FPGA AND ASIC CONVERGENCE

C. Valderrama, L. Jojczyk, P. DaCunha Possa [*]

Electronics Department / Polytechnic Faculty
University of Mons, Mons, Belgium
email: carlos.valderrama@umons.ac.be

J. Dondo Gazzano [†]

School of Computer Science
Universidad de Castilla-La Mancha, Spain
email: juliodaniel.dondo@uclm.es

ABSTRACT

The growing demands on multimedia applications and high-speed high-quality telecommunication systems with real-time constrains oriented to portable, low power consumption, devices, have being driven technologies development, methodologies and design flows of embedded systems during the last years. Through the analysis of design methodologies and strategies facing multi-core, reconfigurability and power consumption challenges, this educational survey will follow that evolution approaching ASIC and FPGA architectures on the embedded systems arena.

1. INTRODUCTION

Embedded systems comprise complete devices ranging from portable such as video cameras and set-top boxes, to industrial controllers. They are designed to perform dedicated specific tasks with real-time processing constraints. The large diffusion of embedded systems raises very demanding requirements in terms of computing performance, power budgeting and functional flexibility. By looking closely at those requirements, we will see the convergence of reconfigurable and embedded trends.

Multimedia processing and high speed telecommunication impose real-time and quality constraints requiring intensive computation capabilities. However, the continuous evolution of standards necessitates a flexible architecture adaptable to new computation patterns after fabrication. Market pressure encourages the use of pre-fabricated platforms [1]. Mass production portable devices should be cheap and power efficient. A combined software/silicon technology provide a way to create and modify highly customized embedded systems applications and develop cost-effective derivatives to address changing market demands at the price of a standard product. In this context, the reuse of a common platform reduces time-to-market and manufacturing costs by increasing productivity. However, those platforms must be highly configurable to be useful for a variety of applications, and hence mass produced.

Multi-core architectures have become the foundation of most common appliances such as digital TV, navigation systems and wireless devices. Indeed, the performance requirements of microprocessors have increased during the last decade at the cost of a great rise of clock frequencies, complexity and power consumption. In addition to performance, splitting computation over multiple processing units also tackles the power dissipation problem. Indeed, processor complexity and clock speed stop climbing while processor cores have multiplied. Homogeneous multi-core DSP architectures appear as a viable solution for high performance applications in packet telephony, 3G wireless and WiMax [2]. Scalar processors supported by highly efficient communication architectures emerge today providing a unified programming environment for parallel high-performance applications [3][4]. Indeed, the Graphics Processing Unit (GPU), especially suited to data parallel computation (with the same program executed on many data elements concurrently) has evolved into an increasingly convincing computational resource for non graphics applications.

The communication volume between cores will increase drastically. Future SoC integrates multiple cores, processing units and storage elements, with performances highly dependent on interconnection facilities. Further, the impact of communication energy consumption becomes critical as communication scales up in multi-processor architectures [5]. That motivated a design paradigm shift towards on-chip communication. Transaction Level Modeling (TLM) appeared to improve modeling and design efficiency of a communication scheme [6]. However, state-of-the-art on-chip bus topologies and protocols suffer from power, performance and scalability limitations [7][8].

Requirements regarding communication scalability and reconfiguration are better satisfied with the Network on Chip (NoC) solution [9]. NoC is an area/speed tradeoff between bus and point-to-point communication. The NoC architecture, based on a set of routers driving the data exchange between a set of Intellectual Property (IP) components, allows having parallel data paths as well as reconfigurable connections. Moreover, the regularity of NoC topologies satisfies the scalability property. As an example, the Intel's power efficient Teraflops Research Chip includes a 2D mesh NoC and fine-grain power management (combining frequency scaling and multiple cores activation) [10].

