A Scalable Black-Box Optimization System for Auto-Tuning VLSI Synthesis Programs

Matthew M. Ziegler[1], Hung-Yi Liu[2*], George Gristede[1], Bruce Owens[3], Ricardo Nigaglioni[4], and Luca P. Carloni[2]
[1] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
[2] Department of Computer Science, Columbia University, NY, USA
[3] IBM Systems & Technology Group, Rochester, MN, USA
[4] IBM Systems & Technology Group, Austin, TX, USA

***Abstract*— Modern logic and physical synthesis tools provide numerous options and parameters that can drastically impact design quality; however the large number of options leads to a complex design space difficult for human circuit designers to navigate. We tackle this parameter tuning problem with a novel system employing intelligent search strategies and parallel computing, thus automating one of the key design tasks conventionally performed by a human designer. We provide an overview of this system, called SynTunSys, as well as results from employing it during the design of the IBM z13 22nm high-performance server chip, currently in production. During this major design, SynTunSys provided significant savings in human design effort and achieved a quality of results beyond what human designers alone could achieve, yielding on average a 36% improvement in total negative slack and a 7% power reduction.**

I. Introduction

The design of modern high-performance processors is a quest to optimally tune and balance multiple objectives, such as performance, power, and reliability. This multi-objective design space is further complicated by the need for more complex VLSI (very-large-scale integration) chips to fuel the ever increasing desire for more compute power. To cope with this design complexity, the VLSI design community has leveraged CAD (computer-aided design) tools for many decades now; however, the high flexibility and sophistication of advanced synthesis tools increases their complexity and makes navigating the design space difficult and sometimes non-intuitive for their users.

The industrial synthesis tool-flow we employ has over 1000 parameters [1]. These parameters span the logic and physical synthesis space and the control settings for modifying the synthesis steps, such as: logic decomposition, technology mapping, placement, estimated wire optimization, power recovery, area recovery, and/or higher effort timing improvement. The parameters also vary in data type (Boolean, integer, floating point, and string). Considering that an exhaustive search of only 20 Boolean-type parameters leads to over one million combinations, and synthesis runs may take several hours or even days, it is clear that intelligent search strategies are required.

As an example of the wide design space available from modifying synthesis parameters, Fig. 1 shows the scatter plot of achievable design points for a portion of a synthesized floating-point multiplier macro. A macro may span from 1K to 1M gates in our context. Each point denotes the timing and power values achieved simply by tuning the input parameters of the synthesis program. The ultimate goal of this process is to reach timing closure at the lowest achievable power.

[*] Hung-Yi Liu is now with the Intel Design Technology & Solutions Group, Hillsboro, OR, USA.

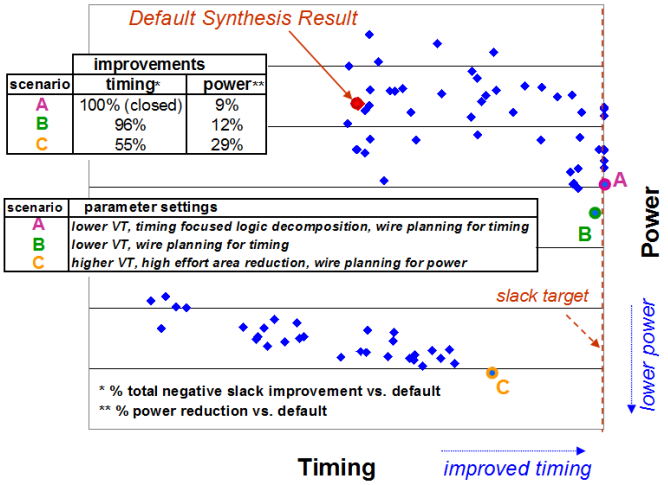

Fig. 1. An example of the available design space by modifying synthesis parameters.

Quite often the default values for the parameters are not ideal for a specific macro, which would benefit instead from parameter customization. Fig. 1 also highlights three scenarios (A, B, and C) along the Pareto frontier. These scenarios show the available tradeoffs between timing closure and power reduction, e.g., point A closes timing with a 9% power reduction, whereas point C improves timing by 55% with a 29% power reduction. These points along the Pareto set provide a number of potential steps towards the ultimate goal, depending on the additional techniques at the designer's disposal beyond parameter tuning. This example of a relatively simple macro underscores how significantly the parameters settings can affect a design.

