cases. These results show that our methods can perform circuit customization efficiently and effectively. In our experience, the most useful properties are the signals proven to be constant because they always simplify part of the logic in the circuit. We also observed that these properties are valid due to the constraints from the testbench instead of the design itself because when running the testbench that allows all possible instructions, these properties do not exist. This result shows that our methodology can successfully mine properties from the testbench and utilize them for design optimization.

In Table II we also show the number of applied properties for each set of instructions. Note that set $G1$ only contains NOP (no operation implemented using left-shift by 0 bits) and is not practical. However, it shows an intereseting conrner case — 97.6% logic can be removed when most functionality of the circuit is not used.

In our second experiment, we applied our techniques to DLX designs which were already optimized by [3]. The results are summarized in Table III. Compared with the runtime in Table II, runtime in this experiment became smaller because the circuits have been optimized already, thus there were few pairs of variables to check and fewer properties to extract. However, the results show that even though the designs were already optimized, we could still find optimizations that further reduce circuit area, suggesting that our optimizations are orthogonal to those provided by [3] and can provide additional area reduction. To identify where the additional optimizations were from, we performed a more detailed analysis, and the results are shown in Table IV. From the results, we can see that our methods reduced some combinational logic. However, most of the area reduction was from sequential logic. This is an area that [3] could not perform well: their methods are based on code reachability analysis and tend to optimize combinational logic, while ours can optimize both combinational and sequential portions of the circuit.

TABLE III
DLX OPTIMIZED USING OUR METHODOLOGY AFTER OPTIMIZATIONS FROM [3] IS PERFORMED.

Inst. Group	Run time (s)	#Properties	Opt. by [3]		Further opt. by this work		Slack time (ns)
			Area (μm^2)	Reduction ratio	Area (μm^2)	Reduction ratio	
G1	0.20	174	7429.5	94.4%	5407.9	95.9%	8.07
G2	2.85	39	101353.4	24.0%	99552.5	25.4%	3.32
G3	3.37	24	116295.6	12.8%	108445.2	18.7%	3.41
G4	18.01	19	124339.6	6.8%	123226.1	7.6%	0.75
G5	34.49	19	130526.6	2.2%	129243.4	3.1%	0.12

TABLE IV
COMPARISON OF OPTIMIZATIONS ACHIEVED BY [3] AND OUR WORK.

	Area after opt. from [3] (μm^2)		
Inst. Group G1	Comb. logic	Seq. logic	Total
	52310.4	63985.2	116295.6
	Area after further opt. by our methodology (μm^2)		
	Comb. logic	Seq. logic	Total
	50329.6	58115.6	108445.2

Runtime comparison between our methods and [3] shows that our runtime (within two minutes) is much shorter than that in [3] (over an hour for most cases). This comparison shows that our methodology based on properties is more efficient than code reachability analysis. On the other hand, a comparison of area reduction between Table II and III shows that utilizing either one of the optimization methods cannot achieve what the combined methods provide. Therefore, it is best to apply both techniques so that designs can be better optimized.

VI. CONCLUSION

Synthesis tools have evolved to a point where most designs can be synthesized and optimized efficiently and effectively. However, most tools only utilize design intention represented in the RTL code and do not consider verification intention encoded in the verification constructs such as testbenches or assertions. Not being able to utilize the information from verification environment greatly limits optimization capabilities of synthesis tools. This problem is especially serious for circuit customization because most environment constraints are found in the testbenches. To address this problem, we proposed a methodology that utilizes functional assertions for design optimization. In addition, we proposed a new technique that extracts assertions from the design under the constraints in the testbench. Experimental results show that our methods can reduce design sizes after synthesis, and the optimization is orthogonal to another optimization based on reachability analysis [3]. These results show that our techniques can help designers better address the circuit customization problem. In the future, we plan to enhance our property miner to recognize more types of properties. By utilizing more properties, more design optimization opportunities can be exposed and utilized.

ACKNOWLEDGMENT

This research is supported by the Institute for Information Industry, Taiwan under Grant 99-FS-C02.

REFERENCES

[1] K.-H. Chang, V. Bertacco, I. L. Markov and A. Mishchenko, "Logic Synthesis and Circuit Customization Using Extensive External Don't-Cares," *ACM Transactions on Design Automation of Electronic Systems,*,Vol. 15, No. 3, Article 26, 2010

[2] K.-H Chang, V. Bertacco, and I. L. Markov, "Customizing IP Cores for System-on-Chip Designs Using Extensive External Don't-Cares," *DATE'09,* pp. 582-585.

[3] H.-Z. Chou, K.-H. Chang and S.-Y. Kuo, "Optimizing Blocks in an SoC Using Symbolic Code-Statement Reachability Analysis" *ASPDAC'10,* pp. 787-792.

[4] S. Vasudevan, D. Sheridan, S. Patel, D. Tcheng, B. Tuohy and D. Johnson, "GoldMine: Automatic assertion generation using data mining and static analysis" *DATE'10,* pp. 626 - 629.

[5] P.-H. Chang and L.-C. Wang, "Automatic assertion extraction via sequential data mining of simulation traces" *ASPDAC'10,* pp. 607-612.

[6] L.-C. Wang, M. S. Abadir and N. Krishnamurthy, "Automatic generation of assertions for formal verification of PowerPC microprocessor arrays using symbolic trajectory evaluation" *DAC'98,* pp. 534-537.

[7] R. E. Bryant, "Symbolic Simulation – Techniques and Applications," *DAC'90*, pp. 517-521

[8] A. Kolbl, J. Kukula and R. Damiano, "Symbolic RTL Simulation," *DAC'01*, pp. 47-52.

[9] A. Kolbl, J. Kukula, K. Antreich and R. Damiano, "Handling Special Constructs in Symbolic Simulation," *DAC'02*, pp. 105-110.

[10] H.-Z. Chou, I.-H. Lin, C.-S. Yang, K.-H. Chang and S.-Y. Kuo, "Enhancing Bug Hunting Using High-Level Symbolic Simulation," *GLSVLSI'09*, pp. 417-420.

[11] Insight, http://www.avery-design.com

[12] Bug UnderGround, http://bug.eecs.umich.edu

[13] http://www.cadence.com