A New Retiming Algorithm for Circuit Design

S.Simon, E. Bernard, M. Sauer, J. A. Nossek
Institute For Network Theory and Circuit Design
Technical University Munich
Arcisstr 21, D-80333 Munich, Germany
(+49)89-2105-8512
simon@nws.e-technik.tu-muenchen.de

ABSTRACT

This paper deals with retiming, a register reconfiguration technique, introduced by Leiserson and Saxe [1, 2], to speed up VLSI circuits. Retiming is generally formulated as an optimization problem which is solvable applying linear-programming algorithms.

This work presents a different way of considering retiming. A loop analysis, related to network theory, is developed to evaluate all possible retiming solutions. Based on the circuit model used in Leiserson's paper, the incidence matrix of the circuit is formulated in order to find the linearly independent loops which are needed to represent all register configurations. Finally, the set of all retiming solutions is efficiently reduced to those, which fulfill design and timing constraints and thus, the designer is able to choose an appropriate one for implementation.

1. INTRODUCTION

An important topic of VLSI design automation is the optimization of timing performance [1, 2, 3, 4, 5, 6, 7]. Due to cost-effectiveness considerations a trade-off has to be made between speed and various parameters, e.g. chip area, regularity, modularity, testability, latency, load of the clock-network and design time [1, 2, 8]. The basic idea to achieve a step towards this goal was introduced by Leiserson and Saxe [1, 2]. Applying a cut-set transfer rule, registers are inserted and deleted in a way, that the data dependencies remain unchanged. Unlike pipelining, this technique, which is called retiming, does not increase circuit latency. In Leiserson's paper polynomial algorithms are given to find a single optimal solution for retiming in regard to one of the following criteria:

- The retiming solution for maximum clock speed.
- The retiming solution for minimum number of registers.
- The minimum cost solution if costs are assigned to each possible register location.

The third version particularly enables a designer to combine retiming with arbitrary parameters. However, the following disadvantages have to be mentioned:

- First of all, the algorithms provide only the optimum solution, whereas solutions nearby the optimum are neglected. In regard to the trade-off, a solution other than the so called optimium may be more advantageous. A designer will probably prefer solutions with maximum usage of already existing cells. This needs not to be the optimum solution concerning speed, chip area or any other parameter.
- It will be difficult to put all aspects of design into a single figure of merit. It would be easier to decide among a set of alternatives, proposed by an algorithm, instead of creating a complicated cost function.

The algorithms presented in this paper provide all retiming solutions with regard to certain constraints. The algorithms can be implemented in a way, that interactive communication between the retiming algorithm and the designer is possible.

The same graph model and circuit, presented in Leiserson's paper, is used for further discussion, see Fig. 1 and 2. A circuit G is characterized by the graph (V,E), the edge weights w and delays d. This can be abbreviated by G=<V,E,d,w>.

Each vertex of Fig. 1 and 2 represents a functional element, containing no timing element, neither a register nor a latch. The worst case of data propagation delay $d(i)$ is associated with each vertex $v_i \in V$. The paths for data transfer are introduced as directed edges $e(i,j) \in E$ from vertex v_i to v_j. The in- and outdegree of every vertex is greater than or equal to one. The number of registers of a data path is defined as the edge weight w.

In [1, 2] an example for speed optimization is given. The circuit and the optimum speed retiming solution is shown in Fig. 1 and 2.

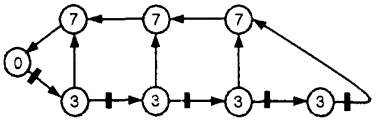

Figure 1: The graph model of example [2],$T_{min} = 24$.

Usually, circuit implementation requires a prescription of specifications which lead to certain constraints for the design. Three constraints are assumed here, in order to deal with a realistic problem:

1. A register should be transferred to the output path e_{out} of the circuit, see Fig. 3 and Fig. 4. (v_8 is the I/O-vertex for data exchange with the system environment,

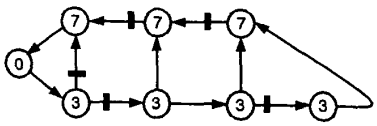

Figure 2: The optimum speed retiming solution of Fig. 1, $T_{min} = 13$.

thus $d(8) = 0$. The vertex numbering is shown in Fig. 5 or [1, 2].)

2. The circuit should operate at clock frequencies higher than 70 MHz ($T_o = 14.3$, assuming that all vertex delays are given in ns.).
3. Cells containing the functional elements of two vertices with the delay 3 and only one register at the horizontal output path (see Fig. 4) already exist as a layout. The use of these cells will reduce design time.

