The Orchestration Stack: The Impossible Task of Designing Software for Unknown Future Post-CMOS Hardware

Marcus Völp[†], Sascha Klüppelholz, Jeronimo Castrillon, Hermann Härtig, Nils Asmussen, Uwe Aßmann, Franz Baader, Christel Baier, Gerhard Fettweis, Jochen Fröhlich, Andrés Goens, Sebastian Haas, Dirk Habich, Mattis Hasler, Immo Huismann, Tomas Karnagel, Sven Karol, Wolfgang Lehner, Linda Leuschner, Matthias Lieber, Siqi Ling, Steffen Märcker, Johannes Mey, Wolfgang Nagel, Benedikt Nöthen, Rafael Peñaloza[‡], Michael Raitza, Jörg Stiller, Annett Ungethüm, and Axel Voigt
Center for Advancing Electronics Dresden (cfaed) — Technische Universität Dresden, Germany
[†] now at SnT-CritiX — University of Luxembourg, [‡] now at KRDB Research Centre — Free University of Bozen-Bolzano

Abstract—Future systems based on post-CMOS technologies will be *wildly* heterogeneous, with properties largely unknown today. This paper presents our design of a new hardware/software stack to address the challenge of preparing software development for such systems. It combines well-understood technologies from different areas, e.g., network-on-chips, capability operating systems, flexible programming models and model checking. We describe our approach and provide details on key technologies.

1 Introduction

In 2012, the large-scale research project *Center for Advancing Electronics Dresden (cfaed)* was set up in Dresden to explore new materials and technologies for electronic information processing, which potentially help overcoming the limits of today's CMOS-technology. The project consists of multiple sub-projects that focus on promising concrete technologies, including reconfigurable transistors based on silicon nanowires (SiNW) [43], [17], [12] and carbon nanotubes (CNT) [38], [36], [32], [29], organic electronics [31], chemical information processing (e.g., microchemomechanical labs-on-chip [42]) and self-assembling nano-structures built with DNA origami [14]).

The long-term - allegedly impossible - task of cfaed's *Orchestration* sub-project is to unleash the full potential of future - yet unknown - computing platforms and to turn breakthroughs in emerging materials and technologies into application performance. We envision *wildly* heterogeneous computing systems with potentially large numbers of possibly unreliable processing elements and deep heterogeneous memory subsystems that are in part built from the above technologies.

Since these new technologies are not yet available and their characteristics unknown, we use heterogeneous CMOS systems as a starting point for our research. Our objective is to initially design CMOS-based systems such that they can be more easily used for novel technologies and architectures. Since heterogeneity already is an important concept for overcoming barriers limiting conventional CMOS-based architectures (e.g., power-density problems [39], [16]), we can start with an already large base of heterogeneity [9], [28], [10], [26], [33], [35], [37], [5], [34]. Sec. 2 describes the *Orchestration Stack* which addresses expected challenges for wildly heterogeneous systems and Sec. 3 presents initial implementations contributing to the stack.

2 The Orchestration Stack

On the lowest layer of the Orchestration Stack (see Fig. 1) we assume to have a variety of heterogeneous components based on different technologies such as SiNWs, CNTs or classical CMOS hardware (possibly with upcoming channel materials [21]) that have different characteristics in terms of performance and costs. These components can be specialized processing elements, e.g., accelerators, heterogeneous memories [44] or interfaces to bridge to peripherals, e.g., wireless communication devices, or to novel computing fabrics, e.g., labs-on-a-chip. Components can also be partially reconfigurable circuits combining different processing elements or providing a platform for application-specific and even software-delivered circuits.

All we require is that the components have a well-defined interface that allows embedding them into a tile-based architecture and that enables to exchange data and commands using some kind of network (e.g., a network-on-chip (NoC) [18]). New materials providing this interface can hence be embedded into this architecture. In Sec. 3 we present such an architecture and illustrate first steps we took towards our technical vision of fast reconfigurable hardware built from SiNWs or CNTs.

