Taxonomy of Reconfigurable Computing and Operating System

Rupashri Barik
Department of Information Technology
JIS College of Engineering, Kalyani, W. B., India

Amitabha Sinha
Dr. B. C. Roy Engineering College, Durgapur, W. B., India

Abstract- Reconfigurable computing is a promising technology to meet future computational demand by leveraging flexibilities and the high degree of parallelism found in reconfigurable hardware fabrics, such as field programmable gate arrays (FPGAs) [1][2]. Reconfigurable computing had played an important role in the course of evolution of the computing paradigms. The journey of FPGAs had carried out from classical Von-Neumann Architecture to Reconfigurable Architecture. The concept of dynamic reconfiguration of the FPGA had been utilized properly with the flexibility of software programming and it has enabled the developers to map the computation intensive functions onto FPGAs. In this paper this journey of Hardware and Software has been described. Programmable hardware with programmable processors is combined in configurable computing systems to capitalize the strengths of hardware and software. This survey paper aims to introduce the research areas of last few years, to present a classification of reconfigurable architectures, and a brief illustration of reconfigurable operating systems [3] [4].

Keywords- Reconfigurable computing, ASIC, FPGA, operating System

I. INTRODUCTION

In early 1940-50s, there was the concept of discrete devices like relays and transistors, etc. In 1950-60s the concept of discrete logic gates were introduced. Later in 1960-70s the integrated circuits were invented. And in 1970s the concept of gate arrays were introduced. Reconfigurable computing is rapidly establishing itself as a major discipline that covers various subjects of learning, including both computing science and electronic engineering. Reconfigurable devices, such as field programmable gate arrays (FPGAs), are used for computing purposes in reconfigurable computing.

II. DESCRIPTION

1. Reconfigurable Computing:

In Application Specific Integrated Circuits (ASIC), the optimized dedicated circuitry is to perform a specific task. As a result it has the advantage of low power dissipation. It gives the highest possible performance at lowest cost but for a limited task. Microprocessors have a limited set of arithmetic and control instructions that can be organized and sequenced to implement any arbitrary computation as it's behaves like a software. But it is inefficient in exploiting parallelism. Reconfigurable computing balances flexibility and efficiency which are associated with ASICs and microprocessors. It overcomes the speed limitation of microprocessor to solve the applications that is elimination of software approach to solve a given problem without losing the flexibility of the software. Reconfigurable computing is needed for configuring highly tuned hardware circuits to provide the functionality needed for a specific task.

Reconfigurable computing is defined as the study of computation using reconfigurable devices. Configuration respectively reconfiguration is the process of changing the structure of a reconfigurable device at start-up-time respectively at run-time. Efficiency and flexibility of hardware can be increased by incorporating programmability to it. A programmable hardware makes a balance between flexibility and efficiency as potential concurrency in the algorithm can be exploited by directly mapping the algorithm onto architecture.

1.1: FPGA Architecture:

Field Programmable Gate Array (FPGA) is one of the most readily available commercial programmable logic devices. Starting as ASIC replacements similar to other PLDs, FPGAs have slowly evolved into complex embedded system platforms that are flexible and are able to deliver performances comparable to ASICs. Introduced

in 1985 by the company Xilinx, an FPGA is a programmable device consisting, like the CPLDs, of three main parts. A set of programmable logic cells also called logic blocks or configurable logic blocks, a programmable interconnection network and a set of input and output cells around the device. A function to be implemented in FPGA is partitioned in modules, each of which can be implemented in a logic block. The logic blocks are then connected together using the programmable interconnection. All three basic components of an FPGA (logic block, interconnection and input output) can be programmed by the user in the field. FPGAs can be programmed once or several times depending on the technology used.

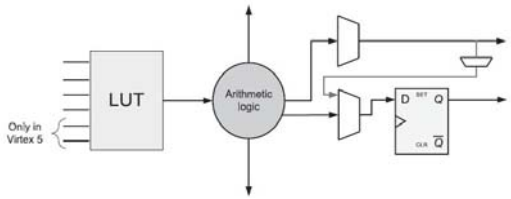

Fig1: Basic block of the Xilinx FPGAs

FPGAs consist of three major resources- Configurable Logic blocks (CLB), Routing blocks (Programmable Interconnect), I/O blocks. Other resources in FPGA are Memory, Multiplexers, Global clock buffers, Boundary scan logic.

