Processing applications, represented by flow graphs. Such transformation can maximize the parallelism of a loop body. Few results on retiming have been obtained for multidimensional (MD) flow graphs.

The authors are with the Department of Computer Science & Engineering, University of Notre Dame, Notre Dame, IN 46556.

Abstract—Transformation techniques are usually applied to get optimal execution rates in parallel and/or pipeline systems. The retiming technique is a common and valuable tool in optimizing 1-D signal processing applications, represented by flow graphs. Such transformation can maximize the parallelism of a loop body. Few results on retiming have been obtained for multidimensional (MD) systems. This correspondence develops a novel framework, which consists of a MD retiming technique that considers the final schedule as part of the optimization process. To the author's knowledge, this is the first retiming algorithm on general MD flow graphs.

I. INTRODUCTION

With the evolution of multidimensional (MD) signal processing applications such as image processing, and the availability of parallel computing hardware, the demand for optimized solutions for MD problems has increased. At a conceptual level, MD signal processing presents a great deal of similarity to 1-D signal processing. At a detailed level, however, there are important differences between 1-D and MD signal processing.

Each data point processed in digital signal processing (DSP) applications requires a sequence of operations that we call an iteration. The iterations, also called loop bodies, are represented by the execution of each node in a flow graph exactly once. Efficient scheduling mechanisms using retiming techniques have been proposed in previous studies for parallel and/or pipelined processing of 1-D problems represented by flow graphs. The retiming technique regroups operations within an iteration in order to produce a new iteration structure with higher parallelism embedded and, consequently, a shorter schedule length [6], i.e., a shorter execution time for one iteration. The large number of existing data points in MD problems shows that an equivalent approach to the MD case will benefit parallel compiler design for VLIW, data-flow, and superscalar architectures [5], as well as high-level synthesis for VLSI design [2], [7], [8], [9].

In our study, loop bodies are generically represented by MD flow graphs (MDFG). Each node in the graph represents the execution of some task. The edges represent the dependence between tasks and are labeled according to the distance between iterations. We restructure the loop body, i.e., the existing delay dependencies, preserving the original data dependence. We model the process of restructuring as a MD retiming.

Traditional 1-D retiming is useful in many areas such as sequential circuits timing optimization, loop transformation in parallel processing, and real-time scheduling. Our study focuses on the characteristics of the retiming technique that can be applied to MD problems. Chao and Sha introduced the basic principles under the constraints of a specific class of problems where the summation of delay vectors in a cycle was always nonnegative [1]. We target a general form of flow graph, eliminating the above constraints and deriving a practical method of finding a MD retiming function.

On the 1-D retiming, the existing constraint not allowing negative delays ensures that proper data is available when required. However, on the MD case, the existence of negative delays is allowed if there exists a sequence of execution of the iterations, namely, a schedule vector, that turns the flow graph realizable. This characteristic increases the complexity of finding a legal MD retiming function.

Under our framework, in order to use an existing efficient linear-programming solver and to guarantee the realizability of the retimed flow graph, a technique based on characteristics of the final execution schedule is developed. We call this technique schedule-based MD retiming. Experiments show impressive results on the parallelism obtained from the transformation of MDFG's. For simplicity, the retiming technique is studied for the 2-D case where the dimensions are generically referred to as $x$ and $y$. The MD case is a straightforward extension of the concepts presented.

We use the solution of a transmission line problem, submitted to the Fettweis method of solving partial differential equations [3], as an introductory example. Its MDFG is shown in Fig. 1(a), where each node represents the execution of some task at iteration $(i,j)$. Each of these tasks is equivalent to a set of arithmetic instructions represented by functions in the code segment shown in Fig. 2. In the first statement of the example, task $D$ depends on results produced by task $C$ at a distance $(1,1)$. The critical path of this graph determines the execution time of one iteration. If we assume that all tasks have the same execution time $t$, the execution time of one iteration is $3t$. However, applying a MD retiming (2,0) to node $D$ in the graph allows us to move delays from the edges $B - D$ and $C - D$ to $D - A$. This change reduces the schedule length to 2. A second retiming applied to node $A$ results in the graph shown in Fig. 1(b). Its execution time for one iteration is only $t$, one-third of the original time.

