Loop Pipelining for Scheduling Multi-Dimensional Systems via Rotation

Nelson Luiz Passos Edwin Hsing-Mean Sha Steven C. Bass
Dept. of Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract– Multi-dimensional (MD) systems are widely used in scientific applications such as image processing, geophysical signal processing and fluid dynamics. Earlier scheduling methods in synthesizing MD systems do not explore loop pipelining across different dimensions. This paper explores the basic properties of MD loop pipelining and presents an algorithm, called *multi-dimensional rotation scheduling*, to find an efficient schedule based on the multi-dimensional retiming technique we developed. The description and the correctness of our algorithm are presented in the paper. The experiments show that our algorithm can achieve optimal results efficiently.

I. Introduction

Computation intensive applications usually depend on time-critical sections consisting of a loop of instructions. To optimize the execution rate of such applications, the designer needs to explore the parallelism embedded in repetitive patterns of a loop. However, the existence of resource constraints makes the problem of scheduling loops an NP-complete problem.

Previous research in loop pipelining for loops with cyclic dependencies produced methods which are only applicable to one-dimensional problems. Such methods appear in several systems [1, 5, 8]. This paper studies loop pipelining and designs an algorithm for scheduling multi-dimensional (MD) loops while considering resources constraints. MD retiming is used to characterize the effect of MD loop pipelining. Chao and Sha introduced the concept of a restricted MD retiming without resource constraints [2]. This concept was applicable to a specific class of multi-dimensional data flow graphs (MDFGs), where any cycle would have a strictly non-negative total MD delay. In a previous study we extended the concept of MD retiming, introducing the idea of *schedule-based multi-dimensional retiming* without resource constraints [7]. In that method, a *feasible linear schedule* allows us to restructure the loop body represented by a general form of MDFG, while preserving data dependencies, and improving the existent parallelism. Since there are more dimensions to search than the one-dimensional (1-D) case, the problem of finding the optimal retiming becomes more complex. This paper develops a method for scheduling cyclic MDFGs with resource constraints. We call this technique *multi-dimensional rotation scheduling*, generalizing the 1-D rotation method [1].

In data-path design, the length of one control-step is consequence of the clock cycle. Two operations that do not exceed the clock cycle may be chained to fit into the same clock cycle. Operations that are longer than one clock cycle require the allocation of resources for multiple control steps. Functional units that have a pipeline design can accept a new operation before the previous one is completed. The *MD rotation scheduling* method is designed to handle all of these types of operations. This paper introduces the solution for problems with operations that fit exactly into one clock cycle, leaving the other cases as an extension of the concepts here presented.

Our algorithm improves a resource-constrained schedule by performing MD retiming incrementally using the multi-dimensional rotation technique. The existing schedule is partially rescheduled to obtain a shorter schedule under resource constraint. The state of a sequence of rotations is recorded by a simple retiming function. In 1-D rotation, a rotation corresponds to incrementing the retiming function of a node by one. For the MD case, a rotation of a node can be regarded as retiming the node by any vector (even negative) as long as the resulting dependence graph does not contain cycles. This paper builds up the framework of the problem of multi-dimensional scheduling.

For simplicity we use a two-dimensional (2-D) problem, shown in Fig. 1(a), as an example, where nodes A, D are multipliers and B, C are adders. Fig. 2(a) shows an optimal schedule with length 4 for the *directed acyclic graph* (DAG) associated to the MDFG in Fig. 1(a). Fig. 2(b) shows a more compact schedule in which node D has been rotated down and then pushed to a new position. This new schedule can be associated with the retimed graph presented in Fig. 1(b), where the 2-D delay $(1,0)$ is pushed through node D of the original MDFG.[1] Intuitively, the

[1]Note that $(1,0)$ is arbitrarily chosen here. The vector $(2,1)$

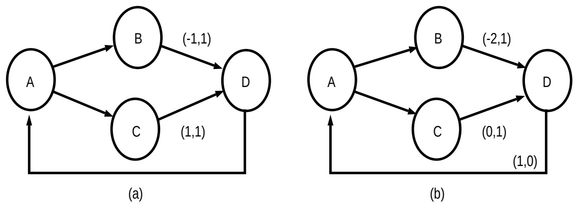
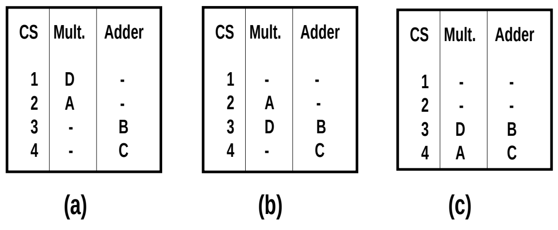


