Ref. 2), described their design of a fault-tolerant clock. Figure 3.8 illustrates the concept proposed for single-fault-tolerance. It is seen that this design conforms to the method of clocking shown in Fig. 3.2 and indeed may be described as a synchronization of oscillators. It should be noted, however, that here a single clock element, apart from the others, is not an oscillator. Rather, as shall be seen in the analysis to follow, each clock element depends on the occurrence of the transition in state of several of the clocks in order to be driven to change its own state.

The quorum function Q_a^n is defined to be 1 if at least a of the n independent variables C_1, C_2, ... C_n are 1, and 0 otherwise. For example:

$$Q_1^4 = C_1 + C_2 + C_3 + C_4$$

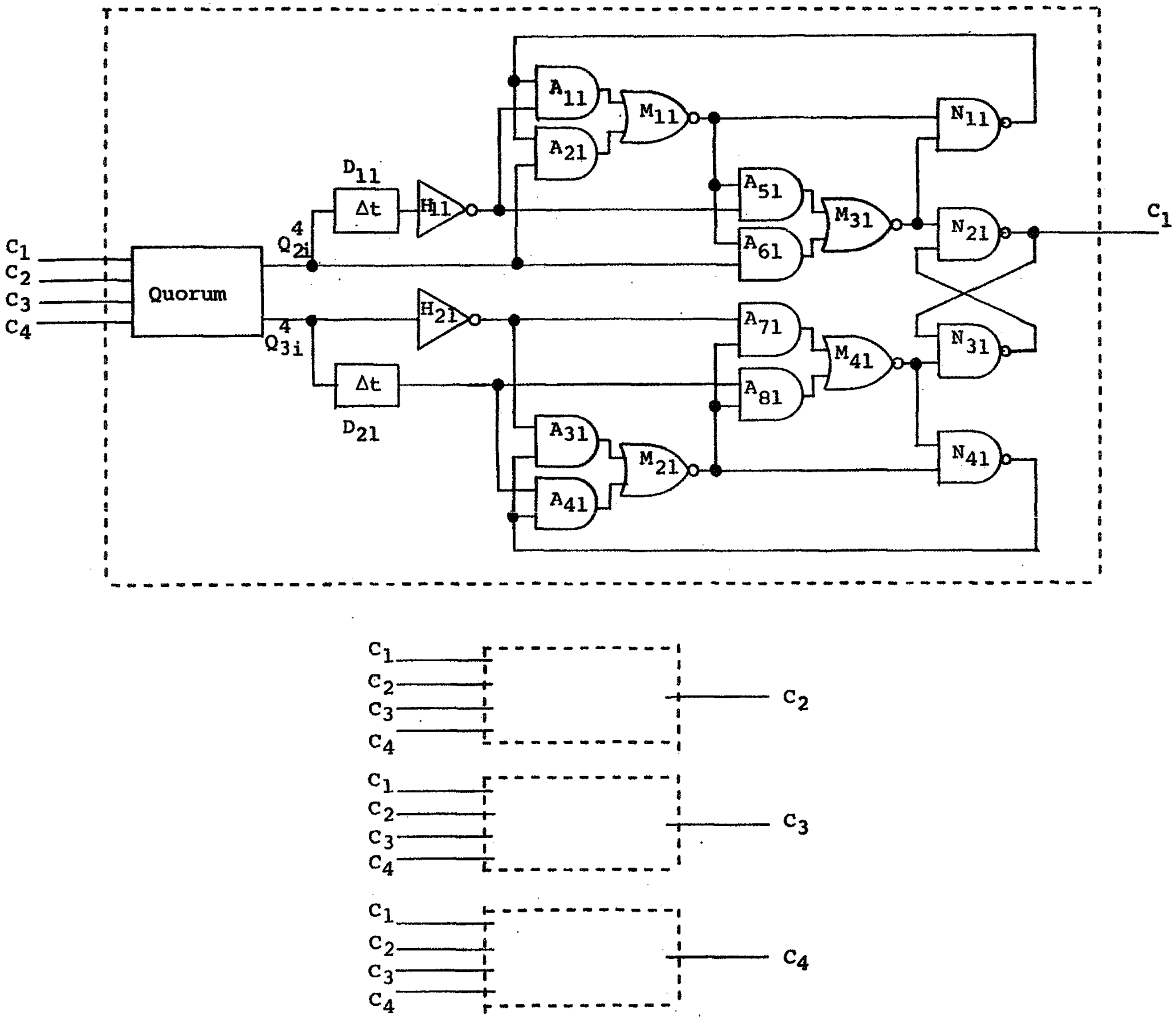


Fig. 3.8 Daly-McKenna Clock

$$Q_2^4 = C_1C_2 + C_1C_3 + C_1C_4 + C_2C_3 + C_2C_4 + C_3C_4$$

$$Q_3^4 = C_1C_2C_3 + C_1C_2C_4 + C_1C_3C_4 = C_2C_3C_4$$

$$Q_4^4 = C_1C_2C_3C_4$$

Q_2^4 and Q_3^4 may each be realized through two levels of Boolean gating or through one level of threshold logic. The use of threshold logic, however, offers no advantage unless LSI threshold logic technology is to be used.

All clock lines are distributed to each synchronous module within the system. It will be demonstrated in a later part of this section that majority voting and subsequent filtering within each module is necessary to maintain system synchronization, and an adequate module transducer will be described.

The logic required to induce free-running oscillation after power-on is not shown in Fig. 3.8. For the purposes of this analysis it will be assumed that the clock elements are already oscillating in synchronism at the time of observation. Timing analyses will be made, to demonstrate system performance.

For purposes of analysis the following assumptions are made: each gate has a propagation delay equal to Δ; the propagation delay in forming a quorum function is 2Δ; Δt is a pure delay, greater than or equal to 8Δ (in order to avoid slivering within the clock element which could yield spikes in the output, Δt must be greater than the propagation delay through the series of gates: H_{21}, A_{31}, M_{21}, N_{41}, which is 4Δ; however, since by most manufacturers' specifications propagation delay within a simple Boolean gate may reach to nearly twice the typical delay time, in the worst case the delay through four gates in series may approach 8Δ.

Assume that the clock elements are oscillating in synchronism. Assume that at time: $t=t_1$, C_1 goes to logic -1; $t=t_2$, $C_2 \rightarrow 1$; $t=t_3$, $C_3 \rightarrow 1$; and at $t=t_4$, $C_4 \rightarrow 1$ (see Fig. 3.9), where $t_4 > t_3 > t_2 > t_1$. Assume that at an initial time of observation, $(t_1 - \epsilon)$, all propagation, within each clock element, caused by the previous transition, $C_i \rightarrow 0$, has ceased (this will be verified within the analysis to follow); therefore the outputs of the gates (Ref. Fig. 3.8) are as follows:

Initially: $C_i = 0$

$Q^4_{2i} = 0;$ $Q^4_{3i} = 0$

$D_{1i} = 0;$ $D_{2i} = 0$

$H_{1i} = 1;$ $H_{2i} = 1$

$A_{1i} = 1;$ $A_{2i} = 0;$ $M_{1i} = 0$

$A_{3i} = 1;$ $A_{4i} = 0;$ $M_{2i} = 0$

$A_{5i} = 0;$ $A_{6i} = 0;$ $M_{3i} = 1$

$A_{7i} = 0;$ $A_{8i} = 0;$ $M_{4i} = 1$

$N_{1i} = 1;$ $N_{4i} = 1$

$N_{2i} \equiv C_i = 0$

$N_{3i} = 1$

Given the initial state and the assumed progression of events, the timing analysis is as follows: for reference purposes all transitions are numbered.

$$C_1 \rightarrow 1 \text{ at } t_1 \ (N_{21} \rightarrow 1) \tag{3.1}$$

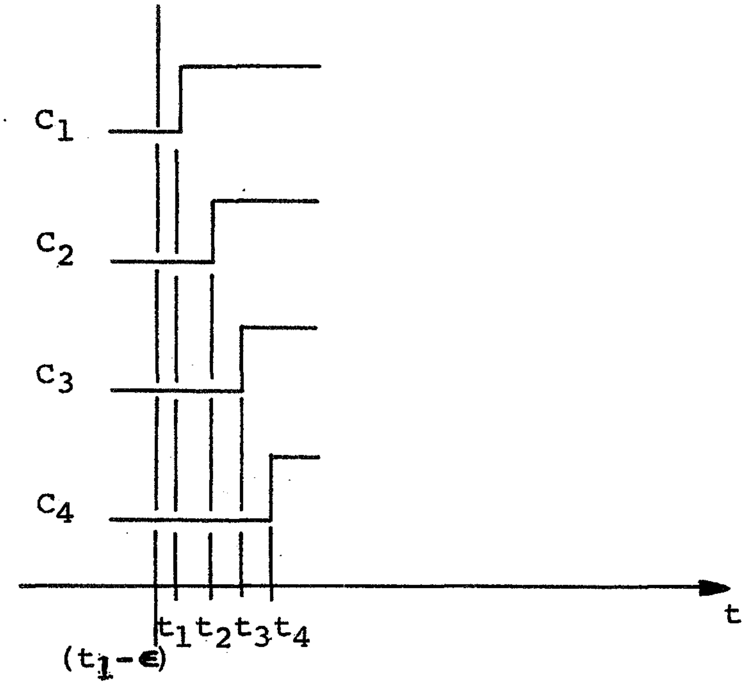


Fig. 3.9 Assumed Synchronism

Q^4_{2i} and Q^4_{3i} remain at 0

$\therefore$ gate X_{xi}, i = 2,3,4, remain unchanged from the initial state

$N_{31} \rightarrow 0$ at $t_1 + \Delta$ (the validity of this statement is dependent on the nature of transition (3.1)) (3.2)

$C_2 \rightarrow 1$ at t_2 ($N_{22} \rightarrow 1$) (3.3)

$Q^4_{2i} \rightarrow 1$ at $t_2 + 2\Delta$ (3.4)

Q^4_{3i} remains at 0

$N_{32} \rightarrow 0$ at $t_2 + \Delta$ (as in (3.2)) (3.5)

$A_{2i} \rightarrow 1$ at $t_2 + 3\Delta$ (3.6)

$D_{1i} \rightarrow 1$ at $t_2 + 2\Delta + \Delta t$ (3.7)

$H_{1i} \rightarrow 0$ at $t_2 + 3\Delta + \Delta t$ (3.8)

$A_{1i} \rightarrow 0$ at $t_2 + 4\Delta + \Delta t$ (3.9)

It has already been assumed that $\Delta t \geq 8\Delta$, but it is still interesting to note that for $\Delta t = 0$, a short duration pulse might or might not be generated by M_{1i}, yielding possible slivering within the clock's logic and hence unpredictable operation.

$C_3 \rightarrow$ 1 at t_3 ($N_{23} \rightarrow 1$) (3.10)

$N_{33} \rightarrow 0$ at $t_3 + \Delta$ (as in (3.2)) (3.11)

Q^4_{2i} remains at 1

D_{1i} remains at 1

$Q^4_{3i} \rightarrow 1$ at $t_3 + 2\Delta$ (3.12)

$H_{2i} \rightarrow 0$ at $t_3 + 3\Delta$ (3.13)

$$A_{3i} \rightarrow 0 \text{ at } t_3 + 4\Delta \tag{3.14}$$

$$M_{2i} \rightarrow 1 \text{ at } t_3 + 5\Delta \tag{3.15}$$

$$N_{4i} \rightarrow 0 \text{ at } t_3 + 6\Delta \tag{3.16}$$

$$D_{2i} \rightarrow 1 \text{ at } t_3 + 2\Delta + \Delta t \tag{3.17}$$

$$A_{8i} \rightarrow 1 \text{ at } t_3 + 3\Delta + \Delta t \tag{3.18}$$

$$M_{4i} \rightarrow 0 \text{ at } t_3 + 4\Delta + \Delta t \tag{3.19}$$

$$N_{3i} \rightarrow 1 \text{ at } t_3 + 5\Delta + \Delta t \tag{3.20}$$

$$(N_{2i} \equiv C_i) \rightarrow 0 \text{ at } t_3 + 6\Delta + \Delta t \tag{3.21}$$

$$N_{4i} \rightarrow 1 \text{ at } t_3 + 5\Delta + \Delta t \tag{3.22}$$

$$A_{4i} \rightarrow 1 \text{ at } t_3 + 6\Delta + \Delta t \tag{3.23}$$

$$M_{2i} \rightarrow 0 \text{ at } t_3 + 7\Delta + \Delta t \tag{3.24}$$

$$A_{8i} \rightarrow 0 \text{ at } t_3 + 8\Delta + \Delta t \tag{3.25}$$

$$M_{4i} \rightarrow 1 \text{ at } t_3 + 9\Delta + \Delta t \tag{3.26}$$

As a result of (3.21):

$$Q^4_{2i} \rightarrow 0 \text{ at } t_3 + 8\Delta + \Delta t \tag{3.27}$$

$$Q^4_{3i} \rightarrow 0 \text{ at } t_3 + 8\Delta + \Delta t \tag{3.28}$$

$$A_{2i} \rightarrow 0 \text{ at } t_3 + 9\Delta + \Delta t \tag{3.29}$$

$$M_{1i} \rightarrow 1 \text{ at } t_3 + 10\Delta + \Delta t \tag{3.30}$$

$$N_{1i} \rightarrow 0 \text{ at } t_3 + 11\Delta + \Delta t \tag{3.31}$$

$$D_{1i} \rightarrow 0 \text{ at } t_3 + 8\Delta + 2\Delta t \tag{3.32}$$

$$H_{1i} \rightarrow 1 \text{ at } t_3 \div 9\Delta + 2\Delta t \tag{3.33}$$

$$A_{5i} \rightarrow 1 \text{ at } t_3 + 10\Delta + 2\Delta t \tag{3.34}$$

$$M_{3i} \rightarrow 0 \text{ at } t_3 + 11\Delta + 2\Delta t \tag{3.35}$$

$$(N_{2i} = C_i) \rightarrow 1 \text{ at } t_3 + 12\Delta + 2\Delta t \tag{3.36}$$

$$N_{3i} \rightarrow 0 \text{ at } t_3 + 13\Delta + 2\Delta t \tag{3.37}$$

$$N_{1i} \rightarrow 1 \text{ at } t_3 + 12\Delta + 2\Delta t \tag{3.38}$$

$$A_{1i} \rightarrow 1 \text{ at } t_3 + 13\Delta + 2\Delta t \tag{3.39}$$

$$M_{1i} \rightarrow 0 \text{ at } t_3 + 14\Delta + 2\Delta t \tag{3.40}$$

$$A_{5i} \rightarrow 0 \text{ at } t_3 + 15\Delta + 2\Delta t \tag{3.41}$$

$$M_{3i} \rightarrow 1 \text{ at } t_3 + 16\Delta + 2\Delta t \tag{3.42}$$

$$H_{2i} \rightarrow 1 \text{ at } t_3 + 9\Delta + \Delta t \tag{3.43}$$

$$A_{3i} \rightarrow 1 \text{ at } t_3 + 10\Delta + \Delta t \tag{3.44}$$

$$D_{2i} \rightarrow 0 \text{ at } t_3 + 8\Delta + 2\Delta t \tag{3.45}$$

$$A_{8i} \rightarrow 0 \text{ at } t_3 + 9\Delta + 2\Delta t \tag{3.46}$$

$$A_{4i} \rightarrow 0 \text{ at } t_3 + 9\Delta + 2\Delta t \tag{3.47}$$

It is seen that the last transition of each gate (up to transition (3.47)) restores the gate to its initial setting. Because of transition (3.36), $C_i \rightarrow 0$ at a time later: $6\Delta + \Delta t$; and because of $C_i \rightarrow 0$, $C_i \rightarrow 1$ in another increment of time: $6\Delta + \Delta t$. The duty-cycle of C_i is 50%. The period is:

$$T_{ci} = 12\Delta + 2\Delta t$$

Hence the maximum frequency is:

$$f_{max} = \frac{1}{28\Delta}$$

For medium speed TTL, $\Delta \approx 12$ ns; therefore $f_{max} \approx 3$ MHz.

