NASA-CR-193122

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

RESOURCE UTILIZATION MODEL FOR THE ALGORITHM TO ARCHITECTURE MAPPING MODEL

By

John W. Stoughton, Principal Investigator

and

Rakesh R. Patel, Graduate Research Assistant

Progress Report
For the period ended June 30, 1993

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

Under
Research Grant NCC1-136
Paul J. Hayes, Technical Monitor
ISD-Information Processing Technology Branch

June 1993

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

RESOURCE UTILIZATION MODEL FOR THE ALGORITHM TO ARCHITECTURE MAPPING MODEL

By

John W. Stoughton, Principal Investigator

and

Rakesh R. Patel, Graduate Research Assistant

Progress Report
For the period ended June 30, 1993

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

Under
Research Grant NCC1-136
Paul J. Hayes, Technical Monitor
ISD-Information Processing Technology Branch

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

June 1993

ABSTRACT

RESOURCE UTILIZATION MODEL FOR
THE ALGORITHM TO ARCHITECTURE MAPPING MODEL

Rakesh R. Patel
Old Dominion University
Director: Dr. John W. Stoughton

The analytical model for resource utilization, and the variable node time and conditional node model for the enhanced ATAMM model for a real-time data flow architecture, is presented in this research. The Algorithm To Architecture Mapping Model, ATAMM, is a Petri net based graph theoretic model developed at Old Dominion University, and is capable of modeling the execution of large-grained algorithms on a real-time data flow architecture. Using the resource utilization model, the resource envelope may be obtained directly from a given graph and, consequently, the maximum number of required resources may be evaluated. The node timing diagram for one iteration period may be obtained using the analytical resource envelope. The variable node time model, which describes the change in resource requirement for the execution of an algorithm under node time variation, is useful to expand the applicability of the ATAMM model to heterogeneous architectures. The model also describes a method of detecting the presence of resource limited mode and its subsequent prevention. Graphs with conditional nodes are shown to be reduced to equivalent graphs with time varying nodes and, subsequently, may be analyzed using the variable node time model to determine resource requirements. Case studies are performed on three graphs for the illustration of applicability of the analytical theories.

ACKNOWLEDGEMENTS

This is a thesis being submitted in lieu of a progress report for the research project entitled "ATAMM Enhancement and Multiprocessing Performance Evaluation" for the period ended June 30, 1993. This work was partially supported by the NASA Langley Research Center through research grant NCC1-136 and monitored by Paul J. Hayes, of the ISD, Information Processing Technology Branch, Mail Stop 473.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
A_i	Boundary across which resource requirement increases ($k \geq 1$)
AMG	Algorithm Marked Graph
ATAMM	Algorithm to Architecture Mapping Model
# CE	Number of Control Edges
CMG	Computational Marked Graph
D	Node Delay Time
DR	Data Read
EF	Earliest Finish
ES	Earliest Start
FDT	Fire Data Time
FU	Functional Unit
GDSC	Graph Description and Simulation Control
IE	Input buffer empty
IF	Input buffer full
k	Change in the number of resources for each node
k_i	Number of initial tokens in i^{th} path
LF	Latest Finish

LS	Latest Start
m	Number of outgoing edges for each node
N	Number of nodes in the AMG
NMG	Node Marked Graph
$NODE_{Ai}$	Node corresponding to A_i boundary
ODU	Old Dominion University
OE	Output buffer Empty
OF	Output buffer Full
p	Process Data
PC	Process Complete
P_i	Earliest finish of i^{th} path, or time to deposit a token at the sink
PID	Proportionate Integration and Derivative
PL	Path Length
PR	Process Ready
Q	Queue size
R	Resources
r	Read Input Data
R_{inc}	Increase in number of Resources
R_{max}	Maximum number of Resources
R_{min}	Minimum number of Resources
SGP	Single Graph Play
S_i	Boundary across which resource requirement decreases (k = -1)

S_I	Source (Input) node in AMG
S_NODES_{Ai}	Successor Nodes of the node contributing A_i boundary
S_O	Sink (Output) node in AMG
S_{SI}	Number of Successor nodes of the Source (Input) Node S_I
T	Number of outgoing edges with waiting tokens for each node
t{ }	Time positions of resource boundaries in one TBO frame
TBI	Time Between Inputs
TBIO	Time Between Input and Output
$TBIO_{LB}$	Lower bound limit of TBIO
TBO	Time Between Outputs
TBO_{LB}	Lower bound limit of TBO
TCE	Total Computing Effort
TGP	Total Graph Play
TRE	Total Resource Envelope
w	Write Output Data

CHAPTER ONE

INTRODUCTION

1.1 Problem Definition

Research is focused on the development of an analytical model for the resource utilization in the execution of large grain algorithms on heterogeneous, multicomputer, data flow architectures. The algorithms which may be implemented in an ATAMM defined data flow architecture are considered. Algorithm To Architecture Mapping Model, ATAMM, is a Petri net based theoretic model which describes data and control flow required for the execution of large grain algorithms on multicomputer, data flow architectures [1, 2].

1.2 Overview

The demand of high computing speeds is becoming predominant for any computer system day by day. This is especially true in applications such as real-time signal processing and complex control algorithms, which often require a timing deadline for the completion of a job. In many cases, it is desirable to increase performance of a computer system using multiple processors running the same algorithm concurrently. The ATAMM model describes the system behavior and predicts its performance for real-time algorithms, and facilitates

the mapping of these algorithms onto multicomputer, data flow architectures. In a data flow architecture [4], the execution of an instruction is controlled by the availability of data needed for its execution. In other words, flow of data causes the execution of instruction as opposed to the flow of control in conventional machines.

The ATAMM model consists of a set of Petri net marked graphs which incorporates the control and data flow definitions associated with each computational event to specify the criteria for the predictable execution of an algorithm with highly reliable performance [2]. It also provides the means for investigating different algorithm decompositions in an architecture independent way. Once the intended hardware is chosen, the model can be used to match the algorithm requirements with the hardware capability in order to achieve optimum performance. With availability of sufficient resources, the system executes algorithm with maximum throughput and minimum computing time.

The determination of resource requirements for achieving optimum time performance under worst case condition is of interest. The resource envelope is a time history of resource requirements over the interval between successive inputs. A model for analytical evaluation of the resource envelope, and consequently, the maximum resources required (R_{max}) is developed. For a system with non-homogeneous processors, node times of an algorithm graph vary as a node may be executed by a different processor in each repetitive

execution of an algorithm. Conversely, if node times vary, it may cause a change in resource requirement and may also cause resource limited mode. The conditions for the presence of resource limited mode under node time variation and its subsequent prevention are investigated. Also, many algorithms, such as control system algorithms, require conditional branching of nodes so that only one of the several successive paths is to be executed. Each node must be able to enable selectively one of the several outgoing paths. A conditional node algorithm graph may be reduced to a variable node time graph and, consequently, its resource requirement may be evaluated.

1.3 Research Objective

The objective of this research is to develop the analytical resource utilization model to evaluate the resource envelope and maximum resource requirement, R_{max}, directly from a given graph, and to investigate resource requirements for time varying node graphs and conditional node graphs. The model also provides an analytical means of determining the presence of resource limited mode with time varying nodes and, subsequently, its elimination without degrading system performance and predictability. The approach taken to evaluate the analytical resource envelope is straightforward, and is based upon calculating the path length from the source input to each deposited output token. The execution of a graph with initial tokens on forward edges is also investigated to evaluate the critical path and the critical

path length, since this type of graph structure is obtained in some control applications. In addition, this class of graphs had not been treated previously in the ATAMM model development. Research is aimed as an enhancement to the ATAMM model with respect to incorporation of graphs with time varying and conditional nodes, and also with initial token markings on the forward edges.

1.4 Thesis Organization

Basics of the ATAMM model is presented in Chapter Two and the performance measures are defined in the ATAMM context. The deposit time of a token is defined along with the fire time of any node in the AMG. The execution of graphs with initial tokens on the forward edges in the systems based on the ATAMM model is investigated, and the critical path and the critical path length are defined for these graphs. Such graphs distinctly represent control system algorithms with initial conditions embedded in them. Path length, modulo-TBO operation and other terminology relevant to this thesis research are defined.

The development of the analytical resource utilization model to evaluate the resource envelope and the Total Graph Play (TGP) diagram is presented in Chapter Three. Consequently, a method for determining the value of R_{max} is described, assuming a worst case analysis. In addition, the analysis of algorithm graphs with time varying nodes is discussed in consideration of

instantaneous increases in resource requirements. A method for analyzing and eliminating the potential for this increase beyond R_{max}, or resource limited mode, with graphs having time varying nodes is presented. An overview of a conditional node graph is presented, and its mapping on the ATAMM based systems is discussed by obtaining an equivalent reduced graph which contains time varying nodes. Consequently, a method for determining the resource requirement for conditional node graphs is outlined.

The analytical model developed in Chapter Three is illustrated by performing case studies on example algorithm graphs in Chapter Four. The analytical results obtained using theoretical model are compared with the experimental results. The experimental results are obtained using ATAMM software support tools such as Simulator (Version 2.3) and Analyzer. The case studies are performed on three algorithm graphs, the first is a graph with three parallel paths, the second is a graph with a closed circuit in it, and the third is a conditional node graph.

The summary of research with appropriate conclusions and evaluations is given in Chapter Five, and topics for future research are outlined.

CHAPTER TWO

OVERVIEW OF ATAMM MODEL AND BASIC DEFINITIONS

2.1 Introduction

In this chapter, an overview of the ATAMM data flow architecture model is presented. Background for the ATAMM model is presented in Section 2.2. The performance measures of the model are described in Section 2.3. The material presented in Sections 2.2 and 2.3 extensively uses information previously reported by Stoughton and Mielke [6], Jones [7], and Mandala [12]. The concept of deposit time, fire time, and related issues are discussed in Section 2.4. These issues are an important factor in the development of the analytical model for resource utilization, and consequently investigating the behavior of graphs with time varying and conditional nodes as described in Chapter 3. The notion of deposit time and fire time is also useful in the evaluation of the critical path and the critical path length for a graph with initial tokens on forward edges, as described in Section 2.5. In Section 2.6, other terminology relevant to the research is described.

2.2 ATAMM Model

Since the last decade, because of continuous increase in dependency on high speed computing environment, multiprocessor and parallel processing

systems have become an area of intensive research. The development of parallel architectures composed of a number of identical, special purpose computing elements is of particular interest [6]. The computing elements of a distributed system must share distributed resources and information for better utilization. Therefore, there is a need to synchronize and control this sharing in order to obtain accurate overall system operation [7].

The ATAMM model is an outcome of research at Old Dominion University, in conjunction with NASA-Langley Research Center, to develop a multicomputer operating strategy for an implementation of large-gained, decomposed algorithms on data flow architectures. This model is of particular importance because it provides a context in which algorithm decomposition strategies can be investigated without the need to specify a specific computer architecture. The model also identifies the data flow and control dialogue required of any data flow architecture which implements the algorithm. In addition, the model provides a basis for analytically calculating the performance bounds for computing speed and throughput capacity [5].

The ATAMM model consists of three Petri net marked graphs called the algorithm marked graph (AMG), the node marked graph (NMG), and the computational marked graph (CMG). A Petri net is a special kind of directed graph which is capable of describing data and control flow of a system [7]. Petri nets serve as both a graphical and mathematical tool. An example of a marked graph is shown in Figure 2.1. In a marked graph, circles represent

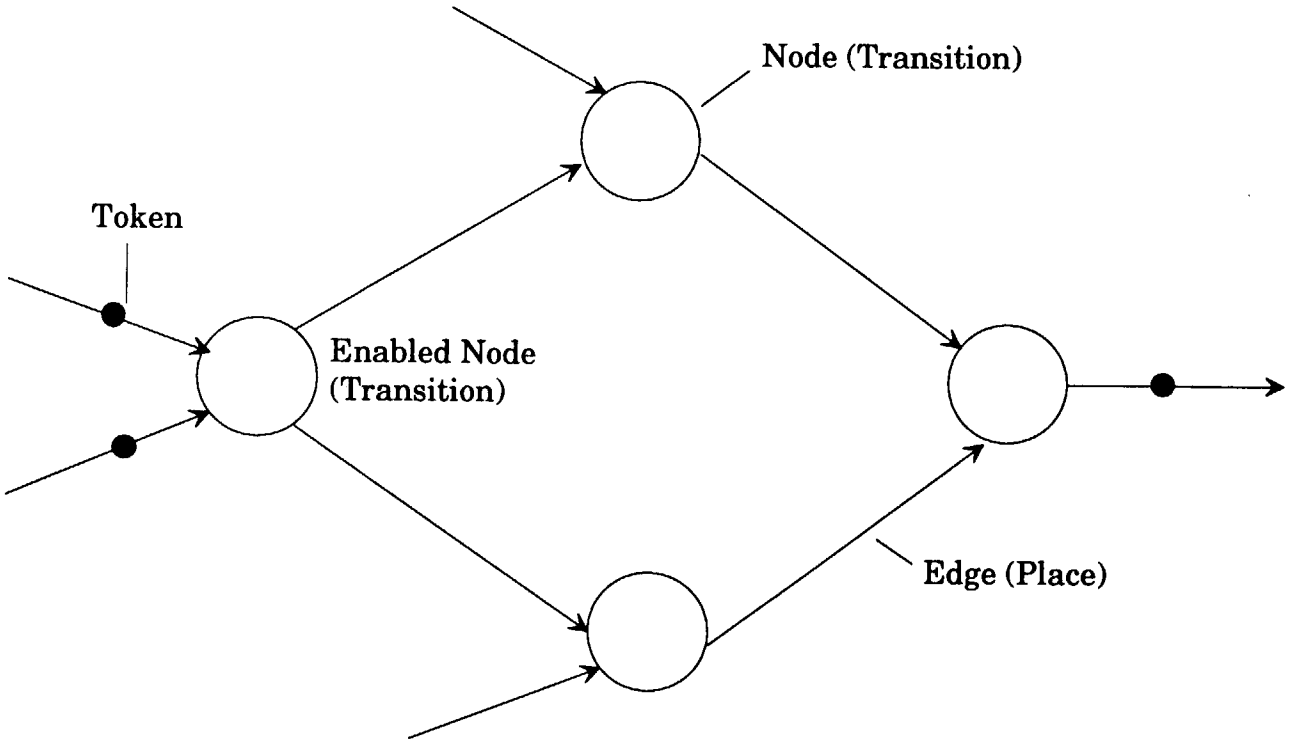


Figure 2.1. Marked graph.

nodes (transitions or actual computations) and line segments represent edges (places or flow of data). The black dots on the edges represent tokens which indicate availability of data. A node is enabled or fired by the presence of tokens on all incoming edges.

The AMG is a representation of a specific algorithm decomposition. Operations are represented by nodes and operands are represented by directed edges. Availability of data is represented by the presence of tokens on incoming edges. Source and sink transitions for input and output signals (data packets) are represented as squares. An example illustration of an AMG for a discrete system equation is shown in Figure 2.2. The AMG does not display procedures that a computing structure must manifest in order to perform the computing task. Also, the issues of control flow, time performance, and resource management are not apparent from this graph.

The NMG is a Petri net graph that represents the performance of an algorithm operation by a functional unit. Three basic activities, reading of an input data from global memory, processing an input data to compute an output data, and writing of output data to global memory, are represented as transitions (nodes) in the NMG. Data and control flow paths are represented as places (edges), and the presence of data is shown by tokens marking appropriate edges. A read transition can be fired only if a functional unit is available in a queue of available functional units and a token is present on each incoming edge. Once assigned, the functional unit is used to implement

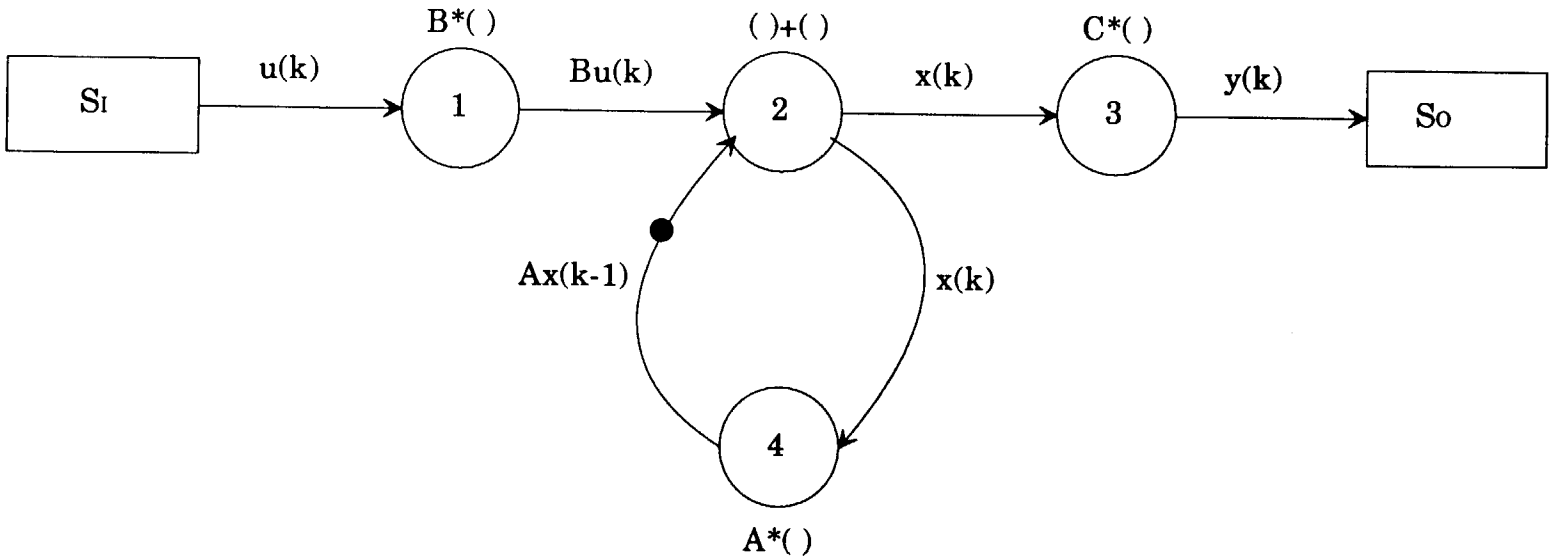


Figure 2.2. Algorithm marked graph (AMG) for discrete system equation $x(k) = A\,x(k-1) + B\,u(k)$ and $y(k) = C\,x(k)$.

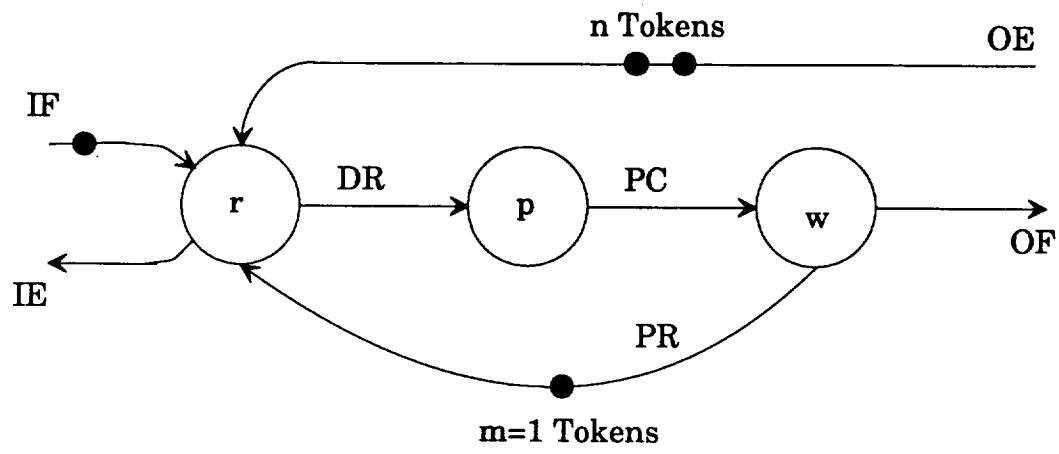
the read, process, and write operations before being returned to a queue of available functional units. An NMG describing these basic activities, along with the meaning of edge labels, is shown in Figure 2.3.

The CMG is constructed from the AMG and the NMG using the following rules:

1. Source and sink nodes in the algorithm marked graph are represented by the source and sink nodes respectively in the CMG.
2. Nodes corresponding to algorithm operations in the algorithm marked graph are represented by NMGs in the CMG.
3. Edges in the algorithm marked graph are represented by the edge pairs, one forward directed edge for data flow and one backward directed edge for control flow, in the CMG.

The play of the CMG proceeds according to the following graph rules:

1. A node is enabled when all incoming edges are marked with a token. An enabled node fires by absorbing one token from each incoming edge, delaying for some specified transition time (equal to time required for node computation), and then depositing one token on each outgoing edge.
2. A source node and a sink node fire when enabled, independent of the availability of a functional unit.
3. A node precess in initiated when the read node of an NMG is enabled and a functional unit is available for assignment to the NMG. A FU



NMG Arc Labels

IF Input Buffer Full
IE Input Buffer Empty
DR Data read
PC Process Complete
PR Process Ready
OE Output Buffer Empty
OF Output Buffer Full

NMG Node Labels

r Read Input Data

p Process Data

w Write Output data

Figure 2.3. ATAMM node marked graph (NMG) model.

(functional unit) remains assigned to an NMG until completion of the firing of the write node of the NMG.

A CMG representation of the AMG of Figure 2.2 is shown in Figure 2.4. The complete ATAMM model consists of the AMG, the NMG, and the CMG. A pictorial view of the ATAMM model with its components is shown in Figure 2.5.

The CMG of Figure 2.4 has some important characteristics. Execution of the CMG results in live, reachable, safe, deadlock free, and consistent behavior. Liveness indicates that every transition of the graph can be fired from the initial marking [5]. Reachability implies that an output will be produced for every input. The CMG is safe because the backward control edges prevent data from being overwritten, or they prevent a graph from being over-crowded with excessive data packets. The backward control edges prevent enablement of a transition until previous output data are being picked up. The CMG is also deadlock free, because once assigned to a node, a functional unit is always able to complete node execution. Consistency implies that the CMG periodically produces output when inputs are applied periodically [5].

There are two types of concurrency possible during the execution of an algorithm as specified by the CMG. Nodes belonging to the same data set and which are independent of each other may be executed simultaneously. This type of concurrency is referred to as parallel concurrency and has a direct effect on computing speed. It is limited by the number of nodes that can be

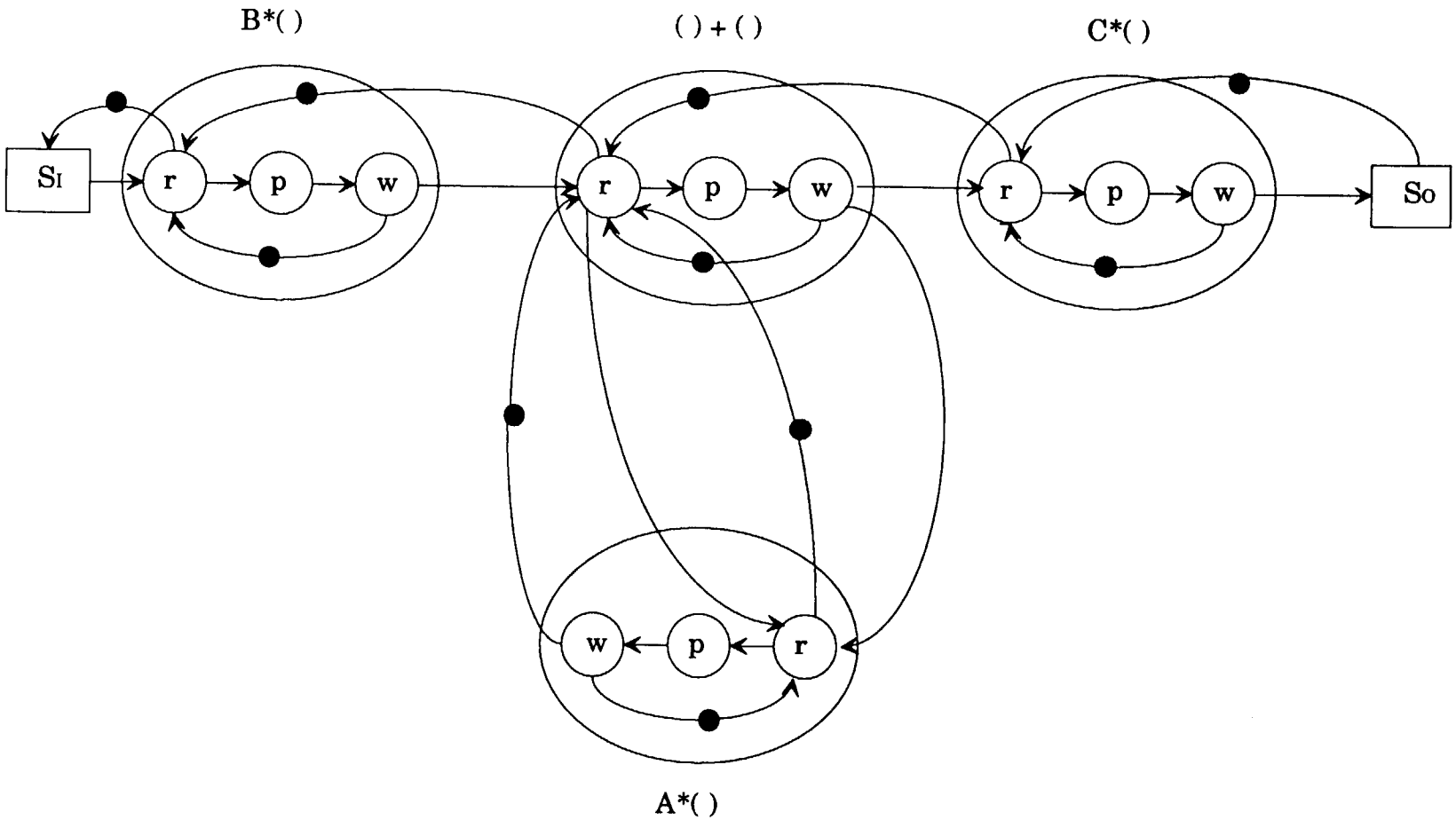


Figure 2.4. ATAMM computational marked graph (CMG) model for discrete system equation.

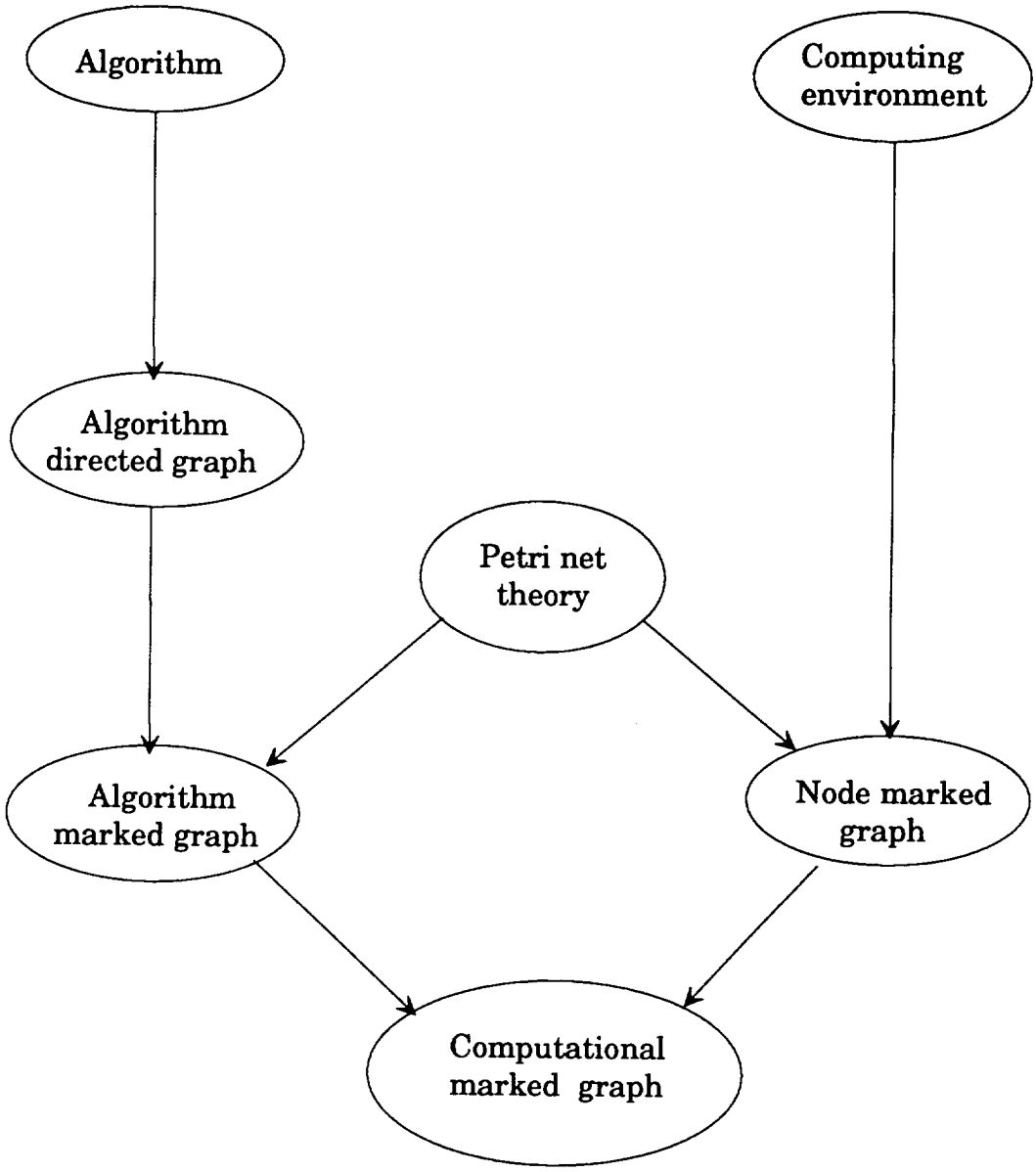


Figure 2.5. ATAMM model components.

performed simultaneously for a given algorithm graph and by the number of functional units available. Also, nodes belonging to different data sets can be performed simultaneously in the computing system. This type of concurrency is referred to as pipeline concurrency [4]. It is limited by the capacity of the graph to accommodate additional data sets and by the number of functional units available to implement the algorithm periodically.

2.3 Performance Measures

In this section, basic measures of time performance in the ATAMM model are described. The determination of resource requirements for the execution of a given graph on a data flow architecture is presented. Also, the ATAMM performance plane is described.

In Section 2.3.1, two time performance measures, TBIO and TBO, are defined. A brief overview of a graph play and corresponding resource requirements is presented in Section 2.3.2. In Section 2.3.3, the ATAMM performance plane is defined, and an example for illustration is presented.

2.3.1 Performance Measures

The performance measure TBIO (Time Between Input and Output) is the elapsed time between an algorithm input and the corresponding output. TBIO is an indicator of the computing speed. The lower bound for TBIO, denoted as $TBIO_{LB}$, is given by the sum of node (transition) times for nodes

contained in the longest directed path from input source to the output sink in the AMG. This is shown in [4]. The performance measure TBO, for the time between outputs, is the elapsed time between successive algorithm outputs when the AMG is operating periodically at steady state. The inverse of TBO is an indicator of output per unit time or throughput. The lower bound for TBO, imposed by the algorithm, is given by the largest time per token of all directed circuits in the CMG [4]. The lower bound for TBO, imposed by available resources, is given by TCE/R where TCE (Total Computing Effort) is the sum of node times for all nodes in the AMG and R is the number of available functional units (resources). The lower bound for TBO, denoted as TBO_{LB}, is the greater of the algorithm bound and the resource bound.

2.3.2 Injection Control and Resource Requirements

In this section, a brief description of injection control is presented. Then, two diagrams which display graph play and are useful for determining the number of resources required to achieve specified performance measures are defined.

Injection control is a control procedure which limits the maximum rate at which new input data packets can be injected. A data packet is an input data set. For real-time control and signal processing applications, the algorithm is repeated periodically with new input data sets [4]. When presented with continuously available input data packets, the natural behavior

of a data flow architecture results in an operation where data packets are accepted as rapidly as available resources and the input node transition permit. This leads to a steady state operating point where TBO = TBO_{LB}, but TBIO > $TBIO_{LB}$. This occurs because the pipeline from input to output becomes congested with extra data packets which must wait for free resources to be processed. Injection control eliminates data packet congestion and thus preserves operation at $TBIO_{LB}$. An example implementation of injection control strategy is shown in Figure 2.6.