Fig. 1: (a) MDFG extracted from a Wave Digital Filter (b) MDFG after retimed by r(D)=(1,0)



(a)

CS	Mult.	Adder
1	D	-
2	A	-
3	-	B
4	-	C

(b)

CS	Mult.	Adder
1	-	-
2	A	-
3	D	B
4	-	C

(c)

CS	Mult.	Adder
1	-	-
2	-	-
3	D	B
4	A	C

Fig. 2: (a) Initial schedule (b) Schedule after rotating D (c) Final schedule

retimed node no longer precedes the node A within the same iteration, resulting in an shorter schedule with length 3. An optimal schedule with length 2, seen in Fig. 2(c), results after the rotation of node A.

The multi-dimensional rotation scheduling method uses a DAG scheduling algorithm, such as *list scheduling*[3] as a subroutine. Notice that only part of the MDFG is rescheduled in each rotation, saving computation time. The next section establishes some of the basic concepts to be used in this paper. Section III defines down-rotation and introduces the basic fundamentals of the multi-dimensional rotation. Also, a heuristic method to solve the scheduling problem using our technique is presented. An example is discussed in section IV. A final section summarizes the concepts presented.

II. Basic Principles

A *multi-dimensional data flow graph* $G = (V, E, d, t)$ is a node-weighted and edge-weighted directed graph, where V is the set of computation nodes, E is the set of dependence edges, d the MD delay between two nodes, and t the computation time of each node. We use $d(e) = (d.x, d.y)$ as a general formulation of any delay shown in a two-dimensional DFG (2DFG).

An *iteration* is the execution of each node in V exactly once. Iterations are identified by a vector i, equivalent to an MD index. An iteration is associated to a static schedule. The static schedule of a loop is repeatedly executed

is also possible, as are many others. This flexibility introduces a high computational complexity in multi-dimensional retiming!

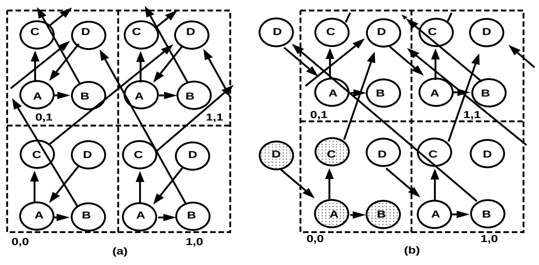


Fig. 3: (a) DG based on the replication of an MDFG, showing iterations starting at (0,0). (b) Retimed DG

for the loop. A static schedule must obey the precedence relations defined by the subgraph of an MDFG, consisting of edges without delays. For any iteration j, an edge e from u to v with delay vector $d(e)$ means that the computation of node v at iteration j depends on the execution of node u at iteration $j - d(e)$. An edge with delay $(0, 0, \ldots, 0)$ represents a data dependence within the same iteration (computational cell). A legal MDFG must have no zero-delay cycle.

An equivalent *cell dependence graph* G_{dg} of an MDFG G is a 2-D dependence graph (2DG) showing the dependencies between infinite copies of nodes representing the MDFG. Fig. 3(a) shows the replication of the MDFG from Fig. 1(a). We say that an MDFG G is *realizable* if its cell dependence graph does not contain any cycle.

To manipulate MDFG characteristics represented on vector notation, such as the delay vectors, we make use of component-wise vector operations. For the two-dimensional vectors P and Q, represented by their coordinates $(P.x, P.y)$ and $(Q.x, Q.y)$, an example of arithmetic operation is $P + Q = (P.x + Q.x, P.y + Q.y)$. The notation $P \cdot Q$ indicates the inner product between P and Q, i.e., $P \cdot Q = P.x * Q.x + P.y * Q.y$.

