Extended Abstract for Poster: "A New Polymorphic Computing Architecture Based on Fine-Grained Instruction Mobility"

David Hentrich, Erdal Oruklu, Jafar Saniie
dhentric@gmail.com,oruklu@iit.edu,saniie@iit.edu
Illinois Institute of Technology, Department of Electrical and Computer Engineering
Chicago, Illinois

ABSTRACT

This is a summary of the base concepts behind David Hentrich's May 2018 Ph.D. dissertation in Polymorphic Computing. Polymorphic Computing is the emerging field of changing the computer architecture around the software, rather than vice versa. The main contribution is a new polymorphic computing architecture. The key idea behind the architecture is to create an array of processors where a program's instructions can be individually and arbitrarily assigned/mobilized to any processor, even during runtime. The key enablers of this architecture are a dataflow instruction set that is conducive to instruction migration, a microarchitectual block called an "operation cell" ("op-cell"), a processor built around the instruction set and the "op-cells", and arrays of these processors.

CCS CONCEPTS

• **Computer systems organization → Parallel architectures**; **Reconfigurable computing**; **Data flow architectures**; • **Hardware → Emerging architectures**.

KEYWORDS

polymorphic computer architectures, dataflow processors, instruction sets

ACM Reference Format:
David Hentrich, Erdal Oruklu, Jafar Saniie. 2019. Extended Abstract for Poster:
"A New Polymorphic Computing Architecture Based on Fine-Grained Instruction Mobility". *J. ACM* 37, 4, Article 111 (August 2019), 2 pages. https://doi.org/10.1145/1122445.1122456

1 SUMMARY

The poster provides a high level overview of the core concepts behind David Hentrich's May 2018 Illinois Institute of Technology Ph.D. Polymorphic Computing dissertation [10] [13] [12]. Polymorphic Computing is an emerging field where the computer architecture is re-arranged around the software (rather than the software rearranged around the computing architecture) [9] [11]. Other examples of polymorphic computing architectures are TRIPS [2] [16] [7] [8] [4], Raw [18] [17] [1] [15], MONARCH [5] [6] [19], MOLEN [20], and ρ-VEX [14].

The key advancement of this research is to individually free the instructions of a program enough from internal processor architecture dependencies and internal program dependencies to allow a program's individual instructions to be arbitrarily mapped to any processor in a connected array of processors. Essentially, this work allows a program's instructions to individually and freely roam (i.e. be "draped") around a processing core array, even at runtime! Moving a program's instructions around inside an array of processors without the program itself being aware of the instruction-to-processor mapping is equivalent to changing the computer architecture underneath the program.

This poster conveys:

(1) A high level overview of Polymorphic Computing,
(2) A list of individual contributions of this work,
(3) A pictorial description of how the individual mechanisms together build-up to form a polymorphic computing architecture in which program instructions can be arbitrarily "draped",
(4) A result graph showing timing performance improvements found for a program running in several different simulated polymorphic array configurations, and
(5) A set of overall takeaways.

2 CONTRIBUTIONS OF THIS WORK

The contributions of this work are:

(1) A computer architecture that can be changed ("morphed") instead of the software being changed,
(2) A method for performing fine-grained parallelism,
(3) A dataflow instruction set [3] that contributes to free instruction mobility, and
(4) A processor microarchitectural construct called an "operation cell" ("op-cell") that complements the instruction set and enables the instruction mobility.

Author's address: David Hentrich, Erdal Oruklu, Jafar Saniie, dhentric@gmail.com, oruklu@iit.edu,saniie@iit.edu, Illinois Institute of Technology, Department of Electrical and Computer Engineering, 3301 S. Dearborn Street, Chicago, Illinois, 60616.

0004-5411/2019/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

3 MECHANISMS THAT ENABLE INSTRUCTION MOBILITY AND HARDWARE POLYMORPHISM

There are four tiered steps to building the polymorphic computing architecture of this work. The first step is to create a dataflow instruction set that is conducive to individual instruction migration. The second step is the creation of two microarchitectural blocks:

(1) A block called an "operation cell" ("op-cell") that stores/manages an instruction and its associated data, and
(2) An arithmetic logic unit that can execute the dataflow instruction set.

The third step is to create a processor core around the "operation cell" block and the instruction set. This processor also enables "operation cells" to communication between processors. The fourth and final step is to tile these processors together into a larger array. Any connected set of two or more of these processors is a polymorphic computing array.

4 RESULTS

The results graph shows the execution times of the best instruction placements found for a single program placed in several different simulated 2-dimensional polymorphic computing arrays of this work. These execution times are expressed in clock cycles. The program performs a bubble sort of 1024 32-bit integers and was identical for all array configurations. The rightmost dotted line indicates the time of the program in a single processor. The "ideal time" line shows the theoretical ideal timing performance of the program for each array configuration. The best execution times measured for the program are shown as grey bars. The observed best execution times of the program generally improve as the number of cores is increased.

5 TAKEAWAYS

The final takeaways of this work are the potential advancements to the field of computer architecture:

(1) A method for enabling fine-grained (individual) instruction migration around an array of processors,
(2) The observation that moving program instructions around a processor array and moving the processor array around a program are equivalent, and
(3) The ability to move instructions around a processing array at runtime (i.e. changing the computer architecture at runtime).

ACKNOWLEDGMENTS

Results presented in this poster were obtained using the Chameleon testbed supported by the National Science Foundation.

