The First 25 Years of the FPL Conference: Significant Papers

PHILIP H. W. LEONG, The University of Sydney
HIDEHARU AMANO, Keio University
JASON ANDERSON, University of Toronto
KOEN BERTELS, Delft University of Technology
JOÃO M. P. CARDOSO, Universidade do Porto
OLIVER DIESSEL, UNSW Australia
GUY GOGNIAT, University of South Brittany
MIKE HUTTON, Altera Corp
JUNKYU LEE, The University of Sydney
WAYNE LUK, Imperial College London
PATRICK LYSAGHT, Xilinx Research Labs
MARCO PLATZNER, University of Paderborn
VIKTOR K. PRASANNA, University of Southern California
TERO RISSA, Nokia Corporation
CRISTINA SILVANO, Politecnico di Milano
HAYDEN KWOK-HAY SO, University of Hong Kong
YU WANG, Tsinghua University

A summary of contributions made by significant papers from the first 25 years of the Field-Programmable Logic and Applications conference (FPL) is presented. The 27 papers chosen represent those which have most strongly influenced theory and practice in the field.

CCS Concepts: • **Computer systems organization** → **Reconfigurable computing**; • **Hardware** → **Reconfigurable logic and FPGAs;**

Additional Key Words and Phrases: Field-programmable Logic and Applications, Significant papers, FPL, 25 years

15

Authors' addresses: P. Leong and J. Lee, The University of Sydney, Australia; emails: {philip.leong, jun.kyu.lee}@sydney.edu.au; H. Amano, Keio University, Japan; email: hunga@am.ics.keio.ac.jp; J. Anderson, University of Toronto, Canada; email: janders@ece.utoronto.ca; K. Bertels, Delft University of Technology, Netherlands; email: K.L.M.Bertels@tudelft.nl; J. M. P. Cardoso, Universidade do Porto, Portugal; email: jmpc@acm.org; O. Diessel, University of NSW, Australia; email: odiessel@cse.unsw.edu.au; G. Gogniat, University of South Brittany, France; email: guy.gogniat@univ-ubs.fr; M. Hutton, Altera Corp, San Jose; email: mhutton@altera.com; W. Luk, Imperial College London, UK; email: wl@doc.ic.ac.uk; P. Lysaght, Xilinx Research Labs, San Jose; email: patrick.lysaght@xilinx.com; M. Platzner, University of Paderborn, Germany; email: platzner@upb.de; V. Prasanna, University of Southern California, USA; email: prasanna@usc.edu; T. Rissa, Nokia, Finland; email: tero.rissa@nokia.com; C. Silvano, Politecnico di Milano, Italy; email: cristina.silvano@polimi.it; H. K.-H. So, University of Hong Kong, Hong Kong; email: hso@eee.hku.hk; Y. Wang, Tsinghua University, China; email: yu-wang@tsinghua.edu.cn.

ACM 1936-7406/2017/03-ART15 $15.00
DOI: http://dx.doi.org/10.1145/2996468

ACM Reference Format:
Philip H. W. Leong, Hideharu Amano, Jason Anderson, Koen Bertels, João M. P. Cardoso, Oliver Diessel, Guy Gogniat, Mike Hutton, JunKyu Lee, Wayne Luk, Patrick Lysaght, Marco Platzner, Viktor K. Prasanna, Tero Rissa, Cristina Silvano, Hayden Kwok-Hay So, and Yu Wang. 2017. The first 25 years of the FPL conference: Significant papers. ACM Trans. Reconfigurable Technol. Syst. 10, 2, Article 15 (March 2017), 17 pages.
DOI: http://dx.doi.org/10.1145/2996468

1. INTRODUCTION

The first International Conference on Field-Programmable Logic and Applications (FPL) was held in 1991 at Oxford University. In the ensuing years, it has become the largest meeting on field-programmable gate array (FPGA) technologies and systems, and many important contributions have been published at the conference.

This article lists the most significant contributions from 1991 to 2014 [Leong et al. 2015]. The selection was made by an international Significant Papers Committee (SPC), composed of the authors of this article.

Only regular papers were considered, and Google Scholar was used for citation counts. Since an objective process was desired, those papers having more than nine citations per year, or more than 100 overall citations, were included on an initial shortlist. Concurrently, an open call to the community was made through the FPGA mailing list (fpga-list@mailman.sydney.edu.au), inviting nomination of papers with impact other than the number of citations. A total of 24 nominations were received and all papers that were not already on the initial short list (13 papers) were included in the final short list.

The resulting 60 short-listed papers were divided into five categories, each with a corresponding subcommittee. SPC members served on a subcommittee according to their expertise. Since some nominated papers were authored by SPC members, subcommittee chairs were chosen to ensure that no SPC member had conflicts of interest during the paper selection process. Following discussions within each subcommittee, recommendations were consolidated for the SPC. A final dialogue regarding the choices was then undertaken.

The total number of papers considered was 1,765 (this figure may be larger than the total number of full papers as it was not possible to distinguish them from posters in some cases). The process described previously resulted in 27 papers (1.5% of the total) being selected. Inevitably, the inclusion and exclusion of certain papers will be the subject of debate; however, the papers presented here represent the best efforts of the SPC to ensure a fair and objective selection process.

2. FPL SIGNIFICANT PAPER LIST (1991–2014)

The selected papers were grouped according to the topics represented by the following subsections.

2.1. Applications and Benchmarks

Software executing on instruction-driven processors such as multicores or graphics processing units (GPUs) faces limitations on computational speed and energy efficiency due to the need for extensive logic to fetch and decode instructions. Data-driven FPGA designs address these issues through higher degrees of customization, allowing more operations to be performed per clock cycle, enabling improved speed, energy efficiency and ability to respond to real-time events.

Eight application papers are introduced, each of which has influenced the development of other FPGA-based applications and benchmarks. FPGAs are well suited to signal processing applications in which multiplication often involves a constant multiplicand, and customization in this case can save hardware resources [Petersen and

Hutchings 1995], and real-time operation, particularly in video signal processing, has always been a challenge [Haynes et al. 1999]. DNA sequence alignment is another computationally intensive kernel, and customization to allow more computational elements to fit on a single FPGA is described [Yu et al. 2003]. Another application that takes customization to the extreme is to exploit process variations to produce physically uncloneable functions [Guajardo et al. 2007]. Arithmetic optimization utilising arithmetic of different precisions has been effectively used to solve global atmospheric equations [Gan et al. 2013]. The general problem of multitasking on FPGAs has been extensively researched, and introducing the restriction that only one task can operate at any time has important advantages [Simmler et al. 2000]. Two other benchmarking papers were included: a quantitative comparison between FPGAs, GPUs, and CPUs for image processing applications [Asano et al. 2009], and a comparison of the round 2 SHA–3 candidates that played an important role in the competition for the standard [Baldwin et al. 2010].

—*An Assessment of the Suitability of FPGA-Based Systems for Use in Digital Signal Processing* [Petersen and Hutchings 1995]. This early work compared serial, parallel, and distributed FPGA multiplier performance with that of digital signal processor (DSP) and application-specific integrated circuit approaches. Their performance in finite impulse response (FIR) filters and a radix-4 fast Fourier transform was studied.