With use of the above timing analysis, the assertion that

Δt must be greater than 4Δ, for successful operation, may be demonstrated. Assume that $\Delta < \Delta t < 4\Delta$; then transition (3.17) occurs before (3.16) and a possible timing sequence is as follows:

$$D_{2i} \rightarrow 1 \text{ at } t_3 + 2\Delta + \Delta t \tag{3.48}$$

$$N_{4i} \rightarrow 0 \text{ at } t_3 + 6\Delta \tag{3.49}$$

$$A_{8i} \rightarrow 1 \text{ at } t_3 + 3\Delta + \Delta t \tag{3.50}$$

$$A_{4i} \rightarrow 1 \text{ at } t_3 + 3\Delta + \Delta t \tag{3.51}$$

$$A_{4i} \rightarrow 0 \text{ at } t_3 + 7\Delta \tag{3.52}$$

$$M_{2i} \rightarrow 0 \text{ at } t_3 + 4\Delta + \Delta t \tag{3.53}$$

$$M_{4i} \rightarrow 0 \text{ at } t_3 + 4\Delta + \Delta t \tag{3.54}$$

$$M_{2i} \rightarrow 1 \text{ at } t_3 + 8\Delta \tag{3.55}$$

$$N_{4i} \rightarrow 1 \text{ at } t_3 + 5\Delta + \Delta t \tag{3.56}$$

$$N_{3i} \rightarrow 1 \text{ at } t_3 + 5\Delta + \Delta t \tag{3.57}$$

$$A_{8i} \rightarrow 0 \text{ at } t_3 + 5\Delta + \Delta t \tag{3.58}$$

$$A_{8i} \rightarrow 1 \text{ at } t_3 + 9\Delta \tag{3.59}$$

$$A_{4i} \rightarrow 1 \text{ at } t_3 + 6\Delta + \Delta t \tag{3.60}$$

$$(N_{21} \equiv C_1) \rightarrow 0 \text{ at } t_3 + 6\Delta + \Delta t \tag{3.61}$$

$$M_{4i} \rightarrow 1 \text{ at } t_3 + 6\Delta + \Delta t \tag{3.62}$$

$$M_{4i} \rightarrow 0 \text{ at } t_3 + 10\Delta \tag{3.63}$$

Transitions (3.61) and (3.62) both occur at $t_3 + 6\Delta + \Delta t$, but if (3.62) occurs just before (3.61) the following transition may occur in some clock elements:

$$N_{3i} \rightarrow 0 \text{ at } t_3 + 7\Delta + \Delta t \tag{3.64}$$

$$(N_{2i} \equiv C_i) \rightarrow 1 \text{ at } t_3 = 8\Delta + \Delta t \tag{3.65}$$

So, it is seen that for $\Delta < \Delta t < 4\Delta$, proper operation cannot be assured.

Now assume that $\Delta t < \Delta$; then as a result of transition (3.12):

$$D_{2i} \rightarrow 1 \text{ at } t_3 + 2\Delta + \Delta t \qquad (3.66)$$

$$H_{2i} \rightarrow 0 \text{ at } t_3 + 3\Delta \qquad (3.67)$$

$$A_{4i} \rightarrow 1 \text{ at } t_3 + 3\Delta + \Delta t \qquad (3.68)$$

$$A_{3i} \rightarrow 0 \text{ at } t_3 + 4\Delta \qquad (3.69)$$

Since (3.68) occurs before (3.69), M_{2i} will not go to 1 and therefore C_i will not go to 0, yielding a non-oscillatory condition.

It has been shown, in support of the original assertion, that Δt must be greater than 4Δ. Also, as mentioned, in order to assure operation in the event that gate propagation delays are nearly double nominal value, Δt should be no less than $8\Delta_{nom.}$.

The principle of operation is seen to be as follows: when $Q_3^4 \rightarrow 1$, $C_i \rightarrow 0$ $6\Delta + \Delta t$ later; when $Q_2^4 \rightarrow 0$, $C_i \rightarrow 1$ $6\Delta + \Delta t$ later. Differences amongst clock elements in the propagation of the signal triggered by the leading edge of the Q_3^4 pulse or in the propagation of the signal triggered by the trailing edge of the Q_2^4 pulse will cause minor time separations in occurrences of leading and trailing edges amongst clock element pulses. It should be noted that if for clock element 1, $Q_{2i}^4 \rightarrow 1$ before $H_{1i} \rightarrow 1$, then C_i is driven to logic-1 5Δ after $Q_2^4 \rightarrow 1$. Similarly, if $Q_{3i}^4 \rightarrow 0$ before $D_{2i} \rightarrow 1$, then C_i is driven to logic-0 6Δ after $Q_3^4 \rightarrow 0$.

For purposes of determining the effect of differences in propagation delays on clock performance consider the following definitions and analysis: first, assume that the delays of each clock element are within tolerances such that the "set-to-agree" ($Q_2^4 \rightarrow 1$ drives $C_i \rightarrow 1$ or $Q_3^4 \rightarrow 0$ drives $C_i \rightarrow 0$) function is not utilized in normal operation; now, define the time between the event $Q_3^4 \rightarrow 1$ (here, Q refers to the conceptual quorum function, not the physical implementation) and the resulting event $C_i \rightarrow 0$ as $(\delta t_d)_i$; define the time between the event $Q_2^4 \rightarrow 0$ and the resulting $C_i \rightarrow 1$ as $(\delta t_u)_i$. Once the event $Q_3^4 \rightarrow 1$ has occurred, $Q_2^4 \rightarrow 0$ will occur $(\delta t_d)_x$ later, where x is the clock possessing the next to the largest (δt_d); after $Q_2^4 \rightarrow 0$ occurs, $Q_3^4 \rightarrow 1$ will occur $(\delta t_u)_y$ later, where y is the clock possessing the next to the largest (δt_u).

Two specifications which considered together offer a significant measurement of clock performance, may be defined as follows:

$\Delta T_u \equiv$ duty-cycle of the function $C_1\ C_2\ C_3\ C_4$

$\Delta T_d \equiv$ duty-cycle of the function $\overline{C}_1\ \overline{C}_2\ \overline{C}_3\ \overline{C}_4$.

These two specifications indicate, respectively, the size of the overlap region of the clock pulses and the size of the overlap region of the clock 0-states. If there were no differences amongst propagation delays of like elements then ΔT_u and ΔT_d would both be $\frac{1}{2}$. Following is a derivation of the relationships between ΔT_u, ΔT_d and the (δt_u)'s, (δt_d)'s:

Assign numbers to the clocks such that:

$$(\delta t_d)_1 < (\delta t_d)_2 < (\delta t_d)_3 < (\delta t_d)_4 \qquad (3.70)$$

Assign letters to the clocks such that:

$$(\delta t_u)_a < (\delta t_u)_b < (\delta t_u)_c < (\delta t_u)_d \tag{3.71}$$

There are 4! distinct sets of the four clock outputs, yet in each case:

the period of the clock is given by:

$$T = (\delta t_d)_3 + (\delta t_u)_c \tag{3.72}$$

and

$$\Delta T_u = \frac{1}{T} [(\delta t_d)_1 - (\delta t_u)_d + (\delta t_u)_c] \tag{3.73}$$

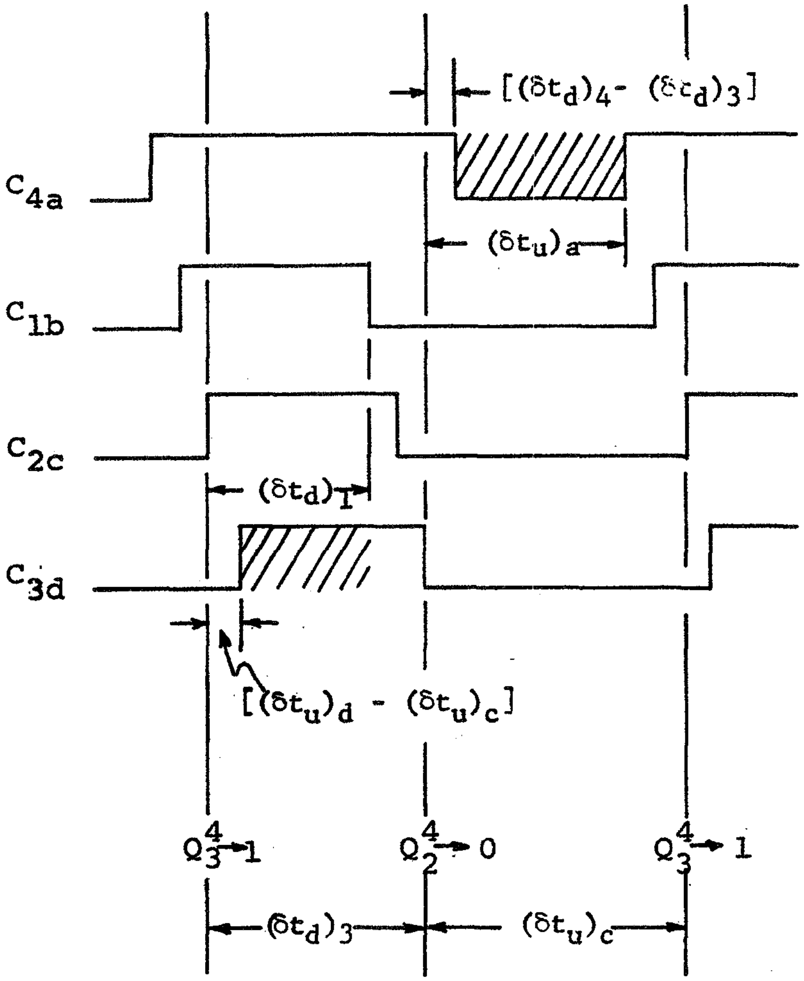
$$\Delta T_d = \frac{1}{T} [(\delta t_u)_a - (\delta t_d)_4 + (\delta t_d)_3] \tag{3.74}$$

Two cases are illustrated (Figs. 3.10, 3.11) for the purpose of shedding some light on why ΔT_u and ΔT_d are independent of the manner of the pairings of the (δt_d)'s with the (δt_u)'s in the four clocks.

The percentage variation of the (δt_d)'s and the (δt_u)'s around some nominal value is dependent on both component specifications and component selection; testing and subsequent selection of components will yield a minimum variation. A worst-case analysis will yield a direct correlation between clock performance, as measured by ΔT_u and ΔT_d, and the tolerances of the (δt_d)'s and (δt_u)'s. If (δt_u) and (δt_d) fall within a range $(\delta t)_{nom} (1-x) < \delta t < (\delta t)_{nom} (1+x)$, then from eqns (3.72), (3.73), and (3.74):

$$[\Delta T_u]_{min} = [\Delta T_d]_{min} = \begin{cases} \frac{1-3x}{2}, & x < \frac{1}{3} \\ 0, & x \geq \frac{1}{3} \end{cases} \tag{3.75}$$

Note that due to the "set-to-agree" function it is unrealistic

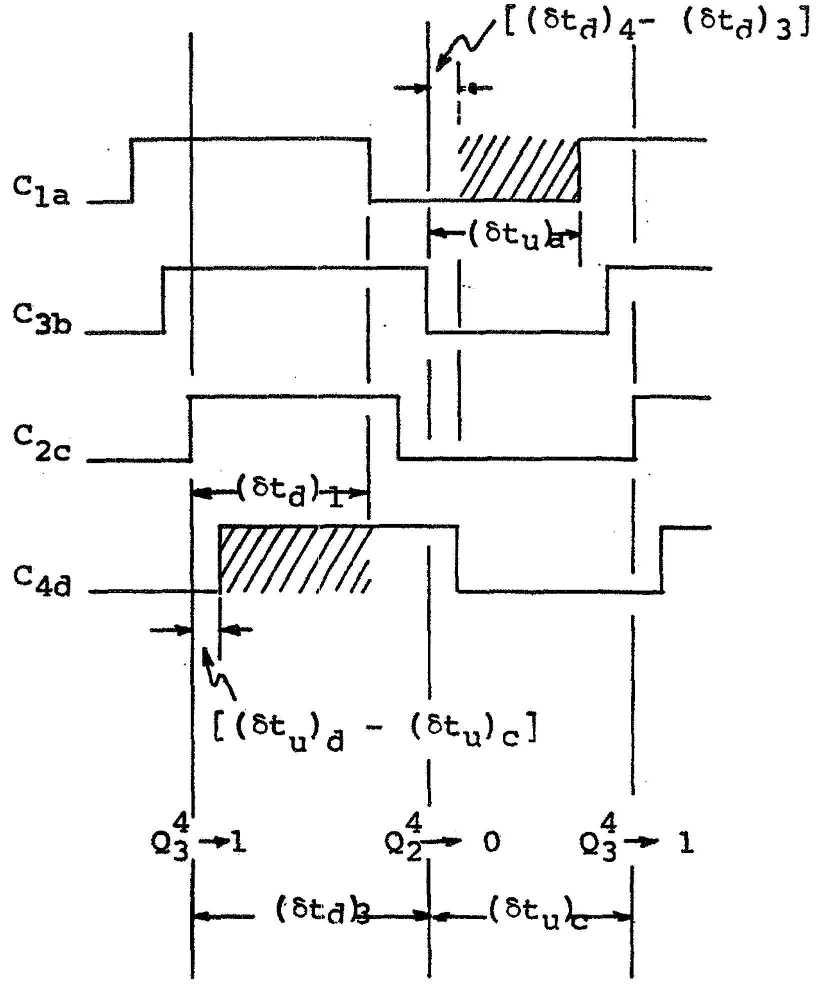


$$T = (\delta t_d)_3 + (\delta t_u)_c$$

$$\Delta T_u = \frac{1}{T}[(\delta t_d)_1 - (\delta t_u)_d + (\delta t_u)_c]$$

$$\Delta T_d = \frac{1}{T}[(\delta t_u)_a - (\delta t_d)_4 + (\delta t_d)_3]$$

Fig. 3.10 Determination of ΔT_u and ΔT_d - Case 1



$$T = (\delta t_d)_3 + (\delta t_u)_c$$

$$\Delta T_u = \frac{1}{T}[(\delta t_d)_1 - (\delta t_u)_d + (\delta t_u)_c]$$

$$\Delta T_d = \frac{1}{T}[(\delta t_u)_a - (\delta t_d)_4 + (\delta t_d)_3]$$

Fig. 3.11 Determination of ΔT_u and ΔT_d - Case 2

to assume that $[(\delta t_d)_3 - (\delta t_d)_2]$ or $[(\delta t_d)_4 - (\delta t_d)_2]$ is larger than 6Δ, or that $[(\delta t_u)_c - (\delta t_u)_b]$ or $[(\delta t_u)_d - (\delta t_u)_b]$ is larger than 5Δ, but that the above analysis is valid because, in the worst case $(\delta t_u)_a$ and $(\delta t_d)_1$ may be equal to $(\delta t)_{nom.}(1-x)$ and, the other (δt_d)'s and (δt_u)'s equal to $(\delta t)_{nom.}(1+x)$; in this worst case the set-to-agree function is not utilized.

