PERFORMANCE MODELING AND ENHANCEMENT IN REAL-TIME DATA FLOW ARCHITECTURES

S. Som, J. W. Stoughton, and R. R. Mielke
Department of Electrical and Computer Engineering
Old Dominion University, Norfolk, Virginia 23529

ABSTRACT
This paper is concerned with performance modeling and enhancement for periodic execution of large-grain, decision-free algorithms in data flow architectures operating in real-time. The mapping of real-time algorithms onto data flow architectures is realized by a new marked graph model called ATAMM (Algorithm To Architecture Mapping Model). Applications include control, surveillance, and signal processing problems. Performance is characterized by computing speed and throughput. Bounds on performance measures are established. A technique for transforming an algorithm to improve throughput while maintaining input-output equivalence is presented. The state equations of a linear time invariant system are modified to illustrate the throughput enhancement technique.

I. INTRODUCTION
Real-time computing is required in diverse areas such as aerospace, process control, military uses, and nuclear power. In a typical real-time system, a number of sensors provide the input data to the computer which then analyzes the input data by some predefined algorithms. This information is used to send output signals to actuators or displays. Several characteristics required of such computers include repeated execution of the same algorithm, very high computing speed and sampling rates, highly predictable and reliable performance, and the ability to meet hard deadlines for outputs. Real-time data flow architectures use the data flow concept to improve performance through the use of concurrency [1, 2]. The data flow concept has attracted the attention of a great many researchers [3]. However, only a few researchers have tried to develop a theoretical model for evaluating computation in data driven architectures operating in real-time [4, 5]. These models do not appear to be adequate to address all issues associated with real-time computing.

Ongoing research efforts at Old Dominion University have lead to the development of a new marked graph model for describing data and control flow associated with the execution of algorithms in real-time data flow architectures. The model is identified by the acronym ATAMM which represents Algorithm To Architecture Mapping Model [6]. An algorithm is expressed as a directed graph where nodes represent algorithm operations and edges represent operands. Algorithms are restricted to the class of decision-free problems. The grain size of algorithm nodes is expected to be large, and the number of such nodes is not expected to be greater than twenty. This range of functional units and algorithm nodes is selected due to the large-grained aspect of the target algorithms and knowledge of target architectures. The architecture is assumed to be a homogeneous multicomputer static data flow architecture consisting of two to twenty identical computers or functional units each having a capability for processing, communication, and memory. The term static implies that only one functional unit can work on a specific algorithm node at a time [7].

Of interest is the definition of a performance model so that the performance of the algorithms can be evaluated and improved. The primary objective is to enhance performance by transforming algorithms without any further algorithm decompositions. Transformation of algorithms include modification of recursive loops and insertion of buffers in the algorithm graph. Although computing speed is not affected by these transformations, throughput can be improved. An algorithm graph representing a linear time invariant system is transformed for throughput enhancement. The analysis, based on the ATAMM model, identifies the necessary changes in the algorithm graph for best possible throughput. Also, as the ATAMM model takes into consideration communication times for each algorithm operation, it generates a realistic measure of throughput enhancement.
In Section II of the paper, a computational problem is represented by the ATAMM model. Time performance measures for concurrent processing are defined and performance bounds are developed. In Section III, performance enhancement is achieved through equivalence transformations of algorithms. The transformations utilized are modifications of recursive loops and insertion of buffers. The recursive loop of a linear time invariant system is modified to enhance its throughput.

\[ x(k) = Ax(k-1) + Bu(k) \]  \hspace{1cm} (1)

and the output equation

\[ y(k) = Cx(k) + Du(k), \]  \hspace{1cm} (2)

where \( x \) is a \( q \)-vector, \( u \) is a \( h \)-vector, and \( y \) is a \( g \)-vector. \( A, B, C, \) and \( D \) are time invariant matrices. The AMG for this decomposed algorithm is shown in Figure 1. Source transitions and sink transitions for input and output signals are represented as squares. The initial marking indicates that initial condition data are available. The AMG indicates that algorithm nodes 3 and 4 can be executed concurrently but only after node 2 is completed. Node times correspond to processing times of the respective algorithm operations.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{algorithm_graph.png}
\caption{Algorithm marked graph for linear time invariant systems.}
\end{figure}

II. PERFORMANCE MODEL

The ATAMM model consists of a set of Petri net marked graphs which incorporates general specifications of communication and processing associated with each computational event in a data flow architecture. In this section, a computational problem is represented by the ATAMM model. Performance measures are defined and lower bounds for these measures are established. Some familiarity with Petri nets and marked graphs is assumed [8]. A detailed description of the ATAMM model and its characteristics are found in [9].