The paper showed the potential of FPGAs in signal processing and laid the foundations for this key application domain. The authors showed how arithmetic units could be assembled on such devices, which were very rudimentary at that time. The key optimization of constant operand multiplication was applied, and the potential for accelerating DSP problems on an emerging hardware platform identified. Many existing applications apply ideas from this work [Meyer-Baese and Meyer-Baese 2007], and research in this domain is ongoing.

—*SONIC: A Plug-in Architecture for Video Processing* [Haynes et al. 1999]. This work proposed a reconfigurable computing architecture for video processing that was composed of multiple Plug-In Processing Elements (PIPEs) consisting of a Pipe Engine, Pipe Router, and Pipe Memory. An application programming interface to handle resource allocation and scheduling was also introduced. Performance was demonstrated with a 19-tap 2D FIR filter with eight PIPEs. This achieved more than 15 frames per second operating on 512×512-resolution video transferred over the PCI bus.

This paper pioneered a reconfigurable hardware architecture with software plug-ins for accelerating video image processing applications and utilized modularity to reduce application design time. It had significant impact on reconfigurable media processing applications, hardware/software codesign research, and embedded video image processing research [Galuzzi and Bertels 2011]. It provides the foundation for the UltraSonic system [Haynes et al. 2002], which was commercialized by Sony Broadcast.

—*Multitasking on FPGA Coprocessors* [Simmler et al. 2000]. This paper details the requirements and overheads associated with preemptive multitasking for FPGA coprocessors. The proposed technique involved a single hardware task being executed at any time, which had three advantages: (1) while parallel execution of several tasks on the FPGA is possible, executing a single task at a time allows the full communication data rate with the host processor; (2) multitasking allows switching of the FPGA at any time, and (3) I/O resources do not need to be shared. A task manager was also described, which was used to manage loading of circuits and states for different hardware tasks.

This paper is significant in that it pioneers preemptive multitasking with an FPGA coprocessor and also inspired other multitasking research [Steiger et al. 2004a].

—*A Smith-Waterman Systolic Cell* [Yu et al. 2003]. This paper presents an improved systolic processing element (PE) for implementing the Smith-Waterman DNA sequence alignment algorithm on an FPGA. The authors proposed a scheme in which two PEs were merged into a compact cell, a technique commonly used in VLSI layout. Only by considering two PEs together could adjacent memory elements be utilized, which serves to achieve a 25% area saving. A large linear systolic array of elements was connected together to achieve a high degree of parallelism, and it was interfaced to an FPGA board via the SDRAM bus to achieve the highest reported density and system performance at the time.

Sequence alignment is the computational bottleneck in many bioinformatics applications, including sequence database searching, multiple sequence alignment, genome assembly, and short read mapping. Thus, acceleration of bioinformatics applications using FPGAs has become an important application domain and the Smith-Waterman algorithm continues to be used. Compared with previous designs, this work removed the requirement of runtime reconfiguration while maintaining equivalent functionality. The paper represents a reference design widely used in that domain [Gokhale and Graham 2006] and inspired other work in FPGA-based pattern matching [Van Court and Herbordt 2007] and biosequence analysis [Oliver et al. 2005].

—*Physical Unclonable Functions, FPGAs, and Public-Key Crypto for IP Protection* [Guajardo et al. 2007]. This paper presented several new protocols to address the IP protection problem on FPGAs, supporting hardware authentication, hardware platform authentication, and complete design confidentiality. The authors' approach relies on public-key cryptography, with secret key information being generated by a physical unclonable function (PUF) implemented in the FPGA. Since a unique PUF output is internally generated in the device, security is enhanced by removing the requirement to transfer the key on or off the FPGA. The paper also details how the PUF can be implemented using the random initialization state of on-FPGA static RAM cells.

This paper provides efficient protocols for IP protection and introduces an original solution for PUFs based on SRAM memories with good properties. It has become one of the key references in this area [Maes and Verbauwhede 2010].

—*Performance Comparison of FPGA, GPU, and CPU in Image Processing* [Asano et al. 2009]. This paper gives a quantitative performance comparison between FPGA, GPU, and CPU implementations of image processing applications. Special characteristics for each platform were highlighted; for example, an FPGA can implement an optimized circuit according to an application, a GPU can utilize many cores, and a CPU can utilize its SIMD features. This paper found that in two-dimensional filters, GPUs are the best choice, since shared variables are not required. In stereo-vision and K-means clustering applications, an FPGA excels because it avoids local memory access conflicts. Finally, a CPU is usually a good choice for applications in which cache miss rates are low.

This research provided informative descriptions of how performance is affected by application characteristics. It has significance from a tutorial perspective and impacted other performance comparison research [Pauwels et al. 2012].

—*FPGA Implementations of the Round Two SHA-3 Candidates* [Baldwin et al. 2010]. This paper provides a summary of the algorithms associated with the round 2 candidates for the secure hash algorithm 3 (SHA-3), along with a detailed study of their

suitability for an efficient hardware implementation. Hardware implementations of all of the algorithms were compared for different message sizes. Moreover, this was the only work to include padding as part of the hardware. At the time of publication, the US National Institute of Standards and Technology (NIST) was holding a competition to determine a successor to the SHA-2 algorithm.

The research work outlined in this paper was considered as part of the SHA-3 algorithm selection process by NIST [Turan et al. 2011]. The Keccak candidate was identified in the paper as having the highest overall performance and best performance per area. Keccak was subsequently chosen for the SHA-3 hashing standard in August 2015 [NIST 2015].

—*Accelerating Solvers for Global Atmospheric Equations Through Mixed-Precision Data Flow Engine* [Gan et al. 2013]. This paper showed that reconfigurable supercomputing can provide faster and more energy-efficient solutions, compared to conventional supercomputers for global climate modeling. The paper also provides guidance on how to apply mixed-precision arithmetic to complex problems that do not easily fit onto limited hardware resources. The experimental results reported two orders of magnitude speedup over a multicore CPU, and the design was one order of magnitude faster and had one order of magnitude lower power than a hybrid CPU-GPU platform.

It has been cited by a number of promising research efforts that follow a similar direction, including Düben and Palmer [2014] and Russell et al. [2015].

2.2. Architecture

The development of reconfigurable architectures to efficiently perform computing tasks has been studied since the introduction of FPGAs. Common issues included striking a balance between fine- and coarse-grained computing resources, time-multiplexing to efficiently utilize resources, interfacing with processors, and reducing power and energy consumption.

Five architecture papers are introduced in this subsection. The RaPiD architecture combined static and cycle-by-cycle dynamic configuration [Ebeling et al. 1996]. SCORE introduced the abstraction of streams of computation, mapped to a programmable fabric via time-multiplexing [Caspi et al. 2000]. The ADRES approach combined a very long instruction word processor with coarse-grained arithmetic blocks [Mei et al. 2003]. One of the advantages of FPGAs is in power and energy consumption. Power modeling for FPGAs has been a major concern and an early paper largely solved this problem in a general manner for the widely used VPR tool [Poon et al. 2002]. Finally, using multiple VDD supplies allows power to be reduced without sacrificing performance [Gayasen et al. 2004].

—*RaPiD: Reconfigurable Pipelined Datapath* [Ebeling et al. 1996]. This early paper proposed a reconfigurable, deeply pipelined datapath tailored to application-specific requirements. Pipelined datapaths consisting of a mixture of arithmetic logic units (ALUs), multipliers, general-purpose registers, and local memories are constructed by a combination of static and dynamic configurations. While the static configuration determines the overall pipeline structure through a set of programmable routing switches, the dynamic configuration determines the cycle-by-cycle operations of the pipeline similar to the operation of a microcontroller.

