EPR - a Synthesis Tool for Speed Optimization

E. Fehlaue, St. Rülke and G. Franke
FhG IIS Erlangen - Department EAS Dresden
Zeunerstr. 38
D-01069 Dresden
Germany
phone: (+49) 351 4640 728 fax: (+49) 351 4640 703
email: ruelke@eas.iis.fhg.de

Abstract

We propose a novel and practicable approach to speed up sequential circuits with pipeline structure at gate level which is called Extended Peripheral Retiming (EPR). Key to EPR is the idea to compute optimal register positions in an iterative manner and only for selected circuit partitions at a time. Different sequential and combinational optimization techniques are combined together. Using EPR, complexity of the optimization task is broken down and bottlenecks of other approaches are removed. Experimental results show the benefit of EPR.

Keywords

Sequential digital circuit, pipelining, optimization, peripheral retiming

1 INTRODUCTION AND OVERVIEW

Methods to optimize a structural description of a synchronous sequential circuit at gate level may be subdivided into combinational, sequential, and combined approaches.

Combinational techniques are the traditional approaches to optimize a circuit but they are limited to circuit partitions between register boundaries only (DeMicheli, 1994). Sequential techniques are more sophisticated but they are usually limited by their restricted scope. Most of them treat only sequential elements or apply a restricted set of transformations to special partitions of the circuit. Besides, techniques like Retiming (Leiserson, Rose and Saxe, 1983)

require run time intensive computations; the optimal register position is determined by a polynomial time procedure. Most interesting are approaches which combine different sequential and/or combinational techniques together. So they augment the optimization potential to consider an enlarged range of optimization goals or to enhance the quality of the solution like (Malik et.al., 1991) and (Martin, 1993). However, up to now, no well defined common method is available taking into account the simultaneous optimization of both, the sequential as well as the combinational part of the circuit in a global manner.

We want to contribute to combined techniques. Our work was inspired most of all by both Peripheral Retiming (Malik et.al., 1991) and Extended CombToPipe (ECTP) (Fehlauer, Rülke and Franke, 1993). EPR (Extended Peripheral Retiming) is a novel and practicable approach to speed up sequential circuits with pipeline structure at gate level taking into consideration area penalties. The original approach Peripheral Retiming performs the following consecutive tasks: Firstly, registers are moved temporarily to the periphery of the circuit (primary inputs and outputs). Then, the resulting enlarged logic between the registers is optimized globally using combinational techniques. Finally, the registers are replaced using Retiming. Unfortunately, this complex approach requires costly computations. Moreover, global combinational optimization may create also additional paths (pseudo dependencies) such that the final register replacement sometimes may not be possible.

The idea of EPR is to perform register replacement in an iterative manner and only for selected partitions of the circuit. The iterative nature of the approach enables us to combine Peripheral Retiming with other techniques gaining timing and/or area optimization in each of the iterations. If pseudo dependencies occur, a backtracking technique ensures that they will be eliminated. The iterative register replacement in combination with the local optimization of logic between the currently created register boundaries utilizes our method ECTP. Based on heuristics, ECTP generates only a minimal number of registers, and ensures that the existing clock cycle constraints will be met - if necessary by generating additional pipeline stages.

In summary, the complexity of the optimization task is broken down, bottlenecks of the original approach are removed, and different sequential and combinational optimization techniques are combined together.

EPR is designed to be a value-added tool to synthesis systems (Synopsys, SIS). So, the user always operates in familiar surroundings. The synthesis tools provide us not only with required algorithms for timing and area optimizations but also with facilities to obtain design analysis.

2 GOALS AND DESIGN OBJECTS

We consider the problem of performance-driven optimization of sequential circuits at gate level, especially the post-synthesis timing improvement if performance (clock cycle) constraints are violated. The goal is to speed-up an initially synthesized circuit without a complete redesign - only by transformations of the structure obtained in the initial design.