ATAMM describes algorithm execution on a data-flow architecture by three marked graphs, the algorithm marked graph (AMG), the node marked graph (NMG), and the computational marked graph (CMG). An algorithm marked graph is a marked graph which represents a decomposed algorithm. Transitions (algorithm nodes) and places (algorithm edges) represent algorithm operations and operands, respectively. The node times represent the computational times required for the algorithm operations. The algorithm marked graph contains an edge \((i, j)\) directed from node \(i\) to node \(j\) if the output of node \(i\) is an input for node \(j\). Edge \((i, j)\) is marked with a token if the output from node \(i\) is available as an input to node \(j\). All edges have a pool of buffers (storage locations) and can accommodate more than one token at a time. To illustrate the representation of a computational problem by the ATAMM model, consider the problem of computing the output of a discrete, linear, time-invariant system given a sequence of inputs to the system. Let the system be described by the state equation
The node marked graph (NMG) is a representation of the execution of an AMG node by a functional unit. Three primary activities, reading of input data from memory, processing of input data to compute output data, and writing of output data to memory, are represented as transitions in the NMG shown in Figure 2. Data and control flow paths are represented as places, and the presence of signals is notated by tokens marking appropriate places. The conditions for firing the process and write transitions of the NMG are as defined for a general Petri net, while the read transition has one additional condition for firing. A functional unit must be available for assignment to the algorithm operation before the read node can fire. The initial marking for an NMG includes a single token in the Process Ready (PR) place and a number of tokens \( n \) on the Output Buffer Empty (OE) place. The Output Buffer Full (OF) place will have initial tokens only if the corresponding algorithm edge has initial tokens. The initial token marking indicates that only one functional unit can work on an AMG node at a time (static data flow architecture [7]), but the AMG node can be repeated by functional units before the output is consumed. The total initial number of tokens on the OE and OF edges is the size of the output queue in edge OF. The computational marked graph (CMG) is constructed from the AMG by replacing every algorithm node by the corresponding NMG. AMG edges are replaced by place pairs, a forward directed place representing data flow and a backward directed place representing control flow. The CMG for Figure 1 is shown in Figure 3 assuming \( n \) equals 1 for all NMGs.

The CMG contains many directed circuits. The directed circuits are formed in four different ways. These circuit types are called node, process, recursion and parallel path circuits. First, each NMG consists of a directed circuit which has \( m \) initial marking token in the Process Ready place. Such a circuit is defined as a node circuit. The time for a node circuit is the sum of the read, process, and write transitions for that NMG.

Second, a directed circuit is formed each time an NMG or source is linked to another NMG or sink by the OE and OF edges. This circuit is called a process circuit, and the circuit time is the sum of the read, process, and...
write transitions of predecessor NMG or source time and read time of the successor NMG or sink. Third, a parallel path circuit is created in the CMG by parallel paths in the AMG. Parallel paths are directed paths in the AMG which have identical beginning and ending algorithm nodes. The circuit is formed by the forward directed places through the NMGs of one directed path, and backward directed places from the successor read to the predecessor read transition from the NMGs of the other directed path. Fourth, all circuits in the AMG are translated into the corresponding CMG. Such circuits are defined as recursion circuit as they represent recursive computation.

As an example, consider Figure 3. For simplicity, read and write times are assumed to be zero. One of the node circuits is formed at the NMG for node 4. The directed circuit can be traced by places passing through the read, process, and write transitions of node 4 and then returning to the read of node 4. This circuit has only one token, the total transition times is 3 time units, and the corresponding time per token is 3. A process circuit can be found in the linking of NMGs for nodes 3 and 6. It consists of the read, process, and write transitions of node 3, and the read transition of node 6. The time per token for this circuit is 2. There are two directed parallel paths between source and node 6 in the AMG of Figure 1. One such path passes through node 5 and the other passes through nodes 1, 2, 3, and 6. The corresponding parallel path circuit for these two paths consists of the read, process, and write transitions of nodes 1, 2, and 3, read transitions of source, node 6, and node 5. The time per token 2.5 for this circuit. The only recursion circuit of the CMG passes through read, process, and write transitions of both nodes 2 and 4. As there is only one token in the recursion circuit, the corresponding time per token is 4.

