7 CONCLUSIONS

With FPGAs finding their way into datacenters, HLS tools are set to play a key role in the future of reconfigurable computing. Yet, they are relying on a paradigm which is conceptually identical to the problem of compilation for VLIW processors: generating good static circuits from high-level languages requires peculiar code restructuring algorithms (e.g., modulo scheduling), demands expert user interaction (e.g., pragmas), forces worst-case assumptions on important issues (e.g., memory and control dependencies), and precludes key performance optimizations (e.g., general forms of speculative execution). In this paper, we have described a dynamically scheduled form of HLS and run a simple synthesizer on a few relevant kernels to compare results to a commercial, statically scheduled HLS tool. When static HLS exploits the maximum parallelism available, our technique achieves similar results with minimal degradation in cycle time; when static HLS misses some key performance optimization opportunities, our circuits seize them, achieving large performance improvements with the investment of more resources. Although much remains to be done to refine the optimizations and to add key features we have only evoked so far, we believe our work points to a very promising avenue to make HLS truly valuable on irregular and control-dominated applications.

REFERENCES

[1] M. Alle, A. Morvan, and S. Derrien. Runtime dependency analysis for loop pipelining in high-level synthesis. In *Proceedings of the 50th Design Automation Conference*, pages 51:1–51:10, Austin, Tex., June 2013.

[2] Amazon.com, Inc. *Amazon EC2 F1 Instances.*

[3] M. Budiu, P. V. Artigas, and S. C. Goldstein. Dataflow: A complement to superscalar. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, pages 177–86, Austin, Tex., Mar. 2005.

[4] M. Budiu and S. C. Goldstein. Pegasus: An efficient intermediate representation. Technical Report CMU-CS-02-107, Carnegie Mellon University, May 2002.

[5] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli. Theory of latency-insensitive design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-20(9):1059–76, Sept. 2001.

[6] J. Carmona, J. Júlvez, J. Cortadella, and M. Kishinevsky. A scheduling strategy for synchronous elastic designs. *Journal Fundamenta Informaticae*, 108(1–2):1–21, Jan. 2011.

[7] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger. A cloud-scale acceleration architecture. In *Proceedings of the 49th International Symposium on Microarchitecture*, pages 1–13, Taipei, Taiwan, Oct. 2016.

[8] S. Chatterjee, M. Kishinevsky, and U. Y. Ogras. xMAS: Quick formal modeling of communication fabrics to enable verification. *IEEE Design & Test of Computers*, 29(3):80–88, June 2012.

[9] S. Cheng and J. Wawrzynek. Synthesis of statically analyzable accelerator networks from sequential programs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 126–33, Austin, Tex., Nov. 2016.

[10] D. Chiou. Intel acquires Altera: How will the world of FPGAs be affected? In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, page 148, Monterey, Calif., Feb. 2016.

[11] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of synchronous elastic architectures. In *Proceedings of the 43rd Design Automation Conference*, pages 657–62, San Francisco, Calif., July 2006.

[12] J. Cortadella, M. G. Oms, M. Kishinevsky, and S. S. Sapatnekar. RTL synthesis: From logic synthesis to automatic pipelining. *Proceedings of the IEEE*, 103(11):2061–75, Nov. 2015.

[13] D. E. Culler and Arvind. Resource requirements of dataflow programs. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*, pages 141–150, Honolulu, Hawaii, May 1988.

[14] S. Dai, M. Tan, K. Hao, and Z. Zhang. Flushing-enabled loop pipelining for high-level synthesis. In *Proceedings of the 51st Design Automation Conference*, pages 1–6, San Francisco, Calif., June 2014.

[15] S. Dai, R. Zhao, G. Liu, S. Srinath, U. Gupta, C. Batten, and Z. Zhang. Dynamic hazard resolution for pipelining irregular loops in high-level synthesis. In *Proceedings of the 25th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 189–194, Monterey, Calif., Feb. 2017.

[16] G. De Micheli. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, New York, 1994.

[17] J. C. Dvorak. *How the Itanium Killed the Computer Industry*, Jan. 2009.

[18] D. Edwards and A. Bardsley. Balsa: An asynchronous hardware synthesis language. *The Computer Journal*, 45(1):12–18, Jan. 2002.

[19] S. A. Edwards, R. Townsend, and M. A. Kim. Compositional dataflow circuits. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pages 175–184, Vienna, Austria, Sept. 2017.