In recent years, hardware reconfigurable architectures received increasing attention due to their flexibility and short design time. Their distinctive main feature is the capability of changing the system structure in the field, a process known as reconfiguration. Reconfiguration enables fast prototyping, incremental architecture refinement and post-implementation error correction [11]. Field-programmable Gate Arrays (FPGAs) emerged in the '80's as reconfigurable logic devices. Twenty years ago, in addition to their reconfigurable main feature, FPGA vendors started combining general purpose processors and

FPGA fabric on a single chip. The Excalibur solution from Altera [12] already included the Nios soft embedded processor and general purpose programmable logic into a single device. Xilinx embed IBM PowerPC processor cores in Virtex FPGAs in addition to the Pico/Micro Blaze soft cores on Spartan-3 FPGA [13]. Current FPGA technology allows the integration of several Generic Purpose Processors (GPPs), on-chip memory, custom accelerators, peripherals, and other complex hard blocks [12] [13] [14].

Tomorrow's multi-core SoC designs will be pervasive with high-performance engines and standard accelerators outpacing front-end computations needed for communication and multimedia applications [15]. Consequently, we must understand how today's components can be built on and integrated into other applications. More video and audio compression standards with higher compression ratios will increase content availability in all consumer devices. Reliable wireless transport of HD (high-definition) video and standard wireless sensors interfaces will arise for high performance processors.

We will summarize the main aspects motivating the coexistence of design techniques in embedded systems design. The first section will explore the evolution of design strategies and methodologies, challenges and requirements. Afterward, we will explore and compare multiprocessors platforms and reconfigurable architectures. Current FPGA architectures will be described regarding its flexibility, processing power, size and reconfiguration capabilities. Section 5 will consider components re-usability as one of key elements to reduce cost and design time. In section 6, Languages and Tools, we will highlight the investment on interoperability of tools taking care of programmable and reconfigurable alternatives. Finally, in section 7 we will provide some conclusions.

2. DESIGN APPROACHES

Modern co-design approaches are based on system level specification and refinement [16][17]. They suggest a design space exploration step as a preliminary estimate of performance requirements before architecture selection [18][19]. Indeed, using a high abstraction level minimizes the modeling effort increasing the simulation speed. The refinement process (partitioning, mapping and resources utilization) is driven by the target architecture characteristics and specification requirements. Partitioning can increase software performance by implementing a critical section in an FPGA fabric. The Y-chart methodology recognizes a clear separation between application (behavior) and architecture (dedicated, programmable and reconfigurable resources): an explicit mapping step correlates application (computational and communication events) and architecture (timing characteristics and performance consequences of an application event) models [20][21]. The mapping of each application onto an architecture instance is evaluated by performance analysis.

Platform-based design (PBD) is an embedded software design style concerned about IP reuse [15][22][23][24]. PBD automates the non-differentiated aspects of a potential implementation, concentrating the effort on value added design. Design effort was reduced by customizing a common and generic platform with a variety of sophisticated IPs sometimes coming from different sources. The interplay of top-down propagation of constraints and bottom-up performance estimation promotes a fast migration of a platform instance (a legal architectural composition) to one specific application. However, as detailed in [25], PBD is an intermediate solution for SoC designers. Any architecture change disrupts the interfaces implying a major effort in verification and set-up required to keep the platform viable for current standards. Furthermore, for legacy and performance reasons, reconfigurable hardware cannot fully support leading edge' applications [22][26]. Nevertheless, an application-specific instruction-set processor (ASIP) can potentially offer the right combination of required flexibility and performance. Indeed, the configurable processor technology can tailor instruction set, processor interface and/or data-path of a base architecture to suit the given application [23]. For instance, the Tensilica Xtensa high-performance data-plane processor optimized for digital signal processing offers a wide range of pre-verified DSP options ranging from a simple floating point accelerator to a 16-MAC vector DSP power-house [22][23]. Embedded reconfigurable platforms combine the flexibility of software (standard programmable cores, DSP and ASIP) with the performance of reconfigurable logic; better adapted to different applications and legacy constraints [27].