In the AMG, the longest path from the input source to the output sink, measured in terms of time, is defined as the critical path. There can be more than one critical path for a given AMG. In the example AMG of Figure 2.7, nodes 1, 2, and 5 form a critical path. In this graph, there is only one critical path. The critical path length (TBIO) is 5. The CMG for AMG shown in Figure 2.7 is given in Figure 2.8.

The single graph play (SGP) diagram is a diagram which displays the execution of each node of the AMG as a function of time. The diagram is constructed for a single input data packet under the assumption that unlimited resources are available to play the graph. Node activity is denoted by a solid line and the symbols (<,>) are used to indicate the beginning and end of node execution. When several nodes are active at the same time, lines indicating node activity are stacked vertically so that computing concurrency is apparent.

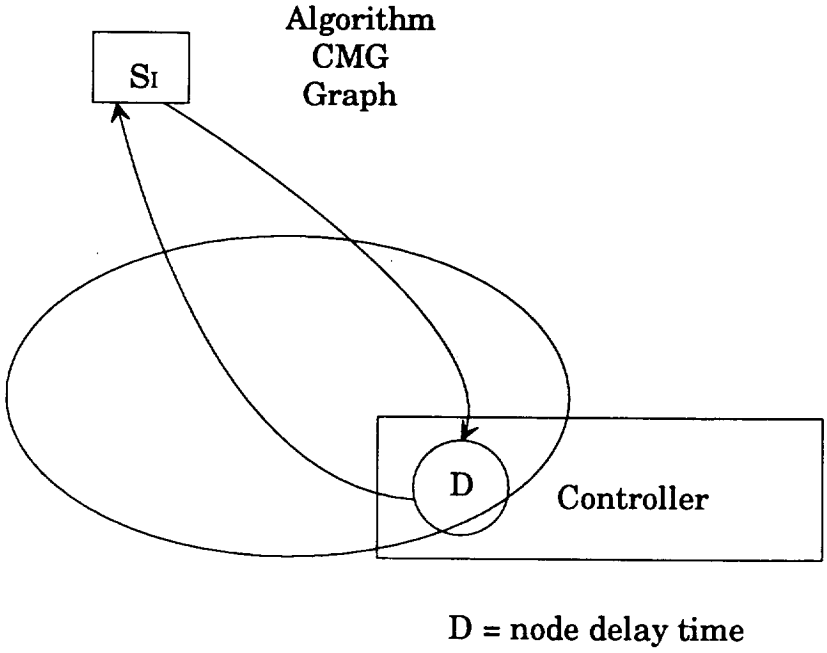


Figure 2.6. Implementation of injection control strategy.

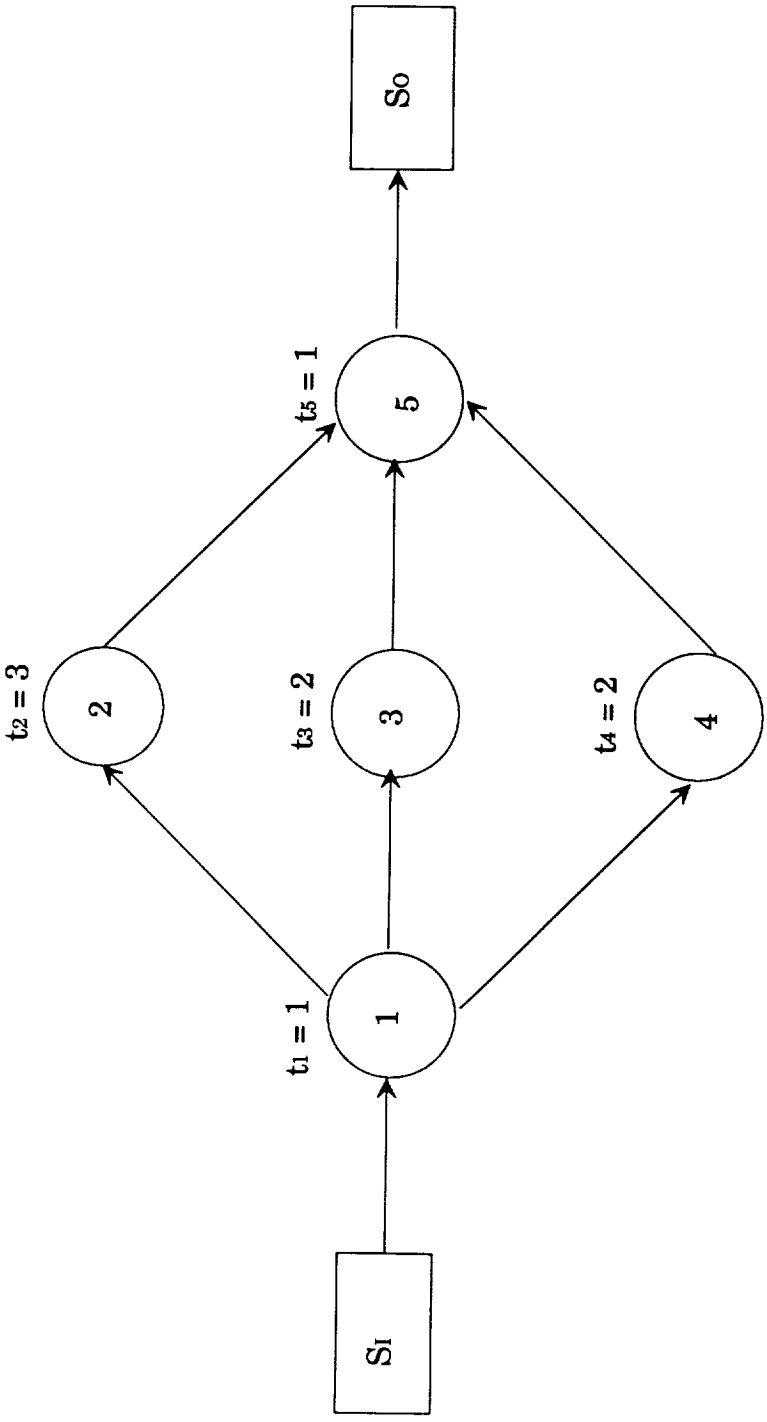


Figure 2.7. Example AMG.

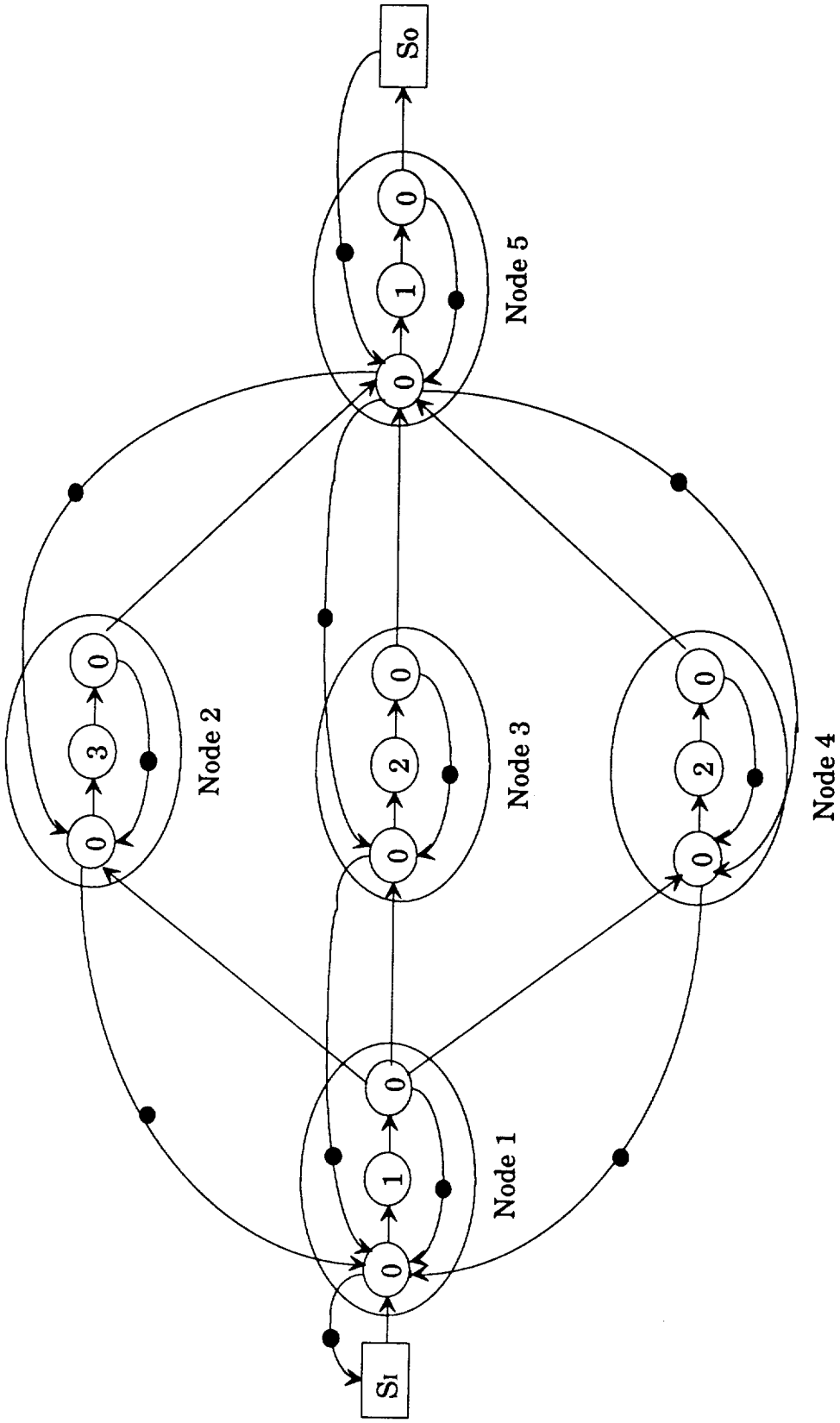


Figure 2.8. CMG for AMG of Figure 2.7.

The SGP diagram for the CMG shown in Figure 2.8 is given in Figure 2.9. The data packets are numbered in the same sequence in which they are injected.

The number of resources required to execute a single data packet is obtained by counting the number of active nodes during each time interval in the SGP diagram. The peak resource requirement is denoted by R_{min}, and it represents the minimum number of resources necessary to achieve operation at TBIO = $TBIO_{LB}$.

The total graph play (TGP) diagram displays the execution of each graph node when the graph is operating periodically in steady state with a period of TBO. The TGP diagram is constructed using information from the SGP diagram. (However, in Chapter 3, we will see that the TGP diagram may be constructed using information from the Total Resource Envelope determined analytically from a given AMG). The SGP diagram is divided into segments of width TBO, and these segments are overlaid to form the TGP diagram. Each segment from the SGP diagram represents a new input data packet. Data packets are numbered sequentially so that the packet numbered i+1 is the data packet which is input to the graph TBO time units after the packet numbered i. The TGP diagram for the SGP diagram of Figure 2.9 is shown in Figure 2.10.

The resource requirements to execute multiple data packets injected with a period equal to TBO are obtained by counting the number of active nodes during each time interval in the TGP diagram. The peak resource

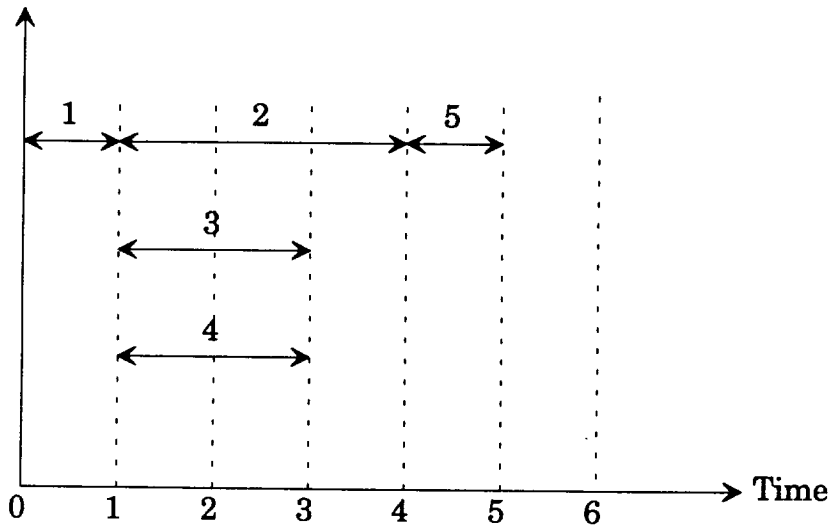


Figure 2.9. SGP diagram for the CMG of Figure 2.8.

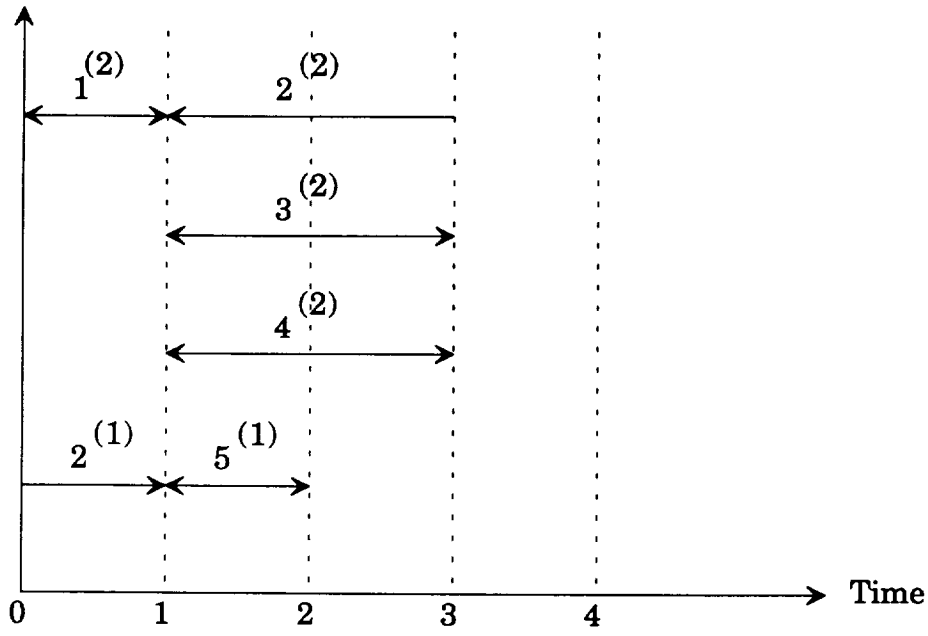


Figure 2.10. TGP diagram for the SGP diagram of Figure 2.9 with TBO = 3.

requirement R_{max} is determined by finding the largest resource requirement in all TGP diagrams drawn for injection intervals greater than or equal to TBO. From Figure 2.10, it is evident that a minimum of four resources are required for TBO_{LB} equal to 3. It can be easily shown that if the TGP diagram is drawn for values of TBO > 3, the resource requirement does not exceed 4. Therefore, the peak resource requirement R_{max} is 4.

2.3.3 ATAMM Performance Plane

The display of all the operating points on a graph of TBO versus TBIO with R as a parameter is called the ATAMM performance plane diagram. An example performance plane diagram is shown in Figure 2.11.

The system exhibits the best time performance when operated at the lower bounds of TBO and TBIO. Operation of the algorithm graph at these lower bounds is achieved using input injection control. The resource requirement at this point is the value R_{max} obtained in the TGP diagram drawn for $TBIO_{LB}$ and TBO_{LB}. Under conditions of nonavailability of sufficient resources, the operating point must be shifted to a place so that fewer resources are required. By using injection control, the operating point can be moved along the vertical line A-V. This operating strategy preserves TBIO but degrades throughput performance [8]. The operating points on the vertical line A-V are calculated from the TGP diagram by increasing TBO until the number of active nodes in any time interval decreases by one from the previous

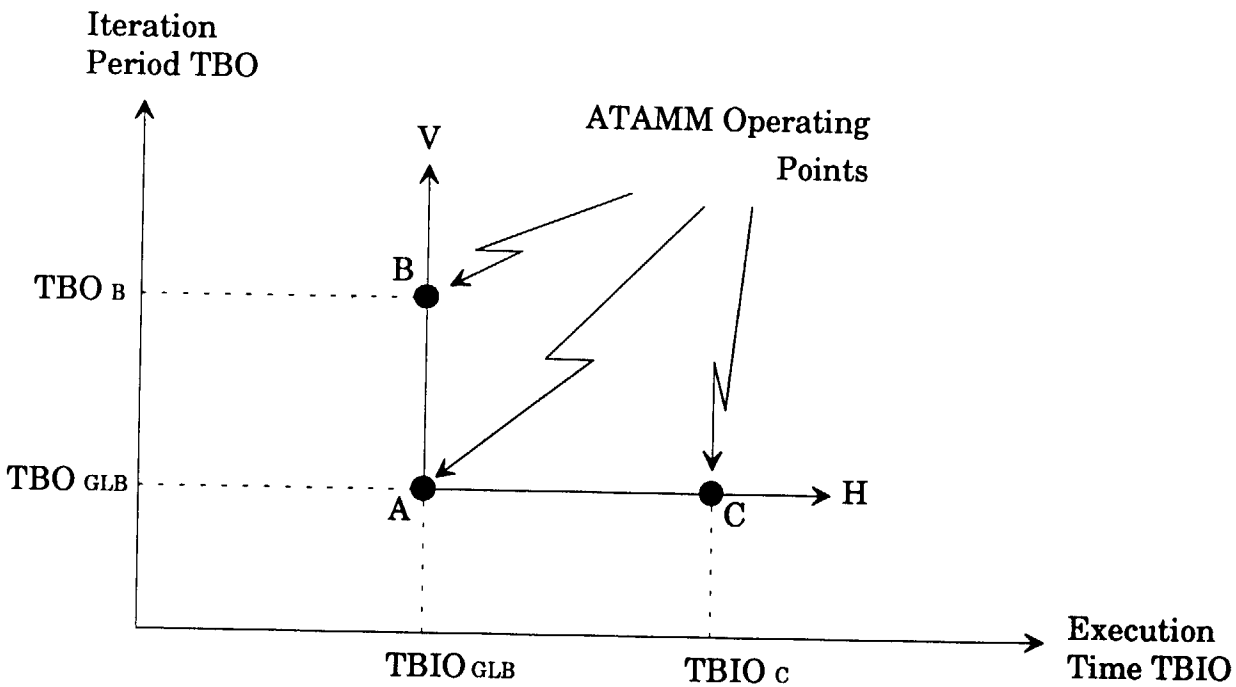


Figure 2.11. ATAMM performance plane diagram.

operating point. As an example, consider the AMG of Figure 2.7. By increasing TBO from 3 to 4, as shown in the TGP diagram of Figure 2.12, the number of required resources decreases to 3. Increasing TBO to 5 would not reduce the resource requirement. These points are shown in Figure 2.13 along the vertical line at TBIO = 5.

To reduce resource requirements, the operating point also can be moved along the horizontal line A-H. This operating strategy degrades computing speed but preserves TBO [9]. This strategy is implemented by adding control edges to the original AMG. A control edge is an AMG place which imposes a precedence relationship among two transitions, but does not imply data dependency [9]. When such an edge is added to an AMG, the longest path from input to output increases, thus increasing TBIO. The addition of control edges can crate new directed circuits having increased time per token values so that TBO is also increased. This can be avoided by increasing the number of buffers (queue size) on an edge in the AMG. Every edge has an initial buffer size of one which serves as a storage for the output of a node. By increasing the number of buffers on an edge, the token count on circuits formed by adding control edges can be increased so that the value of TBO is preserved. Operating point design using control edges and buffer spaces is explained in more detail in [4].

As an illustrative example, consider an AMG of Figure 2.7. Adding a control edge from node 3 to node 4, as shown in Figure 2.14, requires that

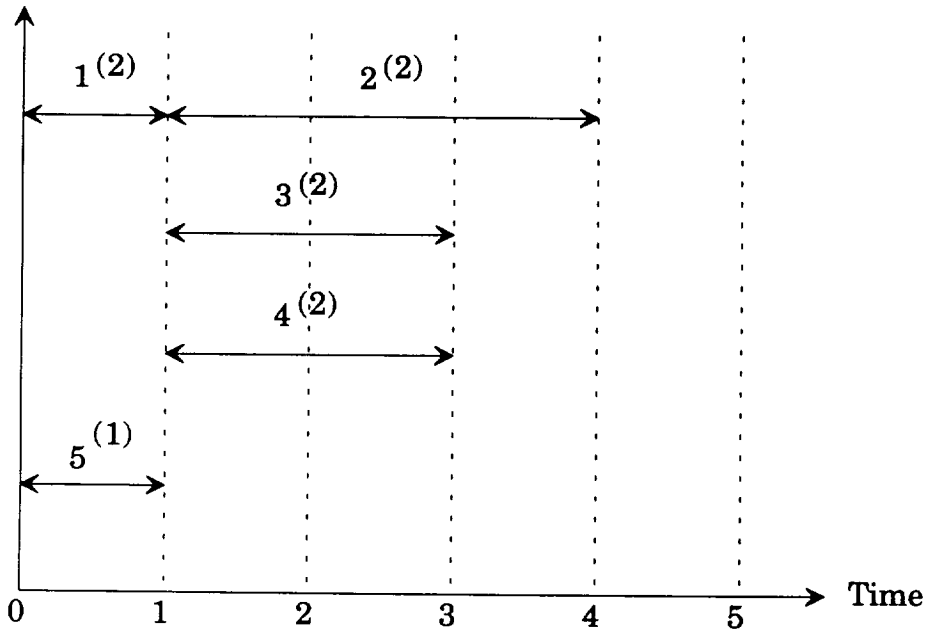


Figure 2.12. TGP diagram for the SGP diagram of Figure 2.9 with TBO = 4.

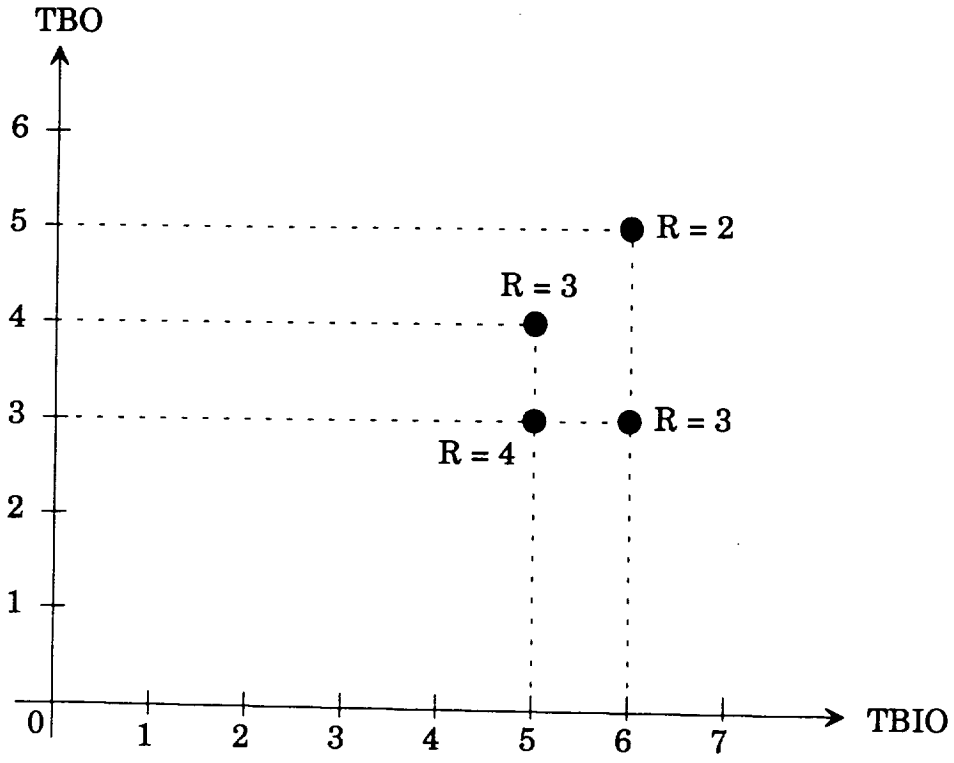


Figure 2.13. ATAMM performance plane diagram for the AMG of Figures 2.7 and 2.14.

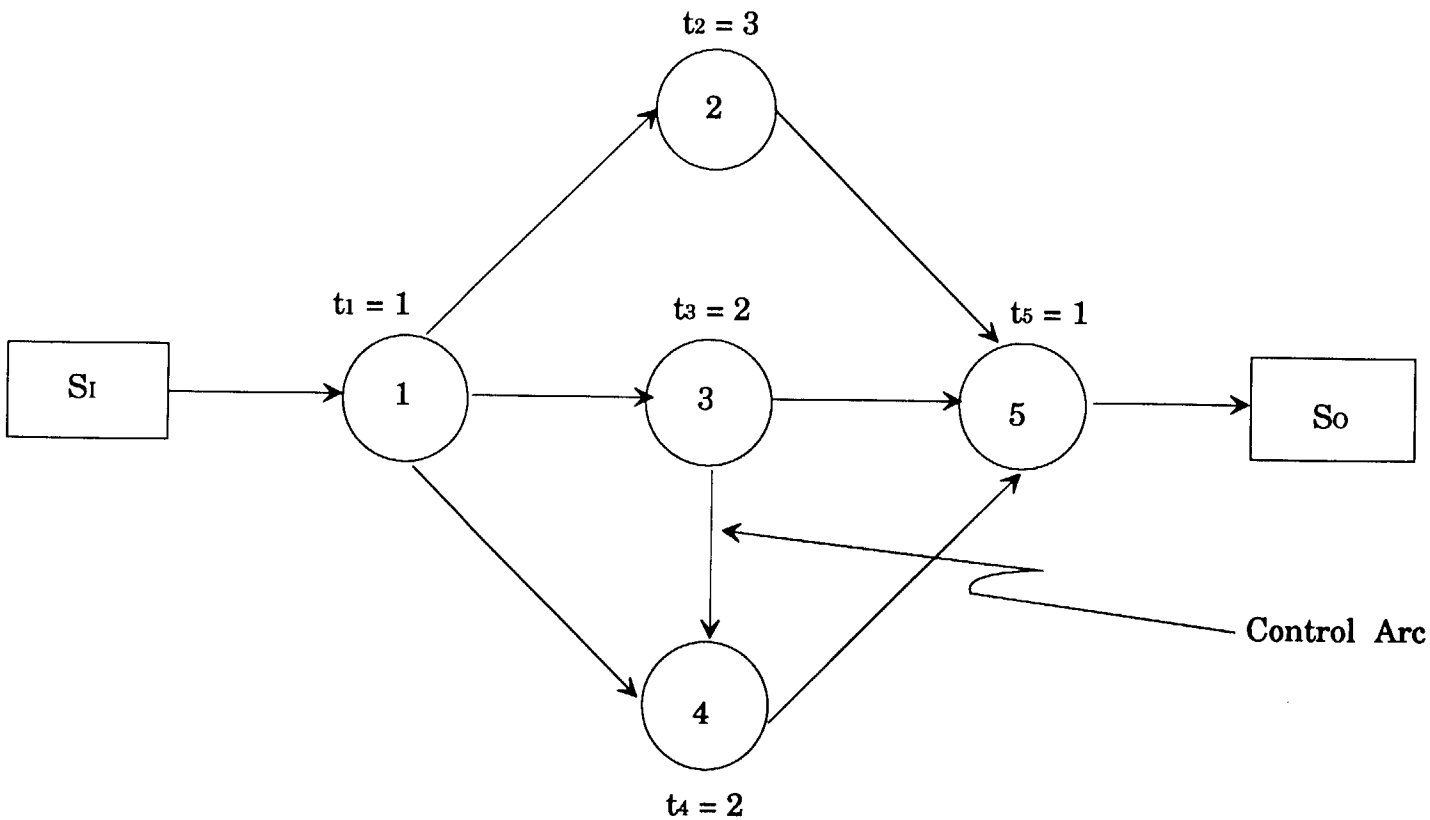


Figure 2.14. Example AMG of Figure 2.7 with control arc added.

buffer size be increased between nodes 1 and 4. The TGP diagram for the AMG of Figure 2.14 is shown in Figure 2.15. Here TBO = 3, TBIO = 6, and R_{max} = 3. The new operating point at TBO = 3 and TBIO = 6 for R = 3 is shown in Figure 2.13. Additional operating point at TBO = 5 and TBIO = 6 for R = 2 is obtained by using injection control.

The performance plane diagram provides information essential for the selection and control of the time performance of algorithms executing under ATAMM rules. Operating points are selected by identifying R points in the performance plane, one point corresponding to each resource number. The point associated with a specific value of R identifies the value of TBIO and TBO when the system is operating with R resources. If the number of resources changes, then a new operating point is identified. Operation at the new point is realized by modifying the graph with control edges and buffers, and adjusting the input injection interval.

2.4 Deposit Time and Fire Time

In this section, description of deposit time of an enabling token for the firing of a node is presented. This will be taken as a basis for the development of an analytical model for resource utilization in Chapter 3. The latest time at which a node may fire, or is ready to fire is called fire time. The necessary condition for firing is outlined and, consequently, the fire time of a node is defined in terms of the deposit time of enabling tokens. This is useful in

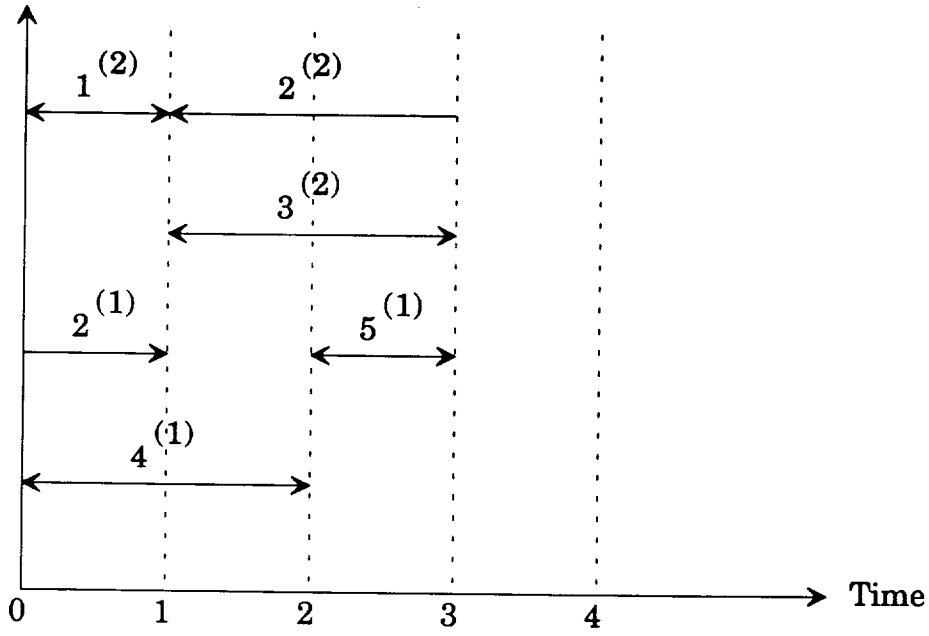


Figure 2.15. TGP diagram for the AMG of Figure 2.14 with TBO = 3.

finding the critical path and TBIO for a graph with forward initial tokens, as will be seen in Section 2.5.

2.4.1 Waiting Token and Deposit Time

An enabling token which waits (on an incoming edge of a node) for tokens from other nodes to become available is called a waiting token. The positions of waiting tokens in the graph, and deposit times of waiting tokens in one TGP (steady state) time interval are important in the construction of the resource envelope. The resource envelope gives the information regarding number of resources utilized at each time instant over one TBO time frame. Therefore, finding the net change in number of resources required at the completion of each node in the AMG along with their time positions over one TBO time interval are sufficient for the development of the resource envelope. The concept of a waiting token and its deposit time directly leads us to obtain basic requirements for analytically developing the resource envelope.

In an abstract sense, a waiting token is present on an edge if the token is waiting for other token(s) from other node(s) to become available. As for example, a node may have several edges directed to it from other nodes. Each of these predecessor nodes must finish execution and deposit an enabling token on all incoming edges of the successor node in order to fire the successor node. Of interest is determining the predecessor node which deposits the last enabling token. An edge directed from this predecessor node to the successor

node of interest constitutes an edge with no waiting token, and all other edges directed to its successor are assumed to have waiting tokens on them, even if there may be more than one paths present with equal lengths. This assumption is relevant because in physical reality, token must be present for an infinitesimally small amount of time.

