Proceedings of the 10th WSEAS International Conference on CIRCUITS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp355-360)

Combining Retiming and Sequential Redundancy Addition and Removal for Sequential Logic Optimization

ENRIQUE SAN MILLÁN, LUIS ENTRENA, LUIS MENGIBAR, MICHAEL GARCÍA
Electronics Technology Department
University Carlos III of Madrid
Butarque 15, E-28911 Leganés (Madrid)
SPAIN

Abstract: - In this paper a new logic optimization method for sequential synchronous circuits is introduced. For this purpose the current main approaches, "Retiming and Resynthesis" and "Redundancy Addition and Removal" are considered. These techniques have some advantages and limitations that have been theoretically proven by several authors. The goal of the new optimization method is to combine these two techniques to get the best of each one. In particular the paper is focused on area optimization. The algorithm proposed in this paper is efficient and delivers interesting optimization results.

Key-Words: - Sequential logic optimization, Digital Circuits, Retiming and Resynthesis, Redundancy Addition and Removal

1 Introduction

Digital circuit logic optimization has been a very important problem in the last decades. Obtaining a good optimized circuit is essential to get smaller and faster circuits. In particular, logic optimization for synchronous sequential circuits is still an open and challenging problem.

There are two main approaches to sequential logic optimization, Retiming and Resynthesis (RaR) [1] and Sequential Redundancy Addition and Removal (SRAR) [2]. The Retiming technique performs the optimization in two steps. The first step involves moving flip-flops across combinational gates (Retiming), and the second step is performed by optimizing the resulting combinational blocks with combinational techniques (Resynthesis). The set of possible transformations that can be provided with Retiming and Resynthesis is limited. The sets of possible transformations have been formalized in [3] and [4]. This method has become quite attractive despite its limitations.

The Redundancy Addition and Removal technique is based on the iterative addition and removal of redundancies. This technique has been proved to produce excellent results for combinational circuits [2][5][6], being the major advantages the low memory usage and the short run times. Sequential logic optimization is also possible using this technique by considering sequential redundancies [2][7].

The goal of this paper is to obtain an algorithm to get the best of both methods to improve optimization results. For this purpose we first study which are the limitations and advantages of RaR and SRAR. Then a new method is proposed focusing into area optimization.

The rest of this paper is organized as follows. Section 2 reviews the Retiming and Resynthesis technique for sequential logic optimization and the set of possible sequential transformations that provides. Section 3 reviews the Sequential Redundancy Addition and Removal technique and its optimization capabilities. Then in section 4 it is shown a new approach for optimization based in a combination of the two previous methods. In section 5 some experimental results obtained with this algorithm are provided. Finally, section 6 presents the conclusions of this work.

2 Retiming and Resynthesis

Retiming and Resynthesis [1] is a sequential optimization method that can be applied to optimize sequential designs described at the logic level.

Retiming and Resynthesis consists on the application of a sequence of two basic steps, Synthesis and Retiming, which are described as follows:

Synthesis: In this step the flip-flops are untouched and the flip-flop inputs and outputs are treated as primary circuit inputs and outputs. This step provides simple combinational optimization.

Retiming: Moves the flip-flops across combinational blocks under certain rules, with the following effects:

a) Change in cycle time: The delay along the

combinational path between flip-flops can change, because those flip-flops are moved.

b) Change in area: The number of flip-flops can increase or decrease due to the movement across combinational blocks

c) Change the interaction between combinational blocks: This is the most important effect of retiming, as it provides the way to further optimization of the circuit by combinational optimization.

d) Changes on the state transition graph: Retiming may change the state transition graph of the original circuit by changing the different encodings of the state transition graph, or changing the transition between states.

That is, in Retiming the flip-flops are relocated across combinational gates, changing the interaction between different combinational blocks, followed by a Resynthesis step that allows making logic optimizations. A sequence of Retiming and combinational Resynthesis steps provides the way to optimize sequential circuits at the logic level. It can be observed that some of these RaR optimization transformations in the circuit are not possible by only combinational methods.