This paper was highly influential, having inspired numerous subsequent works in pipelined and coarse-grained reconfigurable architectures [Baumgarte et al. 2003]. Its pipelined reconfiguration has influenced commercial products such as Tabula, and many ideas originally developed for the RaPiD design have also found applications in application-specific standard products and other tightly coupled accelerator

architecture designs. As this work dates back to the 1990s, it was truly visionary and fundamental to the evolution of field-programmable architectures.

—*Stream Computations Organized for Reconfigurable Execution (SCORE)* [Caspi et al. 2000]. This paper introduces a new computational model for reconfigurable computing, based on streams of computation virtualized onto a programmable fabric via time-multiplexing. The model facilitates a language abstraction and mapping that allows arbitrary computations to be implemented onto a target device without recompilation, overcoming the key challenge of vendor and device generation portability, and the constraint of fixed resources on the physical hardware. The streaming/paging abstraction introduced in the paper offers solutions for important practical problems such as dealing with exceptions via flow control. SCORE continued to grow and develop beyond this extended abstract, and a summary can be found in DeHon et al. [2006].

The formalized model in SCORE influenced subsequent work on overlays, hardware virtualization, and heterogeneous computing. Moreover, virtualization has become one of the largest growth areas for FPGAs, which are now moving into data centers and cloud computing as compute accelerators for processors. This paper provided mechanisms to address many of the key problems: bitstream portability, targeting heterogeneous resources, scaling computational resources to match availability on the physical fabric, and tolerating uncertain latency on links between operators. It not only addressed interesting problems of the day but also foreshadowed those of the future. The language and model proved valuable for other resource- and performance-scaleable implementations, including custom VLIW mappings [Kapre and DeHon 2011]. The streaming model plays a prominent role in some of the more usable commercial software offerings for reconfigurable computers including Ambric's Structural Object Programming Model [Butts et al. 2007] and IBM's Lime [Auerbach et al. 2010].

—*A Flexible Power Model for FPGAs* [Poon et al. 2002]. This was the first work to incorporate a power model into the widely used open-source VPR FPGA architecture and CAD framework. While today's FPGAs incorporate power-aware features and FPGAs in low-power/mobile scenarios are increasing in importance, this power-modeling work was done at a time when power was not a significant axis along which FPGAs were optimized. The proposed power model is flexible in the sense that, instead of modeling the power of a specific FPGA, the model is adaptive to the architecture specified to VPR. Dynamic and static power can be estimated for a range of LUT sizes, cluster sizes, and interconnection structures.

The "Poon power model" has been used and improved [Li et al. 2005] by researchers in a variety of power-aware CAD and architecture research endeavors. The power model, its associated source code, and documentation were released into the public domain, and for several years it was the primary model used to study power consequences of architecture and CAD algorithm changes. This work thus represents a significant service to the FPGA community.

—*ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix* [Mei et al. 2003]. This paper introduces a novel coarse-grained reconfigurable array (CGRA), which is a style of FPGA that uses large ALU-like logic blocks and multibit routing. While the CGRA concept was already known at the time of publication, the ADRES (Architecture for Dynamically Reconfigurable Embedded System) architecture incorporates features not seen in prior work. The CGRA fabric in ADRES is closely coupled with a Very Long Instruction Word (VLIW) processor, and in fact is able to access the same register file as the processor, obviating the need

for costly transfers to/from the CGRA. VLIW processors frequently use a concept called loop pipelining to exploit loop-level parallelism by overlapping adjacent loop iterations. With the attached CGRA fabric, ADRES offers the added ability to exploit spatial parallelism, providing higher computational throughput than the VLIW processor alone. The functional units in the VLIW processor are also part of the CGRA fabric, permitting resources to be shared between the two.

The ADRES architecture has had a far-reaching impact in the commercial space, which took many years to materialize. Samsung Electronics began a long-term collaboration with the ADRES researchers to develop a commercial CGRA, which today is known as the Samsung Reconfigurable Processor (SRP). SRP publications began to appear in the 2010s; it has been used in a variety of commercial Samsung products, including smartphones. ADRES can be viewed as an ancestor of the SRP [Kim et al. 2012].

—*A Dual-VDD Low Power FPGA Architecture* [Gayasen et al. 2004]. This work proposed a CAD-driven approach to configure the power domains of different portions of the FPGA. Both logic and interconnect were optimized [Anderson and Najm 2006] by utilizing slack from routed nets to assign voltage configurations. Similar approaches have been adopted commercially for power reduction using configurable back bias interacting with CAD tools in Altera Programmable Power Technology [Altera 2007] and by reconfigurable circuit fabrics from Xilinx [Tuan et al. 2012]. This work also helped inspire several academic efforts that considered interactions between configurable circuit fabrics and CAD flow optimizations for power reduction.

2.3. Design Methods and Tools

FPGA users often need to master complex design flows and evaluate a large design space to meet requirements. Advances in design methods and tools are thus of paramount importance for both researchers and practitioners, which continues to be a crucial research area.

The four papers selected point to the breadth of research presented at the conference. They focus on placement and routing tools and algorithms [Betz and Rose 1997], generation of stream-based application-specific architectures from object-oriented descriptions [Mencer et al. 2000], an approach proposing power/energy reductions via the insertion of pipelining stages [Wilton et al. 2004], and FPGA support and a methodology for making easier the development of designs taking advantage of dynamic reconfiguration features [Lysaght et al. 2006].

—*VPR: A New Packing, Placement, and Routing Tool for FPGA Research* [Betz and Rose 1997]. This paper presented the Versatile Place and Route (VPR) tool, which performs placement and routing. A key feature is that it targeted a parameterized island-style FPGA defined through a configuration file, allowing it to be adapted to a broad range of different architectures. Its simulated annealing-based place and route algorithm had a smaller routing area than all previous work and the paper introduced a new set of benchmark circuits.

VPR became an important piece of infrastructure for FPGA architecture and CAD tool research and has been widely used and extended by other research groups. This tool was further developed by the authors at Right Track CAD Corp., which was acquired by Altera in 2000, and VPR then became the basis for both Altera's placement and routing and architecture evaluation tools.

This paper made a seminal contribution with the idea of a succinct and human-readable FPGA architecture description that drives a flexible CAD flow in order to explore new FPGA architectures, and has been used in a very wide range of FPGA

architecture studies. The VPR architecture description language and CAD flow have been extended and enhanced in many ways, most recently with version 7.0 of the Verilog-to-Routing CAD flow [Luu et al. 2014]. VTR 7.0 can investigate the architecture not only of programmable logic and routing, as this original VPR could, but also arithmetic structures, RAM and DSP blocks, and other special purpose structures. An enhanced version of VPR also forms the basis of Altera's FPGA Modeling Toolkit (FMT), which has been used to architect all of Altera's FPGAs from 2000 to the present day [Lewis et al. 2013]. VPR has also been used as a framework for the investigation of enhanced placement and routing algorithms, both in academia and in Altera's commercial Quartus II CAD suite, which uses an enhanced version of VPR as its placement and routing engine [Ludwin and Betz 2011].