A. Retiming a Multi-Dimensional Data Flow Graph

The period during which all computation nodes in an iteration are executed, according to existing data dependencies and without resource constraints, is called a *cycle period.* The *cycle period* $C(G)$ of an MDFG G is the maximum computational time among paths that have no delay. For example, the MDFG in Fig. 1(a) has $C(G) = 3$, which can be measured through the paths $p = D \rightarrow A \rightarrow B$ or $p = D \rightarrow A \rightarrow C$. The cycle period for an MDFG is the length of the static schedule for the corresponding DAG without resource constraints.

A *multi-dimensional retiming* r is a function from V to Z^n that redistributes the nodes in the cell dependence graph. A new MDFG $G_r = (V, E, d_r, t)$ is created, such that each iteration still has one execution of each node in G. The retiming vector $r(u)$ of a node $u \in G$ represents the offset between the original iteration containing u, and

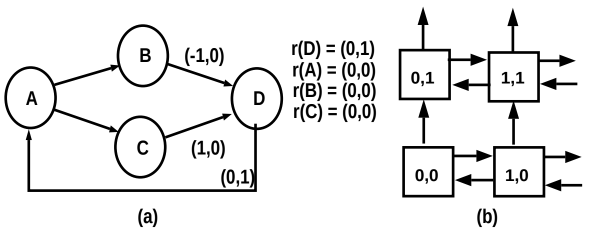


Fig. 4: (a) Example of illegal MD retiming. (b) DG showing cycles in the x-direction.

the one after retiming.

On the 1-D retiming, negative delays represent the existence of a cycle in the equivalent DG, and can not be allowed [6]. However, on the MD case, the existence of negative delays is to be considered natural if there exists a schedule s that turns the DG realizable. The retimed cell DG, for the example in Fig. 1(b), is shown in Fig. 3(b), where the nodes originally belonging to iteration $(0,0)$ are marked. A possible schedule vector for the retimed graph is $s = (1,3)$. Fig. 4(a) shows an illegal retiming function applied to the same example. By simple inspection of the cell dependence graph in Fig. 4(b) we notice the existence of a cycle created by the dependencies $(1,0)$ and $(-1,0)$.

A *prologue* is the set of instructions that are moved on directions x and y, in a 2-D retiming, and that must be executed to provide the initial data for the iterative process.

B. Schedule-Based Multi-Dimensional Retiming

A legal MD retiming on an MDFG $G = (V,E,d,t)$ requires that the cell dependence graph of the retimed MDFG $G_r = (V,E,d_r,t)$ does not contain any cycle, and for a given cycle period c, if the execution time of a path p in G_r is greater than c, then $d_r(p) \neq (0,0,\ldots,0)$. These two constraints are enforced through the use of a feasible linear schedule vector that supports the realization of the retimed graph.

A *schedule-based multi-dimensional retiming* r of an MDFG $G = (V,E,d,t)$, resulting $G_r = (V,E,d_r,t)$, which is realizable according to a schedule s, is characterized by:
(a) for any path $u \overset{p}{\rightsquigarrow} v$, $d_r(p) = d(p) + r(u) - r(v)$
(b) for any cycle $l \in G$, $d_r(l) = d(l)$
(c) for any edge e, if $d_r(e) \neq (0,0,\ldots,0)$, $d_r(e)\cdot s > 0$.

We define the functions $D^{(s)}(u,v)$ and $T^{(s)}(u,v)$ as:
$D^{(s)}(u,v) = min\{d(p)\cdot s | u \overset{p}{\rightsquigarrow} v \in G\}$
$T^{(s)}(u,v) = max\{t(p) | u \overset{p}{\rightsquigarrow} v \in G \wedge d(p)\cdot s = D^{(s)}(u,v)\}$.
The function r is a legal schedule-based MD retiming on G, such that $C(G_r) \leq c$ if and only if
(a) for every edge $u \overset{e}{\longrightarrow} v$, $r(v)\cdot s - r(u)\cdot s < d(e)\cdot s$ or $r(v) - r(u) = d(e)$
(b) for every pair u,v in G, if $T^{(s)}(u,v) > c$ then $r(v)\cdot s - r(u)\cdot s \leq D^{(s)}(u,v) - 1$

The concepts shown above, are the basis for an ILP program to find the schedule-based multi-dimensional retiming function. Due to the space limit, details are omitted. Interested readers refer to [7].