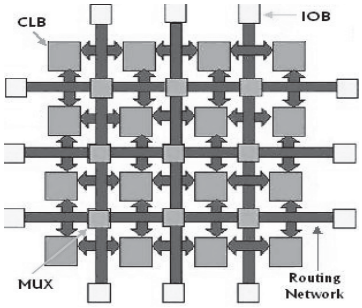

Fig2: FPGA Architecture (Xilinx VirtexII)

a) Configurable Logic Block (CLB):

The basic building blocks of an FPGA are configurable logic blocks (CLBs) that are programmable according to user logic functions. Each configurable logic unit consists of a fine grain programmable combinational logic block that is optionally connected to one or more flip-flops.

The programmable combinational logic block is usually implemented using lookup tables. A number of multiplexers (MUXs) are used to fine tune the behaviors of a CLU. They are used to select output from either the LUT or flip-flop, or both. They also control polarities of clock and other control signals. By carefully connecting the flip-flops and LUTs of different configurable logic units using the on- chip programmable routing network, any synchronous digital logic design can be implemented.

The local routing provides feedback between slices in the same CLB and it provides routing to neighboring slices. A switch matrix in a CLB provides access to general routing resources.

Look Up Table:

A LUT (Lookup table) is a one bit wide memory array. Basically, it is a 4-input AND gate is replaced by a LUT that has four address inputs and one single bit output with 16 one bit locations.

The combinational logics are stored in LUTs. These are also called Function Generators. The capacity of LUT is dependent on the number of inputs only. A reconfigurable hardware device should provide the users with the possibility to dynamically implement and re-implement new functions. This is usually done by means of function generators that can be seen as the basic computing unit in the device.

A combinational logic function can be implemented using LUT. Here is an example of implementing a logic function using LUT.

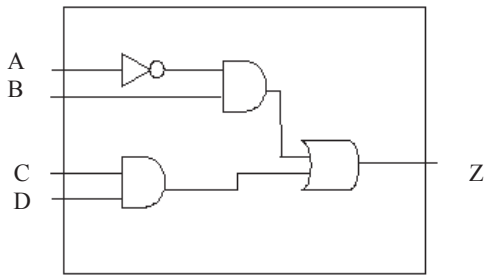

Fig3: A Boolean Logic Circuit

Table 1 represents the LUT implemented for the above logic circuit.

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Table1: LUT for given logic circuit

b) Routing Network:

The routing network of an FPGA links different configurable logic units together to perform user defined logic. Many different routing networks have been proposed to balance performance with routability of an FPGA. FPGA routing networks are generally regular, consisting of groups of short local wires combined with long global wires. Wire to wire connections are made through programmable switch boxes.

c) I/O Blocks:

I/O Blocks provide the interface between the FPGA and the outside world rather it will be better to say I/O Blocks provide the interface between package pins and the CLBs.

1.2 Early Works on Reconfigurable Computing

Attempts to have a flexible hardware structure that can be dynamically modified at run-time to compute a desired function are almost as older as the development of other computing paradigms. To overcome the non flexibility of the first computer, the ENIAC (electronically numerical integrator and computer) [5] that could be programmed only by handwriting an algorithm, Jon Von Neumann proposed a first universal architecture made upon three main blocks (memory, data path and control path), able to run any given and well-coded program.

1.2.1 The Estrin Fix-plus machine

In 1959, Gerald Estrin, a computer scientist of the University of California at Los Angeles, introduced the concept of reconfigurable computing. In Estrin publication in 1960 [6] the fix-plus machine, defines the concept of reconfigurable computing paradigm.

1.2.2 The Rammig Machine

The concept for editing hardware [7] was proposed by Franz J. Rammig, a researcher at the University of Dortmund In the year 1977. The aim of this was the "investigation of a system, which, with no manual or mechanical interference, permits the building, changing, processing and destruction of real (not simulated) digital hardware".

1.2.3 Hartenstein's XPuter

In early 1980s, Reiner Hartenstein, a researcher at the University of Kaiserslautern in Germany presented the concept of Xputer's [8] [9]. The objective was to have a very high degree of programmable parallelism in the hardware, at lowest possible level, to obtain performance not possible with the Von Neumann computers.

