RATE-OPTIMAL SCHEDULING FOR CYCLO-STATIC AND PERIODIC SCHEDULES †

Liang-Fang Chao
lfc@iastate.edu
Dept. of Elec. & Computer Engr.
Iowa State University
Ames, Iowa 50011

Edwin Hsing-Mean Sha
hms@cad.cse.nd.edu
Dept. of Computer Science & Engr.
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

In order to realize DSP applications on multi-processor systems with the optimal throughput, properties and efficient techniques need to be derived. Rate-optimal scheduling with minimum unfolding has been studied in the past for static schedules only. The scheduling models called cyclo-static and periodic schedules allow more flexibility on processor assignment. This paper derives the minimum unfolding factors to achieve rate-optimal schedules for cyclo-static and periodic schedules. The necessary and sufficient conditions for the existence of these schedules are also derived. From these results, it is shown that unfolding is necessary under these two models for certain data-flow graphs to achieve rate-optimality. Furthermore, all the theorems are proved in a constructive way, in which an efficient shortest-path algorithm is used for scheduling.

1 INTRODUCTION

Many modern signal processing applications require high throughput. In order to satisfy such a high throughput, the techniques to achieve optimal throughput based on multiple processor systems have great importance. The scheduling of iterative algorithms or loops in a program is usually critical to the system performance. *Unfolding* is one of the most important techniques to obtain a rate-optimal schedule. In order to improve execution rate, we can schedule f iterations in an instance of a static schedule. This is called *unfolding* (or unrolling) by an *unfolding factor* f. Each schedule instance contains f iterations. Therefore, the *iteration period*, which is the average computation time per iteration can be reduced.

The *iteration bound* $B(G)$, a lower bound of iteration period, is defined as the maximum time-to-delay ratio of every loop in the DFG [1]. A schedule is *rate-optimal* if the iteration period of this schedule equals the iteration bound. An unfolding factor is rate-optimal if the corresponding schedule is rate-optimal.

Much work has been done on finding rate-optimal unfolding factors for *static schedules* [2, 3, 4]. In [4], the minimum rate-optimal unfolding factors are derived for both pipelined and non-pipelined processor models. The model of processor assignment in these papers is *static*, i.e. the processor assignment is the same for every iteration.

The *cyclo-static schedules* and *periodic schedules*, proposed in [5, 6, 7], allow more flexibility on processor assignments. A general permutation function for a periodic schedule and a restrictive permutation function (a cyclo-shift function) for a cyclo-static schedule are used to indicate the sequence of processor assignments. Although it is suspected that cyclo-static and periodic schedules tend to use smaller unfolding factors, even no unfolding, the properties of their rate-optimal unfolding factors have been unknown.

This paper derives the minimum unfolding factors to achieve rate-optimal schedules for cyclo-static and periodic schedules. The necessary and sufficient condition for the existence of cyclo-static (or periodic) schedules is derived for these two processor models. From these results, we show that unfolding is necessary for these two models for certain data-flow graphs to achieve rate-optimality, and we can characterize such graphs and their minimum rate-optimal unfolding factors. Furthermore, all the theorems are proved in a constructive way, in which the $O(|V||E|)$-time scheduling algorithms are presented, where $|V|$ is the number of operations and $|E|$ is the number of precedence relations in a given specification of the application.

2 MODELS AND TERMINOLOGIES

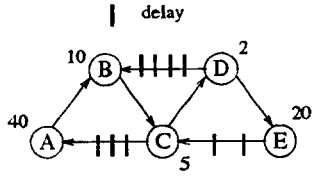
We first introduce the graph model and scheduling model for iterative or recursive algorithms. Then, the three processor sequencing models are characterized using a permuting function. Finally, two processor implementation models we consider are introduced.

A data-flow graph (DFG) is represented by a directed weighted graph $G = (V, E, d, t)$ where V is the set of computation nodes, E is the edge set which defines the directed edges (or precedence relation) from nodes in V to nodes in V, and $d(e)$ is the delay count for edge $e \in E$. Each node v in V is associated with a positive integer $t(v)$ which may represent the computation time for the node v. The graph in Figure 1 is an example of DFG, where the number attaches to a node is its computation time.

