Retiming of Circuits Containing Multiplexers

Sven Simon, Johann Hofner and Josef A. Nossek
Institute for Network Theory and Circuit Design
Technical University Municn
Arcisstr. 21, 80333 Munich, Germany

Abstract—**Classical retiming optimization algorithms do not consider circuits containing multiplexers or demultiplexers driven by a clock signal [1], because the associated retiming equations differ from the special classical form, which make applicable combinatorial algorithms of polynomial order. In order to provide an algorithm for multiplexer circuits it is shown here that retiming, being an integer linear programming problem inherently, can be relaxed to a linear programming formulation with real valued variables. This is due to the unimodularity of the matrices of the retiming formulation. Multiplexer circuits change this property in a way, which suggests how to use an integer linear programming problem to derive an polynomial retiming algorithm.**

I. Introduction

The synthesis of high speed circuits is an important topic of VLSI design. Retiming, a methodology to speed up circuits without introducing additional pipeline stages, was introduced initially by Leiserson *et al.* [1,2]. Registers are relocated in a way that data dependencies remain unchanged but maximize the clock frequency or other optimization criteria. Since the basic concept of retiming was formulated nearly a decade ago, a lot of research work was done in order to make it applicable to CAD designs and tools.

Further work examined the usage of more sophisticated circuit models. Soyata, Friedman and Mulligan integrated clock skew and register delays into retiming [3]. In [4] and [5], Lockyear, Ebeling, Ishii and Leiserson studied retiming with level-sensitive latches. De Micheli examines logic synthesis and cycle-time minimization without separating the combinational elements from registers, see [6]. In [7], Malik *et al.* consider retiming with logic synthesis. Registers are temporarily removed from the circuit in order to apply combinational optimization to the logic elements. Additionally, in [8,9], Potkonjak, Dey *et al.* apply algebraic speed up to those temporarily register free subcircuits.

In this paper multiplexer circuits are examined concerning retiming. It is shown in Section 2 that the introduction of multiplexers, driven by a clock signal, modifiy the usual retiming problem. In Section 3, retiming is formulated as a linear programming problem (LP) for circuits, which do not contain any multiplexer. The associated matrices have the so called unimodular property such that the simplex algorithm can be used to compute an optimum retiming solution. The introduction of multiplexers change this unimodular property. A polynomial algorithm to solve this optimization problem is given in Section 4, which introduces additional edges to the circuit graph and which is based on the cutting plane extension of the simplex algorithm.

[1] These circuits are called multiplexer circuits from now on.

The Model

In the following, the graph model, presented in Leiserson's paper is used. Each circuit is characterized by a directed graph with a set of vertices V and a set of edges E. An edge weight w, which represents the number of registers of an edge, and a vertex delay d, which is the combinational delay of a vertex, is associated with each edge and vertex respectively. Thus, a circuit can be abbreviated by $G =< V, E, d, w >$. The number of registers moved over every vertex v by retiming is assigned to the variable $r(v)$. Thus, the number of registers w_r after retiming of an edge $e(u \rightarrow v)$ can be computed as follows [1]:

$$w_r(e) = w(e) + r(v) - r(u). \qquad (1)$$

This equation is denoted as retiming equation from now on.

II. Retiming equations of multiplexer circuits

The retiming equations (1) for edges adjacent to a multiplexer vertex are derived in the following. A register being moved over a multiplexer vertex u with k incoming edges provides k registers in the outgoing edge $e(u, v)$, which is illustrated in Fig. 1.

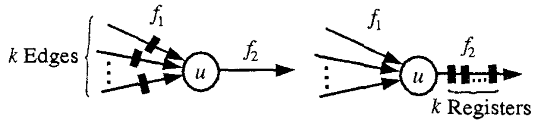

Fig. 1. Moving registers over a multiplexer vertex.