By nature of the design of the clock, any single failure can directly affect the output of only one clock element. In order to examine the post-failure operation of the circuit it is not necessary to examine the failure-modes of every logic gate; rather it is sufficient to examine the effect on clock operation of one failed clock element. The operation of the clock is examined below for three modes of failure of any one clock element: failed to logic-level-1, failed to logic-level-0, and random oscillation. Flaws in the design of the clock will be exposed, and a revised design will be recommended.

The following definitions will be useful in the analysis:

$Q^3_n = 1$ if and only if n out of the three "good" clock elements are at logic-level-1.

Using previously defined nomenclature, assign numbers and letters to the three "good" clock elements such that:

$$(\delta t_d)_1 < (\delta t_d)_2 < (\delta t_d)_3$$

$$(\delta t_u)_a < (\delta t_u)_b < (\delta t_u)_c$$

Denote (δt_d) and (δt_u) of the failed clock element, prior to failure,as $(\delta t_d)_f$ and $(\delta t_u)_f$, respectively.

If the failure is to logic-level-1 and if the system does not fail as a result of the occurrence of the failure, then the event $Q_3^4 \rightarrow 1$ is equivalent to $Q_2^3 \rightarrow 1$, the event $Q_2^4 \rightarrow 0$ is equivalent to $Q_1^3 \rightarrow 0$, and the principle of operation of the three good clocks remains the same: once the event $Q_2^3 \rightarrow 1$, has occurred, $Q_1^3 \rightarrow 0$ will occur $(\delta t_d)_3$ later; as a result of the event $Q_1^3 \rightarrow 0$, $Q_2^3 \rightarrow 1$ will occur $(\delta t_u)_b$ later. Therefore the period of oscillation of each good clock is:

$$T_{f1} = \frac{1}{(\delta t_d)_3 + (\delta t_u)_b} \tag{3.76}$$

where the subscript f1 denotes failure to logic-level-1. Unless $(\delta t_d)_f > (\delta t_d)_3$ and $(\delta t_u)_f < (\delta t_u)_b$, the period prior to failure differs from T_{f1}.

If the failure is to logic-level-0 and if the system does not fail as a result of the occurrence of the failure, then the event $Q_3^4 \rightarrow 1$ is equivalent to $Q_3^3 \rightarrow 1$, the event $Q_2^4 \rightarrow 0$ is equivalent to $Q_2^3 \rightarrow 0$, and: once $Q_3^3 \rightarrow 1$ has occurred, $Q_2^3 \rightarrow 0$ will occur $(\delta t_d)_2$ later; as a result of the event $Q_2^3 \rightarrow 0$, $Q_3^3 \rightarrow 1$ will occur $(\delta t_u)_c$ later. Therefore the period of oscillation of each good clock is:

$$T_{f0} = \frac{1}{(\delta t_d)_2 + (\delta t_u)_c} \tag{3.77}$$

where the subscript f0 denotes failure to logic-level-0. Unless $(\delta t_d)_f < (\delta t_d)_2$ and $(\delta t_u)_f > (\delta t_u)_c$, the period prior to failure differs from T_{f0}.

Prior to failure if $(\delta t_d)_f > (\delta t_d)_3$ and $(\delta t_u)_f > (\delta t_u)_c$ then:

$$T = \frac{1}{(\delta t_d)_3 + (\delta t_u)_c}$$

If $(\delta t_d)_f > (\delta t_d)_3$ and $(\delta t_u)_b < (\delta t_u)_f < (\delta t_u)_c$ then:

$$T = \frac{1}{(\delta t_d)_3 + (\delta t_u)_f}$$

If $(\delta t_d)_f > (\delta t_d)_3$ and $(\delta t_u)_f < (\delta t_u)_b$,

$$T = T_{f1} = \frac{1}{(\delta t_d)_3 + (\delta t_u)_b}$$

If $(\delta t_d)_2 < (\delta t_d)_f < (\delta t_d)_3$ and $(\delta t_u)_f > (\delta t_u)_c$,

$$T = \frac{1}{(\delta t_d)_f + (\delta t_u)_c}$$

If $(\delta t_d)_2 < (\delta t_d)_f < (\delta t_d)_3$ and $(\delta t_u)_b < (\delta t_u)_f < (\delta t_u)_c$,

$$T = \frac{1}{(\delta t_d)_f + (\delta t_u)_f}$$

If $(\delta t_d)_2 < (\delta t_d)_f < (\delta t_d)_3$ and $(\delta t_u)_f < (\delta t_u)_b$,

$$T = \frac{1}{(\delta t_d)_f + (\delta t_u)_b}$$

If $(\delta t_d)_f < (\delta t_d)_2$ and $(\delta t_u)_f > (\delta t_u)_c$,

$$T = T_{f0} = \frac{1}{(\delta t_d)_2 + (\delta t_u)_c}$$

If $(\delta t_d)_f < (\delta t_d)_2$ and $(\delta t_u)_b < (\delta t_u)_f < (\delta t_u)_c$,

$$T = \frac{1}{(\delta t_d)_2 + (\delta t_u)_f}$$

If $(\delta t_d)_f < (\delta t_d)_2$ and $(\delta t_u)_f < (\delta t_u)_b$,

$$T = \frac{1}{(\delta t_d)_2 + (\delta t_u)_b}$$

It has been shown that if the failure is to logic-level-1 or logic-level-0, and if the system does not fail as a result of the occurrence of the failure, then the three good clock elements continue operation, but at a frequency determined by the new set of (δt_d)'s and (δt_u)'s of the three operating elements.

Figures 3.12 through 3.15 show how the occurrence of a failure may induce a spurious short-duration pulse in the waveform of Q_2^4 or Q_3^4. In the remaining text of the thesis such a short-duration pulse will be referred to as a glitch. It is shown below that the clock will tolerate the failure-modes illustrated in Figs. 3.12 and 3.13, providing that $(\delta t_d)_i$ and $(\delta t_u)_i$ are within prescribed tolerances, but that the failure-mode illustrated in Fig. 3.15 may induce a glitch to appear in any clock element output. Because the undesirable transistions which occur in Q_2^4 or Q_3^4 may be extremely rapid, slivering may occur within the clock elements; the study which will be made in each case represents the worst-case analysis.

Case 1. Again, in this case and those following, reference is made to Fig. 3.8. In Fig. 3.12 C_1 fails to logic-1 within the region: $(\delta t_d)_2 < t_{f1} < (\delta t_d)_3$; the analysis of this failure-mode follows:

$$Q_{3i}^4 \rightarrow 0 \text{ at } t = (\delta t_d)_2 + 2\Delta \qquad (3.78)$$

$$Q_{3i}^4 \rightarrow 1 \text{ at } t_{f1} + 2\Delta \qquad (3.79)$$

$$Q_{3i}^4 \rightarrow 0 \text{ at } (\delta t_d)_3 + 2\Delta \qquad (3.80)$$

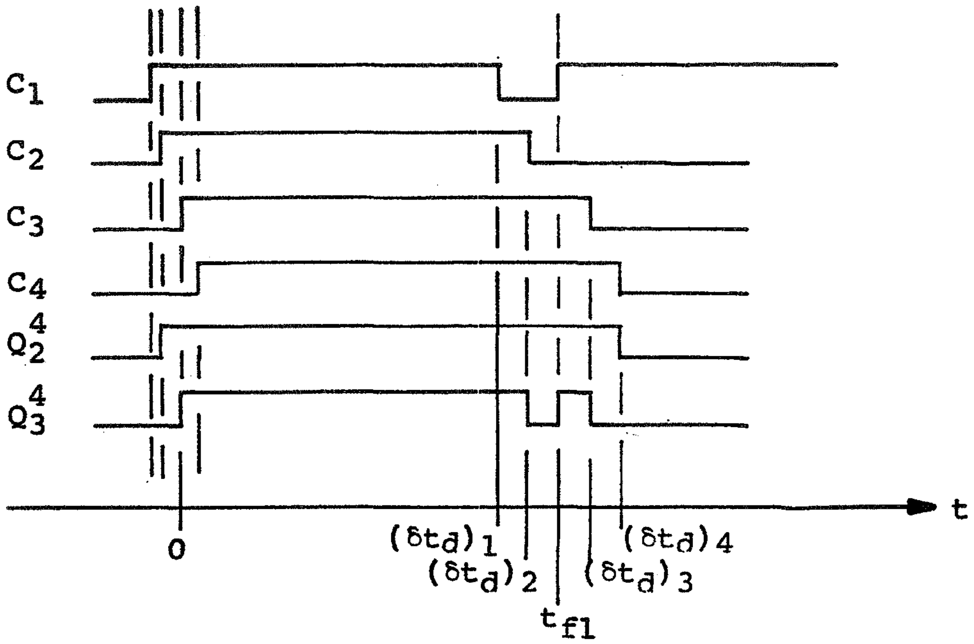


Fig. 3.12 Case 1

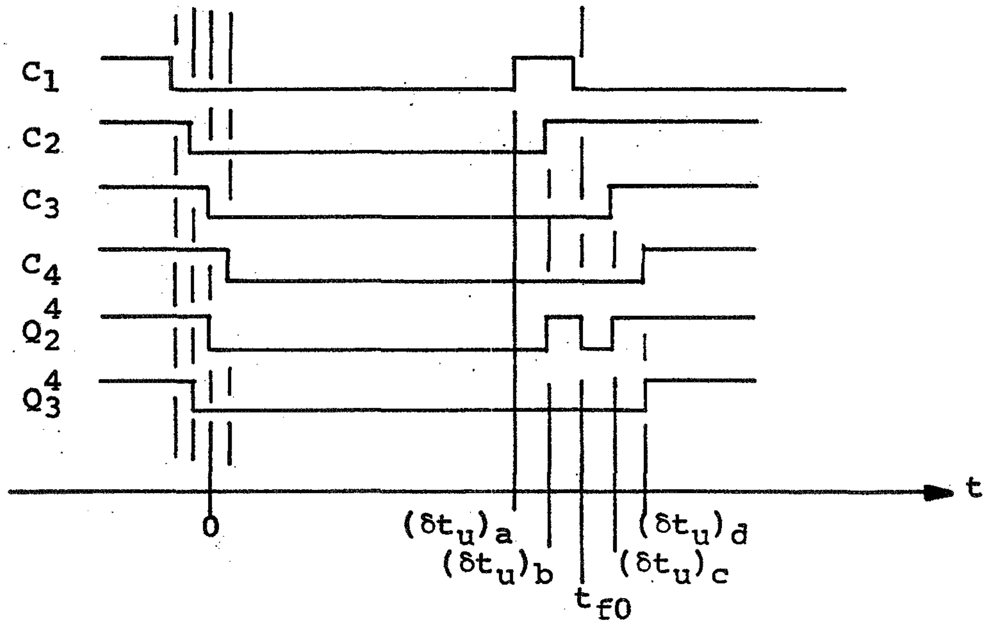


Fig. 3.13 Case 2

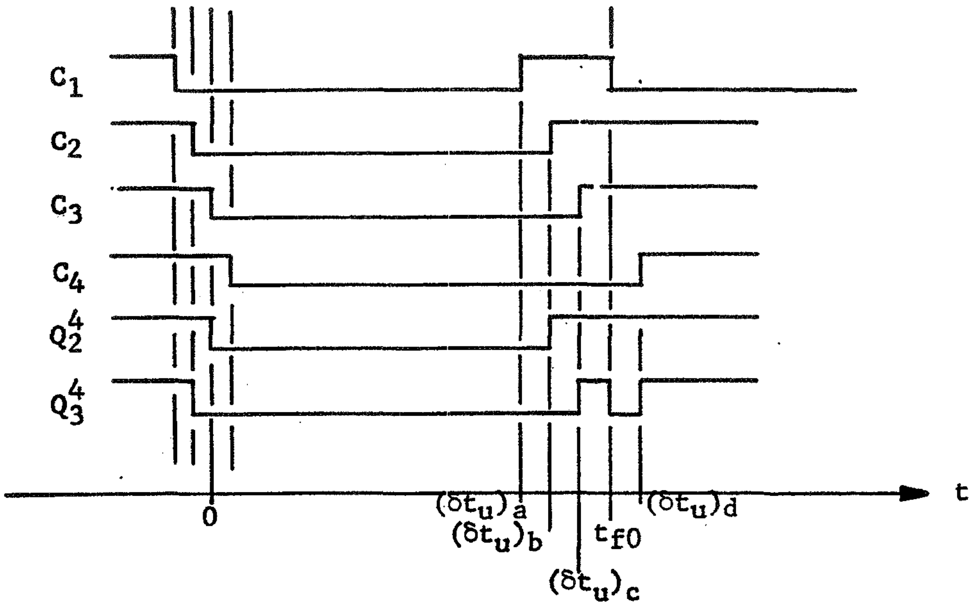


Fig. 3.14 Case 3

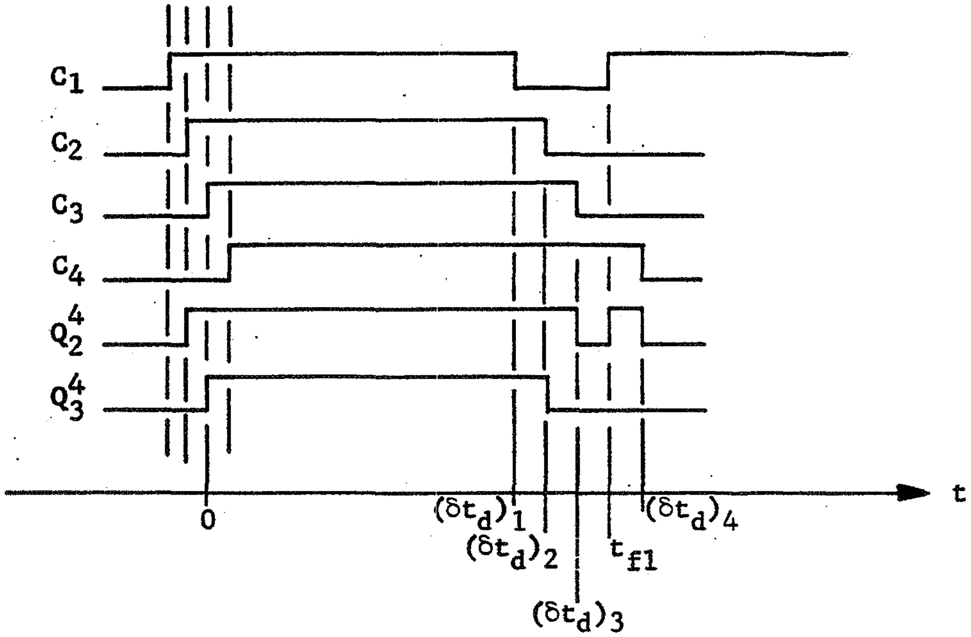


Fig. 3.15 Case 4

$$H_{2i} \rightarrow 1 \text{ at } (\delta t_d)_2 + 3\Delta \tag{3.81}$$

$$H_{2i} \rightarrow 0 \text{ at } t_{f1} + 3\Delta \tag{3.82}$$

$$H_{2i} \rightarrow 1 \text{ at } (\delta t_d)_3 + 3\Delta \tag{3.83}$$

$$A_{3i} \rightarrow 1 \text{ at } (\delta t_d)_2 + 4\Delta \tag{3.84}$$

$$A_{3i} \rightarrow 0 \text{ at } t_{f1} + 4\Delta \tag{3.85}$$

$$A_{3i} \rightarrow 1 \text{ at } (\delta t_d)_3 + 4\Delta \tag{3.86}$$

$$D_{2i} \rightarrow 0 \text{ at } (\delta t_d)_2 + 2\Delta + \Delta t \tag{3.87}$$

