UNFOLDING PROBABILISTIC DATA-FLOW GRAPHS UNDER DIFFERENT TIMING MODELS

Sissades Tongsima *Timothy W. O'Neil* *Edwin H.-M. Sha*

Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

It is known that in many applications, because of selection statements, e.g., if-statement, the computation time of a node can be represented by a random variable. This paper focuses on any iterative application (containing loops) reflecting those uncertainties. Such an application can then be transformed to a probabilistic data-flow graph. A challenging problem is to derive graph transformation techniques which can produce a good schedule. This paper introduces two timing models, the time-invariant and time-variant models, to characterize the nature of these applications. Furthermore, for the time-invariant model, we propose a means of selecting a minimum rate-optimal unfolding factor which guarantees the best schedule length. We also propose a good estimation for choosing an unfolding factor for a graph under the time-variant model.

1. INTRODUCTION

A considerable number of iterative or recursive applications contain conditional statements, such as if and switch-case statements, where a data-flow model, called a *data-flow graph* (DFG), can be used to represent these applications. We can identify a conditional statement as a node in a DFG whose timing information depends on the probability of selecting the branches at the beginning of the conditional statement. Since the computation time of such a node is represented by a random variable, we call this type of node a probabilistic node. The *probabilistic data-flow graph* (PDFG) is introduced to model these applications.

We can further categorize the graph based upon the behavior of its application. In this paper, we introduce two timing models for a PDFG, a *time-invariant* (**TI**) model and a *time-variant* (**TV**) model, which describe how probabilities influence the node computation times. The applications under the time-invariant model are those programs whose branching decisions rely on inputs from outside world, also called the environmental variables. For example, the throughput of a modem may be changed during different sessions and in a session it is determined based on the connection quality and bandwidth. But after the connection is established, the settings are kept the same throughout the session.

On the other hand, applications which are considered to be under the time-variant model do not ensure the aforementioned setup. For example, conditional statements placed in a loop depend on some intermediate results produced by the program itself or even some input from users. The examples in this category range in many areas including artificial intelligence, fuzzy logic, and image processing etc. We discuss the issue of timing models and their properties in this paper.

Due to the uncertainty inherent in a PDFG which implies the presence of more than one possible graph outcome whose timing information is non-probabilistic, optimizing the graph, i.e., increasing the degree of parallelism of nodes in this graph, by traditional methods such as unfolding [5] becomes nontrivial. Since unfolding increases the size of a resulting unfolded graph, it is beneficial to keep the unfolding factor small. This paper discusses how to select a reasonably small unfolding factor for different timing models. How processors schedule an application for execution—either using an *integral* grid time slot in which a task has to be scheduled at the beginning of the grid or using a grid-less (*fractional*) model where a task can start execution anytime—differentiates two model candidates [1]. This paper demonstrates that if we consider a PDFG under the time-invariant model, a minimal unfolding factor which minimizes the schedule length of each possible graph outcome of the PDFG can be obtained. We also show that the traditional iteration bound is no longer a lower bound when considering a PDFG under time-variant models. A suggestion for choosing a decent unfolding factor for graphs under the time-variant model is then presented.

Several scheduling techniques strive to schedule a restricted version of the DFGs, called a *directed acyclic graph* (DAG) [2]. These methods do not explore the parallelism across iterations nor do they address the problem of probabilistic tasks. As the results, several techniques attempt to make use of the fact that the execution of tasks in different loop iterations can be done in parallel such as loop transformations and software pipelining [3, 4, 7].

The remaining sections are organized as following. In Section 2, we discuss some terminology and definitions. Section 3 then presents two timing models which influence how a PDFG will be executed. The concept of optimal schedule length is given for the time-invariant model in this section. We also propose how to choose an unfolding factor which will be beneficial for a PDFG under different timing models. Finally, Section 4 draws conclusions of this work.

2. BACKGROUND

Figure 1 shows a simple data-flow graph (DFG) where a computation time of a node is represented by a number beside them, e.g., node n_0 takes 3 time-units to compute and delays are shown by a number on some edges. In literature, the iteration bound [6] of such a graph is defined to be the maximum time to delay ratio of

all cycles in it. By this definition, the iteration bound of this graph can be computed by maximizing the time-to-delay ratio of both cycles in the graph. As a result, the cycle composed of nodes n_1 and n_2 has a greater ratio (3) than that of any other cycle (1.5) in G, and 3 is therefore the iteration bound.