The delay count, say i, on an edge (u, v) represents the

†This work was supported in part by NSF Research Initiation Award MIP-9410080 and ORAU Junior Faculty Enhancement Award.

0-7803-2431-5/95 $4.00



$\mathcal{B}(G) = \frac{55}{3} = 18\frac{1}{3}$

Figure 1: A data-flow graph from [8]

sequenced relation between computation nodes u and v. For a meaningful data-flow graph, the total delay count of any loop is nonzero. The execution of all computation nodes once in V is called one *iteration*. An edge e from u to v with delay count $d(e)$ means that the computation of node v at iteration j depends on the computation of node u at iteration $j - d(e)$.

The scheduling problem and assignment problems are considered separately. The scheduling problem allocates nodes onto the time dimension, while the assignment problem allocates nodes onto the processor space. We optimize the problem instance in the time space to achieve rate-optimality, and then realize the scheduled instance under different processor models and different processor sequencing models.

Time and processor schedules of a DFG in a parallel system are formally modeled as follows. A *time schedule* is represented by a function $S : V \times N \longrightarrow N$. The starting time of node v in the i-th iteration is $S(v, i)$. A time schedule is *legal* if for every edge $u \xrightarrow{e} v$ and iteration i, we have $S(u, i) + t(u) \leq S(v, i + d(e))$. A time schedule is said to have an *unfolding factor* f and a *cycle period* c if $S(v, i + f) = S(v, i) + c$ for every v in V and iteration i. Thus, such a time schedule can be represented by the partial schedule of the first f iterations. A new instance of this partial schedule of f iterations can be initiated for every interval of length c to form a legal complete schedule.

A *processor assignment*, also called *processor schedule*, is represented by a function $P : V \times N \longrightarrow L$, where L is the set of processors, numbered $1, 2, \ldots,$ to $|L|$. Node v in the i-th iteration is assigned to Processor $P(v, i)$. There are three processor sequencing models: *static, cyclo-static, periodic*. A processor schedule with unfolding factor f is *static* if $P(v, i+f) = P(v, i)$ for every iteration i. A processor schedule is *periodic* if there exists a *permuting function* $Q : L \longrightarrow L$ such that $P(v, i + f) = Q(P(v, i))$ for every v in V and iteration i. Nodes scheduled on processor j in the i-th iteration will be scheduled to processor $Q(j)$ in the $(i+f)$-th iteration.[1] A simpler form of periodic schedules, called *cyclo-static schedules* [5], can be specified with a single processor displacement δ $Q(p) = (p + \delta)\%|L|$ for every processor p.

The *iteration period* of a time schedule is the average computation time per iteration. A time schedule with cycle period c and unfolding factor f has iteration period c/f. We are interested in finding static schedules with minimum iteration period. The *iteration bound* $\mathcal{B}(G)$, a lower bound of iteration period, is defined as the maximum time-to-delay ratio of every loop in the DFG [1]. If the value of c/f equals the iteration bound $\mathcal{B}(G)$, we call such an f to be a *rate-optimal unfolding factor*.

In the work by [5, 6, 7], so-called *time-optimal* (and processor optimal) cyclo-static or periodic schedules are studied. However, the iteration period bound of time-optimality is defined using the *ceiling* of $\mathcal{B}(G)$, i.e. the least integer greater than or equal to $\mathcal{B}(G)$. Such time-optimal schedules are "optimal" only when unfolding is not considered. Therefore, a time-optimal schedule might have larger iteration period than a rate-optimal schedule.

The two processor models we consider are *pipelined and non-pipelined implementations*. In the non-pipelined model, the next copy of a node cannot start execution before the previous copy has finished execution. For the pipelined model, there is no restriction on the scheduling of copies of the same node besides precedence relations. Although the integral timing model is assumed throughout this paper, all the results can be generalized to the fractional timing model as defined in [4].

3 PRELIMINARIES

This section summarizes some results from [4] for rate-optimal static scheduling. These theorems, rephrased in the context of this paper, will form the basis of the technical proofs in the next section.

The necessary and sufficient condition is derived in [4] for a static schedule to be implemented under the non-pipelined processor model: As long as the cycle period of a schedule is is no less than $\max_v t(v)$, it can be implemented by a static schedule with large enough unfolding factor under the non-pipelined design.

