Review Article

Ingredients of Adaptability: A Survey of Reconfigurable Processors

Anupam Chattopadhyay

MPSoC Architectures, UMIC Research Centre, RWTH Aachen University, Mies-van-der-Rohe Strasse 15, 52074 Aachen, Germany

Correspondence should be addressed to Anupam Chattopadhyay; anupam@ice.rwth-aachen.de

Received 18 December 2012; Revised 14 May 2013; Accepted 1 June 2013

Academic Editor: Yann Thoma

Copyright © 2013 Anupam Chattopadhyay. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For a design to survive unforeseen physical effects like aging, temperature variation, and/or emergence of new application standards, adaptability needs to be supported. Adaptability, in its complete strength, is present in reconfigurable processors, which makes it an important IP in modern System-on-Chips (SoCs). Reconfigurable processors have risen to prominence as a dominant computing platform across embedded, general-purpose, and high-performance application domains during the last decade. Significant advances have been made in many areas such as, identifying the advantages of reconfigurable platforms, their modeling, implementation flow and finally towards early commercial acceptance. This paper reviews these progresses from various perspectives with particular emphasis on fundamental challenges and their solutions. Empowered with the analysis of past, the future research roadmap is proposed.

1. Introduction

The changing technology landscape and fast evolution of application standards make it imperative for a design to be adaptable. Adaptability is a mandatory part of some major research initiatives such as cognitive radio, where the radio transceiver implementation takes cognition of its surrounding environment. Adaptability can be present in the device, in the circuit, in the microarchitecture, or even in the runtime software layer or among all of these. In this paper, we survey reconfigurable processors, which provide a complete spectrum of adaptability in the processor microarchitecture. The survey is primarily intended for developers and users of adaptive digital systems.

Since the introduction of programmable logic devices by Xilinx in 1984 [1], the domain of reconfigurable computing has steadily gained acceptance first as the prototyping platform and second as the computing platform of choice. The transition of reconfigurable computing devices from the role of prototyping to computing is what presented in 2001 by Hartenstein in his paper “A decade of reconfigurable computing: a visionary retrospective” [2]. In [2], it is mentioned that reconfigurable platforms are able to bridge the gap between processors and Application-Specific Integrated

Circuits (ASICs) in terms of flexibility and performance. Since this work, notable research has been done in accelerator design (application-specific processors), multicore homogeneous and heterogeneous System-on-Chip (SoC) architectures, and smooth high-level synthesis of various kinds of computing devices. From the application domain, an ongoing blend between high-performance computing, general-purpose computing, and embedded computing is noticeable. This also prompts development of interesting architectures which combine a programmable processor with a reconfigurable device [3] or place a processor as macro block inside a reconfigurable platform [1].

Understanding the dynamics and rationale of this development is important for designing efficient architectures, tools and predict the research roadmap. This is exactly the purpose of this current review paper. Noting that little more than a decade have passed since the visionary retrospective by Hartenstein [2], this paper attempts to chronicle the developments since then. To limit the breadth of the discussion, the emphasis is on a particular kind of reconfigurable devices, which are programmable through high-level languages. Generally, these kinds of computing platforms are denoted as reconfigurable processors.

Before proceeding further, we provide an overview of the existing survey papers in the field of reconfigurable processors and in general in the area of *reconfigurable computing*.

1.1. Related Surveys. Several noteworthy surveys of reconfigurable processors have been done in the past. The survey by Hartenstein [2] contrasted the traditional von Neumann computing paradigm and the evolving behavioral synthesis paradigm with the proposed reconfigurable datapath architecture concept. The existing reconfigurable processors are reviewed in detail from their architectural, programming, and design space exploration perspectives. The emerging theme of cocompilation is placed under the structural programming paradigm instead of procedural programming supported by von Neumann architectures. Apart from being outdated, the major drawback of [2] is relatively less focused on design space exploration. This is natural since the high-level design flow for programmable processors [4–6] was at a nascent stage during the time of the survey. In 2003, a more detailed view of the reconfigurable processor structure is presented in [7]. This paper considers a reconfigurable processor from hardware-software design perspective. Different possible design decisions for each microarchitectural block as well as for each segment of the software tool flow are identified. Existing reconfigurable processors are categorized as per those specific design points. A limitation of this survey is to view the reconfigurable processors as solely a processor where the instruction set can be reconfigured. The reconfigurable processors, which do not expose the reconfigurability through the instruction set, cannot be included in this work. This excludes major reconfigurable processors such as [3]. Nonetheless, the decade-old surveys by Hartenstein [2] and Barat et al. [7] made influential suggestions regarding the emerging course of reconfigurable processor roadmap.

A brief survey of reconfigurable platforms, with focuses on commercially available technologies, is presented in [8]. A detailed survey of reconfigurable systems and software is presented [9] in 2002. This survey focused on fine-grained FPGA architectures and the software tools for synthesizing applications on that. In that regard, the available high-level FPGA-specific synthesis flows are covered in brief. A detailed treatment for reconfigurable processors is omitted partly because that was still an emerging research area. This is included in a later survey by Todman et al. [10]. With the natural progress of close coupling between reconfigurable blocks and base processor since the earlier survey of Compton and Hauck [9], the survey in [10] included an additional reconfigurable system class (Figure 1 in [10]). In [10], coarse-grained fabrics and soft cores are noted as two major emerging design trends. For design methods, major trends are predicted as high-level transformations, special-purpose methods (e.g., word-length optimization for DSP), and low-power transformations. Reconfigurable processors are covered in details for soft-instruction processors, that is, the processors which are embedded in the FPGA fabric. To the best of our knowledge, the most recent authoritative survey on fine- and coarse-grained reconfigurable architectures is presented by Vassiliadis and Soudris [11] in 2007. There,

a survey of reconfigurable architectures and corresponding CAD tools is presented followed by few case studies.

There has been several books with detailed treatment of reconfigurable computing. In chronological order the books are by Bobda [12], by Hauck and DeHon [13], and by Sass and Schmidt [14]. The later two books focused on application development and synthesis on FPGAs. In [12], coarse-grained reconfigurable architectures are discussed, but reconfigurable processors are not provided a detailed treatment. Recently, a book with focus on run-time adaptation by Bauer and Henkel [15] and a book emphasizing high-level modeling approach for reconfigurable processors by Chattopadhyay et al. [16] are published.

There has been several developments in the processor design research, which are partly covered or completely missed in the previous surveys. These developments are briefly mentioned here:

- (i) major research and commercial advances in the field of accelerator/processor design automation [17–19];
- (ii) development of Multi-Processor System-on-Chips (MPSoCs) to tackle the energy efficiency [20];
- (iii) prominent commercial processor vendors adopting reconfigurable devices as part of the computing fabric [3, 21];
- (iv) rise of commercial high-level design flows for custom reconfigurable architectures [22];
- (v) theoretical studies comparing non reconfigurable and reconfigurable processors [23].

In this survey, we intend to review the aforementioned developments together with reconfigurable processor design advances. Additionally, several clear taxonomies and definitions for reconfigurable processor are provided.

Organization. The rest of the paper is organized in such a way to present different perspectives in a coherent yet independent manner. Inquisitive readers interested in any particular aspect of reconfigurable processor can directly jump to any section for a self-sufficient content, though it is recommended to cover at least the definitions in Section 2.1 for better readability. Section 2 discusses a key theoretical insight, which acted as a major driving force behind reconfigurable computing research during the last decade. In Section 3, the evolution of architectural design space for reconfigurable processors is discussed. A slow but steady foray of reconfigurable processors into diverse application domains as the main computing platform is discussed in Section 4. A relatively novel area of high-level modeling and automatic toolsuite generation for reconfigurable processors is surveyed in Section 5. The future challenges of reconfigurable processor research are detailed in Section 6. The paper is concluded in Section 7.

2. Theoretical Insights

It took more than a decade since the original introduction of FPGAs [24] to fully appreciate its advantages as a medium

of computing. In a significant contribution [23], DeHon explained analytically why FPGAs are more efficient compared to say Digital Signal Processors (DSPs) for certain tasks. For that, a simple area model is constructed in [23]. By taking technology feature size as a parameter, a RISC processor and an FPGA are compared in a technology-independent manner. For the RISC processor with 32-bit ALU, it is shown that an FPGA in comparable technology offers 10× more computation in the same die area. This advantage of FPGA is termed as higher *computation density*.