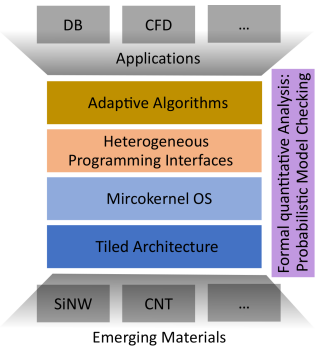

Fig. 1. The Orchestration Stack

The *operating system's* (OS) task is to isolate hardware components and establish communication channels to other tiles and remote memories. As is common practice, the OS consist of a kernel and several *servers* on top of the kernel [15], [13], [30]. However, since no assumptions can be made on the components, the classical user/kernel-mode separation of privilege cannot be expected to be available. Hence, a fairly different design is needed for the OS. Sec. 3 presents a first prototype.

The main challenge in programming wildly heterogeneous, parallel systems is to master and hide the complexity from upper layers of the stack while preserving the opportunities provided by the underlying hardware. The keys for addressing this challenge are *heterogeneous programming interfaces* in terms of domain specific languages (DSLs), programming models and runtime systems. These interfaces help separating the concerns

The research presented in this article is supported by the German research council (DFG) and the state Saxony through the cluster of excellence "Center for Advancing Electronics Dresden" (cfaed), cfaed.tu-dresden.de.

of core algorithmic problems and possible implementations from structural properties of the architecture and particular properties of the heterogeneous components. Along interfaces, compilers are needed to lower the abstractions and, for example, reason about parallel execution and data layouts. Compilers must identify application resource demands specific to the hardware while exploiting heterogeneous resources. This includes methods for deciding where, when and how to run which parts of the application and where to store data, which, in turn, requires models of the specific heterogeneous resources. Compilers should thus generate alternatives equipped with meta-information for possible mappings of algorithms to resources. Based on the generated meta-information, application runtimes and OS-level resource managers negotiate desired and available resources to find a global schedule of resources that meets all application requirements. Once granted by the OS, the runtime adjusts the application by switching to the respective compiler generated alternative.

To benefit best from the flexibility of heterogeneous computing platforms with dynamic resource (re)allocation mechanisms, application algorithms need to provide a high degree of flexibility. Within Orchestration we work on data base applications (DB), computational fluid dynamics (CFD) and computational biologyy and use them as drivers for our approach.

Last but not least, we integrate formal methods in our design process to quantitatively analyze low-level orchestration protocols for stochastically modeled classes of applications and systems [4]. The model-based *formal quantitative analysis*, carried out using probabilistic model checking (cf. [11], [8]), is particularly useful for the comparative evaluation of (existing and future) design alternatives, to compare the performance of heuristic orchestration policies with theoretical optimal solutions or to determine optimal system parameter settings.

3 Towards an Implementation of the Stack

This section provides some details of and pointers to components well fitting to the spirit of the Orchestration Stack. These examples include some of our own work and some others we found in the scientific literature.

SiNW Reconfigurable Circuits: As first steps to replace common CMOS circuits by new technology, we chose to start rebuilding CMOS-based reconfigurable circuits using *Silicon NanoWire* technology. SiNW technology promises simpler design and manufacturing processes, as it is doping-free, has homogeneous physical and electrical characteristics [17] and is inherently CMOS compatible. In SiNW transistors, polarity is individually controllable via a separate polarity gate. Thus, p-type and n-type transistors can be mixed on the die, which eases wiring constraints and allows for tighter placement.

Making use of this property allowed us to reduce the transistor count for a 6-function programmable logic cell over two inputs from 92 transistors for CMOS-based circuits to 26 [40]. We also improved basic gates like NAND, NOR, X(N)OR, majority/minority and MUX. Starting from this, we observe the effects of these improvements on a larger circuit, an 8-bit conditional carry adder. Using SiNW transistors, the speed can be improved by 25%, the area by 14% and the transistor count by almost 50%.