Two retiming solutions fulfilling constraints (1) and (2) are shown in Fig. 3 and 4. Both circuits with $T_{min} = 14$ are only one time unit slower than the optimum in Fig. 2 with $T_{min} = 13$. Faced with these two proposals, a designer

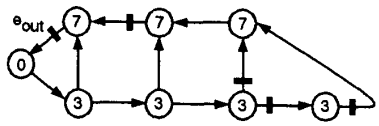

Figure 3: Retiming solution with T_{min}=14

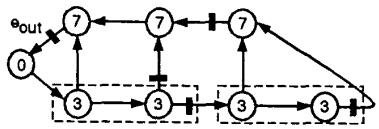

Figure 4: Retiming solution with T_{min}=14

would choose the solution of Fig. 4 because already existing cells are applicable (condition 3).

In general, this paper contributes to retiming algorithms for VLSI design. Interactive algorithms are presented to maximize designers comfort and to reduce design time. Additionally, fundamental analogies to network theory are explored, which are not only useful from a conceptual point of view, but also provide a deeper understanding of retiming.

2. FINDING ALL RETIMING SOLUTIONS

This section presents a circuit loop analysis to evaluate all retiming solutions.

Retiming changes the number of registers in each edge moving registers over nodes. The number of registers after retiming $w_r(i,j)$ for an edge $e(i,j)$ from vertex v_i to v_j is the difference between the number of inserted and deleted registers plus the number before retiming $w(i,j)$. The variable $r(i)$ denotes the number of registers moved over each vertex v_i. Thus:

$$\forall\, e(i,j) \in E: \quad w_r(i,j) = w(i,j) + [r(i) - r(j)]. \quad (1)$$

This equation is identical with (2) in [1, 2]. The system of linear equations (1) can be written in matrix form $\mathbf{w} = \mathbf{A}^{\mathrm{T}}\mathbf{r}$. Matrix A is identical with the incidence matrix well known from circuit theory and r can be identified with the electric potential. Usually, in circuit analysis one reference node is chosen to achieve a unique potential. This does not reduce the generality of the solution as far as measurable quantities are concerned. Due to circuit theory Rank (A) is only $|V|-1$; thus, only $|V|-1$ variables r have to be determined. A general proof and a more detailed examination of this relationship is given in [9]. By summing up all Equations (1) belonging to a directed cycle, we obtain:

$$\sum_{\text{cycle}} w_r(i,j) = \sum_{\text{cycle}} w(i,j). \quad (2)$$

Equation (2) states that the number of registers of a directed cycle or loop is constant during retiming. Similar to circuit theory $|E| - (|V| - 1)$ linearly independent equations in the form of (2) are associated with G because the rank of matrix A is $|V| - 1$ and $|E|$ equations of form (1) can be formulated. Therefore, it must be possible to find $|E| - (|V| - 1)$ equations (2) by linear combination, see [9].

It is important to notice that (2) does not contain the unknowns $r(i)$. We obtain a set of linearly independent cycles form the incidence matrix A if we replace all entries of one column by zero according to the reference node selection and add each row with only one entry to another until all rows contain only one 1 or -1. If a column contains two non-zero entries the corresponding sums of registers are identical and a cycle equation of form (2) is obtained.

As $|E| - (|V| - 1)$ equations of (2) use $|E|$ variables $w_r(i,j)$, there is a set of $|V| - 1$ independent $w_r'(i,j)$ and a set of $|E|-(|V|-1)$ linearly dependent $w_r''(i,j)$. The choice of linearly dependent and independent variables is arbitrary. Let us define a certain subclass of graphs $< G_s >$.

DEFINITION: $< G_s >$ is the set of all graphs, for which it is possible to choose one linearly dependent variable, present in only one of the Equations (2).

Thus, the linearly dependent variables can be eleminated using the inequalities $w_r(i,j) \geq 0$ for positive register weights:

$$w_r'(i,j) \geq 0 \quad (3)$$

$$\sum_{\text{cycle}} w_r'(i,j) \leq \sum_{\text{cycle}} w(i,j). \quad (4)$$

These equations can be solved straightforwardly by applying simple nested-loops, see [9]. In order to deal with Leiserson's example, the following abbrevations are made to simplify the discussion: $w_r'(i,j) = x_i \wedge w_r''(i,j) = y_i$, see Fig. 5.

Equation (4) results in 4 equations according to 8 vertices and 11 edges.

$$\begin{aligned} x_7 + x_8 &\leq 1 \\ x_1 + x_6 + x_7 + x_8 &\leq 2 \\ x_1 + x_2 + x_5 + x_6 + x_7 + x_8 &\leq 3 \\ x_1 + x_2 + x_3 + x_5 + x_6 + x_7 + x_8 &\leq 4. \end{aligned} \quad (5)$$

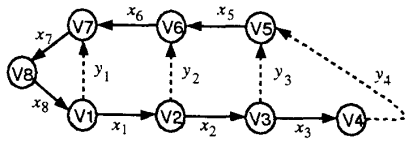

Figure 5: Linearly dependent variables y_i and independent variables x_i.