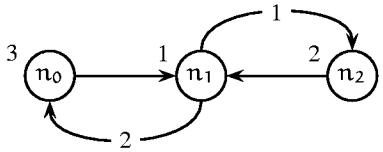

Figure 1: A simple data-flow graph

When implementing a static schedule, we must consider the timing models are architecture If each of our operations must start at an itegral point, we say we have an *integral model*. Otherwise, an instruction can begin in a fractional point and we have a *fractional model* in this case. Second, under either timing model, we can implement our schedule using one of two hardware designs, the *pipelined* or *non-pipelined* architecture. For the pipelined architecture, we allow a second instance of a node to begin execution even before the first instance has finished. On the other hand, the non-pipelined architecture requires the first instance to have completed execution before the second instance begins. In other words, for any node v, we must have $S(v,i) + t(v) \leq S(v,i+f)$ where $S(v,i)$ represents a starting time of node v in i^{th} iteration. In this paper, we assume that our target architecture is based on the non-pipelined implementation.

Definition 2.1. *The optimal schedule length of a DFG G with respect to an unfolding factor f is defined as the minimum cycle period of all legal schedules of G with unfolding factor f, and is denoted by $l^f_{opt}(G)$.*

From the definition of the iteration bound, we know that if the ratio of the cycle period c to the unfolding factor f, also called the *iteration period*, of a schedule S equals $B(G)$, this schedule is a *rate-optimal* schedule.

Definition 2.2. *A probabilistic data-flow graph (PDFG) is a node-weighted edge-weighted directed graph $G = \langle V, E, d, T\rangle$ where V is a set of nodes, E is a set of edges, d is a function from E to $\mathbb{N}$ representing a number of delays on each edge, and $T(v)$ is an independent random variable representing the computation time of node v.*

Since the graph we must work with cannot be determined at the outset of our program execution, we can view a PDFG as a random experiment. Hence we will adopt the terminology of probability theory and define an *outcome* as the non-probabilistic data-flow graph that results when the computation times of the nodes in the PDFG are fixed. The *sample description space* $\mathfrak{S}$ is then the set of all possible outcomes. Figure 2 illustrates a probabilistic data-flow graph and its two possible graph outcomes, E_0 and E_1 where the probabilities associated with the first and second graphs are 0.2 and 0.8 respectively. A probabilistic iteration bound is defined as following:

Definition 2.3. *The iteration bound of a PDFG G, denoted by $B_p(G)$, is a random variable where,*

$$B_p(G) = \max\left\{\frac{T(l)}{D(l)} : l \in G \text{ is a cycle}\right\}.$$

Note that in a probabilistic case the summation of computation times in each loop $T(l)$ is a random variable. The iteration bound for a non-probabilistic DFG is the maximum of all loop-ratios; the iteration bound of a PDFG can be computed by the same definition. One way to explain how it works under the probabilistic graph model is to calculate an iteration bound of each outcome G' in the sample description space. As an example, for the graph outcome E_0 (see Figure 2) $B(E_0) = 3$ with probability 0.2 and for graph E_1, $B(E_1) = 4$ with probability 0.8.

Let $\mathfrak{S}$ be a sample description space and $X(s) = x$ be a random variable for all $s \in \mathfrak{S}$. For notational purposes, we use $\{X = x\}$ to denote the function X that maps onto a set of points in $\mathfrak{S}$. The probability of these events is then represented by $\mathsf{Prob}(X = x)$. Without loss of generality we shall use a notation $\{(x_0, p_0), \ldots, (x_n, p_n)\}$, or $\{(x_i, p_i)\}$ for brevity, to represent a random variable X and its induced probability $\mathsf{Prob}(X = x_i) = p_i$. As an example, the probabilistic iteration bound from the graph in Figure 2 can be represented as $B_p(G) = \{(3, 0.2), (4, 0.8)\}$.

3. TIME-INVARIANT AND TIME-VARIANT MODELS

Definition 3.1. *The time-invariant (**TI**) model of a PDFG G is a model in which the computation time of each node is established prior to beginning execution of the graph and does not change throughout execution. In this model, an outcome of G, called G', is predetermined from computation times of all nodes $v \in V$ with a probability $p = \prod_v \mathsf{Prob}(T(v) = x)$ where x is a possible computation time.*

Having defined the optimal schedule length for any scheduling event whose node-computation times are fixed, we now extend this concept to realize an optimal schedule length for a probabilistic case.