The physical interpretation of a waiting token on an outgoing edge from any node is that the resource assigned to this node is freed and is entered in a queue of available resources, thus reducing the resource requirement by one. An absence of a waiting token means that the release of a resource occupied by a node is now replaced by a resource assigned to its successor, thus causing no net change in number of resources. For example, if node A has three outgoing edges on two of which waiting tokens are present, one of the three successors fires immediately after A is finished, and there is no net change in the number of resources utilized. If a waiting token is present only on one edge, two resources are required immediately after one is released, which gives a net increase of one resource. If none of the edges has a waiting token, the net increase is two.

In summary, for any node in the AMG, N-1 incoming edges to a node out of a total N would have waiting tokens present on them. For each edge with a waiting token, time to deposit the token is equal to the path length of a waiting token, and this may be calculated by finding the longest path from the source to a particular edge, and summing all the node times in this path.

2.4.2 Fire Time of a Node

A node may be fired when each of the incoming edges to the node has an enabling token present on it. An enabling token may correspond to either a current data packet or any of the previous data packets. If the path contains an initial token on any edge, the data dependency is reduced by one packet number at that point, and an enabling token is obtained from any of the previous data packets depending on the number of initial tokens in the path. Data packet index is reduced by one for each initial token present in the path.

Fire time of a node is the time at which all the enabling tokens for the node are available. Fire time is the maximum of deposit times of all the enabling tokens (either from a present or any previous data packets) for a node. The notion of fire time of a node is used in the following section in determining the critical path and TBIO for a graph with initial markings on forward edges.

2.5 Critical Path for AMG with Forward Initial Tokens

In this section, an algorithm for finding the critical path and, consequently, calculating TBIO for an AMG with initial tokens on the forward edges is presented. Consider an algorithm implementation of a control system which has a unit delay operator Z^{-1} in it. This constitutes operation on previous data values. This kind of situation may be handled in the directed graph by inserting an initial token on the appropriate edge. The calculation

of the steady state critical path is not straightforward for this type of graph due to the presence of initial token(s) on forward edge(s). The modified AMG method, to find the critical path, does not address the case when AMG contains forward initial tokens. The algorithm presented here is used to identify the critical path without any misinterpretation.

2.5.1 Identification of the Critical Path

A node requires that data be available on all the incoming edges for its enablement, either in the form of the present or previous data packet(s). Availability of data on edges is represented by the presence of a token on each edge. A node is not fired until the longest directed path from source leading to the node has data available as an input to the node. In other words, the node fires when a token is available on an input edge which corresponds to longest path to the node. This means that enablement of any node is determined by the longest time to deposit of token on each incoming edge.

An initial token on any edge represents data dependency on the previous data packet in firing of a node on which edge directs. Absence of an initial token represents dependency on data corresponding to the present data packet. If the number of initial tokens in one of the several paths from the source leading to any node is n, then the enablement of this node is partially dependent on the availability of the $(i-n)^{th}$ data packet where i is current data packet.

In summary, the maximum of the times to deposit each of the enabling tokens (whether from present or previous data packet) to a particular node decides the longest path from source to this node. The critical path of a given AMG is, by definition, a longest path from input to output of a graph. Therefore, if we find times to deposit a token (P_i) at the input of the sink for all possible paths starting from the source, then the maximum value of P_i gives the critical path length and determines the critical path. To find time to deposit a token for a path having initial token(s) on it, we subtract one TBO interval from the path length for each initial token present on the path, since the token corresponding to previous data packet is available prior to TBO time period.

A method for the determination of the critical path is outlined as follows:

1. Find all possible paths from source to sink and compute corresponding path lengths.
2. Find the number of initial tokens present in each of the paths found above.
3. Find time to deposit a token (P_i) for each path by subtracting number of TBO time units equal to the number of initial tokens in the corresponding path.
4. The maximum value of P_i gives TBIO and the corresponding path is the critical path.

This method may be considered as an extension of the original approach (modified AMG method) to account for initial tokens in the forward path [2].

2.5.2 Critical Path Evaluation (Example)

As an illustration of this method, consider a digital PID controller [11] shown in Figure 2.16. If we determine the transfer function of a digital controller, it can be implemented by a computer. The operator z^{-1} is interpreted as a time delay of T seconds, where T is the sampling period. This time delay is implemented by storing a variable at some storage location and then taking it out after T seconds have elapsed. Once this relation is established, we can easily identify the program of any physically realizable transfer function. The transfer function for the digital differentiator and the digital integrator is given by $G_D(z)$ and $G_I(z)$ respectively, as shown in Figure 2.16. Figure 2.17 shows a block diagram representation of the digital program of the PID controller in Figure 2.16.

The algorithm implementation of the PID controller in Figure 2.17 is given in Figure 2.18. One node has a self loop, and a total of three initial tokens are present in the AMG. Firing node F requires the current data packet from node A and node C, and the previous data packet from node E. The results of applying the critical path evaluation method to the AMG of Figure 2.18 are tabulated in Table 2.1, which is self-explanatory. The critical

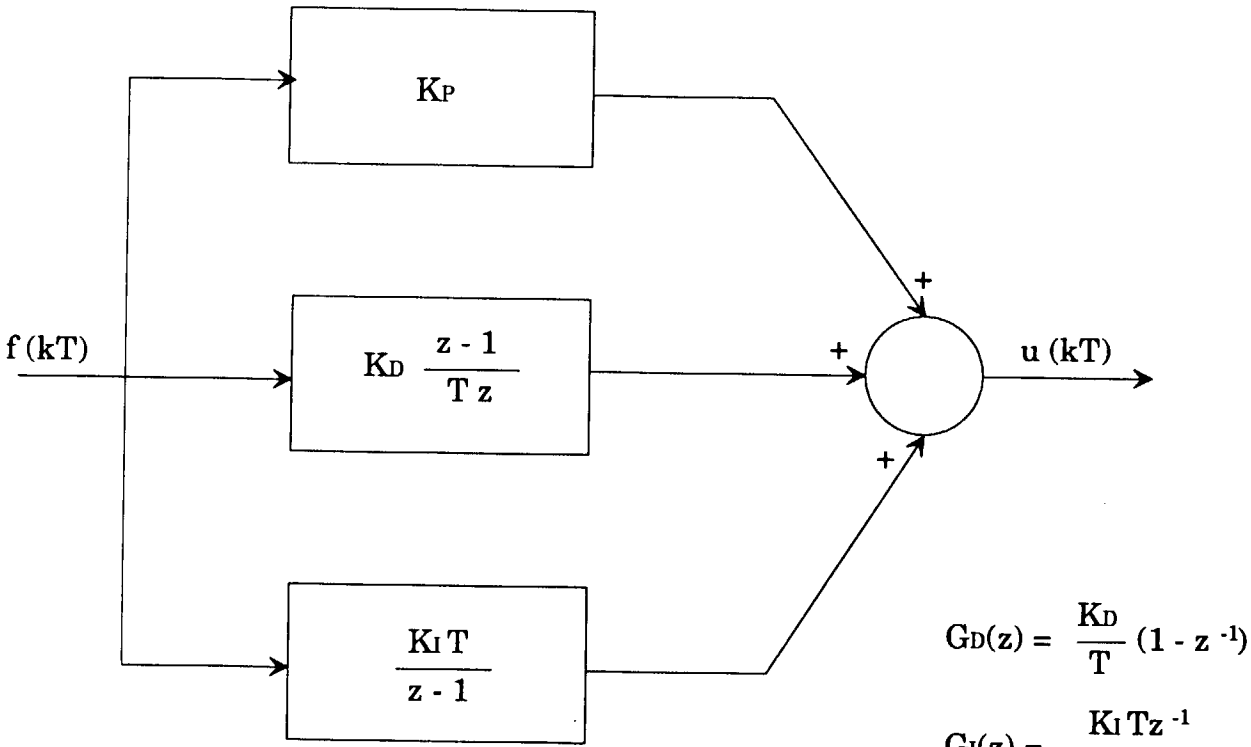


Figure 2.16. Block diagram of a digital PID controller.

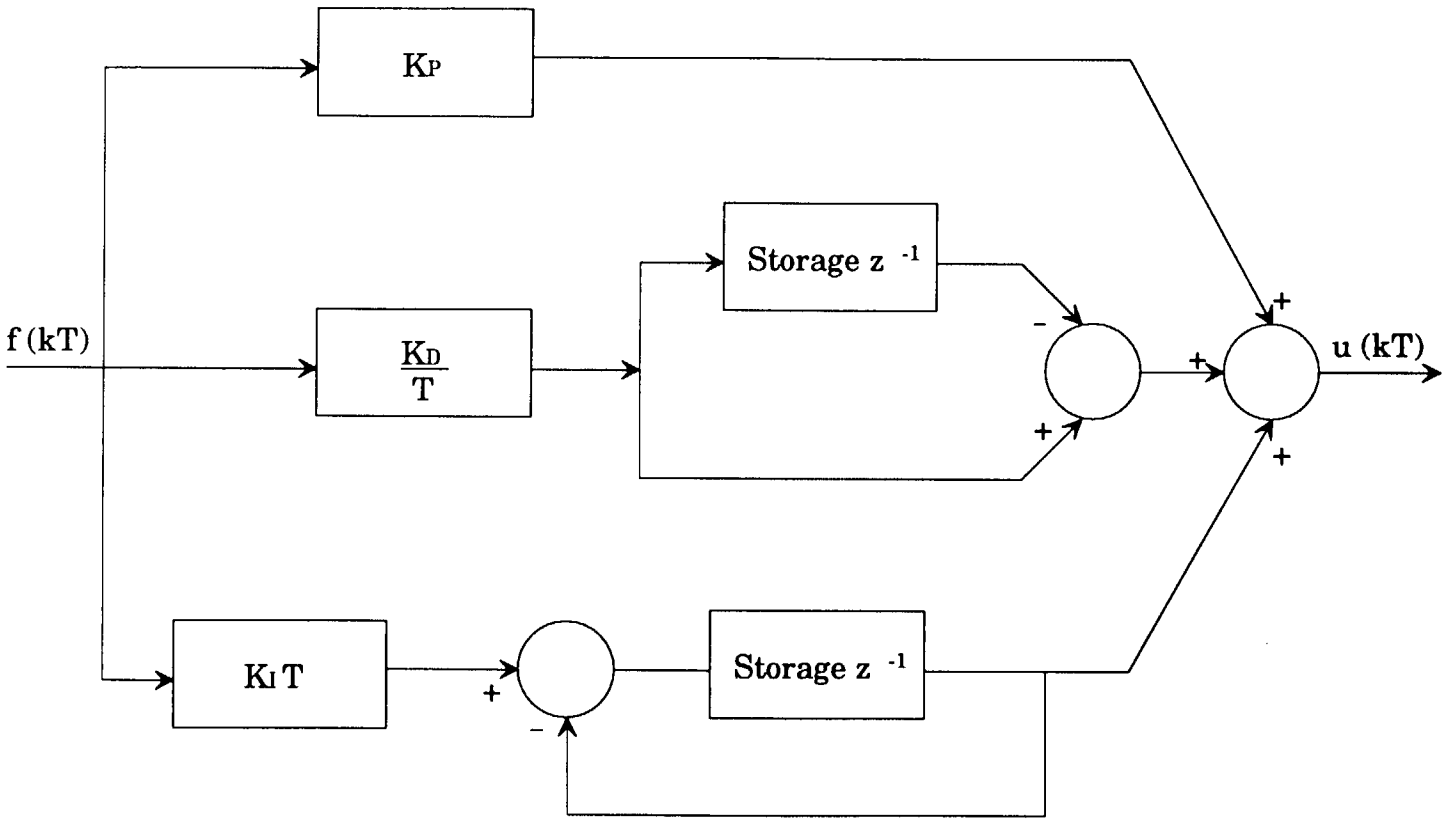


Figure 2.17. Block diagram of a digital program implementation of the PID controller.

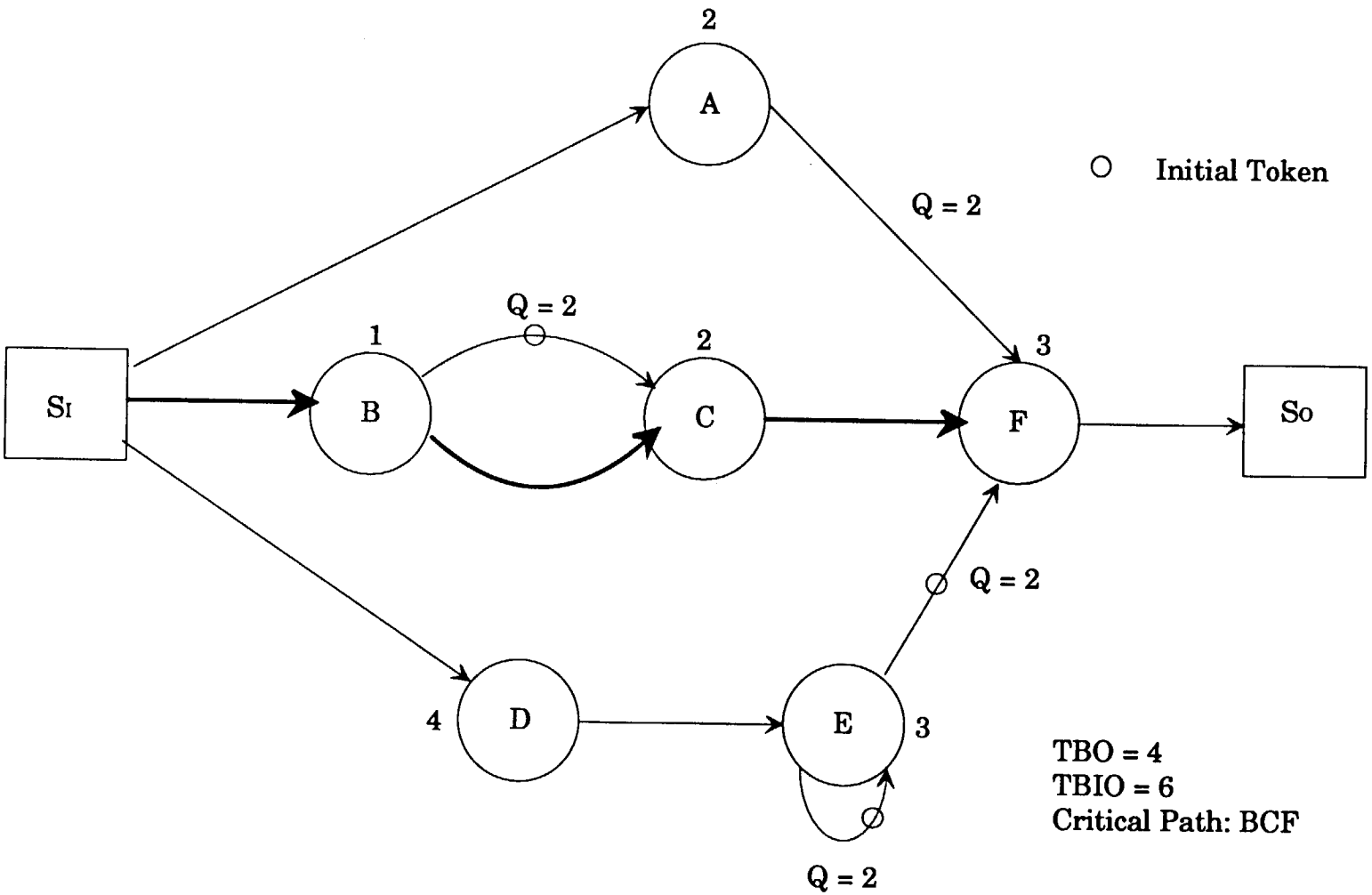


Figure 2.18. The AMG implementation of the PID controller shown in Figure 2.17.

Path	Path Length	# of Initial Tokens, k_i	P_i = PL - k_i TBO
AF	5	0	5
BCF_1	6	1	2
BCF_2	6	0	6
DEF	10	1	6

Table 2.1. Finding the critical path for the AMG of Figure 2.18 with initial tokens on forward edges.

TBO = 4
Max (P_i) = 6 for BCF_2
Critical Path = BCF_2
TBIO = 6

path is B-C-F, and TBIO is equal to 6. The TGP diagram for the AMG in Figure 2.18 is shown in Figure 2.19.

Assume that the AMG in Figure 2.18 is changed by inserting node G between node E and node F as shown in Figure 2.20. Now, node F requires for firing the current data packet from node A and node C, and the previous data packet from node G. The results for the AMG of Figure 2.20 are outlined in Table 2.2. The critical path for the new graph is found to be D-E-G-F, and TBIO is equal to 9.5. Also, the TGP diagram for the AMG in Figure 2.20 is shown in Figure 2.21.

2.6 Other Terminology

The purpose of this section is to define new terminology relevant to the thesis research. This includes mod TBO operation, integer(path length/TBO) operation, relative data packet number, and k-boundary.

Mod TBO operation for a given path length and a given TBO is defined as

$$Mod\ TBO\ (Path\ Length) = Remainder\ (\frac{Path\ Length}{TBO}). \qquad (2.1)$$

In the mod TBO operation, a given path is wrapped around over one TBO time interval (in the TGP diagram), and the remainder time period (left over portion of the path after wrapping, which incorporates the end of the path under consideration) is found. The value of mod TBO(path length) varies from

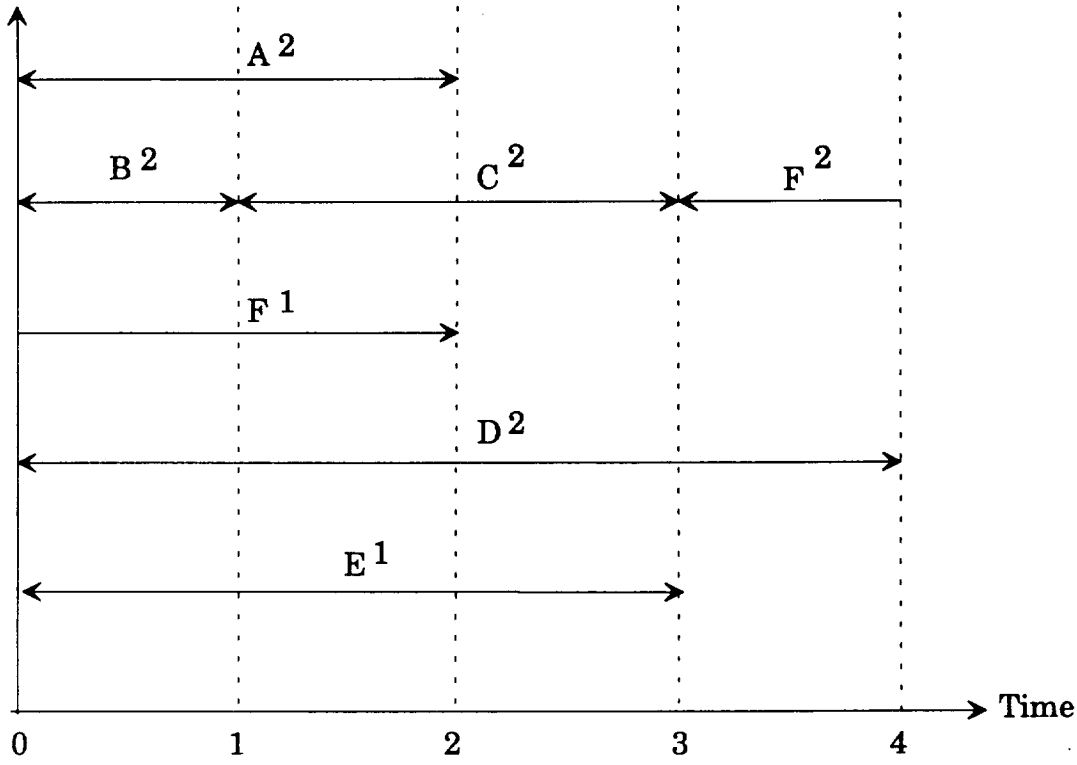


Figure 2.19. The TGP diagram for the AMG of Figure 2.18.

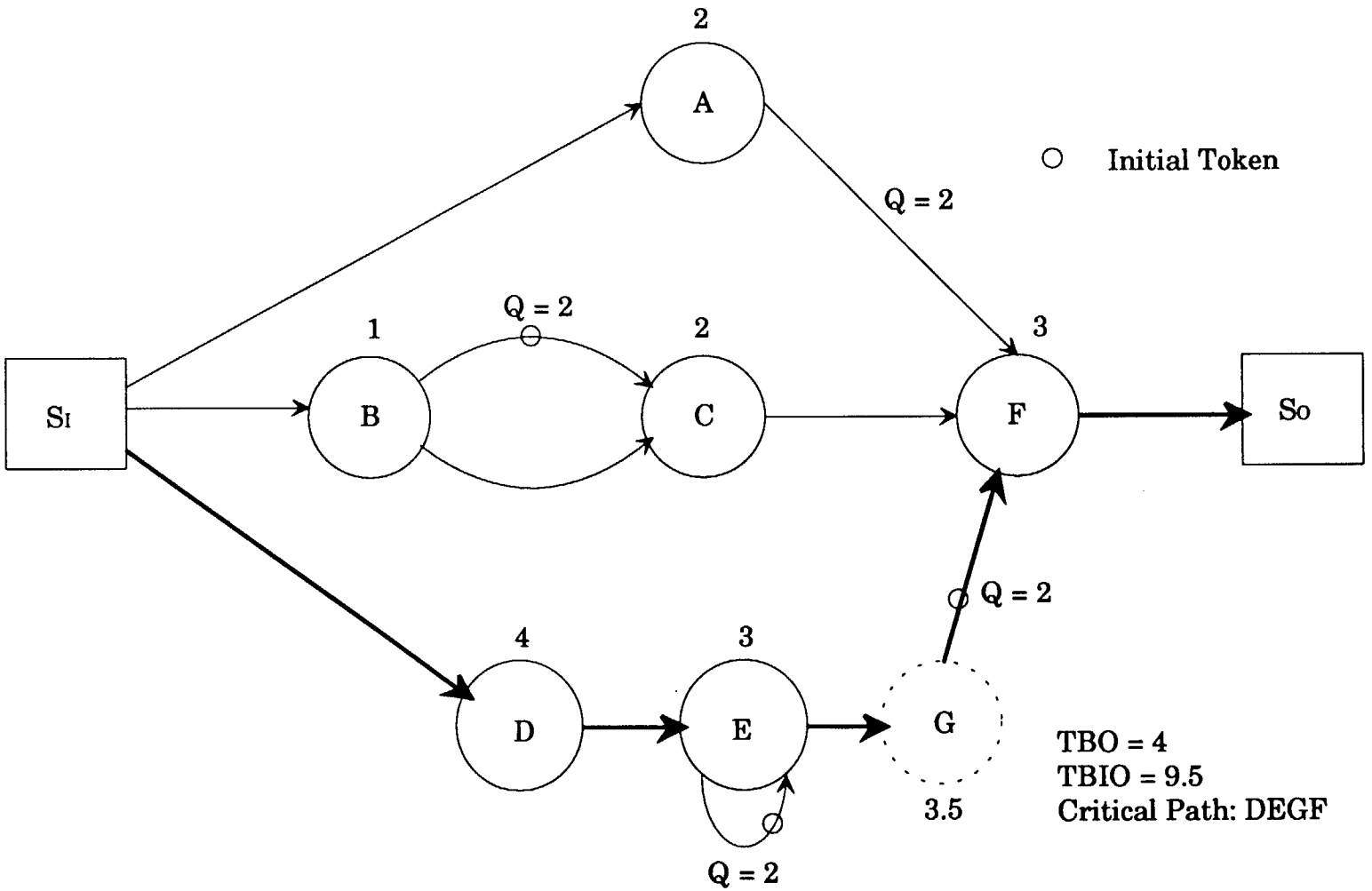


Figure 2.20. The AMG of Figure 2.18 with an extra node G added.

Path	Path Length	# of Initial Tokens, k_i	P_i = PL - k_i TBO
AF	5	0	5
BCF_1	6	1	2
BCF_2	6	0	6
DEGF	13.5	1	9.5

Table 2.2. The critical path for the AMG of Figure 2.20 (with an extra node added to the AMG of Figure 2.18).

TBO = 4
Max (P_i) = 9.5 for DEGF
Critical Path = DEGF
TBIO = 9.5

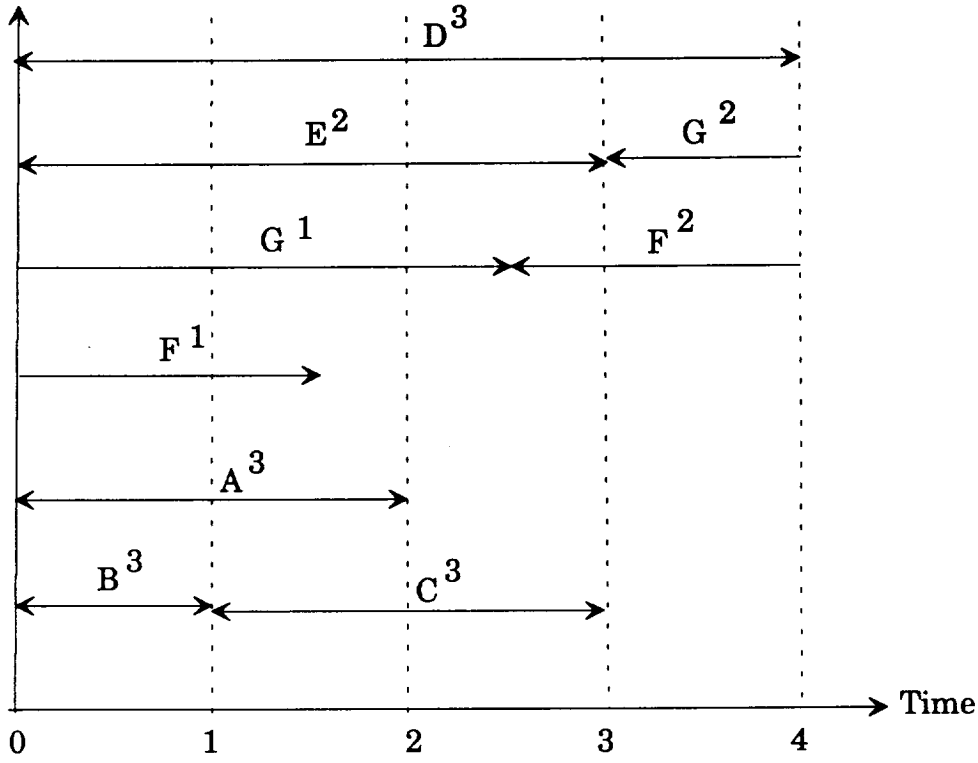


Figure 2.21. The TGP diagram for the AMG of Figure 2.20.

0 to (but not including) TBO. In other words, outcome of the mod TBO operation on a given path length falls in a semi-open interval [0, TBO). Thus, the outcome of mod TBO(path length) is periodic with a period TBO, and hence t = 0 and t = TBO are equivalent points.

The mod TBO operation is of particular importance, because it defines Latest Finish (LF) and Earliest Start (ES) of related nodes in the TGP diagram. The mod TBO operation will be used in the development of the resource envelope in Chapter 3. As an example, if TBO is equal to 3, and path length is 8, then mod TBO(path length) is remainder(8/3) which is equal to 2. If path length is either 4, 7, or 10, then mod TBO(path length) = 1. If path length is either 3, 6, or 9, then mod TBO(path length) is equal to 0 since remainder(path length/TBO) gives 0.

The integer(path length/TBO) operation gives the number of complete wrappings of path in the TGP diagram, and it shows the packet number at the end of path relative to the current data packet. For the current data packet, if the path does not extend to the next wrapping, the data packet at the end of path is the current data packet even if the path extends over a complete TBO time interval. The integer(path length/TBO) operation is defined as

$$Integer\,(Path\,Length/TBO) = \frac{Path\,Length - Mod\,TBO}{TBO}. \qquad (2.2)$$

This operation is utilized to find the data packet number relative to the current data packet of each resource boundary. Assuming that the current

data packet number is i, packet numbers corresponding to each resource boundary may be found by subtracting the value of integer(path length/TBO) from i. The relative data packet numbers may be, in turn, used in the construction of the TGP diagram from the resource envelope, as will be described in Chapter 3. This gives a general view of the TGP diagram for any data packet i.

The k-boundary is defined as a boundary (in one TBO interval) across which there is a net change in resource requirement by k. A positive value of k represents an increase in resource requirement by k on immediate right of position of k as compared to resources on immediate left. A negative value of k represents a decrease in resource requirement by k on immediate right of position of k as compared to resources on immediate left of boundary. The value k = 0 represents no net change in number of resources, and hence it does not contribute to a resource change boundary. The position of k-boundary is determined by mod TBO(path length) operation for a waiting token contributing k-boundary, where "path length" is the time to deposit of a particular waiting token. Because of the use of mod TBO operation, the position of each k-boundary varies in the range [0, TBO).

CHAPTER THREE

DEVELOPMENT OF ANALYTICAL RESOURCE UTILIZATION MODEL

3.1 Introduction

The development of the analytical resource utilization model is presented in this chapter. This model is based on the path length evaluation from the input to each of the waiting tokens in the AMG. The model allows an analytical development of the TGP diagram. The model also allows calculation of resource requirements and identification of resource limited mode for the graphs with time varying and, consequently, conditional nodes. It also allows an analytical determination of R_{max} from the worst case analysis, assuming that all the nodes in the graph take the maximum allocated time to execute.

The analytical resource utilization model, which describes the evaluation of the analytical resource envelope at steady state for one TBO time period, is developed in Section 3.2. Consequently, the value of R_{max} can be determined analytically, as discussed in Section 3.2. Since the deposit times of the waiting tokens are used in the development of the resource envelope, and relative packet indexes are known, the TGP diagram can also be determined from the analytical resource envelope, as discussed in Section 3.3. This approach, which always gives the steady-state view of the TGP diagram, is a refinement to a

method of obtaining the TGP diagram by folding the SGP diagram [4]. The analytical resource utilization model leads us to a method of finding the effect of node time variation in the AMG on the peak resource requirement, which is discussed in Section 3.4. It is found that the time varying nodes in an AMG may cause the temporary (instantaneous) resource limited mode, and subsequently, a preliminary approach for detecting and preventing resource limited mode is presented. This may help in improving the system flexibility by incorporating heterogeneous processors. An overview of the conditional node model, and a method of mapping the conditional node graphs onto the ATAMM model are presented in Section 3.5. Also the evaluation of resource requirement for conditional node graphs under worst case analysis, and under variable node times is discussed in this section.

3.2 Analytical Model for Resource Utilization

The development of the resource envelope from a given algorithm graph basically utilizes the path length from the input to each of the waiting tokens in the AMG. The waiting tokens define the boundaries across which one or more nodes either start or finish execution. There may a change in the resource requirement across these boundaries. Therefore, the evaluation of these boundaries is important in developing the resource envelope. Consequently, finding the positions of waiting tokens is the first, and crucial, step in the determination of the resource envelope.

3.2.1 Waiting Tokens in AMG

In this section, the procedure for marking the waiting tokens in the AMG is outlined. In a given AMG, it is necessary to mark the positions of waiting tokens in order to visualize the change in the number of resources across different regions in the TGP diagram, and subsequently to develop an analytic method to define resource change boundaries in the TGP diagram.

Any node in the graph may be described as a fork node, a join node, a simple node, a node in a self loop, or any combination of these. A fork node may be defined as a node which has at least two immediate successors. A join node may be defined as a node to which two or more edges are directed to it. A simple node has only one incoming edge and one outgoing edge. A node in a self loop has an outgoing edge which is also an incoming edge to the node. Each node defines a boundary at which there may be a change in the number of concurrent resource. All the incoming edges to a join node must have tokens available before a join node can be fired. The time of token deposit on each edge depends on the longest path from source node of the AMG to the node outputing the token of interest. The path length of this longest path gives the time with respect to the input at which the token is deposited on the edge under consideration.

Since a node can be fired only after all the incoming edges have tokens, the join node cannot be fired earlier than the longest of all the paths terminating on the join node. This may cause tokens on the other edges

directed into the join node to wait depending upon the relative timing of the nodes. This means that two or more resources are freed by two or more nodes forming a join, and are replaced by a single resource utilized by the join node. Thus, the join causes the resource requirement to decrease.

If more than one path from the input source to the join have the same length, tokens are still assumed to wait on all except one edge, even if the tokens arrive at the same time. This is done for consistency in further analytical development. Consider the following special cases in the view point of the presence of a waiting token.