1.2.4 The PAM Machine

Introduced by Bertin et al. [10] from the Digital Equipment corporation, a programmable active memory (PAM) is defined as a uniform array of identical cells, the processing array blocks (PAB) that are connected in a given regular fashion. The architecture is built on the FPGA model. The flip flop provides a memory for the realization of finite state machines but can be used for local data storage.

1.2.5 The SPLASH II

SPLASH II [11], [12] is a computing system made upon a set of FPGA modules scolded on different printed circuit boards, connected together to build a massive parallel computer.

1.2.6 The PRISM Paradigm

PRISM is an acronym for processor reconfiguration through instruction set metamorphosis was developed by Athanas[13]. The main idea behind the PRISM is to use information extracted at compile-time from a given application and to build a new set of instructions tailored to that application.

1.2.7 The GARP Approach

The Garp[14] was proposed by approach proposed by Hauser and Wawrzynek. It is very close to today's embedded FPGA systems. The main goal was to overcome the obstacles of reconfigurable computing machines, mostly consisting of a processor attached to a set of FPGA boards. Hauser and Wawrzynek proposed a system consisting of a microprocessor and a reconfigurable array on the same die with for transferring time of data to the reconfigurable array could be drastically reduced.

1.2.8 DISC

The purpose of the dynamic instruction set computer (DISC) [15] was presented to support demand-driven instruction set modifications. In PRISM approach, the specific instructions are synthesized and fixed at compiled-time, but the DISC approach uses partial reconfiguration of FPGAs to place on the FPGA, hardware modules, each of which implements a given instruction. The relocation was also proposed as a means to reduced defragmentation of the FPGA.

1.2.9 The DPGA

Like Garp architecture, dynamically programmable gate array was proposed by Dehon. This was proposed as a means to extend the capability of microprocessor [16]. However, DGPAs provide a better implementation of the reconfigurable array proposed in the Garp. The concept proposed by Dehon is used in many coarse-grained devices available today.

Here are some recent research works on application of reconfigurable computing.

The paper by J. A. Surís et al. [17] "RapidRadio: A Domain-Specific Productivity Enhancing Framework'' addresses a framework enhancing tool reducing the required knowledge base for implementing a receiver on an FPGA-based SDR platform. In "Space-Based FPGA Radio Receiver Design, Debug, and Development of a Radiation-Tolerant Computing System," Z. K. Baker et al.[18] proposed a processing capability which enables very advanced algorithms such as wideband RF compression scheme to operate at the source, allowing bandwidth-constrained applications to deliver previously unattainable performance.

In paper **"**A Decimal Floating-Point Accurate Scalar Product Unit with a Parallel Fixed-Point Multiplier on a Virtex-5 FPGA", by M. Baesler et al.[19] presents a new parallel decimal fixed-point multiplier designed to exploit the features of Virtex-5 FPGA in order to provide important operation with least possible rounding error in the paper In "Partial Reconfigurable FIR Filtering System Using Distributed Arithmetic," D. Llamocca et al. [20] proposed two efficient dynamic partial reconfiguration systems allowing implementing a wide range of 1D FIR filters. For both systems, the required partial reconfiguration region is kept small by using distributed arithmetic implementations.

R. A. Arce-Nazario et al. [21], in "Reconfigurable Hardware Implementation of a Multivariate Polynomial Interpolation Algorithm," address a new methodology based on Lagrange interpolation. The generalized algorithm can be efficiently mapped to a systolic array in which each processing cell implements a pair of binary operations between an incoming and a stored value.