The evaluation of this system delivers 143 retiming solutions. As mentioned above, the linearly dependent variables y_i are determined by (2). For every G $\in <G_s>$ a directed tree can be found, see [9]. If G does not belong to $<G_s>$, it is not possible to choose one linearly dependent $w_r(i,j)$ for each of the equations in (2). Thus, the solution is more complicated because (2) has to be treated as a diophantine system.

3. CONSTRAINTS FOR RETIMING

Section 2 presented a technique to evaluate all retiming solutions, but a designer will neither be interested in a complete solution set nor will computation time and hardware resources allow him to consider all of them. Due to the trade-off mentioned in the introduction, constraints reduce the set of solutions drastically. This section investigates how constraints like modularity, local interconnections and minimum clock frequency can be applied.

Generally, constraints can be divided into two groups, namely structural and timing constraints. Structural constraints only deal with the variables $w_r(i,j)$, whereas the timing constraint take also vertex delays $d(i)$ into account.

A subset of the structural constraints results in a register number greater than or equal to an integer $l(i,j)$ instead of $w_r(i,j) \geq 0$.

$$w_r(i,j) \geq l(i,j) \tag{6}$$

The abbrevations $W(m), L(m)$ and $\tilde{W}(m)$ are used as follows:

$$W(m) = \sum_{\text{cycle}} w(i,j) \qquad L(m) = \sum_{\text{cycle}} l(i,j)$$

$$\tilde{W}(m) = W(m) - L(m) \tag{7}$$

Obviously, a substitution $\tilde{w}_r(i,j) = w_r(i,j) - l(i,j)$ does not change (4) in structure:

$$\tilde{w}_r'(i,j) \geq 0 \tag{8}$$

$$\sum_{\text{cycle}} \tilde{w}_r'(i,j) \leq \tilde{W}(m). \tag{9}$$

If $L(m)$ is close to $W(m)$ we achieve a huge decline in the number of solutions because W has to be replaced by $\tilde{W}$.

If structural constraints are too restrictive, then $L(m)$ is greater than $W(m)$, (9) has a negative upper bound and there is no solution. If too few constraints are given an interactive algorithm proposes additional constraints, e.g. referring to an optimum solution found with algorithms of [1, 2]. The most important structural constraints are listed in the following paragraph.

3.1. Structural Constraints:

3.1.1. Local Data Communication. Broadcast data can be avoided by pipelining. A certain minimum number of registers $l(i,j)$ is necessary for an edge $e(i,j)$ representing a broadcast interconnection:

$w_r(i,j) \geq l(i,j)$.

3.1.2. Fixed Registers at Certain Places. Usually, a designer wants to have some registers at certain places. For example, if registers at the data output path are necessary, or if already existing cells or a library, with incorporated registers are available:

$w_r(i,j) \geq 1 \vee w_r(i,j) = 1$.

3.1.3. Modularity. Vertices of identical functions should have the same surrounding register configuration. Thus, the vertex and its registers can be implemented as one unique cell. If edge $e(k,l)$ corresponds to edge $e(i,j)$ the number of unknows is reduced:

$w_r(i,j) = w_r(k,l)$.

3.1.4. Load of the Clock Network. If we assume the capacitive load of the clock network being proportional to the number of registers a maximum permissable clock load can be introduced by:

$\sum w_r(i,j) \leq$ Maximum-Load.

As mentioned in the introduction, two constraints are applied to the example circuit. These constraints can be formulated as inequalities. Firstly, a register should be transferred to the output path, $x_7 \geq 1$. Secondly, a modular design which considers already existing cells can be formulated as: $(x_1 = x_3) \wedge (x_2 = y_4)$.

$$x_7 = 1 \quad x_8 = 0 \quad x_i \geq 0 \quad x_5 = 3 - 2x_1 - 2x_2 - x_6$$
$$x_1 + x_2 \geq 1 \quad x_1 + x_6 \leq 1 \quad 2x_1 + 2x_2 + x_6 \leq 3 \tag{10}$$

The number of all retiming solutions under both constraints are given in Table 1.

Constraints	NC	Out	Mod	Mod & Out
No Timing Constraint	143	30	14	3
$f \geq 70MHz$ $T \leq 14.3$	51	7	6	1
$f = 76.9MHz$ $T = 13$	4	0	2	0

Table 1: Number of retiming solutions under various constraints.

The following abbrevations are used: NC = no structural constraint, Out = a register in the output path, Mod = modularity.