This is obvious, if the cut-set rule, which inserts a delay Δt for the incoming data of the cut-set and $-\Delta t$ for the outgoing data, is examined. Usually, the time interval Δt is identical to the number of registers $r(u)$ moved over a vertex u because of the uniform clock frequency of the circuit. Data leaving the multiplexer vertex have a greater clock frequency f_2 than arriving data with the clock frequency f_1. Therefore, referring to the clock frequency f_1, the quotient of the frequencies f_2/f_1 has to be taken into account such that the delay Δt for an outgoing edge moving $r(u)$ registers from the incoming edges over vertex u results in: $\Delta t = r(u)f_2/f_1$. The ratio f_2/f_1 is identical to the number k of incoming edges, if multiplexer circuits are considered. We obtain the following retiming equation for u being a multiplexer and demultiplexer vertex respectively:

$$\begin{array}{ll} \text{Multiplexer:} & w_r(e) = w(e) + r(v) - kr(u) \\ \text{Demultiplexer:} & w_r(e) = w(e) + r(v) - k^{-1}r(u). \end{array} \qquad (2)$$

The retiming equations (1) can be regarded as a special integer LP (ILP) concerning the variables r. This ILP consists of inequalities, which only contain the differences of the unknowns

$x_1, x_2, ..., x_n$, which in our case are equal to the unknowns r. This can be written as $x_j - x_i \leq a_{ij}$ with the constants a_{ij}. Thus, shortest path based algorithms can be used to solve this problem. As retiming equations of multiplexers (2) do not have this property ($x_j - kx_i \leq a_{ij}$), a new retiming algorithm has to be developed.

III. RETIMING AS A LINEAR PROGRAMMING PROBLEM

In this section, retiming is formulated as a linear programming problem for circuits, which do not contain multiplexers, in order to derive a polynomial algorithm in the following section for multiplexer circuits. The retiming equation (2) can be written in matrix form [10] with the vector $\mathbf{w}$ and $\mathbf{w}_r$ for the positive register weights: $\mathbf{w}_r - \mathbf{w} = \mathbf{A}^{\mathbf{T}}\,\mathbf{r}$, $\mathbf{w}_r \geq 0$. Matrix $\mathbf{A}$ is the incidence matrix of the circuit G. These equations can be used to formulate the set of all retiming solutions using the vector $\mathbf{r}$ of the vertex labels r:

$$\mathbf{A}^{\mathbf{T}}\,\mathbf{r} \leq -\mathbf{w}. \tag{3}$$

Alternatively, the set of all retiming solutions can be formulated using $|E| - (|V| - 1)$ loop equations, as stated in [10], with $|E|$ and $|V|$ being the number of elements of the sets E and V. Here, the following lemma is expanded on undirected loops instead of directed cycles [10] only.

Lemma 1. *The register sum of a loop is constant during retiming, if the sum is computed as follows: Add the register weights of edges, if the edge direction and loop orientation of which coincide, and subtract the register weights otherwise. This sum is invariant during retiming:*

$$\sum\nolimits_{\text{loop}} w_r(i,j) = \sum\nolimits_{\text{loop}} w(i,j). \tag{4}$$

Proof. Add the Equations (1) of each edge belonging to the loop. Multiply the equation being added with -1, if the direction of the associated edge is opposite to the chosen loop direction. Thus, the variables r are eliminated in the final sum and (4) is obtained. □

The $|E| - (|V| - 1)$ loop equations can be written in matrix form with the vector of non-negative register weights $\mathbf{w}_r$:

$$\mathbf{M}\,\mathbf{w}_r \;=\; \mathbf{M}\,\mathbf{w}, \quad \mathbf{w}_r \geq 0. \tag{5}$$

Matrix $\mathbf{M}$ is the circuit matrix of G with each row representing one loop. The circuit matrix $\mathbf{M}$ and the incidence matix $\mathbf{A}$ belong to a special class of martices:

Definition. *A matrix is totally unimmodular, iff all determinantes of its submatrices are -1, 0, 1.*

Lemma 2. *The incidence and circuit matrix of a circuit graph G are totally unimodular.*

Proof. See [12,13].

The Systems (3) and (5) containing all possible retiming solutions of the circuit G with the integer variables r and w_r represent a convex polytope such that the following Theorem of Hoffman and Kruskal [14] can be used:

Theorem 1. *The extreme points of the convex polytope $\mathbf{C}\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq 0$ are all with integer components for any arbitrary integer vector $\mathbf{b}$, iff matrix $\mathbf{C}$ is totally unimodular.*

Proof. See [12,14].

