MEMORY/TIME OPTIMIZATION OF 2-D FILTERS

Nelson Luiz Passos and Edwin Hsing-Mean Sha

Dept. of Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

Two-dimensional filters are commonly used in digital image processing applications. These filters have the characteristic of processing recursive sets of instructions requiring high computational speed. In this paper, these sets are modeled as cyclic two-dimensional data flow graphs, which are also used to represent the equivalent circuit design. In this new method, such graphs are submitted to a multi-dimensional retiming in order to reduce their cycle time. Such a reduction can achieve a cycle equal to the longest atomic operation in the filter, by inserting a fixed number of registers, independent of the size of the problem, into the circuit paths. Examples, description and the correctness of our algorithm are presented in the paper.

1. INTRODUCTION

Digital image signal processing applications such as high definition television (HDTV) and medical imaging devices are known to require high computing power. Such computation intensive applications usually depend on time critical sections, consisting of loops of sequence of operations also called iterations. An important characteristic of such applications is the multi-dimensionality of the data submitted as input to multi-dimensional (MD) filters. The design of application-specific circuitry solutions for such critical sections is required to improve the overall computing performance. A commonly used way to optimize such designs is the retiming operation. This paper presents a novel algorithm based on MD retiming transformations to obtain a circuit design for any given cycle time greater than the maximum propagation delay among its atomic operations, while considering the consequences in the memory requirements.

Retiming was initially proposed by Leiserson-Saxe [4] focusing on one-dimensional (1-D) problems. By using this approach the original multi-dimensionality of the problem must be initially translated into an 1-D environment. As a consequence of this translation, the achievable performance improvement may be restricted by the number of delays existing in a cycle. Most of the research in this area has followed the 1-D approach and consequently is subject to the same constraints [5, 7]. In this paper, we use the concept of an MD retiming, introduced in [1, 6], to model the placement of registers along the circuit data paths, and to restructure the initial memory elements. Differently from 1-D situation, the MD method is not constrained by the number of existing registers in a cycle. A multi-dimensional data flow graph (MDFG) is used to represent the problem, and is optimized by our transformation.

This work was supported in part by ORAU Faculty Enhancement Award under Grant No. 42265, and by the William D. Mensch, Jr. Fellowship.

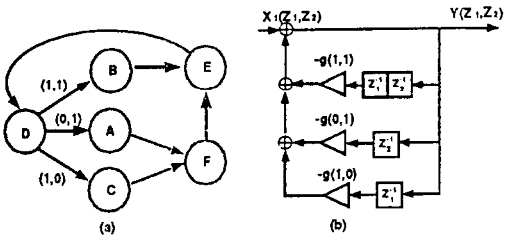


Figure 1: (a) MDFG for a simple filter (b) sketch of equivalent digital circuit

In a 2-D problem the two dimensions are generically referred as x and y. For best understanding, these denominations are interchanged with row and column references when appropriate. Let's examine a simple example, representing a filter with transfer function

$$H(z_1,z_2) = \frac{1}{\left(1-\sum_{n_1=0}^{1}\sum_{n_2=0}^{1} g(n_1,n_2)*z_1^{-n_1}*z_2^{-n_2}\right)}$$

for $n_1, n_2 \neq 0$. Figure 1(a) presents the MDFG representing the problem, while figure 1(b) shows the equivalent digital circuit. The 2-D delay $(1,1)$ is represented by a FIFO structure translated into a serial implementation of z_1^{-1} and z_2^{-1} elements. The delay $(0,1)$ is represented by z_2^{-1} and the delay $(1,0)$ by z_1^{-1}. The current cycle time for this design is equivalent to the sequential execution of three additions and one multiplication. We use a *schedule vector s* to indicate the execu-

