TIGHT COUPLING OF TIMING-DRIVEN PLACEMENT AND RETIMING

Ingmar Neumann *Wolfgang Kunz*

University Frankfurt/Main
Department of Computer Science
Electronic Design Automation Group
60054 Frankfurt/Main, Germany

ABSTRACT

Retiming is a widely investigated technique for performance optimization. In general, it performs extensive modifications on a circuit netlist, leaving it unclear, whether the achieved performance improvement will still be valid after placement has been performed. This paper presents an approach for integrating retiming into a timing-driven placement environment. The experimental results show the benefit of the proposed approach on circuit performance in comparison with design flows using retiming only as a pre- or post-placement optimization method.

1. INTRODUCTION

In the development of high performance circuits, circuit speed can be considered the most important single optimization criterion. Therefore, a lot of methods for minimizing cycle time have been developed. A powerful technique, proposed by Leiserson and Saxe [1][2], is retiming, which relocates registers while preserving the functionality of a circuit. Leiserson and Saxe developed algorithms both for cycle time minimization and for register area minimization of circuits with edge triggered flip flops. Since then, many improvements and extensions to the original ideas have been developed, like acceleration techniques [3], which dramatically speed up execution time, algorithms for retiming level clocked circuits [4][5], algorithms taking registers setup and hold times into account [6][7], algorithms for retiming registers with enable inputs [8] as well as algorithms that can improve testability [9].

The original algorithm developed by Leiserson and Saxe finds a retiming for a circuit such that a given cycle time is met if such a retiming exists, in polynomial time. It is based on a simple timing model which assumes gate delays to be load independent. Unfortunately, for CMOS technology which is the most widely used technology today, this model is not accurate enough as gate delays cannot be considered to be load independent and retiming registers may change the loads of gates. The advent of deep-submicron technologies exacerbated the situation further by increasing the influence of wire length on the total delay. Loads resulting from wires are affected by retiming even more than loads resulting from gate inputs and, above all, are not known before placement.

In [10]-[12], more sophisticated timing models are used to incorporate wire delays. However, the algorithms using these models suffer from long run times. Additionally, all load changes of wires connected to a particular cell that would result from retiming the cell must be known prior to the retiming process. In practice, those changes are hard to predict exactly. Further, from a physical point of view, retiming means to remove some cells from the placement, leaving gaps, and to insert other cells at other locations. Making the placement legal by removing cell overlapping will shift cells and change the lengths of nets which are not directly affected by the retiming process. These effects make it impossible to predict whether the optimum solution produced by the retiming algorithm will still be valid after placement has been performed. In the worst case, retiming can even decrease circuit speed.

An approach to overcome these problems was presented in [13]. After performing a conventional placement and routing, an optimization loop consisting of wire length estimation, retiming, and register placement is entered. Even though this approach produces promising results and ensures that retiming will not deteriorate cycle time, it does not fully exploit the potential of coupling placement and retiming. A placement algorithm tries to optimize the placement with regard to a given netlist topology which is modified significantly by retiming. Especially, a timing-driven placer will aggressively try to shorten wires on critical paths, while paying less attention to less critical wires. This can lead to a balance of path lengths, reducing the optimization potential for retiming.

To overcome these difficulties and to be able to take more advantage of integrating retiming into placement, we propose a much tighter coupling between placement and retiming in this paper. Our approach does not use retiming for a post-placement optimization, but employs it as an optimization technique throughout the whole placement process.

2. TIGHT COUPLING OF PLACEMENT AND RETIMING

2.1 Overview

The core of our approach is a timing-driven simulated annealing-based standard cell placement algorithm using dynamic temperature control and dynamic critical net weighting. At each temperature level, first, load capacitances are calculated from net length estimations and a static timing analysis is performed. If timing constraints are not met, a retiming based cycle time optimization step is performed, based on the previously calculated capacitance values trying to meet the constraint or, if this was not possible, at least to improve circuit performance. Afterwards, the newly created registers are inserted into the placement using a

0-7803-6685-9/01/$10.00©2001 IEEE

very fast placement approach, and the cycle time is calculated again. If timing constraints are met now, or at least an improvement has been achieved, the new configuration is accepted, otherwise it will be rejected. Afterwards, net weights are recalculated and the placer begins another iteration. Figure 1 gives an overview of the placement procedure at a particular temperature level.

