Performance Optimization of Sequential Circuits by Eliminating Retiming Bottlenecks

Sujit Dey Miodrag Potkonjak Steven G. Rothweiler
C&C Research Labs, NEC USA
Princeton, NJ 08540

Abstract

Retiming is an effective technique to optimize the performance of synchronous sequential circuits. This paper proposes a method to improve the effectiveness of retiming by transforming the sequential circuit. Bottlenecks which prevent retiming to achieve a desired clock period are identified. Conditions to eliminate the retiming bottlenecks are derived. These conditions are satisfied by a process of identifying sub-circuits and satisfying a set of timing constraints on the sub-circuits. The transformed circuit, which satisfies the timing constraints, can be retimed to achieve the desired clock period. If the original circuit has its initial state specified, our method always generates the final circuit with an equivalent initial state. Experimental results on a variety of sequential benchmark circuits demonstrate significant performance improvement.

1 Introduction

While logic synthesis has been able to deliver area-efficient testable designs, the performance of synthesized designs remains a major concern. In the past, several efforts have been made in optimizing the delay of combinational circuits [1, 2, 3, 4]. Since most real life designs are sequential in nature, we address the problem of optimizing the performance of sequential circuits.

Combinational delay optimization techniques can be applied to sequential circuits by optimizing the combinational logic blocks between the latches. However, this approach does not exploit the sequential nature of the circuit.

Retiming [5] is an effective technique to optimize the clock period of synchronous sequential circuits. The retiming technique considers only the sequential elements of the circuit, it assumes that the combinational logic structure is fixed. Consequently, though the retiming algorithms in [5] find clock period optimum with respect to retiming, the clock period may be further improved by changing the combinational logic and then applying retiming.

We briefly review the previous work done to optimize the performance of sequential circuits by using retiming with certain other techniques. In [6], a set of logic synthesis operations has been combined with retiming to optimize sequential circuits for area and clock period. A technique has been proposed in [7] to optimize the cycle time of sequential circuits which use multi-phase clock, by moving combinational logic across latches. Peripheral retiming has been used to optimize the performance of pipeline circuits using combinational delay optimization techniques [8]. The retiming tehnique has also been extended to sequential circuits using level-sensitive latches [9, 10].

As a motivation to the performance optimization technique presented in this paper, let us consider the sequential circuit shown in Figure 1(a). The clock period, 3, is optimal with respect to retiming. The combinational logic of the circuit cannot be restructured by combinational delay optimization techniques to reduce the clock period of the circuit. However, retiming can be enabled by transforming the circuit as shown in Figure 1(d). Though the clock period of the transformed circuit is not reduced, retiming can now reduce the clock period to 2, producing the circuit shown in Figure 1(e).

This paper addresses the problem of enabling retiming by transforming the sequential circuit. We identify the bottlenecks in the sequential circuit which prevents retiming to achieve a desired clock period. Eliminating the retiming bottlenecks enables retiming to achieve the desired reduction in clock period. Instead of explicitly identifying and eliminating each retiming bottleneck, we derive a set of conditions that must be satisfied by the paths of the circuit to eliminate all the retiming bottlenecks simultaneously and enable retiming.

We propose a technique to transform the circuit so that the retiming bottlenecks are eliminated. The proposed technique consists of identifying sub-circuits and satisfying a set of timing constraints on the sub-circuits. Satisfying the timing constraints for all the sub-circuits may require an increase in the clock period of the circuit. However, it is guaranteed that the transformed circuit can be retimed to achieve the desired clock period.

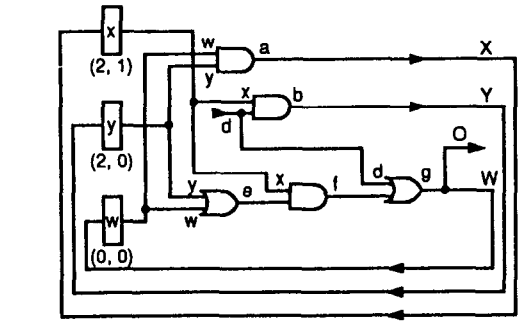
Experimental results show that our technique can significantly improve the effectiveness of retiming. The clock period of the sequential circuits can be significantly reduced beyond the optimum clock period achieved by retiming. The experimental results also compare very well with a combined application of retiming and combinational delay optimization.