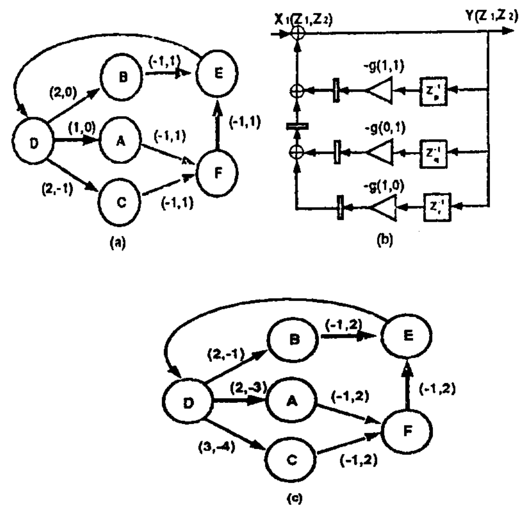


Figure 2: (a) retimed MDFG (b) optimized circuit design(c) retimed MDFG, with possible schedule vector (5,3)

tion sequence. For a row-wise computation, $s = (0,1)$, the z_1^{-1} element represents only one delay, which implies that the 1-D retiming technique can not reduce the initial cycle time due to the constancy on the number of delays in the cycle $C \rightarrow F \rightarrow E \rightarrow D \rightarrow C$. Using our technique on the MDFG representing the circuit, we can ignore such a constraint. If we assume the execution time of one multiplication equal to two times the execution of one addition, we can produce an optimized design with cycle time equivalent to the execution of one multiplication. In this situation, the final solution shown in figures 2(a) and (b). The queues have indices p, q, r to indicate a new queue element if compared to the initial design. All other edges that were modified received one register only.

It is important to the designer to keep the number of inserted registers as small as possible. When there is no technique involved in the optimization process, one can select such a MD retiming function and sequence of execution that can introduce large queues depending on the problem size. This situation can be illustrated by choosing the valid schedule vector $s = (5,3)$ and the valid retiming function $r(B) = r(F) = (-1,2)$ and $r(A) = r(C) = (-2,4)$ for the MDFG shown in figure 1. The resulting dependencies would be those shown in figure 2(c). This design would require 7 queues (one in each non-zero delay path). The solution obtained through our algorithm and shown in figure 2(b) has only 3 queues and four registers. To manage the selection criteria for the retiming function in an appropriate way, we explore some properties of the hyperplanes associated with the schedule vector, in such a way that all new memory elements to be inserted in the circuit are not dependent on the problem size.

2. BASIC PRINCIPLES

A *multi-dimensional data flow graph (MDFG)* $G = (V, E, d, t)$ is a node-weighted and edge-weighted directed graph, where V is the set of computation nodes, i.e., the functional elements in the circuit design, E is the set of dependence edges, equivalent to the circuit data paths, d is a function representing the MD delay between two nodes, implicitly indicating the storage elements required in the circuit design, and t is a function representing the computation time of each node. In figure 1(a), $V = \{A, B, C, D, E, F\}$ and $E = \{e1 : (A,F), e2 : (B,E), e3 : (C,F), e4 : (D,A), e5 : (D,B), e6 : (D,C), e7 : (E,D), e8 : (F,E)\}$ where, $d(e1) = d(e2) = d(e3) = d(e7) = d(e8) = (0,0)$, $d(e4) = (0,1)$, $d(e5) = (1,1)$, and $d(e6) = (1,0)$. The execution time of each operation is assumed to be $t(A) = t(B) = t(C) = 2$, and $t(D) = t(E) = t(F) = 1$.

An *iteration* is the execution of each node in V exactly once. An *iteration space* is the region in the MD discrete cartesian space whose points correspond one-to-one to the iteration indices. An edge with delay $(0,0)$ represents a data dependence within the same iteration. A legal MDFG must have no zero-delay cycle, i.e., the summation of the delay vectors along any cycle can not be $(0,0)$.

Vectors are used to indicate the sequence of computation. A *schedule vector* s is the normal vector for a set of parallel hyperplanes that define the global sequence of execution of the iteration space. Nodes in a same hyperplane will be executed sequentially, according a second level of schedule associated with the hyperplanes. We say that an MDFG $G = (V, E, d, t)$ is *realizable* if there exists a schedule vector s for G, i.e., $s \cdot d \geq 0$ for any $d \in G$ [3] and it has no zero-delay cycle. In a 2-D iteration space, we say that the schedule vector is *orthogonal* if it is parallel to one of the axis representing the row and column directions. Otherwise, the schedule vector is said to be *non-orthogonal*, and requires a non-orthogonal execution sequence.