The possible movements of flip-flops across combinational gates can be built from a sequence of four primitive retiming operations, as stated in the following lemma [3]:

Lemma: A general retiming operation can be constructed as the sequence of retiming moves across primitive transformations i), ii), iii) and iv) shown in the figure 1.

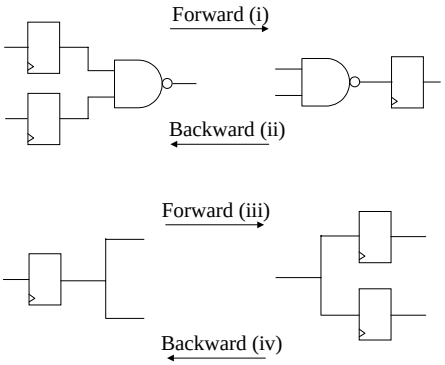

Figure 1: Primitive retiming operations

An example of Retiming and Resynthesis is shown in Fig. 2. The initial circuit has two registers and three combinational gates (fig. 2 a)). The step of retiming reduces the number of registers to one, and produces the interaction between the combinational gates (fig 2 b)). The next step, resynthesis, allows the optimization to be done. The final circuit has only one register and only one gate (fig 2 c)).

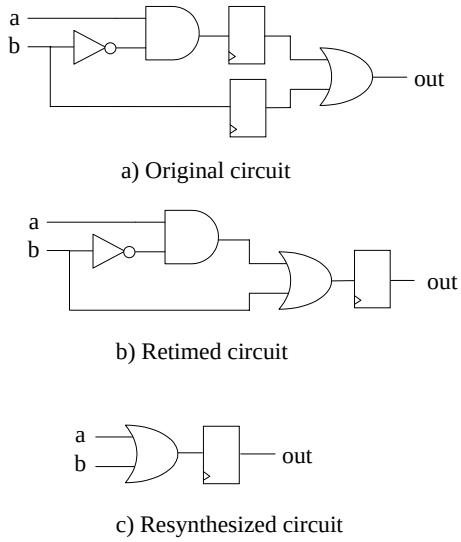

Fig.2. Retiming and Resynthesis example.

Retiming algorithms can be used for area or timing optimization in sequential circuits. They are more suitable for timing optimization because they provide the necessary steps to equilibrate critical paths between memory elements in the circuit. This can be done by simple flip-flop movements. However, Retiming can also be used for area optimization. In this case the goal of the retiming algorithms is obtaining the circuit with minimum number of flip-flops [8].

Retiming and Resynthesis methods have been widely studied, and their optimization capabilities have been formally established by several authors [1] [3] [4].

It has been done by relating transformations in the circuit made by retiming moves with the STG transformations. The characterization of the relationship between Retiming and Resynthesis and STG transformations is given by the two following theorems [1][3]:

Theorem (Malik): Given a machine implementation M1, corresponding to a state transition graph G, with a state assignment S1, it is always possible to derive a machine M2 corresponding to the same state transition graph G, and a state assignment S2 by applying only a series of Resynthesis and Retiming operations on M1.

Theorem: Let M1 be an implementation corresponding to state assignment S1 and STG G1 and M2 be an implementation corresponding to state assignment S2 and STG G2. Then M2 can be obtained from M1 using only a sequence of Retiming

and Resynthesis operations if and only if G1 and G2 are 1-step equivalent.

These two theorems determine all the possible transformations with the Retiming methods. The first theorem shows the encoding capabilities of retiming. Every state codification in a FSM can be reached with retiming transformations. The second theorem characterizes the set of possible transformations with retiming in the STG of the circuit.

It is important to observe that the retiming set of possible transformations is only a subset of all possible transformations in the circuit. This limitation of retiming can be shown by some examples [3][9] where the circuits can not be optimized by only retiming and resynthesis operations.

3 Sequential Redundancy Addition and Removal

Redundancy Addition and Removal has been shown to be a powerful logic optimization method by several authors [2], [6], [7] [10], [11], [12] With this method, a logic network is optimized by iteratively adding and removing redundancies that are identified using Automatic Test Pattern Generation techniques based on the implication of mandatory assignments. If the addition of k redundant wires/gates creates more than k redundant wires/gates elsewhere in the network, the removal of the created redundancies will result in a smaller area.