1. Source. The source is assumed to have no incoming edge. No token wait on the outgoing edges because TBI (Time Between successive Inputs) is set equal to TBO of the AMG, so the very first node is fired as soon as the data packet from the source has been inputed.
2. Sink. The sink is assumed to have no outgoing edge, but has one or more incoming edges. It is also assumed that the sink fires as soon as it is enabled. However, all the incoming edges to the sink are marked with waiting tokens, because completion of each predecessor of the sink releases one resource and the sink itself does not consume a resource.
3. Graph with circuits. For the AMG containing circuits, the critical path is found using the modified AMG. The edges on which tokens wait can easily be determined by marking the critical path in AMG. Case 2 also must be considered.

As an example, consider the AMG shown in Figure 3.1, and its corresponding TGP diagram shown in Figure 3.2. Node 8 in the AMG is a join node. The critical path is 1-4-5-8, the critical path length is 8.5, and TBO is equal to 3. As marked in the AMG, three waiting tokens are present, one on the edge directed from node 7 to node 8, second on the edge directed from node 3 to node 8, and a third on the edge directed from node 8 to the sink. The location of first two waiting tokens are found noting that node 8 is a join node at which three edges are terminated, and one out of three (from node 5 to node 8) is on the critical path. Node 1 is a fork node because it has three edges directed out without waiting token. Successors of node 1, nodes 2, 4 and 6, fire at the same time when node 1 finishes. The completion of node 1 frees up one resource and, at the same time, three more resources are utilized by nodes 2, 4, and 6. The resource requirement increases at the boundary given by the time at which node 1 completes. Of interest is the analytical evaluation of these boundaries to replace the graphical representation as shown in Figure 3.2.

3.2.2 K-Boundaries

In the previous section, the positions of waiting tokens in the AMG were shown to be determined by searching for the longest path from the input source to a particular node. K-boundary is a boundary, in the TGP diagram, across which there is a net change in resource requirement by k. K-boundaries

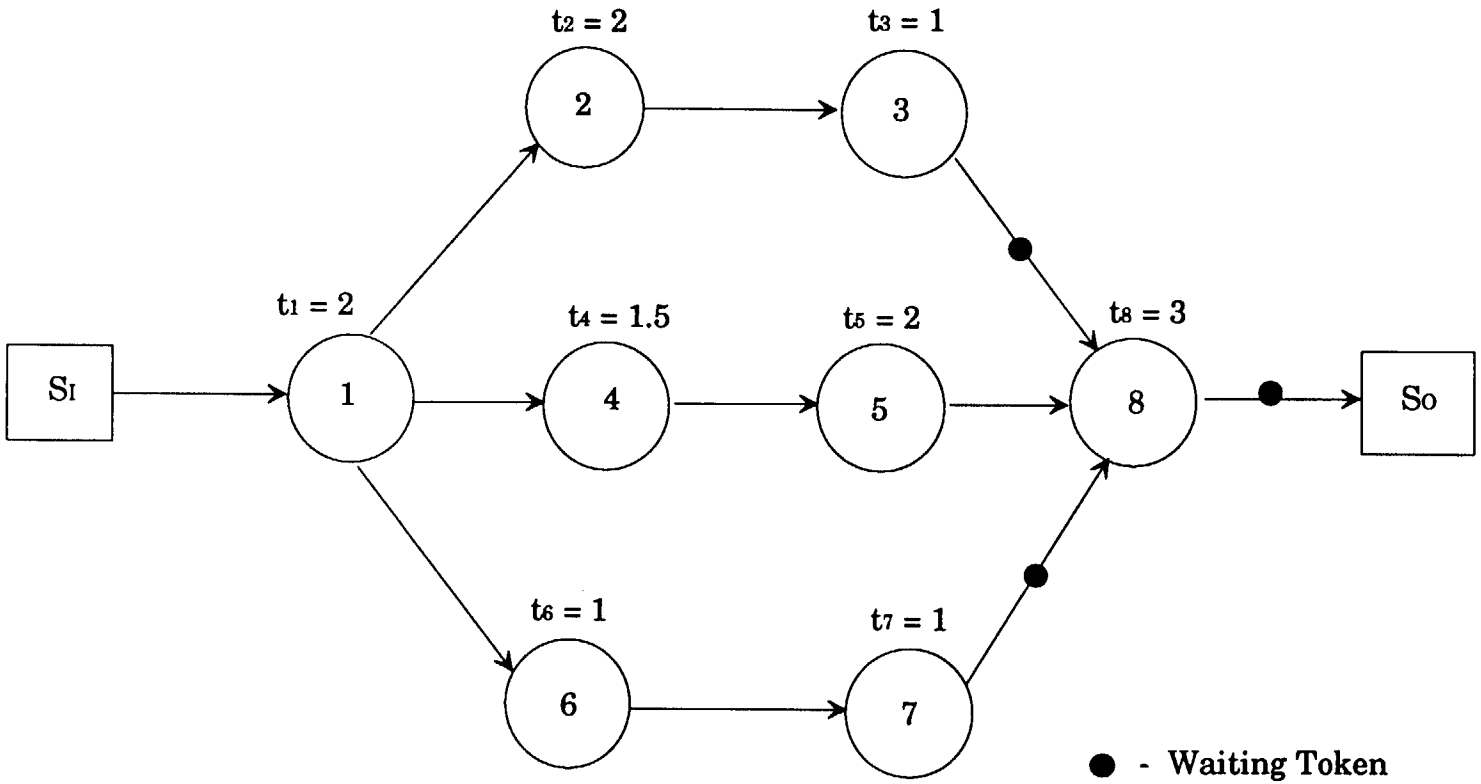


Figure 3.1. Example AMG to illustrate the concept of waiting tokens.

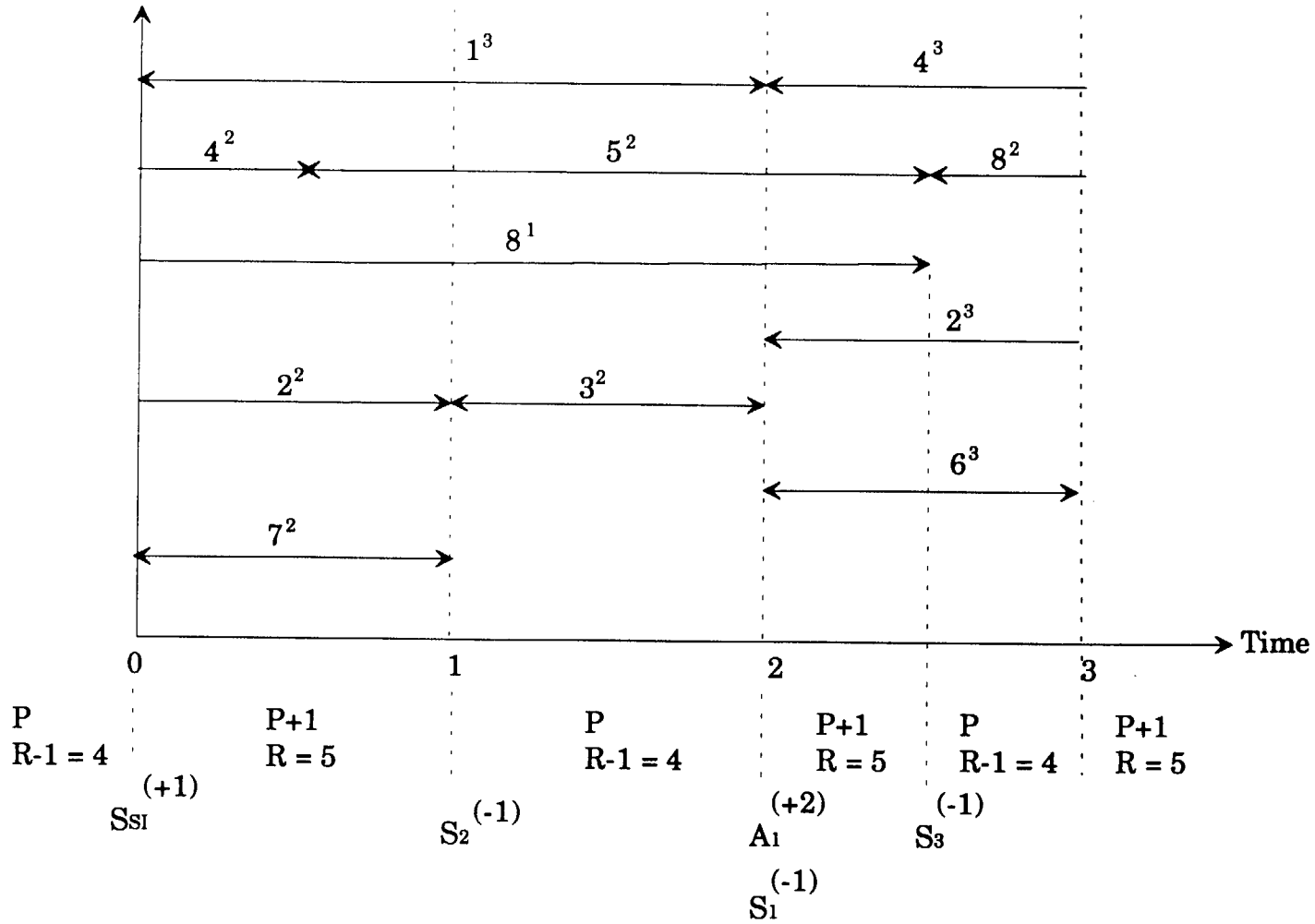


Figure 3.2. The TGP diagram for AMG of Figure 3.1 with TBO = 3.

are associated with resource changes (either increase or decrease) in the TGP diagram. The k-boundaries are helpful in determining the analytical view of the TGP diagram (in a view of number of resources) for a given graph, which gives an analytical model for the resource envelope.

Let us define the following notations:

N = Total Number of nodes in the AMG

m = Number of edges outgoing from each node

T = Number of edges on which tokens wait out of "m" edges from each node. The variable T ranges from 0 to m.

k = Net change in number of resources immediately after a node finishes. "k" ranges from -1 to m-1.

For any node, if T = m, token waits on each outgoing edge. The resource utilized by this node is not used immediately by any other node when the node completes. Thus, there is a net decrease by one in number of resources immediately after the node completes. The net decrease by one is represented by negative k value, which is equal to -1.

If T = m-1, tokens wait on all except one outgoing edge. When the node finishes, a resource used by this node is utilized by its successor which fires immediately. Thus, there is a replacement of resource, and the net change in number of resources is zero (k = 0).

If T = 0, all the "m" successors of the node fire at the same time when a node completes, utilizing "m" number of resources. However, one resource

is freed by the predecessor. This causes a net increase of m-1 in number of resources.

If T = m-2, tokens wait on all except two outgoing edges. When the node completes, two nodes are fired immediately. Therefore, one resource is replaced by two, and the net increase is 1.

For a given m and T, we may summarize:

$$k = m - T - 1,$$

which is valid for any value of m and T.

3.2.3 Resource Use Boundaries and Resource Envelope

K-value is the value of k associated with each k-boundary. The concept of k-values and k-boundaries is useful in further analytic development of the resource envelope and the variable node time model. In the previous section, we have identified the k-boundaries for any given AMG. In this section, we will utilize k-boundaries to define S_i and A_i boundaries, and to develop an analytical form of the resource envelope.

Out of all the k-values (one for each node), only non-zero values of k are to be considered to establish a "resource use boundary" because only these values of k cause a net change in the number of resources. The trivial case where k = 0 causes a replacement of resource, so it is not considered in forming resource use boundaries.

The following information has now been identified:

1. All the non-zero k-values and a node associated with each of them.
2. The longest path from the input source to the node corresponding to each non-zero k-value.

From (2) above, we can find corresponding modulo-TBO (mod TBO) values using-

Mod TBO(path length) = Remainder(path length/TBO).

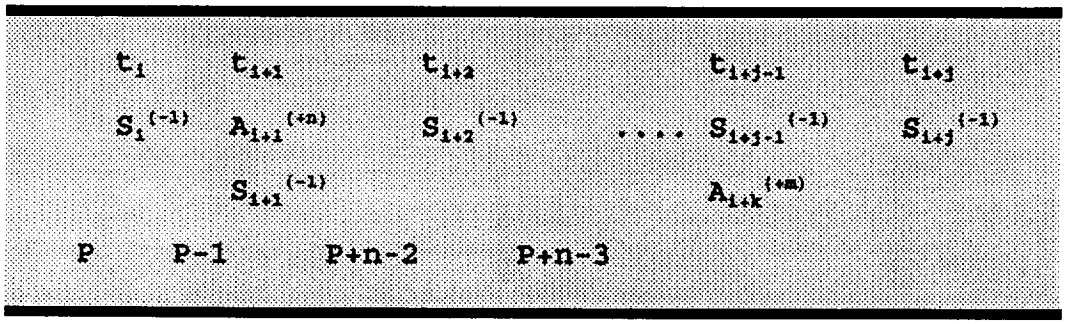
As discussed in Chapter 2, the mod TBO operation on the path length gives the relative path length within one TBO time interval in the TGP frame. It gives the time at which the path ends in the TGP diagram. By definition, mod TBO value lies in the interval [0, TBO). Define

S_i = Boundary across which k = -1 (Subtraction), and

A_i = Boundary across which $k \geq 1$ (Addition).

Each S_i and A_i has a k-value associated with it. Also they represent boundaries in the TGP diagram across which the number of resources change by a value of k. Having found mod TBO values for each S_i and A_i, these values may be ordered in an increasing manner, from 0 to TBO$^-$. In other words, the S_i and A_i boundaries are positioned in an increasing order in the TBO time frame, incorporating k-value for each boundary as a superscript. We now have enough information, in the form of an ordered array, from which the number of resources utilized at any time in the TBO time frame can be found out as outlined in the following.

Let us consider a general form of a sequence of S_i and A_i boundaries, as shown below.



In the ordered list shown above, m and n are known, and subscripts i, j, and k are used to show generalization. It is important to note that there is no net increase or decrease of resources across one TBO time frame in the TGP diagram, i.e., a resource can neither be created nor be destroyed.

Assume that the number of resources (R) on the very start of the TGP diagram (on the immediate right of t = 0 in the TGP diagram) is equal to P. S_i causes a decrease in number of resources by 1, so R = P-1 on the immediate right of S_i. $A_i^{(+n)}$ and $S_{i+1}^{(-1)}$ overlap each other. There is a net increase of n-1 across these boundaries since A_i causes an increase by n and S_{i+1} causes a decrease by 1. Therefore, R = P-1+n-1 = P+n-2 on the immediate right of S_{i+1}. Similarly, R = P+n-3 on the immediate right of S_{i+2}. Following the same procedure, we find resource values in terms of P across each boundary in the ordered array. The maximum R value corresponds to R_{max}, and all other R values are indexed by R_{max}. For example, if R_{max} = P+n-2, then R = P-1 = R_{max}-n+1, R = P+n-3 = R_{max}-1, and so on.

It is important to note that the number of resources about minimum and maximum time values (about t = 0 and t = TBO) must be same due to the periodicity of mod TBO operation. In other words, the number of resources at $t = 0^-$ and $t = 0^+$ must be equal to the number of resources at $t = TBO^-$ and $t = TBO^+$, respectively. In general, boundary conditions are satisfied for any time slot (t = n to t = n+TBO) of width TBO. If we consider $t = 0^+$ and $t = TBO^-$ values in the current data packet, then $t = TBO^+$ and $t = 0^-$ may be considered as corresponding values in the next data packet.

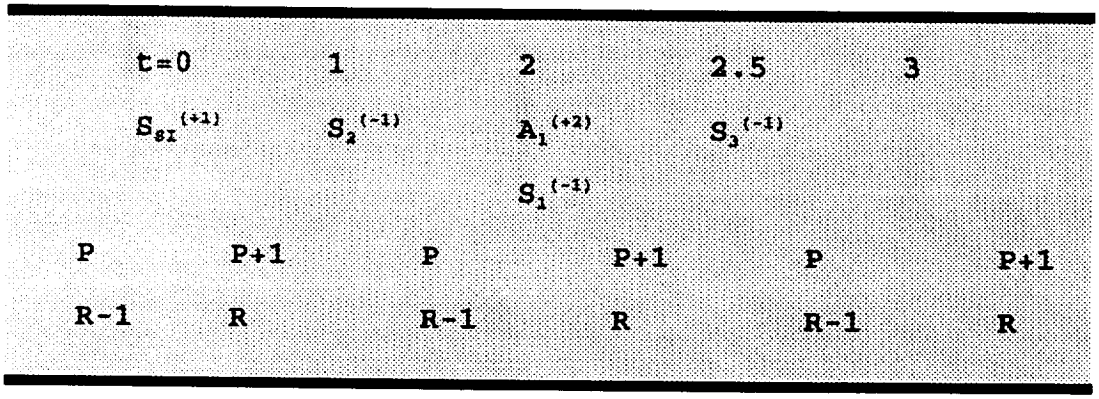
The number of resources at $t = 0^+$ can be determined from the number of resources at $t = 0^-$ since k-value(s) of any S_i or A_i boundaries, if it exists at $t = 0^-$ (or $t = TBO^-$), is known, and the number of immediate successor node(s) of the source is also known. The number of immediate successor node(s) of the source represents the number of new processor assignments at the beginning of each TGP time frame (at t = 0). If, in a given AMG, there are S_{SI} successors of the source, then the resource requirement at $t = 0^+$ is increased by S_{SI} as compared to the resource requirement at $t = 0^-$, in addition to either increase or decrease given by the sum of k-values of boundaries existing at $t = 0^-$. This may be written as-

$$R(t = 0^+) = R(t = 0^-) + S_{SI} + [\text{ k-values of boundaries at } t = 0^-],$$

where square bracket around the last term is used to identify that this term may or may not exist. Since the number of resources at $t = 0^-$ (or $t = TBO^-$) is

known, number of resources at $t = 0^+$ can be found using the above equation, which must be equal to the number of resources at $t = TBO^+$.

The ordered boundary information is now what otherwise would have been conveyed by the resource envelope, or the TGP diagram. For the AMG shown in Figure 3.1, this information is displayed below, and the A_i and S_i boundaries are marked in the TGP diagram of the AMG as shown in Figure 3.2.



It is noted that $R = R_{max}$. The resource requirement from $t = 0^-$ to $t = 0^+$ increases by one ($S_{SI}^{(+1)}$) because of one immediate successor (node 1) of source node. From the analytical model of the resource envelope, the value of R_{max} may be computed as will be discussed in the following section.

3.2.4 R_{max} from Worst Case Analysis

In this section, a method for computing the value of R_{max} from the analytical resource envelope, developed in the previous section, is presented (without drawing of the TGP diagram). The value of R_{max} corresponds to the

worst case analysis assuming that all the nodes in the graph take the maximum allocated time to complete. Define

n = Number of boundaries at distinct times in the array, or the number of "t" values (corresponds to S_i and A_i) in array.

t_i = Distinct "t" values corresponding to boundaries marked in the array. The maximum value of t_i is equal to TBO, and i = 0 to n.

R_i = Number of resources utilized in each time slot between two t_i.

t_n = Time at which right-most boundary is located in the array, or the time position of right-most k-boundary in the analytical resource envelope.

The summation of all node times in the AMG defines Total Computing Time or Total Computing Effort (TCE). It is observed that TCE events occur in one TBO period. Thus, TCE must be equal to the sum of product of the time duration of each slot in the TGP diagram and the corresponding number of resources utilized in a particular slot.

If the number of distinct k-boundaries is equal to n, then

$$TCE = \sum_{i=0}^{n} R_i \, [t_{i+1} - t_i] , \qquad (3.1)$$

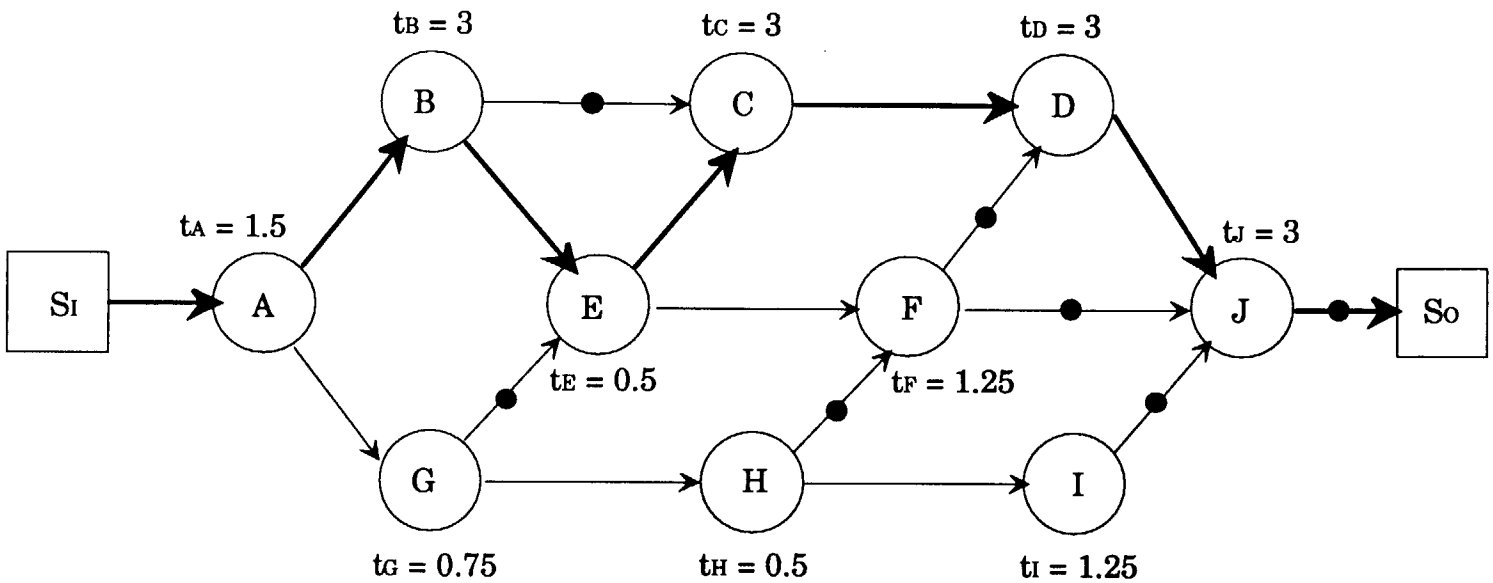
where $t_0 = 0$, and $t_{n+1} = TBO$. Since t_i and TCE are known, and R_i is known in terms of $R = R_{max}$, Equation 3.1 can be solved for the unknown $R = R_{max}$.

3.2.5 An Illustrative Example

As an illustration of the method for development of the resource envelope, consider the AMG shown in Figure 3.3. The critical path is A-B-E-C-D-J as shown by dark lines in the figure, and the critical path length is 14. Also, TCE = 17.75 and TBO = 3. To find the position of waiting tokens, consider each node which has two or more edges directed to it. Find the path length to the node for each incoming edge, and mark the waiting token on each incoming edge except one which has maximum value of the path length. If two or more incoming edges have equal maximum value, then mark a waiting token on all except any one edge.

For a given AMG, a waiting token is present on the edge from node G to node E, because node E cannot start until nodes B and G are finished, and the path length A-B is greater than the path length A-G, i.e., $(t_A + t_B) > (t_A + t_G)$. Similarly, at node F, $(t_A + t_B + t_E) > (t_A + t_G + t_H)$, therefore a waiting token is marked on the edge directed from node H to node F. Other waiting tokens are marked in the AMG by following similar argument. A waiting token is also marked on the edge from node J to the sink S_o, since the completion of node J releases one resource which is not utilized by the sink.

After marking all the waiting tokens in the AMG, we construct the resource envelope table as shown by Table 3.1. In this table, each row corresponds to each node in the AMG. The first column shows the nodes in the AMG, the second column gives the value of m (number of edges directed out



ABECDJ - Critical Path

● - Waiting Token

Figure 3.3. An example AMG for illustration.

Node	m	T	k=m-T-1	Boundary	Path Length	Mod TBO
A	2	0	1	A_1	1.5	1.5
B	2	1	0	-		
C	1	0	0	-		
D	1	0	0	-		
E	2	0	1	A_2	5	2
F	2	2	-1	S_1	6.25	0.25
G	2	1	0	-		
H	2	1	0	-		
I	1	1	-1	S_2	4	1
J	1	1	-1	S_3	14	2

Table 3.1. The resource envelope table showing k-values and boundaries for the AMG of Figure 3.3.

from each node), the third column gives the value of T (number of edges out of m which contains waiting tokens) for each node, and in the forth column the value of k is calculated for each node using k = m-T-1. In "Boundary" column, different resource boundaries are named, using the notation A_i (Addition) for positive value of k and S_i (Subtraction) for negative value of k. Trivial k = 0 value is discarded. In the next column the path length for each boundary is noted, considering the path without any waiting token. In the last column the mod TBO values of path lengths are found.

Table 3.1 shows k-value for each node, A_i and S_i boundaries, the path length from the input source to corresponding node, and mod TBO value of the path length for the AMG given in Figure 3.3. The A_i and S_i boundaries are ordered, using the corresponding mod TBO values and ordering in an increasing manner, as shown in Figure 3.4. Here, the superscript of a boundary represents its k value, and subscript represents boundary numbering. Assuming P resources at the left-most point in the TGP frame, we find resources in each time slot, knowing the amount of increase and decrease at each A_i and S_i boundary. These resource values are then indexed to $R=R_{max}$ as shown in the figure, since R_{max} is the number of resources for which the system is designed.

Figure 3.4 provides the complete information about the total resource envelope for the given graph. It is an analytical view of the resource envelope

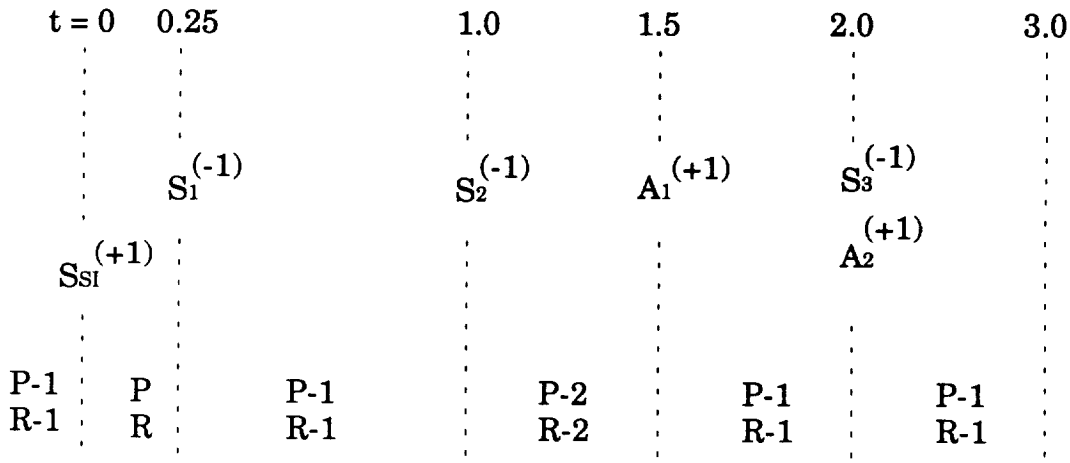


Figure 3.4. Ordered array of boundaries and resources in different regions for the AMG of Figure 3.3.

as opposed to the pictorial view. Using Equation 3.1 and information given in Figure 3.4, $R = R_{max}$ can be determined as follows:

$$TCE = 17.75 = R\,(0.25-0) + (R-1)\,(1.0-0.25) + (R-2)\,(1.5-1.0) + (R-1)\,(2.0-1.5) + (R-1)\,(3.0-2.0)$$

$$17.75 = 3R - 3.25, \text{ or}$$

$$3R = 21, \text{ or}$$

$$R = 7 = R_{max}$$

The analytical resource envelope may be compared with the pictorial view of the resource envelope obtained using the Design Tool, as shown in Figure 3.5. Comparing the value of resources in different time slots in Figure 3.4 with those in Figure 3.5, it is seen that theoretical results match with the experimental results. Also, the value of R_{max} obtained from Figure 3.4 and Figure 3.5 are the same.

3.3 Development of the TGP Diagram

A method for the development of the TGP diagram from the analytical resource envelope is presented in this section. This method is important because it is easier to draw the TGP diagram for a very large graph, a graph with forward initial tokens, or a graph with circuits without any error. This method may be considered as an alternative method of drawing the TGP diagram to the previous method in which the TGP diagram is drawn using the SGP diagram by folding it around about one TBO time interval.

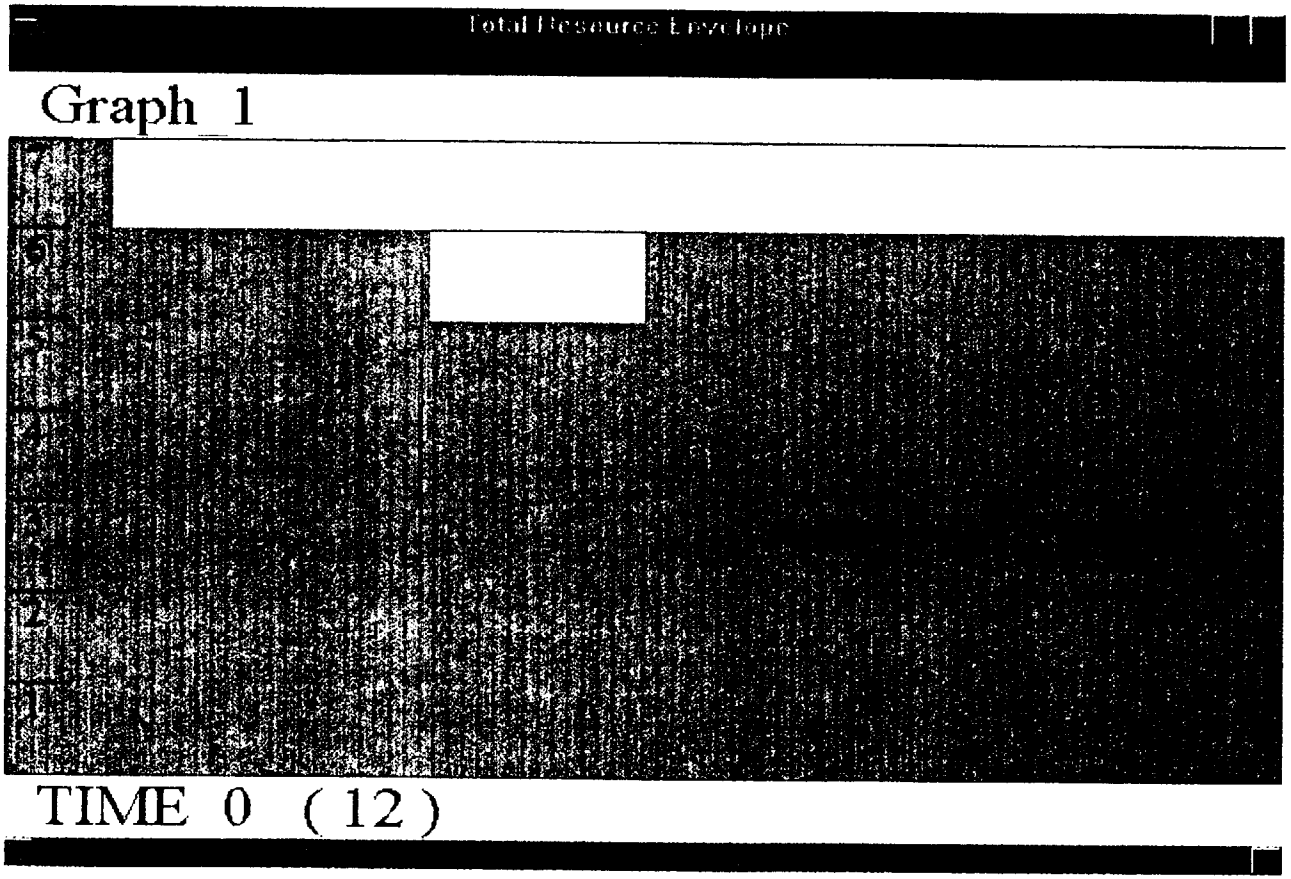


Figure 3.5. The TRE for the AMG of Figure 3.3 obtained using the design tool.

3.3.1 Method Development

As discussed in Chapter 2, the TGP diagram provides information regarding the predecessor-successor relationship for each node in the graph. It displays the execution of each node of a graph when the graph is operating periodically in steady-state with period TBO. The necessary information for the development of the TGP diagram may also be obtained directly from the analytical resource envelope developed in the previous section, since it provides the path length to each node, and also the time instant in one TBO frame (from mod TBO operation) which corresponds to the completion of a node.

