Completing the Resynthesis Flow for Balsa Circuits by Focusing on the Data Path - first Experiments

Norman Kluge
Hasso-Plattner-Institut
University of Potsdam, Germany
norman.kluge@hpi.uni-potsdam.de

Ralf Wollowski
Hasso-Plattner-Institut
University of Potsdam, Germany
ralf.wollowski@hpi.uni-potsdam.de

I. Introduction

Balsa [3] provides a development flow, where asynchronous circuits are created from high-level specifications. The Balsa compiler converts a high-level Balsa programme into a network of handshake (HS-)components. These basic building blocks form the circuit and communicate via asynchronous handshakes. But the syntax-driven translation used by the Balsa compiler often results in performance overhead. To reduce the performance penalty, STG-based resynthesis has been introduced e.g. in [6] [7] [2], focusing on the optimisation of the control path. In [6] only HS-components modelling pure control were resynthesised. In [7] and [2], also mixed HS-components (consisting of control *and* data path) were considered. In contrast, in [7] the control of *all* components used by the Balsa compiler were resynthesised. However, to get simulation results for the entire circuit the data path has to be added. This is not trivial due to the mixed HCs because the control has to be separated from the data path – hence an adjusted data path has to be constructed. Here we follow the most comprehensive approach of [7] and present very first promising results with a completed resynthesis tool.

Fig. 1 shows the overall design flow. Starting with a Balsa programme, the Balsa compiler generates a network of HS-components (syntax-driven translation), which is given to our resynthesis tool producing a verilog netlist. The Synopsys Design Compiler is used then to compute a delay file.

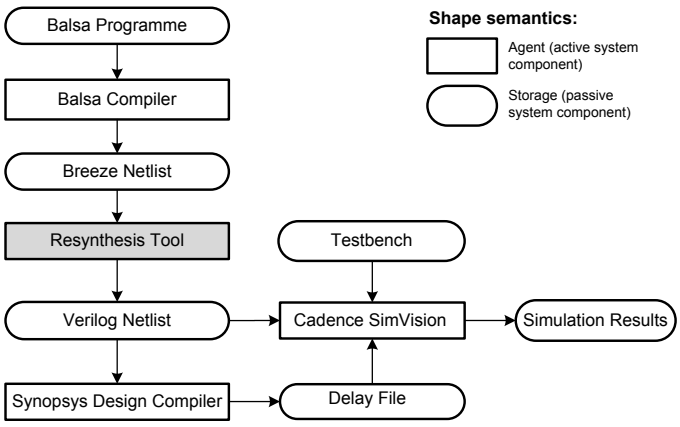

Fig. 1. Overall design flow

This research was supported by DFG-project 'Optacon' VO 615/10-1 and WO 814/3-1.

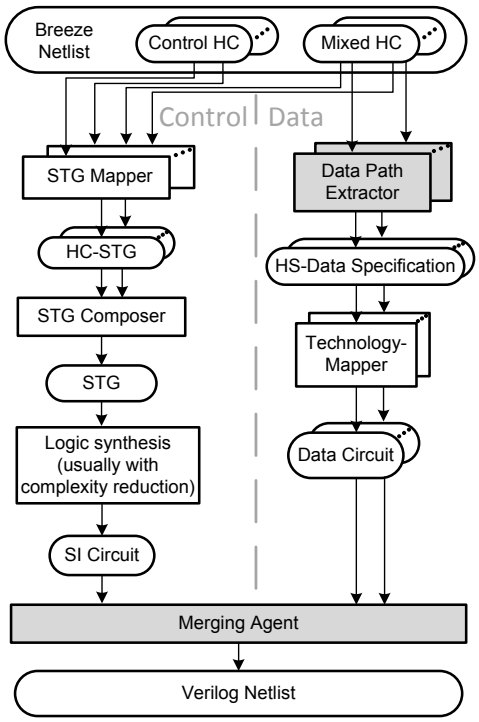

Fig. 2. Resynthesis flow (added new parts in grey)

Finally, a testbench, the verilog netlist, and the delay file is passed to Cadence SimVision to simulate the entire circuit.

Fig. 2 shows the internal flow of the resynthesis tool with control synthesis on the left-hand side. The control part of every handshake component is mapped onto a HC-STG. All HC-STGs are composed to a single STG (e.g. using PCOMP [1]). Next, logic synthesis is applied (e.g. using petrify [4]), usually suffering from state space explosion. To cope with this complexity problem, we generally use STG-decomposition [8]; but for our first experiments presented in this work, this is not necessary due to the simplicity of the used benchmarks.

II. Adding the data path

The synthesis flow for the data path is shown on the right-hand side in Fig. 2. For every mixed handshake component (consisting of control and data path) a data path component which properly communicates with the control of the circuit has to be generated (component "Data Path Extractor").