$$D_{2i} \rightarrow 1 \text{ at } t_{f1} + 2\Delta + \Delta t \tag{3.88}$$

$$D_{2i} \rightarrow 0 \text{ at } (\delta t_d)_3 + 2\Delta + \Delta t \tag{3.89}$$

$$A_{4i} \rightarrow 0 \text{ at } (\delta t_d)_2 + 3\Delta + \Delta t \tag{3.90}$$

$$A_{4i} \rightarrow 1 \text{ at } t_{f1} + 3\Delta + \Delta t \tag{3.91}$$

$$A_{4i} \rightarrow 0 \text{ at } (\delta t_d)_3 + 3\Delta + \Delta t \tag{3.92}$$

If transition (3.90) occurs while A_{3i} is at logic-0 (see transitions (3.84) through (3.86)) then C_i is driven to logic-0 at about the same time that it is being driven to logic-1 by the propagation of the signal: $Q_2^4 \rightarrow 0$; in order to avoid this situation, set:

$$[(\delta t_d)_2 + 3\Delta + \Delta t] - [(\delta t_d)_3 + 4\Delta] > 0$$

or,

$$(\delta t_d)_3 - (\delta t_d)_2 < \Delta t - \Delta \tag{3.93}$$

As previously stated, we may define the tolerances of (δt) as follows:

$$(\delta t)_{nom.}(1-x) < (\delta t) < (\delta t)_{nom.}(1+x) \tag{3.94}$$

Combining eqns. (3.93) and (3.94) in a worst-case condition:

$$(\delta t)_{nom}\ (1+x) - (\delta t)_{nom}\ (1-x) < \Delta t - \Delta$$

or

$$x < \frac{\Delta t - \Delta}{2\ (\delta t)_{nom}} \qquad (3.95)$$

and since $(\delta t)_{nom} = 6\Delta + \Delta t$,

$$x < \frac{\Delta t - \Delta}{2\ \Delta t + 12\Delta} \qquad (3.96)$$

If the system is to be run at maximum frequency, $\Delta t = 8\Delta$, and $x < \frac{1}{4}$. It is seen that there is a tradeoff between the tolerance of components and the frequency.

<u>Case 2.</u> In Fig. 3.13 C_1 fails to logic-0 within the region: $(\delta t_u)_b < t_{f0} < (\delta t_u)_c$; an analysis similar to that of Case 1 yields:

$$(\delta t_u)_c - (\delta t_u)_b < \Delta t + \Delta \qquad (3.97)$$

Here the requirement on x for successful operation is:

$$x < \frac{\Delta t + \Delta}{2\ \Delta t + 12\Delta} \qquad (3.98)$$

This is not as stringent a requirement as Case 1 provides.

<u>Case 3.</u> In Fig. 3.14 C_1 fails to logic-0 within the region: $(\delta t_u)_c < t_{f0} < (\delta t_u)_d$. It can be shown (in a manner similar to Case 4) that M_{4i} may go to logic-0 and back to logic-1, within a time on the order of one propagation delay, Δ, this transition occurring shortly before C_i is driven to logic-0 by the event: $Q_3^4 \rightarrow 1$ at $t = (\delta t_u)_d$; as a result a glitch may occur

at the output of N_{3i}, but not within C_i. In Case 4, however, it is shown that a glitch may occur in C_i.

Case 4. In Fig. 3.15 C_1 fails to logic-1 within the region: $(\delta t_d)_3 < t_{f1} < (\delta t_d)_4$. The analysis of this failure-mode follows:

As a result of $Q_2^4 \rightarrow 0$ at $(\delta t_d)_3$, M_{1i} is driven to logic-1. If when $Q_2^4 \rightarrow 1$ at t_{f1}, M_{1i} is at logic-1, A_{6i} may go to 1, driving M_{3i} to logic-0. The length of time that M_{3i} is at logic-0 is $\leq$ $(\delta t_d)_4 - t_{f1}$; as a result C_i may go to logic-1 at about 3Δ after it had gone to logic-0, or a pulse of duration approximately equal to Δ may occur in C_i, or C_i may be unaffected, yielding a loss of synchronism amongst the clock elements. Also the extra pulse in Q_2^4 may cause a pulse to occur in C_i just before C_i is set at logic-1.

Given that C_1 fails to logic-level-1, the probability that system operation is affected as described in Case 4 is the probability the t_{f1} will occur between $(\delta t_d)_3$ and $(\delta t_d)_4$, or:

$$P_{sf} = \frac{(\delta t_d)_4 - (\delta t_d)_3}{(\delta t_d)_3 + (\delta t_u)_c} \qquad (3.99)$$

where the subscript sf stands for system failure. So the smaller the tolerance on (δt_d), the smaller is the probability of a system failure being induced by the failure of a clock element to logic-1. It should be noted, however, that if a clock element exhibits a failure-mode of random oscillation, that, over an extended period of operation, P_{sf} approaches unity. Therefore in order to assure single-fault-tolerance, the circuit of Fig. 3.8 must be modified. Figure 3.16 is the suggested redesign of a clock element. The additional circuitry increases the values

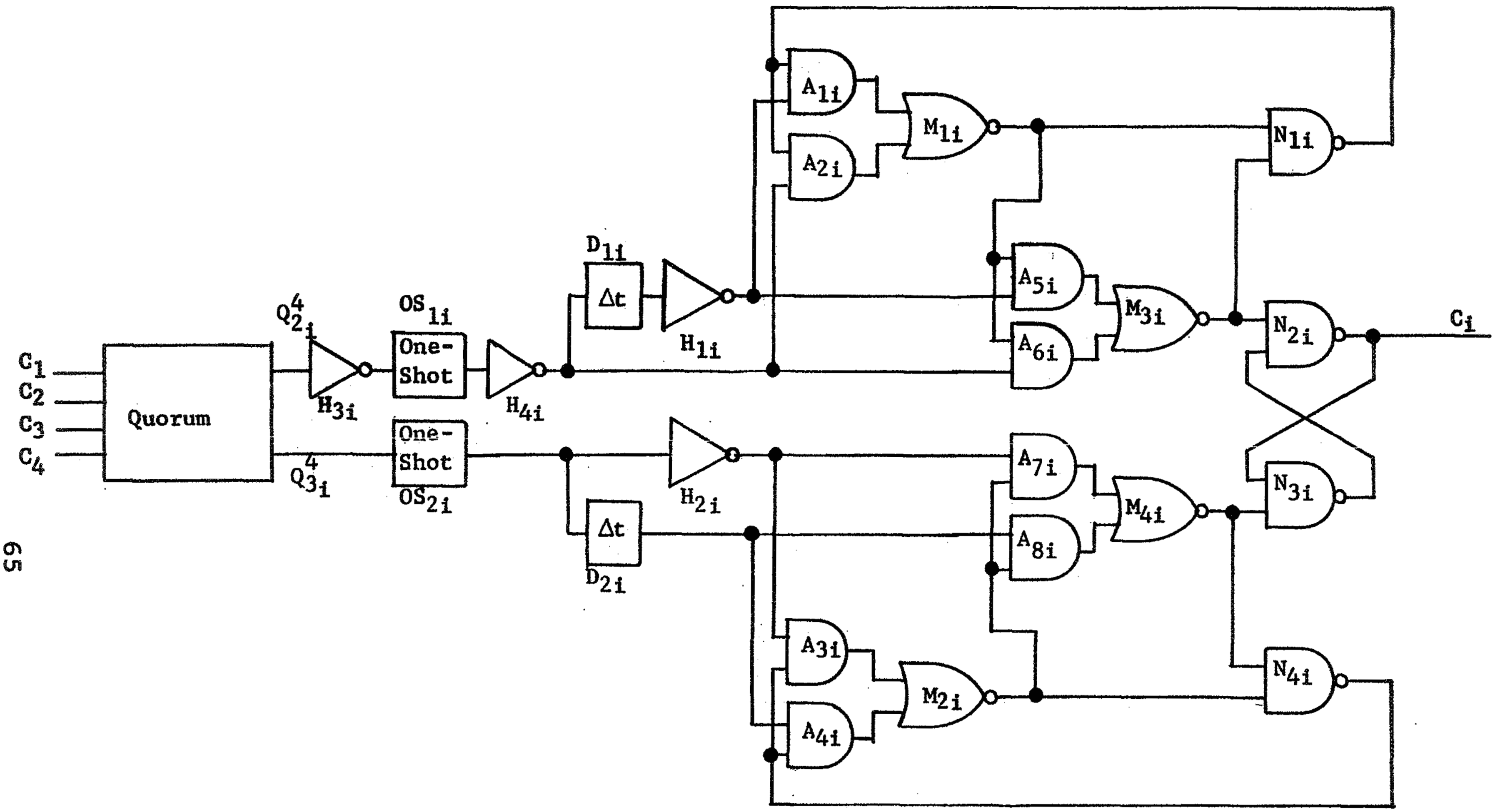


Fig. 3.16 Revised Daly-McKenna Clock

of $(\delta t_d)_{nom}$ and $(\delta t_u)_{nom}$, as follows:

$$(\delta t_u)_{nom} = 9\Delta + \Delta t \tag{3.100}$$

$$(\delta t_d)_{nom} = 7\Delta + \Delta t \tag{3.101}$$

thereby decreasing the duty-cycle slightly. The values of the necessary one-shot pulse widths are dependent on the desired operating frequency of the system (and therefore are dependent on the value of Δt). Figures 3.17 through 3.20 illustrate how the additional circuitry filters out the extraneous pulses of Q^4_{2i} or Q^4_{3i}; the pulse-width of OS_{1i} is $[(\delta t_u)_{nom} + \frac{1}{2}(\delta t_d)_{nom}]$, and the pulse-width of OS_{2i} is $[(\delta t_d)_{nom} + \frac{1}{2}(\delta t_u)_{nom}]$.

As stated earlier all clock lines are distributed to each synchronous module in the computing system. If the module transducer were simply a two out of three voter, the occurrence of a failure could cause a glitch in the output of the transducer. The subsequent slivering which may occur could destroy the synchronism of the system. Therefore the module transducer must filter out the extraneous pulses. Figure 3.21 is the design of a module transducer which may be used in conjunction with the Revised Daly-McKenna Clock. The pulse width of the one-shot is: $[(\delta t_u)_{nom} + \frac{1}{2}(\delta t_d)_{nom}]$. Figure 3.22 illustrates the operation of the module transducer. Note that although the design of the single-fault-tolerant clock requires four clock elements, only three need to be examined to extract a reliable clock. For the sake of uniformity of presentation it has been assumed that all clock lines are distributed. In the event of the development of a reconfigurable voter (2/3 voting with replacement), the implementation of such in place of the simple majority voter in each