Some research has been done on optimization methods for uniform nested-loop scheduling, a similar view to scheduling a MD problem. These techniques differ from our method, since they do not change the structure of iterations, but the sequence in which the instructions are executed. Some research has also been done on the synthesis of MD systems using the "affine-by-statement" scheduling methodology [2], [7], [9] and by applying a new execution order to the iterations in a given block size [8]. These techniques implicitly assume that all operations are executed in one time unit, and that the target...
DO 10 J = 0, T
   DO 100 I = 0, N
   NODE D
      D(I, J) = function1(B(I+1, J-I), C(I-1, J-I))
   NODE A
      A(I, J) = function2(D(I, J))
   NODE B
      B(I, J) = function3(A(I, J))
   NODE C
      C(I, J) = function4(A(I, J))
100 CONTINUE
10 CONTINUE

Fig. 2. Fortran source code segment for the transmission line problem.

Fig. 3. (a) DG based on the replication of a MDFG, showing iterations starting at (0,0); (b) DG based on the replication of a MDFG after retiming.

optimization aims toward a fully parallel solution. Since our method
considers the individual execution time of each instruction as well as
an arbitrary iteration time goal, we can regard the above techniques
as special cases of our algorithm.

II. BASIC PRINCIPLES

A MDFG \( G = (V, E, d, t) \) is a node-weighted and edge-weighted
directed graph, where \( V \) is the set of computation nodes, \( E \subseteq V \times V \)
the set of dependence edges, \( d \) represents the MD delays between
two nodes, and \( t \) is a function representing the computational time
of each node. For the example shown in Fig. 1, \( V = \{A, B, C, D\},
E = \{e1 : (A, B), e2 : (A, C), e3 : (D, A), e4 : (B, D), e5 : (C, D)\},
d(e1) = d(e2) = d(e3) = (0, 0), d(e4) = (-1, 1), d(e5) = (1, 1), \)
each task, for simplicity, is assumed to be executed in one time unit. Therefore \( t(A) = t(B) = t(C) = t(D) = 1 \).

An equivalent cell dependence graph (DG) \( G_{DG} \) of an MDFG \( G \)
is the directed acyclic graph showing the dependencies between
infinite copies of nodes representing the MDFG. Fig. 3(a) shows the
replication of the MDFG in Fig. 1. We call each node representing
a copy of the MDFG, excluding the edges with nonzero delay, a
computational cell. A computational cell is considered an atomic
execution unit. A legal MDFG must have no zero-delay cycle, i.e.,
the summation of the delay vectors along any cycle can not be
\((0, 0, \ldots, 0)\). When sequentially executing the cells in a DG we notice
that some cells will become available to be executed at the same time.
Hyperplanes containing one or more cells with this characteristic are
known as equitemporal hyperplanes [4]. An iteration is the execution
of each node in \( V \) exactly once. An iteration is equivalent to a
computational cell. Iterations are identified by a vector \( i \), equivalent to
a MD index, starting from \((0, 0, \ldots, 0)\). Interiteration dependencies
are represented by vector-weighted edges.

Retiming a MD Data Flow Graph

A cycle period is defined as the period during which all computation
nodes in an iteration are executed according to existing data de-
dendencies and without resource constraints. The cycle period \( C(G) \)
of a MDFG \( G = (V, E, d, t) \) is the maximum computational time
among paths that have no delay, i.e., \( C(G) = \max\{t(p)|p\} \) is a path
in \( G \) with \( d(p) = (0, 0, \ldots, 0) \). Such a path is known as the critical
path of \( G \). Another way to specify the cycle period is based on its
upper bound: \( C(G) \leq c \) if and only if for every path \( p \in G \), if
\( d(p) = (0, 0, \ldots, 0) \) then \( t(p) \leq c \), where \( c \) is a positive constant.
For example, the MDFG in Fig. 1 has \( C(G) = 3 \), which can be
measured through the paths \( D \rightarrow A \rightarrow B \) or \( D \rightarrow A \rightarrow C \).