In "Reconfigurable Multiprocessor Systems: a Review" by T. Dorta et al.[22], run-time reconfigurability uses the dynamic reconfiguration feature of FPGAs to obtain a new degree of freedom in the design of multiprocessor systems, making these systems more flexible to target different applications using the same hardware.
L. Kirischian et al. [23] proposed in "Mechanism of Resource Virtualization in RCS for Multitask Stream Applications" a mechanism for the virtualization of computing resources for multitask and multi-mode stream applications. The comparative analysis made on the Multitask Adaptive Reconfigurable System demonstrated negligible hardware overhead offset by sufficient gains in the main performance parameters. Memory access is a key point for modern design.
In "Traversal Caches: A Framework for FPGA Acceleration of Pointer Data Structures," J. Coole and G. Stitt et al. [24] introduce an original traversal cache framework that enables efficient FPGA execution of applications with irregular memory access patterns. Such approach greatly improves effective memory bandwidth for repeated traversals.
L. Sauvage et al.[25] proposed a new logic for FPGA to prevent robust side channel attacks in "Exploiting Dual-Output Programmable Blocks to Balance Secure Dual-Rail Logics". It describes that the differential placement and routing of an FPGA implementation can be done with a granularity fine enough to improve the security gain.
Generating true random number is a major issue in security concerns. The paper "True-Randomness and Pseudo-Randomness in Ring Oscillator-Based True Random Number Generators," by N. Bochard et al. [26] deals with true random number generators employing oscillator rings. A mathematical analysis, a simulation model, and FPGA implementations are discussed in depth.
FPGA could be efficiently used for robotics applications. J. Kalomiros and J. Lygouras et al. [27] in "Robotic Mapping and Localization with Real-Time Dense Stereo on Reconfigurable Hardware" proposed a new reconfigurable architecture for dense stereo and an observation framework for a real-time implementation of the simultaneous localization and mapping problem in robotics. In "A Reconfigurable System Approach to the Direct Kinematics of a 5 D.o.f Robotic Manipulator" by D. F. Sánchez et al. [28] describes an FPGA implementation of kinematics of a spherical robot manipulator using floating-point units operators. The proposed architecture was designed using a Time-Constrained Scheduling. Synthesis results show that the proposed hardware architecture for direct kinematics of robots is feasible in modern FPGAs families.
S. Shreejith, S. A. Fahmy, and M. Lukasiewycz et al. [29] proposed how a modern vehicles incorporate a significant amount of computation, which has led to an increase in the number of computational nodes and the need for faster in-vehicle networks. Functions range from noncritical control of electric windows, through critical drive-by-wire systems, to entertainment applications; as more systems are automated, this variety and number will continue to increase. Accommodating the varying computational and communication requirements of such a diverse range of functions requires flexible networks and embedded computing devices. As the number of electronic control units (ECUs) increases, power and efficiency become more important, more so in next-generation electric vehicles. Moreover, predictability and isolation of safety-critical functions are nontrivial challenges when aggregating multiple functions onto fewer nodes. Reconfigurable computing can play a key role in addressing these challenges, providing both static and dynamic flexibility, with high computational capabilities, at lower power consumption. Reconfigurable hardware also provides resources and methods to address deterministic requirements, reliability and isolation of aggregated functions. This letter presents some initial research on the place of reconfigurable computing in future vehicles.
In 'The Use of Fuzzy Clustering and Correlation to Implement a Heart Disease Diagnosing System in FPGA' Evaldo Renó Faria Cintra, Tales Cleber Pimenta, Robson Luiz Moreno et al. [30] shows how a signal processing method capable of detecting cardiopathies in electrocardiograms that was implemented in FPGA.
In 'An Intelligent Wireless Sensor Network Temperature Acquisition System with an FPGA', Rachid Souissi, Mohsen Ben-Ammar et al.[31] proposed the use of FPGAs in the design of wireless sensor networks can improve the processing system performance so as to develop sensor nodes with powerful embedded processor.

2. Operating System for Reconfigurable Computing:

Reconfigurable computing dynamically computes the reconfigurable data and control path. It is an addition to a simple processor system in which only a single fixed data and control path is presented. Basically reconfigurable computing platforms are designed to be single user and have been acknowledged to be difficult to design applications for. The first OS is a loader, and at run- time the applications are allocated to FPGA area by it through dynamically partition, placement and routing. Reconfigurable computers consist of a workstation host which is loosely coupled via a general-purpose bus to a Field Programmable Gate Array (FPGA). Managing the time evolution of a single application is an issue for run-time management of reconfigurable systems.