The optimum retiming solution concerning a cost function of the integer variables r or w_r can be computed such that the retiming problem can be regarded as an ILP. Due to the total unimodularity of the matrices the simplex algorithm can be used to solve this retiming ILP, although the simplex algorithm produces real valued solutions in general. The simplex algorithm starts with an arbitrary vertex of the polytope and moves from vertex to vertex improving the value of the cost function each step. As all vertices are integer as stated in Theorem 1, the simplex algorithm provides an integer solution. The simplex algorithm will probably not lead to a reduced computation time compared to classical retiming algorithms but an extension of the simplex algorithm is suited to solve the multiplexer problem, see Section 4. Besides, the simplex algorithm being generally available e.g. in computer algebra systems like Mathematica the optimum retiming solution of a circuit can be computed, if no appropiate CAD tool is available.

In order to find a retiming solution, which fulfills a certain clock frequency or obeys certain structural constraints, the effect of those constraints on the retiming ILP has to be examined. This can be done, considering whether and how these constraints influence the unimodularity of the circuit matrix.

1. *Timing Constraints*

In [1], two matrices $\mathbf{W}$ and $\mathbf{D}$ of dimension $|V|$x$|V|$ are computed from G, to formulate timing constraints. Matrix $\mathbf{W}$ contains the shortest path values concerning register weights of every pair of vertices. The entries $D(u,v)$ of the matrix $\mathbf{D}$ are the maximum path value concerning the vertex delays d among the shortest paths found by the computation of $\mathbf{W}$. A path between two vertices u and v must contain one register minimum after retiming, if the path delay $D(u,v)$ is greater than the required clock period c. This is formulated in [1] as: $r(v) - r(u) \leq W(u,v) - 1$ if $D(u,v) \geq c$. These inequalities with the unknowns r have the same structure as mentioned above ($x_j - x_i \leq a_{ij}$) such that the timing condition can be regarded as additional edges of a modified graph $\tilde{G}$. Let matrix $\tilde{\mathbf{M}}$ be the circuit matrix of the modified circuit graph $\tilde{G}$, $\tilde{\mathbf{M}}$ is totally unimodular according to Lemma 2. Thus, the simplex algorithm will produce integer solutions, if timing constraints are applied, additionally.

2. *Structural Constraints*

2.1. Local Data Communication. Broadcast interconnections can be avoided and local data communication is achieved, if certain edges contain a minimum number of registers: $w_r(u,v) \geq w_m$. This condition can be written as an equation introducing the variable $\tilde{w}_r(u,v) \geq 0$: $w_r(u,v) - \tilde{w}_r(v,u) = w_m$. The equation can be interpreted as an additional loop such that the circuit matrix is modified to an unimodular circuit matrix $\tilde{\mathbf{M}}$ of an extended problem. Therefore, the optimum solution computed by the simplex algorithm is integer in general.

2.1. Modularity. In order to obtain a modular design vertices should have identical surrounding register configurations such that the number of registers $w(u,v)$ and $w(i,j)$ of certain edges $e(u,v)$ and $e(i,j)$ should be identical: $w(u,v) = w(i,j)$. If this identity is applied to the circuit matrix **M** of the original circuit, the column of the edge $e(i,j)$ has to be added to the column $e(u,v)$ and the column of the edge $e(i,j)$ has to be deleted or vice versa. Obviously, this will not lead to unimodular circuit matrices in general. Therefore, the associated polytope may have vertices with non-integer coordinates and the presented retiming algorithms have to be extended.

In order to derive an algorithm for the multiplexer problem the structure of the circuit matrix as done for timing and structural constraints is examined for multiplexer circuits in the following section.