A 2-D retiming \( r \) is a function from \( V \) to \( Z^2 \) that redistributes the
nodes in the original dependence graph created by the replication of
a 2-DHG \( G \). This results in a new 2-DHG \( G_r \) 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
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 \( u, v, w \in G \). For example, Fig. 4 shows the MDFG in Fig. 1 retimed by the function \( r(D) = (1, 0) \). The critical paths of this graph are the edges \( A \rightarrow B \) and \( A \rightarrow C \), which have an execution time of two time units. The lower left portion of the retimed cell DG, equivalent to the MDFG in Fig. 4, is shown in Fig. 3(b), where the nodes originally belonging to iteration (0, 0) are marked. We note that a possible sequence of execution for the computational cells of the retimed graph is represented by the direction vector (1, 3).

A prologue is the set of instructions that are moved on directions \( x \) and \( y \) in a 2-D retiming, and must be executed to provide the necessary data for the iterative process. In our example shown in Fig. 3(b), the instruction D becomes the prologue for that problem. The epilogue is the other extreme of the DG not shown in the figure, where a complementary set of instructions must be executed to complete the process. Considering that the entire problem consists of a large number of iterations, we may assume that the time required to complete the process. Considering that the entire problem consists of a large number of iterations, we may assume that the time required to run the prologue and epilogue are negligible.

III. SCHEDULE-BASED MULTIDIMENSIONAL RETIMING

In the traditional retiming method, the positive delays after retiming guarantee the retiming as legal. For the MD case, however, positive delay vectors are too restrictive, since a dependence graph is still realizable even if it has negative delays. We call a schedule vector a feasible linear schedule if it supports the realization of the retimed graph and does not affect the cycle period. The following properties are derived from the characteristics of a feasible linear schedule.

1) if node \( v \) depends on node \( u \) where \( u, v \in V \), then \( d(u, v) \cdot s \geq 0 \), i.e., there exists a sequence of execution that does not violate the dependencies.

2) if cell \( g \) depends on cell \( f \) where \( f, g \in G dg \), then \( f, g \) cannot be in the same equitemporal hyperplane; i.e., if \( u \in f \) and \( v \in g \) then \( d(u, v) \cdot s > 0 \) to avoid cycles in the cell dependence graph.

3) if node \( v \) depends on node \( u \) where \( u, v \in V \) and \( d(u, v) \cdot s = 0 \), then \( d(u, v) = (0, 0, \ldots, 0) \), which is an immediate consequence of the two conditions above.

Examining the example in Figs. 4 and 3(b), we can identify several possible feasible linear schedules such as \( s = (1, 3) \), \( s = (1, 4) \), etc. We now define the schedule-based retiming function for 2-D problems. This new function is the basis for our optimization algorithm.

Definition 1: A schedule-based 2-D retiming \( r \) is a function from \( V \) to \( Z^2 \) that redistributes the nodes in the original cell dependence graph of a 2-DFG \( G \), resulting in a new 2-DFG \( G_r \) realizable according to a feasible linear schedule \( s \).

The retiming vector \( r(u) \) works exactly as previously defined. The only extension in this new definition is the requirement for a feasible linear schedule compatible with the retimed graph. It is obvious that a schedule-based 2-D retiming is a legal MD retiming, since a cell dependence graph cannot have a cycle while having a legal schedule. Lemmas 1 and 2 introduce the properties that are used in the proof of theorem 2, which shows how to enforce the correctness of the MD retiming function.

Lemma 1: Let \( G = (V, E, d, t) \) be a 2-DFG, \( r \) a 2-D retiming and \( s \) a feasible schedule vector for the retimed graph \( G_r \). The retiming function \( r \) on the 2DFG \( G_r \), resulting in the 2DFG \( G_r = (V, E, d, t) \), is characterized by the following:

- For any path \( u \rightarrow v \), we have \( d_r(p) = d(p) + r(u) - r(v) \).
- For any cycle \( v \in V \) we have \( d_r(l) = d(l) \).
- For any edge \( u \rightarrow v \), if \( d_r(v) \neq (0, 0) \) then \( d_r(v) \cdot s > 0 \).