Domain and service oriented platforms are becoming widespread. Most of today DSPs are used in the wireless sector, especially in mobile handsets, and new applications will increase the demand of processing power and lower power consumption. That represents a shift from platform-oriented to domain-oriented embedded systems [28]. Indeed, recent commercial configurable platforms show a steady migration from application specific circuits to domain oriented platforms. The actual offer ranges from embedded FPGAs to reconfigurable processors and configurable SoC devices [29][30][31]. Flash-based FPGAs have reasonably performance and power consumption balance for embedded platforms [32]. They offer a viable option for higher-volume cost-sensitive applications. Their time-to-market and system flexibility advantages can be truly compelling when compared to fixed-hardware solutions. In particular, Flash memory cells for non-volatile memory chips are now used because of their improved area due to the steady decrease in manufacturing process geometries. With the exception of the overhead in programming circuitry and number of programming cycles, this feature avoids the use of external devices to store configuration data, allowing immediate execution upon power-up. As an example, Actel ProASIC, Fusion, and Igloo Flash-based devices all can include a soft-core platform into their FPGA fabric. The CoreMP7 from Actel includes a 32-bit ARM core, real-time operating system (RTOS) and a variety of peripherals that can be connected via the AMBA bus providing wide-ranging IP interoperability.

TLM appeared managing the communication mechanism and interface requirements at a high level of abstraction. The interconnection infrastructure becomes harder to design due to the endless increase of the number of components and the performance requirements to deal with. A TLM function call models the communication hiding the intended hardware/software implementation (interconnection topologies, communication mechanisms and synchronization primitives) [33]. Beyond its simulation purpose, thus, facilitating architecture design and exploration, TLMs assist the modeling and refinement of communication topologies and protocols [34][35][36][37]. However, it is still required to capture de reactive nature of embedded systems [38]. The simulation of a reactive transaction (the transaction can be killed or reinitialized before its completion) can formally prove the correctness of the implementation choice (the design does not deadlock and is always responsive).

3. MULTI-CORE ARCHITECTURES

Multi-core architectures are today available within a single chip with varying capabilities [39][40]. They provide software flexibility and low power consumption to a vast volume of consumer electronic products. Indeed, their complex architecture benefits from a combination of advanced power management techniques like, for instance, dynamic voltage and frequency scaling, clock gating, and synchronization mechanisms such as GALS (Global Asynchronous – Locally Synchronous). Heterogeneous multi-core architectures include in the same die GPPs, DSPs, memory and dedicated IPs. Commercial application-oriented like, for instance, the Beagle board for embedded applications (a low power and low cost single board computer based on the OMAP3530 device produced by Texas Instrument) includes an ARM Cortex-A8 CPU, a TMS320C64x+ DSP, and an Imagination Technologies PowerVR SGX530 GPU [40]. The Cell Broadband Engine includes a Power-Architecture processor and eight attached streaming processors with their own memory [3]. Each processor has several SIMD (Single-Instruction Multiple-Data) units that can process from 2 double-precision floating-point values up to 16 byte-values per instruction. The platform exploits parallelism at all levels, including data and task level parallelism, as well as SIMD parallelism optimization. SIMDs are homogeneous multi-core architectures dedicated to data-parallel computations on multimedia applications. For instance, The NVidia Tegra family of GPUs, a regular architecture based on multiple SIMD units and multi-level cache and shared memory, brings the power of advanced visual computing to a broad range of handheld and mobile platforms [4]. GPU cores like NVidia, Imagination Technologies PowerVR, Yappa, ZiiLabs, Vivante, Takumi, or ATI, among others, are today present in several embedded SoCs. Embedded devices, such as the OMAP3530, offer the GPU as a co-processor [41]. The Massively Parallel Processor Array (MPPA) is a regular multi-core architecture specifically developed for embedded systems. The MPPA provides scalability to processing power and real-time data flow for streaming media applications with hard real-time requirements [42]. The MPPA architecture is a MIMD (Multiple Instruction Multiple Data) parallel array of CPUs and memories interconnected by a 2D-mesh configurable network. Memory is distributed and individual CPU/RAM blocks are located in tiles stacked up to form the array. Like an FPGA, tiles are connected by configurable routers, but using coarse-grained and hierarchical interconnections (clusters of CPU/RAM blocks). Another example of the next generation of multi-core SoC illustrating a shift towards a programmable on-chip communication platform is the Spidergon IPU (Interconnect Processing Unit) from STMicroelectronics [43]. The Spidergon NoC acts as a distributed programmable set of services to design advanced application features. A similar approach was adopted by the Intel's power efficient Teraflops Research Chip, providing more than one Teraflops of performance [10]. The multi-core Teraflops includes a 2D mesh NoC and fine-grain power management (combining frequency scaling and multiple cores activation).