II. Related Work

The synthesis parameter tuning problem we address can be classified as a black-box optimization problem, i.e., we treat the synthesis program as black-box software by supplying input conditions (input data and parameter settings) and measuring the output response in terms of synthesis quality of results (QoR). Black-box problems are often approached using techniques from the field of simulation optimization [2], which is an umbrella term for optimization techniques that operate in the absence in of an algebraic model of the system. Considering each macro exhibits a unique input-output response to synthesis parameter settings and digital logic can take on an intractable number of functionalities, the synthesis tool-flow of our focus is far too complex to be modelled algebraically. Furthermore, our synthesis program often exhibits non-deterministic behavior, which is also characteristic of many simulation optimization problems.

Black-box optimization techniques can also be employed for design-space exploration (DSE) purposes, however, unlike convention

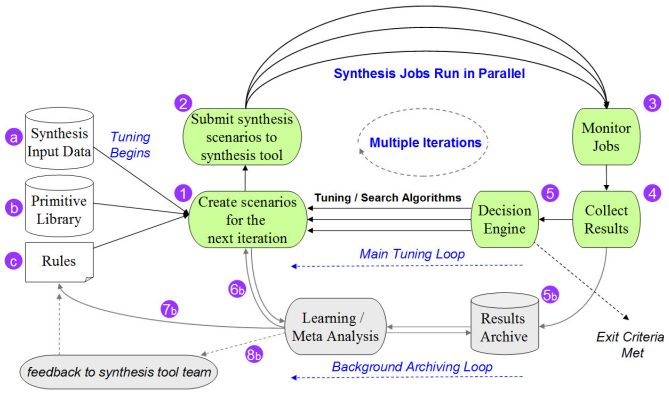

Figure 2. Architecture of the SynTunSys process, which employs a parallel and iterative tuning process to optimize macros.

DSE, the goal of black-box optimization is often to find one or more optimal or near-optimal design points without necessarily requiring a complete exploration of the design space to determine the whole Pareto frontier of design tradeoff points.

Black-box optimization is a common problem seen across a number of fields, e.g., compiler tuning [3] and software engineering [4]. With respect to VLSI design, DSE is becoming a more attractive solution for complex problems across various levels of abstraction. At the architectural level, many DSE studies based on models or simulators have been used to explore multi-objective design spaces, e.g., [5]. Architectural-level studies, however, typically do not result in implemented designs. DSE approaches have also been applied in combination with high-level synthesis by a number of researchers, e.g., [6]. FPGA synthesis parameter tuning has been reported in [7] using genetic algorithms and in [8] using Bayesian optimization. However, prior to our work [9], we know of no publications targeting automated parameter tuning for logic and physical synthesis.

III. System Architecture

The framework of our parameter tuning system is shown in Fig. 2. SynTunSys consists of a main tuning loop that constructs synthesis scenarios consisting of synthesis parameter settings (Step (1)), submits and monitors synthesis jobs (2-3), analyzes the results (4), and iteratively refines the solutions (5). A second background loop archives the results of all runs from all macros, users, and projects, which can be mined for historical trends across multiple macros and to provide feedback in terms of the performance of synthesis parameters.

The SysTunSys cost function conveys the optimization goals. It converts multiple design metrics into a single cost number, allowing cost ranking of scenarios. Examples of available metrics include: multiple timing metrics, power consumption, congestion metrics, area utilization, electrical violations, runtime, etc. The selected metrics are assigned weights to signify their relative importance. The overall cost function is then a "normalized weighted sum" of the *m* selected metrics, expressed by Equation (1) where *Norm(Mi)* is the normalized *Mi* across all the scenario results in a SynTunSys run.

$$Cost = \sum_{i=1}^{m} W_i \cdot Norm(M_i) \qquad \begin{array}{l} where: \\ W_i = weight_i \\ M_i = metric_i \end{array} \quad (1)$$

The SynTunSys decision engine algorithms are key components of the system that determine which scenarios should be run at each iteration, i.e., the decision engine tackles the parameter tuning black-box optimization problem discussed previously. The decision engine can also be upgraded independently and we constantly look to improve these algorithms in terms of QoR prediction accuracy and compute efficiency. Similar black-box tuning problems have been approached using a number of techniques, e.g., machine learning [10], Markov decision processes [11] and Bayesian optimization [8,12]. During the development of SynTunSys we have explored algorithms ranging from pseudo-genetic algorithms to adaptive learning.