Design objects we are focusing our work to are sequential circuits with extended pipeline structure and single-phase edge-triggered clocking. A circuit with pipeline structure consists of a sequence of k processing stages $P_1, \dots, P_k$ separated by registers (e.g. clocked DFF) and has an acyclic topology. Each processing stage is formed by a combinational block which may have different granularity (single gate or complex module of connected gates). The combinational block of stage i communicates with the adjacent block of stage $i+1$ only via registers.

Figure 1 Classification of pipelined circuits.

In case of pure pipeline structures, all primary inputs of the circuit are connected to the combinational block of the first stage and all primary outputs of the circuit are connected to the combinational block of last stage (see Figure 1a). Therefore, on each path between any primary input and any primary output the same number of registers occur.

More flexible structures with additional primary inputs and primary outputs connected to combinational blocks of any stage, but not to two or more blocks of different stages, are called extended pipeline structures (see Figure 1b). Consequently, paths between a primary input PI_x and different primary outputs $PO_{y1}, \dots, PO_{ym}$ may have different numbers of registers. Note that if two or more paths between a certain primary input PI_x and a certain primary output PO_y exist then all of this paths do have the same number of registers.

3 METHODOLOGY OF EPR

Analogous to original Peripheral Retiming, the activities done by EPR can be assigned to three consecutive main steps (see Figure 2):

1. Step S1 moves registers temporarily to the periphery of the circuit.
2. Step S2 optimizes the resulting enlarged combinational logic between the registers globally.
3. Step S3 replaces the registers iteratively (S3a through S3d).

Register shifting to the periphery (step S1)

In a more detailed view, only the idea to move registers after the PIs and/or before the POs in order to enlarge the potential for combinational optimization is common in both approaches Peripheral Retiming and EPR. But unlike Peripheral Retiming, EPR does not require the numbers of registers at the periphery. EPR only detects efficiently, if such a register shifting is possible. If this detection succeeds, the circuit is proven to have an extended pipeline structure and consequently, our approach can be performed successfully. Note that the information about the sequential properties of the circuit is stored in the path weight matrix PWM which can be created tracing the circuit paths between PIs and POs. For the following computation steps in EPR it is enough to consider only the combinational circuit (where all registers are removed) and PWM as the fundamental.

Figure 2 Comparison of Peripheral Retiming and EPR

Global optimization (step S2)

Different to Peripheral Retiming, EPR controls by parameters whether the global optimization should be performed or not. Additionally to the classical approach, EPR is able to handle pseudo dependencies with negative path weight which may result from the optimization. The related backtracking technique is quite simple. After having restored the circuit to the state before optimization there are two types of further treatment (controlled by parameters):

- No global combinational optimization is performed but local optimization will be done in the further EPR steps.
- The optimization is repeated using another optimization method (e.g. within SIS a lot of different optimization scripts are available or you can draw up your individual scripts).

Register replacement (steps S3a through S3d)

In EPR, register replacement is determined by the idea to cope with the inherent complexity of this process by an iterative approach, by partitioning techniques, and by heuristics. Combining different optimization techniques gains different optimization goals simultaneously. Goal is not only to fit the given clock cycle constraints, but also to consider the related area penalties.

At the start of each iteration of register replacement, a special partition of the circuit called SCP (selected circuit part) has to be selected (step S3a of Figure 2). To compute SCP, in a first step all paths of the circuit which currently do not suffice PWM have to be chosen.

Next, SCP will be reduced further by considering limitation of pipeline stages and timing constraints as shown in step S3b of Figure 2. This will be done in two steps:

- Based on the pipeline stage of each PI and each PO as well as considering the relative position of each gate to the pipeline stages of neighboring PI and PO (see Figure 1), we compute the range of possible pipeline stages for each gate. This range is given by an early pipeline level and a late pipeline level.
- Based on the delay of each gate and the relating arrival and required times of input and output signals of these gates, this range is to be restricted further.