Looking again at Fig. 4, where the legal retiming function applied to both sides we have \( d(p) \cdot s = \sum_{i=1}^{k} d_i e_i \cdot s \). Since a feasible linear schedule \( s \) requires \( d(e) \cdot s \geq 0 \), the proof is immediate.

We define the auxiliary functions \( T(k) \) and \( T(k) \) to be used to identify a set of critical paths such that the retiming problem can be solved focusing on these critical paths.

Definition 2: Given a MDFG \( G = (V, E, d, t) \), we define the function \( T^{(k)}(u, v) \) from \( V \times V \) to \( N \) as the minimum number of equitemporal hyperplanes between nodes \( u \) and \( v \) in \( V \) according to the schedule vector \( s \). We can formulate such a definition as:

\[
T^{(k)}(u, v) = \min \{ d(p) \cdot s | u \xrightarrow{p} v \in G \}
\]

Definition 3: Given a MDFG \( G = (V, E, d, t) \), \( u, v \in V \), we define the function \( T^{(k)}(u, v) \) from \( V \times V \) to \( N \), as the maximum computation time between nodes \( u \) and \( v \), inclusive, according to the schedule vector \( s \) such that \( u \xrightarrow{p} v \) and \( d(p) \cdot s = T^{(k)}(u, v) \).

We can formulate such a definition as:

\[
T^{(k)}(u, v) = \max \{ t(p) | u \xrightarrow{p} v \in G \}
\]

A slight modification of a shortest path algorithm can compute \( D^{(k)}(u, v) \) and \( T^{(k)}(u, v) \). The paths \( u \xrightarrow{p} v \) such that \( d(p) \cdot s = D^{(k)}(u, v) \) and \( t(p) = T^{(k)}(u, v) \) are considered the critical paths from \( u \) to \( v \). In our example in Fig. 1, for a schedule vector \( s = (1, 3) \), we obtain \( D^{(k)}(A, D) = \min \{ d(A - B) \cdot s, d(A - C) \cdot s \} = \min \{ (A - B), (A - C) \} = \min \{ (1, 3) \} = 3 \) and \( T^{(k)}(A, D) = \max \{ t(A - B), t(A - C) \} = 3 \).

Theorem 1: Let \( G = (V, E, d, t) \) be a 2-DFG, \( s \) a feasible linear schedule vector for the graph \( G \), and \( c \) a positive integer. Then the cycle period is less than or equals to \( C(G) \leq c \) if and only if for all nodes \( u \) and \( v \) in \( V \), if \( D^{(k)}(u, v) < 1 \) then \( T^{(k)}(u, v) < c \).

Proof: To prove the only if part by contradiction, we assume \( C(G) < c \) and that there exist \( u, v \in V \) such that \( D^{(k)}(u, v) < 1 \) and \( T^{(k)}(u, v) > c \). Consider \( p \) a critical path from \( u \) to \( v \) with \( d(p) \cdot s = D^{(k)}(u, v) \) and \( t(p) = T^{(k)}(u, v) \). Thus, there exists a path \( p \) in \( G \) such that \( t(p) > c \) and \( d(p) \cdot s < 1 \). From lemma 2, \( d(p) \cdot s = 0 \) and \( C(G) \) must be greater than \( c \), which is a contradiction.

For the if part, we assume that for all \( u, v \in V \) if \( D^{(k)}(u, v) < 1 \) then \( T^{(k)}(u, v) \leq c \). We claim that for every path \( p \) in \( G \), either
theorem shows the necessary and sufficient conditions for a 2-D retiming with \( C(G_r) \leq c \) in terms of \( D^{(0)} \) and \( T^{(0)} \).

**Theorem 2.** Let \( G = (V, E, d, t) \) be a 2-DFG, \( r \) a 2-D retiming on \( G \), \( s \) a feasible linear schedule vector for the retimed graph \( G_r \), and \( c \) a positive integer. Then \( r \) is a legal schedule-based MD retiming on \( G \) such that \( C(G_r) \leq c \) if and only if

- a) \( r(e) = r(u) - s < d(e) \cdot s \) or \( r(e) - r(u) = d(e) \), for every edge \( u \rightarrow v \);
- b) if \( T^{(0)}(u, v) > c \) then \( r(v) \cdot s - r(u) \cdot s \leq D^{(0)}(u, v) - 1 \) for every pair \( u, v \) in \( G \).