The new FPGA architectures, such as the Time multiplexed FPGA [32], partially reconfigurable FPGA [33] and FPGA co processing boards, such as SPACE2 [34], accepts multiple dependent or independent circuits. Earlier work on operating systems for reconfigurable computers includes an implementation of an OS for the Xputer [35]. Later the OS was extended for the host operating system which decides that what parts of the circuit will run on the configurable hardware and what parts will run on the host-hardware, in conventional software. This OS supports multiple users, but it achieves this multi-user by computing circuits sequentially, rather than concurrently. Brebner emphasized the issues related to manage a virtual hardware resource [36]. He proposed decomposing reconfigurable computing applications into swappable logic units (SLUs), which describe circuits of fixed area and input/output (I/O) interfaces, so that multiple independent tasks might share a single FPGA. This concept restricts designs to fixed sizes, limiting the designer's flexibility. Diessel [37] described a web-based multi-user operating system that shared the SPACE.2 [34] architecture amongst eight simultaneous users.

The first limitation of this OS was that a complete FPGA is required to be assigned to one circuit and therefore potentially wastes FPGA logic, as not all users would require the complete FPGA surface. Secondly the circuits have to be designed with restricting specifications such as the position of input and output registers. Although the OS is multiuser, it allocates a user to one of the eight FPGAs and doesn't allow any more than one circuit per FPGA, thus ruling out concurrent multi-user. Wigley & David et al.[38] in "The Development of an Operating System for Reconfigurable Computing", proposed an OS which minimizes the restrictions placed onto designers at the design level, performs true single chip concurrent multi-user, which executes more than one circuit on a single FPGA at one time and automatically modify the circuit to accommodate it onto the FPGA.

2.1 An Operating System's Architecture for Reconfigurable Computing:

Loader is a fundamental service provided by a traditional OS is the loading and initiation of execution of programs. Loading a reconfigurable computing "program" means placing the circuit and embedded RAM on the FPGA and then routing its external I/O interface to either neighboring circuits or to a local or global communications bus. Placement and routing are done at the low level and that the module is thus, in the sense of software technology, both a precompiled and relocatable module. The time available for the place and route in the context of a reconfigurable OS is a significant constraint on the types of algorithms used and on the level of optimization that can be achieved. Loading an application onto an FPGA implies immediate execution. Following are the basic functions of OS.

Scheduler:

Scheduling is a major issue as it decides which job has to be done at which time. Scheduling reconfigurable applications doesn't support preemption properly. As a consequence multitasking of reconfigurable applications should be cooperative. The developer has to provide a well defined complete signal for their application or alternatively, where applications are automatically partitioned the partitioning algorithm inserts these signals at the cut point [38].

Virtual Memory

The necessity of virtual memory on a reconfigurable computer is quite same with traditional OS. Here also the applications can be loaded beyond the capacity of the FPGA resources and the OS can select parts of applications that will be placed on the FPGA according to requirement. But in case of reconfigurable application this has to fit into available page frames that are not currently occupied by other circuits is non-trivial.

Cache Management

A cache hierarchy of FPGA chips can be conceived [39] and then augmented by another hierarchy of RAM to temporarily store the configurations. It is possible to purchase various speed grades of FPGA with identical architectures, which could be composed into such a hierarchy, and the OS could be asked to manage the placement of circuits in the hierarchy so that the application kernel would run the fastest chip.

I/O

An I/O mechanism for reconfigurable applications is Direct Memory Access as the DMA hardware can be treated as a part of the application. As there is no obvious processor cycle in most applications so in reconfigurable computing the interrupts can be considered. FPGA designs don't support any active communications those are not initiated by certain software programs running on the controlling processor. According to the usability of the system, the presence of active I/O services is intuitive.

Protection

Protection is a major issue in traditional operating system as well as OS for reconfigurable computing. In FPGA the protection is provided by a bounding box beyond which the application circuit cannot be interconnected. A reconfigurable application might also be able to access RAM of the FPGA.

Inter-process Communication

Reconfigurable computing supports inter-process communication using several hardware and software mechanism. In reconfigurable computer the applications can be wired together and the results are stored in registers for subsequent copying to other parts. Like system call interface of traditional operating systems, there should be a standard interface to be maintained on between application programmers. Finally in OS any applications can be transferred from the FPGA by reading the direct access configuration of the FPGA. The results of one application can be written back to the input of another application programme.