Computational concurrency occurs in two ways. First, several nodes of an AMG for an individual data packet may be performed simultaneously. We refer to this type of concurrency as parallel concurrency because it is the result of inherent parallelism in the algorithm. Parallel concurrency has a direct effect on algorithm computing speed. It is limited by the number of transitions that can be performed simultaneously for the given AMG, and by the number of functional units available to perform the transitions. Second, transitions of the AMG belonging to different data sets can be performed simultaneously in the computing system. This type of concurrency is called pipeline concurrency because the algorithm is repeated for successive data sets, similar to a pipeline. This type of concurrency has a direct effect on throughput capacity. 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.

Two performance measures, TBO and TBIO, are now defined for periodic processing of complex algorithms. TBIO is an indicator of computing speed for an algorithm and thus reflect the degree of parallel concurrency. TBO is a measure of the time interval between algorithm outputs. The inverse of TBO indicates throughput, and thus reflects the degree of pipeline concurrency. The performance measure TBIO (time between input and output) is the elapsed computing time between an algorithm input and the corresponding algorithm output. The performance measure TBO (time between outputs) is the elapsed computing time between successive algorithm outputs when the algorithm marked graph is operating periodically at steady state. It is shown in [8] that the algorithm imposed lower bound for TBIO (TBIO_{LB}) is determined by the longest path (critical path) between the input source and the output sink of the AMG. It is also known [8, 5, 10] that the largest time per token among the directed circuits of the CMG determines the algorithm imposed lower bound for TBO (TBO_{LB}). The circuits which determine TBO_{LB} are called critical circuits of the CMG.

Consider the AMG of Figure 1. There are two directed paths from source to sink. The directed path through nodes 5 and 6 has length 3 time units. Another directed path through nodes 1, 2, 3, and 6 has length 6. Hence, TBIO_{LB} is 6. The corresponding CMG is shown in Figure 3 where zero read and write times are assumed. The CMG contains many directed circuits. However, the circuit formed in the CMG due to the recursion loop of the AMG contains all NMG transitions of nodes 2 and 4 and has only one token and maximizes the ratio T(C)/M(C). Hence, TBO_{LB} is 4.

III. PERFORMANCE ENHANCEMENT

The objective of this section is to explore the potential for performance enhancement. This is to be done by transforming the algorithm marked graph while preserving the algorithm input-output relation. First, equivalency of two algorithm marked graphs is defined. Second, it is shown that throughput can be enhanced by increasing number of tokens in the critical circuits of the CMG. The addition of tokens to critical circuits requires the queue size on some edges to be more than 1. Third,
the throughput enhancement technique is applied to a graph representing a linear time invariant system.

Two algorithm marked graphs are equivalent if they map any set of input variables into the same set of output variables and produce an identical output sequence. An algorithm marked graph can be transformed as long as the new AMG is input-output equivalent with the original one. The purpose of transformation is to improve throughput. The lower bound for TBO indicates that throughput enhancement can be achieved by reducing the time per token of critical circuits in the CMG. The times of process transitions cannot be reduced without further algorithm decomposition which may not always be desirable or possible. The read and write transition times are architecture and data dependent. Therefore, the only practical way of improving throughput for the AMG is to increase the number of tokens in the critical circuits. This idea is used in the ATAMM model to increase throughput.

The number of tokens on OE or OF places of critical circuits are increased to reduce the circuit time per token. In order to accommodate multiple tokens on OE and OF edges, the AMG edges must have buffers or multiple storage locations. The time per token of dominant parallel path and process circuits can be easily reduced by increasing the number of tokens on appropriate OE edges. However, the number of tokens in node circuits cannot be increased for static data flow architectures, and therefore node circuits place a limit in throughput improvement. Also, increasing the number of tokens in recursion circuits requires special consideration. A recursion in the AMG represents data dependency between successive data packets. In general, increasing the number of tokens in recursion circuits requires knowledge of the application algorithm. It is now shown that TBO is for the linear time invariant system of Figure 1 can be improved by increasing number of tokens in the graph's critical recursion circuit [8, 11].

Consider again Figure 3. The critical circuit is a recursion circuit with a time per token value of 4. The recursion circuit contains only one token indicating the initial value of Ax(k-1). The next highest time per token is 3, determined by the node circuit at node 4. If the number of tokens in the recursion circuit can be increased so that the time per token is less than 3, the lower bound for TBO is improved to 3. TBO cannot be less than 3 due to the node circuit at node 4.