Definition 3.2. *The optimal schedule length of a PDFG G with respect to an unfolding factor f, denoted by $L^f_{opt}(G)$, is a random variable where, for each outcome G' with probability $p = \mathsf{Prob}(L^f_{opt}(G) = l^f_{opt}(G'))$.*

Under the **TI** model, each of the possible graph outcomes in Figure 2 has a corresponding scheduling diagram. The optimal schedule length for this probabilistic graph can be denoted using our random variable notation introduced previously. For example, the optimal schedule length for the graph in Figure 2 with respect to an unfolding factor 2 is $\{(3, 0.2), (4, 0.8)\}$. Later we demonstrate that any PDFG under this model can have its optimal schedule with respect to an unfolding factor f.

Definition 3.3. *The time-variant (**TV**) model of a PDFG G is a model in which the execution time of a node in each iteration follows the probabilistic distribution of $T(v)$. The execution time of each iteration is, therefore, not predetermined.*

The computation time of a probabilistic node inside the loop is changed depending on the probability associated with the node. If a graph under the **TV** model is unfolded by f (i.e., there are f copies of node v in one iteration), each of the copies of node v

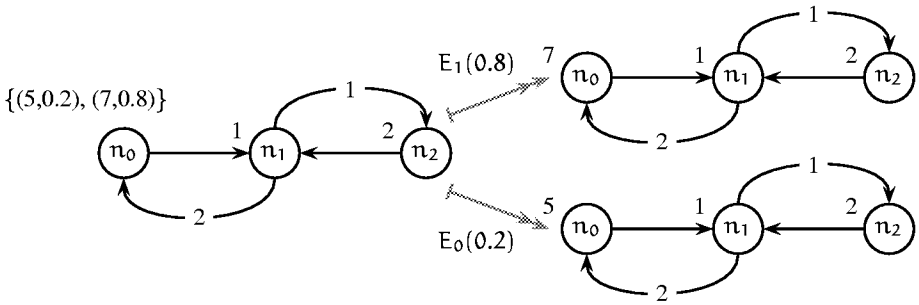

Figure 2: An example of a probabilistic data-flow graph

may take different computation times depending on the random variable $T(v)$ while under the **TI** model, all the copies of node v will assume the same computation time for each graph outcome. Furthermore, the iteration bound definition is not valid under this model. To demonstrate this, let us consider the graph construct of Figure 2 that we have been using but changing the timing assignment of each node to those presented in Table 1. In this table, two timing assignments are shown in columns "pattern 1" and "pattern 2" respectively. The last row of the table presents the probabilistic iteration bound of each pattern.

	pattern 1	pattern 2
$T(n_0)$	12	4
$T(n_1)$	$\{(1, 0.3), (2, 0.7)\}$	1
$T(n_2)$	1	$\{(1, 0.3), (2, 0.7)\}$
$B_p(G)$	$\{(6.5, 0.3), (7, 0.7)\}$	$\{(2.5, 0.3), (3, 0.7)\}$

Table 1: New timing assignments for graph in Figure 2

Assuming that an unfolding factor 2 is chosen to unfold the graph. Let us consider the scenario when pattern 1 is the timing assignment of the graph. Due to the **TV** model, there will be 4 possible unfolded graph outcomes. In particular, the difference can be classified as the change of node n_1's computation time. For comparison purposes, we compute iteration periods—an average computation time per iteration—for each of the outcomes. For this example, it can be done by taking the sum of computation time of the longest path and dividing it by the unfolding factor. The average iteration period from this example is 6.955 which is greater than the average iteration bound of 6.85 (refer to Definition 2.3).

Using the same input as described previously, let us consider what happens when the graph from Figure 2 assumes the timing of pattern 2. After examining all possible outcomes, we have its average iteration period equal to 2.745 which is smaller than its average iteration bound (2.85). This shows that the traditional definition of iteration bound is no longer the lower bound for the **TV** model.

3.1. Properties for time-invariant model

Under this model constraint, we can obtain some nice properties based on how nodes start their execution as mentioned previously in Section 2: *integral-grid* and *fractional-grid* models. The properties on traditional DFG can be found in [1]

3.1.1. Integral-grid model

The first theorem is to show that by choosing the right unfolding factor f, an optimal schedule under integral-grid model, while assuming non-pipelined hardware implementation, can be obtained. Particularly, this f ensures that all possible schedule outcomes, based on such an unfolding factor, are optimal. First let us present some concepts related to this model.

Lemma 3.1. *Let c be the cycle period of a data-flow graph G and S be a legal schedule whose cycle period and unfolding factor are c and f respectively. A static schedule of S can be implemented under a non-pipelined design if and only if $c \geq \max_v t(v)$.*

The above lemma tell us that the condition $c \geq \max_v t(v)$ should be ensured in order to have a static schedule implementable. Based on this lemma, we can derived a formula to compute an optimal unfolding factor for a DFG considering the integral-grid model.