All gates having the same range are comprised to a circuit part called CSCP (constraint selected circuit part). In summary, only such circuit parts remain in a CSCP which have to carry at least one register on each of their paths to meet the timing constraints.

During CSCP computation it can be decided whether pipelining is necessary to fit the timing constraints or not. Pipelining is necessary if the number of required registers on a particular path exceeds its capacitance in the current PWM. In this case, the restrictions resulting from the delay of a gate are not compatible to the restrictions resulting from the pipeline stages of the neighboring PI and PO. Treatment of pipelining in the further running of our approach is very simple. It is enough to increment PWM but all other computation steps are not influenced.

In the partitioning process, usually more than one circuit part fit the stated properties of a CSCP. Therefore, the most promising one is selected using heuristics. Two different heuristics provided good results in experiments. Controlled by parameters it can be selected whether to choose the circuit part with the greatest number of gates or the circuit part with the fewest gates on its critical path.

Only for this selected circuit part an optimal register position will be computed. Based on ideas of (Münzner and Hemme, 1991) and related extensions in (Fehlauer, Rülke and Franke, 1993), we exploit a modified MaxFlow algorithm for this purpose. As a precondition, the selected circuit part has to be transformed into a weighted MaxFlow graph MFG. The modified MaxFlow algorithm cuts MFG completely on a minimal number of edges. This cut represents the optimal register position. The problem to compute the weights of particular edges in MFG will be the objective of an other paper and will not be handled here in detail. It depends on both, the type of the connection which is represented by the edge as well the goal to cut MFG completely by the algorithm used.

According to the determined optimal register position, the related registers have to be inserted in the circuit. Afterwards, combinational timing and/or area optimization can be applied locally for all combinational blocks as shown in step S3c of Figure 2. If pseudo dependencies occur for a particular combinational block, the handling (step S3d of Figure 2) is similar to that in the global optimization step explained above.

The current iteration of register replacement is closed by correcting PWM considering the included registers and possibly the changed (removed or added) paths.

As mentioned above, a couple of options ensure the flexibility of EPR. Briefly summarized, there are options to maintain the global optimization, to handle pseudo dependencies, to control partitioning, and to select heuristics.

4 EXPERIMENTAL RESULTS

Since there is a lack of sequential benchmarks having extended pipeline structure we use the following three different circuits to show results of our approach:

- *exam1* is a simple example with 8 pipeline stages. The 9 PIs as well as the 6 POs are connected to different stages of the pipeline. All combinational blocks in the particular pipeline stages are arbitrary combinations of ANDs, ORs, and INVs.
- *exam2* is the VHDL description of a real designed controller. For experiments *D* and *F*, first we cut all loops, then we optimized the circuit, and finally we connected the loops again.
- *exam3* is a cascade of unrolled FSM circuits. It was obtained using the MCNC benchmark *ex3* after having applied the following operations: a) cutting all loops behind the latches which results in additional inputs and outputs, b) tripling the resulting circuitry, c) connecting the additional outputs of one subcircuit with additional inputs of the other subcircuit.

To set up unique test conditions, we worked on Sun Sparc10 workstation using the Berkeley system SIS for logic optimization and the LSI Logic library LCA10k for timing analysis. Handling these examples in the unique environment, we performed the following experiments:

- *A*: no explicit optimization,
- *B*: combinational timing optimization,
- *C*: Retiming for minimal clock period,
- *D*: Peripheral Retiming (we have added a tool to SIS which performs Peripheral Retiming using routines and data structures of the SIS system),
- *E*: varying sequences of *A* through *D*,
- *F*: EPR with different options.

We analyzed the final timing of the particular circuits and we compared clock cycle (Clock), number of registers (#REGs), and combinational area cost (CAC). The results of our experiments are comprised in Table 1. To illustrate experiment *E*, we show the results of applying *C* behind *B*.