**Proof:** For part (a) from Lemma 1, \( r \) is a legal MD retiming if and only if \( d, r(e) - s > 0 \) or \( d, r(e) = 0 \). Since \( d, r(e) = d(e) + r(u) - r(e) \), the proof is obvious. For part (b), we begin to prove that the retiming function effects on \( D^{(0)} \) and \( T^{(0)} \) are in the same way as on functions \( d \) and \( t \), i.e.,

- 1) \( D^{(0)}(u, v) \leq D^{(0)}(u, v) + r(u) - r(v) - s \)
- 2) \( T^{(0)} = T^{(0)} \)

For 1. \( D^{(0)}(u, v) = \min\{d(p) \cdot s | u \rightarrow v \in G_r\} \) since \( d(p) = d(p) + r(u) - r(v), D^{(0)}(u, v) = \min\{d(p) \cdot s | u \rightarrow v \in G_r\} at(v) - r(v) - s \cdot r(v) \cdot s \cdot r(v) \cdot s \rightarrow v \in G_r \), \( D^{(0)}(u, v) = \min\{d(p) \cdot s | u \rightarrow v \in G_r\} + r(u) - r(v) - s \cdot r(v) - s \cdot r(v) - s \cdot r(v) - s \).

For 2. \( T^{(0)} = \max\{d(p) \mid u \rightarrow v \in G_r \), \( d(p) = D^{(0)}(u, v), \) similarly to (1), \( T^{(0)} = T^{(0)} \).

From Theorem 1 we have \( C(G) \leq c \) if for all nodes \( u \) and \( v \) in \( V \), if \( D^{(0)}(u, v) < 1 \) then \( T^{(0)}(u, v) < c \). Substituting according to 1 and 2, we obtain our claim: If \( T^{(0)}(u, v) > c \) then \( r(v) \cdot s - r(u) \cdot s \leq D^{(0)}(u, v) - 1 \) for every path \( u \rightarrow v \) in \( G \).

At this point, we have the formulation of the constraints for a legal MD retiming function.

### IV. Our Algorithm

In this section, we show how to translate the formulation presented in Theorem 2 into an integer linear program (ILP). The first problem to be solved is how to formulate the existent or condition in part (a) of that theorem. In order to satisfy the constraint \( d, e \cdot s > 0 \) or \( d, e \cdot s = 0 \) (part (a) of Theorem 2), the condition \( d, e \cdot s = 0 \), with \( d, e \neq 0 \) must be avoided. Therefore, we need to exclude the possibility of any retimed dependence vector perpendicular to the schedule vector \( s \). In practice, we use two extra vectors to enforce such exclusion, defining a region that contains the undesired vectors.

**Definition 3:** A rotational exclusion angle \( \alpha_s \) for a 2-D retiming \( G = (V, E, d, t) \) is any angle by applying a rotation to a schedule \( s \), creating a new vector \( s' \), such that the dependency vectors that are perpendicular to \( s \) but that have an angle with \( s \) between \( \frac{\pi}{2} - \alpha \) and \( \frac{\pi}{2} + \alpha \). Point \( P \) in Fig. 5 suggests a vector contained in such region that is excluded from the feasible solution, in addition to those vectors perpendicular to \( s \). We define such an angle below.

**Definition 4:** A rotational exclusion angle \( \alpha_s \) for a 2-D retiming \( G = (V, E, d, t) \) is any angle by applying a rotation to a schedule \( s \), creating a new vector \( s' \), such that the dependency vectors that are perpendicular to \( s \) but that have an angle with \( s \) between \( \frac{\pi}{2} - \alpha \) and \( \frac{\pi}{2} + \alpha \). Point \( P \) in Fig. 5 suggests a vector contained in such region that is excluded from the feasible solution, in addition to those vectors perpendicular to \( s \). We define such an angle below.
Theorem 3: Let $G = (V, E, d, t)$ be a 2-DFG, $r$ a 2-D retiming on $G$, $s$ a feasible linear schedule vector for the retimed graph $G_r$, $\alpha_{s,b}$ a rotational exclusion angle limited by an integer value $b$, and $c$ a positive integer. Then $r$ is a legal schedule-based MD retiming on $G$ such that $C(G_r) \leq c$, excluding dependencies in $\alpha_{s,b}$ if and only if