1.1 Definitions

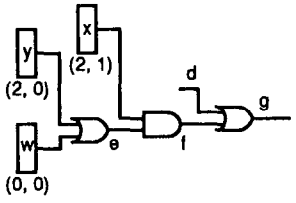
We consider synchronous sequential circuits consisting of gates and edge-trigerred latches. Besides the Primary Inputs (PI) and the Primary Outputs (PO) of the circuit, the outputs and inputs of the latches are considered to be Pseudo-Primary Inputs (PPI) and Pseudo-Primary Outputs (PPO) of the circuit, respectively.

A **path** p from node x to node y is a sequence of nodes and edges starting with x and ending with y. A path is *simple* if all nodes and all edges on the path, except the first and last node, are distinct. A **cycle** is a simple path which begins and ends with the same node. Throughout this paper, by a path, we refer to a simple path. Each node v has a propagation delay $d(v)$ associated with it. The **delay** (or, **length**) of a path p, d(p), is the sum of the propagation delays of

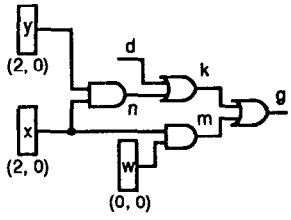
0-8186-3010-8/92 $03.00 © 1992 IEEE



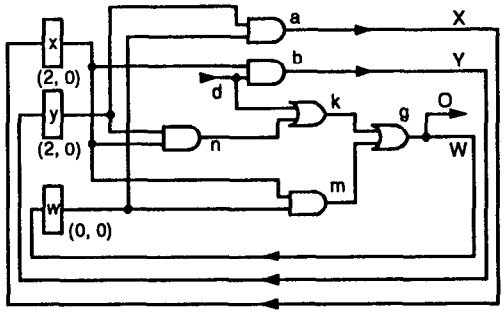
(a)



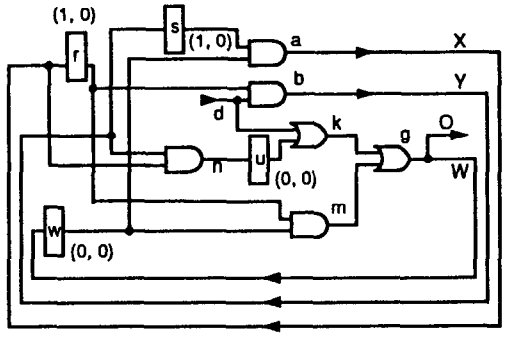
(b)



(c)



(d)



(e)

Figure 1: Enabling Retiming by Eliminating Retiming Bottlenecks

the nodes in the path. The **weight** of a path p, w(p), is the number of latches along path p.

A path p having $w(p) = 0$ is referred to as a combinational path. A path having $w(p) \geq 0$ is referred to as sequential path or simply, a path. For a synchronous sequential circuit, the **clock period** ϕ is the length of the longest combinational path in the circuit: $\phi = \max\{d(p)|w(p) = 0\}$. A sequential path contains one or more combinational paths. The **period** of a sequential path is the length of its longest combinational path.

2 Bottlenecks for Retiming

In this section, we identify the bottlenecks present in the sequential circuit that prevent retiming from acheiving the desired clock period $(\phi - \epsilon)$, where ϕ is the initial clock period, and ϵ is the desired reduction. A combinational path p is termed critical if its length $d(p)$ is greater than $(\phi - \epsilon)$. A sequential path p is termed critical if its length $d(p)$ is such that *any* retiming of the path cannot reduce its period to $\phi - \epsilon$. Considering each node has unit delay, a sequential path is critical if its length $d(p) > (\phi - \epsilon) * (w(p) + 1)$.

Similarly, a cycle C is critical if *any* retiming of the cycle cannot reduce its period to $\phi - \epsilon$. Considering unit delay at every node, a critical cycle satisfies $d(C)/w(C) > (\phi - \epsilon)$. Assuming $\epsilon = 1$, the cycle {W,e,f,g,W} in the sequential circuit of Figure 1(a) is a critical cycle. However, the cycle {Y,e,f,g,W,a,X,b,Y} is not critical.

The **retiming bottlenecks**, which prevent retiming to achieve the desired clock period, are:

R1 Combinational critical path from PI to PO,

R2 Sequential critical path from PI to PO,

R3 Critical cycle.

The following theorem follows from Theorem 11 in [6].