4. RECONFIGURABLE ARCHITECTURES

Reconfigurable platforms based on FPGAs have quickly become the main option to address the flexibility of software with the performance of hardware. FPGA-based systems are faster than a pure software approach in terms of computational power, less static than traditional ASIC-based solutions and with better time-to-market [44]. However, the introduction of reconfigurable platforms into embedded systems attaches more challenges to embedded systems designers. Moreover, the high cost per unit is also a barrier for an expansion of utilization of FPGAs in embedded systems. The cost per unit of FPGA devices is still higher than ASIC's cost for high-volume applications. Though, that difference has been attenuated by the current expansion of the FPGA market [45]. New improvements of FPGA architectures claim that the new generation of FPGAs can reduce significantly the power consumption and area restrictions [46]. Flash-based FPGAs have reasonably performance and lower power consumption compared to SRAM [32][47][48]. New architectures operating on coarser multi-bit data, including bus-based routing, logic blocs and units (raw ALUs and DSPs) has also been proposed to improve performance and efficiency [49]. Another important feature of new FPGA architectures is the possibility to integrate one or more GPPs [50][14][12]. Today, hard processors, with better performance than a soft-core processor, are part of FPGA families. An example is the Virtex-5 FXT family with up to two IBM PowerPC 440 hard processors [14]. Other hard GPPs, like ARM (including the AMBA bus) and MIPS are coming with

Xilinx, Actel and Altera with a tremendous commercial and technology convergence.

We can see that modern FPGAs include enough resources to build a complete SoC. However, most real-life applications are simply too large to fit in the logic available on a single chip [44]. The area/power overhead of FPGAs can be reduced by using alternatives such as partially prefabricated structured ASIC, a partially pre-fabricated die including similar FPGA components (array of gates, memory, hard processors, DSP blocks, pre-routed interconnections plus power distribution) where dedicated switches and configuration elements are replaced by metal layers customization [12][51]. In addition to the ASIC alternative, software-like reconfigurable techniques such as partial dynamic reconfiguration (PDR), thread warping and multi-booting can be also used [52]. PDR allows the partial modification of the FPGA without stopping the execution of the remainder at the cost of reconfigurable regions and latency overhead [11][53]. The growing number of software applications written using multiple threads can benefit from multi-core parallelism [46][54]. Another software-flavor technique similar to multi-booting, thread warping, dynamically remaps critical code regions from processor instructions to FPGA circuits using runtime synthesis [55]. The technique improves performances speeding up individual threads and allowing more threads to execute concurrently [56]. Binary-level partitioning provides competitive results in terms of performance and energy.

4.1. Dynamically Reconfigurable Architectures

Dynamically reconfigurable architectures are those architectures based on FPGAs that allows run-time partial reconfiguration. Dynamic reconfiguration has been an area of intensive research during the last two decades. Nowadays there is a good understanding on the reconfiguration mechanisms, architecture alternatives, reconfiguration implications [57][58]. Reconfiguration overhead is still a bottleneck for most applications, especially when we consider the clock speed gap between standard microprocessors and FPGAs. Dealing with DRA requires taking into account several issues: reconfiguration management, persistence management, IP integration for reconfigurable areas, reconfiguration process activation, and task migrations. Managing dynamic reconfigurable hardware implies having to deal with a new degree of freedom for systems designer: to the traditional partition hw/sw problem, the static and dynamically reconfigurable hardware partition problem is added. The use of this kind of architectures allows the incorporation of new features into the already deployed design, allowing also components updating in run-time [59].

5. IP INTEGRATION

Components re-usability is one of key elements to reduce the cost and time of design. This re-usability requires the development of integration and standardization techniques, methodologies and tools [60][61]. Nowadays big efforts have been made for standardization and integration of components. The Spirit Consortium [62], (Structure for Packaging, Integrating and Re-using IP within Tools-flows) today merged with EDA Standards Organizations Accellera [63], has been created to facilitate and improve IPs integration. This consortium is formed by ARM, Cadence Design Systems Inc., Mentor Graphics, Royal Philips Electronics, STMicroelectronics and Synopsys, Inc. One example of their work is the IEEE 1685™: Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows [63].