The ASIC has a fixed set of processing elements and fixed interconnect. It reacts to the input pins to generate the output. Naturally, the computation density of ASIC is highest among various computing devices. The processor possesses one or few dedicated execution units, which receive the data from the storage. The movement of data to/from the storage and the triggering of the exact operation in the execution unit are controlled by the current instruction. Thus, apart from the actual computation, a considerable die area is spent to decode the instruction, store the current instruction state, and move the operands. This makes the computation density of processors comparatively lower. Finally, for a reconfigurable device, the computing blocks and the interconnects are configured a priori by configuration bits. Once the actual computation starts, the complete die is utilized for performing the computations. This makes the computation density of the FPGA comparatively higher than processors. However, it is still lower than ASICs since reconfigurable devices retain a general-purpose routing and general-purpose execution blocks (based on Look-Up Tables) for accommodating arbitrary application prototyping. Attempts to reduce this gap are done by customizing reconfigurable devices for specific application(s) [25].

The density advantage of reconfigurable computing also comes at a cost [23]. To prepare the FPGA for a series of densely packed computations, the configuration bitstream needs to be large. This affects the total computation capacity of the FPGA and its reconfiguration overhead. As a result, multiple time-consuming reconfiguration is needed if either the application is too large or if the application is too control-oriented.

Note that to attain the density advantage alike reconfigurable devices processors have been customized. To take advantage of the spatial and temporal parallelism of an application, SIMD instructions [26] and fused custom instructions [27] are used. Such instructions basically reduce the decoding overhead and accomplish more operations in one go. On the other hand, the huge parallelism supported by reconfigurable device requires long configuration words and wide databus. These issues affected the customization of processors [27–29].

The density advantage of FPGA applies in general to the family of reconfigurable devices. It shows that efficient design points can be obtained by exploring instruction decoding, computation, and communication patterns. This theoretical insight of DeHon [23] confirms the observations on prior experimental results and has since been empirically validated by many designs [30, 31].

2.1. Definitions. The literature on reconfigurable processors is full of varied terminologies and keywords, which is often confusing for general readers and makes it hard to distinguish a reconfigurable processor from a non-reconfigurable one. In the following, few definitions for key terms are proposed, which are used throughout the paper.

Definition 1. Programmability: when a computing device can be controlled by high-level languages, then it is called programmable. The property to be programmable is called programmability.

Usually, programmable devices are referred to as *processors*. In contrast, computing devices with controllability to alter functionality via low-level switches are said to have *configurability*.

Definition 2. Reconfigurability: when a computing device can be altered to perform a different task by low-level hard-wired then control, it is called reconfigurable. The property to be reconfigurable is called reconfigurability.

Note that a device can be both programmable and reconfigurable. This kind of devices is termed *reconfigurable processor*. For example, the ADRES architecture reported in [30] is a reconfigurable processor.

Definition 3. Partial reconfigurability: when a reconfigurable device supports reconfiguration of part of it, then this property is called partial reconfigurability.

Definition 4. Dynamic reconfigurability: the ability to alter the functionality of a reconfigurable device when a part of the device is busy executing some task is called dynamic reconfigurability.

Note that every reconfigurable device can be loaded with a configuration before it starts executing the tasks. Dynamically loading the configuration makes it necessary that the device is partially reconfigurable so that some part of the device is busy working. However, in practice dynamic reconfigurability is also used to denote reconfigurable devices which are attached to a processor as an execution unit. In these cases, the processor is actually busy doing some tasks (i.e., reconfiguring its execution unit), when the reconfigurable device is being reconfigured.

The supported reconfigurability can be available as dynamic/static and partial/complete. The reconfigurability can be available for the datapath and/or control path. For a processor, the control path also includes the instruction decoding logic. When altering the instruction set is supported by configurability of the decoding logic, then the reconfigurable processor is denoted as *Reconfigurable Instruction Set Processor (RISP)* [7] or *Rotating Instruction-set Processing Platform (RISPP)* [32]. When a reconfigurable processor is application specific (in contrast to general purpose), then it is coined as *reconfigurable Application-Specific Instruction-set Processor (rASIP)* [16].

FIGURE 1: Exemplary fine-grained and coarse-grained reconfigurable architecture.

Definition 5. CGRA: Coarse-Grained Reconfigurable Architecture (CGRA) is a reconfigurable device, which has application-specific or domain-specific operators and routing architecture.

CGRAs are also denoted as application-specific reconfigurable devices or embedded FPGAs (eFPGAs) [25]. CGRAs are marked for their improved performance compared to fine-grained, general-purpose FPGAs. CGRAs also have less reconfiguration time due to the smaller size of their configuration word in comparison with FPGAs.

Reconfigurability in a device, in a basic form, can be supported by multiplexers. Efficient transistor-level implementation of the reconfigurability in routing is done for commercial FPGAs. Reconfigurability of operators is either achieved by Look-Up Tables (LUTs) based on SRAMs or by multiplexing various functional units as in CGRAs. With increasing foray of FPGAs towards computing block instead of traditional prototyping, the primary computing elements inside an FPGA have become feature rich. These primary computing elements or building blocks inside an FPGA typically contain more than one LUT, more than one flip-flop, and a mix of arithmetic, combinatorial, and multiplexing logic. Prominent device vendors use different terminologies to refer to these building blocks. While Xilinx [1] uses Configurable Logic Block (CLB) to indicate that, Altera [33] names it Logical

Element (LE). To indicate the complexity of the FPGA devices further terminologies such as Adaptive Logic Module (ALM), Equivalent Logic Cell (ELC), or Advanced Silicon Modular Block (ASMBL) are also being used. A detailed treatment of physical design of reconfigurable devices is covered in [13].

Figure 1 shows the organization of fine-grained and coarse-grained reconfigurable fabrics. On the left side of the figure, fine-grained logic block, based on LUT, is shown. Below that, a typical mesh-based routing network of an FPGA is presented. The right-side of the figure shows a coarse-grained Processing Element (PE) with ALU as the main operator. The coarse-grained routing network is also designed for specific application(s) with performance improvement in sight.

Programmable Logic Devices: an early precursor of reconfigurable devices, known as Programmable Logic Device (PLD), provided the option to implement arbitrary functionality via a sea of gates, which are typically organized as an AND-OR structure. By allowing flexibility in the AND-plane or in both AND and OR-plane, Programmable Logic Array (PLA) or Programmable Array Logic (PAL) devices could be realized, respectively. Field-Programmable Gate Arrays (FPGAs) became the dominant reconfigurable device by offering significantly increased prototyping complexity, support of large SRAMs, and application-specific combinatorial blocks. Another major difference between PLDs and FPGAs

is the way PLDs are configured. Except for few large complex PLDs, PLDs [34] are typically configured by EEPROMs or silicon antifuses, whereas FPGAs are configured by SRAMs. SRAM-based configuration requires longer time to download but does not require fine configuration control as in silicon antifuses. For few PLD devices, the configuration is stored via EEPROM and then loaded to the SRAM on boot-up providing non volatility and high degree of reconfigurability [35].

Note that, though PLDs are termed as *programmable*, in the proposed taxonomy they belong to reconfigurable devices category.

2.2. Theoretical Models. In order to bridge the gap between highly parallel applications, which have strong optimization potential in reconfigurable processors, an abstract implementation model is required. Several prominent parallel machine models are mentioned in the following.

PRAM [36]: Parallel Random Access Machine (PRAM) consists of a collection of processors which compute synchronously in parallel and communicate with a global random access memory. Its significant limitation is the assumption of large shared storage, which is impractical for high number of independent computing nodes.

BSP [37]: Bulk Synchronous Parallel (BSP) model assumes processing nodes connected by a communication network. Each node has a local storage. Another key idea of a BSP is to have a *superstep*. A superstep consists of first, concurrent computations, second, communication between nodes, and finally barrier synchronisation among all the concurrent computations. An extension of BSP model, known as Heterogeneous BSP (HBSP), with relative speeds of its components in heterogeneous environment is proposed in [38].

LogP [39]: the LogP term is coined as an abbreviation of the four parameters this model represents, namely, latency of communication, overhead of a processing node to send or receive a message, gap between successive transmissions allowed by a computing node, and processor count.

The aforementioned models, arranged in their chronological order of appearance, also represent an evolution from the computation-centric to a communication-centric view. For judging the implementation efficiency of an algorithm to reconfigurable processor, the ratio of computation and communication needs to be studied. Optimal performance is achieved when these are balanced; that is, communication bandwidth is enough to serve the computation demand.

The application of BSP model, for example, is explained in [37] with the help of a matrix multiplication example. Suppose that two matrices of $n \times n$ dimension are multiplied using $p \leq n^2$ processors. It is shown that the optimal runtime of $O(n^3/p)$ is achieved only when a particular distribution of local tasks and communication bandwidth is maintained. This leads to direct practical implementation suggestions [31].

3. Design Choices for Reconfigurable Processor

In this section, the related work in the field of reconfigurable processor design is traced. A complete taxonomy of the

growing number of reconfigurable processors is only possible when the complete design space is taken into account. This is done by carefully studying the development of design choices over the course, starting from the earliest reconfigurable processor. While this means that there is an overlap in the information with respect to the existing survey works, yet this is intended to provide a complete reference. In Figure 2, several design dimensions including the user view, target application domain, and the microarchitectural choices are identified. In each of these dimensions, the progress of reconfigurable processor design points noted. Naturally, a reconfigurable processor combines several design ideas together to best meet its purpose.