—*StReAm: Object-Oriented Programming of Stream Architectures Using PAM-Blox* [Mencer et al. 2000]. This paper presents StReAm, a domain-specific compiler focused on compiling streaming-oriented applications to FPGA-based architectures. The approach was based on the PAM-Blox object-oriented module generation environment and employed a pipelined dataflow graph mapped directly to hardware. The advantage of this approach is that distributed FIFO buffers are able to exploit temporal data locality, while streaming data through pipelines exploits spatial data locality. StReAm uses operator overloading and template functions in C++ to create dataflow graphs, which are consecutively scheduled to obtain a stream architecture.

They key contribution of this paper lies in an efficient compilation of a streaming-oriented programming model to FPGAs, which is evident in contemporary products by Maxeler, a company founded by the authors. Maxeler has become the major provider of FPGA-based solutions in finance- and oil-related applications [Lindtjorn et al. 2011].

—*The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays* [Wilton et al. 2004]. This paper was the first to quantitatively study the impact of pipelining stages on power and energy consumption. It was found that pipelining could reduce energy per operation by between 40% and 90% on an Altera high-end FPGA and a Xilinx low-cost FPGA. In addition, the authors showed that power-aware clustering enables further reductions, and interaction between pipelining and clustering was also studied to understand how the degree of pipelining affects the effectiveness of the lower-level CAD algorithms in reducing energy.

The quantitative results presented for two representative FPGA devices include real measurements and estimations provided by vendor-specific tools. The results presented had many implications and highlighted the importance of considering pipelining stages when datapath design and/or generation are addressed in contemporary FPGAs. This work foresaw the importance of pipelining for reducing energy, a fact that is employed in the recently announced Altera Stratix 10 [Hutton 2015].

—*Enhanced Architectures, Design Methodologies, and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs* [Lysaght et al. 2006]. This paper presents Xilinx FPGA enhancements for providing better support for the creation of dynamically reconfigurable designs. Architecturally, it advocated prerouted bus macros for implementing the communication ports to interface static and dynamic regions of a design, described new features of the Virtex 4 architecture to support finer reconfiguration granularity, and introduced a higher bandwidth reconfiguration port. It also presented a design flow that supported partially reconfigurable designs, enabled by the two key enhancements: allowing arbitrary rectangular regions to be reconfigured and permitting signals in the static design to cross through partially reconfigurable regions without the use of a bus macro.

This paper was the first from a vendor to provide a complete dynamically reconfigurable FPGA and design flow. All major vendors now support similar flows [Kohn 2015; Altera 2010].

2.4. Dynamic Reconfiguration

Dynamic reconfiguration is the modification, at runtime, of a section of an FPGA's configuration memory with the aim of changing the implemented functionality. Reasons for doing so include altered functional requirements, reducing energy consumption, reducing device size, correcting faults, and enhancing performance. As evidenced by the number of citations and publication dates, FPL is one of the most important venues for discussing the opportunities and challenges presented by dynamically reconfigurable FPGA-based systems. Six papers on dynamic reconfiguration were selected.

The six selected papers cover the early exploration of dynamically reconfigurable systems [Lysaght and Dunlop 1994], first efforts to model infrastructure requirements in a general manner [Brebner 1996], hardware virtualization on a coarse-grained reconfigurable array and corresponding co-simulation techniques [Enzler et al. 2003], techniques to save and restore context to facilitate task pre-emption and relocation [Kalte and Porrmann 2005], methods and tools for generating dynamically reconfigurable systems [Koch et al. 2008], and applications to simultaneously error scrub and reconfigure the functionality of a design [Heiner et al. 2009].

—*Dynamic Reconfiguration of FPGAs* [Lysaght and Dunlop 1994]. This paper comprehensively summarized the state of the art and the ideas underpinning dynamically reconfigurable systems at a very early stage in their history. The concepts that were explained set the scene for all subsequent investigation. The paper describes a large number of ideas that have later (in some cases much later) been the topic of significant papers themselves. It is a highly educational paper rather than being purely technical, yet it still includes a system example and discusses implementation issues. The limitations of the technology are also discussed.

This influential paper presented the basis for dynamic reconfiguration of FPGAs [Kohn 2015; Altera 2010]. It includes various concepts that are in use today, including a nomenclature for reconfiguration, FPGA-centric self-adapting systems [Sander et al. 2008], and the concept of logic caching. Many researchers have worked on solving or reducing the limitations identified in this paper, particularly optimizing FPGAs for speed of reconfiguration, state readback and update, automatic partitioning based on temporal specifications, and tools for modeling new FPGA architectures.

—*A Virtual Hardware Operating System for the Xilinx XC6200* [Brebner 1996]. This paper introduced the concept of sea of accelerators and parallel harness virtual hardware models to support time-multiplexed hardware in which designs larger than physical FPGA resources could be executed. Swappable logic units (SLUs) were introduced, which are variable-sized processing elements that could be implemented on the FPGA resources and swapped out when needed. At the time of this paper, the Xilinx XC6200 was a new device that supported fine-grained partial reconfiguration. The sea-of-accelerators model was demonstrated along with an operating system that managed the configuration and utilization of SLUs.

Concepts and challenges in developing an operating system for a reconfigurable computer were introduced, and in the ensuing years, many further developments were inspired by working to address the challenges identified in this paper [Steiger et al. 2004b].

—*Virtualizing Hardware with Multicontext Reconfigurable Arrays* [Enzler et al. 2003]. The main contribution of this paper is an approach to overcome hardware resource limitations via hardware virtualization, enabling forward compatibility for reconfigurable fabrics. To achieve these goals, the paper proposes a coarse-grained, multicontext reconfigurable array that is attached as a coprocessor to a CPU. The coarse-grained array is tailored to processing data streams with a macro-pipelined data path. If sufficient resources are available, pipeline stages are mapped spatially. Otherwise, time-multiplexing on a single-array instance is used to provide virtualization, allowing one to execute macro-pipelines of arbitrary length on restricted hardware resources. The second contribution of this work was that it was one of the first works that proposed cycle-accurate cosimulation of the complete reconfigurable computing system to account for configuration, data transfer, and execution times. This detailed level of simulation allows the designer to accurately assess the performance of a single instance of the architecture and to perform a system-level evaluation for comparing design alternatives.

The impact and commercial relevance of this work is demonstrated by the considerable number of times it has been cited as prior art in patents (e.g., Rohe et al. [2007]).

—*Context Saving and Restoring for Multitasking in Reconfigurable Systems* [Kalte and Porrmann 2005]. This paper described a method for reducing dynamic task deallocation/relocation time by only saving the context of the task that is to be (re)moved. The idea was demonstrated using Xilinx Virtex FPGAs, which supports column-wise reconfiguration. All modules were full height with varying width, reducing the placement problem to one dimension. Communication infrastructure is horizontal, with static modules in their own reserved area. Context relocation involved a configuration manager, state extraction filter, state inclusion filter, and REPLICA filter for relocation, with a database being used to store hardware task information. A detailed quantitative study of relocation time is made in the paper.

The idea and technique presented in the paper are at the core of any runtime system for dynamically reconfigurable computing. The solution was practical and formed the basis for future work in multitasking hardware operating systems [Lübbers and Platzner 2009].