Standards for interconnection interfaces facilitate integration tasks, e. g. VCI [64] and OCP [65] that follows a plug and play approach. To deal with system-level component interaction several techniques were developed such as COSY [66] and ROSES [60]. Both techniques make use of library of adapters for interconnection schemes. Multiflex [67] is a system that enables integration of heterogeneous components (Hw/Sw) into a uniform software platform. Others approaches for components integration and SoC design based on distributed objects models can be found in [68][69].

6. LANGUAGES AND TOOLS

Currently, there exists a plethora of tools available for embedded system designers, but, it is hard to choose which will fit the design needs in terms of usability, time-to-market, and granularity. Traditional design techniques targeting FPGA devices are based on hardware description languages (HDLs). However, it is hard to deal with a complex system, composed by several processors, memory, and IP blocks with just those HDLs. Many companies, e.g. Xilinx, Altera, Mentor Graphics, provide a large set of IP cores that can be used as entry of Integrated Design Environments. IP repositories are available for embedded system designers. IP entry methods based on Matlab/Simulink are supported by tools such as System Generator (Xilinx), DSP Builder (Altera), and Synplify DSP (Synopsys).

High-level languages can accelerate the hardware/software development process. Programming languages, e.g. C/C++, help embedded system designers to handle the complexity at the system level. In addition, C-based languages such as SystemC (supported by AutoPilot from AutoESL, C-to-Silicon Compiler from Cadence or Catapult C from Mentor Graphics), Handel-C (DK Design Suite from Agility), and Impulse C (supported by CoDeveloper from Impulse Accelerated Technologies) enable non-hardware specialists to program FPGAs. As a consequence of the standardization process, widely used standard system-level modeling languages such as SystemC and APIs appeared facilitating model exchange, IP reuse and interoperability [26]. The adoption of TLM as a means of exchange of executable models used for design exploration has been proven to significantly reduce the overall SoC design time [70].

SIMD and GPU architectures provide the advantage of a wide range of parallelism founded on a single compiler solution. Using the CUDA programming environment, developers became more attracted to decompose algorithms that lend to explicit parallelism on GPUs [71]. Apple's operating system Snow Leopard, optimized for multi-core processors, exploits the computing power of GPUs [72]. Snow Leopard further extends GPU computing power with Open Computing Language (OpenCL) [73]. OpenCL is a C-based parallel programming language proposed as an open standard by the Khronos group. The popularity of CPU-GPU programming environments like CUDA and SnowLeopard motivates to broaden that integration to FPGAs and DSPs. As an example, the FCUDA, a CUDA-based programming model for FPGAs, maps the coarse and fine grained parallelism of CUDA onto the reconfigurable fabric [74]. The CUDA-to-FPGA flow employs AutoPilot, a high-level synthesis tool enabling high-abstraction FPGA programming [75].

7. CONCLUSION

In the last decade, the evolution of embedded systems was driven by multimedia applications and mobile devices. The solutions that have evolved over time to meet these growing demands are converging into complex structures formed by specific dedicated hardware and reconfigurable platforms that enable the development of efficient, low-power and run-time reconfigurable embedded systems. In fact, continuous IC technology scaling facilitates the emergence of SoC and the access to reconfigurable hardware. Today Multiprocessor architectures are taking care of flexibility and power consumption concerns, even in FPGAs. Moreover, dedicated hardware such as FPGAs, DSP, GPU, reconfigurable processors and common-use IP cores, are now part of the latest commercial portable devices. The emergence of IP cores and heterogeneous architectures is now experiencing a shift towards easy-of-use, regularity and scalability concerns. Regular and flexible architectures, like FPGAs, GPUs, and MPPAs are part of that move. Reconfigurable on chip communication platforms are now part of multi-core architectures. The growing popularity of programming languages such as OpenCL and CUDA illustrates the request for a parallel programming consensus. Software flexibility requirements are now considered by FPGA providers. As we can see in the updated survey presented in this paper, a trend towards convergence of technologies is seen in the development of embedded systems.

