Proc.of IEEE Asilomar Conf. on Signals, Systems, and Computers, Oct. 31 - Nov. 2, Pacific Grove, CA 1994

On the Optimal Blocking Factor for Blocked, Non-overlapped Multiprocessor Schedules[1]

Praveen. K. Murthy, Edward. A. Lee

EECS Department, University of California, Berkeley, CA 94720
{murthy, eal}@eecs.berkeley.edu

Abstract

This paper addresses the issue of determining the blocked non-overlapped multiprocessor schedule of optimal blocking factor for signal processing programs expressed as synchronous dataflow (SDF) graphs.

The main result of this paper is a graph-theoretic characterization of the behavior of the critical path in the precedence graph of blocking factor J as J is increased. We show that the asymptotic behavior is cyclic in the following sense: there exist constants T and ρ such that the critical path in the precedence graph of blocking factor $J + \rho$ has weight given by

$$CP(J + \rho) = CP(J) + \rho\lambda \qquad \forall J > T \qquad \textbf{(EQ 1)}$$

where λ is the maximum cycle mean in the original graph, and ρ is an integer computable from the graph.

1. Introduction

Synchronous Dataflow (SDF) [2] is a subset of dataflow [6] that has proven to be an elegant model for expressing signal processing programs. In the SDF model, the program is represented as a graph where the nodes represent computations and the arcs represent communication channels and precedence constraints. Each actor produces and consumes a fixed number of tokens on each firing, and these numbers are known at compile time. In this paper, we use the homogenous SDF (HSDF) model; in HSDF graphs, each actor produces and consumes one token on each firing. HSDF is not a limitation since any SDF graph can be converted to an HSDF graph [2]. There is also a parameter on each arc that denotes the number of initial tokens present on that arc. In addition, we assume that each node v has a fixed execution time given by $t(v)$. An example of an HSDF graph appears in figure 1, where the

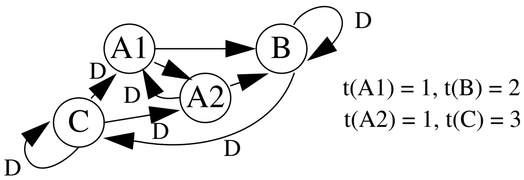

Figure 1. An HSDF graph

presence of one initial token on arc $(C, A1)$ is indicated by a "D" near the arc.

DSP programs are non-terminating in nature since they operate on infinite streams and produce infinite streams. It is well known that a fundamental upper bound on the throughput achievable in an HSDF graph G is given by the inverse of the maximum cycle mean:

$$\lambda_G^H = MAX_{l \in \Lambda}\{E(l)/D(l)\} \qquad \textbf{(EQ 2)}$$

where Λ is the set of all directed cycles in the graph, $E(l)$ is the total computation time of the nodes on cycle l, and $D(l)$ is the total number of delays on cycle l. A cycle that achieves this maximum is called a *critical cycle*. A schedule for an HSDF graph is *rate-optimal* if the iteration-period for the schedule is equal to the maximum cycle mean (also called the *iteration-period bound*). The maximum cycle mean for the graph in figure 1 is 3.5. The quantity λ_G^H can be found in polynomial time using Karps dynamic programming algorithm [14].

From the HSDF graph, we can construct an *acyclic precedence graph* (APG) *of blocking factor one*. This is the graph obtained by deleting all arcs that have one or more delays on them from the original graph. The APG for the graph in figure 1a is shown in figure 2. Notice that the APG shows only the *intra-iteration* precedences between the nodes.

1.This research was supported by ARPA and the US Air Force (under the RASSP program), NSF (MIP9201605), ONT (via NRL), and the California MICRO program.

Define the weight of a path in the APG to be the sum of the execution times of the constituent nodes. Consider the following strategy for constructing a multiprocessor schedule for an HSDF graph. Instead of using the precedence relations specified by the HSDF graph, we will use only the precedences specified by the APG for constructing the schedule. Each node in the APG will be invoked once in the schedule and will be assigned to some processor. Once each processor has finished its tasks, it waits until all other processors have finished their tasks, and then executes its tasks again for the next iteration. This implies that we use some form of barrier synchronization between successive iterations. The first invocation of a node u can only occur if all of its predecessor nodes in the APG have been invoked once. Hence, the total length of the schedule (defined as the maximum of the finishing times for each processor for all of its tasks for one iteration) must be at least equal to the largest-weight path in the APG, the *critical path*. A multiprocessor schedule of this type is called a blocked, non-overlapped schedule of *blocking factor 1*. It is non-overlapped because the n^{th} iteration can only occur after every node has been invoked from the $n-1^{th}$ iteration.