Redundancy Addition and Removal is also applicable to sequential circuits [2]. In this case, sequential redundancies can be considered for addition and removal. Sequential redundancies can be identified by using the well-known time frame model and performing the implication process across time frames. Sequential redundancy addition may involve also the addition of flip-flops [10] in order to temporarily expand the state set and reduce it later on with redundancy removal. Along with other general improvements that have been proposed, such as using BDDs for implication [7] and multiple wire/gate addition [6], a large set of sequential optimization transformations can be identified.

The basic redundancy addition and removal approach can be summarized as follows. A target wire is selected and tested for stuck-at fault. If no test is possible, then the wire is redundant and can be removed. Otherwise, we try to add a wire or a gate to the circuit in order to make the target fault redundant. If the wire or gate added is redundant then the addition preserves the logic functionality of the circuit. Once this redundant logic is added, the target wire, which now has become redundant, can be removed.

In the case of sequential circuits special care must be taken with redundant wires. Only if a set of certain conditions are satisfied an untestable fault can be considered redundant. [13]. In practice in most cases untestable faults are redundant, and the cases where this is not true is always related to the initialitiation state of the circuit. If all the flip-flops have a reset state and the initial state is well defined then the concepts of untestable fault and redundant wire are equivalent.

The problem of identification of untestable faults in sequential circuits is open to improvements, and several authors are still working on it [14].

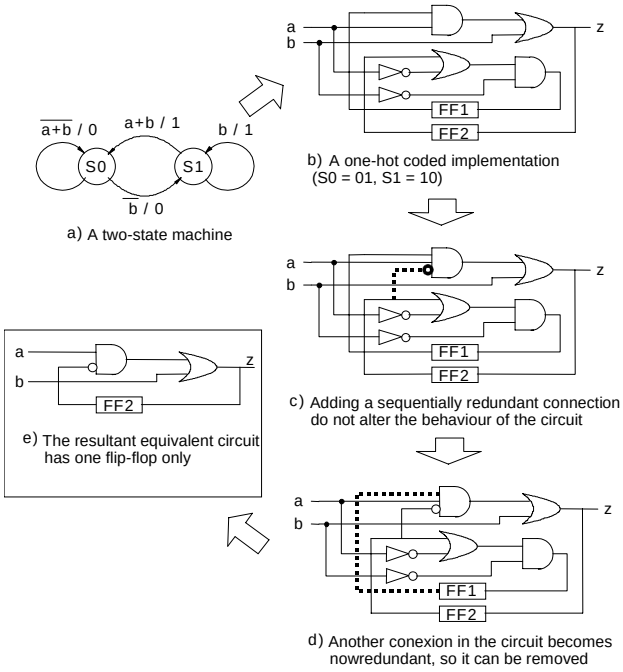

Fig. 3. Example of Sequential Redundancy Addition and Removal

Example (taken from [2]). Fig.3 shows the state graph of a two-state machine and a one-hot coded implementation of this machine. The machine has two inputs, a and b, and one output. To activate the fault x2 stuck-at 1, the circuit must be in state S1 = 10, i.e., x1 = 1 and x2 = 0 are mandatory assignments for this fault. Therefore, adding x1 to the input of g1 will block the propagation of the fault. If we add this connection, as shown in Fig. 3(c), the sequential behavior of the circuit will not change and x2 will become sequentially redundant after this addition (Fig. 3(d)). After removing x2 and its fanins which become floating, the circuit has only one flip-flop and is a minimal-bit encoded machine (Fig. 3(e)).

The search of candidate nodes for addition has been improved by several authors, making the algorithms of redundancy addition and removal very efficient. The possible addition transformations can be identified in only one test pass selecting the appropriate faults [5][6].

Redundancy addition and removal techniques can be used for area and timing optimization as is the case with RaR. The main approaches for timing optimization are based in the selection of target nodes in critical paths in the circuit [15]. In the case of area the approach is based in selecting as target nodes the ones with more area weight, that is the nodes that when removed from the circuit cause more area optimization [2].