TABLE I. RESULTS: SIMULATION TIME (250NM)

	Balsa	Resynthesis (with petrify logic synthesis)			
		Petrify reset		Petreset reset	
gcd(127,1)	2.085,3 ns	1.810,3 ns	-13,2%	1.783,5 ns	-14,5%
counter{20}	216,1 ns	178,6 ns(∗)	-17,4%	176,1 ns	-18,5%
shifter(3)	38,0 ns	30,0 ns	-21,2%	30,1 ns(∗)	-20,8%
buffer{10}	19,8 ns	Sim DNF	N/A	19,2 ns	-3,1%

TABLE II. RESULTS: SIMULATION TIME (130NM)

	Balsa	Resynthesis (with petrify logic synthesis)			
		Petrify reset		Petreset reset	
gcd(127,1)	1.413,6 ns	1.067,1 ns	-24,5%	No solution	N/A
counter{20}	148,0 ns	95,2 ns(∗)	-35,7%	No solution	N/A
shifter(3)	24,3 ns	17,6 ns	-27,8%	No solution	N/A
buffer{10}	13,3 ns	12,2 ns(∗)	-8,3%	No solution	N/A

(∗) Sometimes "Sim DNF" due to not properly working reset logic

For the construction of the data paths, a new Balsa style was created. The implementation is oriented on the Balsa four phase bundled data style. All the logic producing control was deleted, and new signals for the communication with the control part were added.

III. FIRST RESULTS

The results of our first experiments with the entire design flow are listed in Tables I and II. Only the simulation times are shown, because area and power consumption results are not present yet. For the moment, we used only a set of four simple benchmarks (we will experiment with complex Balsa benchmarks soon). We have performed the simulation with two technology libraries, a 250 nm (Table I) and a 130 nm (Table II) technology. In each Table a Balsa solution is compared with two resynthesis solutions using different tools for inserting the reset logic (petrify itself and Petreset [5]). As already mentioned, the benchmarks are not complex, meaning they don't suffer from state space explosion and petrify could solve the CSC problem for the STGs as a whole. The performance improvement for the resynthesis solution is up to 35%, the average is around 19% compared to the original Balsa implementation. However, the simulation sometimes did not finish ("Sim DNF") due to not properly working reset logic. Even the simulations of different synthesis runs on the same benchmark could produce different results, i.e. sometimes they finish successfully and sometimes not ("(∗)").

IV. RESET PROBLEM

To simulate a circuit successfully, it is crucial to have a working reset logic. As shown above, the reset insertion in our design flow is up to now not reliable and depends on the technology used. We are aware of three tools able to solve the reset problem: petrify, Petreset and the tool presented in [9]. Unfortunately, we do not have access to the latter one yet. The other two tools fail in some cases (cf. Tables I and II). So it looks like that there is no general solution or tool implementation for the reset insertion for asynchronous circuits yet.

TABLE III. RESULTS: COMPARATOR OPTIMISATION (250NM)

	Original	Optimised	
cmp(1,0)	3,22 ns	3,19 ns	-0,9%
cmp(1,127)	1,63 ns	1,43 ns	-12,3%
cmp(-128,127)	1,63 ns	1,16 ns	-28,8%
cmp(0,-128)	2,03 ns	1,16 ns	-42,9%

All three tools mentioned add the reset logic after the circuits' synthesis from the STG specification. Maybe it would be a better approach to consider the reset issue earlier, e.g. by inserting the transitions of the reset signal properly into the state graph?

V. DATA PATH OPTIMISATIONS

Up to now, we have focused on the optimisation of the control part only, but how about optimising the data path? Is it possible to find faster algorithms for some HS-components? First experiments with the Balsa comparator component ("BinaryFunc_greaterThan") have shown that a speed up is possible (cf. Table III).

Basically, the separation of control and data path of each HS-component offers the possibility to optimise the data parts by their own. Maybe it is possible to optimise the data paths of suitable handshake circuits *altogether*, e.g. using logic minimisation for the combined combinational logic.

ACKNOWLEDGMENT

We thank the IHP GmbH (Frankfurt/Oder) for providing the design environment and the design kits, in particular Milos Krstic and Steffen Zeidler for their helpful advices. Best thanks to Walter Vogler and Stanislavs Golubcovs for fruitful cooperation making this work possible.

REFERENCES

[1] A. Alekseyev, V. Khomenko, A. Mokhov, D. Wist, and A. Yakovlev, "Improved parallel composition of labelled Petri nets," in *Proceedings of the 2011 Eleventh International Conference on Application of Concurrency to System Design*, ser. ACSD '11, 2011, pp. 131–140.

[2] A. Alekseyev, I. Poliakov, V. Khomenko, and A. Yakovlev, "Optimisation of balsa control path using stg resynthesis," 21st UK Asynchronous Forum, 2009.

[3] A. Bardsley and D. A. Edwards, "The Balsa asynchronous circuit synthesis system," in *Forum on Design Languages*, Sep. 2000.

[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. E80-D, no. 3, pp. 315–325, 1997.

[5] I. David, L. Cohen, and R. Dobkin, "Petreset," 2010, unpublished.

[6] F. Fernández-Nogueira and J. Carmona, "Integrated circuit and system design. power and timing modeling, optimization and simulation," L. Svensson and J. Monteiro, Eds., 2009, ch. Logic Synthesis of Handshake Components Using Structural Clustering Techniques, pp. 188–198.

[7] S. Golubcovs, W. Vogler, and N. Kluge, "STG-based resynthesis for balsa circuits," in *Proceedings of the 2013 13th International Conference on Application of Concurrency to System Design*, ser. ACSD '13, 2013, pp. 140–149.

[8] M. Schaefer, "DesiJ - a tool for STG decomposition," Universitaet Augsburg, Tech. Rep., 2007.

[9] V. S. Vij and K. S. Stevens, "Automatic addition of reset in asynchronous sequential control circuits." in *VLSI-SoC*, M. Margala, R. A. da Luz Reis, A. Orailoglu, L. Carro, L. M. Silveira, and H. F. Ugurdag, Eds. IEEE, 2013, pp. 374–379.