Table 1. Average parameter tuning improvements over best known prior solution for a 22nm processor in production.

Pre / Post SynTunSys Comparison	Latch-to-Latch Slack	Total Negative Slack	Total Power
Improvement	60%	36%	7%
Sum of 200	(ps)	(ps)	(arb. units)
pre-tuning	-1929	-2150385	17770
post-tuning	-765	-1370731	16508

IV. SynTunSys Results

SynTunSys was used during the design of the IBM z13 22nm server processor [13]. The processor underwent two chip releases (tapeouts) over a multi-year design cycle, during which SynTunSys was applied to macros over both releases. The chip consists of a few hundred macros that average around 30K gates in size, with larger macros in the 300K gate range. Prior IBM server processors have also used systematic parameter tuning on a smaller scale. The IBM POWER7+ [14] and POWER8 [15] processors employed an earlier version of SynTunSys during the second chip releases, but mainly for power reduction purposes. In the case of the z13 processor, however, SynTunSys was employed during the entire design cycle for timing closure, power reduction, and improving macro routability.

Based on the efforts of a dedicated tuning team we were able to track SynTunSys results on approximately 200 macros from the processor core. Table 1 shows the average improvements achieved by SynTunSys over the best solution previously achieved by the macro owners for the first chip release.

Note that these results are based on the routed macro timing and power analysis; in most cases the best known prior solutions included manual parameter tuning by the macro owner. SynTunSys resulted in a 36% improvement in total negative slack, a 60% improvement in worst latch-to-latch slack (macro internal slack), and a 7% power reduction. The actual values of the metrics, summed across all the macros, underscores that the changes in the absolute numbers were significant, e.g., ~780,000 picoseconds of total negative slack was saved across ~200 macros.

References

[1] L. Trevillyan, et al., "An Integrated Environment for Technology Closure of Deep-Submicron IC Designs," IEEE Design & Test of Computers, vol. 21:1, pp. 14-22, 2004.

[2] S. Amaran, et al., " Simulation Optimization: A Review of Algorithms and Applications," 4OR - A Quarterly Journal of Operations Research, Dec. 2014.

[3] G. Fursin, et al., "Milepost GCC: Machine Learning Enabled Self-tuning Compiler," International Journal Parallel Programming, 39:296-327, 2011.

[4] A. Arcuri, G. Fraser, "Parameter Tuning or Default Values? An Empirical Investigation in Search-Based Software Engineering," Empirical Software Engineering, June 2013, Volume 18, Issue 3.

[5] O. Azizi, et al., "An Integrated Framework for Joint Design Space Exploration of Microarchitecture and Circuits," DATE 2010.

[6] S. Xydis, et al., "A Meta-Model Assisted Coprocessor Synthesis Framework for Compiler/Architecture Parameters Customization," DATE 2013.

[7] M. K. Papamichael, P. Milder, J. C. Hoe, "Nautilus: Fast Automated IP Design Space Search Using Guided Genetic Algorithms," DAC 2015.

[8] N. Kapre, et al., "Driving Timing Convergence of FPGA Designs through Machine Learning and Cloud Computing," FCCM 2015.

[9] M. M. Ziegler, et al., "A Synthesis-Parameter Tuning System for Autonomous Design-Space Exploration," DATE 2016.

[10] H.-Y. Liu and L. P. Carloni, "On Learning-Based Methods for Design-Space Exploration with High-Level Synthesis," DAC 2013.

[11] G. Beltrame, et al., "Decision-Theoretic Design Space Exploration of Multiprocessor Platforms," IEEE TCAD, 29(7):1083–1095, July 2010.

[12] Z. Wang, et al., "Bayesian Optimization in High Dimensions via Random Embeddings," Int'l Joint Conf. on Artificial Intelligence (IJCAI-13), 2013.

[13] J. D. Warnock, et al., "22nm Next-Generation IBM System z Microprocessor," ISSCC 2015.

[14] M. M. Ziegler, G. D. Gristede, V. V. Zyuban, "Power Reduction by Aggressive Synthesis Design Space Exploration," ISLPED 2013.

[15] M. M. Ziegler, et al., "POWER8 Design Methodology Innovations for Improving Productivity and Reducing Power," CICC 2014.