The data packet dependency for each node relative to the current data packet is also represented in the TGP diagram. The data packet number for each node may be determined from the analytical resource envelope using the integer(path length) operation. In this operation, as defined by Equation 2.2, we divide the path length to a node by TBO, and take an integer value of the result. If we assume the current data packet number be i, then data packet corresponding to each node in the TGP diagram may be determined relative to data packet i by subtracting integer(path length) value for each node from i. As for example, if the path length to a node is 7, and TBO = 3, then integer(7/3) is equal to 2, and the data packet of this node relative to the current data packet i is i-2.

Since the time at which each node finishes in one TBO frame, and the data packet for each node may be determined from the analytical resource

envelope, as outlined above, we may construct the TGP diagram directly from the analytical resource envelope. In the analytical resource envelope, the A_i boundaries are formed due to forks, and the S_i boundaries are formed due joins in the graph. In finding the path length to each join node, times of all the nodes in the path are summed. Thus, each node in a graph is taken into account if we consider all S_i boundaries. An exception to this is when neither of the S_i boundary corresponds to the join immediately before the sink, in which case this join node is not taken into account. To incorporate this join node, we also consider the critical path in addition to all S_i boundaries, since the critical path also takes into account the left over join node.

To summarize, the TGP diagram for the AMG may be constructed using integer(path length) data of a node corresponding to each S_i boundary and the critical path, and a given graph. These provide information regarding data packet indexing and predecessor-successor relationship respectively.

3.3.2 Method Description

A method for the development of the TGP diagram from the analytical resource envelope is outlined in the following.

1. Since the path length for each node which contributes to an S_i boundary is known, its integer(path length) value may be found using Equation 2.2. If the path length is an integer multiple of TBO, i.e. path length = n TBO, then integer(path length) = n-1.

2. Assuming that the current data packet is i, data packet index for each of the above node is found by subtracting its integer(path length) value from i, i.e., data packet index for a node = current data packet i - integer (path length).
3. The critical path, TBIO, and TBO may be found directly from the given AMG.
4. Since mod TBO(path length) for each node contributing S_i is known (which gives the time at which the node finishes in one TBO frame), we draw the path ending at the time given by its mod TBO value, traversing backwards (going upwards in the TGP diagram) with a length equal to the corresponding node time, and considering each node in the path in a reverse manner until we reach at the starting node of the path. Also, mark the last node in the path (node contributing to S_i) with a packet index of its corresponding integer(path length) value, and increase the packet index by one for nodes in each of the upper layers.
5. Repeat step 4 for each S_i boundary and the critical path, and ignore any duplication of a node in the TGP diagram.

3.3.3 An Illustrative Example

For a given AMG, the TGP diagram may be constructed using the method outlined in the previous sub-section. As an illustration of this method, consider the AMG shown in Figure 3.3, and its resource envelope table shown

in Table 3.1. From the last row in Table 3.1 (row for node J), it reveals that the critical path is taken into account in S_3 since node J is last node in the critical path. Therefore, the TGP diagram may be constructed using only S_i boundaries. Table 3.1 is modified as shown in Table 3.2, which is self-explanatory. This table is used to construct the TGP diagram as outlined in the following.

Consider first the boundary S_3. Its mod TBO and integer(path length) values are equal to 2 and 4, respectively. Therefore, in one TBO time frame, start drawing a line segment for node J at time = 2, packet number i-4, and going backwards (upwards in the TGP diagram) for $t_J = 3$. Then, predecessor of node J which does not deposit a waiting token on the edge directed to node J (in this case, node D) is drawn. Accordingly, line segments for nodes C, E, B, and A are drawn in a reverse fashion with appropriate data packet number.

Now, consider the boundary S_1 which constitutes the path A-B-E-F. From Table 3.2, it is seen that node F finishes at time = 0.25 in the TGP frame with a data packet of i-2. Therefore, draw a line segment for node F, starting at time = 0.25 with data packet number i-2, and going backwards. Since the line segments for nodes A, B, and E are already drawn using S_3, they are redundant, and are not considered. Thus, S_1 gives new information regarding only node F.

Consider the boundary S_2 which constitutes the path A-G-H-I, and located at a point where node I finishes. From table 3.2, node I finishes at

Node	Boundary	Path Length	Mod TBO	Int.(PL/TBO)	Relative Data Packet No.
F	S_1	6.25	0.25	2	i-2
I	S_2	4	1	1	i-1
J	S_3	14	2	4	i-4

Table 3.2. The modified resource envelope table for the construction of the TGP diagram of the AMG of Figure 3.3.

time = 1 with data packet of i-1. Therefore, draw a line segment of length t_I = 1.25 starting at time = 1 in one TBO frame, with data packet of i-1, and going backwards. Similarly, draw line segments for node H and node G of lengths t_H = 0.5 and t_G = 0.75, respectively. The line segment for node A is already drawn, so it is not considered. Thus, S_2 gives additional information in terms of nodes G, H, and I. The complete TGP diagram obtained by following the above procedure is shown in Figure 3.6.

This method gives the steady state TGP diagram for a given AMG, and is very useful for the AMG with high complexity.

3.4 Time Varying Nodes and Resource Limited Mode

In the Section 3.2, enough background has been developed in terms of the analytical resource envelope so that we may consider to develop the variable node time model, as presented in this section. This model is important because many real-time algorithms require variable node times for reasons such as heterogeneous processors, data dependent code execution, and time varying deadlines for the completion of a job. The execution of graphs with time varying nodes is investigated with a point of view of the change in number of resources. A method for the detection of a possible resource limited mode is discussed, and also a method for its subsequent prevention is outlined.

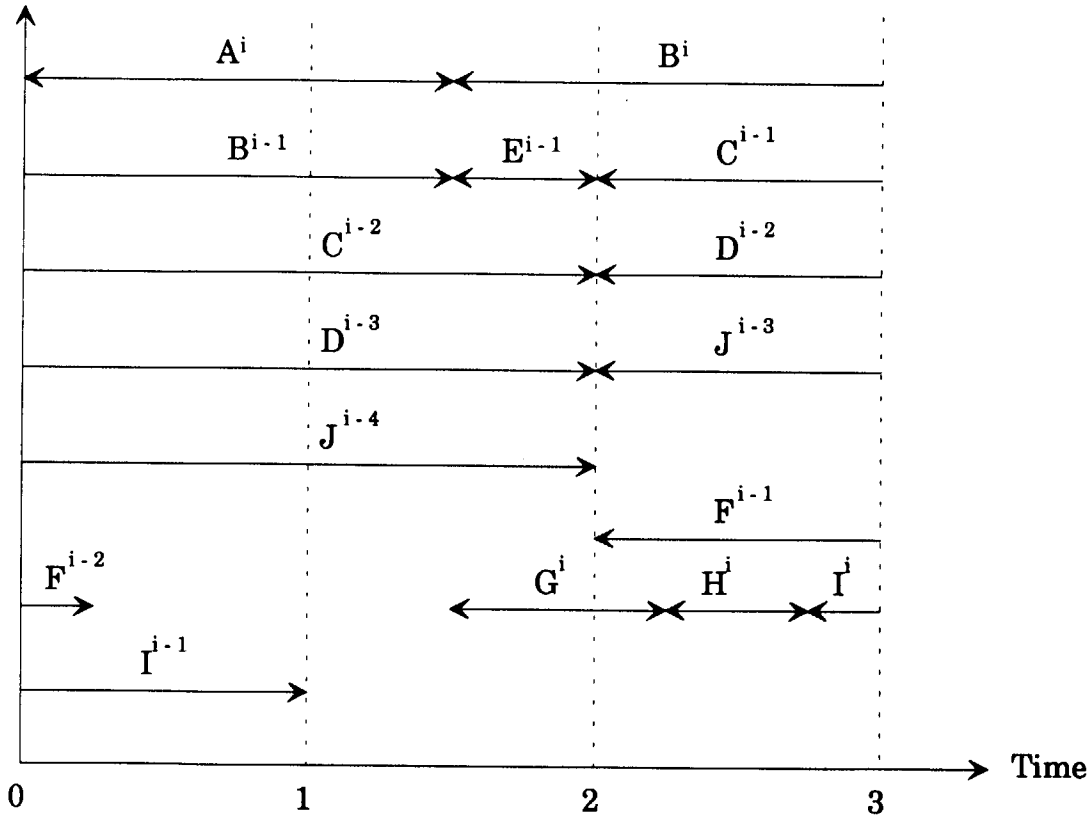


Figure 3.6. The TGP diagram for the AMG of Figure 3.3 constructed from the resource envelope table.

3.4.1 Resource Limited Mode

The multicomputer system is said to be running in resource limited mode if a resource required during algorithm execution is not available in the system. When a node finishes early, there may be an increase in resource requirements in excess of that required under fixed node time conditions. Generally the system is designed for the number of resources necessary for fixed maximum node times in the graph. Now if for some reason the node time decreases, which in turn may cause an increase in the resource requirement, then the system runs into resource limited mode. The resource limited mode of the system needs to be prevented because it causes the system time performance to become unpredictable. Therefore, a method will be developed in the following sections to detect the presence of resource limited mode and subsequently, to modify the graph in such a way as to prevent the system from going into the resource limited mode.

3.4.2 Graph Topology Considerations

The overall topology of the graph is an important contributor factor in resource limited mode. Thus, the presence of resource limited mode depends on the structure of the graph, as well as the timing of each node in the graph.

A node which has at least two immediate successors is called a "fork node". The combination of a fork node and its successors is called a "fork". The basic property of the fork is to increase the resource requirement in the

TGP frame to the immediate right of the boundary at which the fork node finishes. For example, consider a fork node with three successors. Let the number of resources to the immediate left of the point where fork node finishes be S. Then, the number of resources to the immediate right of this point equals S+2, since an increase by two number of resources is encountered across the boundary where the fork node finishes. If the fork node goes short, then the three successor nodes start early at a point in time beginning at the left of this boundary and hence two extra resources may now also be executing in addition to the S resources already being used. If the requirement for extra resources causes the number of resources required to be greater than R_{max} (the maximum number of resources for which the system is designed), then the system is driven into resource limited mode.

Suppose that without any node time variation (fork nodes finishing at maximum allowable time), the number of resources required is equal to R for which the system is designed. When the fork node finishes earlier than its scheduled finish time (under any node time variation), the number of resources required may become greater than R for short time duration. The system is driven into resource limited mode due to unavailability of extra resource(s).

Another important graph attribute of the AMG is a "join". The node at which token waits on at least one of the incoming edges is called a "join node". In other words, a join node may be defined as a node at which two or more edges are directed to it. When more than one edges terminate at a particular

node, tokens wait on all except one edge. The combination of a "join node" and its predecessors defines a "join". An important feature of a join is that the resource requirement may decrease where the join node starts in the TGP diagram.

The behavior of variable node time AMG is investigated in the following sub-sections in the view of resource limited mode, and its subsequent prevention. The approach taken is to observe the change in number of resources when each A_i boundary in the analytical resource envelope is shifted towards left, as outlined in the following section.

3.4.3 Necessary Condition for Resource Limited Mode

In this section, a necessary condition for the presence of resource limited mode is described. This will be useful in developing a method for the prevention of resource limited mode, as will be seen in the following section.

An $A_i^{(+k)}$ boundary is one to the immediate right of which the number of resources required increases by k in the TGP diagram. Consider a case of shifting an A_i boundary towards the left. (If any A_i boundary is located at time $t = 0$, its shift towards left is folded over at $t = TBO^-$ and then shifted to the left). This causes the requirement of k extra resources to the left of A_i in the TGP diagram. If in any time period over which A_i may be shifted in the TGP diagram, the maximum resource requirement appears to be greater than or equal to R_{max}-k+1, then the requirement of k extra resources increases the total

number of resources to be greater than or equal to $R_{max}+1$. However, the system is designed for R_{max} resources, which is now less than the peak resource requirement. Therefore, the system is driven into resource limited mode.

Thus we may conclude that for any A_i boundary, if decreasing the path length corresponding to A_i over possible range causes the total resource requirement to be greater than R_{max}, then the system is driven into resource limited mode. A cause of decrease of a path length may be non-homogeneous resources, variable node times, data dependent code, and many others.

3.4.4 Prevention of Resource Limited Mode

In this section, sufficient conditions are developed for inserting the control edges for eliminating resource limited mode due to node time variations.

When an A_i boundary shifts towards the left, it may or may not cause resource limited mode. We first develop a method of determining whether an A_i boundary may possibly cause resource limited mode, and then insert necessary control edges to prevent resource limited mode. Define

$+k$ = Increase in number of resources across A_i,

j = Index relative to $R = R_{max}$, used to represent resources utilized in different time slots, and

t_i = Time location of different boundaries in one TBO time frame.

For each A_i, observe if moving A_i across S_i causes the resource requirement to be greater than R_{max}. If it does, find the amount of increase beyond R_{max}, and denote it by R_{inc}. Then,

$$R_{inc} = (R_{max} - j + k) - R_{max} = k - j,$$

where k = k-value for A_i under consideration, and j = resource index on the immediate left of S_i (which is to be crossed).

The number of control edges necessary for any A_i boundary, # CE, is given by,

$$\#\ CE = R_{inc} - (\#\ CE \text{ already determined for } A_i \text{ under consideration}).$$

As an example, consider the array shown in Section 3.2.3. The number of control edges are determined as follows.

For A_1 (t = 0 to 1, k = 3), # CE (S_3, j = 2) = 3 - 2 - 0 = 1, and # CE (S_1, j = 1) = 3 - 1 - 1 = 1.

For A_2 (t = 0 to 2.5, k = 2), # CE (S_4, j = 1) = 2 - 1 - 0 = 1, and # CE (S_2, j = 0) = 2 - 0 - 1 = 1.

The upper limit of the sum of # CE for any A_i is equal to the k-value of that particular A_i, which is equal to 3 and 2 for A_1 and A_2 respectively.

3.4.5 Placement of Control Edges

In this section, a procedure for inserting control edges for the prevention of resource limited mode is outlined. After finding the required number of

control edges as described in the previous section, we insert them appropriately in the AMG using the method outlined below.

1. Identify the successor nodes (S_NODES_{Ai}) of a node in the AMG which corresponds to an A_i boundary.
2. Identify a node in the AMG ($NODE_{Si}$) which corresponds to an S_i boundary. The number of control edges required for an S_i is found to be equal to # CE (S_i).
3. Insert control edge(s) from $NODE_{Si}$ to one or more S_NODES_{Ai}, the number being equal to # CE (S_i). The S_NODES_{Ai} to which control edges are directed from $NODE_{Si}$ won't start their execution until $NODE_{Si}$ finishes. This prevents the resources to be utilized by them from moving to the left of the S_i boundary.
4. Following the same method for each A_i - S_i combination determined above, resource limited mode may be prevented under any node time variation.

The insertion of control edges may form circuits in the graph, and therefore changes the value of TBO. To retain the original value of TBO, we increase the queue size for each control edge. Moreover, initial tokens must be appropriately placed on a control edge for the initiation of graph play.

3.4.6 An Illustrative Example

As an example, consider the AMG given in Figure 3.3. The behavior of this AMG with any node time variation is investigated in this section in a view of resource limited mode and its prevention.

Consider the analytical resource envelope as shown in Figure 3.4. Consider shifting of A_1 boundary to the left. Since A_1 is contributed by node A, $t_A = 1.5$, and mod TBO = 1.5, the range of variation of A_1 is between t = 0 and t = 1.5, as shown in Table 3.3. Now if A_1 is shifted in the region between t = 0 and t = 0.25 (in which number of resources = R = R_{max}), or if the S_1 boundary is crossed, extra resource requirement due to A_1 causes total resources requirement to be greater than R_{max}, therefore resource limited mode occurs. This may be prevented by inserting a control edge directed from the node which contributes S_1 (node F) to any one successor of the node which contributes A_1 (node B or node G).

Now consider shifting of A_2 boundary to the left. Since A_2 is contributed by node E, and $t_E = 0.5$, and mod TBO = 2, the range of variation of A_2 is between t = 1.5 and t = 2, as shown in Table 3.3. In this time region, the number of resources are R_{max}-1, therefore the increase of one in number of resources if A_2 is shifted to the left causes the resource requirement to be R_{max}. Therefore shifting A_2 won't cause resource limited mode.

To summarize, the number of control edges for all the S_i boundaries in the range of variation of each A_i boundary are found as follows:

	t_0=0	t_1=0.25	t_2=1.0	t_3=1.5	t_4=2.0
Boundaries	-	S_1	S_2	A_1	S_3 A_2
Latest start of A_i = Mod TBO [t{A_i} - t{$NODE_{Ai}$}]	-	-	-	1.5 - t_A =1.5 - 1.5 =0	2.0 - t_E =2.0 - 0.5 =1.5

Table 3.3. Finding the possible range of variation of A_i boundaries in one TGP frame for the AMG of Figure 3.3.

For A_1 (k = 1): S_2 (j = 1) = 1 - 1 - 0 = 0

S_1 (j = 0) = 1 - 0 - 0 = 1

For A_2 (k = 1): S_3 (j = 1) = 1 - 1 - 0 = 0

Thus, one control edge is required for preventing a possible resource limited mode, which may be inserted starting from node F directed to either node B or G, with appropriate initial token and queue size. The AMG with a control edge from node F to node G is shown in Figure 3.7.

3.5 Conditional Node Model

An overview of the conditional node model is presented in this section. An algorithm graph with conditional nodes is studied, and a possible reduction of a conditional node graph to a time varying node graph is described. Basic modifications necessary for mapping the conditional node graph on the ATAMM based architectures are presented, and finally an example of conditional node graph is given in which conditional node graph is investigated using the variable node time model.

It is assumed that each conditional node has a capability of making decisions depending upon certain data conditions. This capability is derived from the executable code embedded in the node itself. It is also assumed that only one of the several outgoing paths is chosen for execution by each conditional node, and there is no restriction on the number of conditional nodes in an AMG. Also, the probability of execution of any conditional path, in one

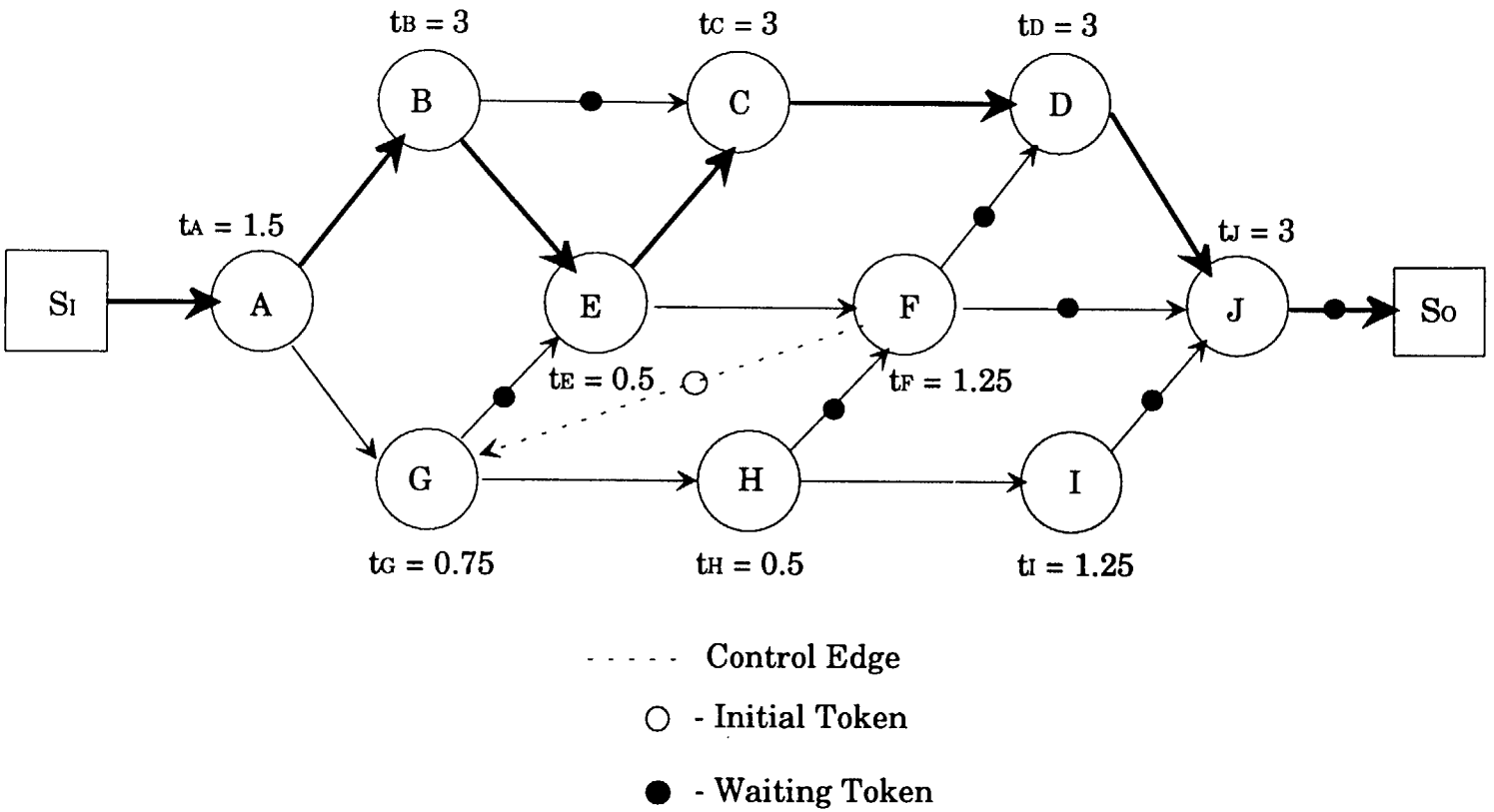


Figure 3.7. The AMG of Figure 3.3 with a control edge inserted to prevent resource limited mode.

particular run of the AMG, is moderately greater than zero, and all conditional paths are executed at nearly same frequency. The decision making process resembles branching such as a "case" statement.

3.5.1 Overview

In many practical applications, it is necessary to change the flow of execution of algorithm depending on the existence of certain condition(s). For example, a portion of the algorithm may change from one range of the data values to another. For these type of applications, the AMG implementation of an algorithm may contain one or more conditional branches (forks). For any conditional branch, only one of the several possible alternative paths is executed when a particular data condition occurs, all other paths being inactive. Also, a jump from one conditional path into the other is not allowed. Intuitively, the conditional branch has at least two alternative paths. Each alternative path may contain any number of nodes in it. A node which initiates a conditional branch is called the conditional (fork) node. When a conditional node finishes, its data output is given to all of its successors. Now, depending upon the data conditions at the output of conditional node, any one of its successors starts its execution and it continues for the rest of the path. The remaining successor nodes (for which data condition is not satisfied) do not execute, but they just pass the data packet to their corresponding successors in the path.

Since only one of the several nodes at the same level in a conditional branch is executed, it is possible to combine these node into a single node by grouping the executable code of each of these nodes in one node. Such a new node may be viewed as a node with partitions, each partition containing the code for one particular node, and any one of these partitions is executed in the presence of any data conditions.

A partitioned node is assigned a node time which is found using a notion of latest time to deposit a token at the output of a node. The latest time to deposit a token at the output of a node is the longest path length from the input source to the node. For a reduced graph with partitioned nodes, the latest time to deposit is the maximum of all deposit times, one for each partition. An equivalent node time, which is assigned to a partitioned node, is found by subtracting the latest time to deposit at the input of the node from the latest time to deposit at the output of the node. The latest time to deposit at a particular edge represents the time, under the worst case, at which a token may be deposited on the edge, when a conditional node AMG is executed. Thus, the notion of latest time to deposit ensures a worst case analysis of the execution of a conditional node graph. (A notion of latest time to deposit is illustrated in Section 3.5.3, using a conditional node AMG). The TCE is calculated by summing node times of all nodes in a reduced graph. The TCE found from a reduced graph (instantaneous TCE) is used in the calculation of R_{max} from the analytical resource envelope.

A new node containing different partitions has to make a decision as to which of its partitions is executed each time data is available at input of the node. Therefore, it requires some basis to make this decision. One criteria may be let the new node figure out in which execution block the conditional node was in when output became available by viewing the status of some flags. This criteria is time consuming and requires backward status information. An easier and feasible approach may be to have the predecessor node provide necessary information to make a decision. This may be done by incorporating a status flag with each data packet, the value of which is determined by the node outputing data packet in such a way as to reflect the partition number, the code in which is to be executed next. Again, value of the status flag depends on which of the several conditions has caused the output. Now each node in the conditional branch checks the status flag when data packet is available at its input, and executes the code in the partition reflected by the status flag.

Since each node in the conditional branch has different partitions containing different pieces of code, the node may take different time each time it is executed. Therefore, each node in the conditional branch may be viewed as a time varying node with a conditional execution.

3.5.2 Example for Illustration

Consider the conditional AMG shown in Figure 3.8. Assume that the fork at node A is a conditional fork, and the top two paths (paths B-C and D-E) are conditional. Therefore, one of the two possible paths is executed at any time. The conditional AMG may be reduced to the AMG without any conditional branching as shown in Figure 3.9. Here, node X is divided into two partitions, which contain code for the successor nodes of a conditional node A (B and D). Node Y also has two partitions; partition 1 and partition 2 contains code for node C and node E, respectively. Partitions in a node are separated by dotted lines within the node. Thus, four nodes in the conditional branch shown in Figure 3.8 (nodes B, C, D, and E) are reduced to two nodes (nodes X and Y), as shown in Figure 3.9.

In Figure 3.8, node F is common to both the conditional paths, i.e., node F is always executed irrespective of the conditional path selected. Therefore, node F is shown, in Figure 3.9, as a successor of node Y without any partition, so that it is executed as soon as data at its input from either of the partitions in node Y is available. Under a worst case, fire time of node F is equal to the latest time to deposit at the input of node F, which is equal to 7 as shown in Figure 3.9. Latest deposit times at different edges are shown along corresponding edges. Node times of nodes X and Y are found as follows:

t_X = latest deposit time at output of X - latest deposit time at input of X.

= 7 - 4 = 3.

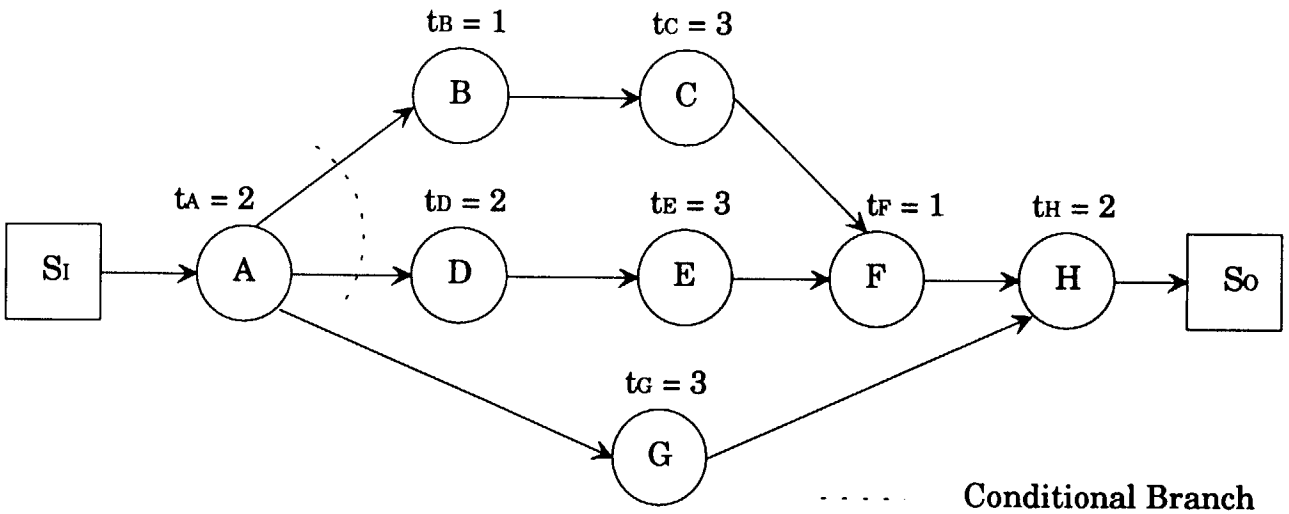


Figure 3.8. An example of the conditional node AMG.

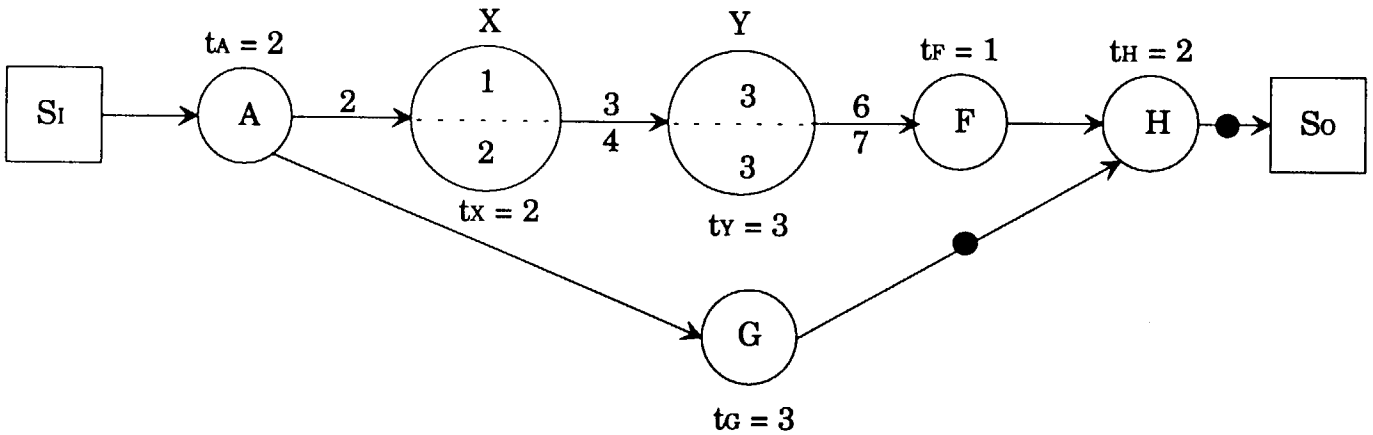


Figure 3.9. Reduced graph of the conditional node AMG shown in Figure 3.8.

Similarly, $t_Y = 4 - 2 = 2$.

The edge from node A to node G is unconditional. Therefore, node G is always executed as soon as node A finishes. Also, node H is always executed when both node G and node F deposit tokens at the input of node H. Nodes G and H in Figure 3.8 are equivalent to those in Figure 3.9.

After reduction, the conditional sub-graph is converted to a simple chain sub-graph, as shown in Figure 3.9. The number in each partition of a node represents the time to execute particular partition, and it corresponds to the appropriate node time. One advantage of this reduction is that the reduced graph is simpler to analyze for resource requirements. The second advantage is that the variable node time model may be utilized to obtain resource requirement of a conditional node graph and to analyze conditional node graph for resource limited mode. However, if node E is removed from the AMG of Figure 3.8, then a reduced graph will have a partitioned node with its second partition empty. Since some partitioned nodes may have null partitions, an extra overhead is present due to unnecessary firing of the node, even for a null partition.

The reduced equivalent AMG, shown in Figure 3.9, does not contain any conditional branch or node, and it may be considered as the AMG with time varying nodes, since partitioned nodes may take different execution times. For this AMG, the analytical resource envelope, maximum resource requirement,

and behavior under node time variation may be found using the variable node time model as described in the following.

The waiting tokens are marked in the AMG as shown in Figure 3.9. Also, TBO = 3, TBIO = 10, the critical path is $(A\text{-}X\text{-}Y\text{-}F\text{-}H)_2$ where subscript 2 represents the partition number for the critical path, and TCE = 13 (This is the maximum instantaneous value of TCE considering $t_X = 2$ and $t_Y = 3$). The resource envelope table, showing k-values, and S_i and A_i boundaries is shown in Table 3.4. In Figure 3.10, these boundaries are ordered in one TBO time frame, and resources in each time slots are calculated. This figure represents the analytical resource envelope. The value of $R = R_{max}$ can be determined as follows using Equation 3.1:

TCE = 13

$= R\,(1\text{-}0) + (R\text{-}1)\,(2\text{-}1) + (R\text{-}1)\,(3\text{-}2)$

$= 3\,R - 2$, or

$R = 5 = R_{max}$

Now consider node time variation in the AMG of Figure 3.9. The possible range of variation of A_1 boundary is found, as shown in Table 3.5. If A_1 is shifted in the time slot in one TBO frame between $t_0 = 0$ and $t_1 = 1$, then the increase in number of resource by one in this slot causes the total resource requirement to be greater than R_{max}. Therefore, decrease in node time of node A may cause resource limited mode. The control edges required to prevent resource limited mode may be found as follows:

Node	m	T	k=m-T-1	Boundary	Path Length	Mod TBO
A	2	0	1	A_1	2.0	2.0
X	1	0	0	-		
Y	1	0	0	-		
F	1	0	0	-		
G	1	1	-1	S_1	5.0	2.0
H	1	1	-1	S_2	10.0	1.0

Table 3.4. The resource envelope table for the AMG of Figure 3.9.