Let the state equation represent a h-input, g-output, and q-element state vector system. The dimensions of A, B, C, and D are then (q, q), (h, q), (g, q), and (g, h) respectively.

![Transformed AMG for the linear time invariant system.](Fig. 4)
Now
\[ x(k) = Ax(k-1) + Bu(k) \]
and
\[ x(k-1) = Ax(k-2) + Bu(k-1) \]
so that
\[ x(k) = A\{Ax(k-2) + Bu(k-1)) + Bu(k). \]

It follows from the linearity of the system that
\[ x(k) = (A * A)x(k-2) + (A * B)u(k-1) + Bu(k). \]

If new system matrices are defined as
\[ E = A * A \]
and
\[ F = A * B, \]
then,
\[ x(k) = Ex(k-2) + Fu(k-1) + Bu(k). \]

It should be noted that the dimension of E and A, and F and B, are the same. Therefore, the amount of computation to calculate \(Ax(k-1)\) and Ex(k-2), and Fu(k-1) and Bu(k), are the same. However, if equation (2) is used instead of equation (1) for representing a linear time-invariant system, the recursion circuit has two initial tokens because x(k) is generated from x(k-2). The new AMG based on equation (3), and the original output equation, is shown in Figure 4. The corresponding CMG is shown in Figure 5. \(T_1, T_2, T_3\), and \(T_4\) are initial tokens representing initial condition data on AMG edges. \(T_1\) represents the signal \(S_1 = F * u(k-1)\). \(T_2\) represents the signal \(S_2 = E * x(k-2)\), and \(T_3\) represents the signal \(S_3 = E * x(k-1)\). Extra buffers are inserted to accommodate tokens on the AMG edges. The total number of buffers required on an AMG edge is the summation of initial tokens on the corresponding OE and OF edges of the CMG.

Let \(k = 1, 2, 3, \ldots\) and the initial state vector be \(x(0)\). Therefore, the first input and output are \(u(1)\) and \(y(1)\) respectively. That is, \(u(s) = 0\) for value of \(s\) less than or equal to zero. Therefore, the initial values of \(S_1, S_2, S_3\) correspond to \(k = 1\). Hence, the initial values of signals \(S_1\) and \(S_3\) are \(S_1 = F * u(0) = 0\) and \(S_3 = E * x(0)\). From (2), the signal \(S_2\) is \(Ex(k-2) = x(k) - Fu(k-1) - Bu(k)\). Therefore, the initial value of \(S_2\) is given by \(S_2 = E(1) - Fu(0) - Bu(1)\). As \(u(0) = 0\), the initial value of \(T_2 = x(1) - Bu(1)\). Hence, it follows from the equation (1) that the initial value of \(T_2 = Ax(0) + Bu(1) - Bu(1) = Ax(0)\). Therefore, all the values of the initial AMG tokens can be calculated from the initial state vector. The recursion circuit now consists of read, process, and write transitions of AMG nodes 2 and 4. There are two tokens in that circuit. The computation level of transition 4 has not changed, although that of transition 2 has doubled. Assuming that the processing time of node 2 is doubled, the new time per token ratio of the recursion circuit is \((3+2)/2 = 2.5\). Considering all the circuits of the CMG in Figure 5, TBO\textsubscript{lb} is given by the node circuit of node 4 whose time per token ratio is 3.

**CONCLUSIONS**

ATAMM is a new marked graph model for mapping real-time algorithms onto multicomputer data flow architectures.
Graph time performance is expressed by time between input and output (TBIO) and time between outputs (TBO). Resource independent performance bounds are stated and improved by algorithm transformation. The performance of a linear time invariant system is predicted and improved by the ATAMM model.

There are several topics which are the subject of continuing and future research. The performance model and performance enhancement ideas are validated by simulations and experiments on a testbed [8]. At present, a VHSIC ATAMM data flow architecture based on 1750As is under development [12]. The ATAMM model is being extended to include multiple concurrent instantiations of node operations. An overhead model to account for architecture dependent overhead such as interprocessor communication, contentions for broadcasting bus etc. under development. This model will provide an estimate of overhead on lower bounds for TBIO and TBO. Execution of multiple algorithms is to be investigated. Finally, the performance of algorithms with conditional data flow is a more difficult problem and needs to be analyzed.

ACKNOWLEDGMENT

The work reported here was supported in part by the NASA Langley Research Center under Grants NAG1-683 and NCC1-136.

REFERENCES