III. Multi-Dimensional Rotation Scheduling

A. Basic Concepts

In this section we introduce the theoretical foundations for our algorithm. The MD down-rotation operation is defined and main properties are presented. Spatial properties involving the dependence vectors are explored to support the multi-dimensional retiming used in the rotation operation.

Definition III..1 Given an MDFG $G = (V,E,d,t)$ and X a subset of V, the *MD down-rotation* of set X pushes a multi-dimensional delay vector from each of the incoming edges of X to each of the outgoing edges of X, resulting in a transformed MDFG G_X.

The following property is obtained by the definition of MD down-rotation.

Property III..1 *Let $G = (V,E,d,t)$ be a legal MDFG. If X is a subset of V, then the set X is down-rotatable if and only if every path $(V - X) \rightarrow X$ contains a delay different from $(0,0,\ldots,0)$.*

Fig. 1(a) shows the node D as a source node for the DAG associated to the MDFG presented. Submitting D to a down-rotation, it changes from source to a sink node in the new MDFG, shown in Fig. 1(b). We say that X is *down-rotatable* if there exists a legal retiming function that transforms G to G_X. The selection of the retiming function has a relevant importance in the process. An arbitrary choice of the retiming vector may induce a cycle in the rotated graph. As an example, let's examine the problem shown in Fig. 1(a). Assume that we arbitrarily decide to use a retiming vector $(0,1)$, after the down rotation of node D, and pushing it up to control step 3, we would apparently reduce the length of the schedule to 3. However, we have seen before, in Fig. 4, that such a retiming generates an MDFG that has a cycle in the equivalent cell dependence graph, which implies that this solution is not realizable. We define a region that contains all dependence vectors to support the idea of a realizable MDFG.

Definition III..2 Given an MDFG $G = (V,E,d,t)$ and $D = \{d(e) : e \in E, d(e) \neq (0,0,\ldots,0)\}$, a *dependent region* is the subspace containing all dependence (delay) vectors in D. A *normal vector* n_D is the vector normal to a hyperplane that divides the space in two regions, with the normal vector n_D pointing to the dependent region.

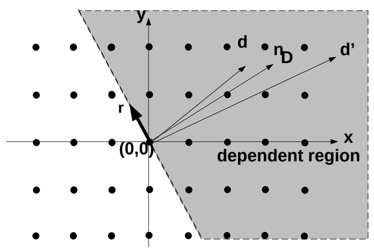


Fig. 5: An example of dependent region

In a two-dimensional case, the dependent region is equivalent to a half-plane. Fig. 5 shows a two-dimensional example of a dependent region. The normal vector has the following property:

Property III..2 *Let $G = (V, E, d, t)$ be a realizable MDFG. There exists a normal vector n_D such that the inner product of any non-zero dependence vector d and n_D is strictly positive, i.e, $d \cdot n_D > 0$.*

Considering such properties, we introduce the method of predicting a legal MD retiming function, which is used in our rotation scheduling algorithm.

Theorem III..1 *Let $G = (V, E, d, t)$ be a realizable MDFG, and n_D be a normal vector for G. A legal retiming $r(u)$ is any vector perpendicular to n_D, where all the incoming edges of u have non-zero delays.*

Therefore, given a set of dependence vectors, after obtaining a dividing hyperplane, we can predict a legal retiming function.

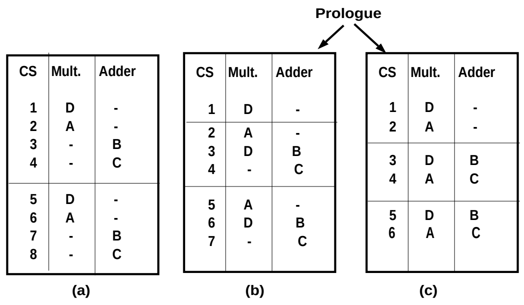
B. The Rotation Algorithm

We now present a technique to compact the initial schedule, considering the resource constraints, and using the MD down-rotation operation.

A schedule h is the mapping from V to control steps $[a, b]$, where $[a, b] = \{i : a \le i \le b, i \in N\}$, such that the resource constraints are satisfied. The computation of node $u \in V$ starts its execution at control step $h(u)$ in the schedule h. Considering the data dependencies within an iteration, we derive the following property to characterize a legal static schedule and associate it to a retiming function.