Allocation

The circuit allocation determines which parts of the FPGA logic are unused and which parts can be assigned to circuits for execution. In the traditional single user FPGA system, the designer completes the process of allocating tasks to the FPGA logic. As there isn't any other circuits to take into consideration the designer can allocate them to any part of the FPGA they desire. In the multi-user FPGA system the
OS must manage the allocation of circuits to the logic space as the FPGA resource varies over time. The OS can manage the allocation of FPGA logic as when circuits are placed onto the FPGA, the area they consume is noted in a separate OS table and when a circuit requires allocation the OS consults this area free table. The OS would 'scan' the FPGA surface when a circuit requires allocation, looking for the amount of free logic requested.

Dynamic Partitioning

All applications in the OS are represented by a task graph. In a task graph each node represents an operator or a function and the directed edges represent the data dependencies between the nodes of the graph. Dynamic partitioning is the process whereby a large task graph is broken down into smaller components, similar to that of dividing standard programs up into pages.

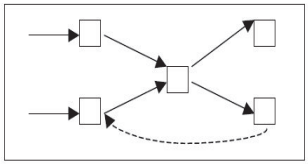

Fig 4: Task Graph

Placement

The nodes of the circuit are to be placed onto the FPGA cells. Placement determines that which node should be placed on which FPGA cell. If a part of the FPGA has been allotted for the specified circuit, then the nodes of the circuit have to be individually allocated within that specified region on the FPGA.

Routing

With run-time circuit placement comes the issue of runtime routing. In traditional routing it is a problem to create an electrical connection between two points by setting the appropriate routing switches. As most FPGAs (for example the Xilinx 6200 series FPGAs) have several hierarchy routing resources. Here the number of different routing paths is increased substantially.

III. CONCLUSION

Reconfigurable computing is becoming one of the most important areas of research in the field computer of architectures and software systems. By placing the computationally intense part of an algorithm can be executed by the reconfigurable hardware, the overall execution speed can be greatly accelerated. This is because reconfigurable computing combines the benefits the flexibility of the software and the speed of the hardware. The traditional Von Neumann or Harvard architecture model can't exploit the inherent parallelism of an algorithm which exhibits parallelism. In a reconfigurable computing engine the parallelism can be exploited. Now a day in various research domain reconfigurable computing is being used. In the field of robotics, number systems as well as in various medical aspects, reconfigurable computing is induced. However, the success of a reconfigurable architecture lies in the development of reconfigurable operating systems. Hence, future research should be carried on that direction.