Theorem 1 *Retiming cannot reduce the clock period of a circuit by ϵ if and only if the circuit has at least one retiming bottleneck.*

2.1 Conditions for Eliminating Retiming Bottlenecks

Theorem 1 ensures that eliminating the retiming bottlenecks from a sequential circuit would *enable* retiming to achieve the desired clock period. However, there may be an exponential number of retiming bottlenecks in a circuit. Enumerating each bottleneck and eliminating it explicitly may be prohibitive. Instead, we identify a set of conditions, which, if satisfied, eliminates all the retiming bottlenecks simultaneously. Our approach is based on the observation that unlike combinational nodes, a latch on a critical path may have a slack greater than ϵ associated with it.

The slack at a combinational node v, $s(v) = r(v) - a(v)$, where $r(v)$ is the required time of v, and $a(v)$ is the arrival time of v. With each latch node X, we associate a tuple $(s_i(X), s_o(X))$, where $s_i(X)$ is the input_slack of X, and $s_o(X)$ is the output_slack of X. $s_i(X) = \phi - a(\text{fanin(X)})$; $s_o(X) = min_{v \in fanout(X)}\{s(v)\}$. The tuples $(s_i(X), s_o(X))$ associated with the latches are shown in Figure 1(a).

A latch X is a **forward slack latch** (FSL) if its input_slack $s_i(X) > \epsilon$. An FSL X can be moved forward by a distance of $(s_i(X) - \epsilon)$ without making any combinational path to X critical.

Moving FSL X forward by $(s_i(X) - \epsilon)$ increases the length of paths to X by $(s_i(X) - \epsilon)$, and decreases the length of paths from X by $(s_i(X) - \epsilon)$. In the sequential circuit shown in Figure 1(a), latches X and Y are FSLs, while latch W is not.

Similarly, a latch whose output_slack $s_o(X) > \epsilon$ can be moved backward by $(s_o(X) - \epsilon)$, without making any combinational path from X critical. However, to guarantee that the final circuit after retiming has an equivalent initial state, we choose to use only the forward movement of FSLs during retiming. We derive a set of conditions on the paths of the circuit such that satisfying the conditions ensures that the FSLs in the transformed circuit can be moved forward (retimed) to achieve the desired clock period $(\phi - \epsilon)$.

Let the inputs (outputs) of the circuit be partitioned into forward slack latches (FSL) and all other inputs (outputs) which are not forward slack latches (NFSL). If a combinational path beginning with an FSL, X, (whose movement can *decrease* its length by $(s_i(X)-\epsilon)$) and ending with an FSL, Z, (whose movement may *increase* its length by $(s_i(Z)-\epsilon)$) has length no greater than $\phi-\epsilon+(s_i(X)-\epsilon)-(s_i(Z)-\epsilon)$, the length of the path can be reduced to $(\phi-\epsilon)$ by moving latch X forward during retiming. Similarly, any combinational path beginning with an FSL, X, and ending with an NFSL must have length no greater than $\phi - \epsilon + (s_i(X) - \epsilon)$. Any combinational path beginning with an NFSL and ending with an FSL, Z, must have length no greater than $\phi - \epsilon - (s_i(Z) - \epsilon)$. Finally, combinational paths between NFSLs must have length no greater than $\phi-\epsilon$, because retiming may not be used to reduce these paths. We list below the conditions, applicable to the circuit before retiming.

B_{FF}. All combinational paths p_{ff} from any FSL X to any FSL Z have length $d(p_{ff}) \leq \phi - \epsilon + s_i(X) - s_i(Z)$.

B_{FN}. All combinational paths p_{fn} from any FSL X to any NFSL j have length $d(p_{fn}) \leq \phi + s_i(X) - 2\epsilon$.

B_{NF}. All combinational paths p_{nf} from any NFSL i to any FSL Z have length $d(p_{nf}) \leq \phi - s_i(Z)$.

B_{NN}. All combinational paths p_{nn} from any NFSL to any NFSL have length $d(p_{nn}) \leq \phi - \epsilon$.

Theorem 2 *Satisfying conditions B_{FF}, B_{FN}, B_{NF} and B_{NN} eliminates all the retiming bottlenecks.*

Proof: Let η be the original circuit and $\hat{\eta}$ be the new circuit which satisfies the conditions B_{FF}, B_{FN}, B_{NF} and B_{NN}.