—*ReCoBus-Builder: A Novel Tool and Technique to Build Statically and Dynamically Reconfigurable Systems for FPGAs* [Koch et al. 2008]. This paper presents an easily usable tool chain for implementing dynamically reconfigurable systems on FPGAs. ReCoBus-Builder provides sophisticated communication architectures allowing for direct bus connection as well as point-to-point connections. With this tool, over a hundred partially reconfigurable modules can be placed into a shared reconfigurable region. Here, modules can be relocated and instantiated multiple times while still being able to carry out dedicated communication with each module. This provides outstanding flexibility and is essential for reducing fragmentation in reconfigurable systems. The ReCoBus-Builder approach is further applicable for a component-based design methodology where modules from a library can be plugged together by a process called *bitstream linking,* which can fully eliminate long synthesis runs or place and route steps.

ReCoBus-Builder was used for implementing several very sophisticated reconfigurable systems. For instance, in Oetken et al. [2010], a system is presented where various two-dimensionally relocatable reconfigurable video processing modules directly access external DDR memory using fast 64-bit DMA transfer concurrently with a complex PowerPC SoC on a Virtex-II Pro FPGA. The development of ReCoBus-Builder continued with the GoAhead tool [Beckhoff et al. 2012], which supports

recent Xilinx FPGAs and provides a powerful scripting interface. The next release of GoAhead will provide a TCL interface to the Vivado suite from Xilinx.

—*FPGA Partial Reconfiguration via Configuration Scrubbing* [Heiner et al. 2009]. Many critical FPGA-based systems rely on configuration memory scrubbing to repair radiation-induced errors, and a variety of configuration scrubbers have been proposed. One disadvantage of performing continuous configuration scrubbing is it occupies the configuration port, preventing the ability to perform other configuration functions such as partial reconfiguration. This paper proposed a simple method for performing partial reconfiguration in the course of a typical scrub cycle, thereby eliminating the need to interrupt scrubbing while the configuration port is used to perform a partial reconfiguration. To achieve this, the paper introduced a self-scrubber, utilizing a small portion of the FPGA, which performs the necessary operations to reconfigure a portion of the design while continuously scrubbing the entire FPGA. While the proposed technique is simple in nature, the paper's significance arises from the interesting and clearly presented concepts, the convincing experimental evaluation, and the fact that the ideas have high practical value. Radiation-induced errors are gaining increasing attention due to their increased presence in modern FPGAs.

A number of interesting projects have been developed based on some of the ideas presented in this paper. A generic lightweight configuration controller was developed using ideas in this paper to provide local fault detection and isolation [Straka et al. 2013]. A novel configuration controller was created to facilitate FPGA bootstrapping for a multi-FPGA configuration process [Ostler et al. 2011]. A self-adaptive FPGA system was developed to modify the implementation of a function in runtime using both partial reconfiguration and configuration scrubbing to address changes in a radiation environment [Glein et al. 2014].

2.5. Security and Network on Chip

Networking has always been an important application domain since FPGAs are the only programmable technology that can keep up with network line rates, and they can combine on-chip networking hardware with custom pattern matching. Furthermore, the success of off-chip networking has raised the question of whether FPGAs should contain soft or hard networks on chip (NoCs). Advantages are that this can reduce required FPGA routing resources, introduce greater routing flexibility, and facilitate dynamic reconfiguration.

Four significant papers were selected around this topic. Two early papers identified the advantages of network intrusion detection on FPGAs and described designs that achieved performance far exceeding processor-based implementations [Gokhale et al. 2002; Sourdis and Pnevmatikatos 2003]. Significant papers concerning NoCs include one that advocates using them to support reconfigurable operating systems [Marescaux et al. 2003] and another quantifying the advantages of hardware NoCs for performance and energy efficiency [Abdelfattah and Betz 2013].

—*Granidt: Towards Gigabit Rate Network Intrusion Detection Technology* [Gokhale et al. 2002]. This paper described an FPGA-accelerated network intrusion detection system (NIDS). A compiler that translated filter rules from the public domain Snort database into content-addressable memories (CAMs) was implemented on FPGA hardware, and a software representation of the rules was used as an input to the software. The hardware appends the result of the match to the packet, and the rule software performs the action associated with the matching rules. The approach allowed changes to the rules to be implemented by creating new CAM entries, avoiding the need for reconfiguration. Using this approach, it was shown that the design could

process data at a line rate of 2Gbps, this being 24.9× the speed of the Snort software implementation.

This paper was seminal in FPGA-based network filtering technology and has been cited by other influential papers such as Kumar et al. [2006] and Clark and Schimmel [2004b]. It heralded commercial FPGA-based network intrusion detection systems from companies including TippingPoint Technologies, Sensory Networks, and Emulex Endace. Deep packet inspection has since evolved into deep content inspection, pioneered by Wedge Networks, which can do object-level analysis of network traffic. As network speeds continue to increase and the importance of the Internet continues to rise, network filtering will increase in importance and is an ideal application of FPGA technology.

—*Fast, Large-Scale String Match for a 10Gbps FPGA-Based Network Intrusion Detection System* [Sourdis and Pnevmatikatos 2003]. This paper addressed the problem of pattern matching for intrusion detection, which compares a set of strings against a (high-speed) network traffic. It proposed a micro-architecture and a number of circuit-level optimizations exploiting parallelism between string matchers, pipelining of comparison units and large-fanout signals, parallel pipelined merge of individual results, and so forth to achieve a very high clock rate. The implementation on a Virtex2 FPGA exceeded 10Gbps processing throughput, a 3 to 5× improvement over previous work. Even ASIC approaches for NIDS pattern matching in the following years remained at this performance level [Tan and Sherwood 2005].

This paper paved the way for subsequent FPGA-based influential research that further exploited parallelism to improve pattern matching (area) efficiency while maintaining the 10Gbps performance goal such as Clark and Schimmel [2004a] and Sourdis and Pnevmatikatos [2004]. Work in this area was later extended to cover regular expressions [Sourdis et al. 2008; Brodie et al. 2006].

—*Networks on Chip as Hardware Components of an OS for Reconfigurable Systems* [Marescaux et al. 2003]. This paper is a significant refinement of a paper presented at FPL the previous year, which suggested that interconnection networks could enable fine-grained dynamic multitasking on FPGAs with low hardware overhead. The new paper proposed multiple networks on chip as hardware support for reconfigurable operating systems (OSs), with reconfiguration, OS, and application data communications being implemented as separate networks, each optimized appropriately; for example, the reconfiguration network is optimized for latency and the application network optimized for bandwidth. This allowed the elegant support of features such as dynamic task relocation with state transfer, hardware debugging, and security.

Subsequent operating systems supporting dynamic reconfiguration were inspired by ideas from this paper, notable examples including HERMES [Moraes et al. 2004], automotive runtime systems [Ullmann et al. 2004], and Dynoc [Bobda et al. 2005].

—*The Power of Communication: Energy-Efficient NoCs for FPGAs* [Abdelfattah and Betz 2013]. This quantitative study of hard networks on chip (NoCs) in FPGAs described two novel implementations: mixed NoCs, which used hard routers and soft links, and hard NoCs, which utilized hard routers and hard links. When compared with soft NoCs implemented entirely using the FPGA fabric, the mixed/hard NoCs were 20×/23× smaller and 5×/6× faster. Moreover, a 64-node hard NoC adds less than 1% to the total area of a large FPGA. The authors found that hard NoCs consume 4.5 to 10.4mJ of energy per GB of data transferred, which is comparable to the energy efficiency of soft point-to-point links requiring 4.7mJ/GB.