Property III..3 *Let $G = (V, E, d, t)$ be an MDFG, and h a proposed schedule for G under resource constraints. The schedule h is a legal static schedule of G if and only if there exists a legal retiming function r that transforms G to $G_r = (V, E, d_r, t)$, such that $h(u) + t(u) \le h(v)$, whenever $d_r(u, v) = (0, 0, \ldots, 0)$, for $u, v \in V$.*

We say that the retiming r *realizes* the schedule h when it satisfies this property.



(a)

CS	Mult.	Adder
1	D	-
2	A	-
3	-	B
4	-	C
5	D	-
6	A	-
7	-	B
8	-	C

(b)

CS	Mult.	Adder
1	D	-
2	A	-
3	D	B
4	-	C
5	A	-
6	D	B
7	-	C

(c)

CS	Mult.	Adder
1	D	-
2	A	-
3	D	B
4	A	C
5	D	B
6	A	C

Fig. 6: Sequential iterations for: (a) initial schedule (b) schedule after rotating D (c) final schedule

Our algorithm uses the technique of rotation to compact an initial schedule, obtaining a legal static schedule. Consider the example in Fig. 1(a), the initial schedule has a length 4, see Fig. 6(a). We construct the subset $X = \{D\}$ to be rotated down. Then we try to push node D to its earliest control step, which is control step 3, according to the precedence and resource constraints. We obtain the schedule in Fig. 6(b) with length 3. In another rotation we reschedule the set $X = \{A\}$. The optimal schedule is then obtained with a length 2, as Fig. 6(c) shows. Intuitively, when a node is rotated, each copy of the node is pushed up by $r.x$ iterations in the x-direction and $r.y$ iterations in the y-direction, for a retiming function $r = (r.x, r.y)$. A prologue is created to provide initial values for the repetitive iterations. Such prologue consists of those nodes that have been rotated.

It is easy to verify that, for unit-time operations, the multi-dimensional rotation will never increase the length of the initial schedule. Assume X_i to be the set of nodes scheduled in the first i control steps. From property III..1, we know that X_i is a down-rotatable set, for every i in $[1, k]$ where k is the length of the original schedule. After the rotation, we obtain a new valid schedule h' with the same length k in the interval $[i+1, i+k]$, i.e.,

$$h'(u) = \begin{cases} h(u) + k, & 0 < h(u) \le i \\ h(u), & i < h(u) \le k \end{cases}$$

Therefore, after any down rotation of unit-time nodes, there always exist a schedule for the retimed MDFG, which is at least as short as the initial one. A reschedule of the nodes in the last i control steps on the new MDFG will push those nodes to earlier control steps, and eventually produce a shorter schedule[2]. We can summarize our algorithm in the following steps:

1. Select the MD retiming function.
2. Compute an initial schedule for G and assume it is minimum.
3. Rotate down the set of nodes X consisting of nodes

[2]When handling multi-cycle operations, the whole node is rotated down, which implies that the post-rotation schedule may be longer than the initial one. To overcome this problem, the multi-cycle node may require the usage of some splitting technique as the *wrapping* discussed in [1]

found in the first k control steps (where k is an user option), retiming G accordingly.

4. Reschedule the nodes in X.
5. If the new schedule is shorter than the one saved as minimum, replace the minimum one.
6. If initial requirements were satisfied, then outputs the minimum schedule found until now, otherwise, replace the initial schedule with the retimed one and go back to step 3.

More complex problems may require the use of the multi-dimensional schedule-based retiming to find an initial reduced schedule.

C. The Selection of the Retiming Function

Differently from the 1-D rotation, where the down-rotation operation on set X was associated with the retiming of X by one time unit, the MD case allows us to retime in any *legal* direction. We know that larger components in the MD retiming vector imply larger prologues, so intuitively we must try to keep the absolute values of the components of the retiming vector as low as possible. The selection of the retiming function is done at the beginning of the algorithm. Two approaches are presented below.