The optimization capabilities of these methods have also been studied. It has been proven that these methods are able to perform all the possible Retiming transformations [15]. Additionally there are some examples where a transformation is not possible by Retiming but it is possible by redundancy addition and removal. A transformation in a circuit that it is possible with SRAR and not possible with RaR is shown in the following example:

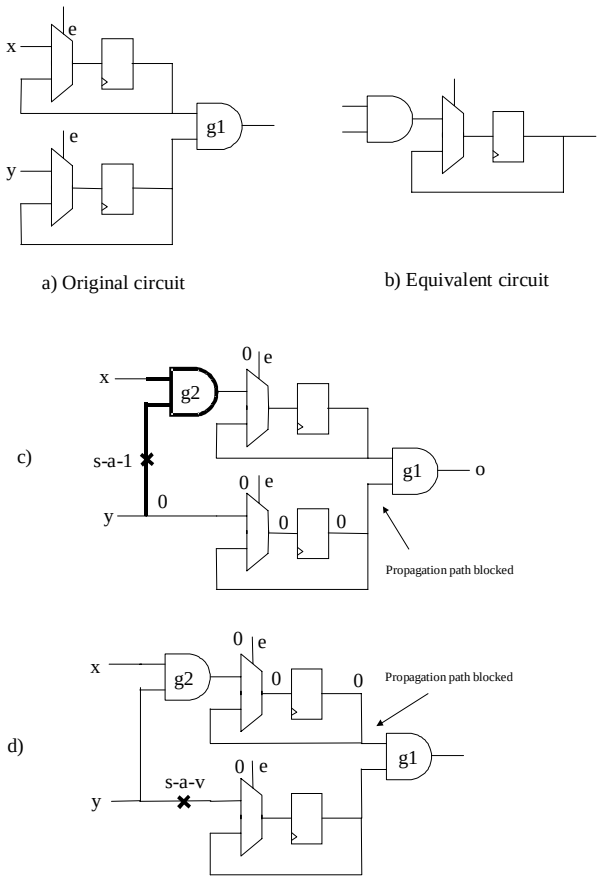

Figure 4. Example

Example: In this example it is not possible to reach circuit b) from circuit a) by a sequence of Retiming and Resynthesis transformations as it is shown in [9]. However, redundancy addition of a gate and removal of a wire makes this transformation possible. The addition of the redundant OR gate shown in figure c) makes the wire shown in figure d) become redundant. The removal of this redundant wire leads to circuit b). This means that SRAR provides a larger set of possible transformations in the circuit.

However this is a theoretical point of view and successful application of a series of transformations for optimization with any technique depends a lot on the practical implementation.

4 Combining RaR and SRAR for sequential logic optimization

Both RaR and SRAR techniques have their limitations. For RaR there is a theoretical limitation and some transformations are not possible. The implementation is very dependent on the second step in combinational resynthesis. Performing several iterations with Retiming and Resynthesis is not very efficient because it is needed to make a full combinational optimization of the circuit in each step. Besides, these methods are more oriented to timing optimization than area.

On the other hand, for the SRAR techniques there are more transformations available. The optimization is very efficient because all transformations are found in only one test step for each target connection. However is more difficult to understand how the transformations are affecting to the behavior of the STG of the circuit. This means that even if all the transformations in the STG are possible, it is unknown how to select the target connections to reach the desired STG.

But both methods have their strengths too: SRAR is a very efficient optimization method that delivers good results and, in general, a single retiming optimization step without resynthesis is efficient and provides some fast and easy sequential transformations.

We try to use the strengths of both methods. The new algorithm proposed is a series of Retiming and SRAR steps. It is a similar method to the original Retiming and Resynthesis, but instead of a combinational only resynthesis, a full sequential redundancy addition and removal is performed. This way it is clear that it should deliver better results than the original RaR, as the SRAR step provides more optimization possibilities.