The unexpected result that hard NoCs can be as energy efficient as the simplest traditional FPGA point-to-point soft interconnect, with minimal area overhead and higher flexibility, makes a compelling case for including them in future devices. While the impact is difficult to judge at this early stage, this work has the potential to enhance the system-level interconnection of future FPGA applications.

Embedded NoCs have since been used to prototype Ethernet switching [Bitar et al. 2014] and packet processing [Bitar et al. 2015] applications on FPGAs, and the addition of embedded NoCs improved both the performance and efficiency of these applications significantly. Follow-on work in this area [Abdelfattah et al. 2015] focused on making embedded NoCs easier to use on FPGAs by defining design rules and creating CAD tools that enable hardware-friendly abstractions for this new kind of FPGA interconnect.

3. CONCLUSION

This article highlighted a selection of FPL publications that advanced the field over the last 25 years. We hope that it will help to inspire the next generation of researchers and industrialists over the next 25 years and beyond.

ACKNOWLEDGMENTS

Authors Leong and Lee acknowledge partial support from the Australian Research Council's Linkage Projects funding scheme (project number LP130101034) Zomojo Pty Ltd, and the Faculty of Engineering and Information Technologies, the University of Sydney, by the Faculty Research Cluster Program.

REFERENCES

M. S. Abdelfattah and V. Betz. 2013. The power of communication: Energy-efficient NoCs for FPGAs. In *2013 23rd International Conference on Field Programmable Logic and Applications (FPL'13)*. IEEE, 1–8. DOI: http://dx.doi.org/10.1109/FPL.2013.6645496

M. S. Abdelfattah, A. Bitar, and V. Betz. 2015. Take the highway: Design for embedded NoCs on FPGAs. In *Proceedings of the 2015 ACM / SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'15)*. ACM, New York, NY, 98–107. DOI: http://dx.doi.org/10.1145/2684746.2689074

Altera. 2007. *Stratix III Programmable Power*. Technical Report WP-01006-1.1. San Jose, CA. https://www.altera.com/en_US/pdfs/literature/wp/wp-01006.pdf.

Altera. 2010. *Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs*. Technical Report WP-01137-1.0. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01137-stxv-dynamic-partial-reconfig.pdf.

J. H. Anderson and F. N. Najm. 2006. Active leakage power optimization for FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 25, 3 (March 2006), 423–437. DOI: http://dx.doi.org/10.1109/TCAD.2005.853692

S. Asano, T. Maruyama, and Y. Yamaguchi. 2009. Performance comparison of FPGA, GPU and CPU in image processing. In *International Conference on Field Programmable Logic and Applications, 2009 (FPL'09).* IEEE, 126–131. DOI: http://dx.doi.org/10.1109/FPL.2009.5272532

J. Auerbach, D. F. Bacon, P. Cheng, and R. Rabbah. 2010. Lime: A Java-compatible and synthesizable language for heterogeneous architectures. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'10)*. ACM, New York, NY, 89–108. DOI: http://dx.doi.org/10.1145/1869459.1869469

B. Baldwin, A. Byrne, L. Lu, M. Hamilton, N. Hanley, M. ONeill, and W. P. Marnane. 2010. FPGA implementations of the round two SHA-3 candidates. In *2010 International Conference on Field Programmable Logic and Applications (FPL'10)*. IEEE, 400–407. DOI: http://dx.doi.org/10.1109/FPL.2010.84

V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt. 2003. PACT XPPA self-reconfigurable data processing architecture. *Journal of Supercomputing* 26, 2 (2003), 167–184.

C. Beckhoff, D. Koch, and J. Torresen. 2012. Go ahead: A partial reconfiguration framework. In *Proceedings of the 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'12)*. 37–44.

V. Betz and J. Rose. 1997. VPR: A new packing, placement and routing tool for FPGA research. In *Field-Programmable Logic and Applications*, W. Luk, P. Y. K. Cheung, and M. Glesner (Eds.). Lecture Notes in Computer Science, Vol. 1304. Springer, Berlin, 213–222.

A. Bitar, M. Abdelfattah, and V. Betz. 2015. Bringing programmability to the data plane: Packet processing with a NoC-enhanced FPGA. In *2015 International Conference on Field-Programmable Technology (FPT'15)*.

A. Bitar, J. Cassidy, N. E. Jerger, and V. Betz. 2014. Efficient and programmable ethernet switching with a NoC-enhanced FPGA. In *Proceedings of the 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'14)*. ACM, New York, NY, 89–100. DOI:http://dx.doi.org/10.1145/2658260.2658272

C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete, and J. Veen. 2005. Dynoc: A dynamic infrastructure for communication in dynamically reconfugurable devices. In *Proceedings of the International Field Programmable Logic and Applications Conference*. 153–158.

G. Brebner. 1996. A virtual hardware operating system for the Xilinx XC6200. In *Field-Programmable Logic Smart Applications, New Paradigms and Compilers*, R.W. Hartenstein and M. Glesner (Eds.). Lecture Notes in Computer Science, Vol. 1142. Springer, Berlin, 327–336.

B.C. Brodie, R. K. Cytron, and D. E. Taylor. 2006. A scalable architecture for high-throughput regular-expression pattern matching. In *33rd International Symposium on Computer Architecture, 2006 (ISCA'06)*. 191–202. DOI:http://dx.doi.org/10.1109/ISCA.2006.7

M. Butts, A. M. Jones, and P. Wasson. 2007. A structural object programming model, architecture, chip and tools for reconfigurable computing. In *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines*. 55–64.

E. Caspi, M. Chu, R. Huang, J. Yeh, J. Wawrzynek, and A. DeHon. 2000. Stream computations organized for reconfigurable execution (SCORE). In *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, Reiner Hartenstein and Herbert Grünbacher (Eds.). Lecture Notes in Computer Science, Vol. 1896. Springer, Berlin, 605–614.

C. R. Clark and D. E. Schimmel. 2004a. Scalable pattern matching for high speed networks. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2004 (FCCM'04)*. 249–257. DOI:http://dx.doi.org/10.1109/FCCM.2004.50

C. R. Clark and D. E. Schimmel. 2004b. Scalable pattern matching for high speed networks. In *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*. IEEE Computer Society, 249–257. http://dl.acm.org/citation.cfm?id=1025123.1025834.

A. DeHon, Y. Markovsky, E. Caspi, M. Chu, R. Huang, S. Perissakis, L. Pozzi, J. Yeh, and J. Wawrzynek. 2006. Stream computations organized for reconfigurable execution. *Journal of Microprocessors and Microsystems* 30, 6 (September 2006), 334–354.

P. D. Düben and T. N. Palmer. 2014. Benchmark tests for numerical weather forecasts on inexact hardware. *Monthly Weather Review* 142, 10 (2014), 3809–3829.

C. Ebeling, D. C. Cronquist, and P. Franklin. 1996. RaPiD reconfigurable pipelined datapath. In *Field-Programmable Logic Smart Applications, New Paradigms and Compilers*, Reiner Hartenstein and Manfred Glesner (Eds.). Lecture Notes in Computer Science, Vol. 1142. Springer, Berlin, 126–135.