The Tomahawk Architecture: Tomahawk [2] is a CMOS-based multiprocessor system-on-a-chip (MPSoC) with processing elements and accelerators for digital signal processing and database querying [1], [41], [23]. Processing elements are equipped with local scratch-pad memories and connected via a NoC. The Tomahawk architecture allows for connecting arbitrary, untrusted hardware components, e.g., freely programmable FPGAs. To unify the control over these tiles, each tile is connected to the NoC via a Data Transfer Unit (DTU). The DTU provides controlled message passing and memory access to other networked components. It has two interfaces, one for the untrusted component to access outside memory and to send/receive messages, the other for higher privileged components to control the permissions of these accesses. The only requirement for the untrusted component to access memory and to send messages is the ability to access the DTU registers. Notably, it does not require complex architectural properties such as virtual memory for protection. In addition to the DTU-mechanisms, the Tomahawk provides a logically decoupled processing element called CoreManager, coordinating the processing elements and responsible for the allocation and configuration of processing elements and global memory and tile-to-tile data transfers (similar to [22]).

The M³ Operating System: Similar to other microkernel-based approaches, M³ systems [3] are split into privileged kernels and unprivileged servers and applications. However, unlike in traditional OS approaches, the kernel cannot rely on processor features like user/kernel mode and memory management units to shield itself from applications.

Instead, in M³ systems, one or more M³ kernels run on dedicated and privileged tiles, while servers and applications run on unprivileged tiles. The key for isolation is that only privileged tiles can configure DTUs to, e.g., create communication channels. With this design, arbitrary components can be integrated as tiles and controlled by the M³ kernel. The configuration of DTUs is controlled by means of capabilities modeled after the L4 capability [27] system. Capabilities are created and protected by the kernel and can be exchanged between servers and applications. OS functionality like file systems or network stacks are provided by servers on unprivileged tiles and can be accessed from applications via DTU messages. For example, a file system can be built as an untrusted FPGA.

Heterogeneous Programming Interfaces: As mentioned in Sec. 2, we achieve separation of concerns with *interfaces* at different levels. As an example at a lower level, we have built a dataflow-based language together with a retargetable compiler that generates optimized code for the Tomahawk architecture [7]. This includes a mapping of actors to heterogeneous processing elements, and of data transfers to the underlying message passing interface over the DTU. At higher levels, we have made significant progress in new algorithms for CFDs [19], [20], a DSL for computational biology [25] and a general skeleton framework [24]. We also see great potential on interfaces that allow describing memory access patterns, as successfully shown by Ben Nun et al [6] using template metaprogramming for different GPU architectures.

4 Conclusion

It is much too early to draw conclusions on whether or not the Orchestration Stack will successfully enable the effective and efficient usage of novel post-CMOS technologies. However, our initial experiences in building parts of the stack - mostly restricted to CMOS - did not expose any obvious show stoppers.

References

[1] O. Arnold, S. Haas *et al.*, "An application-specific instruction set for accelerating set-oriented database primitives," *SIGMOD*, 2014.

[2] O. Arnold, E. Matus *et al.*, "Tomahawk: Parallelism and heterogeneity in communications signal processing MPSoCs," *ACM Transactions on Embedded Computer Systems*, vol. 13, no. 3s, pp. 107:1–107:24, Mar. 2014.

[3] N. Asmussen, M. Völp *et al.*, "M3: A hardware/operating-system co-design to tame heterogeneous manycores," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16. New York, NY, USA: ACM, 2016, pp. 189–203.

[4] C. Baier, C. Dubslaff *et al.*, "Probabilistic model checking for energy-utility analysis," in *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, ser. Lecture Notes in Computer Science, vol. 8464. Springer, 2014, pp. 96–123.

[5] A. Barbalace, M. Sadini *et al.*, "Popcorn: Bridging the programmability gap in heterogeneous-ISA platforms," in *Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15)*. New York, NY, USA: ACM, 2015, pp. 29:1–29:16.