Arbitrary Selection

In this case, we do not consider the problem being scheduled, but the advantages of having a specific retiming function. Note that in the one-dimensional case, the unit value was the most appropriate retiming for keeping the size of the prologue as shorter as possible. For a two-dimensional problem, the retiming vector could be chosen to be $(1,2)$, $(1,1)$, $(1,0)$ or one of many other options. To reduce the size of the prologue, we assume that the retiming function is parallel to one of the Cartesian axis. For a 2DFG, we choose $(0,1)$ or $(1,0)$. To assure the realizability of the final schedule, we must check if all resulting non-zero dependence vectors are contained in a dependent region, to avoid the occurrence of a cycle.

Predicting a Legal Retiming

Here we introduce a corollary for theorem III..1.

Corollary III..2 *Given a realizable MDFG $G = (V,E,d,t)$, a normal vector n_D for G, and a vector r perpendicular to n_D. If a set $X \subseteq V$ is down-rotatable then retiming X by r never causes a cycle.*

We use the idea of theorem III..1 to find a legal retiming by solving the inequalities $n_D \cdot d(e) > 0$ for every $e \in E$, where n_D is the unknown; we may choose a retiming function from the hyperplane with n_D as the normal vector. The advantage of this option is to avoid the risk of a cycle. Also, the required additional time to select the best retiming function is compensated by the fact that the algorithm does not need to verify the graph, looking for cycles, after each rotation.

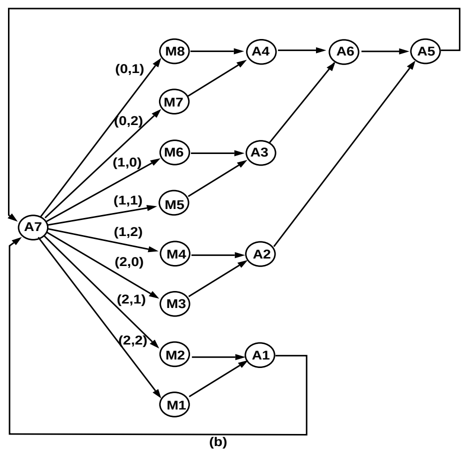


Fig. 7: Initial MDFG representing an IIR filter

IV. Experiments

Due to space limits, we present a single example, which consists of an IIR filter (Infinite Extent Impulse Response) [4], represented by the transfer function:

$$H(z_1,z_2) = \frac{1}{\left(1-\sum_{n_1=0}^{2}\sum_{n_2=0}^{2} c(n_1,n_2)*z_1^{-n_1}*z_2^{-n_2}\right)}$$

which for $k_1, k_2 \neq 0$ can be translated into

$$y(n_1,n_2) = x(n_1,n_2)+ \sum_{k_1=0}^{2}\sum_{k_2=0}^{2} c(k_1,k_2)*y(n_1-k_1,n_2-k_2)$$

Fig. 7(b) shows the MDFG. Assume both adders and multipliers take one time unit to compute. A control step also consists of one time unit. Our list schedule algorithm gives priority to nodes with larger number of successor nodes. Fig. 8(a) shows an optimal schedule with length 10. Let's assume an arbitrary retiming vector $(0,1)$. Fig. 8(b) shows the schedule after node $M8$ has been rotated, with no change in the length of the schedule, since $M8$ and $A7$ are in the same iteration. A more compact schedule of length 9, presented in Fig. 8(c), is obtained when node $M7$ is rotated down and then pushed to a new position. The next rotation operation affects nodes $M6$ and $A4$. $M6$, when pushed down goes to a new position, but $A4$ now is in the same iteration as $M8$ and $M7$, so the length is not reduced, as we can see in Fig. 8(d). One more rotation, give us the optimal schedule length of 8 time units, when node $M5$ is rotated down. The new schedule is shown in Fig. 8(e), and the final retimed graph in Fig. 9(a). Notice, that at every rotation, it was necessary to check the graph for possible cycles.