During the evolution, as shown in Figure 2, every design represents a consolidation and expansion of earlier research results rather than a unique, completely novel design point. In many of the designs, the reconfigurability is achieved by a separate, closely coupled block, and the programmability is achieved by an existing mainstream processor. The former is referred to as the reconfigurable block following existing terminology and for latter, base/host processor is used. It is interesting to note that for earlier designs, where the reconfigurable block is coupled on a board with a separate processor, host processor is more commonly used to denote the base processor. In the recent designs, this is integrated on a single chip, and, base processor is used more appropriately. On-board coupling is still a common integration method for emulation platforms.

According to the design dimensions presented in Figure 2, a tabular, chronological feature summary of the reconfigurable processors is presented in Table 1.

For Pleiades [56], the work reports one architecture out of a concept, which can have the coprocessor as an ASIC, FPGA, or CGRA. For that reason, the granularity of reconfigurable block is categorized as flexible. Note that the control is distributed in RaPiD [47], KressArray [46], and MATRIX [48] and shows a shift from uniprocessor to multiprocessor systems. This is also noted in MorphoSys [54] and Chameleon [57], which are referred to as a reconfigurable System-on-Chip (SoC) instead of a reconfigurable processor. Along similar lines, the RAW processor [70] closely resemble, recent homogeneous multicore architectures. In fact, the RAW architecture research acted as the foundation of the commercial manycore Tiler [71] and from that perspective, it is hard to group these architectures within reconfigurable processors. A classification is proposed in Section 3.3 of this paper.

The developments of Pleiades, ADRES, and Stretch underscore the fact that after considerable advances in identifying design points of reconfigurable processor, tools and design methodology for comprehensive design space exploration started emerging. FLEXDET is the latest example in this trend, where the complete reconfigurable processor including its software programming tools and the hardware implementation is automatically derived from a high-level language specification. The aforementioned list of reconfigurable processors is not exhaustive. Nevertheless, those well represent the expansion of the design space and the historical trends.

TABLE 1: Feature summary of reconfigurable processors.

Reference	General		Software view		Coupling	Reconfigurable block	
	Year	Domain	Programming	ISA		Granularity	Reconfiguration
PADDI [40, 41]	1992	DSP	Silage compiler	VLIW	Loose	Coarse	—
PRISM [42]	1993	General purpose	HLL compiler	M68010	Loose	Fine	Static
Nano processor [43]	1994	General purpose	Assembly	RISC	Tight	Fine	Static
Spyder [44]	1995	General purpose	HLL compiler	VLIW	Coprocessor	Coarse	—
DISC [45]	1995	General purpose	Assembly	CISC	Tight	Fine	Dynamic
KressArray [46]	1995	General purpose	ALE-X compiler	VLIW	Loose	Coarse	—
RaPiD [47]	1996	DSP	HLL compiler	RISC	Loose	Coarse	—
MATRIX [48]	1996	General purpose	Assembly	SIMD/VLIW	Loose	Coarse	Dynamic
Garp [49, 50]	1997	General purpose	HLL compiler	MIPS	Coprocessor	Fine	Dynamic
NAPA [51, 52]	1998	General purpose	NAPA C compiler	RISC	Coprocessor	Fine	Dynamic
REMARC [53]	1998	Multimedia	Assembly	Custom MIPS	Coprocessor	Coarse	Dynamic
MorphoSys [54, 55]	2000	Multimedia	HLL compiler	Tiny RISC	Tight	Coarse	Dynamic
Pleiades [56]	2001	DSP	HLL compiler	ARM	Coprocessor	Flexible	Dynamic
Chameleon [57, 58]	2002	DSP	HLL compiler	ARM	Coprocessor	Fine	Dynamic
XiRISC [59, 60]	2003	Embedded, multimedia	HLL compiler	Custom RISC	Tight	Fine	Dynamic
Chimaera [61, 62]	2004	Multimedia	HLL compiler	MIPS	Tight	Coarse	Dynamic
MOLEN [63, 64]	2004	General purpose	HLL compiler	PowerPC	Tight	Fine	Dynamic
Stretch [65]	2004	Embedded	HLL compiler	RISC	Tight	Fine	Dynamic
ADRES [30, 66]	2005	Wireless	HLL compiler	VLIW	Tight	Coarse	Dynamic
BUTTER [67]	2008	Multimedia	HLL compiler	RISC	Tight	Coarse	Dynamic
KAHRISMA [68]	2010	Embedded	HLL compiler	RISC/VLIW	Tight	Mixed	Dynamic
DySER [69]	2012	General purpose	HLL compiler	RISC/CISC	Tight	Coarse	Dynamic
FLEXDET [31]	2012	Wireless	HLL compiler	Custom RISC	Tight	Coarse	Dynamic

FIGURE 3: Reconfigurable processor design space: microarchitectural view.

FIGURE 4: Reconfigurable processor design flow: programming view.

FIGURE 5: Reconfigurable processor versus homogeneous multicore SoC.

3.3. Homogeneous Multicore SoC or CGRA? The unbalanced scaling of power and performance in shrinking CMOS technologies led to the growth of multicore architectures. Armed with huge parallelism, General-Purpose Graphics Processing Units (GPGPUs) offer significant performance improvement for embarrassingly parallel applications. The structure of GPGPUs and some multicore machines closely resemble the CGRA microarchitecture. In that way, it gets hard to distinguish between data flow architectures such as REDEFINE [75], multicore SoCs [71, 76], and a CGRA [30]. In particular, several reconfigurable architectures, for example, RAW [70] and MATRIX [48], offered distributed decoding in its elementary processing units like the cores of a multicore SoC. In fact, several research developments of

RAW led to the foundation of multicore SoC provider Tileria [71].

In Figure 5 a reconfigurable processor with CGRA as execution unit is shown vis-a-vis a GPGPU architecture. We note several strict classifications that can be used to draw the boundary between a multicore SoC and a reconfigurable architecture.

- (i) A reconfigurable architecture offers a richer inter block communication than a multicore SoC. The rationale is that homogeneous multicore SoCs are geared for general-purpose programming. There, to aid the programmer, a simplistic view of storage is necessary. This is achieved in GPGPUs by having

dedicated local registers and shared memories. The presence of rich interconnect is also what makes the task of compilation for heterogeneous multicore SoCs challenging.

- (ii) Because of the rich interconnect architecture and detailed low-level configuration control, reconfigurable blocks require a detailed compilation flow with placement and routing. In contrast, multicore SoCs rely on dynamic task management via control-plane cores and support of intelligent communication architectures for task distribution across its constituent cores. Note that there are some reconfigurable processors, which take idleness of resources into account [68] during runtime. However, the dynamic decisions for reconfigurable processors are usually limited to reconfiguration.
- (iii) From the user's point of view, a reconfigurable processor is a single processing platform. This has implications on the programming model. For multicore SoCs, for example, GPGPU, the user is required to provide explicit computation partitioning among tasks in the CUDA programming language. For reconfigurable processors, the partitioning of operators among the PEs of a CGRA/FPGA is fully automated.

In short, the homogeneous multicore SoCs trade off performance in order to achieve programmer productivity. This is reflected in the programming view (unified programming model for reconfigurable architecture), architecture details (rich interconnect for reconfigurable architecture), and also in the support of tools (automated compilation support for reconfigurable architectures).

4. Classification of Reconfigurable Processors

The move of reconfigurable devices from prototyping to computing [77] is motivated by theoretical insights as well as convincing design studies as presented in the previous sections. In the last few years, this trend is more prominent due to several reasons. *First*, the increasing complexity of modern embedded devices without significant increase in the battery capacity makes it important to understand the interplay between performance and energy efficiency. It is not anymore sufficient to design a highly advanced processor without any concern for energy. This prompted designers to look for various architectural choices, which match the pattern of the application [78, 79]. *Second*, increasing manufacturing and Non-Recurring Engineering (NRE) costs demands a design to be more flexible and tolerant of manufacturing and process variations via post fabrication design alterations. Reconfigurable devices provide a way to mitigate the effects of late design errors, process variations, and also ensure longer time-in-market by regular design updates. *Finally*, the emergence of intelligent devices in our everyday life makes embedded computing a major market compared to general-purpose computing. This made the prominent general-purpose computing vendors to look into the embedded

architecture demands [3], resulting into general-purpose and domain-specific reconfigurable processors. However, there are general-purpose reconfigurable processors being applied with strong benefit within certain domains. Therefore, such a classification is hard. Instead, we offer a classification based on structure and application domains.