[6] T. Ben-Nun, E. Levy *et al.*, "Memory access patterns: The missing piece of the multi-gpu puzzle," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: ACM, 2015, pp. 19:1–19:12.

[7] J. Castrillon and R. Leupers, *Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap*. Springer, 2014.

[8] P. Chrszon, C. Dubslaff *et al.*, "Family-based modeling and analysis for probabilistic systems - featuring ProFeat," in *Proc. of the 19th International Conference on Fundamental Approaches to Software Engineering (FASE)*, ser. Lecture Notes in Computer Science, vol. 9633. Springer, 2016, pp. 287–304.

[9] F. Conti, C. Pilkington *et al.*, "He-p2012: architectural heterogeneity exploration on a scalable many-core platform," in *Proceedings of the 24th edition of the great lakes symposium on VLSI*. ACM, 2014, pp. 231–232.

[10] K. V. Craeynest and L. Eeckhout, "Understanding fundamental design choices in single-isa heterogeneous multicore architectures," *ACM Transactions on Architecture and Code Optimization*, vol. 9, no. 4, pp. 32:1–32:23, January 2013.

[11] C. Dubslaff, C. Baier, and S. Klüppelholz, "Probabilistic model checking for feature-oriented systems," *Transactions on Aspect-Oriented Software Development*, vol. 12, pp. 180–220, 2015.

[12] P.-E. Gaillardon, H. Ghasemzadeh, and G. De Micheli, "Vertically-stacked silicon nanowire transistors with controllable polarity: A robustness study," in *Test Workshop (LATW), 2013 14th Latin American*, April 2013, pp. 1–6.

[13] D. B. Golub, D. P. Julin *et al.*, "Microkernel operating system architecture and mach," in *In Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures*, 1992, pp. 11–30.

[14] F. N. Gür, F. W. Schwarz *et al.*, "Toward self-assembled plasmonic devices: High-yield arrangement of gold nanoparticles on dna origami templates," *ACS Nano*, vol. 10, no. 5, pp. 5374–5382, 2016, pMID: 27159647.

[15] P. B. Hansen, "The nucleus of a multiprogramming system," *Communications of the ACM*, vol. 13, no. 4, pp. 238–241, 1970.

[16] N. Hardavellas, M. Ferdman *et al.*, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. EPFL-ARTICLE-168285, pp. 6–15, 2011.

[17] A. Heinzig, T. Mikolajick *et al.*, "Dually active silicon nanowire transistors and circuits with equal electron and hole transport," *Nano Letters*, vol. 13, no. 9, pp. 4176–4181, 2013, pMID: 23919720.

[18] A. Hemani, A. Jantsch *et al.*, "Network on chip: An architecture for billion transistor era," in *Proceeding of the IEEE NorChip Conference*, vol. 31, 2000.

[19] I. Huismann, J. Stiller, and J. Fröhlich, "Two-level parallelization of a fluid mechanics algorithm exploiting hardware heterogeneity," *Computers & Fluids*, vol. 117, pp. 114 – 124, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S004579301500167X

[20] ——, "Factorizing the factorization - a spectral-element solver for elliptic equations with linear operation count," *Arxiv*, vol. abs/1601.08179, 2016, submitted to *Journal of Computational Physics*. [Online]. Available: http://arxiv.org/abs/1601.08179

[21] ITRS technology working groups, "International technology roadmap for semiconductors," ITRS, Tech. Rep., 2013. [Online]. Available: http://www.itrs2.net/2013-itrs.html

[22] M. Jeffrey, S. Subramanian *et al.*, "Swarm: A scalable architecture for ordered parallelism," *Top Picks IEEE Computer Society*, pp. 105–117, May 2016.