A row-wise execution is equivalent to a schedule vector $(0,1)$ and it implies that delays $(1,0)$ must be translated into single register delay elements. Delays of the form $(d.x, 0)$ produce queues of size $d.x$, which is independent of the number of points in the x- and y-directions (the problem size). If M is the number of points in the x-direction, the delays of the form $(0, d.y)$ are equivalent to queues of size $d.y \times M$. For the general case, this size depends on the distance between the hyperplanes determined by the schedule vector.

Using such concepts, commonly applied to the design of *array processors* [3], we generalize the computation of the distance between hyperplanes for each dependence vector d as $\Delta h = s \cdot d$ where s is the schedule vector. When $\Delta h = 0$, then d indicates either a zero delay or a delay vector perpendicular to s. In the first case, d represents a dependence in the same iteration which is equivalent to a direct data path in the circuit design. In the second case, d indicates a dependence between iterations in the same hyperplane, which is equivalent to a fixed-size queue.

The queue size required by delay vectors not perpendicular to the schedule vector is bounded by $\Delta h \times \max(\textit{number of integer points on a hyperplane})$. The regularity of the iteration space produces variable number of points in the hyperplanes when the schedule vector is non-orthogonal. For example, for a schedule vector $(s.x, s.y)$ with $s.x > 0$ and $s.y > 0$, and an iteration space $M \times N$, the number of points p in a hyperplane would be $1 \leq p \leq \min(\lfloor \frac{M}{s.y} \rfloor, \lfloor \frac{N}{s.x} \rfloor) + 1$.

3. THE MD RETIMING FUNCTION

A *MD retiming* r is a function from V to Z^n that redistributes the nodes in the original iteration space produced by the replication of an MDFG G. A new MDFG G_r is created, such that each iteration still has one execution of each node in G. This transformation is equivalent to a redistribution of the delay elements in a circuit. The retiming vector $r(u)$ of a node $u \in G$ represents the offset between the original iteration containing u, and the one after retiming. The delay vectors change accordingly to preserve dependencies, i.e., $r(u)$ represents delay components pushed into the edges $u \rightarrow v$, and subtracted from the edges $w \rightarrow u$, where $u, v, w \in G$. Therefore, we have $d_r(e) = d(e) + r(u) - r(v)$ for every edge $u \xrightarrow{e} v$ and $d_r(l) = d(l)$ for every cycle $l \in G$.

After identifying the different aspects of relating dependence vectors to memory elements according to some schedule vector, we must be sure to obtain a legal MD retiming function. We say that an MD retiming r is *legal* if given a realizable MDFG $G = (V, E, d, t)$, r transforms G into G_r such that G_r is still realizable.

To find a legal MD retiming is more complex than the use of traditional retiming on 1-D cases. In those cases, the positive number of delays after retiming guarantee that the retiming is legal. However, for the MD problem, positive delay vectors are too restrictive since an MDFG is still realizable even if it has negative delays. We know that the existence of a linear schedule vector for the execution of the MDFG is the necessary and sufficient condition for its realizability [3]. Therefore, the following theorem provides the theoretical foundations to support the selection of the legal MD retiming function.

Theorem 3.1 *Given an MDFG $G = (V, E, d, t)$, a schedule vector s for G such that $s \cdot d(e) > 0$ for any $d(e) \neq 0$, a node $u \in V$ which has all incoming edges with non-zero delay, and an integer constant $k \geq 1$, a legal MD retiming for G is any vector $k \times r(u)$ where $r(u)$ is orthogonal to s.*

The application of such a retiming function transforms the new MDFG in a graph with a new set of dependence vectors. The equivalent circuit design is then constructed through a translation process where the graph nodes are the circuit elements and the delays are translated into queues (FIFO elements), according to the chosen schedule vector.