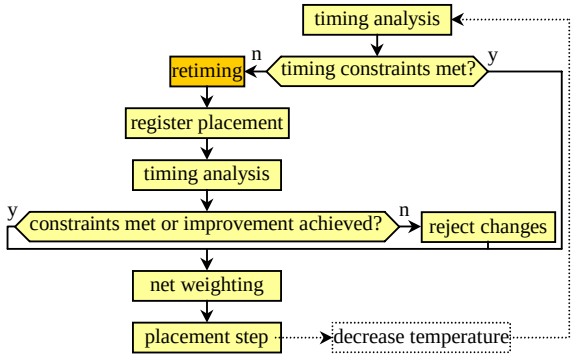

Figure 1. **Placement at a particular temperature level**

2.2 Retiming

To be able to retime circuit netlists with tens or even hundreds of thousands of gates, very fast algorithms are needed. Especially, when retiming is used as an optimization step, which is performed numerous times in a timing-driven placement environment, it is not sufficient to have algorithms of polynomial complexity, but near-linear complexity algorithms are required. For this reason we use the original FEAS algorithm [2], extended by an acceleration technique similar as [13].

2.3 Register Placement

In general, a simulated annealing-based placer will be able to find good positions for the newly created registers, independent from their initial position. But this process will take a lot of time if these initial positions are chosen randomly, making it impossible to verify immediately after the registers have been inserted whether or not a cycle time improvement has been achieved. Furthermore, it can save a lot of effort for the placer, if the new registers are inserted at “reasonable” locations, especially at low temperatures, when cells aren’t allowed to make large jumps.

Therefore we use a separate register placement step to provide the timing analyzer quickly with realistic assumptions about the wire lengths after retiming has been performed. For each new register a position is determined such that the sum of the lengths of the nets connected to this register is minimized. In many situations, the result will not be a particular vertex, but a target area of rectangular shape. In the latter case, we look for the most suitable cell gap inside this area and position the register there. This helps to keep the modifications of the original placement as small as possible. If the gap isn’t large enough, neighbor cells are pushed aside first. By doing so it is always guaranteed that no cell overlapping occurs. At this point, no further work is done to reuse gaps left by deleted registers, also no work is done to balance the total row length, because these tasks are performed by the simulated annealing placer later. An example of inserting a single additional register is shown in Figure 2.

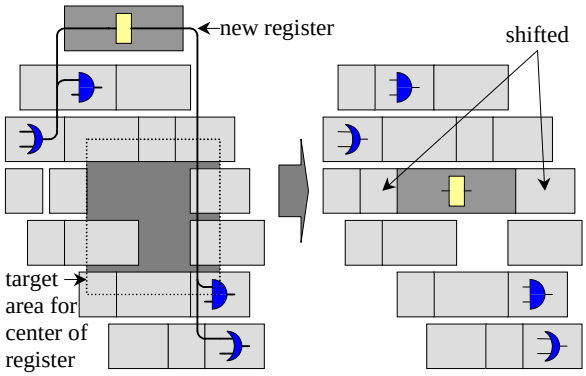

Figure 2. **Single register placement example**

2.4 Repeated Check of Retiming and Placement

After retiming has been performed and the newly introduced registers have been added to the placement, wire lengths are estimated again, and timing analysis is repeated to check whether cycle time really has been improved. At a first glance this check may seem unnecessary, because retiming-based cycle time optimization of a synchronous network should at least not deteriorate the cycle time. However, in our experiments it turned out that when the effect of retiming on the placement is taken into account, it is indeed possible that a retiming step increases cycle time. The first reason for this is that retiming may increase the number of registers in a circuit, sometimes dramatically. In typical standard cell libraries, flip flops and latches have a far greater area requirement than simple logic cells. So already small increases in register count may result in a significant increase in area requirement. This leads to longer wires and may offset the performance gain achieved by retiming. The second reason is that modifying placement changes the positions of cells and the lengths of nets, which are not directly affected by retiming. These effects cannot be taken into account by the retiming algorithm and may also result in a cycle time deterioration.

Therefore, a newly retimed configuration will only be accepted, if immediately after the registers have been inserted a performance improvement has been achieved, otherwise the modifications of placement and netlist will be rejected.

3. EXPERIMENTAL RESULTS

We have implemented all components of the placement environment in C++ and linked them together into one single application. The experimental results have been obtained on a Sun Ultra Sparc 5 Workstation.

The main aim of our work was to investigate the optimization potential of a tight coupling of timing-driven placement and retiming. Therefore, a comparison of three different design flows is of interest:

- A conventional design flow consisting of retiming a logic netlist, followed by timing-driven placement
- Timing-driven placement, followed by a single retiming step using the delay values calculated from the final placement. After performing a register placement step as described in chapter 2.3, additionally some placement steps at very low temperatures are performed to achieve uniform row lengths again.
- A tight coupling of retiming and placement as described in this paper.

