Retiming of Latches for Power Reduction of DSP Designs

S. Simon, C. V. Schimpfle, M. Wroblewski and J. A. Nossek
Institute for Network Theory and Circuit Design
Technical University Munich
Arcisstr. 21, 80333 Munich, Germany
Email: svsi@nws.e-technik.tu-muenchen.de

Abstract– In this paper a retiming methodology is proposed which reduces the power consumption of digital CMOS circuits. The application of high level synthesis tools for arbitrary designs usually leads to the usage of edge triggered registers. However, VLSI implementations of DSP algorithms which are considered here make level sensitive registers applicable. Level sensitive registers consist of two latches which store the data for half a clock period. If these latches are placed separately in the circuit then glitching can be reduced and single latches can store data on the gate capacity of the logic instead of the gate of additional inverters. These two effects reduce the power dissipation of the total circuit and savings of the considered DSP implementation up to 20 % or more have been achieved.

I. Introduction

The efficient implementation of DSP applications is an important topic of VLSI design. Traditionally, a full custom design style is used for dedicated DSP implementations because the regularity and structure of DSP architectures can be used on all design levels from top level description down to layout and placement. Thus, the usage of top down techniques and tools known from high level synthesis which transfer only the minimum amount of information to the next synthesis level do not lead to efficient solutions compared to typical manual DSP implementations.

In the past, the final handcrafted implementation of DSP designs on the layout level has lead to manual design methodologies on the levels above. Due to market constraints on DSP designs, automation of the design process is unavoidable and tools will be available in the near future. Therefore, algorithms especially suited to solve DSP design problems efficiently are necessary.

This paper deals with an algorithm to relocate storage elements of DSP architectures for low power design. The first algorithms to find an optimal register position of a given signal flow graph has been published by Leiserson *et al.* [1,2] nearly a decade ago. Since their paper in which the term retiming has been coined, a lot of research work has been 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, retiming is formulated as a linear programming problem (LP). The associated matrices have the so called unimodular property such that the simplex algorithm can be applied to compute an optimum retiming solution. A piecewise linear cost function for the simplex algorithm is used to separate level sensitive latches for power reduction in DSP designs. Finally, the gain of the methodology is verified by simulations with the analog model of the considered circuits.

II. The Models

a) The Circuit 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. 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.

a) The Register Model

In many DSP designs level sensitive registers are used which typically consist of two latches. A latch is implemented by a transmission gate to switch off after half a clock period and store the state on the gate capacity on the following inverter. If these latches are placed separately as shown in Fig. 1 the gate capacity of the combinational logic can be used for storage and the inverter can be omitted. In the following a retiming algorithm is derived which enforces this splitting of registers wherever possible.

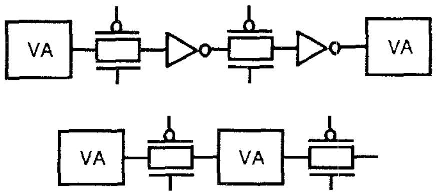

Fig. 1. Splitting of registers.

III. Retiming as a linear programming problem

In this section, retiming is formulated as a linear programming problem. The retiming equation (1) can be written in matrix form [10] with the vector $\mathbf{w}$ and $\mathbf{w}_r$ for the positive register weights: $\mathbf{w}_r - \mathbf{w} = \mathbf{A}^{\mathrm{T}}\ \mathbf{r}$, $\mathbf{w}_r \geq 0$. 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:

$$\mathbf{A}^{\mathrm{T}}\ \mathbf{r} \leq -\mathbf{w}. \tag{2}$$

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:

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:*

$$\sum\nolimits_{\text{loop}} w_r(i,j) = \sum\nolimits_{\text{loop}} w(i,j). \tag{3}$$

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 (3) is obtained. □

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

$$\mathbf{M}\ \mathbf{w}_r \ = \ \mathbf{M}\ \mathbf{w}, \quad \mathbf{w}_r \geq 0. \tag{4}$$

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

Definition. *A matrix is totally unimodular, if 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 (2) and (4) 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 $\boldsymbol{C x} \leq \boldsymbol{b}$, $\boldsymbol{x} \geq 0$ are all with integer components for any arbitrary integer vector $\boldsymbol{b}$, if matrix $\boldsymbol{C}$ 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 the simplex algorithm is generally available e.g. in computer algebra systems like Mathematica as used here.