Considering the concepts seen above, in order to find a legal MD retiming, we begin by solving the inequalities $s \cdot d(e) > 0$ for every $e \in E$, where s is the unknown. We choose a retiming function from the hyperplane with s as the normal vector. The selected retiming function when applied to any node that has all incoming edges with non-zero delays and at least one outgoing edge with zero delay is a legal MD retiming according to theorem 3.1. We now present an algorithm for transforming the MDFG and, consequently, its equivalent circuit in such a way to produce the desired cycle time. Our algorithm CyclOp (Cycle time Optimization), inputs the MDFG representing the circuit and the desired cycle time δ as shown below:

```
Algorithm CyclOp($G = (V, E, d, t), \delta$)
 Choose $s = (s_x, s_y)$ such that $s \cdot d(e) > 0$ for any $e \in E$
 $r_s \leftarrow (-s_y, s_x)$
 $count \leftarrow 0$
 while $V \neq \phi$ {
  if OUTDEGREE($u$) = 0, $u \in V$
   PUT($u, Queue$); REMOVE($u, V$)
  while $Queue \neq \phi$ {
   GET($u, Queue$); LEVEL($u$) $\leftarrow count$
   for every PRED($u$) {
    OUTDEGREE(PRED($u$)) $\leftarrow$ OUTDEGREE(PRED($u$)) $-1$
    if $t(u)$ +$t$(PRED($u$)) $\leq \delta$ AND OUTDEGREE(PRED($u$)) = 0
     PUT(PRED($u$),$Queue$)
     $t$(PRED($u$)) $\leftarrow t(u)$ +$t$(PRED($u$)) } }
  count++ }
 for every $u \in G$
  Compute $r(u) \leftarrow$ LEVEL($u$) $\times r_s$
end
```

The correctness of the algorithm is shown below:

Theorem 3.2 *Given a realizable MDFG $G = (V, E, d, t)$, equivalent to a circuit design C, and a target cycle time $\delta \geq t(u)$ for any $u \in V$. The algorithm CyclOp transforms G to G_r in $O(|E|)$ time, s. t. the cycle time of G_r is less than or equal to δ and the inserted delays are equivalent to a finite number of registers, independent of the problem size.*

4. EXAMPLE

In this section we present the application of our method to the IIR section of a 2-D filter design presented in [2]. It consists of a filter represented by the transfer function:

$$H(z_1, z_2) = \frac{\sum_{i=0}^{2} \sum_{j=0}^{2} f(i,j) * z_1^{-i} * z_2^{-j}}{\sum_{i=0}^{2} \sum_{j=0}^{2} g(i,j) * z_1^{-i} * z_2^{-j}}$$

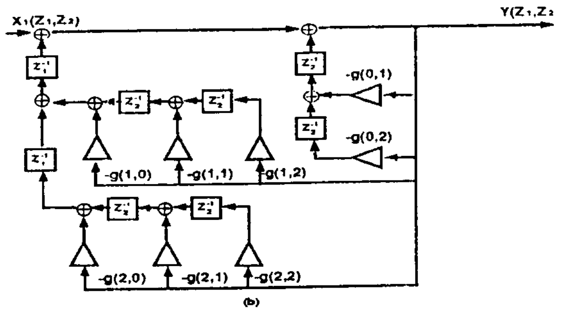


Figure 3: Example of an IIR section of a two-dimensional filter

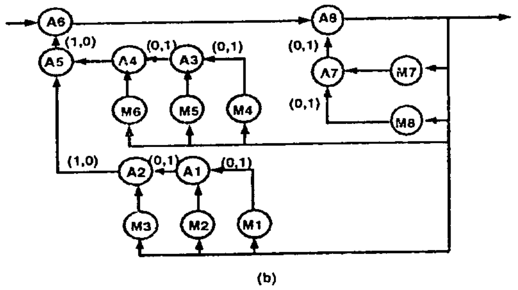
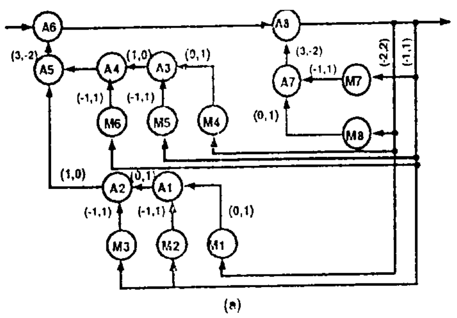