Since condition B_{NN} is satisfied, $\hat{\eta}$ does not have any combinational critical path from a primary input to a primary output (R1). Consider a cycle C having m latches in the circuit $\hat{\eta}$. The set of combinational paths, P, in the cycle can be partitioned into four mutually exclusive sets: P_{FF}, containing those paths in C which begin with an FSL and end with an FSL, P_{FN}, containing paths in C which begin with an FSL and end with an NFSL, P_{NF}, containing paths in C beginning with an NFSL and ending with an FSL, and P_{NN}, conatining paths in C which begin with an NFSL and end with an NFSL.

The total slack on the nodes of the cycle C,

$$slack(C) = \sum_{p \in P_{FF}} slack(p) + \sum_{p \in P_{FN}} slack(p) + \sum_{p \in P_{NF}} slack(p) + \sum_{p \in P_{NN}} slack(p)$$

For any path $p \in P_{FF} \cup P_{FN}$, let X_p denote the FSL at the beginning of p. For any path $p \in P_{FF} \cup P_{NF}$, let Z_p denote the FSL at the end of p. Substituting in the appropriate quantities for $slack(p)$ from B_{FF}, B_{FN}, B_{NF} and B_{NN}:

$$slack(C) \geq \sum_{p \in P_{FF}} (s_i(Z_p) - s_i(X_p) + \epsilon) + \sum_{p \in P_{FN}} (2\epsilon - s_i(X_p)) + \sum_{p \in P_{NF}} s_i(Z_p) + \sum_{p \in P_{NN}} \epsilon$$

Algebraic manipulation yields:

$$slack(C) \geq \sum_{p \in P_{FF}} s_i(Z_p) + \sum_{p \in P_{NF}} s_i(Z_p) - (\sum_{p \in P_{FF}} s_i(X_p) + \sum_{p \in P_{FN}} s_i(X_p)) + (|P_{FF}| + 2|P_{FN}| + |P_{NN}|)\epsilon$$

It can be shown that $\sum_{p \in P_{FF}} s_i(Z_p) + \sum_{p \in P_{NF}} s_i(Z_p) = \sum_{p \in P_{FF}} s_i(X_p) + \sum_{p \in P_{FN}} s_i(X_p)$. Also, $(|P_{FF}| + 2|P_{FN}| + |P_{NN}|) = m$. Hence, $slack(C) \geq m\epsilon$. This shows that in the circuit $\hat{\eta}$, any cycle C having m latches has $slack(C) \geq m\epsilon$. Consequently, there is no critical cycle (R3) in the circuit $\hat{\eta}$. Similarly, it can be shown that $\hat{\eta}$ does not contain any sequential critical paths (R2).

Consequently, circuit $\hat{\eta}$, which satisfies conditions B_{FF}, B_{FN}, B_{NF} and B_{NN}, does not contain any retiming bottleneck, R1, R2 and R3. □

Note that in order to achieve the desired clock period reduction of ϵ by purely combinational delay optimization techniques like [1, 2], all combinational paths p of the circuit need to satisfy the condition: $d(p) \leq \phi - \epsilon$. In contrast, conditions B_{FF} to B_{NN} require the reduction of some paths while allowing the increase of some other paths of the circuit; hence, they may be easier to satisfy.

Satisfying the conditions B_{FF} to B_{NN} may increase the clock period of the circuit. However, Theorem 2 guarantees that the new circuit does not contain any retiming bottlenecks. This enables a subsequent retiming step to achieve the desired clock period.

3 Timing Constraints to Enable Retiming

In this section, we outline our approach to satisfy the conditions B_{FF} to B_{NN} to eliminate the retiming bottlenecks and enable retiming. Our approach consists of identifying a set of cones, and satisfying a set of timing constraints on the cones.

Let a **retime-critical input** be either a PI j with slack $s(j) < \epsilon$ or a PPI X with $s_o(X) < \epsilon$ AND $s_i(X) + s_o(X) < 2\epsilon$. Any critical path from a latch X satisfying $s_i(X) + s_o(X) \geq 2\epsilon$ can be made non-critical during retiming by moving the latch forward. The **retime-critical network** consists of all nodes with slack less than ϵ in the transitive fanout of the retime-critical inputs. We identify a set of nodes {v}, called **cnodes**, such that each path in the retime-critical network has a cnode on it. We extract the cone of each cnode v, cone(v), which is the sub-circuit comprising of all combinational paths from v to the PI/PPIs. Our approach to satisfy conditions B_{FF} to B_{NN} is to identify the set of cnodes and satisfy a set of timing constraints for the corresponding cones.