Table 1 Experimental Results

		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
<i>exam1</i>	Clock	5.20	3.78	4.95	4.31	3.78	3.51
	#REGs	13	13	10	10	12	10
	CAC	43	40	40	32	21	19
<i>exam2</i>	Clock	11.55	11.59	10.93	8.01	7.63	8.67
	#REGs	30	30	30	51	71	52
	CAC	528	503	526	259	259	184
<i>exam3</i>	Clock	8.28	8.02	7.25	9.59	8.02	8.14
	#REGs	7	7	15	9	7	8
	CAC	189	189	190	213	189	134

Table 1 shows the benefits of EPR. As expected, EPR produces the best trade-off between permissible clock cycle, required register count, and inevitable combinational area costs. Using EPR area and/or computation time can be saved. In some cases only EPR and no other sequential optimization (Retiming, Peripheral Retiming) was able to master the large complexity of the optimization task. Moreover, using other control parameters, EPR is able to produce faster designs - may be by generating additional pipeline stages or by combining EPR with Retiming. E.g. using one further pipeline stage for *exam1*, the results are 2.81 / 17 / 19 (Clock / #REGs / CAC). Combining EPR with Retiming, the results for *exam2* are 6.96 / 64 / 184.

5 CONCLUSION

Summary

We have presented a new approach to optimize sequential circuits. EPR combines sequential and combinational techniques to speed up a circuit at gate level considering area penalties. Working iteratively and heuristically, EPR is applicable to large circuits. Experimental results have shown the practicability of our approach.

Future Research

EPR may be improved by extending it into several directions. Incorporating further performance optimization techniques (e.g. repowering and buffering) may lead to better combinational optimization results. Considering false paths may enhance the efficiency of our algorithm significantly. Also, the iterative manner of EPR allows for looking at don't-care conditions which may result in an improved combinational optimization potential. Moreover, since high-level synthesis is getting more and more practicable, EPR could be applied to simplify timing optimization of extended pipelining structures resulting from high-level synthesis.

6 REFERENCES

- Fehlauer, E., Rülke, St. and Franke, G. (1993) Design Improvement by Combining Sequential and Combinational Techniques. Proceedings of the Third PATMOS Workshop, 21 - 29.
- Leiserson, C., Rose, F. and Saxe, J. (1983) Optimizing Synchronous Circuits by Retiming. Proceedings of the Third Caltech Conference on VLSI, 87-116.
- Martin, H.-G. (1993) Retiming by Combination of Relocation and Clock Delay Adjustment. Proceedings of the EURO-DAC, 384-389.
- DeMicheli, G. (1994) Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York.
- Malik, S. et.al (1991) Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques. IEEE Transactions on CAD, 74-91.
- Münzner, A. and Hemme, G. (1991) Converting Combinational Circuits into Pipelined Data Path. Proceedings of the ICCAD, 368-371.

7 BIOGRAPHY

Dr.-Ing. Erhard Fehlauer

Diplom-Ingenieur (Information Technology) from Technological University of Dresden (TUD) in 1984; 1984-1988: Assistant Professor with TUD; Ph.D. Degree (Information Technology) from TUD in 1988; Present: Research Assistant with FhG IIS / EAS Dresden; Current research interests: high level and logic synthesis, esp retiming

Dr.-Ing. Steffen Rülke

Diplom-Ingenieur (Information Technology) from Technological University of Dresden (TUD) in 1981; 1981-1991: Research Assistant with Academy of Science; Ph.D. Degree (Applied Informatics) from Academy of Science in 1990; Present: Research Assistant with FhG IIS / EAS Dresden; Current research interests: high level and logic synthesis, esp retiming

Dr.-Ing.habil. Günter Franke

Diplom-Ingenieur (Electronical Engineering) from Technological University of Dresden (TUD) in 1968; 1969-1991: Research Assistant with Academy of Science; Ph.D. Degree (Control Engineering) from TUD in 1972; Present: Research Assistant with FhG IIS / EAS Dresden; Current research interests: high level and logic synthesis, HW/SW-codesign