Theorem 1 [4] *Let G be a DFG, and S be a time schedule with cycle period c and unfolding factor f. The time schedule S can be realized by a static processor assignment P under the non-pipelined design if and only if $c \geq \max_v t(v)$.*

An algorithm is presented in [4] to find a schedule for given cycle period c and unfolding factor f under the fractional and integral models. We briefly summaries the results here.

For a DFG $G = (V, E, d, t)$ and given c and f, we define a modified graph with different weights on edges: scheduling graph $G^s = (V, E, w)$ where $w(e) = d(e) - t(u) \cdot f/c$ for every edge $u \xrightarrow{e} v$ in E. Assume that there is no negative-weight cycle in the scheduling graph G^s. We add a node v_0 and directed edges from v_0 to every other node with zero weight in G^s. Let $sh(v)$ be the length of the shortest path from v_0 to v in the scheduling graph G^s. The values of $sh(v)$ can be computed by any single-source shortest path algorithm. It is easy to observe that for every node v we have $sh(v) \leq 0$, and there exists a node u in V such that $sh(u) = 0$.

The following theorem obtains a legal time schedule from the scheduling graph.

[1] The cycling vector used in [7] equals $(Q^{-1}(1)\ Q^{-1}(2)\ \ldots Q^{-1}(|L|))^T$.

Theorem 2 [4] *Let G be a general-time DFG, c a cycle period, f an unfolding factor. Assuming that there is no negative-weight cycle in the corresponding scheduling graph G^s, let $sh(v)$ be the length of the shortest path from v_0 to v. There exists a legal time schedule S^i with cycle period c and unfolding factor f: $S^i(v,i) = \lceil -sh(v) \cdot c/f + i \cdot c/f \rceil$ for every v and i.*

The following theorem provides the necessary and sufficient condition for the existence of a time schedule with cycle period c and unfolding factor f. No assumption on processor models is made. The schedules can be derived with Theorem 2 for a given cycle period and an unfolding factor.

Theorem 3 [4] *Let G be a general-time DFG, c a cycle period, and f an unfolding factor. $c/f \geq \mathcal{B}(G)$ if and only if there exists a legal time schedule with unfolding factor f and cycle period c.*

4 RATE-OPTIMAL SCHEDULES

This section shows the criteria of rate-optimal schedules and the rate-optimal unfolding factors under cyclo-static and periodic scheduling models. We first summarize the main results in Theorems 4 and 5 and then provide the formal proofs.

4.1 Main Results

The necessary and sufficient condition for the existence of cyclo-static (resp. periodic) schedules and the minimum rate-optimal unfolding factor are derived in Theorem 4 (resp. Theorem 5).

Theorem 4 *Let G be a DFG and ρ the denominator of $\mathcal{B}(G)$ in its irreducible form.*

(a) *There exists a cyclo-static schedule with unfolding factor f and cycle period c if and only if $c/f \geq \mathcal{B}(G)$.*

(b) *The minimum rate-optimal unfolding factor for cyclo-static scheduling under the non-pipelined implementation is ρ.*

Proof: Part (a) can be derived from Theorem 3 and Lemma 8, to be proved in the next subsection.

Let σ be the numerator of $\mathcal{B}(G)$ in its irreducible form. From Part (a), we know ρ is a rate-optimal unfolding factor for a schedule with cycle period σ. Since σ/ρ equals to $\mathcal{B}(G)$ and is irreducible, ρ is the minimum rate-optimal unfolding factor. Thus, Part (b) is proved. □

This theorem shows that unfolding is necessary to achieve rate-optimality on the non-pipelined processor model even for cyclo-static schedules. This situation happens when the iteration bound $\mathcal{B}(G)$ is not integral. Similar results are proved for periodic schedules.

Theorem 5 *Let G be a DFG and ρ the denominator of $\mathcal{B}(G)$ in its irreducible form.*

(a) *There exists a periodic schedule with unfolding factor f and cycle period c if and only if $c/f \geq \mathcal{B}(G)$.*

(b) *The minimum rate-optimal unfolding factor for periodic scheduling under the non-pipelined implementation is ρ.*

Proof: This theorem is derived from Theorem 3 and Lemma 8. □

These theorems are proved by constructing cyclo-static schedules (or periodic schedules) on the non-pipelined model from static schedules on the pipelined models.