For a cnode v, the timing constraints for cone(v) are specified in terms of the arrival times of its inputs (PIs and PPIs), and the required

time of its output v. For each input X of cone(v), the new arrival time

$$\hat{a}(X) = \begin{cases} 2\epsilon - s_i(X) & \text{if X is a forward slack latch (FSL)} \\ \epsilon & \text{otherwise} \end{cases}$$

The new required time for cnode v, $\hat{r}(v)$, is set to be its original required time $r(v)$. If the timing constraints are satisfied, all nodes and edges in cone(v), which are not in the transitive fanin of any other node besides v, are eliminated. The cone for v is now replaced by the new circuit Scone(v), which satisfies the constraints. This operation preserves the functionality of the circuit.

Theorem 3 *Consider a set of cnodes $\{v\}$ which is a cutset of the retime-critical network of circuit η. If the timing constraints are satisfied for each cone(v), then the conditions B_{FF} to B_{NN} are satisfied.*

Proof: See [11]. □

Consider the sequential circuit shown in Figure 1(a). The clock period, 3, cannot be reduced by retiming. Let $\epsilon = 1$. The retiming bottleneck in the circuit is the critical cycle {W,e,f,g,W}. x and y are FSLs, while w and d are not-FSLs. While the critical inputs of the circuit are y and w, the only retime-critical input is w.

Suppose node g is selected as the cnode for the retime-critical input w. Cone(g) is shown in Figure 1(b). The timing constraints that need to be satisfied are: $\{a(x) = a(y) = 0;\ \ a(w) = a(d) = 1;\ \ r(g) = 3;\}$ The new cone satisfying the constraints is shown in Figure 1(c). Replacing cone(v) by the new cone produces the transformed circuit shown in Figure 1(d). Though the clock period of the transformed circuit has not reduced, it does not contain any retiming bottleneck. Hence, it can be retimed to reduce the clock period by the desired value $\epsilon = 1$. The FSLs x and y can be moved forward to produce the final circuit, with the desired clock period of 2, as shown in Figure 1(e).

Notice that to improve the delay of the circuit in Figure 1(a) by $\epsilon = 1$, a combinational delay optimization technique would try to satisfy the conditions $\{a(x) = a(y) = a(w) = a(d) = 0;\ \ r(g) = 2;\}$, and would fail.

3.1 Choice of Cnodes

We next address the issue of identifying cnodes such that the timing constraints can be satisfied. Replacing cone(v) by a new cone Scone(v) leads to replication of logic. The two main concerns in the choice of a cnode is the area penalty due to replication of logic and the potential of cone(v) to satisfy its timing constraints.

A heuristic measure of the potential of cone(v) to satisfy its timing constraints is derived from the following observations. (a) Satisfying the constraints consists of making paths from retime-critical inputs shorter at the expense of making paths from FSLs longer. Hence, the more the number of FSL inputs of a cone as compared to the retime critical inputs, the larger the potential of satisfying the constraints. (b) If cone(v) is a balanced tree, chances of satisfying the constraints is reduced. Let cone(v) have n inputs. The higher the value of $a(v)/\log_2 n$, the less balanced cone(v) is, and the better the chances of satisfying the constraints. (c) Reconvergence of signals in cone(v) increases the potential of optimizing cone(v) [12]. We attempt to find cnodes v with smallest level from the PI/PPIs such that cone(v) has a good potential to satisfy its timing constraints [11].

4 Algorithm to Eliminate Retiming Bottlenecks

We outline the algorithm **ERB** to eliminate the retiming bottlenecks by identifying and satisfying timing constraints of cones, and retiming the circuit to produce a final circuit with the desired clock period and an equivalent initial state. For each retime-critical input i, the function **cnodes(i)** identifies the cnodes with smallest possible level, such that each critical path from i has a cnode. From the set of cnodes, cnodeSet, for all the retime-critical inputs, a minimum set of cnodes, min_cnodeSet, is identified by the function **cnode_cover**, such that min_cnodeSet is a cutset of the retime-critical network. This is the initial set of cnodes.

Cnodes in min_cnodeSet are now processed in a reverse levelized order. For selected cnode v, the cone(v) is extracted. The cone is collapsed into a single node, and the **speed-up** algorithm [1] in SIS [13] is used to satisfy the timing constraints for the cone. If the timing constraints are satisfied, all nodes and edges in cone(v), which are not in the transitive fanin of any other node besides v, are eliminated. The cone for v is now replaced by the new circuit Scone(v). This operation preserves the functionality of the circuit. Cnode v is deleted from min_cnodeSet, and the next cnode with highest level is selected.