Since a blocked schedule of blocking factor 1 does not exploit *inter-iteration* parallelism, it is useful to schedule the graph over several iterations. A blocked schedule of *blocking factor* J consists of a schedule for the HSDF graph *unfolded* J times. Unfolding a graph J times means considering J successive iterations of the graph. The J-unfolded precedence graph consists of J copies of the APG, and some additional arcs. A node u in the i^{th} copy of the APG, and a node v in the j^{th} copy of the APG, where $j > i$, are connected by an arc in the J-unfolded precedence graph if there is an arc (u, v) in the original graph having $j-i$ delays. The J-unfolded precedence graph can also be referred to as the *APG of blocking factor* J. In this paper, we will use the term "APG" for the APG of blocking factor 1, and the term "J-unfolded graph" interchangeably with the term "APG of blocking factor J" for the J-unfolded precedence graph. Once a J-blocking factor schedule is constructed, it is repeated forever to get a J-periodic schedule. Again, barrier synchronization is assumed between successive blocks of the J-blocking factor schedule.

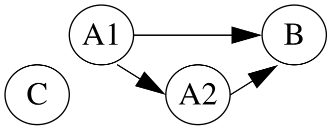

Figure 2. Acyclic precedence graph for graph in figure 1.

It is of interest to determine what the optimal value of J should be in order to construct rate-optimal schedules. For example, the critical path in the graph in figure 1 is the path $A1 \rightarrow A2 \rightarrow B$, and this path has a weight of 4 (recall that the execution times of nodes A, B, C were 1,2, and 3 respectively). Hence, no schedule (of blocking factor 1) for this graph can have an iteration period of less than 4. The acyclic precedence graph for a blocking factor of 2 (or, equivalently, the 2-unfolded graph) is shown in figure 3(a). It can be verified that the weight of the critical path in this graph is 7. Hence, a schedule can be constructed that has an iteration period of 7. Since two iterations of the original graph occur in 7 time units, the iteration period achieved is 7/2=3.5. Therefore, a blocking factor of 2 is optimal for this graph since it is theoretically possible to construct a blocked, rate-optimal schedule. A rate-optimal schedule using two processors is shown in figure 3(b).

When the blocked schedule is implemented, it is not necessary that we actually use barrier synchronization; the assumption is necessary only for analytical tractability. It has been shown in [15] that if the blocked schedule is implemented in a self-timed manner (where the interprocessor communication points (sends and receives) are the only points of synchronization), then some improvement in throughput can result as the schedule unfolds. However, this improvement depends on the particular schedule that is constructed, and there may not be an improvement for some schedules. Of course, there is no improvement if the original blocked schedule (assuming barrier synchronization) is rate-optimal. Assuming barrier synchronization to determine the throughput achievable (by calculating the critical path) gives a worst-case estimate for the actual performance of any blocked schedule.

While *overlapped* schedules [3][4](where every iteration is overlapped with the previous one, enabling full exploi-

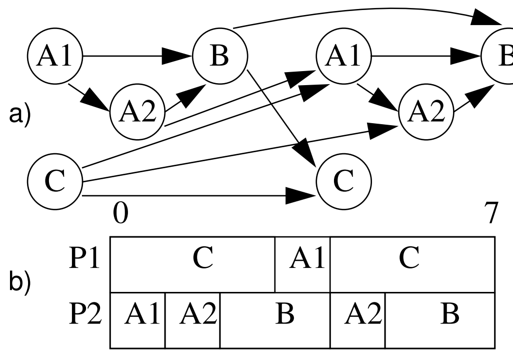

Figure 3. a) Acyclic precedence graph of blocking factor 2 for SDF graph in fig. 1. **b)** A blocked, rate-optimal 2-processor schedule.

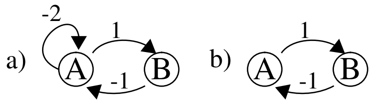

Figure 4. A graph, and its critical graph

tation of inter-iteration parallelism) that meet the throughput bound can always be constructed for HSDF graphs if there are no resource constraints, there has been little work so far for constructing these schedules under processor and communication constraints. In contrast, there are many heuristics that are able to construct good non-overlapped schedules when these constraints are present [7]-[12]. Hence, a practical reason for studying blocked, non-overlapped schedules is to get upper bounds on the performance we can expect when we use these heuristics. In [13], an algorithm is given that systematically increases the blocking factor until the critical path in the unfolded graph is less than some required iteration period. This paper studies the dynamics of increasing the blocking factor in a theoretical framework, and is thus complementary to the results reported in [13].