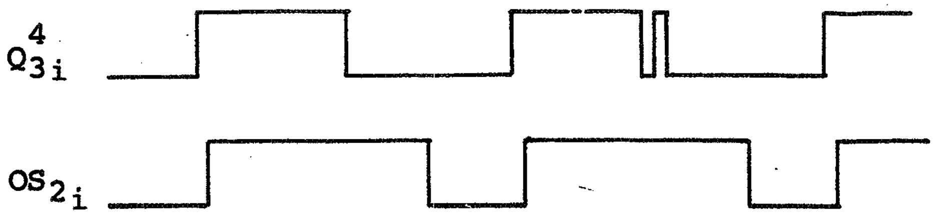


Fig. 3.17 Case 1

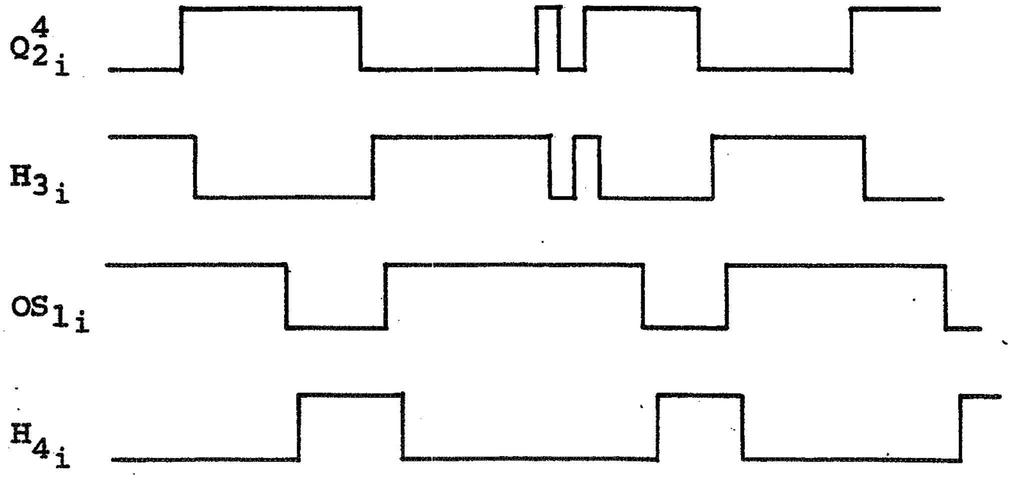


Fig. 3.18 Case 2

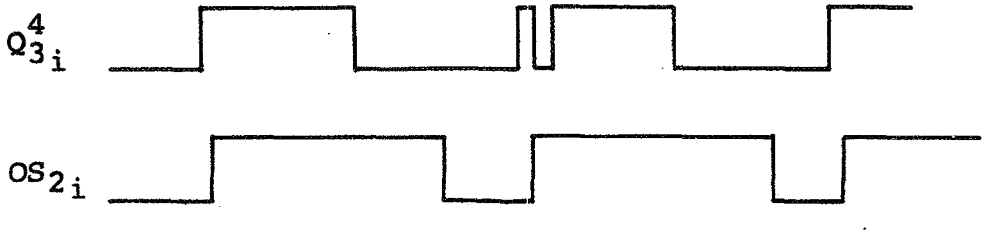


Fig. 3.19 Case 3

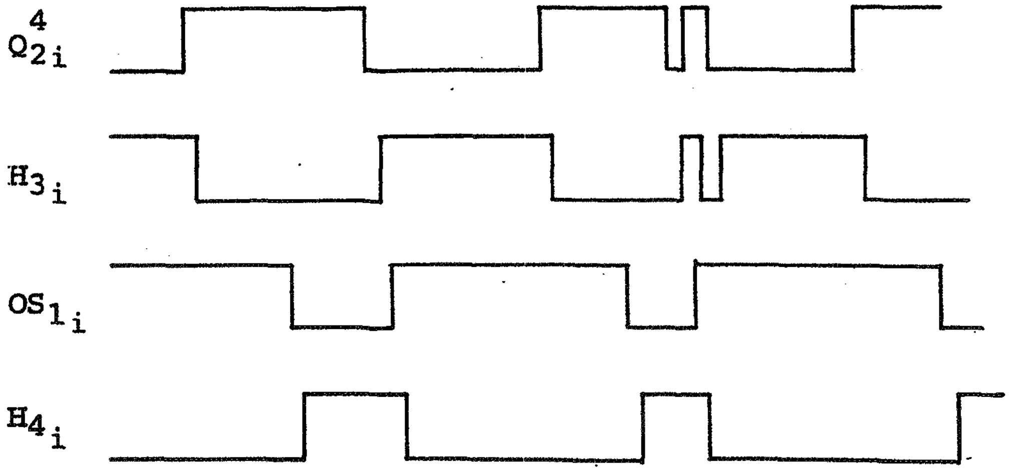


Fig. 3.20 Case 4

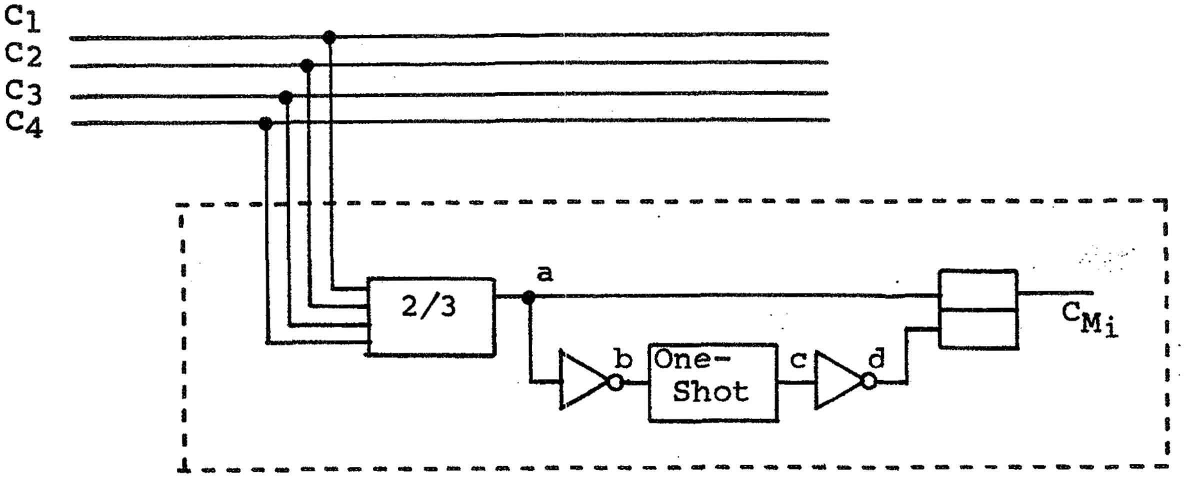


Fig. 3.21 Module Transducer for use with the Revised Daly-McKenna Clock

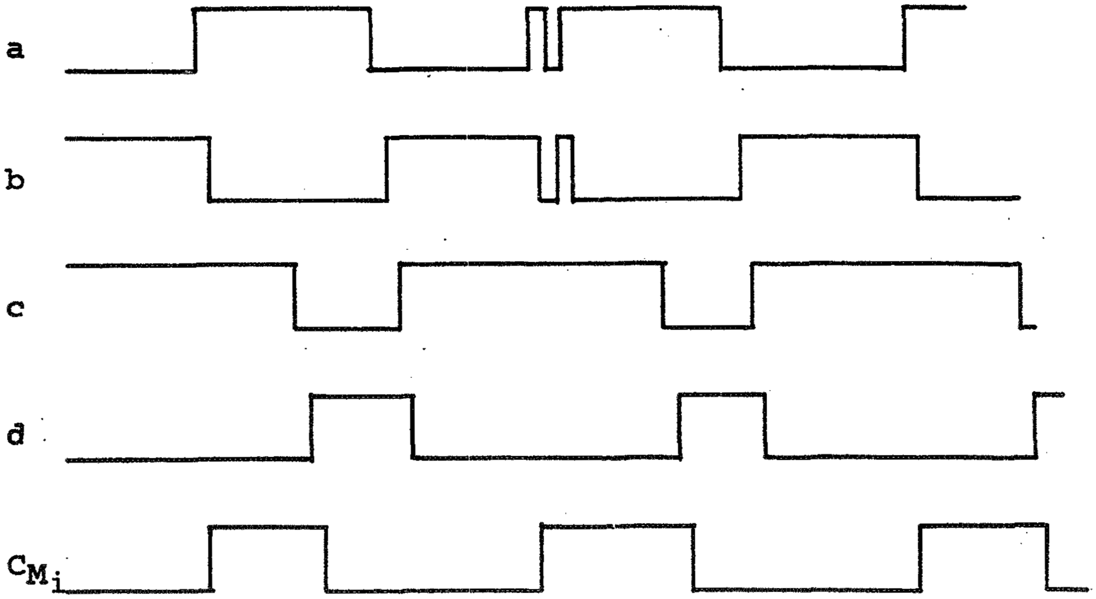


Fig. 3.22 Module Transducer Operation

module transducer could increase the reliability of the clocking system.

For n-fault-tolerance, the number of required clock elements is 3n + 1, as demonstrated by Daly and McKenna. Within the module transducer, the voter increases in complexity but the remainder of the circuitry remains the same. On the clock system level the Daly-McKenna Clock is more costly than the method of clocking examined in Section 3.3, but on the module level it is much less costly.

3.4.2 Current Concept

In October, 1971, McKenna designed a single-fault-tolerant clock for use in the prototype fault-tolerant multiprocessor (CERBERUS) currently being built at the C.S. Draper Laboratory. The design is shown in Fig. 3.23. The clock has been built, with medium speed TTL technology, and has been demonstrated to survive the imposition of single faults. The frequency of the clock now in existence is about 0.7 MHz.

The operating principles of this clock are similar to those of the first concept. It should be noted that here, each clock element will oscillate at its own chosen frequency if separated from the others. When the system is interconnected, however, each element will conform to one common frequency. As will be demonstrated, the system behaves as follows: C_i, after a short delay, is set to logic-1 by the occurrence of $Q_2^4 \rightarrow 1$, or after a much longer delay, by $Q_2^4 \rightarrow 0$; C_i is reset to logic-0, after a short delay, by $Q_3^4 \rightarrow 0$, or after a much longer delay, by $Q_3^4 \rightarrow 1$.

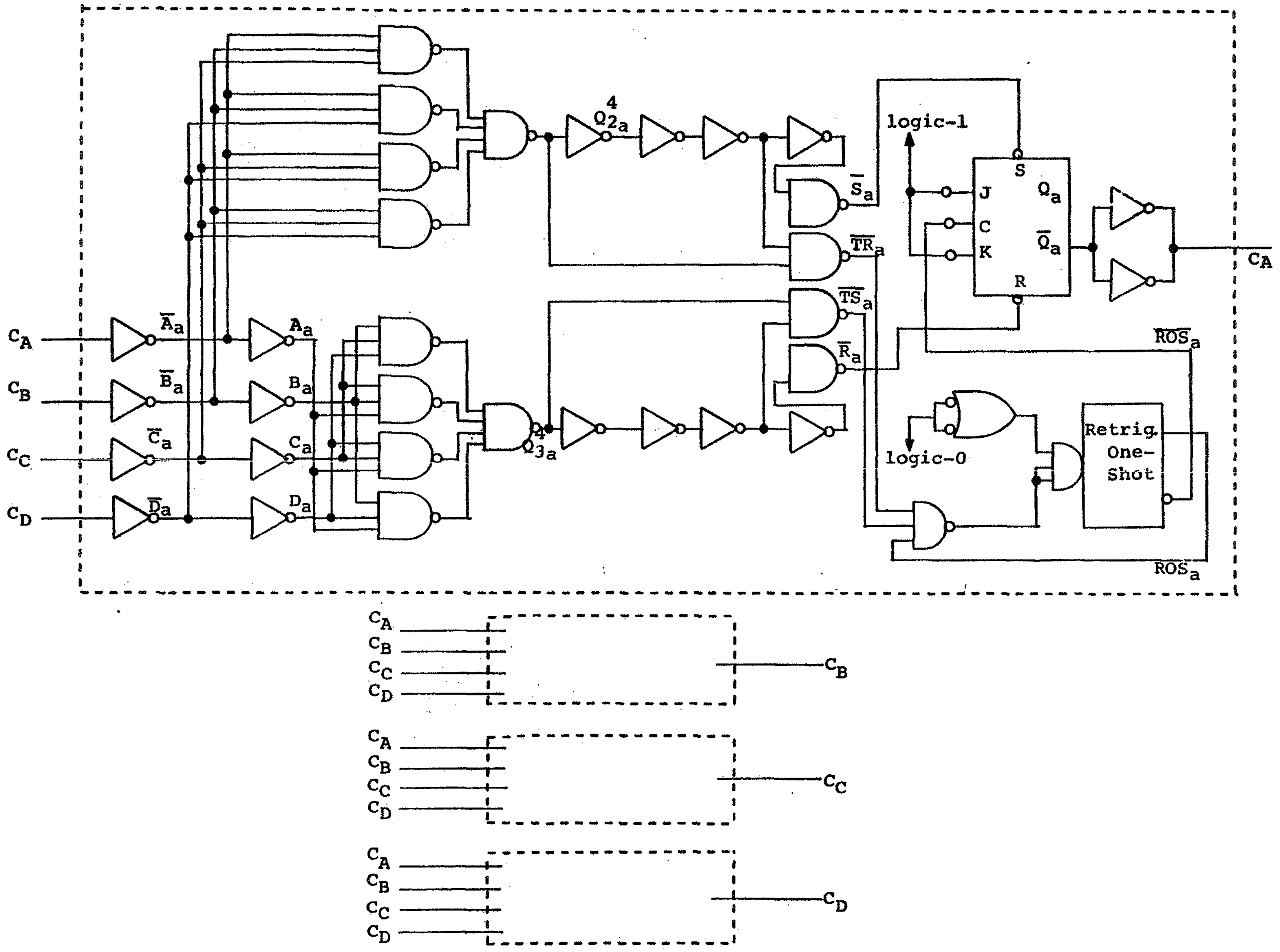


Fig. 3.23 McKenna Clock

The clock is self-starting; remember that the first concept (Fig. 3.8) requires additional logic to induce free-running oscillation. Some explanation of the clock element structure is called for: the pair of parallel inverters at the $\overline{Q}$ output of the J-K flip-flop are there to increase the fan-out capability; similarly the eight inverters at the input of each clock element serve to overcome fan-out limitations; as can be seen, $(Q_2^4)_i$ and $(Q_3^4)_i$ are each implemented through four levels of gating; the strings of inverters, to which $(Q_2^4)_i$ and $(Q_3^4)_i$ are applied, serve to time certain key signals; and the peculiar configuration of the region around the retriggerable-one-shot (see Fig. 3.24) is the internal configuration of the circuit element used.

As in the first concept, all clock lines are distributed to each synchronous module within the system. Simple two-out-of-three majority voting within the module is <u>not</u> sufficient, but, as before, the module transducer illustrated in Fig. 3.21 may be used.

For purposes of analysis of the circuit, the following simplifying assumptions are made:

(1) An element's propagation delay is the same for a

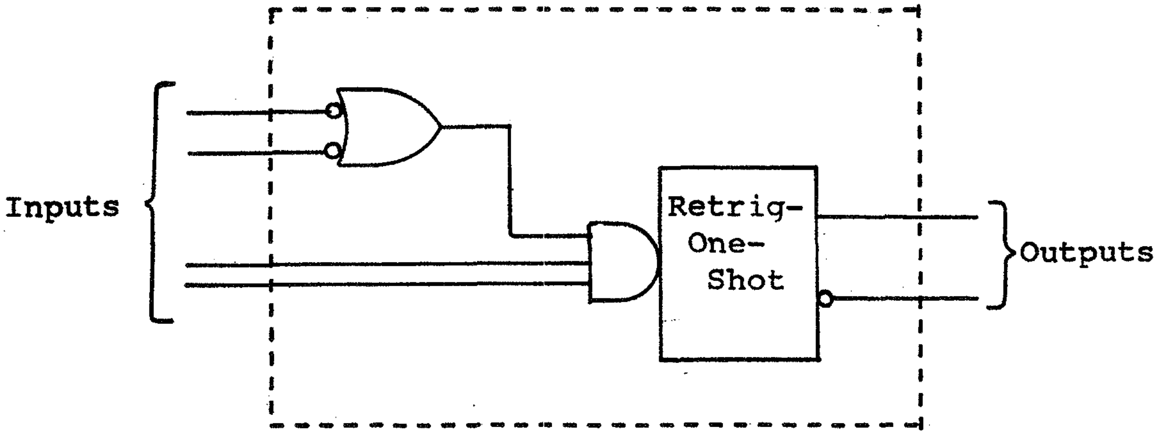


Fig. 3.24 Retriggerable One-Shot used in McKenna Clock

logic-level-1 to 0 transition as for a 0-to-1 transition,

(2) the delay associated with any inverter or NAND gate is equal to Δ,

(3) the delay associated with a transition caused by clocking the J-K flip-flop is 2Δ,

(4) the delay associated with a transition caused by an applied zero to the SET or RESET input of the J-K flip-flop is 3Δ,

(5) the one-shot triggers when its input AND gate experiences a transition from logic-level-0 to 1; the delay between the input of the AND gate and the output of the one-shot is 2Δ; after the last triggering input to the AND gate, the one-shot will go to logic-level-0 at a time Δt later.