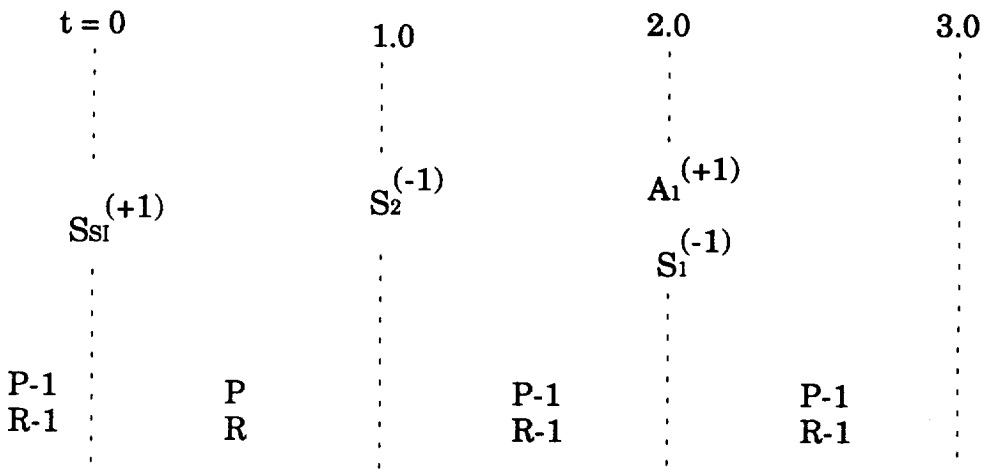


Figure 3.10. The analytical view of the resource envelope for the AMG of Figure 3.9.

	t_0=0	t_1=1.0	t_2=2.0
Boundaries	-	S_2	S_1 A_1
Latest start of A_i = Mod TBO [t{A_i} - t{$NODE_{Ai}$}]	-	-	2.0 - t_A =2.0 - 2.0 = 0

Table 3.5. Finding the possible range of variation of A_i boundaries in one TGP frame for the AMG of Figure 3.9.

For A_1 (k = 1, and variation between 0 and 2):

S_1 (j = 1) = 1 - 1 - 0 = 0

S_2 (j = 0) = 1 - 0 - 0 = 1

Thus, one control edge is required to prevent resource limited mode. This control edge may be directed from the node contributing to S_2 (node H) to any one successor of node A (either node X or node G).

3.5.3 Example to Illustrate the Latest Time to Deposit in Conditional Graphs

Consider the conditional AMG shown in Figure 3.11. It contains a conditional branch with two paths, and each path has three nodes. The corresponding reduced graph with unconditional nodes is shown in Figure 3.12. Here, nodes B and E are combined in node X, nodes C and F are combined in node Y, and nodes D and G in node Z. Times to deposit from the source to the output of nodes A, X, Y, and Z are shown in Figure 3.12. The latest time to deposit at each edge is the maximum of all deposit times shown on the edge.

Node times of nodes X, Y, and Z are found as follows:

t_X = latest deposit time at output of X - latest deposit time at input of X

= 200 - 100 = 100

Similarly, t_Y = 350 - 200 = 150, and t_Z = 600 - 350 = 250.

Node times of all other nodes remain unchanged. From the reduced graph, the TCE is equal to 1000.

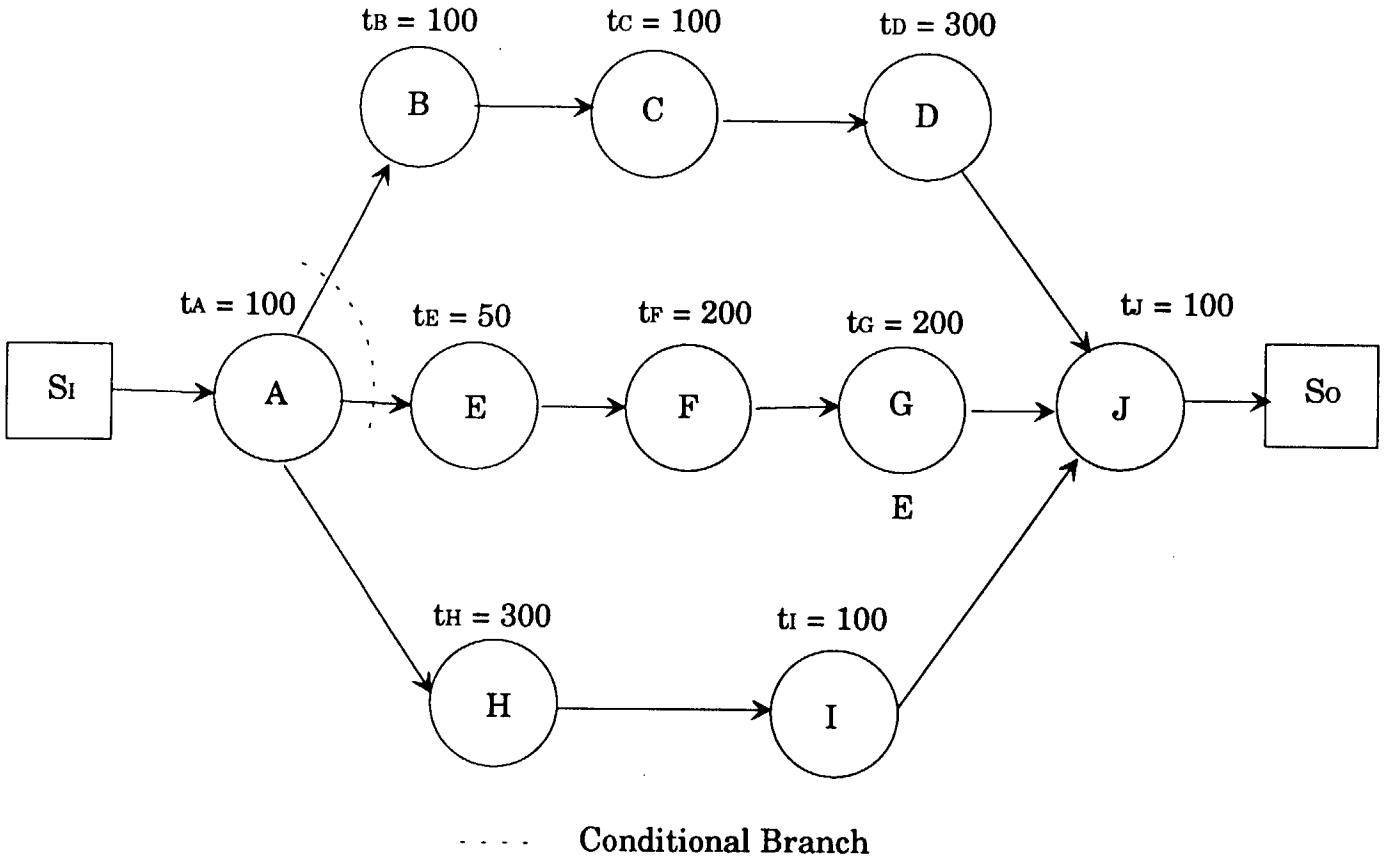


Figure 3.11. A conditional node AMG to illustrate the latest time to deposit.

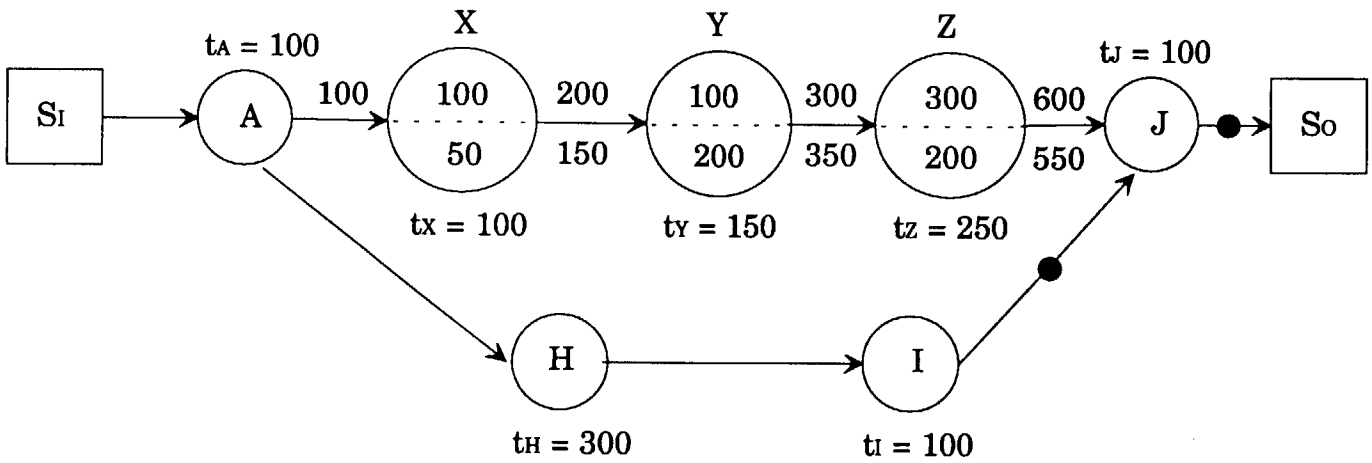


Figure 3.12. Reduced graph of the conditional node AMG shown in Figure 3.11.

A consideration of the latest time to deposit in finding equivalent node times of the partitioned nodes provides a worst case analysis of conditional node graphs, and resource requirements for the execution of conditional node graphs may be found under a worst case.

CHAPTER FOUR

CASE STUDIES THROUGH SIMULATION/EXPERIMENTS

4.1 Introduction

In this chapter, case studies are performed on three different example AMGs to illustrate the applicability of the resource utilization model and the variable node time model. In Section 4.2, an AMG with three parallel paths is investigated. The analytical resource envelope is evaluated, and conditions for resource limited mode and its prevention are found. The AMG is simulated and TREs are obtained under different AMG conditions for comparison with the theoretical results. In Section 4.3, a graph with a circuit is investigated under node time variation, and the same results are obtained as in Section 4.2. A conditional node graph is studied in Section 4.4.

4.2 Case Study - I: A Graph with Three Parallel Paths

An AMG with three parallel paths, and node time variation is investigated in this section. The AMGs with parallel paths are important because of the possibility of high concurrence in the execution of tasks. However, they possess a potential of resource limited mode under node time variation. The AMG used in the first case study is shown in Figure 4.1. This AMG has three parallel paths. Node times are allowed to vary to values less

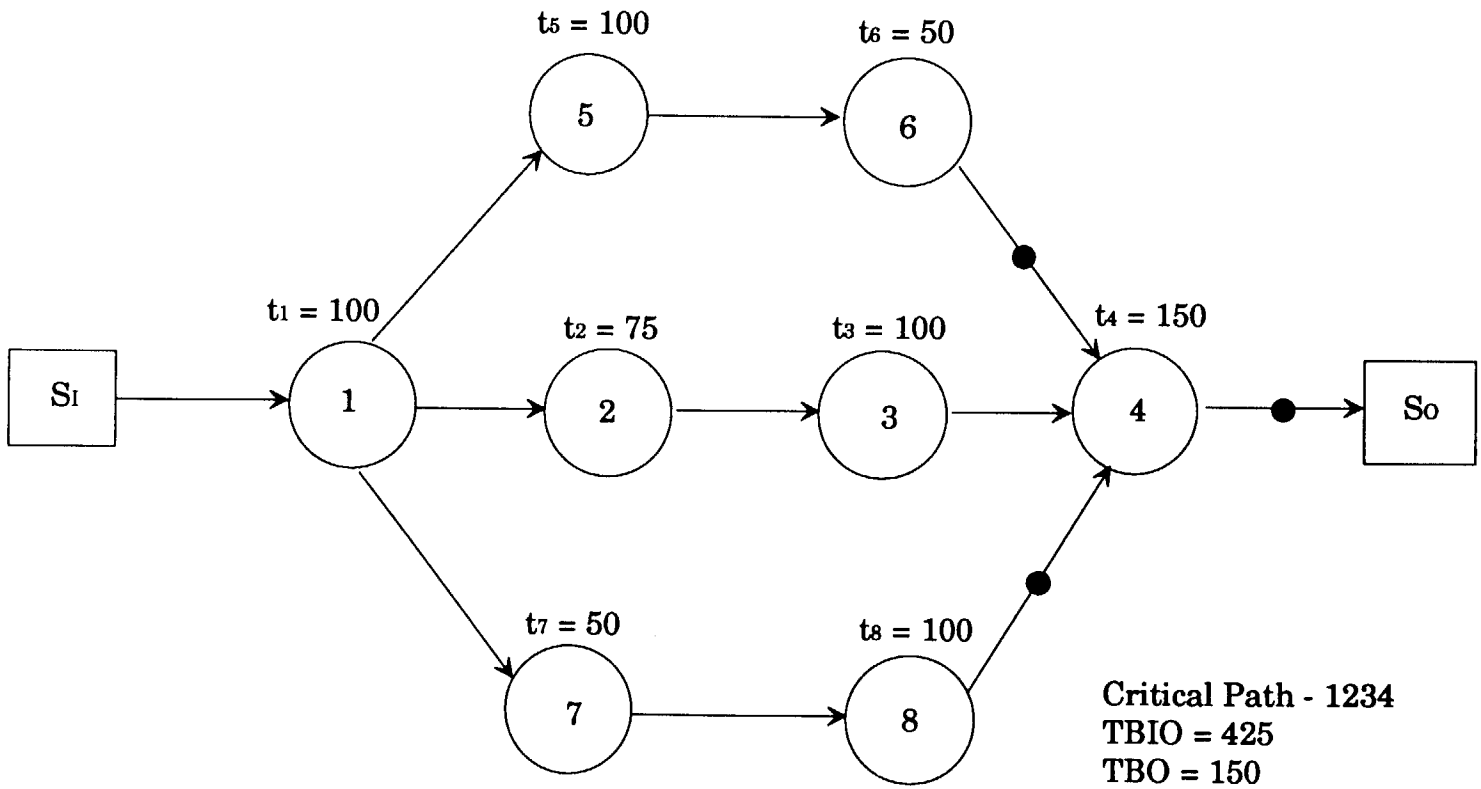


Figure 4.1. Example AMG for case study I with three parallel paths.

than the maximum value assigned to the node. Performance and operating conditions are based on these maximum node times.

For the AMG shown in Figure 4.1, the critical path is 1-2-3-4, and the critical path length is 425. Also, TBO = 150 and TCE = 725. The waiting tokens are marked in the AMG as shown in Figure 4.1.

The k-value for each node, significant k-values (those forming A_i or S_i boundary), the path length from input to corresponding node, and mod TBO value of the path length are presented in Table 4.1. In Table 4.2, these boundaries are ordered with respect to mod TBO values (t_i), and the allowable range of variation of A_i is calculated (with respect to one TBO frame). Assuming P resources at the left-most point in the TGP frame, number of resources in each time slot are found since the amount of increase and decrease at each A_i and S_i boundary is known. Figure 4.2 shows an analytical form of the TRE. From Figure 4.2 and using Equation 3.1, $R = R_{max}$ can be determined as follows:

$$TCE = 725 = R(100-0) + R(125-100) + (R-1)(150-125)$$

$$= 150\ R - 25$$

or, $R = R_{max} = 5$

From Figure 4.2, if A_1 is shifted to the left, it causes the resource requirement to be equal to R+2. Consequently, the system is driven into resource limited mode if the node time of node 1 falls below 100.

Node	m	T	k=m-T-1	Boundary	Path Length	Mod TBO
1	3	0	2	A_1	100	100
2	1	0	0	-		
3	1	0	0	-		
4	1	1	-1	S_1	425	125
5	1	0	0	-		
6	1	1	-1	S_2	250	100
7	1	0	0	-		
8	1	1	-1	S_3	250	100

Table 4.1. The resource envelope table showing k-values and boundaries for the AMG of Figure 4.1.

	t_0=0	t_1=100	t_2=125
Boundaries	-	S_3 S_2 A_1	S_1
Latest start of A_i = Mod TBO [t{A_i} - t{$NODE_{Ai}$}]	-	100 - t_1 = 100 - 100 = 0	-

Table 4.2. Finding the possible range of variation of A_i boundaries in one TBO time frame for the AMG of Figure 4.1.

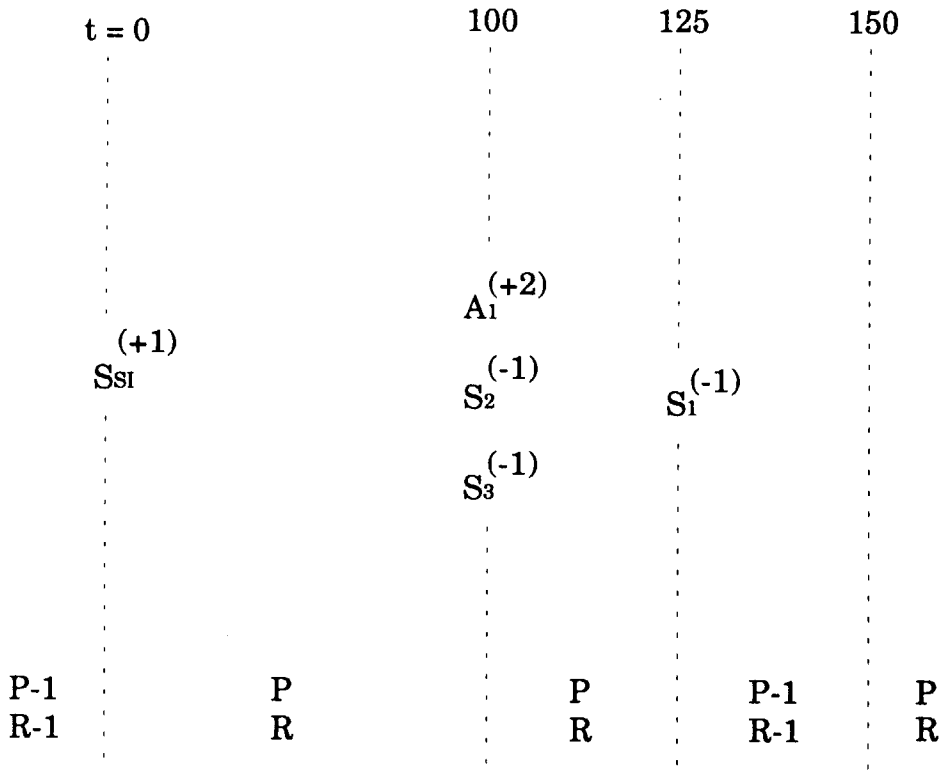


Figure 4.2. Ordered array of boundaries and resources in different regions for the AMG of Figure 4.1.

The number of control edges for all S_i boundaries in the range of variation of A_1 boundary to prevent resource limited mode are found as:

For A_1 (k = 2); S_2 (j = 0) = 2 - 0 - 1 = 1, and S_3 (j = 0) = 2 - 0 - 1 = 1.

Thus, two control edges are required to prevent a possible resource limited mode, one for both S_2 and S_3. The first control edge may be inserted from the node corresponding to S_2 (node 6) to any one successor of node 1 (node 5), and the second control edge may be inserted from the node contributing S_3 (node 8) to node 7. An initial token is placed on each control edge to account for the packet differential between the nodes.

The TRE for the AMG of Figure 4.1 without any node time variation is shown in Figure 4.3 which is obtained using the ATAMM Design Tool (Version 2.1). This pictorial view of the TRE corresponds to the analytical TRE shown in Figure 4.2. The Graph Description and Simulation Control (GDSC) file used for the AMG of Figure 4.1 with node time variation is shown in Figure 4.4, and the corresponding TRE obtained using System Simulator/Analyzer Version 2.6 (1987-88) is shown in Figure 4.5. From Figure 4.5, it is evident that a total seven (7) resources are required if the node time of node 1 is varied. The AMG may be modified, as shown in Figure 4.6, by inserting two control edges. The GDSC file for the AMG of Figure 4.6 under node time variation is shown in Figure 4.7. The TRE for the AMG of Figure 4.6 is shown in Figure 4.8, which is obtained using the System Simulator and GDSC file of Figure 4.7. From Figure 4.8, it is seen that even under node time variation a total five (5)

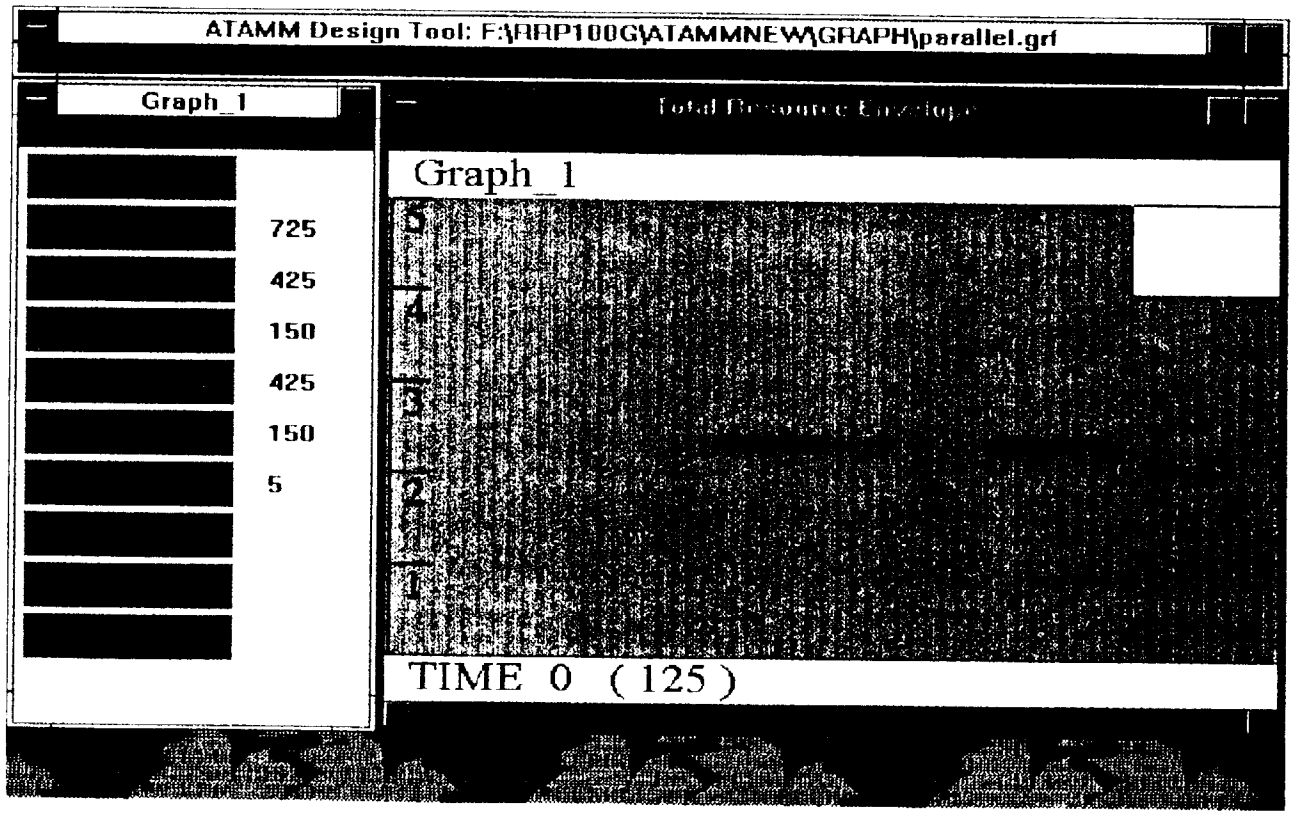


Figure 4.3. The TRE for the AMG of Figure 4.1 without any node time variation.

```
# Graph Description File for Case Study I #
# Graph with three parallel paths and time varying node 1 #
# To find resource requirement under node time variation #
# TBO(GLB) = 150, TBIO(GLB) = 425 #

        Nodes  8
        Sources 1
        Sinks   1
        Places 11
        Priority 4 8 6 3 5 7 2 1
        Resources 8
        Input 1
        Output 4
        Times
             Read     20
             Process  70
             Write    10

        Node 1
        Inputs 1
        Outputs 2 6 9
        times
              Read    20
              Process 70 R 30
              Write   10

        Node 2
        Inputs 2
        Outputs 3
        Time
             Read     15
             Process  50
             Write    10

        Node 3
        Inputs 3
        Outputs 4

        Node 4
        Inputs 4 8 11
        Outputs 5
```

Figure 4.4. The GDSC file for the AMG of Figure 4.1 with time varying node 1.

```
Time
     Read    20
     Process 120
     Write   10

Node 5
Inputs 6
Outputs 7

Node 6
Inputs 7
Outputs 8
Time
     Read    10
     Process 30
     Write   10

Node 7
Inputs 9
Outputs 10
Time
     Read    10
     Process 30
     Write   10

Node 8
Inputs 10
Outputs 11

Source 1
Outputs 1
Time
     Process 10
     Write 130

Sink 2
Inputs 5
Time
     Read 20

End
```

Figure 4.4. (Continued)

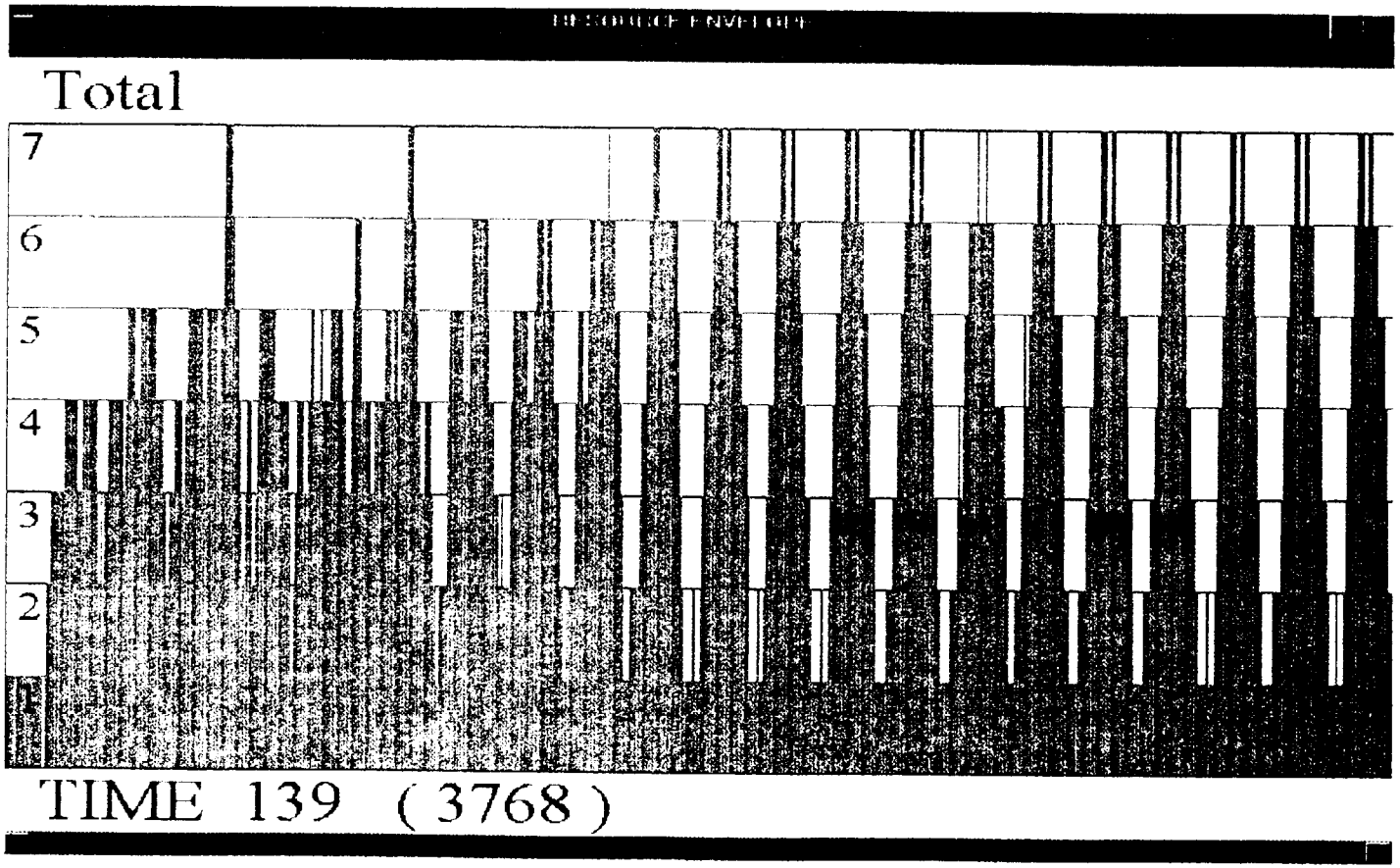


Figure 4.5. The TRE for the AMG of Figure 4.1 under node time variation.