2. Max Algebra Formulation

Max-plus algebra is an algebra where maximization is the addition operation and addition is the multiplication operation [5]. Max-plus addition will be denoted by $\oplus$ and max-plus multiplication will be denoted by $\otimes$. Thus, $5 \oplus 2 = 5$ and $5 \otimes 2 = 7$. The additive identity in this algebra is $-\infty$, denoted by ε, and the multiplicative identity is 0.

To see how the algebra is relevant to graphs, consider matrices in max-plus. Let A be an $N \times N$ matrix with entries in $\Re \cup \{-\infty\}$. There is an associated graph with N nodes, called the graph of A, where $A(i, j)$ is the weight of the edge (i, j) in the graph. If $A(i, j) = \varepsilon$, then there is no arc between i and j. Conversely, any weighted, directed graph with real-valued weights can be represented by a matrix in max-plus. We use the notation G_A to denote the graph of a matrix A.

An entry in the matrix A^2, where the matrix multiplication is done using max-plus operators, represents the *longest two-arc* path between the corresponding nodes. For example, let

$$A = \begin{bmatrix} -2 & 1 \\ -1 & \varepsilon \end{bmatrix}$$

This represents the graph shown in figure 4 (a). We have

$$A^2 = \begin{bmatrix} -2 & 1 \\ -1 & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} -2 & 1 \\ -1 & \varepsilon \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -3 & 0 \end{bmatrix}.$$

The longest two-arc path from node 1 to node 1 has weight $max(-2-2, 1-1) = 0$. The matrix notation A^k allows us to compactly write down the maximum weight paths of length k between any two nodes in G_A. We can write down an expression for the matrix with maximum weight paths between any pair of nodes. It is given by the matrix

$$A^+ \equiv A \oplus A^2 \oplus A^3 \oplus A^4 \oplus \ldots \oplus A^{N-1} \oplus \ldots \qquad \textbf{(EQ 3)}$$

This series converges only if there are no positive weight cycles in the graph (if there were, we would get paths of arbitrarily large weight by traversing positive weight cycles). Hence, if there are no positive weight cycles, the series can be truncated at $N-1$, where N is the number of nodes, since any path of length greater than N has to traverse a cycle, and the cycle cannot increase the weight. Notice that the implied computation in equation 3 is actually the dynamic programming algorithm for finding the all-pairs longest paths in a graph.

2.1 HSDF graphs in Max-plus

We will assume that the HSDF graph is strongly connected; the case where it isn't requires a slightly more involved model and analysis, and can be found in [1]. HSDF graphs can have arcs with delays; hence we need a way of modeling this. For reasons explained in [1], we only allow 0 or 1 delays to exist on arcs. The case where more than 1 delay exists on an arc can be modelled easily by adding extra nodes [1]. Also, the nodes in an HSDF graph have weights that represent execution times; we need a way of modelling this. The delays can be modeled as follows: let the edge set E in a homogenous graph $G = (V, E)$ be partitioned as $E = E_0 \cup E_1$ where E_i is the set of edges having i delays. Let matrices A_0, A_1 be defined where a matrix A_i represents the HSDF graph (V, E_i). Notice that (V, E_0), the graph corresponding to the matrix A_0, is the APG defined earlier.

The arcs in the graph can be weighted by assigning to each outgoing arc from a node u the weight $t(u)$, the execution time of node u.

Matrix products can be used to find largest weight paths of various types. The entry (i, j) in the matrix product A_0A_1, for instance, represents the largest weight two-arc path between i and j with the first arc being a zero-delay arc and the second being a one-delay arc. This is evident when we write down the expression for the entry:

$$(A_0A_1)(i, j) = max_{1 \le k \le N}(A_0(i, k) + A_1(k, j)).$$

Define

$$A_{N0}^{*} = E \oplus A_0 \oplus A_0^2 \oplus A_0^3 \oplus A_0^4 \oplus \ldots \oplus A_0^{N-1} \quad \textbf{(EQ 4)}$$