If the timing constraints for cone(v) cannot be satisfied, function cnodes(v) is called to identify a new set of cnodes C_v which covers v. Some nodes of C_v may cover other cnodes present in min_cnodeSet. These cnodes are identified and deleted from min_cnodeSet by the function **cnode_cover**. Subsequently, each new cnode is processed. The algorithm fails when cnodes(v) fails to find a set of cnodes which covers the node v.

After the timing constraints of all the cones have been satisfied, the new circuit can be retimed to achieve the desired clock period $(\phi - \epsilon)$. Given the initial state of the original circuit, it may not always be possible to find an equivalent initial state of the final retimed circuit. To circumvent this problem, we use function **retime_InitSt** to move the latches and simultaneously compute the initial states. Consequently, ERB returns a final circuit with the desired clock period $(\phi - \epsilon)$, and the new initial state. An outline of algorithm **ERB** is presented below:

Algorithm ERB

```
1.  for each retime-critical input i
2.      if ((C_i = cnodes(i)) is empty) return (FALSE);
3.      else cnodeSet = cnodeSet ∪ C_i;
4.  min_cnodeSet = cnode_cover(cnodeSet);
5.  while (min_cnodeSet not empty) {
6.      select cnode v ∈ min_cnodeSet with highest level
7.      if ∃ Scone(v) satisfying timing constraints of cone(v){
8.          remove signals in cone(v) in the transitive fanin of only v;
9.          replace cone(v) by Scone(v);
10.         min_cnodeSet = min_cnodeSet - {v};
        }
        else {
11.         if ((C_v = cnodes(v)) is empty) return (FALSE);
            else {
12.             min_cnodeSet = min_cnodeSet ∪ C_v;
13.             min_cnodeSet = cnode_cover(min_nodeSet);
            }
```

Ckt	Type	Initial			Retime			ERB			Ratio (ERB/Ret)		
		lits	latch	clk	lits	latch	clk	lits	latch	clk	lits	latch	clk
mm4a	SML	566	12	25	566	12	25	2162	20	13	3.82	1.67	0.52
mm9a	SML	1063	27	54	1063	27	54	1456	33	24	1.37	1.22	0.44
mm9b	SML	1274	26	74	1274	29	61	2305	55	31	1.81	1.90	0.51
s208.1	SML	161	8	10	161	8	10	157	14	7	0.98	1.75	0.70
s420.1	SML	343	16	12	343	16	12	377	47	9	1.10	2.94	0.75
s510	SML	438	6	12	438	6	11	642	7	9	1.47	1.17	0.82
s641	SML	342	19	24	342	19	24	1621	28	12	4.74	1.47	0.50
s820	SML	939	5	14	939	6	13	1238	7	9	1.32	1.17	0.69
s1423	SML	1019	74	55	1019	77	50	2266	148	17	2.22	1.92	0.34
s1494	SML	1504	6	15	1504	7	14	2008	7	9	1.34	1.00	0.64
sbc	SML	1370	28	15	1370	28	15	1746	29	9	1.27	1.04	0.60
dk14	FSM	283	3	14	283	3	13	240	3	7	0.85	1.00	0.54
dk17	FSM	174	3	12	174	3	11	137	3	6	0.79	1.00	0.55
dk512	FSM	112	4	16	112	4	16	138	4	6	1.23	1.00	0.38
ex1	FSM	542	5	15	542	5	14	687	5	9	1.27	1.00	0.64
ex6	FSM	249	3	12	249	3	11	322	3	7	1.29	1.00	0.64
keyb	FSM	1007	5	27	1007	6	26	1876	6	11	1.86	1.00	0.42
kirkman	FSM	1077	5	16	1077	5	16	637	9	8	0.59	1.80	0.50
display	RTL	234	14	12	234	14	12	114	16	5	0.49	1.14	0.42
fpu	RTL	1583	112	19	1583	167	12	2511	221	9	1.59	1.32	0.75
highway	RTL	294	12	12	294	12	12	198	15	6	0.67	1.25	0.50
aver	-	-	-	-	-	-	-	-	-	-	1.46	1.31	0.54

Table 1: Effect of ERB on Optimally Retimed Circuit

```
    }
  }
14. retime_InitSt;
15. return(TRUE);
```