8. REFERENCES

[1] D. Mansur, "A New 40-nm FPGA and ASIC Common Platform," IEEE Micro, vol. 29, no. 2, pp. 46-53, Mar./Apr. 2009.

* Work under the Research Action ARC-OLIMP (550.3 / ARC/OLIMP / 1104) Communauté Wallonie-Bruxelles.

[2] S. Bhattacharyya et al., "Advances in hardware design and implementation of signal processing systems," IEEE Signal processing Magazine, vol. 25 , no. 6, pp. 175-180, Nov. 2008.

[3] IBM. (2006) Cell broadband engine: Compiler Technology for Scalable Architectures. [Online]. domino.research.ibm.com

[4] NVidia. (2009) Tegra. [Online]. http://www.nvidia.com/page/handheld.html

[5] L. Benini and G. De Micheli, "Networks on chip: a new SoC paradigm," IEEE Computer, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[6] D. Shin, A. Gerstlauer, J. Peng, R. Domer, and D. Gajski, "Automatic Generation of Transaction Level Models for Rapid Design Space Exploration," in Proc. of CODES+ISSS, Oct. 2006, pp. 64-69.

[7] G. Martin, "System-on-Chip design," in Embedded systems design and verification.: CRC Press, 2009, pp. 13 1-18.

[8] M. Loghi, F. Angiolini, D. Bertozzi, L. Benini, and R. Zafalon, "Analyzing on-chip communication in a MPSoC environment," in Proc. of DATE, Feb. 2004, pp. 752-757.

[9] R. Marculescu, U.Y. Ogras, L-S. Peh, N.E. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 1, Jan. 2009.

[10] Intel. (2007, Feb.) Teraflops Research Chip. [Online]. http://download.intel.com/Teraflops_Research_Chip_Overview.pdf

[11] P. Lysaght and W. Rosenstiel, New Algorithms, Architectures and Applications for Reconfigurable Computing. New York: Springer-Verlag , 2005.

[12] Altera Corp. [Online]. http://www.altera.com

[13] Xilinx. (2004) FPGA embedded solutions. [Online]. http://www.xilinx.com/prs_rls/ip/0492_cmpembedded.htm

[14] Xilinx, Inc. (2009) UG200: Embedded Processor Block in Virtex-5. [Online]. http://www.xilinx.com

[15] D. Shaver. (2008) Platform-based design in the year 2020. [Online]. http://e2e.ti.com/platform-based-design-inthe-year-2020.aspx

[16] J. Henkel, "A low power hardware software partitioning approach for core-based embedded systems," in Proc. of DAC, 1996, pp. 122–127.

[17] D. Gajski, F. Vahid, S. Narayan, and J. Gong, "SpecSyn: an environment supporting the specify-explore-refine paradigm for hardware/software system design," IEEE Transactions on VLSI Systems, vol. 6, no. 1, pp. 84-100, 1998.

[18] A. D. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," IEEE Transactions on Computers, vol. 55, no. 2, pp. 99-112, 2006.

[19] A-J. Gadient, J-A. Debardelaben, and V-K. Madisetti, "Incorporating cost modeling in embeddedsystem design," IEEE Design and Test of Computers, vol. 14, no. 3, pp. 24-35, 1997.

[20] V. Zivkovic, P. VanDerWolf, E. Deprettere, and E-A. DeKock, "Design space exploration of streaming multiprocessor architectures," in Proc. of SiPS, Oct. 2002, pp. 228-234.

[21] S. Mohanty and V. Prasanna, "Rapid system-level performance evaluation and optimization for application mapping onto SoC architectures," in Proc. of ASIC+SOC, 2002, pp. 160-167.

[22] Tensilica. (2009) Xtensa: New DPU Family Delivers 10 GigaMAC/sec DSP Performance. [Online]. www.tensilica.com

[23] S. Leibson, Designing SoC with Configured Cores: Unleashing the Tensilica DIamond Cores Technology, Elsevier Morgan Kaufmann, Ed., 2006.