4.1. Reconfigurable Processors: Structural Classification. Reconfigurable processors are offered with various configurations, though there are three distinct structural alternatives. While the survey in [10] adopted five design classifications based on the coupling between reconfigurable block and the base processor, we advocate the following three classes.

Class I: Add-On Reconfigurability (AoR). This trend is started by the general-purpose desktop/embedded processor vendors. Here, the goal is to maintain the baseline of processing and full software programmability support. On top of that, reconfigurability is provided for a block in order to increase performance and postmanufacturing flexibility. A prominent example of this is the Intel Atom Processor E6x5C series-based platform, which combines the processor with a fine-grained commercial FPGA from Altera via a high-speed PCI Express bus [3]. Note that, though the processor itself does not provide any reconfigurable block, it can offload its computing task to the reconfigurable logic via the high-speed bus, putting it in the same category as with earlier loosely coupled reconfigurable processors such as Spyder [44]. New studies of this class of reconfigurable processors keep appearing such as DySER [69], which clearly demonstrates the advantage of CGRA computing fabrics as execution units within general-purpose processors.

Class II: Add-On Processing (AoP). This trend is started by reconfigurable device vendors, for example, Xilinx. In this class, the overall architecture is reconfigurable block with a part of it being allocated to processors. A prominent example is Xilinx Zynq-7000 family of FPGAs, for which ARM Cortex-A9 MPCore is provided [1]. The base fabric is reconfigurable, where the processing is added on for ease of programmability and dynamic control.

Class III: Custom Processing and Reconfigurability. A third implementation option, which allows picking up a customizable processor with a custom-designed reconfigurable fabric, is offered by few designs, for example, ADRES [66]. Here the goal is to completely tune the processing and reconfigurability to a specific target domain [31, 80].

These first two classes are shown graphically in Figure 6. The upper part of the figure shows a reconfigurable processor from the AoP class, whereas the bottom one gives an example of the AoR class of reconfigurable processor. The reconfigurable logic could be more easily utilized with high-level compilation flow, as done in the AoP class of reconfigurable processors. On the other hand, reconfigurable logic provides parallelism and pipelining opportunities at subword-level granularity, which is leveraged in the AoR class of reconfigurable processors. However, in both cases,

FIGURE 6: Reconfigurable processor organization.

a general-purpose nature of the reconfigurable processor is maintained.

Belonging to a structural classification does not impose any strong restriction on the usability of the reconfigurable processors except the last one. For domain-specific processors, the hard performance constraints in terms of area, processing speed, and energy efficiency demand a customized design as in Class III. This implementation option is rarely found in general-purpose reconfigurable processors [1, 59, 63].

4.2. Reconfigurable Processors: Usage Classification. While the primary use-case of general-purpose reconfigurable processors still remains as prototyping and demonstration, increasing use of all classes of reconfigurable processors in diverse application domains is notable. Naturally, this also creates a strong pull of new reconfigurable processor designs with custom processing and reconfigurability (Class III), as discussed below.

Stretch [65] offers a customizable processor template connected to a coarse-grained reconfigurable fabric (ISEF). Designers can customize Stretch series of processors to target specific application(s). For the W-CDMA protocol, a partially reconfigurable processor is designed in [74]. For the MIMO-OFDM multimode detector FLEXDET [31] is proposed. ADRES [66] is designed for Software Defined Radio applications. For image/video processing a reconfigurable ASIP is proposed in [81]. Shafique and Henkel proposed a reconfigurable architecture [82] for low-power multimedia applications. For baseband receiver [83] and fast arithmetic operations [25] dedicated reconfigurable processors are proposed. Not all of these architectures are deployed for commercial purposes. The massive parallelism offered by reconfigurable platforms has been successfully used in big data analytics and high-performance computing applications [84]. To gauge the commercial impact of the emerging reconfigurable processors, few interesting details of commercial reconfigurable processors are presented in

Table 2. The year of introduction is used as the reference year in the first column.

From the application perspective a study of reconfigurable processors reveals that, while new application areas are exploited by Class III reconfigurable processors, Class I and Class II reconfigurable processors typically are used for applications demanding massive parallel processing, like data analytics and, in some cases, public-key cryptography, which require arithmetic with large numbers. Reconfigurable processors are capable of supporting instruction-level and data-level parallelism with mixed granularity and therefore remain the ideal implementation option when all these features are present in a certain application. In absence of the right set of profiling tools to rate the processors for certain applications, the approach that has been taken by current designers is to perform a bottom-up kernel-specific design [79] and then apply that architecture to an application domain [31]. The structural and domain-wise classification of reconfigurable processors is presented in Figure 7. It can be noted that there are many instances of each structural class of reconfigurable processor being used in diverse application domains. The known mappings are presented with dotted arrows in Figure 7.

It can be observed from Table 2 that, despite significant promise of reconfigurable processors, there is still a dearth of commercial interests in those. This is due to the two following reasons.

First, there is a lack of advanced design tools. Even for major design houses, the reconfigurable processor is usually taking two different pieces of architecture and joining those. This is shown in the scarcity of integrated tool-flow in the commercial reconfigurable processors. A separated tool-flow moves the issue of programming completely to the high-level programming and compilation environment. However, a unified view of the datapath, resources, and ISA is presented in many academic designs.

The secondary reason behind slow uptrend of commercial reconfigurable processors is increasing manufacturing costs. Unlike many digital designs, which are ported on a low-cost FPGA, the reconfigurable processor may require to have a custom coarse-grained/fine-grained reconfigurable architecture [30]. This makes it important for a reconfigurable processor to have a manufacturing phase, possibly with semi-custom ASIC design flow, which is prohibitively expensive for advanced CMOS technology nodes.

An interesting and nice trend observable in the commercial reconfigurable processors is the conversion of several academic reconfigurable processors into commercial ones, notably ADRES [90] and Recore Systems [91], which started from ADRES [30] and Montium/Chameleon [58, 93], respectively.

The above usage classification is solely based on the application domains. A more abstract classification based on design patterns has been proposed [94]. A detailed survey of usage of those patterns in reconfigurable architecture research is also presented. However, in absence of a pattern-identification tool flow or a seasoned application developer, it is hard to exploit the optimization possibility with abstract

TABLE 2: Survey of commercial reconfigurable processors.

Reconfigurable processor (year)	Application domain	Base processor, ISA	Max. frequency	Reconf. block	Tool support
Stretch (2004) [65]	Embedded	Custom VLIW [85]	400 MHz @65 nm (Stretch S7100)	Coarse-grained	Integrated
AMD Torrenza (2006) [21]	High performance, general purpose	x86-64	3.3 GHz	Fine-grained, Altera, Xilinx	Separate
Intel Atom processor E6x5C series (2010) [3]	Embedded, general purpose	MMX, SSE, SSE2, SSE3, SSSE3, IA-32, x86-64	1.3 GHz	Fine-grained, Altera	Separate
IBM DB2 analytics accelerator (2010) [84]	High performance, data analytics	—	—	Fine-grained, Commercial	Integrated flow, Lime [86]
Altera FPGA Series (2011) [87, 88]	Embedded	ARM Cortex-A9 MPCore, Nios II soft processor	2 GHz, 340 MHz [89]	Fine-grained, DSP macro blocks	Separate
Xilinx FPGA Series, Zynq-7000 family (2011) [1]	Embedded	ARM Cortex-A9 MPCore, MicroBlaze soft processor, PowerPC	2 GHz — —	Fine-grained, DSP macro blocks	Separate
ADRES (2011) [30, 90]	Wireless baseband	Custom VLIW	700 MHz @40 nm	Coarse-grained, Customizable	Integrated
Recore Systems (2011) [91]	Embedded, DSP	Custom DSP [58]	—	Coarse-grained	Integrated
ARM SoC FPGA (2012) [92]	Embedded	ARM v4T ISA	200 MHz	Fine-grained, Altera	Separate

FIGURE 7: Reconfigurable processor classification.

patterns. This remains as one of the important open challenges.

5. Modeling and Design Space Exploration Tools

The advent of customized accelerators and high-level accelerator design methodologies (via high-level synthesis and processor description languages) during the last decade has made the task of processor design simpler. For processor design, a range of processor description languages [95] are

suggested, several of which were made into prominent commercial [6, 96] and academic [97] tools. On the other hand, high-level synthesis became a major component of Electronic System Level (ESL) tools. A large number of high-level synthesis tools are currently offered by companies [98–101]. Academic offerings are also showing maturity and acceptance [102, 103]. A slightly outdated survey of high-level synthesis tools is presented in [104]. However, the complex design space of reconfigurable processors demands further research effort in this area. The current methodologies and tools for reconfigurable processor design are schematically presented in Figure 8. As in any high-level design methodology, the

FIGURE 8: Reconfigurable processor design methodology.

coverage of design space is compromised when speed of design exploration is desired.