Another point of view of this approach can be done. The new method proposed is like the original SRAR,

but now an additional Retiming step is performed each iteration. This helps specially in those cases where SRAR can not found a new transformation. In these cases a simple retiming movement of flip-flops can transform the circuit into another one (changing the STG) with new generated SRAR possibilities. This way the new algorithm is able to optimize the circuit in some situations where original SRAR is not. This new method can be summarized in the following algorithm:

1- Perform a Retiming Step. In this step constraints of area or delay can be considered.
2- Perform a SRAR Step. Again, area or delay optimization oriented.
3- If the selected number of iterations is reached then stop, else iterate (go to step 1).

This new algorithm can be used both for area or timing optimization. In the next section we show some experimental results for area optimization. The reason why area optimization is selected is because it is the case where Retiming and Resynthesis is more limited, and therefore more space for optimization is available.

5 Experimental results

In this section we show experimental results for area optimization obtained in ISCAS89 [16] and MCNC benchmark circuits[17].

For the retiming steps SIS Tool from Berkeley is used. Options for area optimization are selected within this tool [18]. For the SRAR steps RAMBO_S is used [7] with some modifications taken from [6].

The results are summarized in the following tables: Table 1 shows the results for a subset of sequential circuits of ISCAS89, and Table 2 shows the results for a subset of MCNC.

Columns labeled as #G, #C and #F show the number of gates, connections and flip-flops in the circuits.

As we are interested in the sequential optimization provided by the different algorithms, the initial circuits are obtained from a previous strong combinational optimization using script.rugged in SIS and RAMBO_C Redundancy Addition and Removal Algorithm. Thus the optimization results shown in these tables are consequence of sequential transformations.

The results show a 22% average optimization in the number of connections in the circuits. These results can be compared with the ones obtained with the previous RaR [8] and SRAR algorithms [7].

	Inicial			SRAR+Retiming			
Circuit	#G	#C	#F	#G	#C	#F	%
s298	103	230	14	72	158	14	30,10
s344	138	295	15	99	240	15	28,26
s382	143	322	54	100	246	19	30,07
s386	88	227	27	63	180	6	28,41
s400	140	311	21	109	228	19	22,14
s420	157	330	41	126	280	16	19,75
s444	141	311	21	85	213	19	39,72
s499	200	502	22	124	330	22	38,00
s526n	174	386	21	145	337	21	16,67
s635	194	480	32	148	394	32	23,71
s820	179	469	41	136	385	5	24,02
s832	168	433	5	133	364	5	20,83
s838	313	658	32	255	572	32	18,53
s953	340	818	29	227	596	23	33,24
s967	314	749	29	212	588	23	32,48
s991	339	695	19	286	610	19	15,63
s1512	488	1068	57	445	999	57	8,81
s3384	1426	2971	183	1215	2638	157	14,80
s4863	1560	3173	84	1179	2481	75	24,42
s6669	2630	5421	231	2381	5081	201	9,47

Table 1. ISCAS 89 results

	Inicial			SRAR+Retiming			
Circuit	#G	#C	#F	#G	#C	#F	%
bbara	48	118	4	34	92	4	29,17
bbsse	64	177	4	50	137	4	21,88
bbtas	20	49	3	12	35	3	40,00
beecount	30	79	3	19	65	3	36,67
dk14	63	152	3	46	122	3	26,98
dk15	43	106	2	26	85	2	39,53
dk512	45	110	4	33	93	4	26,67
ex1	130	351	5	104	299	5	20,00
ex2	86	222	5	64	191	5	25,58
ex5	39	97	3	24	65	3	38,46
ex6	60	159	3	39	109	3	35,00
ex7	52	127	4	27	84	4	48,08
lion	14	31	2	9	23	2	35,71
lion9	22	48	3	17	38	3	22,73
mc	26	57	2	14	41	2	46,15
opus	47	128	4	33	110	4	29,79
s1	197	492	5	143	392	5	27,41
sand	273	747	5	225	653	5	17,58
tav	19	46	2	15	41	5	21,05
train11	46	103	4	36	86	4	21,74

Table 2. MCNC results

In this comparison it can be observed that the optimization is improved in most cases.

6 Conclusions

We have presented a new approach to sequential logic optimization which combines the original Retiming and Resynthesis and Sequential Redundancy Addition and Removal methods.

This new algorithm uses efficiently the strengths of both of them. It has the advantage of the SRAR efficiency and improves the optimization by adding fast retiming capabilities. Another strong point of the algorithm proposed is that it does not add extra computational effort to perform the optimizations.