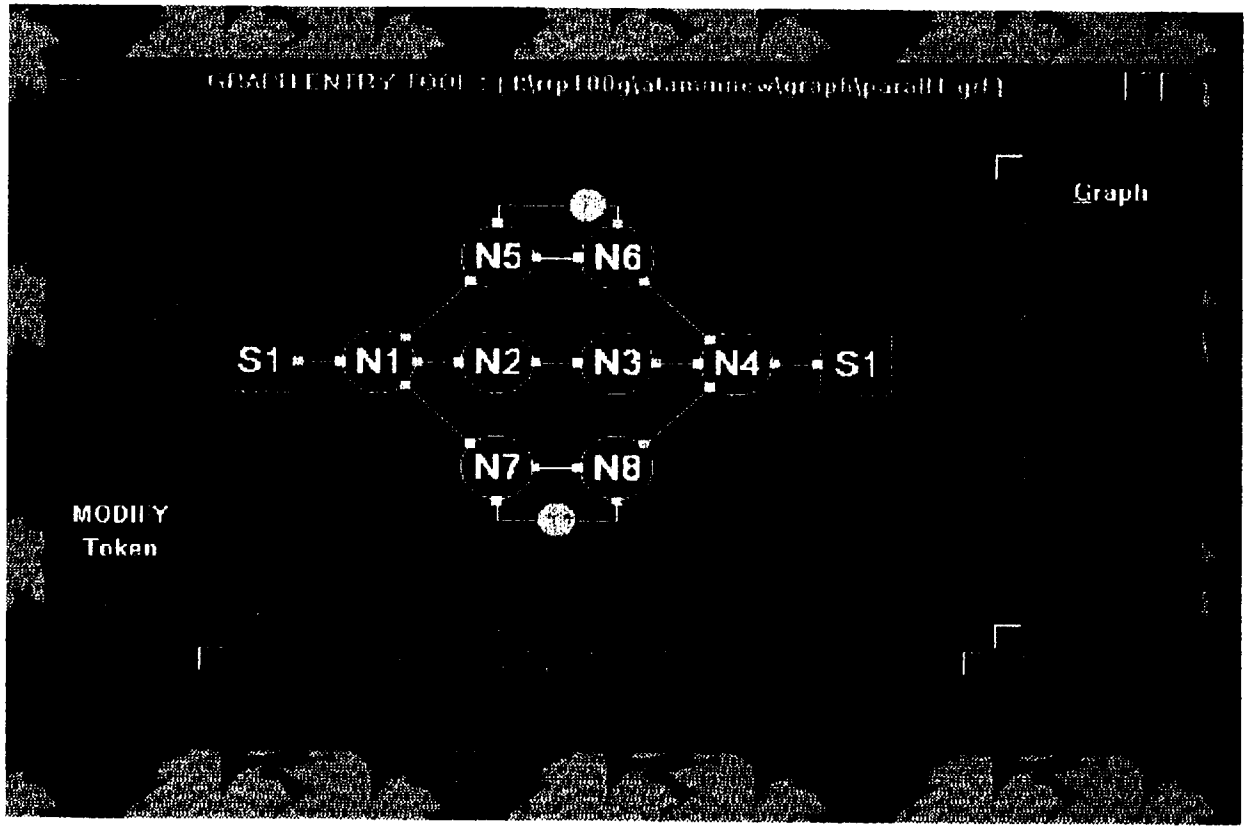


Figure 4.6. The AMG of Figure 4.1 with control edges inserted to prevent resource limited mode.

```
# GDSC File for AMG to eliminate resource limited mode #
# Graph with three parallel paths with two control edges inserted #
# TBO(GLB) = 150, TBIO(GLB) = 425 #

        Nodes  8
        Sources 1
        Sinks   1
        Places 13                       # Increase by 2 due to two control edges
        Priority 4 8 6 3 5 7 2 1
        Resources 8
        Input 1
        Output 4
        Times                           # Global time assignment #
             Read     20
             Process  70
             Write    10

        Node 1
        Inputs 1
        Outputs 2 6 9
        times                           # Local time assignment #
              Read    20
              Process 70 R 30           # Time varying node #
              Write   10

        Node 2
        Inputs 2
        Outputs 3
        Time
             Read     15
             Process  50
             Write    10

        Node 3
        Inputs 3
        Outputs 4

        Node 4
        Inputs 4 8 11
        Outputs 5
```

Figure 4.7. The GDSC file for the AMG of Figure 4.6 to prevent resource limited mode.

```
Time
     Read    20
     Process 120
     Write   10

Node 5
Inputs 6 12
Outputs 7

Node 6
Inputs 7
Outputs 8 12
Time
     Read    10
     Process 30
     Write   10

Node 7
Inputs 9 13
Outputs 10
Time
     Read    10
     Process 30
     Write   10

Node 8
Inputs 10
Outputs 11 13

Source 1
Outputs 1
Time
     Process 10
     Write 130                  # TBO - Read time of node 1 #

Sink 2
Inputs 5
Time
     Read 20                    # Read time of node 4 #

End
```

Figure 4.7. (Continued)

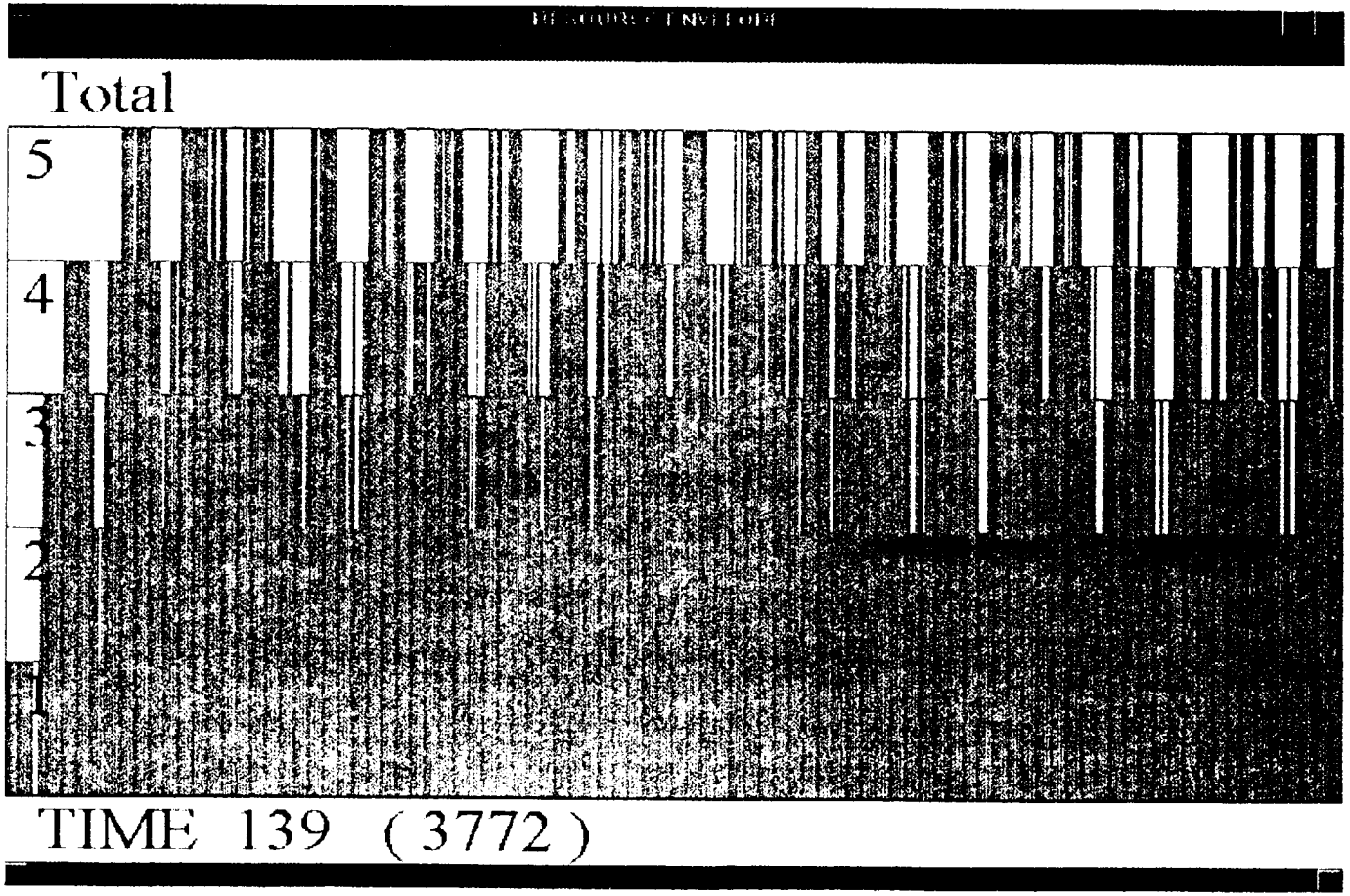


Figure 4.8. The TRE for the AMG of Figure 4.6 under node time variation.

resources are required, which is equal to the maximum number of resources in the system. Therefore, resource limited mode under node time variation is eliminated.

4.3 Case Study - II: A Graph with a Circuit

An AMG with a closed loop (circuit) in it is investigated with time varying nodes. The algorithm graphs with circuits are generally found in digital signal processing and control applications. In a graph containing a circuit, data from the predecessor cycle is needed for computation of a current data packet. Figure 4.9 represents an example of a graph with a circuit. This AMG is investigated under node time variation.

The critical path for a graph with a circuit may be found by using the modified AMG method. Subsequently, for the AMG of Figure 4.9, the critical path is found to be 1-8-9-4-5, and the critical path length (TBIO) is equal to 600. Also, the values of TBO and TCE are 500 and 1000, respectively. The waiting tokens are marked in the AMG as shown in Figure 4.9.

The k-value for each node, significant k-values, resource boundaries, the path length from input to the corresponding node, and its mod TBO value are shown in Table 4.3. In Table 4.4, these boundaries are ordered with respect to mod TBO values (t_i), and the possible range of variation of A_i in one TBO interval is calculated. Assuming P resources at the left-most point in the TGP

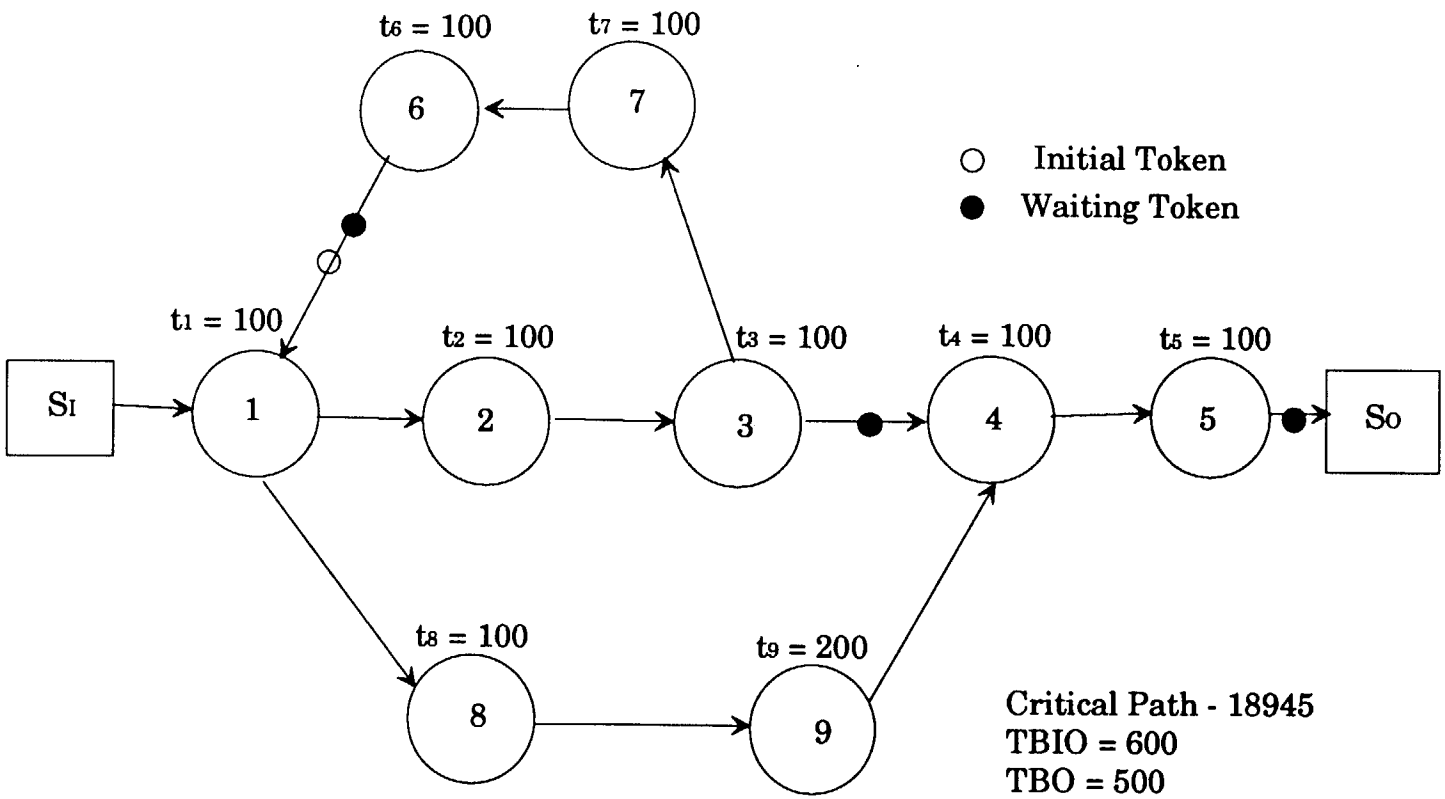


Figure 4.9. Example AMG for case study II with a circuit.

Node	m	T	k=m-T-1	Boundary	Path Length	Mod TBO
1	2	0	1	A_1	100	100
2	1	0	0	-		
3	2	1	0	-		
4	1	0	0	-		
5	1	1	-1	S_1	600	100
6	1	1	-1	S_2	500	0
7	1	0	0	-		
8	1	0	0	-		
9	1	0	0	-		

Table 4.3. The resource envelope table showing k-values and boundaries for the AMG of Figure 4.7.

	t_0=0	t_1=100
Boundaries	S_2	S_1 A_1
Latest start of A_i = Mod TBO [t{A_i} - t{$NODE_{Ai}$}]	-	100 - t_1 = 100 - 100 = 0

Table 4.4. Finding the possible range of variation of A_i boundaries in one TBO time frame for the AMG of Figure 4.7.

frame, the number of resources in each time slot are found since the amount of increase and decrease at each A_i and S_i boundary is known.

For AMG of Figure 4.9, the analytical resource envelope without any node time variation is shown in Figure 4.10. It is seen that $R = R_{max}$ resources are required for the entire TGP time frame. From Figure 4.10 and using Equation 3.1, $R = R_{max}$ can be determined as follows:

$$TCE = 1000 = R(100-0) + R(500-100)$$

$$= 500\ R$$

$$\text{or, } R = R_{max} = 2$$

From Figure 4.10, it is observed that if boundary A_1 is shifted to the left, the resource requirement is increased to R+1 due to an increase of 1 caused by the A_1 boundary. Consequently, the system is driven into resource limited mode if the node time of node 1 falls below 100.

The number of control edges to prevent resource limited mode are found as:

For A_1 (k = 1); S_1 (j = 0) = 1 - 0 - 0 = 1.

Thus, one control edge is required to prevent a possible resource limited mode due to node time variation of node 1. A control edge may be inserted from the node corresponding to S_1 (node 5) to any one successor of node 1 (either node 2 or node 8). In the simulation, the control edge is inserted from node 5 to node 8. An initial token is placed on the control edge.

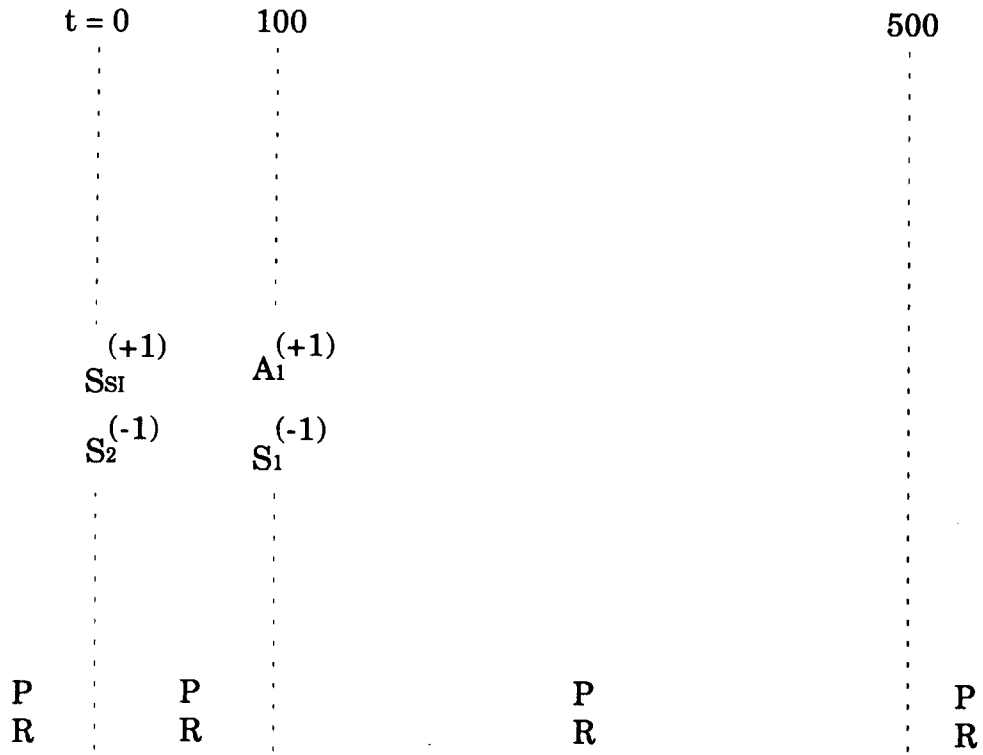


Figure 4.10. Ordered array of boundaries and resources in different regions for the AMG of Figure 4.9.

The TRE for the AMG of Figure 4.9 without any node time variation is shown in Figure 4.11, which is obtained using the ATAMM Design Tool. The pictorial view of the TRE shown in Figure 4.11 corresponds to the analytical TRE shown in Figure 4.10. The GDSC file used for the AMG of Figure 4.9 with time varying node 1 is shown in Figure 4.12, and the corresponding TRE is shown in Figure 4.13 which is obtained using the System Simulator. From Figure 4.13, it is evident that a total three (3) resources are required if the node time of node 1 is varied. The AMG may be modified, as shown in Figure 4.14, by inserting the control edge directed from node 5 to node 8. The GDSC file for the AMG of Figure 4.14 under node time variation of node 1 is shown in Figure 4.15, and the corresponding TRE obtained using the System Simulator (Version 2.6) is shown in Figure 4.16. From Figure 4.16, it is seen that a total two (2) resources are required even under node time variation which is equal to the maximum number of resources in the system. Therefore, resource limited mode under node time variation was eliminated for this case.

4.4 Case Study - III: A Conditional Node Graph

A conditional node AMG, and node time variation is investigated in this section. The conditional node algorithm graphs are encountered in signal processing and control applications. The AMG used for Case Study - III is shown in Figure 4.17. This AMG has three parallel paths. However, top two parallel paths are conditional paths. Any one of the top two parallel paths is

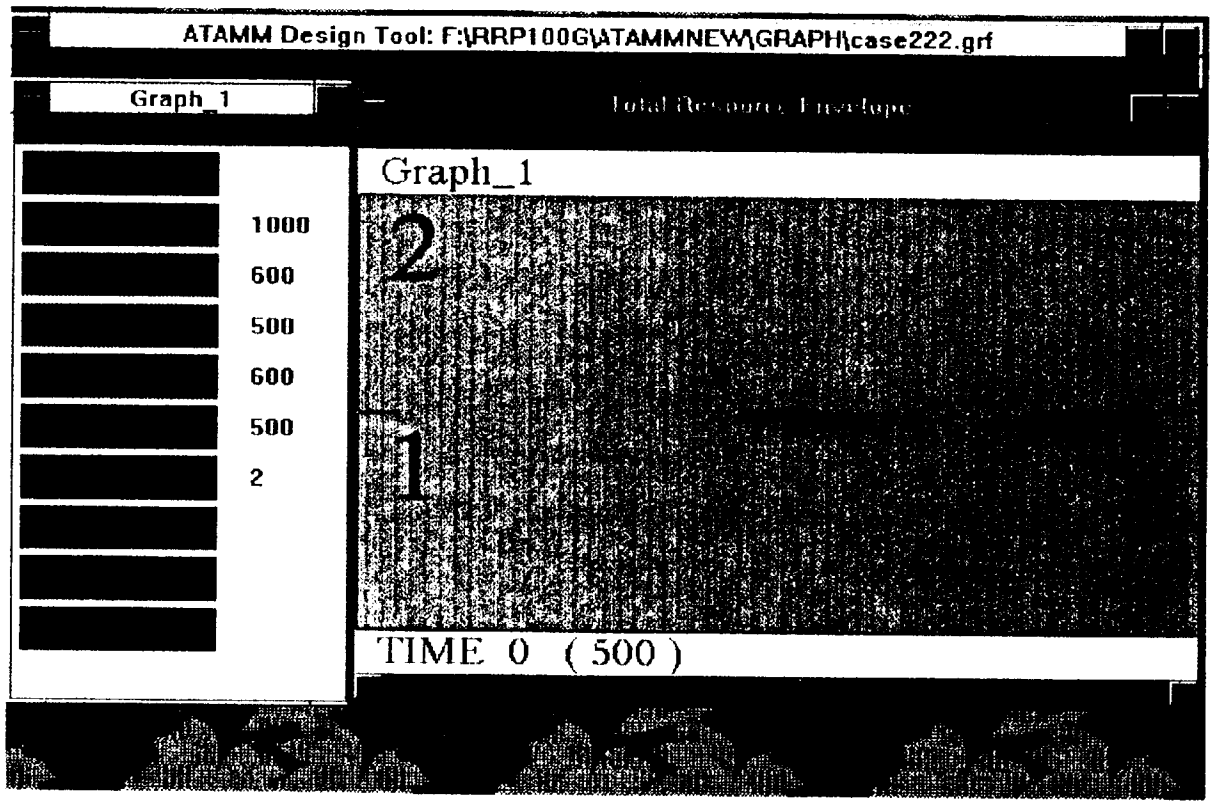


Figure 4.11. The TRE for the AMG of Figure 4.9 without any node time variation.

```
# The AMG for Case Study - II #
# Graph with circuit. TBO(GLB) = 500. TBIO(GLB) = 600. #
# To Find TRE under Node Time Variation #

    Nodes 9
    Sources 1
    Sinks  1
    Places 12
    Priority 5 4 6 7 9 3 8 2 1
    Resources 2
    Input 1
    Output 5
    Times                       # Global time assignment #
        Read    20
        Process 70
        Write   10

    Node 1
    Inputs 1 7
    Outputs 2 10
    Times                       # Local time assignment #
        Read    20
        Process 70 R 50         # Time variation of node 1 #
        Write   10

    Node 2
    Inputs 2
    Outputs 3

    Node 3
    Inputs 3
    Outputs 4 9

    Node 4
    Inputs 4 12
    Outputs 5

    Node 5
    Inputs 5
    Outputs 6
```

Figure 4.12. The GDSC file for the AMG of Figure 4.9 with time varying node 1.

```
Node 6
Inputs 8
Outputs 7

Node 7
Inputs 9
Outputs 8

Node 8
Inputs 10
Outputs 11

Node 9
Inputs 11
Outputs 12
Times                          # Local time assignment #
     Read    30
     Process 150
     Write   20

Source 1
Outputs 1
Time
     Process 10
     Write 480

Sink 2
Inputs 6
Time
     Read 20

 End
```

Figure 4.12. (Continued)

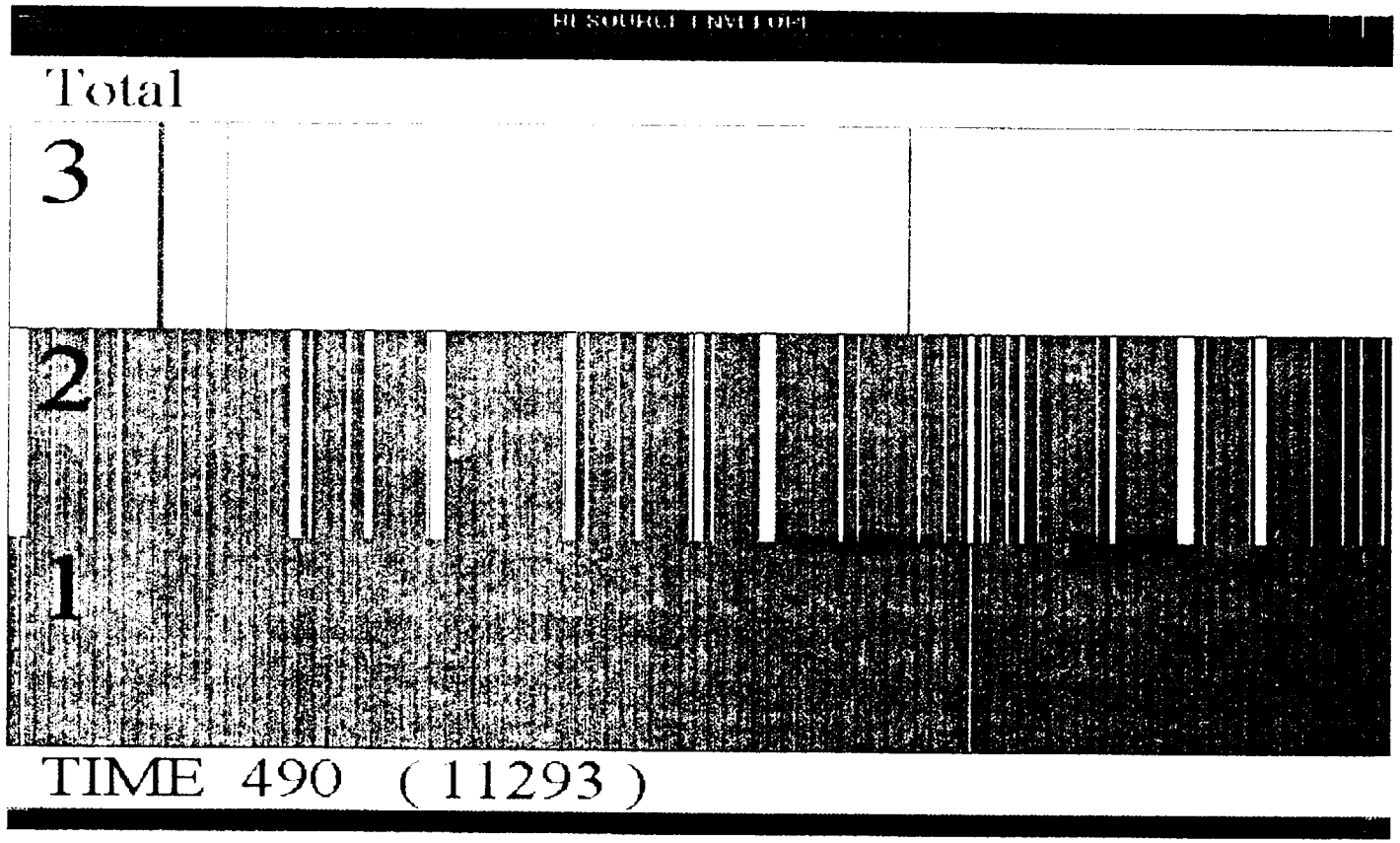


Figure 4.13. The TRE for the AMG of Figure 4.9 under node time variation.

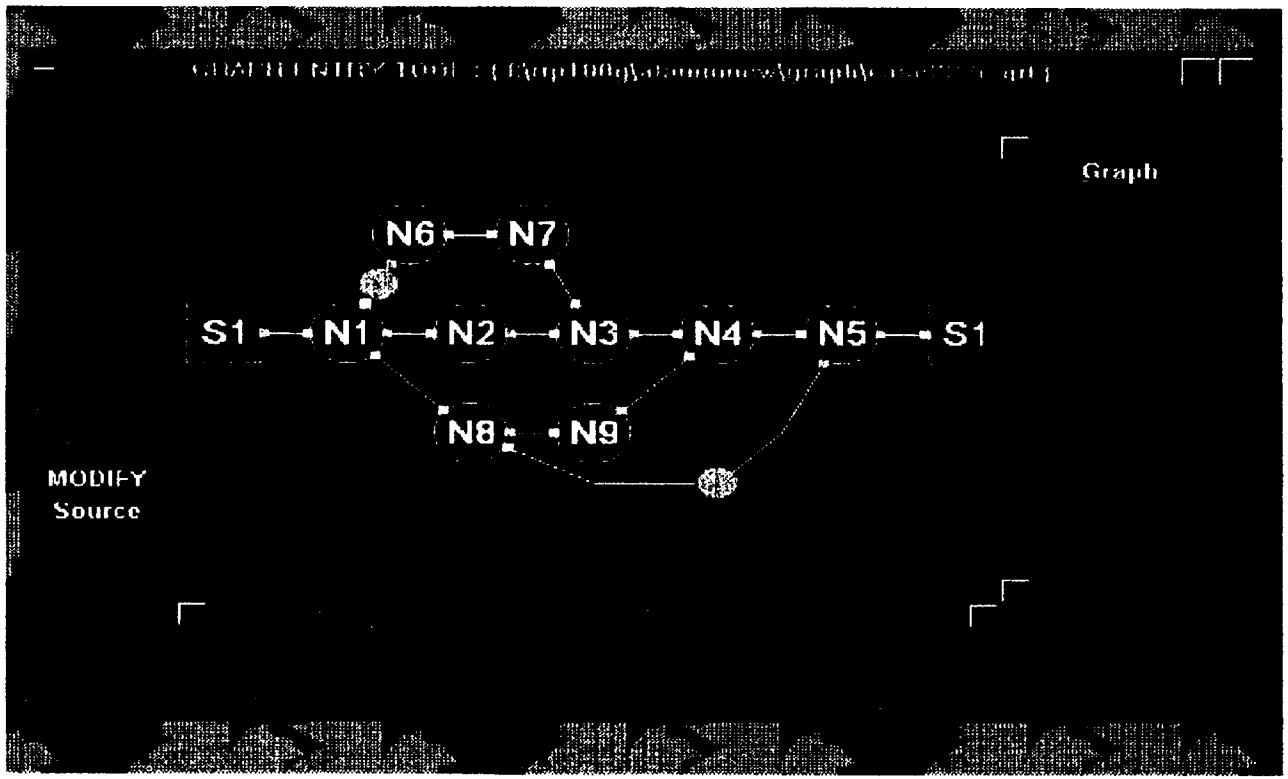


Figure 4.14. The AMG of Figure 4.9 with the control edge inserted from node 5 to node 8 to prevent resource limited mode.

```
# The AMG for Case Study - II #
# Graph with circuit. TBO(GLB) = 500. TBIO(GLB) = 600. #
# Control edge (directed from node 5 to node 8) is added #
# to prevent resource limited mode #

    Nodes  9
    Sources 1
    Sinks   1
    Places 13                  # One control edge is added #
    Priority 5 4 6 7 9 3 8 2 1
    Resources 2
    Input 1
    Output 5
    Times                      # Global time assignment #
        Read     20
        Process  70
        Write    10

    Node 1
    Inputs 1 7
    Outputs 2 10
    Times                      # Local time assignment #
        Read     20
        Process  70 R 50       # Time varying node 1 #
        Write    10

    Node 2
    Inputs 2
    Outputs 3

    Node 3
    Inputs 3
    Outputs 4 9

    Node 4
    Inputs 4 12
    Outputs 5
```

Figure 4.15. The GDSC file for the AMG of Figure 4.14 of case study II to prevent resource limited mode.

```
Node 5
Inputs 5
Outputs 6 13

Node 6
Inputs 8
Outputs 7

Node 7
Inputs 9
Outputs 8

Node 8
Inputs 10 13
Outputs 11

Node 9
Inputs 11
Outputs 12
Time                          # Local time assignment #
     Read    30
     Process 150
     Write   20

Source 1
Outputs 1
Time
     Process 10
     Write 480

Sink 2
Inputs 6
Time
     Read 20

End
```

Figure 4.15. (Continued)

Figure 4.16. The TRE for the AMG of Figure 4.14 under node time variation.

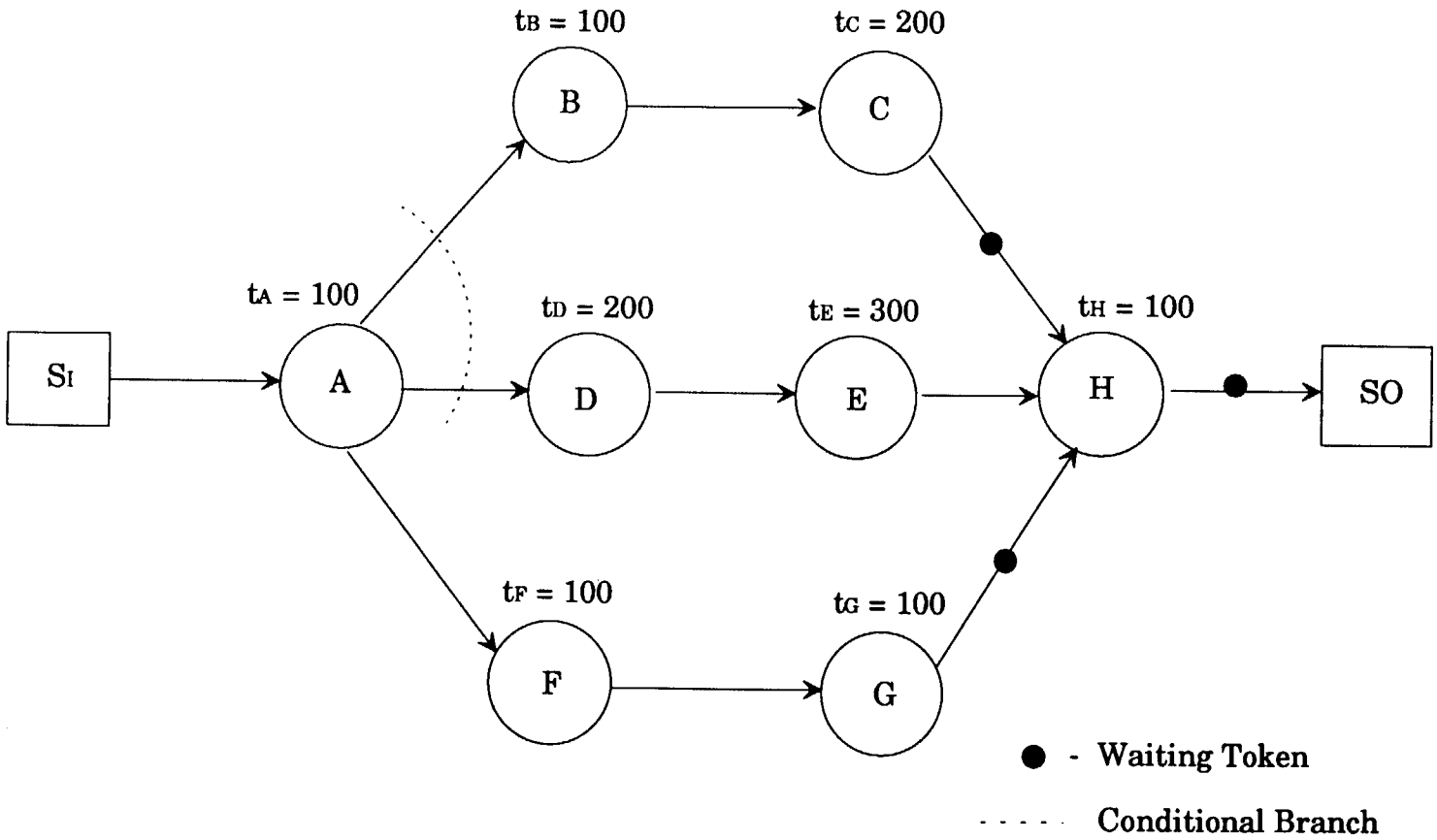


Figure 4.17. A conditional node example AMG for case study III.

executed at a given time, but the bottom path is always executed since it is not a part of the conditional branch.