R. Enzler, C. Plessl, and M. Platzner. 2003. Virtualizing hardware with multi-context reconfigurable arrays. In *Field Programmable Logic and Applications*, Peter Y. K. Cheung and George A. Constantinides (Eds.). Lecture Notes in Computer Science, Vol. 2778. Springer, Berlin, 151–160.

C. Galuzzi and K. Bertels. 2011. The instruction-set extension problem: A survey. *ACM Transactions on Reconfigurable Technology Systems* 4, 2, Article 18 (May 2011), 28 pages. DOI:http://dx.doi.org/10.1145/1968502.1968509

L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang. 2013. Accelerating solvers for global atmospheric equations through mixed-precision data flow engine. In *2013 23rd International Conference on Field Programmable Logic and Applications (FPL'13)*. IEEE, 1–6. DOI:http://dx.doi.org/10.1109/FPL.2013.6645508

A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan. 2004. A dual-VDD low power FPGA architecture. In *Field Programmable Logic and Applications*, Jürgen Becker, Marco Platzner, and Serge Vernalde (Eds.). Lecture Notes in Computer Science, Vol. 3203. Springer, Berlin, 145–157.

R. Glein, B. Schmidt, F. Rittner, J. Teich, and D. Ziener. 2014. A self-adaptive SEU mitigation system for FPGAs with an internal block RAM radiation particle sensor. In *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'14)*. 251–258. DOI:http://dx.doi.org/10.1109/FCCM.2014.79

M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. Hogsett. 2002. Granidt: Towards gigabit rate network intrusion detection technology. In *Field-Programmable Logic and Applications: Reconfigurable*

Computing Is Going Mainstream, Manfred Glesner, Peter Zipf, and Michel Renovell (Eds.). Lecture Notes in Computer Science, Vol. 2438. Springer, Berlin, 404–413.

M. B. Gokhale and P. S. Graham. 2006. *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*. Springer Science & Business Media.

J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. 2007. Physical unclonable functions and public-key crypto for FPGA IP protection. In *International Conference on Field Programmable Logic and Applications, 2007 (FPL'07)*. IEEE, 189–195. DOI: http://dx.doi.org/10.1109/FPL.2007.4380646

S. D. Haynes, P. Y. K. Cheung, W. Luk, and J. Stone. 1999. SONIC – A plug-in architecture for video processing. In *Field Programmable Logic and Applications*, Patrick Lysaght, James Irvine, and Reiner Hartenstein (Eds.). Lecture Notes in Computer Science, Vol. 1673. Springer, Berlin, 21–30.

S. D. Haynes, H. G. Epsom, R. J. Cooper, and P. L. McAlpine. 2002. UltraSONIC: A reconfigurable architecture for video image processing. In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*. Springer, 482–491.

J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb. 2009. FPGA partial reconfiguration via configuration scrubbing. In *International Conference on Field Programmable Logic and Applications, 2009 (FPL'09)*. IEEE, 99–104. DOI: http://dx.doi.org/10.1109/FPL.2009.5272543

M. Hutton. 2015. Architectural paths to faster and more robust FPGAs (keynote). In *2015 International Conference on Field-Programmable Technology (FPT'15)*.

H. Kalte and M. Porrmann. 2005. Context saving and restoring for multitasking in reconfigurable systems. In *International Conference on Field Programmable Logic and Applications, 2005.* IEEE, 223–228. DOI: http://dx.doi.org/10.1109/FPL.2005.1515726

N. Kapre and A. DeHon. 2011. VLIW-SCORE: Beyond C for sequential control of SPICE FPGA acceleration. In *2011 International Conference on Field-Programmable Technology (FPT'11)*. 1–9. DOI: http://dx.doi.org/10.1109/FPT.2011.6132678

C. Kim, M. Chung, Y. Cho, M. Konijnenburg, S. Ryu, and J. Kim. 2012. ULP-SRP: Ultra low power samsung reconfigurable processor for biomedical applications. In *2012 International Conference on Field-Programmable Technology (FPT'12)*. 329–334. DOI: http://dx.doi.org/10.1109/FPT.2012.6412157

D. Koch, C. Beckhoff, and J. Teich. 2008. ReCoBus-Builder: A novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In *International Conference on Field Programmable Logic and Applications, 2008 (FPL'08).* IEEE, 119–124. DOI: http://dx.doi.org/10.1109/FPL.2008.4629918

C. Kohn. 2015. *Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite*. Technical Report XAPP1231. San Jose, CA. http://www.xilinx.com/support/documentation/application_notes/xapp1231-partial-reconfig-hw-accelerator-vivado.pdf.

S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner. 2006. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*. ACM, New York, NY, 339–350. DOI: http://dx.doi.org/10.1145/1159913.1159952

P. H. W. Leong, H. Amano, J. Anderson, K. Bertels, J. M. P. Cardoso, O. Diessel, G. Gogniat, M. Hutton, JunKyu Lee, W. Luk, P. Lysaght, M. Platzner, V. K. Prasanna, T. Rissa, C. Silvano, H. So, and Yu Wang. 2015. Significant papers from the first 25 years of the FPL conference. In *Proc. International Conference on Field Programmable Logic and Applications (FPL)*. 1–3. DOI: 10.1109/FPL.2015.7293747

D. Lewis, D. Cashman, M. Chan, J. Chromczak, G. Lai, A. Lee, T. Vanderhoek, and H. Yu. 2013. Architectural enhancements in stratix V^{TM}. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'13)*. ACM, New York, NY, 147–156. DOI: http://dx.doi.org/10.1145/2435264.2435292

F. Li, Y. Lin, L. He, D. Chen, and J. Cong. 2005. Power modeling and characteristics of field programmable gate arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 11 (2005), 1712–1724.

O. Lindtjorn, R. Clapp, O. Pell, Haohuan Fu, M. Flynn, and Haohuan Fu. 2011. Beyond traditional microprocessors for geoscience high-performance computing applications. *IEEE Micro* 31, 2 (March 2011), 41–49. DOI: http://dx.doi.org/10.1109/MM.2011.17

E. Lübbers and M. Platzner. 2009. ReconOS: Multithreaded programming for reconfigurable computers. *ACM Transactions on Embedded Computing Systems (TECS)* 9, 1 (2009), 8.

A. Ludwin and V. Betz. 2011. Efficient and deterministic parallel placement for FPGAs. *ACM Transactions on Design Automation of Electronic Systems* 16, 3, Article 22 (June 2011), 23 pages. DOI: http://dx.doi.org/10.1145/1970353.1970355

J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz. 2014. VTR 7.0: Next generation architecture and CAD

system for FPGAs. *ACM Transactions on Reconfigurable Technology and Systems* 7, 2, Article 6 (July 2014), 30 pages. DOI:http://dx.doi.org/10.1145/2617593

P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford. 2006. Invited paper: Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs. In *International Conference on Field Programmable Logic and Applications, 2006 (FPL'06)*. IEEE, 1–6. DOI:http://dx.doi.org/10.1109/FPL.2006.311188

P. Lysaght and J. Dunlop. 1994. Dynamic reconfiguration of FPGAs. In *Selected Papers from the Oxford 1993 International Workshop on Field Programmable Logic and Applications on More FPGAs*. Abingdon EE&CS Books, Oxford, UK, 82–94.