For now, assume that the clock elements are oscillating in synchronism; the clock's self-starting capability may be examined separately. As in the manner of the analysis of the first concept, assume that at time $t=t_A$, $C_A \rightarrow 1$; $t=t_B$, $C_B \rightarrow 1$; $t=t_C$, $C_C \rightarrow 1$; $t=t_D$, $C_D \rightarrow 1$, where $t_D > t_C > t_B > t_A$. Assume that at an initial time of observation, $(t_A - \epsilon)$, all propagation, within each clock element, caused by the previous transition, $C_i \rightarrow 0$, has ceased; then:

Initially: $C_i = 0$

$(Q_2^4)_i = 0$; $(Q_3^4)_i = 0$

$\overline{S}_i = 1$; $\overline{R}_i = 1$

$\overline{TR}_i = 1$; $\overline{TS}_i = 1$

ROS_i is indeterminable from a static analysis.

Given the initial state and the assumed progression of events, the timing analysis is as follows:

$$C_A \rightarrow 1 \text{ at } t = t_A \tag{3.102}$$

$$C_B \rightarrow 1 \text{ at } t_B \tag{3.103}$$

$$(Q_2^4)_i \rightarrow 1 \text{ at } t_B + 4\Delta \tag{3.104}$$

$$\overline{S}_i \rightarrow 0 \text{ at } t_B + 5\Delta \tag{3.105}$$

$$\overline{S}_i \rightarrow 1 \text{ at } t_B + 8\Delta \tag{3.106}$$

$$C_C \rightarrow 1 \text{ at } t_C \tag{3.107}$$

where, $t_B < t_C < t_B + 9\Delta$

$$C_D \rightarrow 1 \text{ at } t_D \tag{3.108}$$

where, $t_C < t_D < t_B + 9\Delta$

$$(Q_3^4)_i \rightarrow 1 \text{ at } t_C + 4\Delta \tag{3.109}$$

$$\overline{TS}_i \rightarrow 0 \text{ at } t_C + 5\Delta \tag{3.110}$$

$$\overline{TS}_i \rightarrow 1 \text{ at } t_C + 8\Delta \tag{3.111}$$

ROS_i is triggered either at $t = t_C + 6\Delta$ or within $t_C + 3\Delta < t < t_C + 6\Delta$ by its own expiration ($ROS_i \rightarrow 0$); therefore ROS_i, $i = A,B,C$, is triggered at $t = t_C + 6\Delta$; ROS_D is triggered within $t_C + 3\Delta < t < t_C + 6\Delta$

$$ROS_D \rightarrow 0 \text{ within } t_C + 3\Delta + \Delta t < t < t_C + 6\Delta + \Delta t \tag{3.112}$$

$$ROS_{A,B,C} \rightarrow 0 \text{ at } t_C + 6\Delta + \Delta t \tag{3.113}$$

$$C_D \rightarrow 0 \text{ within } t_C + 6\Delta + \Delta t < t < t_C + 9\Delta + \Delta t \qquad (3.114)$$

$$C_{A,B,C} \rightarrow 0 \text{ at } t_C + 9\Delta + \Delta t \qquad (3.115)$$

ROS_D is triggered within $t_C + 4\Delta + \Delta t < t < t_C + 7\Delta + \Delta t$, and ROS_i, $i = A,B,C$, is triggered at $t = t_C + 7\Delta + \Delta t$.

$$(Q_2^4)_i \rightarrow 0 \text{ at } t_C + 13\Delta + \Delta t \qquad (3.116)$$

$$(Q_3^4)_i \rightarrow 0 \text{ at } t_C + 13\Delta + \Delta t \qquad (3.117)$$

$$\overline{TR}_i \rightarrow 0 \text{ at } t_C + 13\Delta + \Delta t \qquad (3.118)$$

$$\overline{TR}_i \rightarrow 1 \text{ at } t_C + 16\Delta + \Delta t \qquad (3.119)$$

$$\overline{R}_i \rightarrow 0 \text{ at } t_C + 15\Delta + \Delta t \qquad (3.120)$$

$$\overline{R}_i \rightarrow 1 \text{ at } t_C + 18\Delta + \Delta t \qquad (3.121)$$

ROS_i is triggered at $t_C + 14\Delta + \Delta t$.

$$ROS_i \rightarrow 0 \text{ at } t_C + 14\Delta + 2\Delta t \qquad (3.122)$$

$$C_i \rightarrow 1 \text{ at } t_0 \equiv t_C + 17\Delta + 2\Delta t \qquad (3.123)$$

$$(Q_2^4)_i \rightarrow 1 \text{ at } t_0 + 4\Delta \qquad (3.124)$$

$$(Q_3^4)_i \rightarrow 1 \text{ at } t_0 + 4\Delta \qquad (3.125)$$

$$\overline{TS}_i \rightarrow 0 \text{ at } t_0 + 5\Delta \qquad (3.126)$$

$$\overline{TS}_i \rightarrow 1 \text{ at } t_0 + 8\Delta \qquad (3.127)$$

ROS_i is triggered at $t_0 + 6\Delta$.

$$ROS_i \rightarrow 0 \text{ at } t_0 + 6\Delta + \Delta t \qquad (3.128)$$

$$C_i \rightarrow 0 \text{ at } t_0 + 9\Delta + \Delta t \qquad (3.129)$$

$$(Q_2^4)_i \rightarrow 0 \text{ at } t_0 + 13\Delta + \Delta t \tag{3.130}$$

$$(Q_3^4)_i \rightarrow 0 \text{ at } t_0 + 13\Delta + \Delta t \tag{3.131}$$

Before proceeding any further with the analysis of the current concept of the McKenna clock, a major fault in the design should be pointed out: $Q_2^4 \rightarrow 1$ forces $C_i \rightarrow 1$ (if this transition has not already occurred) at a time 9Δ later and $Q_3^4 \rightarrow 0$ forces $C_i \rightarrow 0$ at a time 10Δ later, but in the clock element in which a transition of C_i occurs as a result of $Q_2^4 \rightarrow 1$ or $Q_3^4 \rightarrow 0$, the retriggerable one-shot expires, thereby clocking the flip-flop and forcing an unintended change of state; the occurrence of these glitches in two clock outputs could induce a loss of synchronization within the system. A possibility for correcting the design is: remove $\overline{TR}$ and $\overline{TS}$, retrigger the one-shot with $\overline{S}$ or $\overline{R}$, and insert delays between $\overline{S}$ and the SET input of the flip-flop and between $\overline{R}$ and the RESET input of the flip-flop, such that the one-shot cannot expire in the half-cycle in which C_i has been changed by $\overline{S}$ or $\overline{R}$. To serve such a purpose delay times of 6Δ are sufficient.

Here, as in the first concept, the occurrence of a failure may introduce a glitch in Q_2^4 or Q_3^4, subsequent slivering within the clock elements, and possible loss of synchronization. The same modification is recommended here as was introduced for the first concept. Figure 3.25 illustrates the author's revised design of the McKenna clock.

Assume that the rise-time of the one-shots used for filtering Q_2^4 and Q_3^4 is Δ. Then when $Q_3^4 \rightarrow 1$, $C_i \rightarrow 0$, $11\Delta + \Delta t$ later; when $Q_2^4 \rightarrow 0$, $C_i \rightarrow 1$, $10\Delta + \Delta t$ later. The period of the clock is:

$$T = 21\Delta + 2\Delta t$$

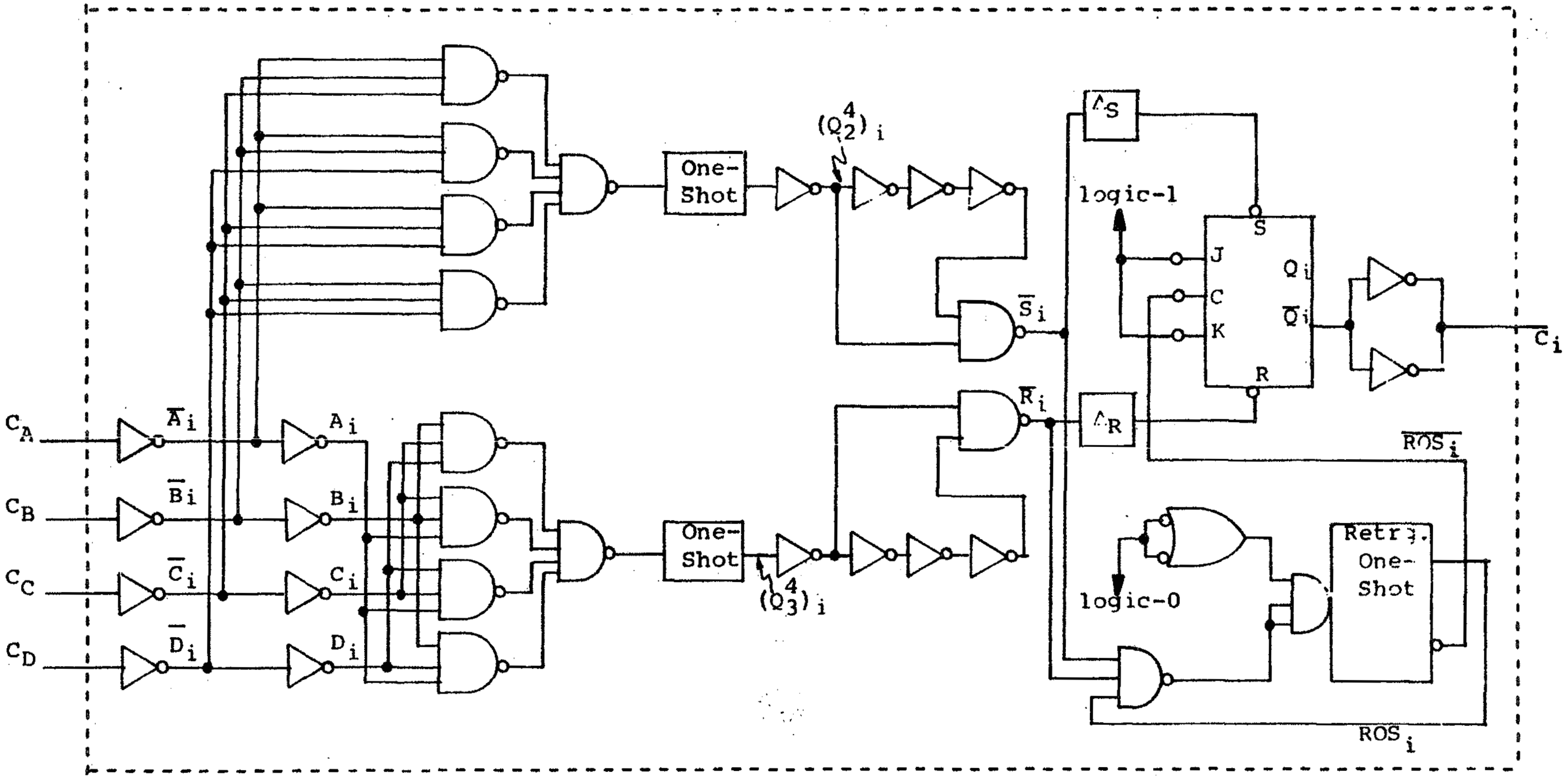


Fig. 3.25 Revised McKenna Clock

The expiration of the one-shot triggers the one-shot, Δ later; Δt must be such that the one-shot will not expire again before it is retriggered by $\overline{S}_i$ or $\overline{R}_i$. From time of expiration to retriggering by $Q_3^4 \rightarrow 1$ or $Q_2^4 \rightarrow 0$ (assuming perfect synchronism) the time which lapses is: 10Δ or 11Δ. In order to allow for the results of differences in propagation delay amongst clock elements (e.g., internal delay, phase differences amongst C_i), Δt should be greater than or equal to 20Δ. Therefore:

$$f_{max} = \frac{1}{61\Delta}$$

and if medium-speed TTL technology is used ($\Delta \approx 12$ns), $f_{max} \approx 1.4$ MHz.

Retriggerable one-shots with timing adjustments are available; if these are used they may be adjusted after the clock is wired, thereby allowing the synchronous quality to be "peaked". Indeed the use of adjustable retriggerable-one-shots obviates concern with ΔT_u, ΔT_d (defined in Sec. 3.4.1), or frequency variation caused by a single failure. After the clock system has been peaked, deterioration of the synchronous quality is restricted to that introduced by component aging or environmental factors (such as temperature variations).

The retriggerable-one-shot is responsible for the self-starting property of the clock. At power-on the flip-flops come up in an arbitrary state. If ROS_i comes up logic-1, then it has just been triggered and will expire Δt later; if it comes up logic-0, it triggers itself and will expire Δt later. Within short order $Q_2^4 \rightarrow 0$ or $Q_3^4 \rightarrow 1$ occurs and the clock elements begin oscillating in synchronism.

Fault-tolerance is achieved in this design in the same manner as described in Section 3.4.1; if the clock is to be n-fault-tolerant, 3n+1 clock elements are required.

3.5 Speeding Up the McKenna Clock

3.5.1 Advantages of Greater Speed

The speed of a synchronous processor is directly related to the clock frequency. Current technology allows the design of a synchronous processor which runs at 20 MHz or more. The maximum frequency obtainable by the clock shown in Fig. 3.25 is, with TTL technology, 1.4 MHz; to run a processor at this rate can be highly inefficient. It is, therefore, desirable to have the capability of operating a fault-tolerant clock at a substantially greater speed.

3.5.2 Application of Advanced Device Technology

The numerical values of clock frequencies which have been derived in this thesis, have been derived assuming the usage of medium-speed TTL technology. But there is, today, higher speed technology available. In 1968, Motorola began marketing a logic family which offers 1 ns gate propagation delays (MECL III). The McKenna-type clock shown in Fig. 3.25, if implemented in MECL III, could operate at 16.4 MHz.

Certainly, with the development of higher speed technologies, proportionately higher speed fault-tolerant clocks may be built; it should be realized, however, that the availability of higher speed technologies will also allow the design of higher speed processors. As such it is desirable to design a fault-tolerant clock which can run at 20 MHz utilizing medium-speed TTL technology; then the implementation of both processor and clock with any higher speed technology will yield a proportionately higher speed computing system.