IV. A RETIMING ALGORITHM FOR MULTIPLEXER CIRCUITS

Lemma 1 has to be modified for multiplexer circuits as follows:

Lemma 3. *Compute the sum as done in Lemma 1 replacing the register weight by the product of register weight and associated clock period $T(i,j)$ of the considered edge $e(i,j)$:*

$$\sum\nolimits_{\text{loop}} T(i,j)w_r(i,j) = \sum\nolimits_{\text{loop}} T(i,j)w(i,j). \qquad (6)$$

Proof. To proof the correctness of (6), the cut-set rule is considered, which inserts the delays Δt in the edges of G. The sum $\sum \Delta t$ around a loop of G is zero. The delay Δt of an edge applying a cut-set rule is the difference of registers before and after retiming multiplied with the clock period of data propagating along this edge: $\Delta t = T(i,j)(w_r(i,j) - w(i,j))$. Applying this to the sum $\sum \Delta t$, which equals zero, Equation (6) is obtained. □

Equation (4) is identical to (6), if the circuit contains no multiplexer at all because all $T(i,j)$ have ths same value $T = 1/f$. Equation (6) can also be written in matrix form, introducing a square matrix **K** of the dimension $|E|\text{x}|E|$, additionally. This square matrix **K** is diagonal with the diagonal entries $T(i,j)$ because to obtain (6), the columns of **M** have to be scaled with $T(i,j)$. Lemma 3 is formulated in matrix form as follows:

$$\mathbf{M\,K\,w}_r = \mathbf{M\,K\,w}, \quad \mathbf{w}_r \geq 0. \qquad (7)$$

Matrix **K** is the product of the $|E|$ dimensional indentity matrix **I** and the vector **T** of the clock periods: $\mathbf{K} = \mathbf{I\,T}$. If the clock period of each edge is not given, matrix **K** can be computed with a unit time scale using the retiming equations (2) with the multiplexer constants k, see [11].

Since the matrix product **M K** is not necessarily unimodular, the optimum solution may be non-integer. This optimum can either be computed with the simplex algorithm applied to (7) or classical retiming algorithms of polynomial order using the following substitution: $\mathbf{w}_r' = \mathbf{K\,w}$. The retiming formulation results in $\mathbf{M\,w}_r' = \mathbf{M\,K\,w}$, which is identical to a circuit with every multiplexer being treated as a usual combinational vertex but modified edge weights before retiming.

Obviously, the final values $\mathbf{w}_r$ need not to be integer. In order to provide an algorithm to compute integer register values let us assume that a loop containing a multiplexer vertex with constant k contains an associated demultiplexer with constant k such that only pairs of multiplexers and demultiplexers exist. Otherweise, registers would be created, if a register is moved around this loop and a conflict of clock periods of date associated with a vertex would occur.

Algorithm RMC *(retiming of multiplexer circuits)*

1. Add an edge between each multiplexer and demultiplexer vertex of a pair with the clock period associated with the leaving edge of the multiplexer. Choose an arbitrarily high number of registers (e.g. number of all registers of the circuit multiplied with all multiplexer constants) to each of those new edges before retiming such that the same optimum retiming solution can be computed as before.
2. Find the optimum retiming solution with the simplex algorithm or the classical retiming algorithm applying the substitution: $\mathbf{w}_r' = \mathbf{K\,w}$. Let $\hat{w}_i$ be the optimum register weights of the elements of the vector $\mathbf{w}_r'$.
3. Consider all paths from a demultiplexer to a multiplexer vertex of a pair with constant k. If a path p contains an edge e_i with a non-integer register weight w_i, assign the following number of registers w_c before retiming to the edge e_c, inserted in step 1: $w_c = k\sum_i(\lfloor\hat{w}_i\rfloor - w_i)$. The summation concerns the values $(\lfloor\hat{w}_i\rfloor - w_i)$ of all edges belonging to the chosen path p.
4. Use the register weights w_i before retiming with the modified value w_c of step 3 and go to step 2 until all multiplexer and demultiplexer pairs with non-integer path weights inbetween are treated. If multiplexer demultiplexer pairs are nested, begin with the innermost pair.