An equivalent (reduced) AMG for the conditional node graph of Figure 4.17 is shown in Figure 4.18. Nodes B and D, and nodes C and E of Figure 4.17 are combined as node 2 and node 3 of Figure 4.18, respectively. The latest times to deposit are shown along edges and, subsequently, node times of partitioned nodes 2 and 3 are calculated. Nodes 1, 5, 6, and 4 in Figure 4.18 corresponds to nodes A, F, G, and H respectively, in Figure 4.17.

For the reduced AMG shown in Figure 4.18, the critical path is 1-2-3-4, and the critical path length is 700. Also, TBO = 300 and TCE = 900. The waiting tokens are marked in the AMG, as shown in Figure 4.18.

The k-value for each node, significant k-values along with resource boundaries, the path length from input to corresponding node, and its mod TBO value are presented in Table 4.5. In Table 4.6, these boundaries are ordered with respect to mod TBO values (t_i), and the allowable range of variation of A_i in one TBO frame is calculated. Figure 4.19 shows the analytical resource envelope. From Figure 4.19 and using Equation 3.1, $R = R_{max}$ can be determined as follows:

$$TCE = 900 = R(100-0) + R(300-100)$$

$$= 300\ R$$

$$\text{or, } R = R_{max} = 3$$

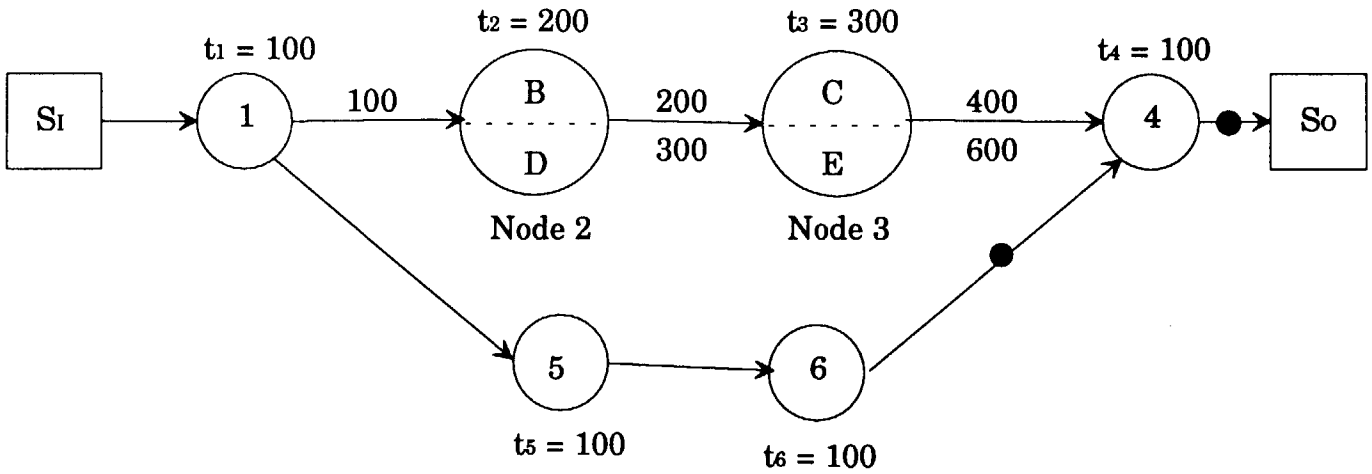


Figure 4.18. Reduced graph for the conditional node AMG shown in Figure 4.17.

Node	m	T	k=m-T-1	Boundary	Path Length	Mod TBO
1	2	0	1	A_1	100	100
2	1	0	0	-		
3	1	0	0	-		
4	1	1	-1	S_1	700	100
5	1	0	0	-		
6	1	1	-1	S_2	300	0

Table 4.5. The resource envelope table showing k-values and boundaries for the AMG of Figure 4.18.

	$t_0=0$	$t_1=100$
Boundaries	S_2	S_1 A_1
Latest start of A_i = Mod TBO [t{A_i} - t{$NODE_{Ai}$}]	-	100 - t_1 = 100 - 100 = 0

Table 4.6. Finding the possible range of variation of A_1 boundaries in one TBO time frame for the AMG of Figure 4.18.

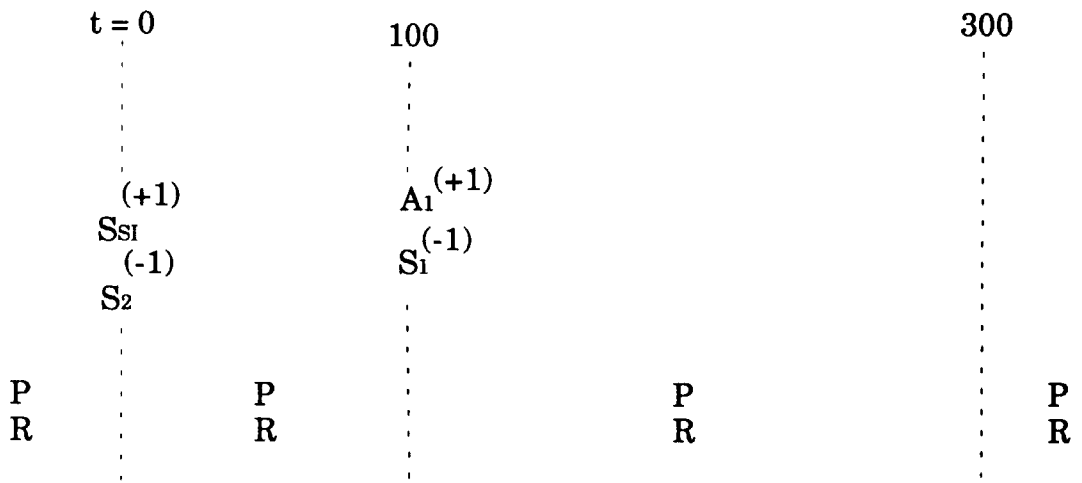


Figure 4.19. The analytical view of the resource envelope for the AMG of Figure 4.18.

From Figure 4.19, if A_1 is shifted to the left, it causes the resource requirement to be equal to R+1. Consequently, the system is driven into resource limited mode if the node time of node 1 falls below 100. The number of control edges to prevent resource limited mode are found as:

For A_1 (k = 1); S_1 (j = 0) = 1 - 0 - 0 = 1, and S_2 (j = 0) = 1 - 0 - 1 = 0.

Thus, one control edge (for S_1) is required to prevent a possible resource limited mode. The control edge may be inserted from the node contributing to S_1 (node 4) to any one successor of node 1 (node 2 or 5). An initial token is placed on the control edge to account for the packet differential between the nodes.

The TRE for the AMG of Figure 4.18 without any node time variation is shown in Figure 4.20 which is obtained using the Design Tool. The TRE exactly corresponds to the analytical resource envelope shown in Figure 4.19. The GDSC file used for simulation of the AMG of Figure 4.18 with node time variation is shown in Figure 4.21, and the corresponding TRE obtained is shown in Figure 4.22. From Figure 4.22, it is seen that a total four (4) resources are required if the node time of node 1 is varied, i.e., resource limited mode is present. To prevent resource limited mode, the AMG may be changed, as shown in Figure 4.23, by inserting the control edge (found above) from node 4 to node 5. The GDSC file for simulation of the AMG of Figure 4.23 under node time variation is shown in Figure 4.24, and the corresponding TRE is shown in Figure 4.25. From Figure 4.25, it is seen that even under node time

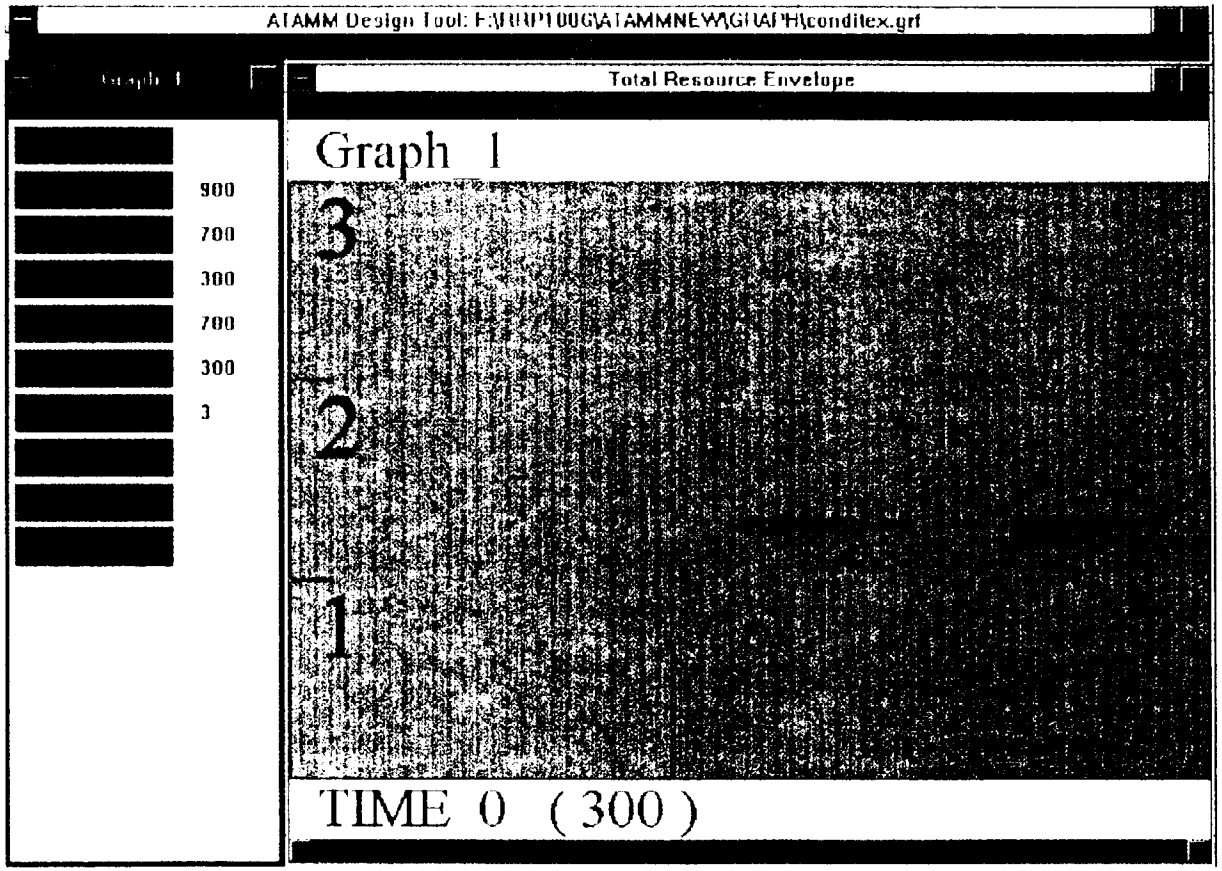


Figure 4.20. The TRE for the AMG of Figure 4.18 without any node time variation.

```
# Graph Description File for the AMG of Case Study - III #
# A Conditional Node Graph with Three Parallel Paths #
# TBO(GLB) = 300. TBIO(GLB) = 700 #
# To Find the TRE under Node Time Variation #

    Nodes 6
    Sources 1
    Sinks  1
    Places 8
    Priority 4 3 6 5 2 1
    Resources 3
    Input 1
    Output 4

    Node 1
    Inputs 1
    Outputs 2 3
    Time
        Read    10
        Process 80 R 10          # Time variation of node 1 #
        Write   10

    Node 2
    Inputs 2
    Outputs 4
    Time
        Read    10
        Process 180 R 50         # Time varying combined node 2 #
        Write   10

    Node 3
    Inputs 4
    Outputs 5
    Time
        Read    10
        Process 280 R 30         # Time varying combined node 3 #
        Write   10
```

Figure 4.21. The GDSC file for the AMG of Figure 4.18 with time varying node 1, and combined nodes 2 and 3.

```
Node 4
Inputs 5 7
Outputs 8
Time
     Read    10
     Process 80
     Write   10

Node 5
Inputs 3
Outputs 6
Time
     Read    10
     Process 80
     Write   10

Node 6
Inputs 6
Outputs 7
Time
     Read    10
     Process 80
     Write   10

Source 1
Outputs 1
Time
     Process 290
     Write 10

Sink 2
Inputs 8
Time
     Read 10

End
```

Figure 4.21. (Continued)

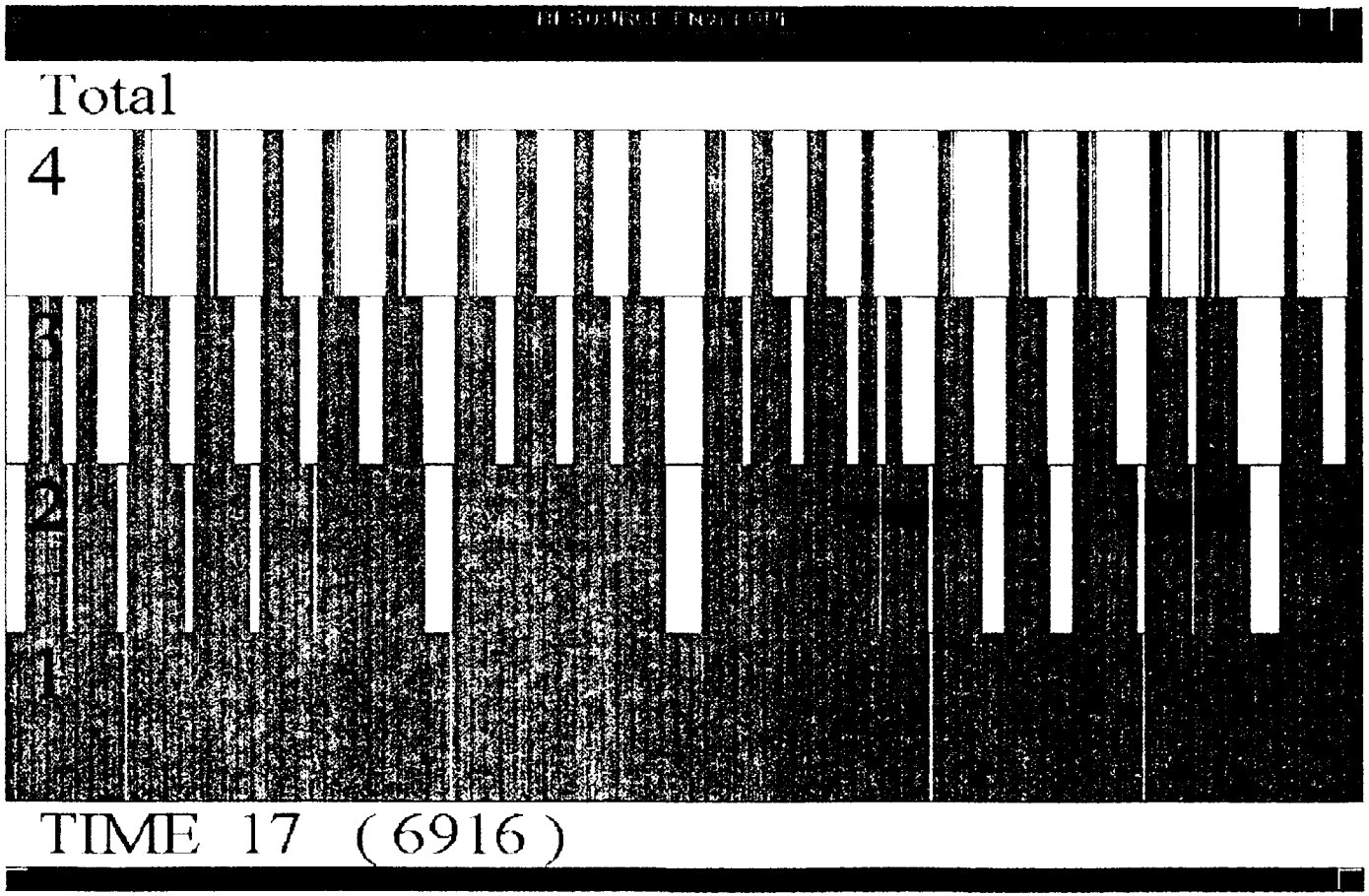


Figure 4.22. The TRE for the AMG of Figure 4.18 under node time variation.

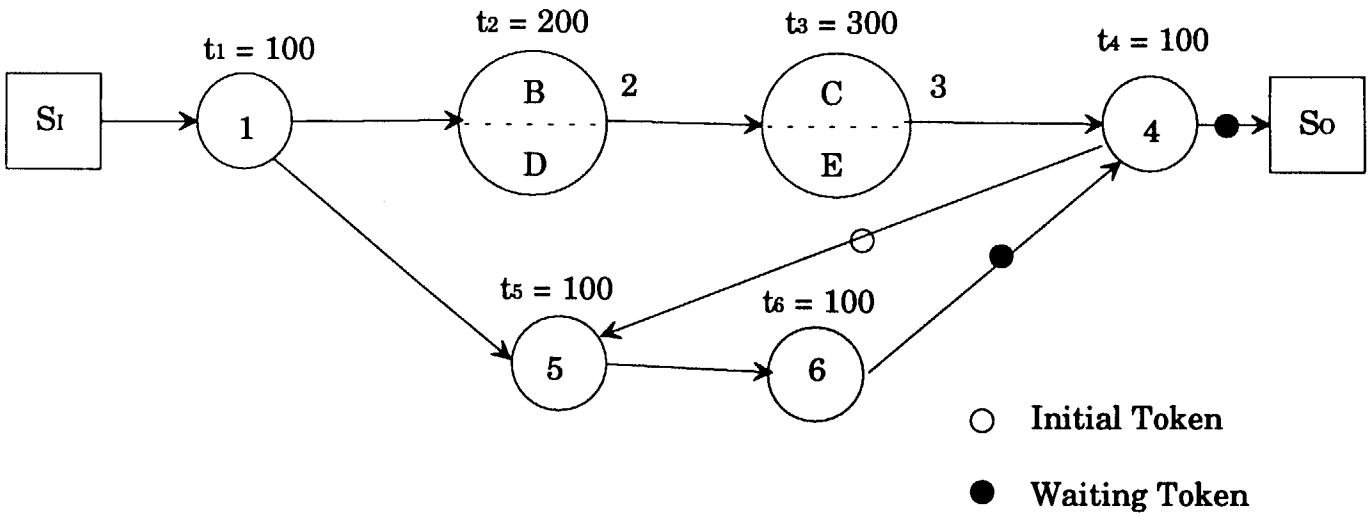


Figure 4.23. The AMG of Figure 4.18 with the control edge inserted from node 4 to node 5 to prevent resource limited mode.

```
# The GDSC File for the AMG of Case Study - III #
# A Conditional Node Graph with Three Parallel Paths #
# TBO(GLB) = 300. TBIO(GLB) = 700 #
# Control Edge directed from Node 4 to Node 5 is added #
# to Prevent Resource Limited Mode #
# To find the TRE with the Control Edge added #

    Nodes  6
    Sources 1
    Sinks   1
    Places  9                    # One Control Edge Added #
    Priority 4 3 6 5 2 1
    Resources 3
    Input 1
    Output 4

    Node 1
    Inputs 1
    Outputs 2 3
    Time
        Read    10
        Process 80 R 10          # Node Time Variation of Node 1 #
        Write   10

    Node 2
    Inputs 2
    Outputs 4
    Time
        Read    10
        Process 180 R 50         # Time Varying Combined Node 2 #
        Write   10

    Node 3
    Inputs 4
    Outputs 5
    Time
        Read    10
        Process 280 R 30         # Time Varying Combined Node 3 #
        Write   10
```

Figure 4.24. The GDSC file for the AMG of Figure 4.23 to prevent resource limited mode.

```
Node 4
Inputs 5 7
Outputs 8 9
Time
     Read    10
     Process 80
     Write   10

Node 5
Inputs 3 9
Outputs 6
Time
     Read    10
     Process 80
     Write   10

Node 6
Inputs 6
Outputs 7
Time
     Read    10
     Process 80
     Write   10

Source 1
Outputs 1
Time
     Process 290
     Write 10

Sink 2
Inputs 8
Time
     Read 10

End
```

Figure 4.24. (Continued)

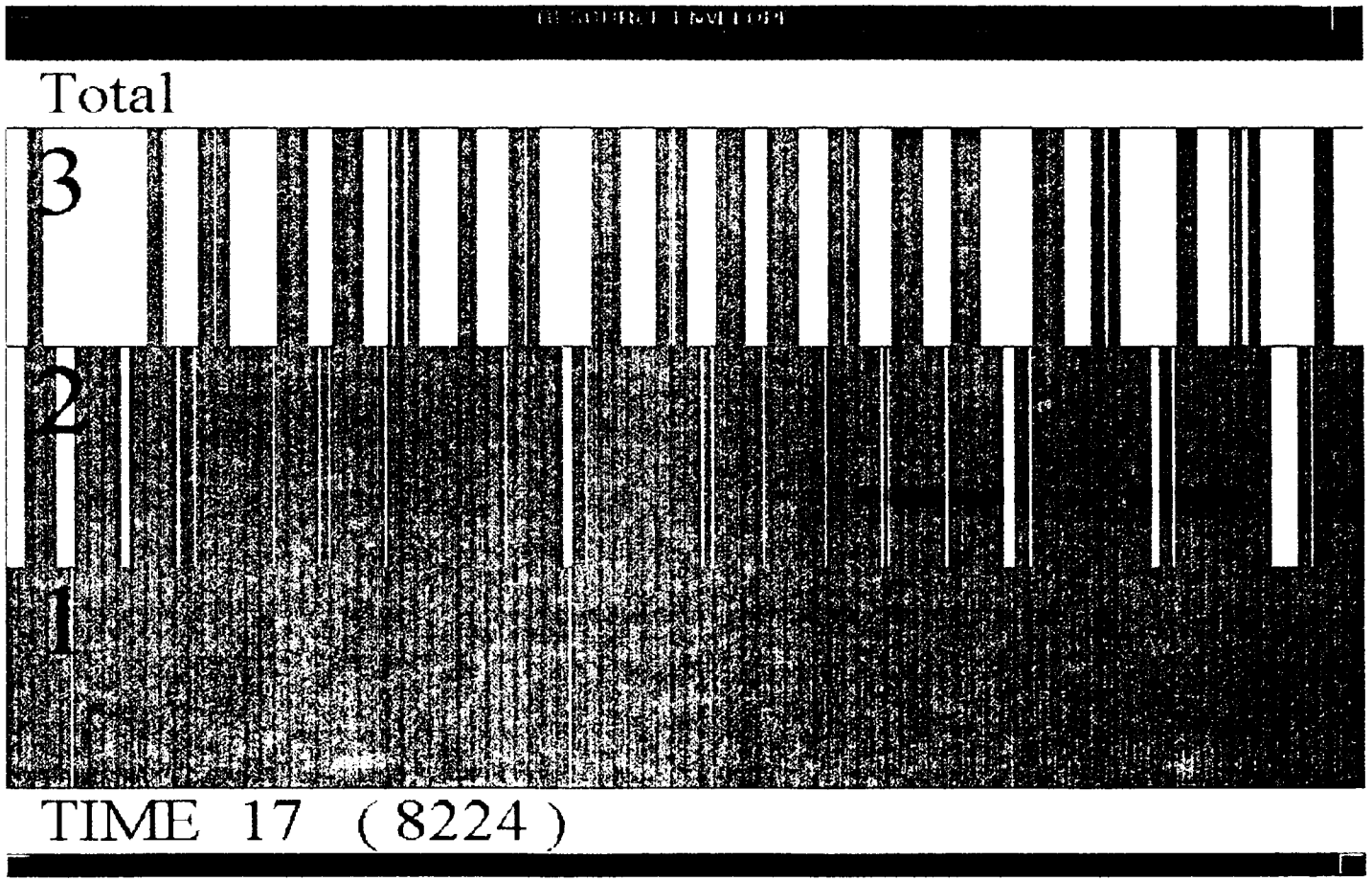


Figure 4.25. The TRE for the AMG of Figure 4.23 under node time variation.

variation a total three (3) resources are required, which is equal to the maximum number of resources in the system. Therefore, resource limited mode under node time variation is eliminated.

CHAPTER FIVE

CONCLUSION

5.1 Summary

The development and application of the analytical model for resource utilization, in an ATAMM based real-time data flow architecture, has been the primary goal of this research. Several useful results are obtained during the process of model development. First, the analytical resource envelope may be found directly from the given Algorithm Marked Graph (AMG) and, consequently, the maximum number of resources are evaluated from a worst case analysis. Second, the Total Graph Play (TGP) diagram may be constructed using information from the analytical resource envelope and a given AMG. Third, the behavior of an AMG, with time varying nodes, is investigated in view of the change in resource requirements using the analytical resource envelope of a given AMG. From the study of node time variation in the AMG, a potential problem of resource limited mode in time varying node graphs is encountered. A condition for the presence of resource limited mode is found which, consequently, leads us to a method for preventing resource limited mode.

An interesting extension to the time varying node case is the modeling of an AMG with conditional branches. It has been shown that the conditional

node graph may be made equivalent to the unconditional AMG by merging nodes. Therefore, a conditional node graph may be analyzed using the variable node time model to determine resource requirements. An algorithm [4] which determines the critical path and the critical path length (TBIO) has been refined so as to take into account an AMG with initial tokens on forward edges.

5.2 Evaluation

The analytical resource utilization model is a model which describes a method for obtaining the analytical resource envelope directly from a given AMG. The analytical resource envelope so obtained corresponds to the pictorial view of the total resource envelope (obtained using the ATAMM software support tools). The analytical resource envelope is obtained using the notion of a waiting token, where it is noted that a deposit of the waiting token may be viewed as the release of a resource unit. From the analytical resource envelope, the value of R_{max}, maximum number of resources for optimum time performance, may be derived using a worst case analysis. The analytical resource envelope shows each and every resource change boundary in the TGP diagram, and therefore is useful to evaluate changes in resource requirements under various conditions. In the case studies, which are performed for three algorithm graphs, the analytical resource envelope is obtained for each graph, using the analytical resource utilization model. For each graph, the analytical

resource envelope directly corresponded to its Total Resource Envelope (TRE), which is obtained using software simulation of an ATAMM based system.

For a given AMG, a method is described to construct the TGP diagram from the analytical resource envelope instead of from the Single Graph Play (SGP) diagram [4]. The new approach is useful since it provides the correct steady-state view of the TGP diagram for a graph with initial tokens on forward edges, whereas the SGP approach does not. The TGP diagram so obtained gives the number of required resources in each region over one TBO interval, which may also be obtained from the analytical resource envelope.

The basis for development of the variable node time model is provided by the analytical resource utilization model. The variable node time model is developed as an ATAMM enhancement, which describes the change in resource requirements for the execution of an algorithm under node time variation. It is shown that an increase in the number of required resources beyond R_{max}, or resource limited mode, may occur at any instant in a time varying node graph. A sufficient condition for the presence of resource limited mode is presented. Resource limited mode is undesirable because it causes the system performance to become unpredictable. Consequently, a method of modifying the graph with control edges so as to prevent resource limited mode is described. The approach of using the analytical resource envelope is significant because the change in maximum resource requirement can be easily viewed, and it also leads us to a method for detecting resource limited mode.

The variable node time model is useful in analyzing resource requirements of a conditional node graph. In the case studies, the potential of resource limited mode and control edges for its prevention are found for example algorithm graphs, using the variable node time model. Also, each AMG, with variable node times, is run in a software simulation and, subsequently, resource limited mode is observed from the TRE of each AMG. After inserting the control edges to prevent resource limited mode, each AMG is simulated again. From the TRE of each AMG, it is observed that resource limited mode has been prevented. Thus, the experimental results illustrate the applicability of the analytical results.

An overview of the conditional node model is presented, which describes a method of reducing the conditional node graphs to equivalent graphs with time varying nodes, using a notion of partitioned or combined nodes. The idea is to make use of the variable node time model to predict resource requirements for the execution of a conditional node graph. In case study III, a conditional node graph is reduced to an equivalent graph. From the equivalent graph, the analytical resource envelope and the value of R_{max} are found using the resource utilization model, which corresponded to the experimental results. Using the variable node time model, the equivalent graph is investigated for resource limited mode and its prevention. Also, the equivalent graph is simulated under node time variation, without and with control edges, for illustration of the analytical results.

The case studies formed a basic demonstration tool for the illustration of the analytical results. From these results, and without more direct proof, the resource utilization model is robust, consistent, and is a natural extension to the existing ATAMM modeling concept.

5.3 Topics for Future Research

Research leads us to several topics of continuing and future research. First, the ATAMM design tool could be modified to incorporate the idea of the analytical resource envelope, the TGP diagram, and the variable node time model. Second, the conditional node model could be investigated for its implementation in the ATAMM based data flow architectures.

Current version of the design tool utilizes the SGP diagram of a given graph to evaluate performance measures and resource requirements. The design tool might be modified to obtain the analytical resource envelope from a given AMG and, consequently, to evaluate resource requirements by utilizing the analytical resource envelope. The design tool might also be extended to predict resource requirements of graphs with time varying nodes. The detection of the presence of resource limited mode might be included with its subsequent prevention, by inserting control edge(s) in the given AMG in real time. The reduction process of a conditional node graph into a variable node time graph might also be automated.

The implementation of a conditional node graph in the ATAMM based data flow architecture might be a topic of future research. The implementation method should take care of the null partitions in the reduced graph in such a way as to prevent the firing of these null partitions to remove the overhead associated in firing.

LIST OF REFERENCES

[1] J.W. Stoughton and R.R. Mielke, "Petri-Net Model for Concurrent Processing of Complex Algorithms," Proceedings of Government Microcircuit Applications Conference, San Diego, CA, November 1986.

[2] R.R. Mielke, J.W. Stoughton, and S. Som, "Modeling and Performance Bounds for Concurrent Processing," Proceedings of the 8th International Conference on Distributed Computing Systems, San Jose, CA, June 1988.

[3] J.L. Peterson, "Petri Net Theory and the Modeling of Systems," Englewood Cliffs, NJ, Prentice Hall, 1981.

[4] S. Som, "Performance Modeling and Enhancement for the ATAMM Data Flow Architectures," Ph.D. Dissertation, Old Dominion University, Norfolk, Virginia.

[5] R.R. Mielke, J.W. Stoughton, and S. Som, "Modeling and Optimum Time Performance for Concurrent Processing," NASA Technical Paper 4167, Grant NAG1-683, August 1988.

[6] J.W. Stoughton and R.R. Mielke, "Strategies for Concurrent Processing of Complex Algorithms in Data Driven Architectures," NASA Technical Paper 181657, Grant NAG1-683, February 1988.

[7] R.L. Jones, "Diagnostics Software for Concurrent Processing Computer Systems," M.S. Thesis, Old Dominion University, Norfolk, Virginia, April 1990.

[8] S. Som, J.W. Stoughton, and R.R. Mielke, "Performance Modeling in the ATAMM Data Flow Architecture," Technical Paper Presented at the Ninth IEEE International Phoenix Conference on Computers and Communications, Scottsdale, Arizona, March 1990.

[9] R.R. Mielke, J.W. Stoughton, S. Som, R. Obando, M. Malekpour, and B. Mandala, "ATAMM Multicomputer Operating System Functional Specification," Progress Report Prepared for NASA for the period February 1990, Grant NCC1-136, August 1990.

[10] J.W. Stoughton, R.R. Mielke, S. Som, R. Obando, M. Malekpour, R.L. Jones, and B. Mandala, "ATAMM Enhancement and Multiprocessor Performance Evaluation," Year End Report for 1990 Prepared for NASA, Grant NCC1-136, June 1991.

[11] B.C. Kuo, "Digital Control Systems," Englewood Cliffs, NJ, Prentice Hall, 1990.

[12] B. Mandala, "A Software Design Tool for Predictable Performance in Real-time, Data Flow Architectures," M.S. Thesis, Old Dominion University, Norfolk, Virginia, December 1990.