3.5.3 Revised Circuit

Simply by connecting $\overline{TR_i}$ to the SET input of the flip-flop and $\overline{TS_i}$ to the RESET input of the flip-flop, the frequency is tripled. The revised clock element is shown in Fig. 3.26. The basic operating principles are: $Q_2^4 \rightarrow 0$ induces $C_i \rightarrow 1$ (after a delay, 9Δ) and $Q_3^4 \rightarrow 1$ induces $C_i \rightarrow 0$ (after a delay, 10Δ). The period is given by:

$$T = 19\Delta$$

For medium-speed TTL technology, $f \approx 4.4$ MHz. The retrigerable-one-shot remains high during synchronous operation; it is needed here only for initialization of oscillation. In order to assure a start, only three of the four clock elements of a single-fault-tolerant clock need be provided with a retriggerable-one-shot and its associated circuitry. Consider such a design where for ROS_a, $\Delta t = 20\Delta$; ROS_b, $\Delta t = 35\Delta$; and ROS_c, $\Delta t = 55\Delta$. At power-on the flip-flops come up in an arbitrary state and the retriggerable-one-shots are triggered. Fig. 3.27 shows how the clock is started from each of the sixteen possible initial conditions. If one of the retriggerable-one-shots fails, a start is still assured.

The disadvantages of this design stem from the circumstance that the frequency is dependent solely on gate delays. Because there are no devices having adjustable time delays within the clock elements, it is not simple to design the clock for a particular frequency; neither is it possible

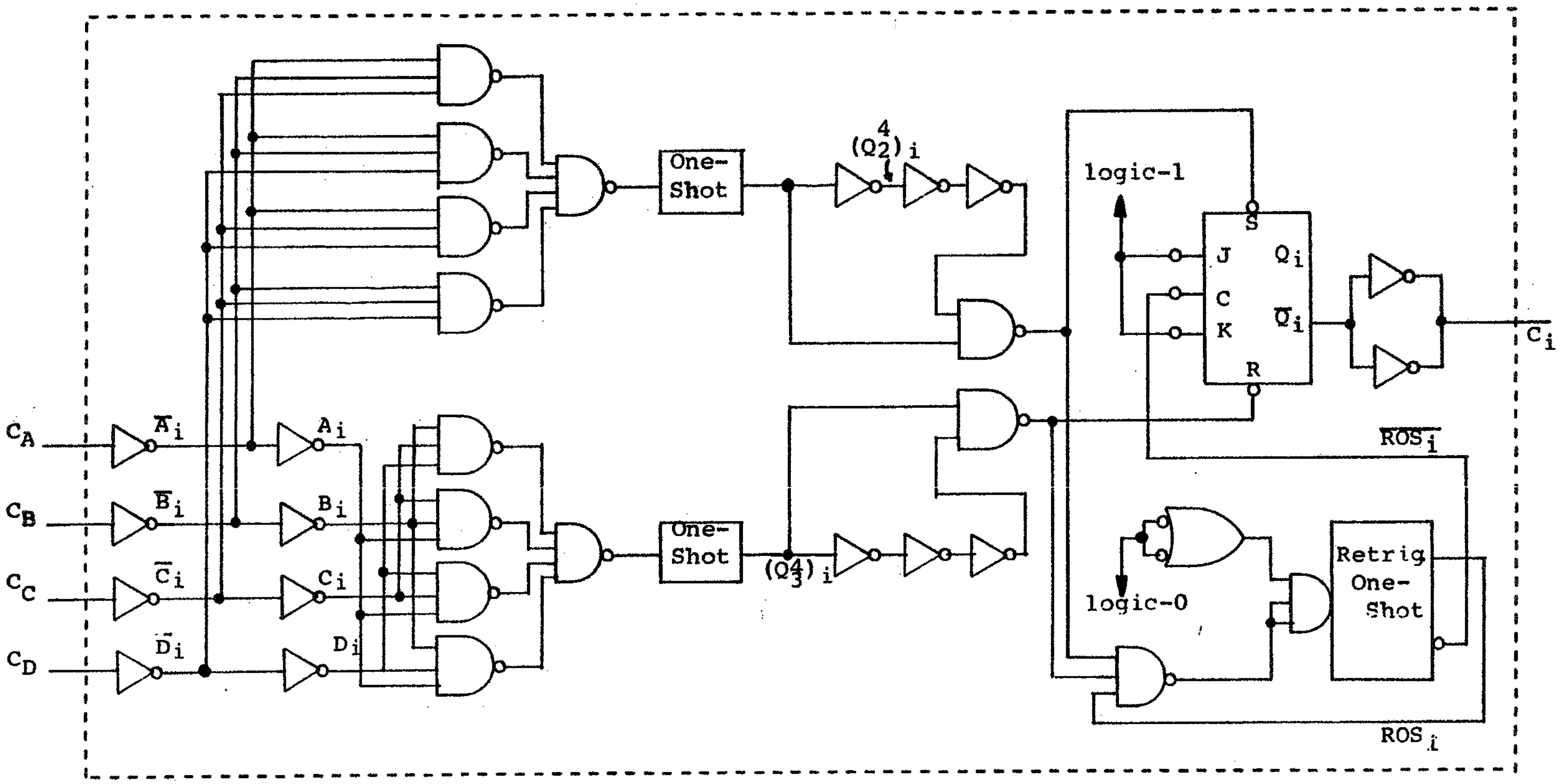


Fig. 3.26 Revised Circuit

t = 0				t = 23Δ	t = 38Δ	t = 44Δ	t = 58Δ	Synchronization has been achieved when:
C_A	C_B	C_C	C_D					
0	0	0	0	$C_A \to 1$	$C_B \to 1$	$Q_2^4 \to 0$		$C_i \to 1$ at 56Δ
0	0	0	1	$C_A \to 1$	$Q_3^4 \to 1$			$C_i \to 0$ at 51Δ
0	0	1	0	$C_A \to 1$	$Q_3^4 \to 1$			$C_i \to 0$ at 51Δ
0	0	1	1	$Q_3^4 \to 1$				$C_i \to 0$ at 36Δ
0	1	0	0	$C_A \to 1$	$Q_2^4 \to 0$			$C_i \to 1$ at 50Δ
0	1	0	1	$Q_3^4 \to 1$				$C_i \to 0$ at 36Δ
0	1	1	0	$Q_3^4 \to 1$				$C_i \to 0$ at 36Δ
0	1	1	1	$C_A \to 1$	$C_B \to 0$	$C_A \to 0$	$Q_2^4 \to 0$	$C_i \to 1$ at 70Δ
1	0	0	0	$C_A \to 0$	$C_B \to 1$	$C_A \to 1$	$Q_2^4 \to 1$	$C_i \to 0$ at 71Δ
1	0	0	1	$Q_2^4 \to 0$				$C_i \to 1$ at 35Δ
1	0	1	0	$Q_2^4 \to 0$				$C_i \to 1$ at 35Δ
1	0	1	1	$C_A \to 0$	$Q_3^4 \to 1$			$C_i \to 0$ at 51Δ
1	1	0	0	$Q_2^4 \to 0$				$C_i \to 1$ at 35Δ
1	1	0	1	$C_A \to 0$	$Q_2^4 \to 0$			$C_i \to 1$ at 50Δ
1	1	1	0	$C_A \to 0$	$Q_2^4 \to 0$			$C_i \to 1$ at 50Δ
1	1	1	1	$C_A \to 0$	$C_B \to 0$	$Q_3^4 \to 1$		$C_i \to 0$ at 57Δ

Fig. 3.27 Analysis of self-starting operation of revised circuit

to minimize the phase differences between clock elements through the method of peaking described in Section 3.4.2.

3.5.4 Increased Speed by Frequency Multiplication

The frequency of the distributed clock can be increased through use of the concept illustrated in Fig. 3.28 for single-fault-tolerance. C_A, C_B, C_C, and C_D are the outputs of a single-fault-tolerant McKenna-type Clock; these "low frequency" clocks are supplied to triplicated two-out-of-three voters, the outputs of which are each supplied to a frequency multiplier (x N). The outputs of the multipliers, C_1, C_2, and C_3, are distributed to the data management system.

Several methods of synthesizing the frequency multiplication come to mind; the first is illustrated in Fig. 3.29. The method of operation of Fig. 3.29 may be described as a burst generator; for each pulse of the input, the multiplier produces a "burst" of n pulses. The limitations of this circuit are determined by: frequency and frequency variation of the input, minimum pulse width producible by the one-shot, tolerance in the propagation delay of the one-shot, tolerance in the pulse width, minimum delay-times available, tolerances in delay-times, and the tolerance in the propagation delay of the OR gate. The limitations imposed by the tolerances may be minimied by careful component selection; nevertheless some tolerance must be allowed: ± 2% seems reasonable. Assume that a one-shot is available with a pulse width of 25 ns and a rise time of 10 ns. Assume that a tapped delay line is available with total delay 500 ns and taps available at 50 ns intervals (the Digital Equipment Corporation manufacturers just such a delay line. They claim a tolerance from the input to each

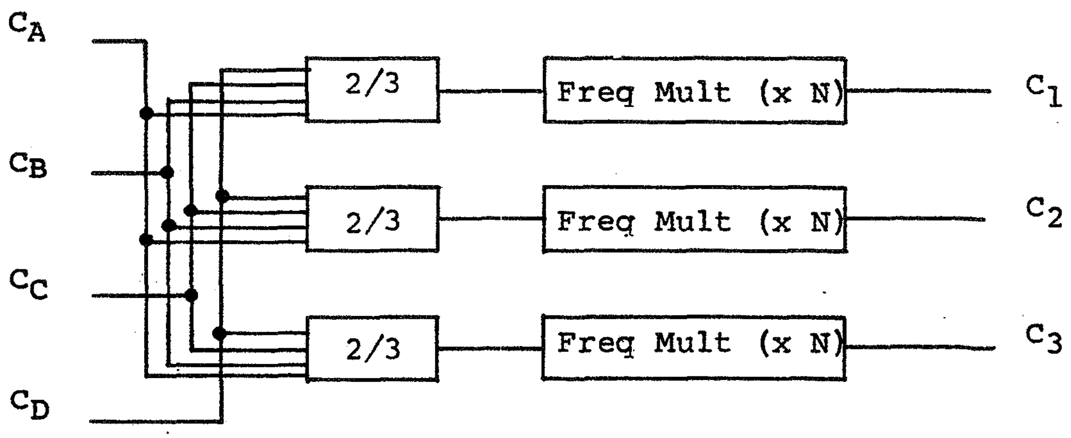


Fig. 3.28 Single-Fault-Tolerant Frequency Multiplication

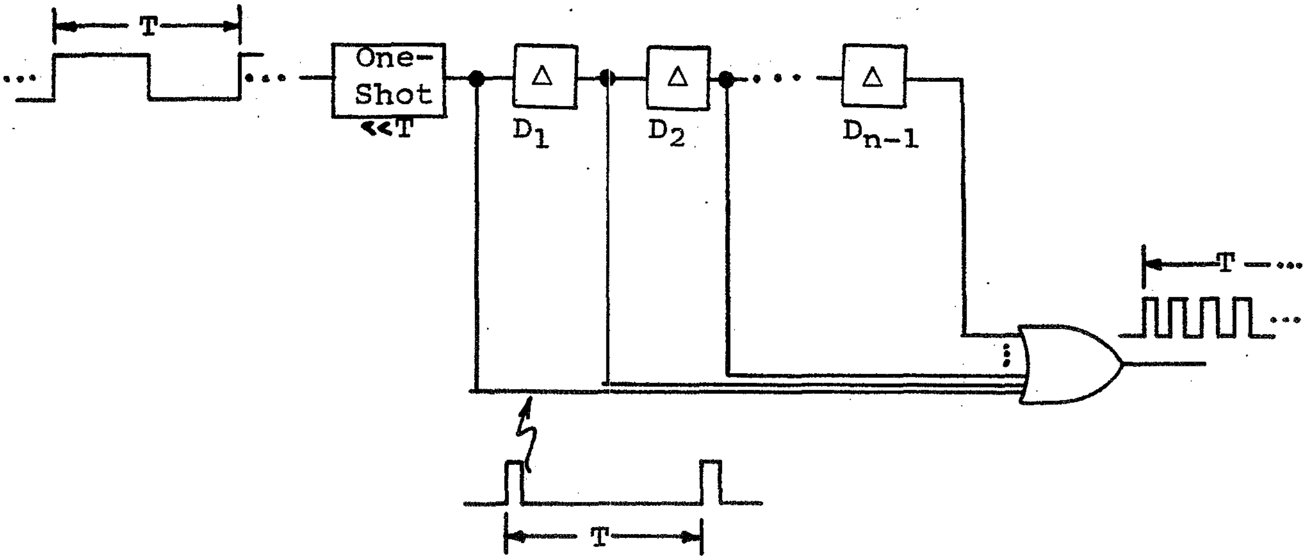


Fig. 3.29 Burst Generator

delay tap of ± 5%; hence that tolerance will be allowed for the delays considered here). There is a trade-off between the quality of synchronization at the outputs of the three frequency multipliers and the number of 50 ns delays used. Not including delays introduced by the gates, corresponding pulses passing through independent 500 ns delays may be separated, at the output, by as much as 50 ns (if 25 ns pulses are considered, the trailing-edge-to-leading-edge separation is 25 ns; synchronism has been destroyed). If 25 ns pulses pass through independent delays, and it is desired at the output to have a minimum overlap of 15 ns, then the maximum difference between delay-times is 10 ns and hence for an allowed ± 5% tolerance, the largest delay-time which may be used is 100 ns. Therefore, it is concluded that unless precision delay lines are made available, the frequency multiplier of Fig. 3.29 is restricted to producing an integral multiple of 3 or less if it is to produce a 20 MHz clock, synchronously with other similar frequency multipliers. The limitation on the multiplying capability of Fig. 3.29, requires that the input frequency be 6.7 MHz for the output frequency to be 20 MHz. As developed in Section 3.5.3, the operating frequency of the revised, and fastest, McKenna-type clock is 4.4 MHz for a medium-speed TTL implementation.