[23] T. Karnagel, D. Habich, and W. Lehner, "Local vs. global optimization: Operator placement strategies in heterogeneous environments," in *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Brussels, Belgium, March 27th, 2015.*, ser. CEUR Workshop Proceedings, P. M. Fischer, G. Alonso *et al.*, Eds., vol. 1330. CEUR-WS.org, 2015, pp. 48–55. [Online]. Available: http://ceur-ws.org/Vol-1330/paper-10.pdf

[24] S. Karol, "Well-formed and scalable invasive software composition," Ph.D. dissertation, Technische Universität Dresden, 2015.

[25] S. Karol, P. Incardona *et al.*, "Towards a next-generation parallel particle-mesh language," in *Proceedings of the 3rd Workshop on Domain-Specific Language Design and Implementation*, ser. DSLDI'15, Prague, Czech Republic, Jul. 2015, pp. 15–18.

[26] N. Kavaldjiev and G. Smit, "An energy-efficient network-on-chip for a heterogeneous tiled reconfigurable systems-on-chip," in *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, Aug 2004, pp. 492–498.

[27] G. Klein, K. Elphinstone *et al.*, "seL4: Formal verification of an OS kernel," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2009, pp. 207–220.

[28] R. Kumar, K. I. Farkas *et al.*, "Single-isa heterogeneous multi-core architectures: The potential for processor power reduction," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003, pp. 81–.

[29] S. Li, Z. Yu *et al.*, "Carbon nanotube transistor operation at 2.6 GHz," *Nano Letters*, vol. 4, no. 4, pp. 753–756, 2004.

[30] J. Liedtke, "On μ-kernel construction," in *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP)*, Copper Mountain Resort, CO, Dec. 1995.

[31] B. Lüssem, M. L. Tietze *et al.*, "Doped organic transistors operating in the inversion and depletion regime," *Nat Commun*, vol. 4, 11 2013.

[32] I. Meric, N. Baklitskaya *et al.*, "RF performance of top-gated, zero-bandgap graphene field-effect transistors," *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pp. 1–4, 2008.

[33] T. D. Ngo, "Runtime mapping of dynamic dataflow applications on heterogeneous multiprocessor platforms," Theses, Université de Bretagne-Sud, Jun. 2015. [Online]. Available: https://hal.archives-ouvertes.fr/tel-01167316

[34] E. B. Nightingale, O. Hodson *et al.*, "Helios: Heterogeneous multiprocessing with satellite kernels," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP '09)*. New York, NY, USA: ACM, 2009, pp. 221–234.

[35] W. Quan and A. D. Pimentel, "A hybrid task mapping algorithm for heterogeneous MPSoCs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 1, p. 14, 2015.

[36] S. Rodriguez, A. Rusu, and J. M. de la Rosa, "Overview of carbon-based circuits and systems," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2912–2915.

[37] W. Sheng, S. Schürmans *et al.*, "A compiler infrastructure for embedded heterogeneous MPSoCs," *Parallel Computing*, vol. 40, no. 2, pp. 51–68, 2014.

[38] S. J. Tans, A. R. M. Verschueren, and C. Dekker, "Room-temperature transistor based on a single carbon nanotube," *Nature*, vol. 393, no. 6680, pp. 49–52, May 1998.

[39] M. Taylor, "A landscape of the new dark silicon design regime," *Micro, IEEE*, vol. 33, no. 5, pp. 8–19, Sept 2013.

[40] J. Trommer, A. Heinzig *et al.*, "Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 169–174.

[41] A. Ungethüm, D. Habich *et al.*, "Query processing on low-energy many-core processors," *HARDBD@ ICDE*, 2015.

[42] A. Voigt, R. Greiner *et al.*, "Towards computation with microchemomechanical systems," *International Journal of Foundations of Computer Science*, vol. 25, no. 04, pp. 507–523, 2014.

[43] W. Weber, A. Heinzig *et al.*, "Reconfigurable nanowire electronics – a review," *Solid-State Electronics*, vol. 102, pp. 12 – 24, 2014, selected papers from ESSDERC 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038110114001439

[44] Y. Xie, "Future memory and interconnect technologies," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 964–969.