For our experiments we mapped the larger circuits of the ISCAS-89 benchmark set [14] onto a 0.18 µm standard cell library. The properties of these circuits are shown in Table 1. Column 2, 3, and 4 contain the total number of cells, the number of nets, and the number of registers for each circuit. Column 5 contains the minimum cycle time (c.t.), expressed in nanoseconds, when wire loads are ignored.

circuit	#cells	#nets	#FF	c.t.
S1423	731	749	74	7.93
S1488	659	668	6	3.22
S1494	653	662	6	3.29
S5387	2958	2994	179	2.52
S9234	5825	5845	228	5.83
S9234.1	5805	5804	269	5.83
S13207	8620	8652	669	6.71
S13207.1	8589	8652	638	6.45
S15850	10369	10384	597	8.54
S15850.1	10306	10384	534	8.27
S35932	17793	17829	1728	3.15
S38584	20705	20718	1452	5.49
S38584.1	20679	20718	1426	6.72
S38417	23815	23844	1636	5.19

Table 1. Used ISCAS-89 benchmarks

The performance results of our experiments are shown in Table 2. Column 2 contains the achieved cycle time for a conventional timing-driven placement approach without any application of retiming. A comparison with the cycle time values from Column 5 of Table 1 shows a portion of wire delay on the total delay of 26%-70%. Then, for each of the previously described design flows which use retiming, the achieved cycle time in nanoseconds and the total number of registers are shown. Columns 3 and 4 contain the results for pre-placement retiming, columns 5 and 6 contain the results for post-placement retiming, and columns 7 and 8 show the results for the approach presented in this paper.

The wire lengths values used for the final cycle time calculation have been estimated for each net by using the semi-perimeter of the bounding rectangle times a correcting factor depending on the number of terminals and the length to width ratio of the bounding box.

circuit	none	pre-placement		post-placement		tight coupling	
	c.t.	c.t.	#FF	c.t.	#FF	c.t.	#FF
S1423	12.4	10.6	113	10.7	112	10.6	114
S1488	4.39	4.53	7	4.39	6	4.30	6
S1494	4.46	4.53	7	4.46	6	4.45	6
S5387	3.95	4.37	325	3.95	179	3.85	348
S9234	10.2	7.31	268	7.08	249	6.26	462
S9234.1	10.3	7.34	269	7.66	242	7.36	553
S13207	10.0	9.31	950	10.0	669	8.30	943
S13207.1	10.5	9.79	640	9.38	641	9.18	641
S15850	15.3	12.5	962	15.3	597	12.6	3355
S15850.1	14.8	13.4	586	11.4	610	10.3	659
S35932	10.6	10.4	2826	10.5	2193	8.98	2841
S38584	17.0	19.0	3379	17.0	1452	16.2	3330
S38584.1	14.1	13.0	1428	13.4	1428	12.8	1429
S38417	15.2	12.5	2006	10.7	2193	10.2	2479

Table 2. Performance results and register counts

The experimental results show that in most cases applying retiming only before placement achieved the smallest performance improvement of all strategies. In a few cases cycle time was even larger after placement. If retiming was applied once after placement we achieved somewhat better results, and in no case there was an increase of cycle time. However, this approach was outperformed by our new approach using tight coupling, which produced equal or better results for each benchmark. Table 3 gives a summarizing overview of the approaches by comparing the achieved cycle time improvements.

improvement	pre-placement retiming	post-placement retiming	tight coupling
minimum	-11.8%	0%	+0.02%
maximum	+28.7%	+31.2%	+38.6%
average	+6.7	+9.98%	+15.9%

Table 3. Achieved improvements

Table 4 contains the CPU run times in seconds for a conventional timing-driven placement without retiming and for a tight coupling of placement and retiming. The results show that the increase in run time caused by integrating retiming is moderate. Despite the fact, that retiming is performed numerous times, for the majority of the benchmarks the overall run time of our approach is still dominated by the simulated annealing-based placer core.

circuit	placem. only	tight coupling	circuit	placem. only	tight coupling
S1423	100	104	S13207.1	5747	7598
S1488	115	126	S15850	7073	18102
S1494	115	125	S15850.1	7665	9721
S5387	1066	1591	S35932	29107	48365
S9234	3227	4014	S38584	22249	46323
S9234.1	3148	4091	S38584.1	21518	33945
S13207	5464	9075	S38417	37614	52284