where E is the max-plus identity matrix with zeros along the diagonal and ε elsewhere. An entry (i, j), $i \neq j$, in A_{N0}^{*} corresponds to the weight of the largest weight path between nodes i and j in G_{A_0}. To see this, recall that the series in equation 3 could be truncated at $N-1$ if there were no cycles in G_A. Here, A_0 represents the APG, which is acyclic by definition. Since an entry (i, j) in A_0^k represents the maximum weight over all paths of length k between i and j in the APG, the maximum over $1 \leq k \leq N-1$ of the A_0^k gives the maximum weight over all paths between i and j. It follows that the largest element of the matrix A_{N0}^{*} is the critical path in the APG. The inclusion of the identity matrix E in equation 4 means that $A_{N0}^{*}(i, i) = 0 \ \forall i$. This is done to ensure that paths where the first arc is a delay arc will be representable by appropriate matrix products. For example, the matrix product $A_{N0}^{*}A_1$ represents maximum weight paths where the last arc in the path is an arc with one delay. We could have a case where the maximum weight path between a pair of nodes consists of only one arc, and that arc is a delay arc. If equation 4 did not have E in the maximization, we would not be able to represent this case.

Consider now the J-unfolded graph of G, denoted $G^{(J)}$. Recall that $G^{(J)}$ consists of J copies of the graph (V, E_0) and some additional arcs. We will refer to the i^{th} copy of a node $u \in G_{A_0}$ in $G^{(J)}$ as u^i. There is an arc (u^i, v^{i+1}) in $G^{(J)}$ if and only if $(u, v) \in E_1$.

Definition 1: The term $max\{A\}$, where A is a matrix, is defined to be the maximum element of A.

Define $C(J)$ to be an $N \times N$ matrix, where N is the number of nodes in the original HSDF graph, containing the largest weight paths in $G^{(J)}$. This means that an entry (i, j) in $C(J)$ is the maximum weight of all paths between nodes i and j having $J-1$ delays or fewer in the original HSDF graph.

Theorem 1: [1]

$$C(J) = \bigoplus_{k=0}^{J} \left(A_{N0}^{*}A_1\right)^k \qquad J \geq 1 \quad \textbf{(EQ 5)}$$

Since we are only interested in the behavior of the critical path (the maximum element of $C(J)$), we can ignore all of the lower order terms in the summation in equation 5 if the graph is strongly connected since the critical path in $G^{(J)}$ must end at a node in the last copy of the graph (V, E_0). Hence, the maximum element of $C(J)$ will always be from the matrix $\left(A_{N0}^{*}A_1\right)^J$.

Define $B = A_{N0}^{*}A_1$.

Note that the matrix B, defined in equation, corresponds to a graph $G_B = (V, E')$ where V is the set of nodes from the original graph, and $(u, v) \in E'$ iff there is a path in the original graph G from u to v with the last arc in the path being a unit delay arc. The weight of the edge is the maximum of the weights of all such paths.

2.2 Properties of the G_B graph

Denote the maximum cycle mean in the graph of matrix B, $G_B = (V, E)$, by λ_B, and denote the maximum cycle mean for the HSDF graph G as λ_G^H.

Definition 2: The maximum cycle mean for the graph of matrix B, G_B, is defined as

$$\lambda_B = MAX_{l \in \Lambda}\{T(l)/|l|\} \quad \textbf{(EQ 6)}$$

where Λ is the set of circuits, $T(l)$ is the sum of the weights of the arcs on circuit l, and $|l|$ is the number of nodes in circuit l.

Definition 3: A maximal strongly connected subgraph (m.s.c.s) S of G is a strongly connected subgraph of G such that there is no other strongly connected component that properly contains S.

The following lemmas are useful for proving theorem 2, and the proofs may be found in [1].

Lemma 1: If G is strongly connected, then G_B is connected.

Lemma 2: Let $V_d \subseteq V$ be the set of nodes having at least one input delay arc. If G is strongly connected, then G_B has only one m.s.c.s, and the node set of the m.s.c.s is V_d.

Lemma 3: $\lambda_B = \lambda_G^H$, where G is the HSDF graph.

Henceforth, we denote the maximum cycle mean by λ.

3. Cyclicity of B

We are interested in the asymptotic behavior of B^J as J goes to infinity because we are interested in the maximum element of $C(J)$, the critical path. Since B has positive weight cycles, this will be ill-defined since every entry goes to infinity in the limit. Therefore, we normalize B by subtracting λ from each entry. In max-plus notation, this is represented as $\lambda^{-1}B$ (recall that max-algebra has a multiplicative inverse, namely subtraction). The actual weight in B^J can be gotten from the formula $\lambda^J(\lambda^{-1}B)^J$. Hence, every cycle in $\lambda^{-1}B$ has non-positive weight with the critical cycles having 0 weights.