IV. The algorithm and results

The basic idea to derive an algorithm for register splitting as formulated in Section II is to define costs which depend on the number of latches of each edge. If only one latch is moved to an edge unit costs are computed. If several latches are moved to this edge, the cost of each additional latch is greater than the unit cost. Thus, an algorithm which minimizes this cost function will split the registers wherever possible. The cost function can be formulated as follows:

$$C = \sum_{i=1}^{|E|} c_i; \quad c_i = \begin{cases} w_r & \text{for } w_r(i) \leq 1 \\ \alpha w_r & \text{for } w_r(i) > 1 \end{cases}, \ \alpha > 1$$

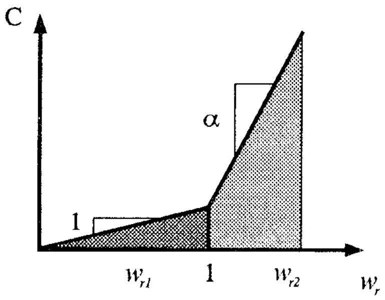

Fig. 2. Cost function for register splitting.

Costs which depend on the the number of registers lead to a non-linear cost function but here we can avoid this non-linearity by adding one edge to each existing edge of the circuit. The old edge w_{r1} and the new edge w_{r2} are connected serial. Unit costs are assigned to the edges w_{r1} whereas higher costs α are assigned to w_{r2}, see Fig. 2. As cost minimization will move all registers to w_{r1}, we limit the number of latches by additional inequalities $w_{r1} \leq 1$. The optimization problem can be formulated in matrix form as follows:

Minimize $\mathbf{1w_{r1}} + \alpha\mathbf{1w_{r2}}$

$$[\mathbf{M},\mathbf{M}]\begin{bmatrix}\mathbf{w_{r1}}\\ \mathbf{w_{r2}}\end{bmatrix} = [\mathbf{M},\mathbf{M}]\begin{bmatrix}\mathbf{w_1}\\ \mathbf{0}\end{bmatrix}, \quad \mathbf{w_{r1}} \leq 1.$$

In order to assign costs to latches with more than one latch per interconnection being high enough, we choose $\alpha \geq \mathrm{Max}\{outdegree(v), indegree(v)\} \cdot |V|$.

If we apply this algorithm to circuits which are used typically in data paths, we obtain a reduction of the power dissipation up to 20 %, see Fig. 3 and 4. Registers in position a) are moved to the inputs. Thus, the paths for glitch propagation have maximum length and we obtain the maximum power dissipation. If we move the registers to position b) the logical depth is reduced which leads to a power reduction due to reduced glitch propagation. If we split the registers and move the latches to positions c) we obtain the lowest power dissipation because the inverters have been omitted and the logical depth is again reduced. We used SPICE simulations to determine the power dissipation with random input patterns. Due to simulation time examples of limited size have been examined.

VI. Conclusion

In this paper, a retiming algorithm for level sensitive registers is derived for the power reduction of DSP designs. The proposed algorithm is based on a non-linear cost function to enforce a suited register distribution. Despite this non-linearity, standard algorithms such as the simplex algorithm or minimum cost flow algorithms can be used. Future work will concentrate on how to integrate this methodology into a general data-path compiler for low power DSP design.

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.

