Experiments in the Iterative Application of Resynthesis and Retiming

Soha Hassoun and Carl Ebeling
Department of Computer Science and Engineering
University of Washington, Seattle, WA
{soha,ebeling}@cs.washington.edu

July, 1997
Submitted for TAU97

Abstract

Many attempts have been made to combine some form of retiming with combinational optimization techniques to improve the performance of sequential circuits. To achieve improvements, registers are shifted to expose different and sometimes larger combinational blocks for resynthesis. A simple, yet unexplored, sequential optimization consists of iteratively applying retiming and resynthesis. Retiming changes the boundaries of the combinational blocks in the circuit, and resynthesis changes the delays. It is only logical to repeat this procedure until the circuit performance converges. In this paper we evaluate a set of experiments based on the iterative application of resynthesis and retiming. For a set of the original MCNC benchmark circuits, we show that with a total of 18 alternating resynthesis and retiming steps, we obtain an average of 37% reduction in the clock period at an area increase of 5%. This improvement translates to an 18% further reduction in clock period over a single step of synthesis followed by one step of retiming. We consider the effects of (a) varying the number of iterations, (b) starting with an initial retiming step instead of a resynthesis step, and (c) retiming without register minimization. We also describe the limitations of the iterative procedure in improving the performance of circuits with single-register cycles.

Contact Author:

Soha Hassoun
Department of Computer Science and Engineering, Box 352350
University of Washington, Seattle, WA 98195-2350
Fax: (206)543-2969.
Phone: (206)543-5143.
e-mail: soha@cs.washington.edu

Experiments in the Iterative Application of Resynthesis and Retiming

Soha Hassoun and Carl Ebeling
Department of Computer Science and Engineering
University of Washington, Seattle, WA

1 Introduction

There has been considerable research into making retiming [9] a more practical algorithm. There are now efficient implementations for retiming to achieve the minimum clock period [16], and to achieve the minimum number of registers under clock period constraints [12]. Other recent research in retiming includes incorporating more accurate delay models [8], and dealing with state equivalence with the original circuit [4]. There is also recent work on using retiming at a higher-abstraction level [14], and earlier in the design process [11]. Significant progress has also been made for level-clocked circuits [10, 7].

The concept of retiming has also been instrumental in using combinational optimization techniques for the optimization of sequential circuits at the structural level. It is intuitive to try to use some form of retiming to change the boundaries of combinational stages, and then apply existing combinational optimization techniques to different, and sometimes larger, combinational blocks. The register movements may be large, such as in peripheral retiming [13] which attempts to expose the whole circuit for combinational resynthesis. The register movement could also be restricted to a portion of the circuit. For example, Dey et al. identify subcircuits with equal-weight reconvergent paths (petals) to which peripheral retiming can be applied [3]. Precomputation-based architectural retiming, where a signal is computed one cycle earlier using the combinational logic in the previous pipeline stage, is another technique that involves retiming with logic duplication of a portion of the circuits [5, 6]. Sequential optimization techniques based on more fine-grain register movements include applying local algebraic transformations across register boundaries [2], and implicit retiming, where a restricted form of precomputation is applied to specific kernels in the circuit [1]. The more common use of retiming in conjunction with synthesis, however, has been to perform sequential optimization, technology mapping, and follow that with retiming to achieve the desired performance.

The experiments presented here are based on a simple, yet unexplored, sequential optimization technique consisting of iteratively applying retiming and resynthesis. Retiming changes the boundaries of the combinational blocks in the circuit while resynthesis changes the delays. It is only logical to repeat this procedure until the circuit performance converges.

In this paper we investigate the effectiveness of the iterative application of retiming and resynthesis. In Section 2 we describe the details of the experiment. In Section 3 we present the results, and analyze the limitations of this technique. We then examine some variations of the experiment. In Section 4 we vary the number of retiming and synthesis steps and show our example circuits cannot further benefit from the process. In Section 5 we consider some secondary effects on the iterative process. More specifically, we consider the effects of starting the iterative process with a retiming step instead of a synthesis step, and also the effects of retiming without minimizing the register number of registers. We conclude with some comments on the run times of the experiments, and a discussion of challenges in sequential optimization.