5 Experimental Results

We have applied our algorithm ERB on a wide variety of MCNC sequential benchmark circuits [14]: multi-level sequential circuits (MSL), sequential circuits derived from state-transition graph descriptions of Finite State Machines (FSM), and sequential circuits derived from RT-Level descriptions (RTL). The FSM examples are synthesized using *state-minimize* (Stamina) and *state-assign* (Nova) routines in SIS [13], and optimized using the standard SIS script to obtain multi-level sequential circuits. The initial circuits for our experiments are derived by decomposing the multi-level circuits into 2-input NAND gates using the *tech_decomp -a 2* option in SIS. The parameters measured are the number of literals (**lits**), latches (**latch**) and the clock period (**clk**) of the 2-input NAND circuits. The clock period is measured using the unit-delay model.

Table 1 shows the effect of applying ERB on optimally retimed circuits. The columns **Initial** and **Retime** show the parameters of the initial circuit and the optimally retimed circuit. Retiming cannot improve the clock period of the circuit beyond what is reported in Table 1. We apply our technique to the optimally retimed circuit, eliminating the retiming bottlenecks, and retiming the circuit again. The results show that ERB can significantly reduce the clock period of optimally retimed sequential circuits. The column **Ratio (ERB/Ret)** shows the ratio of the number of literals, latches and the clock period of the final circuits with the optimally retimed circuits. On an average, our technique achieves a clock period reduction of 46%, while increasing the number of literals by 46% and the number of latches by 31%.

Next, we compare our results with the best results obtained by applying both optimal retiming and the combinational delay optimization technique, speed_up [1]. The columns in Table 2 show the results obtained by applying retiming followed by speed_up **(Ret + Sp)**, and speed_up followed by retiming **(Sp + Ret)**. The speed_up command is used with options *-d 6 -m unit*. We consider the best result obtained by retiming and speed_up in terms of the clock period obtained. If the clock period obtained by (Ret + Sp) and (Sp + Ret) are the same, we consider as best the result with smallest area penalty. The column **Ratio (ERB/Best)** shows the ratio between the results obtained by our technique and the best result produced by retiming and speed_up. The experimental results show that our technique consistently performs better than the best combination of retiming and speed_up. On an average, our technique performs better by 29% in terms of the clock period, while paying a penalty of 30% in number of literals and 34% in number of latches.

6 Conclusion

We have presented a new technique to optimize the performance of sequential circuits. The aim of the technique is to transform the sequential circuit so that retiming can be more effective in reducing the clock period of the circuit. Experimental results show a significant reduction in the clock period of optimally retimed sequential benchmarks circuits. Our performance optimization results also compare very well with combined applications of retiming and combinational delay optimization.

Ckt	Type	Ret + Sp			Sp + Ret			ERB			Ratio (ERB/Best)		
		lits	latch	clk	lits	latch	clk	lits	latch	clk	lits	latch	clk
mm4a	SML	526	12	22	526	12	22	2162	20	13	4.11	1.67	0.59
mm9a	SML	1432	27	27	1482	27	26	1456	33	24	0.98	1.22	0.92
mm9b	SML	1249	29	51	2133	26	36	2305	55	31	1.08	2.12	0.86
s208.1	SML	179	8	8	179	8	8	157	14	7	0.88	1.75	0.88
s420.1	SML	355	16	10	355	16	10	377	47	9	1.06	2.94	0.90
s510	SML	441	6	10	468	6	10	642	7	9	1.46	1.17	0.90
s641	SML	599	19	16	599	19	16	1621	28	12	2.71	1.47	0.75
s820	SML	729	6	11	719	5	11	1238	7	9	1.72	1.40	0.82
s1423	SML	1969	77	22	2265	74	22	2266	148	17	1.15	1.92	0.77
s1494	SML	1376	7	12	1355	6	12	2008	7	9	1.48	1.17	0.75
sbc	SML	1453	28	11	1453	28	11	1746	29	9	1.20	1.04	0.82
dk14	FSM	270	3	12	271	3	12	184	3	7	0.68	1.00	0.58
dk17	FSM	173	3	10	171	3	10	137	3	6	0.80	1.00	0.60
dk512	FSM	134	4	11	134	4	11	138	4	6	1.03	1.00	0.55
ex1	FSM	484	5	14	544	5	14	677	5	9	1.40	1.00	0.64
ex6	FSM	252	3	10	343	3	9	313	3	7	0.91	1.00	0.78
keyb	FSM	790	6	19	819	5	21	1876	6	11	2.37	1.00	0.58
kirkman	FSM	821	5	13	809	5	13	637	9	8	0.79	1.80	0.62
display	RTL	299	14	8	295	14	8	114	16	5	0.39	1.14	0.63
fpu	RTL	1603	167	11	1635	138	12	2511	221	9	1.57	1.32	0.82
highway	RTL	246	12	8	246	12	8	198	15	6	0.80	1.25	0.75
aver	-	-	-	-	-	-	-	-	-	-	1.30	1.34	0.71