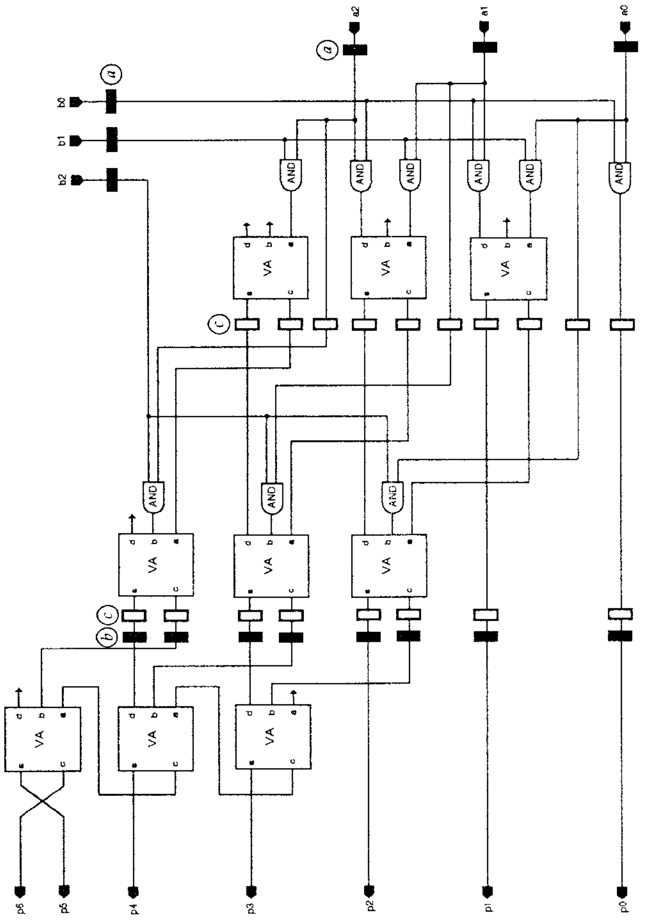

Fig. 3. Different latch postions for a 3-bit-multiplier (circuit1).

Power dissipation of circuit 1 (← $[10^{-4}$ W])

a	b	c
1.41	1.23	1.02

Power dissipation of circuit 2 (↓ $[10^{-4}$ W])

a	b	c
5.21	3.91	3.41

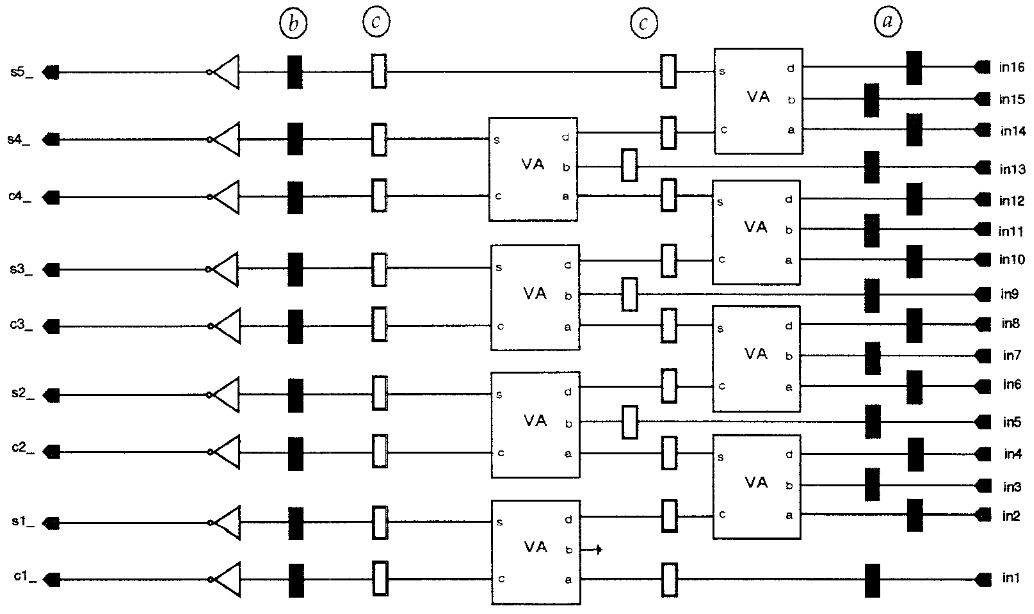

Fig. 4. Different latch postions for a carry save adder (circuit2).