3.2. Timing Constraint

As mentioned before, specifications deliver a minimum clock frequency $T \leq T_o = 1/f_{min}$, denoted as the timing constraint.

A cirtical path with a path delay greater than T_o must contain one register minimum. For our example, a register must be between either vertex $v6$ and $v7$ or $v5$ and $v6$ because $d(V5) + d(V6) + d(V7) = 21 \geq T_o = 14.3$. This can formulated as: $x_5 + x_6 \geq 1$.

We have to find all critical paths from each vertex as the root. Therefore $O(|V|)$ equations have to be considered. Only one solution is left if timing and structural constraints are applied for our example: $x_1 = 0$; $x_2 = 1$; $x_3 = 0$; $x_5 = 1$; $x_6 = 0$; $x_7 = 1$; $x_8 = 0$. A specific C-program has been implemented for this example. The computation time was only 45% compared to Leiserson's Algorithm OPT2 [1, 2] to evaluate a solution with $T = 14$. Generally, the computation time depends intensively on the number of solutions. If the algorithm needs too much time for evaluation further constraints should be introduced.

In addition to the ability to find all retiming solutions for a certain clock period and for specific structural constraints, an algorithm to find all maximum speed solutions can be formulated as follows:

Algorithm SOAR (compute the set of all retiming solutions with maximum clock speed).

Step 1. Find all equations (4).

Step 2. Evaluate all possible clock periods, applying e.g. the method of Leiserson [1, 2].

Step 3. Binary search among all possible clock periods:

Step 3.1. Find all critical paths using the clock period selected by Step 3.

Step 3.2. Check, whether or not a retiming solution is left using the Equations of Step 1 and Step 3.1.

Step 3.3. If no solution for the next lower clock period is left, but the actual clock period still has solutions, the optimum is found. Else, return to Step 3.

Step 4. Find all solutions of the last supposed clock period.

Applying this algorithm to our example, 4 high speed solutions with $T = 13$ are found, see Fig. 6. The designer is now able to choose one of them for implementation.

If structural constraints should be considered too, they can be applied in Step 3.1. of the algorithm SOAR.

4. CONCLUSION

A new algorithm to evaluate all retiming solutions with regard to conditions for certain register locations as well as timing constraints has been described. Based on an analogy to network theory, a loop analysis was developed to represent all register configurations. The computation time depends strongly on the number of constraints applied to the retiming process because the algorithm provides all possible retiming solutions. An obvious advantage, compared to other retiming algorithms [1, 2] is that a designer can formulate and program his specific problem within a few lines.

The algorithms presented so far can be incorporated into both circuit compilers and interactive design tools. As retiming is a valuable technique to produce efficient circuits, future work will deal with an efficient implementation with regards to other algorithms of VLSI synthesis and design automation such as compacting, routing and placing.

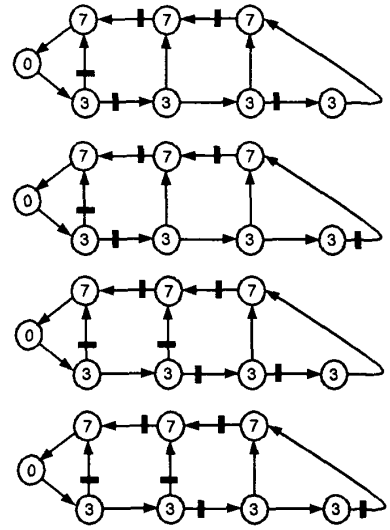

Figure 6: All retiming solutions with $T = 13$.

REFERENCES

[1] C.E. Leiserson and J. B. Saxe. Optimizing Synchronous Circuitry and Retiming . In *Proceedings of the 3rd Caltech Conference on VLSI*, pages 5-35, 1983.

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

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

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

[5] T. Soyata, E. G. Friedman, and J. H. Mulligan. Integration of Clock Skew and Register Delays into a Retiming Algorithm. In *Proceedings of the International Symposium on Circuits and Systems*, pages 1483-1486, 1993.

[6] J. Krammer. Optimal Retiming of Regular Processor Arrays . *Archiv fuer Uebertragungstechnik*, 44(2):97-103, 1990.

[7] S. Dey A. Parker Z. Iqbal, M. Potkonjak. Critical Path Minimization Using Retiming and Algebraic Speed-Up. In *DAC*, 1993.

[8] S. Malik and E. M. Sentovich. Optimizing sequential networks with combinatorical techniques. *IEEE Transactions on Computer-Aided Design*, 10:74-84, 1991.

[9] S. Simon. Circuit Loop Analysis: A New Approach To Retiming. Technical Report TUM-LNS-TR-93-10, Technical University Munich, 1993.