Table 2: Comparison of ERB with Retiming and Combinational Delay Optimization

An advantage of ERB over other sequential optimization techniques that involve retiming is that ERB always produces initial states along with the final circuit. This is possible because ERB uses only the forward movement of latches, and thus can easily compute the new initial states as it moves the latches. In contrast, the optimization achieved by the other techniques may or may not be valid, depending on whether the initial state of the final circuit can be computed or not.

The penalty in number of latches can be effectively minimized by applying a latch-minimization technique in [5] on the final circuit. However, the time/space requirements of such a technique is prohibitive for most circuits. We are currently investigating techniques which will minimize the number of latches required by ERB to achieve the desired clock period.

The conditions to eliminate retiming bottlenecks are sufficient, but may not be necessary. We are currently investigating the problem of finding the necessary conditions by considering the output slacks of latches.

Acknowledgements

We would like to acknowledge Alex Ishii and Srimat T. Chakradhar for their helpful discussions.

References

[1] K.J. Singh, A.R. Wang, R.K. Brayton, and A. Sangiovanni-Vincentelli. Timing Optimization of Combinational Logic. In *Proc. IEEE International Conference on Computer-Aided Design*, pages 282 – 285, November 1988.

[2] P.C. McGeer, R.K. Brayton, A.Sangiovanni-Vincentelli, and S.K. Sahni. Performance Enhancement through the Generalized Bypass Transform. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 184 – 187, November 1991.

[3] H. Touati, H. Savoj, and R.K. Brayton. Delay Optimization of Combinational Logic Circuits by Clustering and Partial Collapsing. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 188 – 191, November 1991.

[4] D. Hathaway, L.H. Trevillyan, C.L. Berman, and A.S. LaPaugh. Efficient Techniques for Timing Correction. In *Proc. ISCAS*, pages 391–394, June 1985.

[5] C.E. Leiserson and J.B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, 6:5 – 35, 1991.

[6] G. De Micheli. Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization. *IEEE Transactions on Computer Aided Design*, 10(1):63 – 73, January 1991.

[7] K. Bartlett, G. Borriello, and S. Raju. Timing Optimization of Multiphase Sequential Logic. *IEEE Transactions on Computer Aided Design*, 10(1):51 – 62, January 1991.

[8] S. Malik, K.J. Singh, R.K. Brayton, and A. Sangiovanni-Vincentelli. Performance Optimization of Pipelined Circuits. In *Proc. IEEE International Conference on Computer-Aided Design*, pages 410 – 413, November 1990.

[9] N. Shenoy and R.K. Brayton. Retiming of Circuits with Single Phase Transparent Latches. In *Proceedings of the International Conference on Computer Design*, October 1991.

[10] A.T. Ishii, C.E. Leiserson, and M.C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. In *Advanced Research in VLSI and Parallel Systems: Proc. of the 1992 Brown/MIT Conference*, pages 245 – 264, March 1992.

[11] S. Dey, M. Potkonjak, and S.G. Rothweiler. Performance Optimization of Sequential Circuits by Eliminating Retiming Bottlenecks. Technical Report 92-C013-4-5016-2, C&C Research Labs, NEC USA, May 1992.

[12] S. Dey, F. Brglez, and G. Kedem. Circuit Partitioning for Logic Synthesis. *IEEE Journal of Solid-State Circuits*, 26(3):350 – 363, March 1991.

[13] E.M. Sentovich, K.J. Singh, C. Moon, H. Savoj, R.K. Brayton, and A. Sangiovanni-Vincentelli. Sequential Circuit Design using Synthesis and Optimization. In *Proceedings of the International Conference on Computer Design*, October 1992.

[14] Saeyang Yang. Logic Synthesis and Optimization Benchmarks, User Guide Version 3.0. In *International Workshop on Logic Synthesis*, MCNC, Research Triangle Park, NC, May 1991.