REFERENCES

[1] Anant Agarwal. 1999. Raw Computation. *Scientific American* 281, 2 (Aug. 1999).
[2] Doug Burger, Stephen W. Keckler, Kathryn S. McKinley, Mike Dahlin, Lizy K. John, Calvin Lin, Charles R. Moore, James Burrill, Robert G. McDonald, William Yoder, and The TRIPS Team at the University of Texas at Austin. 2004. Scaling to the End of Silicon with EDGE Architectures. *IEEE Computer* 37, 7 (July 2004), 44–55.
[3] Jack B. Dennis and David P. Misunas. 1975. A Preliminary Architecture for a Basic Data-Flow Processor. *Proceedings ofthe 2nd annual symposium on Computer Architecture (ISCA '75)* (1975), 126–132.
[4] Mark Gebhart, Bertrand A. Maher, Katherine E. Coons, Jeff Diamond, Paul Gratz, Mario Marino, Nitya Ranganathan, Behnam Robatmili, Aaron Smith, James Burrill, Stephen W. Keckler, Doug Burger, and Kathryn S. McKinley. 2008. *An Evaluation of the TRIPS Computer System (Extended Technical Report).* Technical Report TR-08-31. Computer Architecture and Technology Laboratory, Department of Computer Sciences, The University of Texas at Austin, Austin, Texas. 25 pages.
[5] John Granacki and Mike Vahey. 2003. *MONARCH: A Morphable Networked micro-ARCHitecture.* Technical Report. USC/Information Sciences Institute and Raytheon.
[6] John J. Granacki. 2005. *MONARCH: Next Generation SoC (Supercomputer on a Chip).* Technical Report. USC/Information Sciences Institute.
[7] Paul Gratz, Changkyu Kim, Robert McDonald, Stephen W. Keckler, and Doug Burger. 2006. Implementation and Evaluation of On-Chip Network Architectures. In *IEEE International Conference on Computer Design (ICCD 2006).* San Jose, California, 477–484.
[8] Paul Gratz, Changkyu Kim, Karthikeyan Sankaralingam, Heather Hanson, Premkishore Shivakumar, Stephen W. Keckler, and Doug Burger. 2007. On-Chip Interconnection Networks of the TRIPS Chip. *IEEE Micro* 27, 5 (Oct. 2007), 41–50.
[9] Robert B. Graybill. 2000 (accessed February 4, 2018). *Future Embedded Computing Architectures.* http://archive.darpa.mil/DARPATech2000/Presentations/ito_pdf/3GraybillFutureECAB&W.pdf.
[10] David Hentrich. 2018. *A Polymorhic Computing Architecture Based on a Dataflow Processor.* Ph.D. Dissertation. Illinois Institute of Technology.
[11] David Hentrich, Erdal Oruklu, and Jafar Saniie. 2011. Polymorphic Computing: Definition, Trends, and a New Agent-Based Architecture. *Circuits and Systems* 2, 4 (Oct. 2011), 358–364.
[12] David Hentrich, Erdal Oruklu, and Jafar Saniie. 2018. A Dataflow Processor as the Basis of a Tiled Polymorphic Computing Architecture with Fine-Grain Instruction Migration. *IEEE Transactions on Parallel and Distributed Systems* 29, 10 (Oct 2018), 2164–2175. https://doi.org/10.1109/TPDS.2018.2819670
[13] David Raymond Hentrich. 2017. Operation Cell Data Processor Systems and Methods.
[14] J. Hoozemans, J. van Straten, and S. Wong. 2017. Using a polymorphic VLIW processor to improve schedulability and performance for mixed-criticality systems. In *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).* 1–9. https://doi.org/10.1109/RTCSA.2017.8046315
[15] Walter Lee, Rajeev Barua, Matthew Frank, Devabhaktuni Srikrishna, Jonathan Babb, Vivek Sarkar, and Saman Amarasinghe. 1998. Space-Time Scheduling of Instruction-Level Prallelism on a Raw Machine. In *Proceedings ofthe Eighth International Conference on Architectural Support for Programming Languages and Operating Systems 1998.* San Jose, California, 46–57.
[16] Robert McDonald, Doug Burger, Stephen W. Keckler, Karthikeyan Sankaralingam, and Ramadass Nagarajan. 2005. *TRIPS Processor Reference Manual.* Technical Report TR-05-19. Department of Computer Sciences, The University of Texas at Austin, Austin, Texas.
[17] Michael Taylor. 2005. *The Raw Prototype Design Document V5.02.* Department of Electrical and Computer Science, Massachusetts Institue of Technology, Cambridge, MA.
[18] Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Paul Johnson, Jae-Wook Lee, Walter Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. 2002. The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs. *IEEE Micro* 22, 2 (April 2002), 25–35.
[19] Michael Vahey, John Granacki, Lloyd Lewins, Drew Davidoff, Jeff Draper, Craig Steele, Gillian Groves, Matt Kramer, Jeff LaCoss, Kenneth Prager, Jim Kulp, and Charles Channell. 2006. MONARCH: A First Generation Polymorphic Computing Processor. In *Proceedings of the 10th Annual High Performance Embedded Computing Workshop 2006 (HPEC 2006).* MIT Lincoln Laboratory, Lexington, Massachusetts.
[20] S. Vassiliadis, S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov, and E. M. Panainte. 2004. The MOLEN polymorphic processor. *IEEE Trans. Comput.* 53, 11 (Nov 2004), 1363–1375. https://doi.org/10.1109/TC.2004.104