Since we are interested in the critical path of the J-unfolded graph, we are interested in the quantity

$$\begin{aligned} CP(J) &= max\{C(J)\} = max\{\lambda^J\left(\lambda^{-1}B\right)^J\} \\ &= J\lambda + max\{\left(\lambda^{-1}B\right)^J\} \end{aligned},$$

where we have abused notation by combining normal algebra and max-algebra; the term $J\lambda$ uses normal multiplication while the second term uses max-algebraic operations. Define

$$M_B(J) = max\{\left(\lambda^{-1}B\right)^J\}$$

Therefore, the iteration period for the J-unfolded graph, $T(J)$, is given by $T(J) = CP(J)/J = \lambda + M_B(J)/J$. If $M_B(J) \neq 0$, then J is not a rate-optimal blocking factor since $T(J) > \lambda$.

The following definitions apply to generic, weighted digraphs [5]:

Definition 4: The critical graph of a graph G, denoted G^C, is a graph consisting of those nodes and arcs of G that belong to some critical circuit of G. This graph pays a key role in the asymptotic analysis of B^J.

Definition 5: The cyclicity of an m.s.c.s is the greatest common divisor (gcd) of the lengths of all its circuits. The cyclicity of a graph G is the least common multiple (lcm) of the cyclicities of all its m.s.c.s's.

Figure 4(b) shows the critical graph for the graph in 4(a). The cyclicity of the only m.s.c.s in the critical graph is 2; therefore, the cyclicity of the critical graph is 2.

Definition 6: [5] A matrix A is said to be cyclic if there exist d and T such that $\forall m > T, A^{m+d} = A^m$. The least such d is called the cyclicity of matrix A and A is said to be d-cyclic.

For the graph in figure 4, the first few powers of matrix A are given by

$$A = \begin{bmatrix} -2 & 1 \\ -1 & \varepsilon \end{bmatrix}, A^2 = \begin{bmatrix} 0 & -1 \\ -3 & 0 \end{bmatrix}, A^3 = \begin{bmatrix} -2 & 1 \\ -1 & -2 \end{bmatrix}$$

$$A^4 = \begin{bmatrix} 0 & -1 \\ -3 & 0 \end{bmatrix}, A^5 = \begin{bmatrix} -2 & 1 \\ -1 & -2 \end{bmatrix}$$

and we see that $\forall m > 2, A^{m+2} = A^m$. Of course, it is also true that $\forall m > 2, A^{m+d} = A^m$ where $d = 2k$, $k = 1, 2, 3, \ldots$; we pick the least such d, which is 2, and A is 2-cyclic.

Now we state the main result in the paper, namely, the cyclicity of B.

Theorem 2: [1] If the HSDF graph G is strongly connected, then B is ρ-cyclic, where ρ is the cyclicity of $G^C(B)$.

Therefore, we get that

$$\begin{aligned} CP(J+\rho) &= max(C(J+\rho)) \\ &= (J+\rho)\lambda + max\{\left(\lambda^{-1}B\right)^{J+\rho}\} \\ &= \rho\lambda + J\lambda + max\{\left(\lambda^{-1}B\right)^J\} \\ &= \rho\lambda + CP(J) \qquad \forall J > T \end{aligned}$$

Example 1: The HSDF graph, the graph G_B, and the critical graph are depicted in figure 5. From the critical graph, we see that the cyclicity is 6. Through simulation, T turns out to be 8. The critical path has no transient and is cyclic with cyclicity 3 as shown by

$$M_B(1) = 8, M_B(2) = 5, M_B(3) = 4$$
$$M_B(4) = 8, M_B(5) = 5, M_B(6) = 4$$

The iteration period is thus

$$T(J) = \begin{cases} \lambda + 8/J & J = 1, 4, 7, \ldots \\ \lambda + 5/J & J = 2, 5, 8, \ldots \\ \lambda + 4/J & J = 3, 6, 9, \ldots \end{cases}$$