Circuit	# inputs	# outputs	# nodes	# registers
mm4a	7	4	35	12
s208.1	10	1	104	8
s27	4	1	10	3
s298	3	6	119	14
s349	9	11	161	15
s386	7	7	159	6
s400	3	6	162	21
s444	3	6	181	21
s641	20	14	379	19
s713	20	13	393	19

Table 1: Circuits from the MCNC benchmarks used in the experiments.

2 Experiment

Our set of benchmark circuits was selected from the MCNC sequential multilevel circuits. The circuits were available in blif format, which can be read the Berkeley SIS logic optimization program [15]. The reference, or initial, circuit in each experiment was obtained by first decomposing all the nodes into 2-input OR gates (tech_decomp -o 2), and then performing technology mapping to respect the driving load assuming all arrival and required times are zero (map -n 1 -AFG -p). We used the lib2.genlib and lib2_latch.genlib libraries for mapping in the initial circuit and also in reporting the results of the experiments.

In the experiment, there is a resynthesis step, and a retiming step. The *resynthesis step* was performed by first running the SIS script script.rugged, which is recommended for area minimization. This script was followed by script.delay without the redundancy removal command[1] which synthesizes a circuit for a final implementation that is optimal with respect to speed. In the *retiming step*, we minimized the clock period while minimizing the number of registers (retime -mi).

The experiment we conducted was straightforward. We alternated resynthesis and retiming steps beginning with a resynthesis step. The goal of this experiment was to measure the effectiveness of the iterative procedure in minimizing the clock period while also minimizing the register and logic area. The initial circuit is labeled as step number 0, and each subsequent step is labeled accordingly. Thus, we performed a total of 30 steps – that is 15 resynthesis steps interleaved with 15 retiming steps.

Table 1 summarizes the circuits we used in our experiments. We list the number of primary inputs, primary outputs, logic functions (.names in blif format), and registers. From the LGsynth91 sequential multilevel circuits, we report the results of those that completed at least 6 iteration steps for this experiment and its variations that are presented in Section 4 and in Section 5.

3 Experimental Results

We first summarize the results of the experiment. We then examine more closely the results for one of the circuits, and comment on the results in general.

[1] SIS was unable to perform redundancy removal reliably.

	reference circuit		resy-ret		iterative procedure				
					normalized to reference		normalized to resy-ret		step
Circuit	T_{clk}	Area	T_{clk}	Area	T_{clk}	Area	T_{clk}	Area	It
mm4a	28.50	410176	0.81	0.62	0.61	0.79	075	1.28	30
s208.1	10.66	138272	1.03	0.93	1.00	1.00	0.97	1.08	0
s27	7.44	30160	0.52	0.92	0.52	0.92	1.00	1.00	2
s298	11.14	223648	0.91	0.89	0.61	1.32	0.67	1.49	16
s349	13.22	238496	0.98	1.38	0.87	1.73	0.89	1.26	12[a]
s386	15.83	262624	0.83	0.89	0.66	0.88	0.79	0.99	27
s400	15.23	309488	0.73	1.04	0.56	1.16	0.77	1.12	12
s444	15.94	322944	0.73	0.97	0.55	1.65	0.75	1.70	24
s641	20.43	341504	0.82	1.04	0.71	1.13	0.87	1.08	7[b]
s713	20.53	348464	0.84	0.99	0.72	1.09	0.85	1.10	8[c]

[a]Maximum iteration completed is 14
[b]Maximum iteration completed is 10
[c]Maximum iteration completed is 10

Table 2: *Results of applying the iterative resynthesis and retiming on some MCNC benchmarks*

In Table 2 we report the changes in the clock period and area. The leftmost column lists the names of the circuits. The next two columns report the clock period (in time units) and area for the reference circuit. The next two columns labeled resy-ret report the clock period and area, both normalized to the reference circuit, after one step of resynthesis and retiming, which is typical of what has been used so far in sequential logic optimization. The next two columns report the minimum clock period and the corresponding area obtained through the iterative procedure. They are both normalized to the clock period and area of the reference circuit. The following two columns present the same information, but normalized to results of the resy-ret step. The final column lists the number of the step during which the minimum clock period was achieved. Section 4 provides detailed geometric mean calculations based on the number of iterations completed.

