Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

Optimization Principles for Arithmetic Functions in Hardware-Software Co-Design

Vladimir Delengov, Yuan Li,
Jennifer Thompson, Lukas Kroc
Claremont Graduate University
Claremont, CA 91711

Nandakishore Santhi
Stephan Eidenbenz
Los Alamos National Laboratory
Los Alamos, NM 87544

ABSTRACT

As traditional hardware scaling laws have started to break down, *Co-Design of hardware and software* has become the most promising avenue towards exa-scale computing. We present a bottom-up approach as part of a larger project that develops an optimization framework for computational codesign for molecular dynamics applications. Our approach finds optimum circuit designs for arithmetic functions, such as square root or multiplication, which are the basic building blocks of the domain-specific arithmetic calculations in molecular dynamics simulations. Our design approach employs the Boolean satisfiability problem (SAT) as a vehicle for circuit design, using state-of-the-art SAT solvers that show their algorithmic power on mid-range performance computing platforms to rein in the inevitable combinatorial explosion of possible circuit designs as we increase the bit-length of our operations. While the main emphasis is on the modeling methodology, we show initial results of automated designs for a 4-bit square root circuit and a mini-calculator.

1 MODELING HARDWARE/SOFTWARE CODESIGN AS COMBINATORIAL OPTIMIZATION

When exploring the hardware and software design spaces for a given application domain, such as *molecular dynamics* simulations, co-design requires performance prediction of hardware-software pairs, where one or the other component may not exist in real life because it simply has not been built yet, thus making a modeling and simulation approach a necessity. Different levels of detail ranging from pure compute node-connectivity models down to individual-gate level on the hardware side and from pseudo-code to machine code on the software side, deliver varying degrees of fidelity. We present a conceptual model of hardware and software in tandem on which we can explore optimization methods. We would operate on hardware and software search spaces that are unrestricted by standard structures and preconceived architectural designs in order to find new efficient systems.

Computational optimization is well-established in the field of developing circuit design with the most common approaches being genetic algorithms, circuit synthesis SAT and local search techniques. Kamath et al. (1993) proposes to use SAT in logical design synthesis. Estrada (2003) experiments with modern SAT solvers using this translation. Williams (2008) has noted that it is easy to construct optimal circuits for 2×2-matrix multiplication, while for 3×3 this becomes a difficult task. Kojevnikov et al. (2009) demonstrates finding efficient circuits using SAT-solvers. By using composite fitness functions in genetic algorithms, circuits can be produced which are logically correct and minimize the number of gates used (Kalganova and Miller 1999). Techniques for scaling are shown in Vassilev and Miller (2000) and Torresen (2003). Asynchronous circuits can also be evolved, as in Shanthi and Singaram (2005). There is very little prior work in optimizing hardware and software together. Jan Beutel and Iuliana Bacivarov (2011) is an exception, but is primarily focused on embedded systems.

We start by breaking down more complex problems to elementary polynomial operations. We then employ a systematic way to generate optimized hardware sets and corresponding sequence of software from such elementary operations. A hardware instruction is a circuit block that operates by a clock. All blocks are arrangements of generalized logical gates, which may have arbitrarily many input or output bits. We define a software function as a sequence of hardware instructions which delivers an answer to the

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

initial task upon execution. In this sequence any instruction can be called an unlimited number of times. A software function may be duplicated by hard wiring multiple instructions together. We seek to develop a method for comparing the speed and efficiency of possible computational systems. This we accomplish, by developing an objective function which depends on the logical gates, hardware instructions, software functions and lines of program code required to construct the system.

Our goal is to develop an optimization algorithm which satisfies the certain logic requirements and is executable on some scale. We take as input: target expressions (simple arithmetic functions), logic gates that may be reused arbitrarily, and cost functions that capture the desired circuit efficiency. We produce as output: A set of hardware instructions H and software functions S, one function per target expression, performance data. We synthesize the circuit blocks by encoding the specification into Boolean variables. We then use off-the-shelf satisfiability solvers to find solutions which correspond to physical circuits. Finally, we extend to encoding specification to include the software layer.

2 RESULTS

We implemented an algorithm to generate a SAT-based model given a logic functionality as a truth-table and used MiniSat 2.0 as the SAT solver. We constructed several test circuit blocks in reasonable computational time: (1) a conventional task having a well-known solution: XOR gate using 4 NAND gates. (2) a NOT gate using a XOR gate and a constant gate: the whole truth table is represented by just one number. (3) a 2-bit full-adder built using NAND gates: The solver took 13 minutes to find a solution requiring 16 gates, with one redundant gate. It took 55 minutes to reduce this to 14 NAND gates. Running the SAT solver for a further 15 hours produced neither a smaller circuit nor proof of optimality. (4) a 2×2-bit multiplier built from AND and XOR gates: the SAT solver achieved a solution with 5 AND and 2 XOR gates within a second. We will show more details and circuit designs for more complex problems on the poster.

REFERENCES

Jan Beutel and Iuliana Bacivarov 2011. "Lecture Notes in Hardware-Software Codesign". University Lecture.

Estrada, G. 2003. "A Note on Designing Logical Circuits Using SAT". In *Evolvable Systems: From Biology to Hardware*, edited by A. Tyrrell, P. Haddow, and J. Torresen, Volume 2606 of *Lecture Notes in Computer Science*, 410–421. Springer Berlin / Heidelberg. 10.1007/3-540-36553-2_37.

Kalganova, T., and J. Miller. 1999. "Evolving More Efficient Digital Circuits by Allowing Circuit Layout Evolution and Multi-Objective Fitness". In *Proceedings of the 1st NASA/DOD workshop on Evolvable Hardware*, EH '99, 54–. Washington, DC, USA: IEEE Computer Society.

Kamath, A., N. Karmarkar, K. Ramakrishnan, and M. Resende. 1993, aug. "An interior point approach to Boolean vector function synthesis". In *Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on*, 185 –189 vol.1.

Kojevnikov, A., A. Kulikov, and G. Yaroslavtsev. 2009. "Finding Efficient Circuits Using SAT-Solvers". In *Theory and Applications of Satisfiability Testing - SAT 2009*, edited by O. Kullmann, Volume 5584 of *Lecture Notes in Computer Science*, 32–44. Springer Berlin / Heidelberg. 10.1007/978-3-642-02777-2_5.

Shanthi, A. P., and L. K. Singaram. 2005. "Evolution of Asynchronous Sequential Circuits". In *Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware*, 93–96. Washington, DC, USA: IEEE Computer Society.

Torresen, J. 2003. "Evolving multiplier circuits by training set and training vector partitioning". In *Proceedings of the 5th international conference on Evolvable systems: from biology to hardware*, ICES'03, 228–237. Berlin, Heidelberg: Springer-Verlag.

Vassilev, V. K., and J. F. Miller. 2000. "Scalability Problems of Digital Circuit Evolution: Evolvability and Efficient Designs". In *Proceedings of the 2nd NASA/DoD workshop on Evolvable Hardware*, 55–. Washington, DC, USA: IEEE Computer Society.

Williams, R. 2008, November. "Applying practice to theory". *SIGACT News* 39:37–52.