Table 4. CPU run times

To illustrate the importance of taking the increase in register area during retiming into account as described in chapter 2.4, we additionally conducted some experiments using our new approach

without checking the result after register placement. A new configuration produced by the retiming algorithm is accepted in every case. The results given in table 5 show an increase in cycle time of 6% and an increase of register count of 34% on average, if review was disabled.

circuit	cycle time review after register placement			
	enabled		disabled	
	c.t.	#FF	c.t.	#FF
S1423	10.6	114	10.5	128
S1488	4.30	6	4.52	9
S1494	4.45	6	4.70	9
S5387	3.85	348	3.82	452
S9234	6.26	462	6.24	472
S9234.1	7.36	553	7.88	469
S13207	8.30	943	8.31	931
S13207.1	9.18	641	9.99	948
S15850	12.6	3355	14.1	6034
S15850.1	10.3	659	10.7	660
S35932	8.98	2841	9.93	2846
S38584	16.2	3330	17.7	3872
S38584.1	12.8	1429	14.7	3371
S38417	10.2	2479	11.5	3631

Table 5. Results with and without review after register placement

4. CONCLUSION

A new approach for integrating retiming into the physical design process has been proposed. Instead of using retiming as a pre- or a post-placement optimization method, it is applied as a cycle time improvement technique throughout the whole placement process. At each temperature level of a simulated annealing-based standard cell placement algorithm, a retiming step is performed first. Immediately after inserting new registers using a fast insertion technique, cycle time is checked to ensure that no performance deterioration due to area increase occurs. By using an efficient retiming approach it is ensured that, despite the fact that retiming is performed very often, the overall run time of the approach is still dominated by the placement algorithm. The experimental results show that this approach exploits the optimization potential of coupling retiming and placement significantly better than applying retiming only before or after placement. The results also show that an increase in register count which can result from a retiming step cannot be ignored, not only because of the increase in area but also for performance reasons.

5. REFERENCES

[1] Leiserson C., Saxe B., “Optimizing Synchronous Systems”, Journal of VLSI and Computers Systems, pp. 41-67, 1983.

[2] Leiserson C., Saxe B., “Retiming Synchronous Circuitry”, pp.5-35, Algorithmica 6(1) 1991.

[3] Shenoy N., Rudell R., “Efficient Implementation of Retiming”, Proc. ICCAD-94, pp. 226-233, 1994

[4] Lockyear B., Ebeling C., “Optimal retiming of level-clocked circuits using symmetric clock schedules”, IEEE Transactions on Computer Aided Design, vol. 13, no. 9, pp. 1097-1109 1994.

[5] Ishii A., Leiserson C., Papaefthymiou M., “Optimizing two-phase, level-clocked circuitry”, Advanced Research in VLSI and Parallel Systems, Proc. of the 1992 Brown/MIT Conference, pp. 246-264, 1992

[6] Papaefthymiou M., “Asymptotically Efficient Retiming Under Setup and Hold Constraints”, Proc. ICCAD-98 pp.288-295, 1998

[7] Sundararajan V., Sapatnekar S., Parhi K., “MARSH: Min-Area Retiming with Setup and Hold Constraints”, Proc. ICCAD-99, pp. 2-13, 1999

[8] Eckl K., Madre J., Zepter P., Legl C., “A Practical Approach to Multiple-Class Retiming”, Proc. DAC-99, pp. 237-242, 1999

[9] El-Maleh A., Marchok T., Rajski J., Maly W., “Behavior and Testability Preservation Under the Retiming Transformation”, IEEE Transactions on Computer Aided Design, vol. 16., no. 5, pp. 528-542, 1997

[10] Soyata T., Friedman E., “Retiming with Non-Zero Clock Skew, Variable Register, and Interconnect Delay”, Proc. ICCAD-94, pp. 234-241, 1994

[11] Soyata T., Friedman E., Mulligan J., “Incorporating Interconnect, Register, and Clock Distribution Delays into the Retiming Process”, IEEE Transactions on Computer Aided Design, vol. 16, no. 1, pp.105-120, 1997

[12] Lalgudi K., Papaefthymiou M., “DELAY: An Efficient Tool for Retiming with Realistic Delay Modeling”, Proc. DAC-95, pp. 304-309, 1995

[13] Tien T. et al., “Integrating Logic Retiming and Register Placement”, Proc. ICCAD-98, pp. 136-139, 1998

[14] Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http://www.cbl.ncsu.edu