This algorithm leads to the integer optimum solution because step 3 can be interpreted as a cutting-plane algorithm for integer linear programs as shown in the following. Path p and edge e_c form a loop which yields the loop equation: $\sum_i w_i + \frac{1}{k}w_c = \sum_i \hat{w}_i + \frac{1}{k}\hat{w}_c$. Obviously, the optimum retiming solution concerning the edges w_i is not excluded, if the value $w_c = k\sum_i(\hat{w}_i - w_i)$ before retiming is chosen such that the number of register after retiming is $\hat{w}_c = 0$. The considered loop represents one row of the associated simplex tableau. According to the cutting-plane algorithm for integer linear programs, see e.g. [15, p. 329], the following inequaltiy derived from this row can be added to the tableau: $w_c \geq k\sum_i(\hat{w}_i - \lfloor\hat{w}_i\rfloor)$. This inequality cuts off the non-integer optimum vertex of the polytope and can be fulfilled by the proposed retiming algorithm, if the value $k\sum_i(\hat{w}_i - \lfloor\hat{w}_i\rfloor)$ is subtracted from the register weight $w_c = k\sum_i(\hat{w}_i - w_i)$ before retiming because the register weights are non-negative inherently. Thus, the value w_c of step 3 is obtained.

As timing constraints can be interpreted as additional edges without changing the algorithm, those edges can be instered, if the clock period of the associated vertices are equal.

If classical retiming algorithms are applied like the minimum cost-flow algorithm of polynomial computation time $f(|V|,|E|)$ [1] and if N is the number of multiplexer demultiplexer pairs, then the computation time of the Algorithm RMC is $O(Nf(|V|,|E|))$ because the optimium retiming solution is computed worst case $N+1$ times.

V. Results

A. An Application

In order to illustrate the Algorithm RMC the following example is chosen, see Fig. 2. This circuit is the arithmetic cell of a DSP application called block matching, which is used for motion estimation in image processing [16]. Two succesive images are divided into blocks. In order to find the block of the current image, which matches best with the block of the previous image, the distance L $= \sum_i |x_i - y_i|$ with the pixel values x_i, y_i of the current and previous block is computed. The arithmetic cell of this problem is implemented by the circuit shown in Fig. 2a.

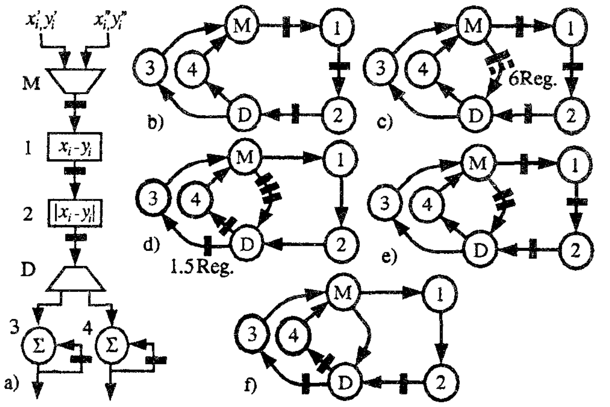

Fig. 2. Example circuit and circuit graph G.

As the wordlength for the computation of the absolute value $|x_i - y_i|$ is less than the wordlength of the final summation $\sum$, it is assumed that the pixel rate and propagation delay of the vertices allow hardware sharing with a multiplexer and demultiplexer. The associated circuit graph G for retiming is shown in Fig. 2b. The self loops of the vertices $3, 4$ are removed from G because they will have no effect on retiming. The input and output edge of the circuit are connected to obtain a strongly connected circuit graph as usually required for retiming. The edge weights $w_r(i, j)$ are abbreviated as the vector elements w_j. The results of each step of the Algorithm RMC are given as follows:

1. Fig. 2c shows the circuit graph with the additional edge between multiplexer and demultiplexer with a register weight before optimization, which does not exclude the optimum solution computed without this edge.
2. The cost function $S = -w_3$ is chosen such that the optimum register configuration results in the circuit given in Fig 2d.
3. The computation of the register weight $w_c = k \sum_i (\lfloor \hat{w}_i \rfloor - w_i)$ yields $w_c = 2$ as shown in Fig. 2e.
4. The final optimization provides the integer optimum solution, see Fig 2f.