The design flow for reconfigurable processor from an user perspective is presented earlier in Figure 4. There, the design space exploration is outlined with a feedback loop from the architecture. This feedback loop for repeated performance evaluation leading to a design closure is detailed in Figure 8. The design process ideally begins from a set of applications, which map onto the architecture with the help of software toolsuite. Based on the performance evaluation in terms of cycle-count, energy efficiency, or total area, the designer may undertake one or more of the following steps:

- (i) modify/optimize the application
- (ii) modify/optimize the tools
- (iii) modify/optimize the (soft) implementation.

When all of the above can be modified, the design space exploration is the most exhaustive. However, such an exhaustive exploration comes at the cost of slow exploration speed. Alternatively, the design space exploration can be done quickly by obtaining an approximate performance estimation, for example, via instruction-accurate processor simulation [105]. For general-purpose and domain-specific reconfigurable processors, limited customizability for the architecture is allowed. For several domain-specific and application-specific reconfigurable processors, facility for detailed architecture customization is provided. The customizability of architecture comes with (semi)automatic generation of software toolsuites as well as independent optimization plugins. Few prominent approaches for customization of architecture and tools are discussed in the following.

5.1. Architecture Customizability. Early reconfigurable processors relied on fixed architectures for the base processor and off-the-shelf fine-grained reconfigurable blocks. Implementation was done with RTL. This, naturally, prevented detailed design space exploration. Architecture customizability evolved with the advent of processor description languages [95]. An early reconfigurable processor to have a high-level processor description is XiRisc [60]. Stretch [65] used the architecture template from Tensilica [85] for limited design exploration capability. ADRES architecture [106] provided a VLIW model for the base processor with design exploration options for the CGRA. A complete language-driven reconfigurable processor design option is proposed in [16] by extending the ADL LISA [107].

To address high-level modeling platforms for CGRAs, several works have been done to fill this gap [108–110]. These languages or templates for CGRA are proposed with an associated toolflow that is CGRA synthesis for application mapping and CGRA simulation for performance evaluation. In a recent work, it is shown that fine-grained CGRAs could also be modeled at high level with an associated toolsuite generation capabilities [111]. Most notable of the CGRA modeling efforts is a commercial endeavor from Menta [22]. Menta Origami Designer allows the user to design an eFPGA in a GUI-based environment. Following the design entry, RTL implementation, software toolsuite, and a performance analyzer are automatically generated.

While optimizing a processor implementation from high-level specification is a well-researched topic [112–114], specific optimizations for reconfigurable processors are rarely proposed. In a recent work, reconfigurability is used for dynamically power gating the architecture to deliver increased energy efficiency [115]. In a specific CGRA implementation with

separate control for storage movement Zhang et al. [116] demonstrated increased data locality, thereby improving the overall energy and runtime performance.

5.2. Tools Customizability. Customizability of the tools is intricately linked with the modeling environment. Fixed architectures provided limited options for customizing the tools. Few works have been done to simultaneously explore the kernel partitioning, scheduling, and mapping of the application onto the reconfigurable block. In MOLEN architecture, the reconfiguration overhead is taken into account for instruction scheduling [117]. Exploration of custom instruction synthesis and the coupling between the base processor and the reconfigurable block is jointly done in [74]. For different configurations of the base processor and the reconfigurable block, the mapping and synthesis tools are automatically generated for all reconfigurable processor design frameworks, which allow high-level language-based design entry [16, 65, 106]. In [118], Ansaloni et al. proposed a simultaneous kernel partitioning and scheduling for CGRA.

The increased prominence of fine- and coarse-grained reconfigurable architectures is well received with the compiler community. In recent years, several important works have been done for efficient mapping of application on a reconfigurable architecture. There is a large body of work already present for efficient (and in some cases optimal) mapping of RTL/data flow graph to commercial fine-grained reconfigurable architectures [119–121]. However, unlike *fixed* fine-grained reconfigurable architectures, customizable architectures need to automatically generate the tools from a high-level specification. This problem is addressed in [16, 72]. In [72], based on the reconfigurable architecture specification, a variant of modulo scheduling is proposed to perform the mapping and clustering of the input application graph onto the CGRA. In [16] the CGRA specification is used to derive a mapping and clustering algorithm based on the Simultaneous Mapping And Clustering (SMAC) algorithm proposed for fine-grained FPGAs [120]. In both [16, 72] placement and routing are performed via heuristics (e.g., based on simulated annealing [122]). In a recent extension of [16] Chen et al. proposed a force-directed heuristic for placement and routing of the CGRA [111].

For standalone CGRA/eFPGA with limited design space exploration option, several advancements in the synthesis flow are recently made. By extending on the iterative modulo scheduling presented in [72], an edge-centric modulo scheduling is proposed in [123], where the routing overhead is also considered during scheduling operations on a CGRA. In [124], a dynamic operation fusion is proposed by implementing a local bypass network between CGRA operators. There, any available slack is recognized by the compiler, and back-to-back operations are executed in a single cycle. The mapping of an application onto a CGRA is recognized as a graph embedding problem over multiple cycles in [125]. Based on this understanding, a modulo graph embedding problem is constructed for the mapping, where loop bodies are mapped onto the target CGRA graph with resource constraints. Along the same line of thought, an input application graph is

converted to an epimorphic equivalent graph based on the architecture constraints in [126]. This reduces the search space for possible mapping strategies. Furthermore, in [126] local recomputation is proposed in order to address resource limitations. This is, in principle, similar to the duplication of subgraphs during delay-optimal mapping of application onto an FPGA. In [127], resource locality is considered during application mapping onto CGRA. An Integer Linear Programming (ILP) formulation for CGRA synthesis is proposed in [128]. It is interesting to note that the independent CGRA synthesis approaches considered a simple PE structure for the CGRA and considered a one-pass synthesis instead of traditional four passes of mapping, clustering, placement, and routing.

5.3. Performance Estimation and Verification. Performance estimation forms a core part of design space exploration. In existing high-level reconfigurable processor design frameworks [16], more importance is laid on runtime performance in terms of, for example, cycles. This is due to the fact that high-level estimation of area and power is still a subject of active research. For template-based reconfigurable processor designs, the design space is limited. However, more detailed performance analysis is available [65]. In CGRA mapping algorithms [123–127], runtime is used as the performance metric, though in some cases energy or energy-delay product is also used.

Needless to mention that, for the off-the-shelf reconfigurable processors [1, 3, 84, 87], the design space exploration is limited to the tuning of the application, and in those cases highly accurate performance figures can be obtained.

Functional verification of a reconfigurable processor is still an unexplored research area, though high-level processor verification [129–131] is relatively well studied. This is expected since options for detailed design exploration are absent in the state of the art for reconfigurable processors. The high-level modeling and synthesis flow for CGRA shows a highly regular structure, and therefore the generated RTL is assumed to be correct-by-construction. However, the same cannot be assumed for more complex CGRA and fine-grained eFPGAs [22, 25]. Furthermore, with growing concerns of process variation and unreliability in advanced CMOS technology nodes, there is a pressing need for reconfigurable processor verification research.

6. Future Challenges

The existing research already hints at the future challenges for reconfigurable processor design community. In the following, a systematic outline of the possible challenges and solution approach is presented.

Theoretical Understanding. Despite the modeling of DeHon [23], clear understanding of the reconfigurable processing design space is still limited. This is partly due to the complex structure of processors. A processor may support wide-issue, heterogeneous VLIW ISA and support multiple chained operators. How such an organization is better/worse from a

CGRA, which supports a rich interconnect network, is not clearly understood. The model [23] could also be enriched by more detailed physical estimations. Leveraging existing models [37, 39] for application in reconfigurable processor is another important open problem.

The theoretical understanding is important, for first, increased design efficiency and, second, to support cross-layer exploration of power, area, and novel constraints like reliability both in design time and execution time. For example, a reconfigurable processor may provide a compromised QoS for the end user while improving the thermal footprint. Without accurate modeling of the physical constraints at the architectural level, such adaptability is difficult. As an early hint of this direction, it is shown that a theoretical model of the CGRA [125, 126] can lead to more efficient tools.

Seamless Design Exploration Capability. As an offshoot of better high-level architecture modeling capacity, one should have better design exploration capability. It is conjectured that adaptability is going to be an important design metric for which understanding different design points is essential. As a part of detailed design exploration capability, the synthesis, compilation, and verification flow need to grow in a balanced manner. In particular, absence of accurate profiling tools leaves the designer to develop either the architecture in a kernel-specific bottom-up fashion [31] or develop a general-purpose reconfigurable processor. It is important that application profiling tools identify the opportunity of parallelism, which could be leveraged by a reconfigurable processor.