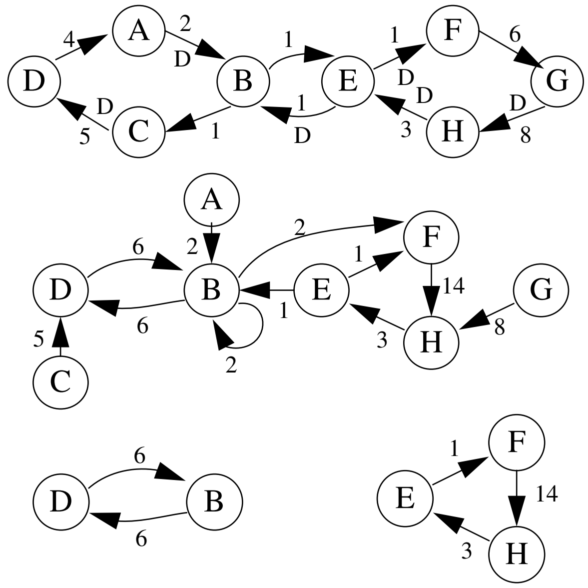

Figure 5. The graph for example 1.

From the above equation, we can see that we will get the best performance if we choose a blocking factor that is a multiple of 3.

In general, the theorem implies that there are only $T + \rho$ blocking factors to consider; this characterizes the critical path for all unfoldings greater than $T + \rho$.

4. Conclusion

In this paper, we have developed a formulation for the problem of finding optimal blocking factors for blocked, non-overlapped schedules. We have used a max-algebra formulation to show that the critical path in a J-unfolded HSDF graph becomes cyclic as J becomes large enough. We have shown that it is possible to determine this cyclicity by analyzing the critical graph of the B-matrix, a matrix that arises out of the model. The cyclicity of the critical path implies that we have to examine only a finite number of unfoldings to determine whether a rate-optimal unfolding exists. This number is equal to the sum of the cyclicity and a transient. While this paper has contributed to our theoretical understanding of the dynamics of increasing the blocking factor when there are no processor or communication constraints, finding the optimal blocking factor under those constraints is still an open problem. The results in this paper certainly give us a technique to determine upper bounds on the performance we can expect when these constraints are present.

5. References

[1] P. K. Murthy, E.A. Lee, "On the Optimal Blocking Factor for Blocked, Non-Overlapped Schedules," Memo. No. UCB/ERL M94/46, Electronics Research Lab., UC Berkeley, June 1994

[2] E. A. Lee, "A Coupled Hardware and Software Architecture for Programmable Digital Signal Processors", Ph.D. Thesis, UC-Berkeley, 1986

[3] K. K. Parhi, D. G. Messerschmitt, "Static Rate-Optimal Scheduling of Iterative Data-Flow Programs via Optimum Unfolding", IEEE Transactions on Computers, February 1991

[4] L. F. Chao, E. M. Sha, "Static Scheduling for Synthesis of DSP Algorithms on Various Models", Technical Report, Princeton University, 1993

[5] F. Baccelli, G. Cohen, G. J. Olsder, J. P. Quadrat, "Synchronization and Linearity", Prentice Hall, 1993

[6] J. B. Dennis, "First Version of a Data Flow Procedure Language", MAC Technical Memo. 61, Laboratory for Computer Science, MIT, May 1975

[7] G. Sih, "Multiprocessor Scheduling to Account for Interprocessor Communication", Ph.D. Thesis, Memo. No. UCB/ERL M91/29, Electronics Research Lab., UC Berkeley, April 1991

[8] T. L. Adam, K. M. Chandy, J. R. Dickson, "A Comparison of List Schedules for Parallel Processing Systems", Comm. ACM 17 (12), December 1974

[9] T. C. Hu, "Parallel Sequencing and Assembly Line Problems", Operations Research 9 (6), 1961

[10] W. H. Kohler, "A Preliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiprocessor Systems", IEEE Trans. on Computers, December 1975

[11] G. Liao, G. Gao, E. Altman, V. K. Agarwal,"A Comparative Study of DSP Multiprocessor List Scheduling Heuristics", Technical Report, McGill University, 1993

[12] J-J. Hwang, Y-C Chow, F. D. Anger, C-Y. Lee, "Scheduling Precedence Graphs in Systems with Interprocessor Communication Times", Siam J. Computing, Vol. 18, No. 2, April 1989

[13] L. E. Lucke, K. K. Parhi, "Data-Flow Transformations for Critical Path Time Reduction in High-Level DSP Synthesis", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, No. 7, July 1993

[14] R. Karp, "A Note on the Characterization of the Minimum Cycle Mean in a Digraph", Discrete Mathematics, 23, 1978

[15] S. Sriram, "Statically Scheduling Communication Resources in Multiprocessor Architectures", Asilomar Conf., October 1994.