Figure 1 shows the results of our experiment applied to circuit s386. The performance improvement here is close to the average results for the test circuits. The lowest clock period, however, is obtained later in the iteration process when compared to the other circuits. Figure (a) shows the clock period normalized to the clock period of the initial circuit. The left most column thus has a clock period of 1. In the next step, the clock period is reduced by synthesis. The following retiming step further reduces the clock period. In the rest of the steps, it is typical that resynthesis increases the clock period, while retiming reduces it. The resynthesis step is supposed to optimize for delay. However, since no target clock period is specified and the `reduce_depth` command in the delay script restructures a *technology-independent* network rather than a mapped circuit, it is typical that the delay of the circuit is increased. The trend may have been different had we used the technology-dependent command `speed_up` instead. The resynthesis step then changes the maximum delay of the circuit; the following retiming step places the registers optimally in the circuit. The minimum clock period for s386 over the 30 iteration steps occurs in iteration 27. Notice however that most of the improvement occurs earlier on in the process. In iteration 15, for example, the normalized clock period is 0.675, and it is 0.657 for iteration 27 – a difference of 1.80%. Notice also that retiming does not move any registers in the last few iterations. In this case, it is the synthesis step that changes the delays as the circuit is unmapped and resynthesized and mapped again with each iteration step.

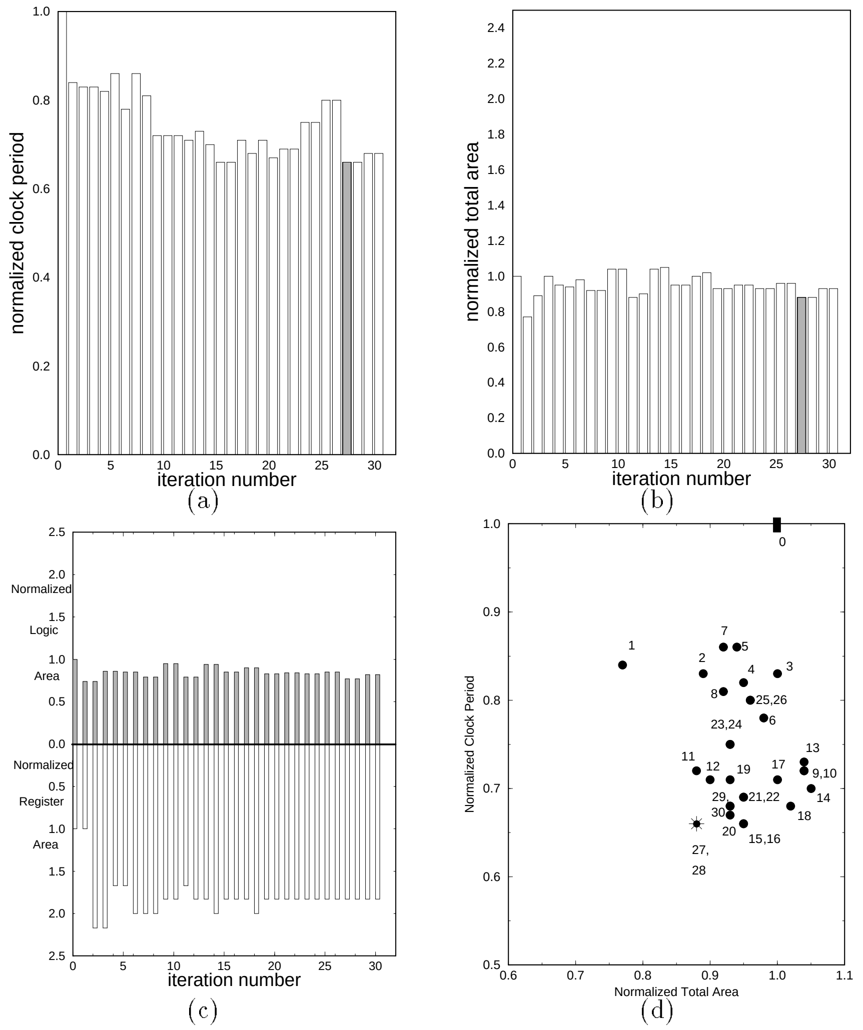


Figure 1: Circuit s386. All numbers are normalized to the reference circuit. (a) Clock period vs. iteration number. (b) Total area vs. iteration number. (c) Logic and register areas vs. iteration number. (d) Area vs. clock period.