Programming Models. As the number of components on a SoC is increasing, a clean programming model is required more than ever. For the reconfigurable processor viewed as a single computing block, a top-level programmer's view is needed. This is addressed in different works via automatic kernel partitioning, user-directed pragmas, or proprietary language proposals. It is an important open question to identify the best programming model for reconfigurable processors or in general for fine-grained massively parallel processing blocks. It is interesting to determine which architectural details are better to hide from the application developers and which ones to expose.

Novel Late-CMOS and Post-CMOS Architectures. In the late-CMOS era, digital designs are facing novel constraints due to power, reliability, and demand for flexibility due to high manufacturing cost. It is important to investigate novel reconfigurable processor architectures, which can improve the thermal/energy efficiency without sacrificing flexibility. The fact that spatial/temporal redundancy is required for increased reliability is well established. However, the architectural benefits of reconfigurable processor from the perspective of reliability are still to be understood [132].

There are a range of elementary devices, which are being proposed as alternatives to the CMOS transistors. It is interesting to study the repercussion of these devices at the architecture level and if necessary position reconfigurable processors accordingly. For example, memristors, an emerging device, shows the unique capability of nonvolatile

storage and computing. This can be easily mimicked by a reconfigurable processor with its constituent basic elements serving alternately as storage or processing elements. This, in principle, takes the computing further away from von Neumann paradigm. This is demonstrated recently with some prototypes [133, 134].

7. Conclusion

In this paper, a survey of reconfigurable processors is presented. Included are the design options, design methodology, and concrete processor instances. Based on the rapid advancement in this interesting research area, several new updates are included compared to the prior surveys. In particular, it is noted that reconfigurable computing is grown to be a vast discipline and therefore demands separate attention to the sub disciplines like reconfigurable processors and the corresponding survey papers like this. Anticipating the future challenges, several research directions are proposed.

It is likely that our understanding of SoC architectures will evolve with time. To look back at this survey and taking cue from earlier research is on the agenda of our future work. At the end of this decade or even earlier, it is expected that another detailed survey, possibly dealing with sub topics presented in this paper, will be needed.

Acknowledgments

The author would like to thank the anonymous reviewers for providing critical feedback on the earlier version of this paper. The author would like also like to express his gratitude to Narendra Nath Datta for inspiring this study, among others.

References

- [1] Xilinx, <http://www.xilinx.com/>.
- [2] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," in *Proceedings of the Conference on Design, Automation and test in Europe (DATE '01)*, pp. 642–649, IEEE Press, Piscataway, NJ, USA, 2001.
- [3] Intel Atom, <http://ark.intel.com/products/family/29035>.
- [4] ASIP Meister, http://www.asip-solutions.com/en/asip_meister.html/.
- [5] P. Grun, A. Halambi, A. Khare, V. Ganesh, N. Dutt, and A. Nicolau, "EXPRESSION: an ADL for system level design exploration," Tech. Rep., Department of Information and Computer Science, University of California, Irvine, Calif, USA, 1998.
- [6] Synopsys Processor Designer, <http://www.synopsys.com/systems/blockdesign/processordev/pages/default.aspx>.
- [7] F. Barat, R. Lauwereins, and G. Deconinck, "Reconfigurable instruction set processors from a hardware/software perspective," *IEEE Transactions on Software Engineering*, vol. 28, no. 9, pp. 847–862, 2002.
- [8] P. Schaumont, I. Verbauwhede, K. Keutzer, and M. Sarrafzadeh, "A quick safari through the reconfiguration jungle," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 172–177, June 2001.
- [9] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, 2002.

- [10] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: architectures and design methods," *IEE Proceedings of Computers and Digital Techniques*, vol. 152, no. 2, pp. 193–207, 2005.
- [11] S. Vassiliadis and D. Soudris, *Fine- and Coarse-Grain Reconfigurable Computing*, Springer, New York, NY, USA, 2007.
- [12] C. Bobda, *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*, Springer, New York, NY, USA, 2007.
- [13] S. Hauck and A. Dehon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann, Boston, Mass, USA, 2007.
- [14] R. Sass and A. G. Schmidt, *Embedded Systems Design with Platform FPGAs: Principles and Practices*, Morgan Kaufmann, Boston, Mass, USA, 2010.
- [15] L. Bauer and J. Henkel, *Run-Time Adaptation for Reconfigurable Embedded Processors*, Springer, New York, NY, USA, 2011.
- [16] A. Chattopadhyay, R. Leupers, H. Meyr, and G. Ascheid, *Language-Driven Exploration and Implementation of Partially Re-Configurable ASIPs*, Springer, New York, NY, USA, 2008.
- [17] J. Henkel, "Closing the SoC design gap," *Computer*, vol. 36, no. 9, pp. 119–121, 2003.
- [18] K. Keutzer, S. Malik, and A. R. Newton, "From ASIC to ASIP: the next design discontinuity," in *Proceedings of the International Conference on Computer Design: VLSI in Computers and Processors (ICCD '02)*, pp. 84–90, Washington, DC, USA, September 2002.
- [19] Synopsys, <http://www.synopsys.com/>.
- [20] N. Goulding-Hotta, J. Sampson, G. Venkatesh et al., "The greendroid mobile application processor: an architecture for silicon's dark future," *IEEE Micro*, vol. 31, no. 2, pp. 86–95, 2011.
- [21] AMD Torrenza, http://www.amd.com/us/press-releases/Pages/Press_Release_120740.aspx.
- [22] S. A. S. Menta, <http://www.menta.fr/>.
- [23] A. DeHon, "Density advantage of configurable computing," *Computer*, vol. 33, no. 4, pp. 41–49, 2000.
- [24] W. S. Carter, K. Duong, R. H. Freeman et al., "A user programmable reconfigurable logic array," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 233–235, 1986.
- [25] T. Von Sydow, B. Neumann, H. Blume, and T. G. Noll, "Quantitative analysis of embedded FPGA-architectures for arithmetic," in *Proceedings of the IEEE 17th International Conference on Application-Specific Systems, Architectures and Processors (ASAP '06)*, pp. 125–131, Los Alamitos, Calif, USA, September 2006.
- [26] M. Hohenauer, C. Schumacher, R. Leupers, G. Ascheid, H. Meyr, and H. Van Someren, "Retargetable code optimization with SIMD instructions," in *Proceedings of the 4th International Conference on Hardware Software Codesign and System Synthesis (CODES+ISSS '06)*, pp. 148–153, Seoul, Republic of Korea, October 2006.
- [27] K. Atasu, G. Dündar, and C. Özturan, "An integer linear programming approach for identifying instruction-set extensions," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS '05)*, pp. 172–177, September 2005.
- [28] K. Atasu, R. G. Dimond, O. Mencer, W. Luk, C. Özturan, and G. Dündar, "Optimizing instruction-set extensible processors under data bandwidth constraints," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '07)*, pp. 588–593, Nice, France, April 2007.
- [29] K. Karuri, A. Chattopadhyay, M. Hohenauer, R. Leupers, G. Ascheid, and H. Meyr, "Increasing data-bandwidth to instruction-set extensions through register clustering," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '07)*, pp. 166–171, November 2007.
- [30] B. Mei, A. Lambrechts, D. Verkest, J.-Y. Mignolet, and R. Lauwereins, "Architecture exploration for a reconfigurable architecture template," *IEEE Design and Test of Computers*, vol. 22, no. 2, pp. 90–101, 2005.
- [31] X. Chen, A. Minwegen, Y. Hassan et al., "FLEXDET: flexible, efficient multi-mode MIMO detection using reconfigurable ASIP," in *Proceedings of the 20th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM '12)*, pp. 69–76, 2012.
- [32] L. Bauer, M. Shafique, S. Kramer, and J. Henkel, "RISPP: rotating instruction set processing platform," in *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07)*, pp. 791–796, San Diego, Calif, USA, June 2007.
- [33] Altera, <http://www.altera.com/>.
- [34] Xilinx CoolRunner-II, <http://www.xilinx.com/products/silicon-devices/cpld/coolrunner-ii/index.htm>.
- [35] "Non-volatility and infinite reconfigurability in PLDs," <http://www.latticesemi.com/>.
- [36] S. Fortune and J. Wyllie, "Parallelism in random access machines," in *Proceedings of the 10th Annual Symposium on Theory of Computing*, 1978.
- [37] L. G. Valiant, "Bridging model for parallel computation," *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, 1990.
- [38] T. L. Williams and R. J. Parsons, "The heterogeneous bulk synchronous parallel model," in *Proceedings of the 15th Workshops on Parallel and Distributed Processing (IPDPS '00)*, vol. 1800 of *Lecture Notes in Computer Science*, pp. 102–108, Springer, Cancun, Mexico, 2000.
- [39] D. E. Culler, R. M. Karp, D. Patterson et al., "LogP: a practical model of parallel computation," *Communications of the ACM*, vol. 39, no. 11, pp. 78–85, 1996.
- [40] D. C. Chen and J. M. Rabaey, "A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, pp. 1895–1904, 1992.
- [41] A. K. Yeung and J. M. Rabaey, "A 2.4 GOPS data-driven reconfigurable multiprocessor IC for DSP," in *Proceedings of the 42nd IEEE International Solid-State Circuits Conference (ISSCC '95)*, February 1995.
- [42] P. M. Athanas and H. F. Silverman, "Processor reconfiguration through instruction-set metamorphosis," *IEEE Computer*, vol. 26, no. 3, pp. 11–18, 1993.
- [43] M. J. Wirthlin, B. L. Hutchings, and K. L. Gilson, "The nano processor: a low resource reconfigurable processor," in *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines*, D. A. Buell and K. L. Pocek, Eds., pp. 23–30, IEEE Computer Society Press, Los Alamitos, Calif, USA, 1994.
- [44] C. Iseli and E. Sanchez, "Spyder: a SURE (SUPerscalar and REconfigurable) processor," *The Journal of Supercomputing*, vol. 9, no. 3, pp. 231–252, 1995.
- [45] M. Wirthlin and B. Hutchings, "A dynamic instruction set computer," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, P. Athanas and K. L. Pocek, Eds., pp. 99–107, IEEE Computer Society Press, Los Alamitos, Calif, USA, 1995.