[20] M. Fingeroff. *High-Level Synthesis Blue Book.* Xlibris Corporation, first edition, 2010.

[21] M. Galceran-Oms, J. Cortadella, and M. Kishinevsky. Speculation in elastic systems. In *Proceedings of the 46th Design Automation Conference*, pages 292–95, San Francisco, Calif., July 2009.

[22] N. George, H. Lee, D. Novo, T. Rompf, K. Brown, A. Sujeeth, M. Odersky, K. Olukotun, and P. Ienne. Hardware system synthesis from domain-specific languages. In *Proceedings of the 23rd International Conference on Field-Programmable Logic and Applications*, pages 1–8, Munich, Sept. 2014.

[23] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, fifth edition, 2011.

[24] G. Hoover and F. Brewer. Synthesizing synchronous elastic flow networks. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 306–11, Munich, Mar. 2008.

[25] Y. Huang, P. Ienne, O. Temam, Y. Chen, and C. Wu. Elastic CGRAs. In *Proceedings of the 21st ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 171–80, Monterey, Calif., Feb. 2013.

[26] L. Josipović, P. Brisk, and P. Ienne. An out-of-order load-store queue for spatial computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):125:1–125:19, Sept. 2017.

[27] T. Kam, M. Kishinevsky, J. Cortadella, and M. Galceran-Oms. Correct-by-construction microarchitectural pipelining. *Proceedings of the 27th International Conference on Computer-Aided Design*, pages 434–41, Nov. 2008.

[28] M. S. Lam. Software pipelining: An effective scheduling technique for VLIW machines. In *Proceedings of the 1988 ACM Conference on Programming Language Design and Implementation*, pages 318–28, Atlanta, Ga., June 1988.

[29] J. Liu, S. Bayliss, and G. A. Constantinides. Offline synthesis of online dependence testing: Parametric loop pipelining for HLS. In *Proceedings of the 23rd IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 159–62, Vancouver, May 2015.

[30] The LLVM Compiler Infrastructure. *http://www.llvm.org.*

[31] Mentor Graphics. ModelSim, 2016.

[32] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–80, Apr. 1989.

[33] S. F. Nielsen, J. Sparsø, J. B. Jensen, and J. S. R. Nielsen. A behavioral synthesis frontend to the Haste/TiDE design flow. In *Proceedings of the 15th International Symposium on Asynchronous Circuits and Systems*, pages 185–94, Chapel Hill, N.C., May 2009.

[34] E. Nurvitadhi, J. C. Hoe, T. Kam, and S.-L. L. Lu. Automatic pipelining from transactional datapath specifications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(3):441–54, Mar. 2011.

[35] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger. A reconfigurable fabric for accelerating large-scale datacenter services. In *Proceedings of the 41st International Symposium on Computer Architecture*, pages 13–24, Minneapolis, Minn., June 2014.

[36] A. R. Putnam, D. Bennett, E. Dellinger, J. Mason, and P. Sundararajan. CHiMPS: A high-level compilation flow for hybrid CPU-FPGA architectures. In *Proceedings of the 16th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 173–178, Monterey, Calif., Feb. 2017.

[37] B. R. Rau. Iterative modulo scheduling. *International Journal of Parallel Programming*, 24(1):3–64, Feb. 1996.

[38] J. Sparsø. Current trends in high-level synthesis of asynchronous circuits. In *Proceedings of the 16th IEEE International Conference on Electronics, Circuits, and Systems*, pages 347–50, Yasmine Hammamet, Tunisia, Dec. 2009.

[39] M. Tan, G. Liu, R. Zhao, S. Dai, and Z. Zhang. ElasticFlow: A complexity-effective approach for pipelining irregular loop nests. In *Proceedings of the 34th International Conference on Computer-Aided Design*, pages 78–85, Austin, Tex., Nov. 2015.

[40] L. Torczon and K. Cooper. *Engineering a Compiler.* Morgan Kaufmann, second edition, 2011.

[41] R. Townsend, M. A. Kim, and S. A. Edwards. From functional programs to pipelined dataflow circuits. In *Proceedings of the 26th International Conference on Compiler Construction*, pages 76–86, Austin, TX, USA, Feb. 2017.

[42] M. Vijayaraghavan and Arvind. Bounded dataflow networks and latency-insensitive circuits. In *Proceedings of the 9th International Conference on Formal Methods and Models for Codesign*, pages 171–80, Cambridge, MA, July 2009.

[43] Xilinx Inc. *Vivado High-Level Synthesis.*