Figure 4: MDFG of the example filter

Only the IIR section of the filter is used in this example, since the solution for the FIR section is considered trivial. The circuit design is shown in figure 3. We represent it by its equivalent MDFG shown in figure 4. Assume adders take one time unit to execute and multipliers take two. Let us also assume that the desired cycle time is two time units. Using algorithm *CyclOp*, we begin by finding a possible schedule vector. The selected schedule is $s = (1,1)$. We proceed by choosing the MD retiming function, obtaining $r = (-1,1)$. Nodes labeled $A1$, $A2$, $A3$, $A4$, $A5$, $A7$, $M1$, $M4$, and $M8$ are assigned to level 0, nodes $M2$, $M3$, $M5$, $M6$ and $M7$ to level 1, and finally, $A6$ and $A8$ to level 2. In the next step, we compute the MD retiming function. The resulting retiming function is: $r(A6) = r(A8) = (-2,2)$, and $r(M2) = r(M3) = r(M5) = r(M6) = r(M7) = (-1,1)$. The final IIR section of the graph and the modified circuit design are shown in figure 5. The critical path was reduced to one multiplication (two time units) as desired.

5. CONCLUSION

A novel technique for optimizing a 2-D filter circuit design was presented. The circuit was represented by a MD data flow graph and it was submitted to an MD retiming method that we developed. Using a sequence of execution associated to hyperplanes as in a wavefront approach allows us to insert single registers in the circuit paths to optimize its cycle time. Common mistakes that could produce results requiring new large queues are avoided through the use of our algorithm. A pre-selected MD retiming function is the core of the algorithm. The algorithm is able to achieve any desired cycle time longer than or equal to the slowest operation in the circuit.



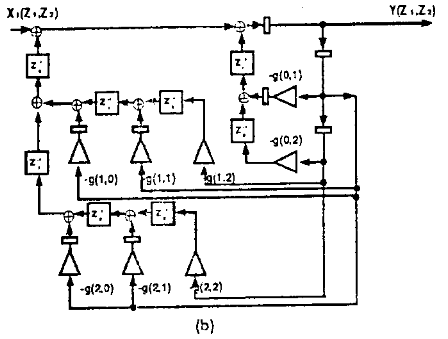


Figure 5: Retimed filter: (a) MDFG (b) circuit design

6. REFERENCES

[1] L.-F. Chao and E. H.-M. Sha, " Static Scheduling of Uniform Nested Loops," *Proceedings of 7th International Parallel Processing Symposium* , Newport Beach, CA, April, 1993, pp. 1421-1424.

[2] Gnanasekaran, R., " 2-D Filter Implementation for Real-Time Signal Processing". *IEEE Transactions on Circuits and Systems*, 1988, vol. 35, n. 5, pp. 587-590.

[3] S. Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[4] C. E. Leiserson and J. B. Saxe, " Retiming Synchronous Circuitry". *Algorithmica*, 6, 1991, pp. 5-35.

[5] T.-F. Lee, A. C.-H. Wu, D. D. Gajski, and Y.-L. Lin, " Performance Optimization of Pipelined Circuits". *Proc. of the International Conference on Computer Aided Design*, November, 1990, pp. 410-413.

[6] N. L. Passos, E. H.-M. Sha, and S. C. Bass, " Schedule-Based Multi-Dimensional Retiming". *Proceedings of 8th International Parallel Processing Symposium*, 1994, vol 4, pp. 195-199.

[7] C.-Y. Wang and K. K. Parhi, " High Level DSP Synthesis Using the MARS Design System". *Proc. of the International Symposium on Circuits and Systems*, 1992, pp. 164-167.