B. Computation Time

In this section a comparision between the incidence matrix and circuit matrix formulation concerning the computation time of the optimum retiming solution with the simplex algorithm is presented. The correlator given in [1] is generalized to a n-bit correlator with $V = 2n$ vertices. The optimization concerning maximum clock speed using the circuit matrix formulation is 6.7 times faster than the incidence matrix formulation in average, which corresponds to the constant gap of both curves in Fig. 3 with a logarithmic scale of the computation time. For other circuits the circuit matrix formulation also seems to be the profitable choice. As the simplex algorithm is no competitor concerning computation time compared to classical retiming algorithms, we refrained from implementing an efficient sparse matrix simplex algorithm for retiming.

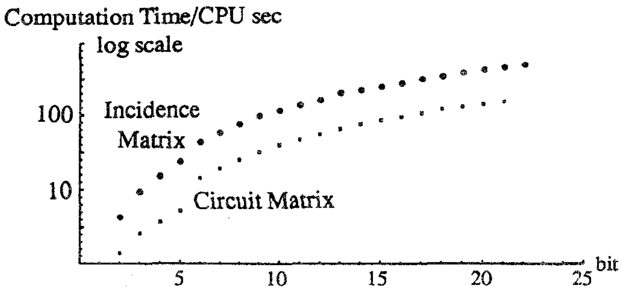

Fig. 3. Computation time of the n-bit correlator.

VI. Conclusion

In this paper retiming is regarded as a linear programming problem with the total unimodular property of the associated matrices. This formulation is used to derive a polynomial algorithm for multiplexer circuits from general non-polynomial integer linear programming methodologies. As a result beyond the scope of introducing retiming to multiplexer circuits the unimodularity can be used generally for examining the influence of additional constraints or properties to retiming. Future work will focus on how other linear programming methodologies can be modified to develop further profitable retiming algorithms.

References

[1] C. E. Leiserson and J. B. Saxe. Optimizing Synchronous Circuitry and Retiming . *Proc. of the 3rd Caltech Conf. on VLSI*, pages 87–116, 1983.

[2] C. E. Leiserson and J. B. Saxe. Retiming Synchronous Circuitry . *Algorithmica*, pages 5–35, 1991.

[3] T. Soyata and E. G. Friedman and J. H. Mulligan. Integration of Clock Skew and Register Delays into a Retiming Algorithm, ISCAS 93.

[4] B. Lockyear and C. Ebeling. Optimal Retiming of Multi-Phase Level-Clocked Circuits . *Advanced Research in VLSI 1992*, pages 265–280.

[5] A. T. Ishii and C. E. Leiserson. Level-Clocked Circuitry. *Advanced Research in VLSI 1992*, pages 245–264.

[6] G. De Micheli. Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization *IEEE Trans. on CAD*, 10:63–73, 1991.

[7] S. Malik and E. M. Sentovich. Optimizing Sequential Networks with Combinatorical Techniques . *IEEE Trans. on CAD*, 10:74–84, 1991.

[8] Z. Iqbal M. Potkonjak, S. Dey and A. C. Parker. High Performance Embedded System Optimization Using Algebraic and Generalized Retiming Techniques. In *ICCD 93*, 1993.

[9] M. Potkonjak S. Dey and S. G. Rothweiler. Performance Optimization of Sequential Circuits by Eliminating Retiming Bottlenecks. In *ICCAD*, pages 510–517, 1992.

[10] S. Simon, E. Bernard, M. Sauer and J. A. Nossek. A New Retiming Algorithm for Circuit Design . In *Proc. of the Int. Symp. on Circuits and Systems*, vol. 4, pages 35–38, 1994.

[11] S. Simon and J. Hofner. Retiming Algorithms for Multiplexer Circuits. Technical Report TUM-LNS-TR-94-8, Institute for Network Theory and Circuit Design, Technical University Munich, 1994.

[12] T.C. Hu. *Integer Programming and Network Flows*. London, Addison-Wesley, 1969.

[13] W. Mayeda. *Graph Theory*. New York, Wiley-Interscience, 1972.

[14] A. J. Hoffman and J. B. Kruskal. Integral Boundary Points of Convex Polyhedra" . *Annals of Mathematics Study No. 38, 1956*, pages 223–246.

[15] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*, Prentice-Hall, 1982.

[16] L. De Vos and M. Stegherr. A Family of Application Specific VLSI Architectures for the Block-Matching Algorithm. In *Systolic Array Processors*, pages 421–430, 1989.