1) $r(v) - s_r - r(u) - s_r \leq d(e) \cdot s_r$, and $r(v) - s_{ec} - r(u) - s_{ec} \leq d(e) \cdot s_{ec}$, for every edge $v \rightarrow u$ in $G$.
2) if $T(u, v) > c$ then $r(v) \cdot s - r(v) \cdot s \leq D(u, v) - 1$, for every path $u \rightarrow v$ in $G$.

Proof: For part (a), it is clear that, using Lemma 3, part (b) is proven in Theorem 2.

A heuristic method has been developed considering the constraints presented in Theorem 3. Other constraints can be added to the choice of the final schedule vector, such as computing the minimal execution time in a massive parallel processing architecture, or considering an optimal pipeline rate when mapping the DG to an array processor. With the schedule vector chosen, a shortest path algorithm, modified from Floyd algorithm, is used to compute the functions $D^{(s)}$ and $T^{(s)}$. Finally, an ILP program written according to the constraints in Theorem 3 is used to compute the retiming function. The schedule-based MD retiming algorithm for $C(G) \leq c$ can be summarized in the following steps:

1) Choose a range of possible values for the schedule vector components, and a value for the lower limit $b$ in the $b$-rotational exclusion angle.
2) From the range defined in step 1, select the schedule vector within the constraints imposed in Theorem 2.

3) Compute functions $D^{(s)}$ and $T^{(s)}$ using the modified Floyd algorithm.
4) Calculate $s_r$ and $s_{ec}$ to determine the $b$-rotational exclusion angle.
5) Run the ILP solver, considering the $b$-rotational exclusion angle and a target cycle period $c$. The set of constraints is generated according to Theorem 3.
6) If there is no solution, go back to step 2 and select a new schedule.

VI. EXPERIMENTS

In this section, we present a practical example on the application of our technique. The target problem consists of a wave digital filter designed to solve a system of partial differential equations. We assume that the target cycle period is one time unit and that all nodes have execution time equal to one unless otherwise stated. This target cycle period is equivalent to obtaining full parallelism among the nodes in one iteration, which is a special case of the application of our method. The filter was designed according to the Fettweis method [3] to solve the transmission line problem characterized by the following equations:

$$\frac{\partial l_i}{\partial t_i} + r_i + \frac{\partial u}{\partial t_i} = f_i$$

and

$$\frac{\partial l_i}{\partial t_i} + c\frac{\partial u}{\partial t_i} + g_u = f_2.$$  

After applying the Fettweis transformations we obtain the wave digital filter shown in Fig. 6(a), which is equivalent to the MDFG in Fig. 6(b) where the inputs $e_1$ and $e_2$ are zero. The two-port adaptors, A and B, are expanded to their internal configuration. For simplicity, one output port and one adder in each adaptor were deleted because the particular boundary conditions applied. Fig. 7 shows the retimed MDFG with a total of 12 nodes for a schedule vector $= (1, 7)$. The resulting retiming function was $r(A1) = (3, 0), r(A2) = (2, 0), r(A3) = (1, 0), r(B1) = (3, 0), r(B2) = (2, 0), r(B3) = (1, 0), r(C) = r(D) = (0, 0), r(E) = r(F) = (5, 0), r(G) = r(H) = (4, 0)$.

VI. CONCLUSION

We have introduced a novel technique on transforming a multidimensional data flow graph to obtain a high level of parallelism.
This new technique, schedule-based MD retiming, considers a feasible linear schedule as part of the process of redistributing operations along iterations, while keeping the original data dependencies. The method consists of applying some constraints in the process of identification of the appropriate schedule vector for the expected retimed graph and formulating a set of inequalities to be used in an ILP solver to get the final results. From the experiments we conclude that there always exists a schedule-based MD retiming function that allows all nodes in the flow graph representing a problem with two or more dimensions to be executed in parallel.

REFERENCES