[24] C. Cooke, H. McNelly, and T. Martin, Surviving the SOC Revolution: A Guide to Platform-Based Design.: Kluwer, 1999.

[25] G Smith, "Platform Based Design: Does it Answer the Entire SoC Challenge?," in Proc. of 41st DAC, 2004, p. 407.

[26] F. Campi et al., "A dynamically adaptive DSP for heterogeneous reconfigurable platforms," in Proc. of DATE, 2007, pp. 9-14.

[27] A. DeHon, "The density advantage of configurable computing," Computer, vol. 33, no. 4, pp. 41-49, Apr. 2000.

[28] L-V Perre, J. Craninckx, and A. Dejonghe, Green Software Defined Radios: Enabling seamless connectivity while saving on hardware and energy.: Springer, 2009.

[29] eSilicon. Embedded FPGA ASIC Design Services. [Online]. http://www.esilicon.com

[30] Abound Logic. [Online]. http://www.aboundlogic.com

[31] J.M. Arnold, "S5: the architecture and development flow of a software configurable processor," in Proc. of ICFPT, Dec. 2005, pp. 121-128.

[32] K. Morris. (2007, Jan.) Actel Activates Platforms. [Online]. http://www.techfocusmedia.net/embeddedtechnologyjournal/feature_articles/20070130-actel/

[33] S. Swan, "SystemC transaction level models and RTL verification," in Proc. of DAC, vol. 43, 2006, pp. 90-92.

[34] K. Hines and G. Borriello, "Dynamic communication models in embedded system co-simulation," in Proc. of DAC, 1997, pp. 395-400.

[35] J-A. Rowson and A. Sangiovanni-Vincentelli, "Interface-based design," in Proc. of the 34th DAC, vol. 34, 1997, pp. 178-183.

[36] L. Cai and D. Gajski, "Transaction-level Modeling: an Overview," in Proc. of CODES+ISSS, 2003, pp. 19–24.

[37] T-H. Tsai, Pan Y-N., and C-H. Lin, "An Electronic System Level Design and Performance Evaluation for Multimedia Applications," in Proc. of ICESS, 2008, pp. 621-624.

[38] F. Doucet, R-K. Shyamasundar, I-H. Kruger, S. Joshi, and R-K. Gupta, "Reactivity in SystemC Transaction-Level Models," in Proc. of HVC, 2007, pp. 34–50.

[39] IBM. (2008, Aug.) Power XCell8i. [Online]. http://www-03.ibm.com/technology_cell_IDC_report_on_PowerXCell.pdf

[40] Texas Instruments. OMAP35x Applications Processors. [Online]. http://focus.ti.com/dsp/docs/dspcontent.tsp?contentId=53403

[41] beagleboard.org.

[42] M. Butts, A-M Jones, and P. Wasson, "A Structural Object Programming Model, Architecture, Chip and Tools for Reconfigurable Computing," in Proc. of FCCM, 2007, pp. 55-64.

[43] STMicroelectronics. (2005, Dec.) "STM Unveils Innovative NoC Technology for New SoC Interconnect Paradigm". [Online]. http://www.stm.com/stonline/press/news/year2005/t1741t.htm

[44] V. Rana, M. Santambrogio, and D. Sciuto, "Dynamic Reconfigurability in Embedded System," in Proc. of ISCAS, 2007, pp. 2734-2737.

[45] J. Worchel. (2006) The FPGA: Expanding Its Boundaries. [Online]. http://www.instat.com/abstract.asp?id=68& SKU=IN0603187SI

[46] S. Gupta, N. Dutt, R. Gupta, and A. Nicolau, "SPARK : a high-level synthesis framework for applying parallelizing compiler transformations.," in Proc. of VLSID, 2003.

[47] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, FPGA-based Implementation of Signal Processing Systems. West Sussex, UK: John Wiley & Sons, 2008.

[48] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An Overview of Reconfigurable Hardware in Embedded Systems," EURASIP Journal on Embedded Systems, p. 19, 2006.

[49] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," Foundations and Trends in Electronic Design Automation, vol. 2, no. 2, pp. 135–253, 2007.