- [46] R. W. Hartenstein and R. Kress, "Datapath synthesis system for the reconfigurable datapath architecture," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '95)*, pp. 479–484, Chiba, Japan, September 1995.
- [47] C. Ebeling, D. C. Cronquist, and P. Franklin, "RaPiD—reconfigurable pipelined datapath," in *Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers*, pp. 126–135, Springer, 1996.
- [48] E. Mirsky and A. DeHon, "MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 157–166, Los Alamitos, Calif, USA, April 1996.
- [49] J. R. Hauser and J. Wawrzynek, "Garp: a MIPS processor with a reconfigurable coprocessor," in *Proceedings of the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '97)*, pp. 12–21, April 1997.
- [50] T. J. Callahan, J. R. Hauser, and J. Wawrzynek, "Garp architecture and C compiler," *Computer*, vol. 33, no. 4, pp. 62–69, 2000.
- [51] C. R. Rupp, M. Landguth, T. Garverick et al., "The NAPA adaptive processing architecture," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, p. 28, IEEE Computer Society, 1998.
- [52] J. M. Arnold, "An architecture simulator for National Semiconductor's Adaptive Processing Architecture (NAPA)," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 271–272, 1998.
- [53] T. Miyamori and K. Olukotun, "REMARC (abstract): reconfigurable multimedia array coprocessor," in *Proceedings of the ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays*, p. 261, ACM Press, 1998.
- [54] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Transactions on Computers*, vol. 49, no. 5, pp. 465–481, 2000.
- [55] S. Majzoub, R. Saleh, and H. Diab, "Reconfigurable platform evaluation through application mapping and performance analysis," in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, pp. 496–501, 2006.
- [56] M. Wan, H. Zhang, V. George et al., "Design methodology of a low-energy reconfigurable single-chip DSP system," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 28, no. 1-2, pp. 47–61, 2001.
- [57] G. J. M. Smit, P. J. M. Havinga, L. T. Smit, P. M. Heysters, and M. A. J. Rosien, "Dynamic reconfiguration in mobile systems," in *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream (FPL '02)*, pp. 171–181, Springer, Montpellier, France, 2002.
- [58] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "Energy-efficiency of the MONTIUM reconfigurable tile processor," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '04)*, pp. 38–44, Las Vegas, Nev, USA, June 2004.
- [59] A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, and R. Guerrieri, "A VLIW processor with reconfigurable instruction set for embedded applications," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1876–1886, 2003.
- [60] C. Mucci, F. Campi, A. Deledda, A. Fazzi, M. Ferri, and M. Bocchi, "A cycle-accurate ISS for a dynamically reconfigurable processor architecture," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, April 2005.
- [61] S. Hauck, T. W. Fry, M. M. Hosler, and J. P. Kao, "The Chimaera Reconfigurable Functional Unit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 206–217, 2004.
- [62] Z. Alex Ye, A. Moshovos, S. Hauck, and P. Banerjee, "CHIMAERA: a high-performance architecture with a tightly-coupled reconfigurable functional unit," in *Proceedings of the 27th International Symposium on Computer Architecture (ISCA '00)*, pp. 225–235, Vancouver, Canada.
- [63] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. L. M. Bertels, G. Kuzmanov, and E. M. Panainte, "The MOLEN polymorphic processor," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1363–1375, 2004.
- [64] E. M. Panainte, K. Bertels, and S. Vassiliadis, "Compiler-driven FPGA-area allocation for reconfigurable computing," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, pp. 369–374, March 2006.
- [65] Stretch, <http://www.stretchinc.com/>.
- [66] D. Novo Bruna, B. Bougard, P. Raghavan et al., "Energy-performance exploration Of a CGA-based SDR processor," in *Proceedings of the Software Defined Radio Technical Conference and Product Exposition*, Leuven, Belgium, 2006.
- [67] C. Brunelli, F. Garzia, and J. Nurmi, "A coarse-grain reconfigurable architecture for multimedia applications featuring subword computation capabilities," *Journal of Real-Time Image Processing*, vol. 3, no. 1-2, pp. 21–32, 2008.
- [68] R. Köenig, L. Bauer, T. Stripf et al., "KAHRISMA: a novel hypermorphic reconfigurable-instruction-set multi-grained-array architecture," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '10)*, pp. 819–824, March 2010.
- [69] V. Govindaraju, H. Chen-Han, T. Nowatzki et al., "DySER: unifying functionality and parallelism specialization for energy-efficient computing," *IEEE Micro*, vol. 32, no. 5, pp. 38–51, 2012.
- [70] E. Waingold, M. Taylor, D. Srikrishna et al., "Computer baring it all to software: raw machines," *Computer*, vol. 30, no. 9, pp. 86–93, 1997.
- [71] Tiler, <http://www.tiler.com/>.
- [72] B. Mei, S. Vernalde, D. Verkest, H. Man, and R. Lauwereins, "DRESC: a retargetable compiler for coarse-grained reconfigurable architectures," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT '02)*, pp. 166–173, 2002.
- [73] Xilinx AutoESL, <http://www.xilinx.com/tools/autoesl.instructions.htm>.
- [74] K. Karuri, A. Chattopadhyay, X. Chen et al., "A design flow for architecture exploration and implementation of partially reconfigurable processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1281–1294, 2008.
- [75] REDEFINE, <http://www.morphingmachines.com/>.
- [76] nVIDIA, <http://www.nvidia.com/>.
- [77] A. DeHon, "Reconfigurable architectures for general-purpose computing," Tech. Rep., MIT, 1996.
- [78] P. Dubey and S. Engineer, "Teraflops for the masses: killer apps of tomorrow," in *Proceedings of the Workshop on Edge*