4.2 Proofs

We show that cyclo-static or periodic schedules in the non-pipelined processor model have equivalent time schedules as static schedules in the pipelined processor model. Their processor assignments are different because different processor implementation and sequencing models are used. Thus, the results for rate-optimality of static schedules under the pipelined model (Theorem 3) can be extended to cyclo-static and periodic schedules. Without loss of generality, we prove lemmas in this section with unfolding factor 1.

First, we characterize the pipelined processors to derive the number of pipeline stages from the given time schedule.

Definition 1 *Let G be a DFG and S a time schedule of G with cycle period c. Let P be a processor assignment realized on processors $M_1, \ldots, M_{|L|}$. Define $ET(l)$ (resp. $LT(l)$) be the earliest starting time (resp. the latest finishing time) of nodes in the first iteration allocated to processor M_l.*

We can characterize $ET(l)$ and $LT(l)$ as $ET(l) = \min\{S(v,1) |$ for every v in V such that $P(v,1) = M_l\}$ and $LT(l) = \max\{S(v,1) + t(v) - 1 |$ for every v in V such that $P(v,1) = M_l\}$. If the time schedule has cycle period c, the earliest starting time of the i-th iteration is $ET(l) + (i-1)c$.

The following lemma derives the number of stages needed for the pipelined processors to realize a time schedule.

Lemma 6 *Let G be a DFG and S a time schedule of G with cycle period c. The schedule S is realized on processors $M_1, \ldots, M_{|L|}$ in the pipelined model by a static processor assignment P such that M_l has $stage(l)$ pipeline stages. It can be derived that $stage(l) = \left\lceil \frac{LT(l) - ET(l) + 1}{c} \right\rceil$ or $LT(l) < stage(l) \cdot c + ET(l)$.*

Proof: From the time schedule S, the pipeline structure of each processor can be decided. The number of pipeline stages, $stage(l)$, needed in each processor M_l equals the maximum number of overlapping schedule instances. Since the time schedule has cycle period c, the earliest starting time of the i-th iteration is $ET(l) + (i-1)c$. We can derive $stage(l)$ as the largest i such that $ET(l) + (i-1)c \leq LT(l)$, i.e. $stage(l) = \left\lceil \frac{LT(l) - ET(l) + 1}{c} \right\rceil$. Since $LT(l) - ET(l) + 1 \leq stage(l) \cdot c$, we have $LT(l) \leq stage(l) \cdot c + ET(l) - 1$; therefore, $LT(l) < stage(l) \cdot c + ET(l)$. □

From this lemma, a processor M_l is not pipelined if $LT(l) < c + ET(l)$, i.e. $LT(l) - ET(l) + 1 \leq c$. A static processor assignment can be realized in non-pipelined model if $LT(l) < c + ET(l)$ for every processor M_l.

This lemma shows that, for any static schedule under the pipelined model, a cyclo-static schedule with the same time schedule can be constructed for the non-pipelined model.

Lemma 7 *Let G be a DFG and S a time schedule of G with cycle period c. Let P be a static processor assignment on processors, denoted by $M_1, M_2, \ldots, M_{|L|}$, to realize S in the pipelined model. There exists a cyclo-static processor assignment P' with processor displacement δ to realize S in the non-pipelined model.*

Proof: Let maxstage be the maximum number of pipeline stages in any pipelined processor, i.e. $maxstage = \max_l stage(l)$. We construct the cyclo-static processor assignment P' as $P'(v,1) = P(v,1)$ and $P'(v,i) = Q^{i-1}(P(v,i))$ for every node v in V and iteration i, where $Q(p) = (p+\delta)\%|L'|$, where $\delta = |L|$ and L' is a set of $|L| \cdot$ maxstage processors. It is easy to prove that this cyclo-static processor schedule can be realized in the non-pipelined model. □

Since cyclo-static schedule is a special case for periodic schdule, the following lemma can be easily proved. However, we will give another proof which gives less number of processors.