[50] Atmel Corp. (2009) FPSLIC™ (AVR with FPGA). [Online]. http://www.atmel.com/products/FPSLIC/

[51] eASIC Corp. [Online]. http://www.easic.com/

[52] F. Berthelot, F. Nouvel, and D. Houzet, "Partial and dynamic reconfiguration of FPGAs: a top down design methodology for an automatic implementation," in Proc. of IPDPS, vol. 20, 2006.

[53] E-M. Panainte, K. Bertels, and S. Vassiliadis, "The Molen Compiler for Reconfigurable Processors," ACM Transactions on Embedded Computing Systems, vol. 6, no. 1, Feb. 2007.

[54] R. Lysecky, G. Stitt, and F. Vahid, "Warp processors," ACM Transactions on Design Automation of Electronic Systems, vol. 11, no. 3, pp. 659-681, 2006.

[55] J. Lu, H. Chen, P. Yew, and W. Hsu, "Design and implementation of a lightweight dynamic optimization system.," Journal of Instruction-Level Parallelism, vol. 6, pp. 1-24, Jun 2004.

[56] G. Stitt and F. Vahid, "Thread Warping: A Framework for Dynamic Synthesis of Thread Accelerators," in Proc. of CODES+ISSS, 2007, pp. 93-98.

[57] E-L. Horta, J-W. Lockwood, and D. Parlour, "Dynamic Hardware Plugins in an FPGA with Partial Run-Time Reconfiguration," in Proc. of DAC, 2002.

[58] N. Moore, A. Conti, M. Leeser, and L. Smith-King, "Writing Portable Applications that Dynamically Bind at Run Time to Reconfigurable Hardware," in International Symposium on Field-Programmable Custom Computing Machines FCCM, 2007.

[59] J. Dondo et al., "Persistence management model for dynamically reconfigurable hardware," in Proc. of 13th Euromicro-DSD, 2010.

[60] W-0. Cesario et al., "Component-based design approach for multicore SoCs," in Proc. of DAC, 2002, pp. 789-794.

[61] H-J. Brand, S. Rulke, and M. Radetzki, "IPQ, IP Qualification for Efficient System Design," in Proc. of the 5th International Symposium on Quality Electronic Design ISQED, 2004, pp. 478-482.

[62] SPIRIT. [Online]. www.spiritconsortium.org

[63] (2010) Accelera. [Online]. www.accellera.org

[64] VSI Alliance. (2001) Virtual Component Interface Standard, OCB 2 2.0. [Online]. www.ocpip.org

[65] OCP-IP Association. (2009) Open Core Protocol. [Online]. www.ocpip.org

[66] J. Brunel et al., "COSY communication IP's," in Proc. of DAC, 2003.

[67] P. Paulin et al., "Parallel Programming Models for a Multiprocessor SoC Platform Applied to Networking and Multimedia," IEEE Transactions on VLSI systems, 2006.

[68] P. Paulin et al., "Distributed Object Models for Multi-Processor SoCs, With Application to Low-Power Multimedia Wireless Systems," in Proc. of DATE, Munich, Germany, 2006, pp. 482-487.

[69] F. Rincón et al., "Transparent IP Cores Integration Based on the Distributed Object Paradigm," Lecture Notes in Electrical Engineering, vol. 38, no. 10, pp. 131-144, 2009.

[70] Synopsys. (2008) CoCentric System Studio. [Online]. www.synopsys.com

[71] A. Bleiweiss, "GPU Accelerated Pathfinding," in Proc. of the 23rd ACM SIGGRAPH/EUROGRAPHICS, 2008, pp. 65-74.

[72] Apple. (2008) Snow Leopard. [Online]. http://www.apple.com/pr/library/2008/06/09snowleopard.html

[73] Khronos. (2009) OpenCL The open standard for parallel programming of heterogeneous systems. [Online]. http://www.khronos.org/opencl/

[74] A. Papakonstantinou et al., "FCUDA: Enabling Efficient Compilation of CUDA," in Proc. of the 23rd ICS, 2009, pp. 515-516.

[75] AutoESL. (2009) AutoPilot. [Online]. http://www.autoesl.com