Figure 1(b) presents the corresponding normalized area for circuit c386. The total area is reduced by 23% after the first synthesis step, and the following retiming step increases the area to achieve a smaller clock period. This was a typical behavior for all the circuits we examined as we were optimizing for performance, and not for area. After the first two iterations, however, the change in area was unpredictable.

The top half of Figure 1(c) displays the logic area of circuit s386 for each iteration step, while the bottom half displays the register area. Both are normalized to the reference circuit. The normalized logic area corresponds to 158 gates from the library, and the normalized register area is due to 6 registers. With each iteration, the logic area does not increase beyond that of the reference circuit. The number of registers however, more than doubles in iterations 2, 3, 6, 7, and 8. Notice that the logic area does not change with each retiming step, as expected; however, the number of registers sometimes decreases with logic minimization. For example, the number of registers decrease by one after resynthesis in iterations 9 and 11.

Figure 1(d) plots each point we were able to obtain in the design space. The label next to each point is the step number. The reference circuit is at (1.0, 1.0). The circuit resulting from resynthesis is labeled 1. It has the minimum area. Points labeled 27 and 28 correspond to the circuit that had the minimum clock period over the 30 iterations.

Now that we understand the iterative process better, let us take a closer look at the results table and explain the more atypical results. Circuit s208.l initially runs at 10.66 time units. After the first synthesis step, the minimum clock period is 11.44, and then it drops to 11.00 after the first retiming step. During the next iterations, synthesis slightly changes the delay, but retiming does not move any registers. This is due to a single-register cycle, whose delay imposes a lower bound on the minimum clock period. The normalized clock period after one resynthesis and retiming step is greater than one. The numbers that indicate that the iterative procedure does better than a single step of resynthesis and retiming (0.97 reduction in clock period) does not then indicate that the iterative procedure did better. It simply states that we have chosen the circuit with the minimum clock period to report, and it happened to be the reference circuit, which runs at a faster clock period than the one obtained via a single resynthesis and retiming step.

Circuit s27 also has a single register-cycle. Unlike s208.l, however, s27 benefits significantly from the first resynthesis (33% reduction in clock period) and also from the first retiming step (another 15% reduction). Circuit s27 reaches a minimum clock period after a single step of resynthesis and a step of retiming. The iterative procedure cannot further improve the performance.

We are certainly not discouraged by results for circuits s27 and s208.1. They simply point to the ineffectiveness of retiming, and thus the iterative procedure, in optimizing circuits dominated by single-register cycles. If the critical path in the circuit, the one that limits performance, is a single-register cycle, then retiming cannot further improve the performance of the circuit, and thus does not change the position of the register along that cycle. Techniques that change the combinational stage boundaries might then be more suitable than the iterative procedure for optimizing the performance of circuits with single-register cycles [2, 13, 3, 6, 1].

4 Varying the Number of Iterations

In Table 2 we reported the circuit with the minimum clock period obtained through a total of 30 retiming and resynthesis step. As we noticed for circuit s386, Figure 1(a), most of the benefits occur earlier in the iterative process. We must then examine the relation between the number of iterations and the reduction in clock period to determine when the iterative procedure has exhausted its potential.

	basic experiment				additional retiming step first				no reg minimization during retiming			
Circuit	T_{clk}	logic area	reg area	It	T_{clk}	logic area	reg area	It	T_{clk}	logic area	reg area	It
mm4a	0.61	0.75	1.08	30	0.65	0.62	1.00	2	0.61	0.91	1.58	15
s208.1	1.00	1.00	1.00	0	0.82	1.00	1.88	1	0.94	1.10	2.50	4
s27	0.52	0.86	1.00	2	0.60	0.94	1.33	2	0.52	0.86	1.00	2
s298	0.61	0.95	2.21	16	0.68	1.46	3.07	23	0.63	1.37	5.21	18
s349	0.87	1.43	2.47	12[a]	0.81	1.02	2.40	5[b]	0.74	2.12	6.13	20
s386	0.66	0.77	1.83	27	0.64	0.90	2.00	15	0.70	0.82	3.50	12
s400	0.56	1.06	1.38	12	0.52	1.20	1.57	7	0.63	1.72	4.00	10
s444	0.55	1.27	2.52	24	0.53	1.02	1.62	7[c]	0.59	2.37	5.81	28
s641	0.71	1.14	1.11	7[d]	0.70	0.99	0.95	5[e]	0.64	1.29	1.53	23
s713	0.72	1.01	1.32	8[f]	0.75	0.97	1.16	5[g]	0.66	1.08	1.16	9