Lemma 8 *Let G be a DFG and S a time schedule of G with cycle period c. Let P be a static processor assignment on processors, denoted by $M_1, M_2, \ldots, M_{|L|}$, to realize S in the pipelined model. There exists a periodic processor assignment P' to realize S in the non-pipelined model.*

Proof: From Lemma 6, each pipelined processor M_l can be implemented by $stage(l)$ non-pipelined processors, denoted by $M_{l,1}, M_{l,2}, \ldots, M_{l,stage(l)}$. We will construct a periodic processor assignment P' with a permuting function Q on the set of non-pipelined processors $L' = \{M_{l,1}, \ldots, M_{l,stage(l)} \mid$ for every l from 1 to $|L|\}$. In the function Q, non-pipelined processors corresponding to the same pipelined processor M_l form a cycle of size $stage(l)$: $Q(M_{l,j}) = M_{l,j+1}$ for $j = 1, \ldots, t-1$ and $Q(M_{l,t}) = M_{l,1}$. Since each non-pipelined processor corresponds to only one pipelined processor, the function Q is a permutation. The processor assignment is defined as $P'(v,1) = M_{l,1}$ if $P(v,1) = M_l$ and $P'(v,i) = Q^{i-1}(P'(v,1))$ for every $i \geq 2$.

Using Lemma 6, we prove that the schedule P' can be realized on non-pipelined processors, i.e. an iteration can not be scheduled on a processor before the previous iteration finishes. Therefore, the constructed periodic processor assignment P' can be realized in the non-pipelined model with $\sum_{l=1}^{l=|L|} stage(l)$ processors. □

Note that the cyclo-static and periodic schedules constructed in the proofs may not be processor-efficient. Since our goal is to prove the criteria of rate-optimal time schedules under different processor allocation models, the issue of processor-optimality is not considered.

5 EXAMPLES

The example in Figure 1 from [8] has iteration bound $\mathcal{B}(G) = 55/3$. From Theorems 4 and 5, we know that an unfolding of factor 3 is necessary to achieve rate-optimality for cyclo-static or periodic schedules in the non-pipelined process implementation. For the 4-stage lattice filter used in [9], $\mathcal{B}(G)$ is $3/2$; thus, an unfolding of factor 2 is necessary. For the least-mean-square filters, normalized lattice and orthogonal lattice filters used in [7], $\mathcal{B}(G)$ are all integral. Therefore, rate-optimal schedules can be derived without unfolding.

References

[1] M. Renfors and Y. Neuvo, "The maximum sampling rate of digital filters under hardware speed constraints," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, pp. 196–202, Mar. 1981.

[2] K. K. Parhi and D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Transactions on Computers*, vol. 40, pp. 178–195, Feb. 1991.

[3] D. J. Wang and Y. H. Hu, "Optimal scheduling of linear recurrence equations on a multi-processor array," in *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, (Toronto), pp. 1581–1584, May 1991.

[4] L.-F. Chao and E. H.-M. Sha, "Static scheduling for synthesis of dsp algorithms on various models," *Journal of VLSI Signal Processing*, June 1994. Accepted for publication.

[5] D. A. Schwartz and T. P. Barnwell III, "Cyclo-static multiprocessor scheduling for the optimal realization of shift-invariant flow graphs," in *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, vol. 3, pp. 1384–1387, 1985.

[6] D. A. Schwartz and T. P. Barnwell III, "Cyclo-static solutions: optimal multiprocessor realizations of recursive algorithms," in *VLSI Signal Processing II* (K. S.Y, R. Owen, and J. Nas, eds.), New York: IEEE Press, 1986.

[7] P. R. Gelabert and T. P. Barnwell, "Optimal automatic periodic multiprocessor scheduler for fully specified flow graphs," *IEEE Transactions on Signal Processing*, vol. 41, pp. 858–888, Feb. 1993.

[8] L. E. Lucke, A. P. Brown, and K. K. Parhi, "Unfolding and retiming for high-level DSP synthesis," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 2351–2354, 1991.

[9] J.-G. Chung and K. Parhi, "Design of pipelined lattice IIR digital filters," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, pp. 1021–1025, Nov. 1991.