Our second approach is to find a legal retiming function before rotating any node. After finding the vector $n_D = (1,1)$, we use the vector $(-1,1)$, which is perpendicular to n_D as retiming function. Using the same orig-

CS	Mult.	Adder
1	M8	-
2	M7	-
3	M6	A4
4	M5	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	-	A1
10	-	A7

(a)

CS	Mult.	Adder
1	-	-
2	M7	-
3	M6	A4
4	M5	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	-	A1
10	-	A7
11	M8	-

(b)

CS	Mult.	Adder
1	-	-
2	-	-
3	M6	A4
4	M5	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	M7	A1
10	-	A7
11	M8	-

(c)

CS	Mult.	Adder
1	-	-
2	-	-
3	-	-
4	M5	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	M7	A1
10	M6	A7
11	M8	-
12	-	A4

(d)

CS	Mult.	Adder
1	-	-
2	-	-
3	-	-
4	-	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	M7	A1
10	M6	A7
11	M8	-
12	M5	A4

(e)

CS	Mult.	Adder
1	-	-
2	-	-
3	M6	A4
4	M5	-
5	M4	A3
6	M3	A6
7	M2	A2
8	M1	A5
9	M8	A1
10	M7	A7

(f)

Fig. 8: Arbitrary selection: (a) Initial schedule (b) Schedule after rotating M8 (c) Schedule after rotating M7 (d) Schedule after rotating M6 and A4 (e) Final schedule after rotating M5. (f) Final Schedule for the Prediction method

inal schedule, after the first rotation, we get a schedule where the length was reduced from 10 to 9 time units. After rotating the node $M7$, we get the optimal schedule as presented in figures 8(f) and 9(b). In this case, it was not necessary to verify the existence of cycles according to corollary III..2.

Our experiments had always achieved optimal results. Through them, we notice that the arbitrary selection of the retiming function, generally, give us a shorter prologue. The disadvantage of such approach, is that we may eventually end up with a cycle in our graph. The prediction method does not cause cycle, increasing the chances of getting an optimal schedule.

V. Conclusion

We have introduced a novel technique on scheduling a multi-dimensional data flow graph through the use of a multi-dimensional retiming function that we developed. This new method, named *multi-dimensional rotation scheduling* considers an initial schedule for a multi-dimensional data flow graph, and through an iterative process of successive node rotations, using a pre-selected retiming function, it reduces the schedule length under the resource constraints. The algorithm is presented in detail, showing that the retiming function may be decided through two different approaches. One may arbitrarily select a retiming vector, with the disadvantage of causing a possible cycle during the rotation process, before an optimal result is reached. A second approach generates a retiming vector that guarantees no cycles through the whole process of rotations, allowing a more extensive search for the optimal schedule. Our experiments have shown that our algorithm can obtain the schedule with the shortest possible length efficiently, confirming that the multi-dimensional rotation scheduling is an effective technique in scheduling MD data flow graphs under resource constraints.

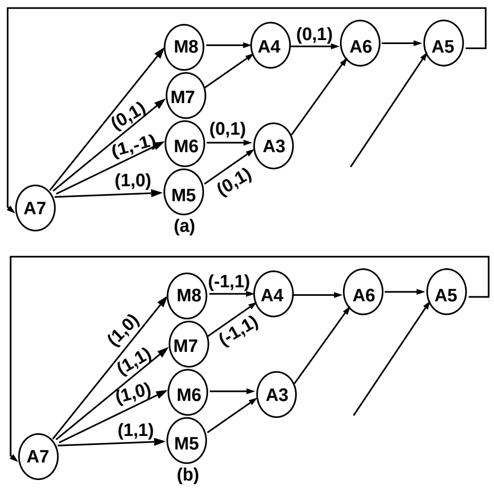


Fig. 9: Retimed subgraphs (a) using arbitrary selection (b) using prediction approach

References

[1] L.-F. Chao, A. LaPaugh, and E. H.-M. Sha, " Rotation Scheduling: A Loop Pipelining Algorithm," *Proc. 30th ACM/IEEE Design Automation Conference* , Dallas, TX, June, 1993, pp. 566-572.

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

[3] S. Davidson, D. Landskov, B. D. Shriver, and P. W. Mallett," Some Experiments in Local Microcode Compaction for Horizontal Machines". *IEEE Transactions on Computers*, C-30, 7, 1981, pp. 460-477.

[4] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[5] G. Goosens, J. Wandewalle, and H. de Man " Loop Optimization in Register Transfer Scheduling for DSP Systems," *Proc. ACM/IEEE Design Automation Conference* , 1989, pp. 826-831.

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

[7] N. L. Passos, E. H.-M. Sha, and S. C. Bass, " Schedule-Based Multi-Dimensional Retiming". To appear in *Proceedings of 8th International Parallel Processing Symposium* , April 1994.

[8] 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.