[a]Maximum iteration completed is 14
[b]Maximum iteration completed is 24
[c]Maximum iteration completed is 30
[d]Maximum iteration completed is 10
[e]Maximum iteration completed is 11
[f]Maximum iteration completed is 10
[g]Maximum iteration completed is 7

Table 3: *Results of applying iterative resynthesis and retiming with an additional retiming step and without register minimization during retiming. All the numbers are normalized to the reference circuit.*

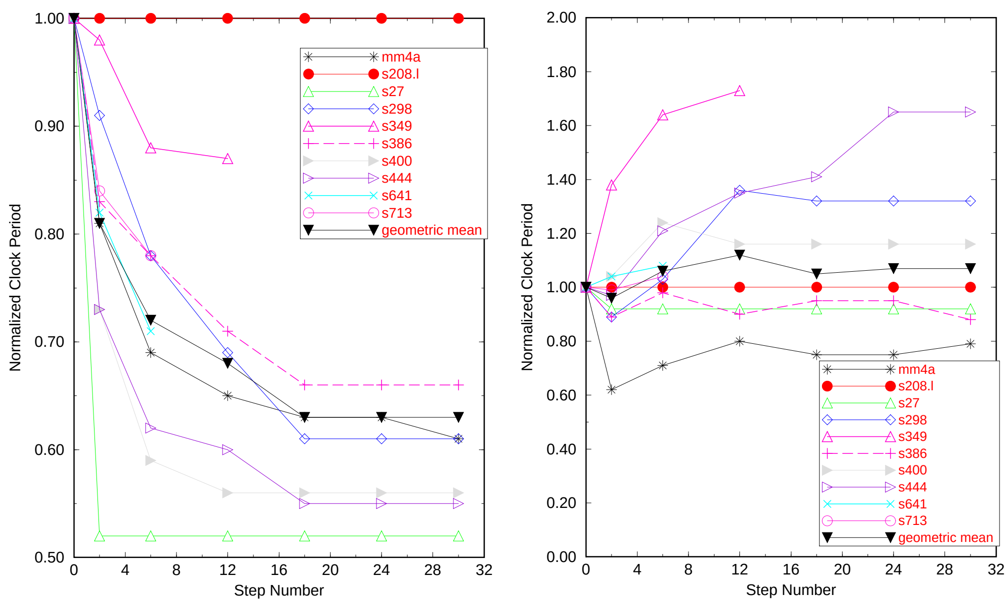


Figure 2: The effects of varying the iteration numbers. (a) The normalized clock period for our test circuits v.s. the step number. (b) The normalized area v.s. the step number.

The graph in Figure 2(a) plots the normalized best clock period that can be obtained at the end of a single resynthesis-retiming iteration (step 2), and at the end of three iterations (step 6), and every three thereafter. We also plot the geometric mean at each of the evaluation points. The average reduction in clock period after one synthesis-retiming step is then 19%. After three resynthesis-retiming steps, it is 28%, and so on. The clock period does not further improve after step 18 for all the circuits except for circuit mm4a. After 18 steps, the geometric mean indicates a 37% reduction in clock period. In other words, the iterative procedure can improve an *optimized* circuit by an additional 18%, for the seven circuits that completed 9 iterations of resynthesis-retiming.

The graph in Figure 2(b) plots the corresponding normalized area. The area does not change significantly after step 12 except for circuit s444. The area change after 18 steps ranged between −25% and +40%, which is within the same change range for the circuit after a single synthesis/retiming step. The geometric mean indicates an average increase of 5%.

5 Secondary Effects

The most important factor that affects iterative resynthesis and retiming is varying the number of steps used in the procedure. In this section we examine two other factors that may influence the results. We examine the effects on circuit s386 for each factor, and then present the results for the rest of the circuits.