The difficulty with the multiplier of Fig. 3.29, suggests the design shown in Fig. 3.30. The voting between multiplying stages corrects for the phase differences amongst the outputs of the multipliers of the preceding stage. M_{1i}, M_{2i}, and M_{3i} are illustrated in Figs. 3.31, 3.32, and 3.33 respectively. In Fig. 3.31, the one-shot pulse width is 100 ns; in Fig. 3.32, 50 ns; in Fig. 3.33, 25 ns. Note that for a 20 MHz output only

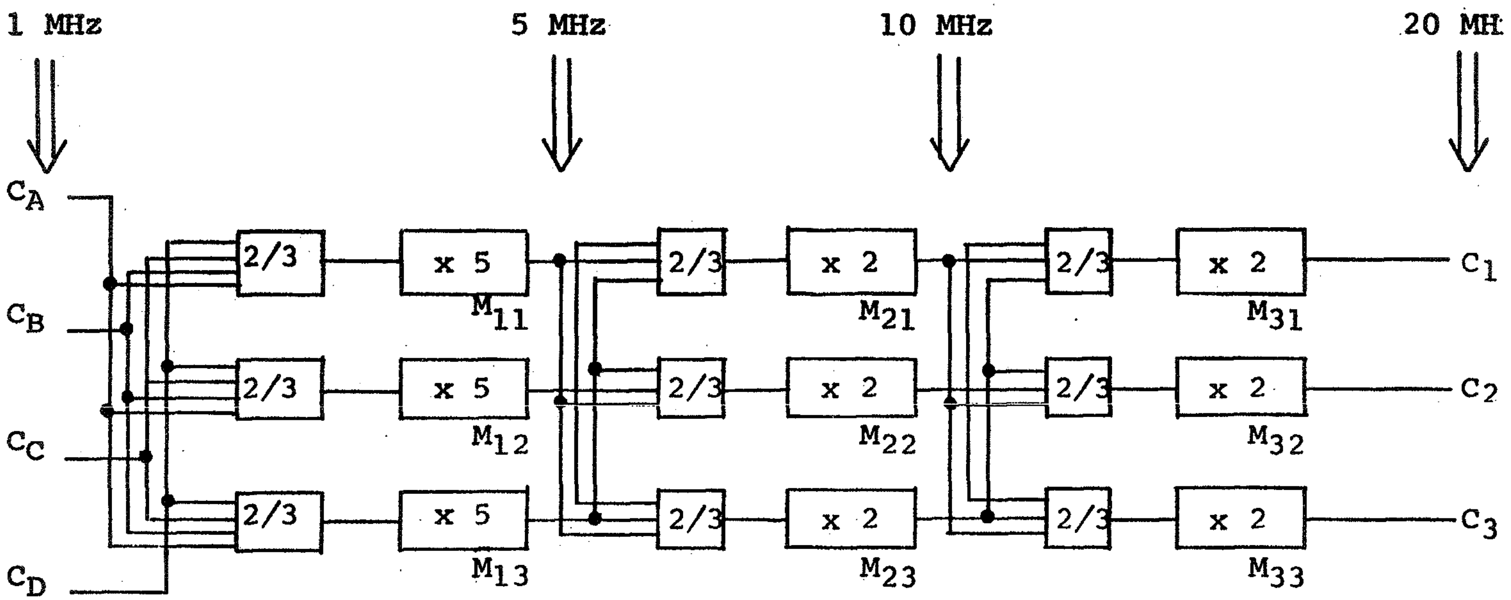
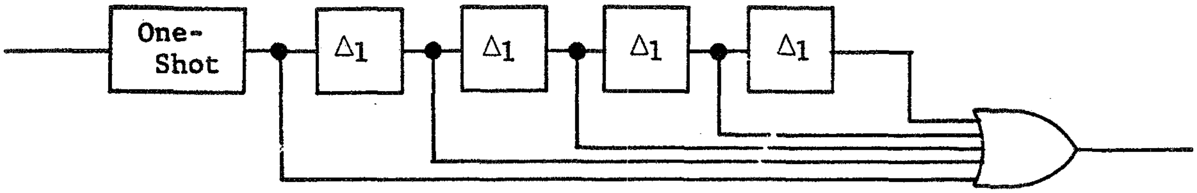


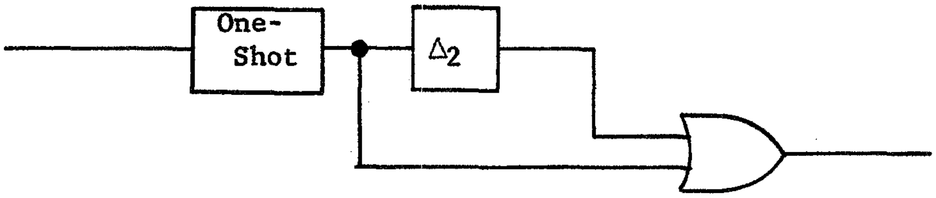
Fig. 3.30 Frequency Multiplication through use of cascaded burst generators



One-shot pulse width = 100 ns

Δ_1 = 200 ns

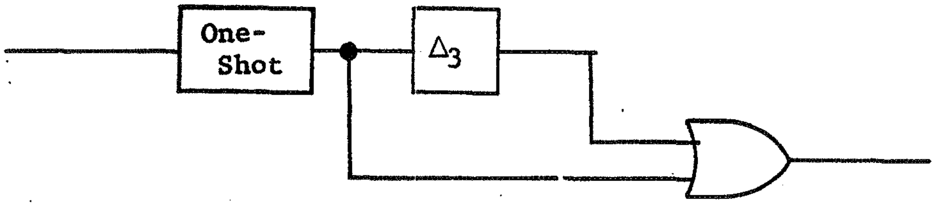
Fig. 3.31 M_{1i} of fig. 3.30



One-shot pulse width = 50 ns

Δ_2 = 100 ns

Fig. 3.32 M_{2i} of fig. 3.30



One-shot pulse width = 25 ns

Δ_3 = 50 ns

Fig. 3.33 M_{3i} of fig. 3.30

a 1 MHz input is required; hence the requirements placed on the design of the fault-tolerant clock may be lessened.

Another method of synthesizing the frequency multiplication is through use of a phase-locked loop, as illustrated in Fig. 3.34. The failure of a system which utilizes phase-locked loops for frequency multiplication, however, is that in the event of an input clock failure, and hence a subsequent change in input frequency, independent phase-locked loops may not maintain synchronism while their outputs adjust to the new frequency.

It is concluded that for speeding up the McKenna Clock (or any other slow clock) the cascaded burst generator design offers the most viable solution.

3.6 Methods of Synchronization Used in Pulse-Code Modulation

The synchronization of pulse-code modulation (PCM) networks has long been a subject of interest. The question of synchronizing switching centers, in addition to transmission links, arose in 1956, when the PCM telephone switching experiment, later named Essex, was planned.

Possible methods of synchronization as described by Mumford and Smith (Ref. 3), are as follows:

(1) Homochronous system. One station in the network has a master oscillator, and all the others are locked to it.

(2) Synchronous system. Each station is phase-locked to the average of several signals.

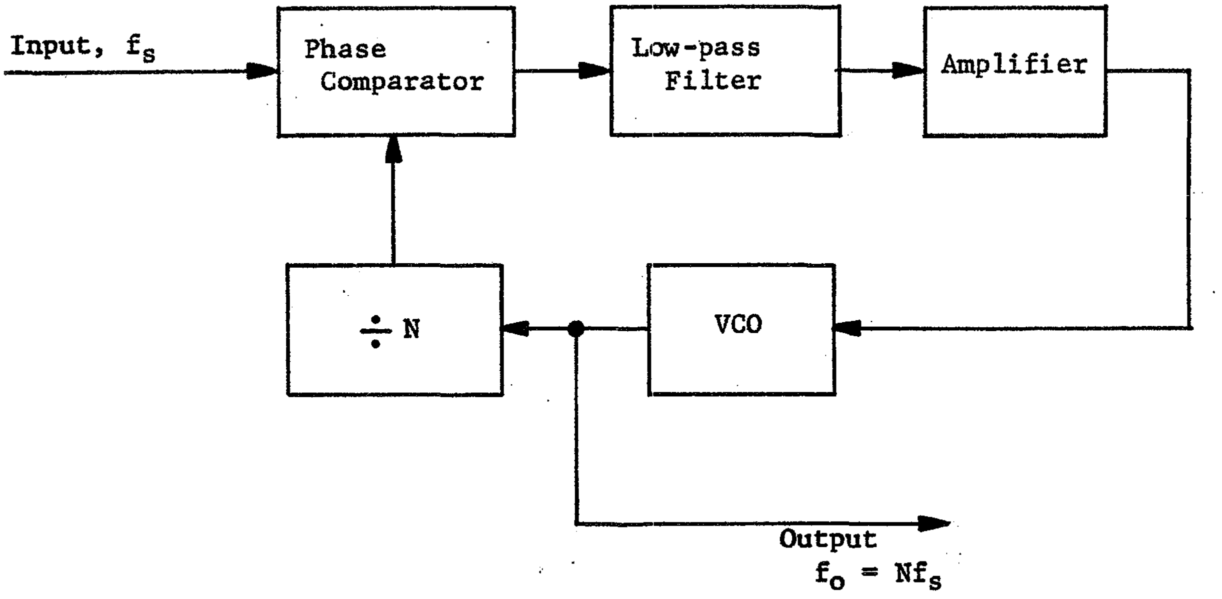


Fig. 3.34 Frequency multiplication by use of a phase-locked loop

(3) Quasisynchronous or mesochronous system. Each station has its own frequency control, but the bit rate can be changed to one or two or three discrete rates, or the frame rate may be changed by adding or dropping special bits, so as to ensure that the average frequency of each station is the same as any other, if a long period is considered. At any instant there will be a phase error between an incoming signal and the local signal, which must not be allowed to exceed a specified maximum in operation.

(4) Heterochronous system. Each station generates its own frequency within a specified tolerance of the nominal frequency. The tolerance must be kept small enough to reduce to negligible proportions the loss of information which occurs when a fast signal arrives at a slower station.

The only one of the four systems named above which offers potential application to the design of a fault-tolerant clock is the synchronous system. Such a method of synchronization of oscillators falls into the general method illustrated in Fig. 3.2. For use in PCM, schemes have been developed which consist of averaging the phases of all oscillators, comparing the result with each oscillator phase, and applying an error signal as a correction to the oscillator frequency. The oscillators which are used have frequencies which may be altered in proportion to a control signal; in the absence of external control, each oscillator operates at a different frequency. A system which has been examined by Gersho and Karafin (Ref. 4) is illustrated in Fig. 3.35. It has been proven that under suitable conditions the system is stable; i.e., the oscillators asymptotically settle to a common frequency and the phase differences have finite asymptotic values. The system illustrated in Fig. 3.35

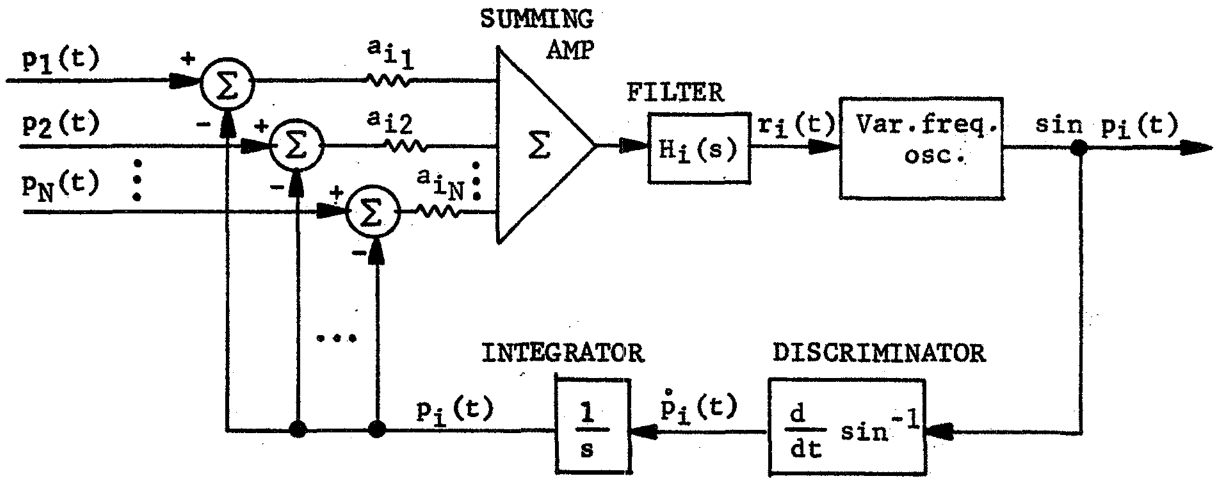


Fig. 3.35 Synchronization of PCM station oscillators - station i

is offered by Gersho and Karafin as an abstraction of the more complex practical systems that have been developed; the system of Fig. 3.35 requires a total phase comparison to be made, which is impractical to implement. In the PCM system, failure of a transmission link will lead to resynchronization if the remaining network is still connected. Also, in the case of oscillator failure, the remaining N-1 oscillators will resynchronize to a new frequency, if the resulting network of N-1 stations is still connected, after removal of all transmission links entering or leaving the inoperative station.

It is impractical to utilize the above system in the design of a fault-tolerant clock. Not only must provision be made to remove a failed oscillator from the system, but during the time when the system is adjusting to a new frequency, as a result of the failure of a link or oscillator, synchronous operation cannot be assured.

CHAPTER 4

CONCLUSIONS

In a multiprocessor which achieves fault-tolerance through replication, corresponding units must be kept in synchronism. In Chapter 2 the synchronization requirements have been defined and methods of maintenance of synchronism have been developed; the primary conclusion to be drawn from Chapter 2 is that a synchronous fault-tolerant multiprocessor driven by a fault-tolerant clock is more efficient and more easily implemented than is an asynchronous fault-tolerant multiprocessor. This conclusion leads naturally to the examination of fault-tolerant clocking.

In Chapter 3 specifications have been developed for a fault-tolerant clock and two general methods of design have been explored. It has been concluded that fault-tolerant clocking through failure-detection and subsequent clock substitution is possible but impractical to implement in a system of many synchronous modules due to the high cost of each module transducer. Fault-tolerant clocking through the concepts advanced by William Daly and John McKenna has been studied intensively; clocks developed by Daly and McKenna have been examined, refined, and revised. It has been concluded that it is desirable to have available a fault-tolerant clock which runs at 20 MHz, but that such a frequency is not achievable by a McKenna-type clock (with use of current technology). A method of achieving a 20 MHz clock by the use of a relatively slow McKenna-type clock in conjunction with a frequency multiplier has been developed in Section 3.5.4.

REFERENCES

1. Larsen, R.W., and Reed, I.S., "Redundancy by Coding Versus Redundancy by Replication for Failure-Tolerant Sequential Circuits," I.E.E.E. Transactions on Computers, Vol. C-21, No. 2, February, 1972, pp. 130-137.

2. Daly, W., and McKenna, J., M.I.T. C.S. Draper Laboratory, Digital Development Memo #627, August, 1971.

3. Mumford, H., and Smith, P.W., "Synchronization of a p.c.m. Network Using Digital Techniques," Proc. of the Institution of Electrical Engineering, Vol. 113, No. 9, September, 1966, pp. 1420-1428.

4. Gersho, A., and Karafin, B.J., "Mutual Synchronization of Geographically Separated Oscillators," Bell System Technical Journal, Vol. XLV, No. 10, December, 1966.

5. M.I.T. C.S. Draper Laboratory, <u>A Fault-Tolerant Information Processing System For Advanced Control, Guidance, and Navigation</u>, Report R-659, Cambridge, Mass., May, 1970.

6. M.I.T. C.S. Draper Laboratory, <u>STS Data Management System Design (Task 2)</u>, Report E-2529, Cambridge, Mass., June, 1970.

7. M.I.T. C.S. Draper Laboratory, Data Management System Configuration Studies for Off-The-Shelf Computers, (Task 28-S), Vol. I, Cambridge, Mass., December, 1971.