Lemma 3.2. *Let $\frac{\sigma}{\rho}$ be the irreducible form of the iteration bound of a DFG G, $B(G)$. Under the integral-grid model, the minimum rate-optimal unfolding factor f for creating a static schedule is that $f = \left\lceil \frac{\max_v t(v)}{\sigma} \right\rceil \rho$.*

According to these properties, we derive a formula to compute an unfolding factor which can guarantee to produce optimal schedule for each of all possible PDFG outcomes.

Theorem 3.1. *Under the integral-grid model, given a PDFG (time-invariant model) G, and probabilistic iteration bound $B_p(G) = \{(\frac{\sigma_i}{\rho_i}, p_i)\}$ where $\frac{\sigma_i}{\rho_i}$ is the irreducible form of the iteration bound $B(G')$ of the graph outcome G' of the PDFG, let L^f_{opt} be the optimal schedule length with $f = \text{lcm}_{\forall i} \left\{ \left\lceil \frac{\max_v t(v)}{\sigma_i} \right\rceil \rho_i \right\}$. Then $\frac{1}{f} \cdot L^f_{opt}(G) = B_p(G)$.*

Proof. Since both side of the equation $\frac{1}{f} \cdot L^f_{opt} = B(G)$ are random variables, we first need to show that there exists the same number of outcomes in the sample description space $\mathfrak{S}$ for both sides. Without loss of generality, let us assume that, for all $v \in V$, $T(v)$ has a distribution of the countable discrete type, i.e., there is a finite number in a sample description space.

According to Definition 3.1, we know that for a PDFG G there exists a finite number of G', say n, each of which has its corresponding probability $p = \prod_v \text{Prob}(T(v) = x)$. Furthermore, we want to show that for any graph outcome G', the equation

$\frac{1}{f} \cdot l^{f}_{opt}(G') = B(G')$ is preserved. Since each G' is a non-probabilistic DFG, the iteration period is rate-optimal if $\frac{c}{f} = B(G')$. We have $l^{f}_{opt}(G') = f \cdot B(G')$ by Definition 2.1. If unfolding factor $f = \left\lceil \frac{\max_v t(v)}{\sigma} \right\rceil \cdot \rho$ is chosen, a schedule $S(G')$ will have an optimal schedule length.

Let $f_{lcm} = lcm_{\forall i} \left\{ \left\lceil \frac{\max_v t(v)}{\sigma_i} \right\rceil \rho_i \right\}$ be the least common multiple of all unfolding factors computed for all outcome graphs G'. Using f_{lcm} in G' results in having its optimal schedule length in a form of $k \cdot l^{f_{lcm}}_{opt}$ where k is the constant from dividing f_{lcm} by the minimum unfolding factor f of G'. Now the random variable $L^{f_{lcm}}_{opt}$ is in a form $\{(k_0 \cdot l_0, p_0), \ldots, (k_n \cdot l_n, p_n)\}$. If we multiply this random variable with $\frac{1}{f_{lcm}}$, we get $B_p(G)$. □

As an example, reconsider the graph in Figure 2. We know that the irreducible form of $B_p(G)$ is equal to $\left\{ \left(\frac{3}{1}, 0.2\right), \left(\frac{4}{1}, 0.8\right) \right\}$. According to Theorem 3.1, the desired unfolding factor should be the least common multiple of $\left\{ \left\lceil \frac{\max_v t(v)_0}{\sigma_0} \right\rceil \rho_0, \left\lceil \frac{\max_v t(v)_1}{\sigma_1} \right\rceil \rho_1 \right\}$. Note here that the $\max_v t(v)$ of each of possible graph outcomes are 5 and 7 respectively. By calculating this, we have the unfolding factor f equal to 2. Both schedules for each of the graph outcomes should also be optimal that is the cycle period of outcome 0 will be 6 and 8 for the outcome 1.

3.1.2. Fractional-grid model

In both previous results, we used G' to represent a graph outcome of the PDFG. Since we can assume that there exists a finite number of the outcomes, a notation G'_i denotes the i^{th} copy of the PDFG. The following result shows that if $f = lcm_{\forall i} \left\{ \left\lceil \frac{\max_v t(v)}{B(G'_i)} \right\rceil \right\}$, then the probabilistic optimal schedule length can be achieved. The following lemma discusses how to choose an unfolding factor for a non-probabilistic data-flow graph.