5.1 Retiming Step First

In our original experiment we chose to begin our iterative procedure with a synthesis step. How would the results have been modified had we begun with a retiming step?

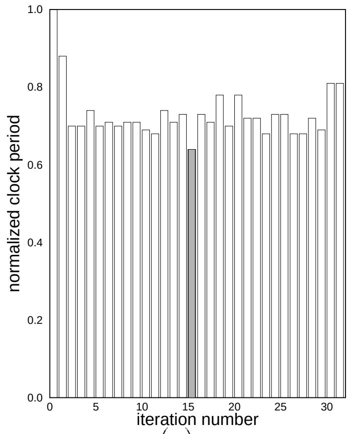
The left 3 graphs in Figure 3(a,c,e) illustrate the clock period, the area, the logic area, and the register area, all normalized to the reference circuit. There are a couple of minor differences between the graphs here and the ones for the original experiment. First, the minimum clock period is achieved at an earlier time step. Second, there is more variations in the register area.

5.2 No Register Minimization

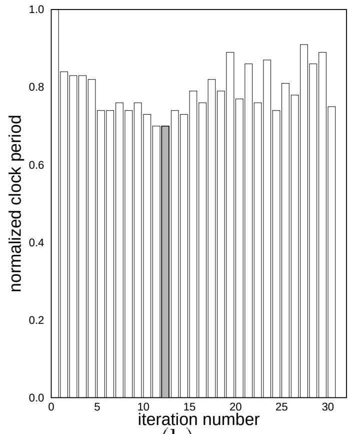
We examined the effect of retiming without register minimization on the test circuits. The right 3 graphs in Figure 3(b,d,f) illustrate the clock period, the area, the logic area, and the register area, all normalized to the reference circuit. While there is a reasonable 6% reduction in the clock period when compared to the minimum clock period obtained using the original setup, there is a large difference in the total area. Upon closer examination, it is the area of the registers that goes through a rapid growth with each iteration while the area of the logic remains about the same.

5.3 Results Summary

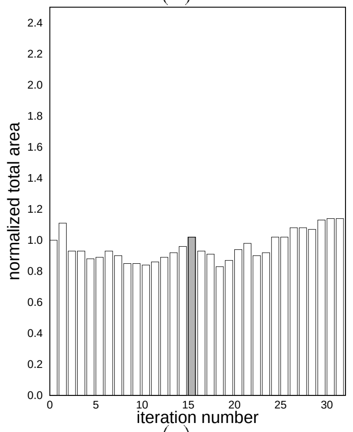
Table 3 reports the normalized clock period and the corresponding register area, logic area, and step number during which the circuit with the smallest clock period was found. All the results are normalized to the reference circuit. Figure 4 summarizes the results of our experiment and its variations. We plot the geometric mean for with the smallest clock period at the end of 2, 6, 8, 12, 18, 24, and 30 for the original experiment and for the experiment that had retiming without register minimizing. We also plot the geometric mean at the end of iterations 1, 3, 9, 13, 19, 25, and 31 for the experiment with the retiming step first.


normalized clock period
iteration number

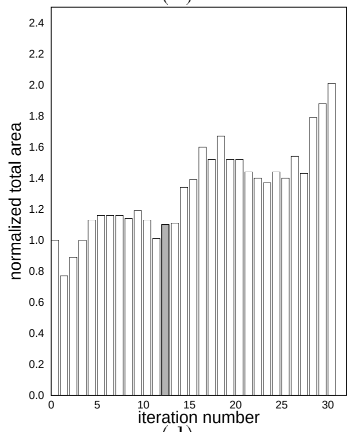
(a)


normalized clock period
iteration number

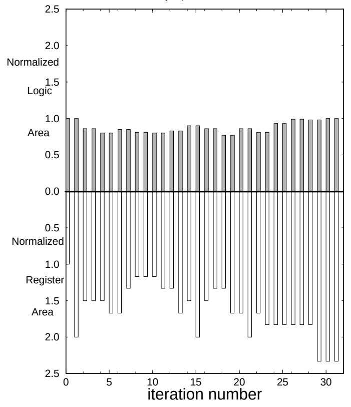
(b)


normalized total area
iteration number

(c)


normalized total area
iteration number

(d)


Normalized
Logic
Area
Normalized
Register
Area
iteration number