R. Maes and I. Verbauwhede. 2010. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*. Springer, 3–37.

T. Marescaux, J.-Y. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde, and R. Lauwereins. 2003. Networks on chip as hardware components of an OS for reconfigurable systems. In *Field Programmable Logic and Applications*, Peter Y. K. Cheung and George A. Constantinides (Eds.). Lecture Notes in Computer Science, Vol. 2778. Springer, Berlin, 595–605.

B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins. 2003. ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix. In *Field Programmable Logic and Applications*, Peter Y. K. Cheung and George A. Constantinides (Eds.). Lecture Notes in Computer Science, Vol. 2778. Springer, Berlin, 61–70.

O. Mencer, H. Hübert, M. Morf, and M. J. Flynn. 2000. StReAm: Object-oriented programming of stream architectures using PAM-Blox. In *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, Reiner Hartenstein and Herbert Grünbacher (Eds.). Lecture Notes in Computer Science, Vol. 1896. Springer, Berlin, 595–604.

U. Meyer-Baese and U. Meyer-Baese. 2007. *Digital Signal Processing with Field Programmable Gate Arrays*. Vol. 65. Springer.

F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost. 2004. HERMES: An infrastructure for low area overhead packet-switching networks on chip. *Integrated VLSI Journal* 38, 1 (October 2004), 69–93. DOI:http://dx.doi.org/10.1016/j.vlsi.2004.03.003

NIST. 2015. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standards Publication FIPS Pub 202. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

A. Oetken, S. Wildermann, J. Teich, and D. Koch. 2010. A bus-based SoC architecture for flexible module placement on reconfigurable FPGAs. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'10)*. 234–239.

T. F. Oliver, B. Schmidt, and D. L. Maskell. 2005. Reconfigurable architectures for bio-sequence database scanning on FPGAs. *IEEE Transactions on Circuits and Systems II: Express Briefs,* 52, 12 (2005), 851–855.

P. S. Ostler, M. J. Wirthlin, and J. E. Jensen. 2011. FPGA bootstrapping on PCIe using partial reconfiguration. In *2011 International Conference on Reconfigurable Computing and FPGAs (ReConFig'11)*. 380–385. DOI:http://dx.doi.org/10.1109/ReConFig.2011.40

K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros, and M. M. Van Hulle. 2012. A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features. *IEEE Transactions on Computers,* 61, 7 (July 2012), 999–1012. DOI:http://dx.doi.org/10.1109/TC.2011.120

R. J. Petersen and B. L. Hutchings. 1995. An assessment of the suitability of FPGA-based systems for use in digital signal processing. In *Field-Programmable Logic and Applications*, Will Moore and Wayne Luk (Eds.). Lecture Notes in Computer Science, Vol. 975. Springer, Berlin, 293–302.

K. K. W. Poon, A. Yan, and S. J. E. Wilton. 2002. A flexible power model for FPGAs. In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, Manfred Glesner, Peter Zipf, and Michel Renovell (Eds.). Lecture Notes in Computer Science, Vol. 2438. Springer, Berlin, 312–321.

A. Rohe, S. Teig, H. Schmit, J. Redgrave, and A. Caldwell. 2007. Operational time extension. (June 26 2007). US Patent 7,236,009.

F. P. Russell, P. D. Düben, X. Niu, W. Luk, and T. N. Palmer. 2015. Architectures and precision analysis for modelling atmospheric variables with chaotic behaviour. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'15)*. 171–178. DOI:http://dx.doi.org/10.1109/FCCM.2015.52

O. Sander, L. Braun, M. Hübner, and J. Becker. 2008. Data reallocation by exploiting FPGA configuration mechanisms. In *Reconfigurable Computing: Architectures, Tools and Applications*. Springer, 312–317.

H. Simmler, L. Levinson, and R. Männer. 2000. Multitasking on FPGA coprocessors. In *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, Reiner Hartenstein and Herbert Grünbacher (Eds.). Lecture Notes in Computer Science, Vol. 1896. Springer, Berlin, 121–130.

I. Sourdis, J. Bispo, J. M. P. Cardoso, and S. Vassiliadis. 2008. Regular expression matching in reconfigurable hardware. *Journal of Signal Processing Systems* 51, 1 (2008), 99–121. DOI:http://dx.doi.org/10.1007/s11265-007-0131-0

I. Sourdis and D. Pnevmatikatos. 2003. Fast, large-scale string match for a 10Gbps FPGA-based network intrusion detection system. In *Field Programmable Logic and Applications*, Peter Y. K. Cheung and George A. Constantinides (Eds.). Lecture Notes in Computer Science, Vol. 2778. Springer, Berlin, 880–889.

I. Sourdis and D. Pnevmatikatos. 2004. Pre-decoded CAMs for efficient and high-speed NIDS pattern matching. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2004 (FCCM'04)*. 258–267. DOI:http://dx.doi.org/10.1109/FCCM.2004.46

C. Steiger, H. Walder, and M. Platzner. 2004a. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions on Computers,* 53, 11 (November 2004), 1393–1407. DOI:http://dx.doi.org/10.1109/TC.2004.99

C. Steiger, H. Walder, and M. Platzner. 2004b. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. *IEEE Transactions on Computers,* 53, 11 (2004), 1393–1407.

M. Straka, J. Kastil, Z. Kotasek, and L. Miculka. 2013. Fault tolerant system design and SEU injection based testing. *Microprocessors and Microsystems* 37, 2 (2013), 155–173. DOI:http://dx.doi.org/10.1016/j.micpro.2012.09.006 Digital System Safety and Security.

L. Tan and T. Sherwood. 2005. A high throughput string matching architecture for intrusion detection and prevention. In *Proceedings of the 32nd International Symposium on Computer Architecture, 2005 (ISCA'05)*. 112–122. DOI:http://dx.doi.org/10.1109/ISCA.2005.5

T. Tuan, R. L. Cline, and A. Rahman. 2012. Programmable integrated circuit with voltage domains. (2012). https://www.google.com/patents/US8159263 US Patent 8159263.

M. S. Turan, R. A. Perlner, L. E. Bassham, W. E. Burr, D. H. Chang, S. H. Chang, M. J. Dworkin, J. M. Kelsey, S. Paul, and R. C. Peralta. February 2011. *Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition*. Technical Report. NIST Interagency Report 7764.

M. Ullmann, M. Huebner, B. Grimm, and J. Becker. 2004. An FPGA run-time system for dynamical on-demand reconfiguration. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.* 135. DOI:http://dx.doi.org/10.1109/IPDPS.2004.1303106

T. Van Court and M. C. Herbordt. 2007. Families of FPGA-based accelerators for approximate string matching. *Microprocessors and Microsystems* 31, 2 (2007), 135–145.

S. J. E. Wilton, S.-S. Ang, and W. Luk. 2004. The impact of pipelining on energy per operation in field-programmable gate arrays. In *Field Programmable Logic and Applications*, Jürgen Becker, Marco Platzner, and Serge Vernalde (Eds.). Lecture Notes in Computer Science, Vol. 3203. Springer, Berlin, 719–728.

C. W. Yu, K. H. Kwong, K. H. Lee, and P. H. W. Leong. 2003. A smith-waterman systolic cell. In *Field Programmable Logic and Applications*, Peter Y. K. Cheung and George A. Constantinides (Eds.). Lecture Notes in Computer Science, Vol. 2778. Springer, Berlin, 375–384.

Received January 2016; accepted September 2016