Lemma 3.3. *Under a fractional-grid model, the minimum unfolding factor* f *for creating an optimal schedule from a data-flow graph* G *is that* $f = \left\lceil \frac{\max_v t(v)}{B(G)} \right\rceil$.

Theorem 3.2. *Under the fractional-grid model, given a PDFG (time-invariant model)* G, *and probabilistic iteration bound* $B_p(G)$ *represented by* $\{(B(G'_i), p_i)\}$ *where* $B(G')$ *is the iteration bound of a graph outcome* G' *of the PDFG, let* $f = lcm_{\forall i} \left\{ \left\lceil \frac{\max_v t(v)}{B(G'_i)} \right\rceil \right\}$. *Then* $\frac{1}{f} \cdot L^{f}_{opt} = B_p(G)$.

Proof. As in Theorem 3.1, we know that there exists a finite number of graph outcomes G'. From Lemma 3.3, we obtain $\frac{1}{f} \cdot l^{f}_{opt}(G') = B(G')$ where f is chosen to be the result from ratio $\left\lceil \frac{\max_v t(v)}{B(G'_i)} \right\rceil$. Consider when selecting $f_{lcm} = lcm_{\forall i} \left\{ \left\lceil \frac{\max_v t(v)}{B(G'_i)} \right\rceil \right\}$. Each of the optimal schedule length $l_{opt}(G'_i)$ with respect to this unfolding factor f_{lcm} can be presented in the following form: $\{(k_0 \cdot l_0, p_0), \ldots, (k_n \cdot l_n, p_n)\}$ where k_i is a constant from dividing f_{lcm} by its original $f = \left\lceil \frac{\max_v t(v)}{B(G'_i)} \right\rceil$, l_i is the resulting optimal schedule length and p_i is the corresponding probability. Therefore, the equation $\frac{1}{f} \cdot L^{f}_{opt} = B_p(G)$ is satisfied.. □

If the two outcomes from Figure 2 are studied, the unfolding factor under the fractional-grid model will also be two. This is because the irreducible forms from the integral one are not fractional numbers, i.e., their denominators are both one. In practice, the irreducible form of an iteration bound may not be integer. Thus the resulting unfolding factor from Theorem 3.2 which adopts the fractional-grid model, will produce a smaller number.

3.2. Good estimation for time-variant model

Since we cannot guarantee that a given unfolding factor f will help us derive the optimal schedule (see example of Definition 3.3), a good estimation of the unfolding factor is preferable. There are several criteria to decide if a given f is good enough. The simplest criteria will be the comparison of the average iteration period. This paper suggests that using a formula presented in Lemma 3.3 to compute f for each of the original PDFG outcomes and select the maximum factor to be the candidate should give a reasonable estimation.

4. CONCLUSION

The probabilistic data-flow graph (PDFG) model can be used to represent an application whose computation times are associated with some probability such as ones from conditional statements. Two timing models, the time-invariant and time-variant models, have been presented to differentiate the PDFGs according to the nature of their original applications. For graphs under the time-invariant model, we presented two theorems, one for the integral-grid model and the other for the fractional-grid model, which tell us what unfolding factors guarantee that the resulting static schedules of the PDFG will be optimal. For graphs under time-variant model, it has been shown that the definition of the probabilistic iteration bound is not a lower bound for these graphs. Therefore, we proposed the means for estimating an unfolding factor for them.

5. ACKNOWLEDGMENT

This work is partially supported by NSF MIP95-01006, NSF MIP97-04276 and Schmitt Scholarship.

6. REFERENCES

[1] L.-F. Chao and E. H.-M. Sha. Static scheduling for synthesis of DSP algorithms on various models. *Journal of VLSI Signal Processing*, 10:207–223, 1995.

[2] A. A. Khan, C. L. McCreary, and M. S. Jones. A comparison of multiprocessor scheduling heuristics. In *Proceedings of the 1994 International Conference on Parallel Processing*, volume II, pages 243–250, 1994.

[3] M. Lam. Software pipelining. In *Proceedings of the ACM SIGPLAN'88 Conference on Programming Language Design and Implementation*, pages 318–328, Atlanta, GA, June 1988.

[4] W. Li and K. Pingali. A singular loop transformation framework based on non-singular matrices. Technical Report TR 92-1294, Cornell University, Ithaca, NY, July 1992.

[5] K. K. Parhi and D. G. Messerschmitt. Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding. In *Transactions on Computers*, volume 40, pages 178–195. IEEE, February 1991.

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

[7] M. E. Wolfe. *High Performance Compilers for Parallel Computing*, chapter 9. Addison-Wesley, Redwood City, CA, 1996.