The experimental work for area optimization shows that this method improves the results obtained with the original RaR and SRAR algorithms.

References:

[1] S. Malik, E. M. Sentovich, R. Brayton, A. Sangiovanni-Vincentelli. *"Retiming and Resynthesis: Optimizing Sequential Circuits Using Combinational Techniques"*, IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 10, p. 74-84. January 1991

[2] L.A. Entrena, K.-T. Cheng. *"Combinational and sequential logic optimization by redundancy addition and removal"*, IEEE Trans. on CAD, Vol. 14, No. 7, July 1995, p. 909-916

[3] R.K. Ranjan, V. Singhal, F. Somenzi, R.K. Brayton. *"On the optimization Power of Retiming and Resynthesis Transformations"*. Proc. ICCAD'98, p 402-407, November 1998.

[4] H.Zhou, V. Singhal, A. Aziz. *"How Powerful is Retiming?"*. Proc. IEEE/ACM intl. Workshop on Logic Synthesis, p 111-125, May 1998.

[5] C.J. Chang, M. Hsiao, M. Marek-Sadowska, *A new reasoning scheme for efficient redundancy addition and removal*, IEEE Trans. on CAD, Vol. 22, No. 7, July 2003, p. 945-952

[6] J. A. Espejo, L. Entrena, E. San Millán, C. López. *"Generalized Reasoning Scheme for Redundancy Addition and Removal"*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E84-A. pp 1665-1672. November 2002.

[7] E. San Millán, L. Entrena, J.A. Espejo, Silvia Chiusano, Fulvio Corno. *"Integrating symbolic Techniques in ATPG-Based Sequential Logic Optimization"*, Proc. DATE'99, p. 516-523 March 1999.

[8] N. Maheshware, S. S. Sapatnekar.*S"An Improved Algorithm for Minimun-Area Retiming"*, Proc. DAC-97, June 1997

[9] R.K.Ranjan. *"Design and Implementation Verification of Finite State Systems"*. PhD thesis. Electronics Research Laboratory. University of California. Berkeley. CA 94720. 1997. Memorandum No. UCB/ERL M97/99.

[10] U. Gläser, K.-T. Cheng. *"Logic Optimization by an Improved Sequential Redundancy Addition and Removal Technique"*, Proc. ASP-DAC. September, 1995

[11] W. Kunz, P. R. Menon. *"Multi-level Logic Optimization by Implication Analysis"*, Proc. ICCAD-94, pp. 6-13. November, 1994

[12] S. C. Chang, M. Marek-Sadowska, *Perturb and Simplify: Multi-level Boolean Network Optimizer*, Proc. ICCAD-94, p. 2-5. November, 1994

[13] V. Singhal, C. Pixley, A. Aziz, R. K. Brayton, *"Theory of safe replacements for sequential circuits"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, No 2, February 2001.

[14] M. Syal, M. Hsiao, *New techniques for untestable fault identification in sequential circuits*, IEEE Trans. on CAD, Vol. 25, No. 6, June 2006, p. 1117-1131

[15] L. Entrena, J.A. Espejo, E. Olías, J. Uceda, *"Timing optimization by an improved redundancy addition and removal technique"*, Proc. European Design Automation Conference, September 1996

[15] Enrique San Millán, Luis Entrena, José A. Espejo, Celia López, *"Theoretical Comparison between Sequential Redundancy Addition and Removal and Retiming Optimization Techniques"*, Journal of Systems Architecture on Synthesis and Verification, vol. 49, pp 529-541, 2003

[17] *SIS: A System for Sequential Circuit Synthesis"*, Report M92/41, University of California, Berkeley, May 1992.

[16] F. Brglez, D. Bryant, K. Kozminski: *"Combinational Profiles of Sequential Benchmark Circuits"*, Proc. International Symposium on Circuits and Systems (ISCAS), 1989.

[18] S. Yang. *"Logic Synthesis and Optimization Benchmarks. User Guide"*. PMicroelectronics Center of North Carolina, Technical Report, January 1991.