- [75] Computing Using New Commodity Architectures (UNC '06), vol. 23, 2006.
- [79] K. Asanovic, R. Bodik, B. Catanzaro et al., "The landscape of parallel computing research: a view from Berkeley," Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Calif, USA, 2006.
- [80] K. Shahzad, A. Khalid, Z. E. Răăkossy, G. Paul, and A. Chattopadhyay, "CoARX: a coprocessor for ARX-based cryptographic algorithms," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2013.
- [81] S. Saponara, M. Casula, and L. Fanucci, "ASIP-based reconfigurable architectures for power-efficient and real-time image/video processing," *Journal of Real-Time Image Processing*, vol. 3, no. 3, pp. 201–216, 2008.
- [82] M. Shafique and J. Henkel, *Hardware/Software Architectures for Low-Power Embedded Multimedia Systems*, Springer, New York, NY, USA, 2011.
- [83] G. Kappen and T. G. Noll, "Application specific instruction processor based implementation of a GNSS receiver on an FPGA," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, March 2006.
- [84] IBM Netezza, <http://www-01.ibm.com/software/data/db2/zos/analytics-accelerator/>.
- [85] Tensilica, <http://www.tensilica.com/>.
- [86] IBM, "Liquid Metal," http://researcher.watson.ibm.com/researcher/view_project.php?id=122.
- [87] Altera FPGA Series, <http://www.altera.com/devices/fpga/fpga-index.html>.
- [88] Altera, "NIOsII," <http://www.altera.com/devices/processor/nios2/ni2-index.html>.
- [89] Altera, "NIOsII performance," http://www.altera.com/literature/ds/ds_nios2-perf.pdf.
- [90] IMEC reconfigurable 4G baseband radio, <http://www.imec.be/ScientificReport/SR2011/1414035.html>.
- [91] Recore Systems, "Montium reconfigurable DSP IP core," <http://www.recoresystems.com/products/montium-reconfigurable-dsp-ip/>.
- [92] ARM-SoC FPGA SO-DIMM, <http://www.orsoc.se/?p=495>.
- [93] Y. Guo, H. Broersma, G. J. M. Smit, and P. M. Heysters, "A graph covering algorithm for a coarse grain reconfigurable system," in *Proceedings of the ACM SIGPLAN Conference on Languages, Compilers, and Tools for Embedded Systems*, pp. 199–208, June 2003.
- [94] A. DeHon, J. Adams, M. DeLorimier et al., "Design patterns for reconfigurable computing," in *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04)*, pp. 13–23, April 2004.
- [95] P. Mishra and N. Dutt, *Processor Description Languages*, Morgan Kaufmann, Boston, Mass, USA, 2008.
- [96] Target Compiler Technologies, <http://www.retarget.com/>.
- [97] UC Irvine/EXPRESSION, <http://www.ics.uci.edu/~express/index.htm>.
- [98] "BlueSpec Compiler," <http://www.bluespec.com/products.html>.
- [99] Cadence, "C-to-Silicon," http://www.cadence.com/products/sd/silicon_compiler/pages/default.aspx.
- [100] Forte Design Systems, <http://www.forteds.com/>.
- [101] Synopsys, "Symphony C Compiler," <http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/default.aspx>.
- [102] C-to-Verilog, <http://www.c-to-verilog.com/>.
- [103] Z. Wang, X. Wang, A. Chattopadhyay, and Z. E. Rakosi, "ASIC synthesis using architecture description language," in *Proceedings of the International Symposium on VLSI Design, Automation, and Test (VLSI-DAT '12)*, pp. 1–4, Hsinchu, Taiwan, 2012.
- [104] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuit*, Springer, New York, NY, USA.
- [105] A. Nohl, G. Braun, O. Schliebusch, R. Leupers, H. Meyr, and A. Hoffmann, "A universal technique for fast and flexible instruction-set architecture simulation," in *Proceedings of the 39th Annual Design Automation Conference (DAC '02)*, pp. 22–27, June 2002.
- [106] B. Mei, S. Vernalde, D. Verkest, and R. Lauwereins, "Design methodology for a tightly coupled VLIW/reconfigurable matrix architecture: a case study," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, pp. 1224–1229, February 2004.
- [107] A. Chattopadhyay, H. Meyr, and R. Leupers, "LISA: a uniform ADL for embedded processor modelling, implementation and software toolsuite generation," in *Processor Description Languages*, chapter 5, pp. 95–130, Morgan Kaufmann, Boston, Mass, USA, 2008.
- [108] J. O. Filho, S. Masekowsky, T. Schweizer, and W. Rosenstiel, "CGADL: an architecture description language for coarse-grained reconfigurable arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 9, pp. 1247–1259, 2009.
- [109] K. Patel, S. McGettrick, and C. J. Bleakley, "Rapid functional modelling and simulation of coarse grained reconfigurable array architectures," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 383–391, 2011.
- [110] A. Chattopadhyay, X. Chen, H. Ishebabi, R. Leupers, G. Ascheid, and H. Meyr, "High-level modelling and exploration of Coarse-grained Re-configurable Architectures," in *Proceedings of the Design, Automation and Test in Europe (DATE '08)*, pp. 1334–1339, March 2008.
- [111] X. Chen, S. Li, J. Scheliefer et al., "High-level modeling and synthesis for embedded FPGAs," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE '13)*, 2013.
- [112] E. M. Witte, A. Chattopadhyay, O. Schliebusch et al., "Applying resource sharing algorithms to ADL-driven automatic ASIP Implementation," in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '05)*, pp. 193–199, October 2005.
- [113] A. Chattopadhyay, B. Geukes, D. Kammler et al., "Automatic ADL-based operand isolation for embedded processors," in *Proceedings of the Design, Automation and Test in Europe (DATE '06)*, pp. 600–605, European Design and Automation Association, Leuven, Belgium, March 2006.
- [114] O. Schliebusch, A. Chattopadhyay, E. M. Witte et al., "Optimization techniques for ADL-driven RTL processor synthesis," in *Proceedings of the 16th International Workshop on Rapid System Prototyping (RSP '05)*, pp. 165–171, Washington, DC, USA, June 2005.
- [115] D. Kissler, D. Gran, Z. Salcić, F. Hannig, and J. Teich, "Scalable many-domain power gating in coarse-grained reconfigurable processor arrays," *IEEE Embedded Systems Letters*, vol. 3, no. 2, pp. 58–61, 2011.
- [116] C. Zhang, T. Lenart, H. Svensson, and V. Öwall, "Design of coarse-grained dynamically reconfigurable architecture for

- DSP applications,” in *Proceedings of the 2009 International Conference on ReConfigurable Computing and FPGAs, (ReConFig '09)*, pp. 338–343, December 2009.
- [117] E. M. Panainte, K. Bertels, and S. Vassiliadis, “Instruction scheduling for dynamic hardware configurations,” in *Proceedings of the Design, Automation and Test in Europe (DATE '05)*, pp. 100–105, March 2005.
- [118] G. Ansaloni, K. Tanimura, L. Pozzi, and N. Dutt, “Integrated kernel partitioning and scheduling for coarse-grained reconfigurable arrays,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 12, pp. 1803–1816, 2012.
- [119] V. Betz, J. Rose, and A. Marquardt, Eds., *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Norwell, Mass, USA, 1999.
- [120] J. Y. Lin, D. Chen, and J. Cong, “Optimal simultaneous mapping and clustering for FPGA delay optimization,” in *Proceedings of the 43rd ACM/IEEE Annual Conference on Design Automation (DAC '06)*, pp. 472–477, ACM Press, San Francisco, Calif, USA.
- [121] G. Chen and J. Cong, “Simultaneous placement with clustering and duplication,” in *Proceedings of the 41st Annual Conference on Design Automation (DAC '04)*, pp. 740–772, ACM Press, 2004.
- [122] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [123] H. Park, K. Fan, S. Mahlke, T. Oh, H. Kim, and H.-S. Kim, “Edge-centric modulo scheduling for coarse-grained reconfigurable architectures,” in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT '08)*, pp. 166–176, October 2008.
- [124] Y. Park, H. Park, and S. Mahlke, “CGRa express: accelerating execution using dynamic operation fusion,” in *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09)*, pp. 271–280, October 2009.
- [125] H. Park, K. Fan, M. Kudlur, and S. Mahlke, “Modulo graph embedding: mapping applications onto coarse-grained reconfigurable architectures,” in *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES '06)*, pp. 136–146, October 2006.
- [126] M. Hamzeh, A. Shrivastava, and S. Vrudhula, “EPIMap: using epimorphism to map applications on CGRAs,” in *Proceedings of the 49th Annual Design Automation Conference (DAC '12)*, pp. 1284–1291.
- [127] Y. Kim, J. Lee, A. Shrivastava, J. W. Yoon, D. Cho, and Y. Paek, “High throughput data mapping for coarse-grained reconfigurable architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1599–1609, 2011.
- [128] G. Lee, K. Choi, and N. D. Dutt, “Mapping multi-domain applications onto coarse-grained reconfigurable architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 637–650, 2011.
- [129] A. Chattopadhyay, A. Sinha, D. Zhang, R. Leupers, G. Ascheid, and H. Meyr, “Integrated verification approach during ADL-driven processor design,” in *Proceedings of the 17th IEEE International Workshop on Rapid System Prototyping (RSP '06)*, pp. 110–116, Chania, Crete, June 2006.
- [130] M. Puig-Medina, G. Ezer, and P. Konas, “Verification of configurable processor cores,” in *Proceedings of the 37th Design Automation Conference (DAC '00)*, pp. 426–431, June 2000.
- [131] P. Mishra and N. Dutt, “Functional coverage driven test generation for validation of pipelined processors,” in *Proceedings of the Design, Automation and Test in Europe (DATE '05)*, pp. 678–683, March 2005.
- [132] T. Schweizer, A. Kuster, S. Eisenhardt, T. Kuhn, and W. Rosenstiel, “Using run-time reconfiguration to implement fault-tolerant coarse grained reconfigurable architectures,” in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW '12)*, pp. 320–327, Shanghai, China, May 2012.
- [133] Q. Xia, W. Robinett, M. W. Cumbie et al., “Memristor-CMOS hybrid integrated circuits for reconfigurable logic,” *Nano Letters*, vol. 9, no. 10, pp. 3640–3645, 2009.
- [134] K.-H. Kim, S. Gaba, D. Wheeler et al., “A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications,” *Nano Letters*, vol. 12, no. 1, pp. 389–395, 2012.