REFERENCES

[1] Introduction to Reconfigurable Computing Architectures Algorithms and Application by Christophe Bobda University of Kaiserslautern, Germany
[2] Application Hardware-Software Co-Design for Reconfigurable Computing Systems by Proshanta Saha, Electrical and Computer Engineering of The George Washington University, May, 2008.
[3] BORPH: An Operating System for FPGA-Based Reconfigurable Computers,Hayden Kwok-Hay So Robert W. Brodersen, Electrical Engineering and Computer Sciences University of California at Berkeley, July, 2007.
[4] G. Wigley & D. Kearney ,"The First Real Operating System for Reconfigurable Computers", Australian Computer Science Communications , Volume 23 Issue 4, January 2001 .
[5] Advanced Computing Research Centre School of Computer and Information Science, University of South Australia.
[6] G. Estrin and R. Turn, "Automatic assignment of computations in a variable structure computer system," IEEE Transactions on Electronic Computers, vol. 12, no. 5, pp. 755–773, 1963.
[7] F. J. Rammig, "A concept for the editing of hardware resulting in an automatic hardware-editor," in Proceedings of 14th Design Automation Conference, New Orleans, 1977, pp. 187–193.
[8] R. Hartenstein, A. Hirschbiel, and M.Weber, "Xputers - an open family of non von Neumann architectures," in 11th ITG/GI Conference on Architektur von Rechensystemen. VDE-Verlag, 1990.
[9] Xilinx Inc., "The early access partial reconfiguration lounge," (registration required). [Online]. Available: http://www.xilinx.com/support/prealounge/protected/index.htm
[10] P. Bertin, D. Roncin, and J. Vuillemin, "Introduction to programmable active memories," in Systolic Array Processors, J. McCanny, J. McWhirter, and E. Swartzlander, Eds. Prentice Hall, 1989. [Online]. Available: citeseer.ist.psu.edu/bertin89introduction.html
[11] M. Gokhale, W. Holmes, A. Kopser, D. Kunze, D. P. Lopresti, S. Lucas, R. Minnich, and P. Olsen, "Splash: A reconfigurable linear logic array." in ICPP (1), 1990, pp. 526–532.
[12] D. A. Buel, J. M. Arnold, andW. J. Kleinfelder, Splash 2 FPGAs in a Custom Computing Machine. IEEE Computer Society Press, Los Alamitos, California, 1996.
[13] P. M. Athanas and H. F. Silverman, "Processor reconfiguration through instruction-set metamorphosis," IEEE Computer, vol. 26, no. 3, pp. 11–18, 1993, iteseer.nj.nec.com/athanas93processor.html. [Online]. Available: citeseer.nj.nec.com/athanas93processor.html
[14] J. R. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," in IEEE Symposium on FPGAs for Custom Computing Machines, K. L. Pocek and J. Arnold, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1997, pp. 12–21. [Online]. Available: citeseer.nj.nec.com/hauser97garp.html
[15] M. Wirthlin and B. Hutchings, "A dynamic instruction set computer," in IEEE Symposium on FPGAs for Custom Computing Machines, P. Athanas and K. L. Pocek, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1995, pp. 99–107. [Online]. Available: citeseer.nj.nec.com/wirthlin95dynamic.html
[16] A. DeHon, "DPGA-coupled microprocessors: Commodity ICs for the early 21st century," in IEEE Workshop on FPGAs for Custom Computing Machines, D. A. Buell and K. L. Pocek, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1994, pp. 31–39. [Online]. Available: citeseer.ist.psu.edu/dehon94dpgacoupled.html
[17] J. A. Surís, "RapidRadio: A Domain-Specific Productivity Enhancing Framework''
[18] Z. K. Baker, "Space-Based FPGA Radio Receiver Design, Debug, and Development of a Radiation-Tolerant Computing System," International Journal of Reconfigurable Computing Volume 2010 (2010)
[19] M. Baesler, S.O. Voigt, T. Teufel , " A Decimal Floating-Point Accurate Scalar Product Unit with a Parallel Fixed-Point Multiplier on a Virtex-5 FPGA", International Journal of Reconfigurable Computing Volume 2010 (2010)
[20] D. Llamocca, "Partial Reconfigurable FIR Filtering System Using Distributed Arithmetic," International Journal of Reconfigurable Computing Volume 2010 (2010)
[21] R. A. Arce-Nazario, "Reconfigurable Hardware Implementation of a Multivariate Polynomial Interpolation Algorithm," International Journal of Reconfigurable Computing Volume 2010 (2010)
[22] T. Dorta, "Reconfigurable Multiprocessor Systems: a Review" , International Journal of Reconfigurable Computing Volume 2010 (2010)
[23] L. Kirischian, "Mechanism of Resource Virtualization in RCS for Multitask Stream Applications", International Journal of Reconfigurable Computing Volume 2010 (2010)
[24] J. Coole and G. Stitt, "Traversal Caches: A Framework for FPGA Acceleration of Pointer Data Structures," International Journal of Reconfigurable Computing Volume 2010 (2010)
[25] L. Sauvage, "Exploiting Dual-Output Programmable Blocks to Balance Secure Dual-Rail Logics," International Journal of Reconfigurable Computing Volume 2010 (2010)
[26] N. Bochard, "True-Randomness and Pseudo-Randomness in Ring Oscillator-Based True Random Number Generators," International Journal of Reconfigurable Computing Volume 2010 (2010)
[27] J. Kalomiros and J. Lygouras, "Robotic Mapping and Localization with Real-Time Dense Stereo on Reconfigurable Hardware", International Journal of Reconfigurable Computing Volume 2010 (2010)
[28] D. F. Sánchez, "A Reconfigurable System Approach to the Direct Kinematics of a 5 D.o.f Robotic Manipulator", International Journal of Reconfigurable Computing Volume 2010 (2010)
[29] S. Shreejith, S. A. Fahmy, and M. Lukasiewycz, "Reconfigurable Computing in Next Generation Automotive Networks", Embedded Systems Letters, IEEE, 2013
[30] R. E. R. Faria Cintra, T. C. P. Luiz Moreno 'The Use of Fuzzy Clustering and Correlation to Implement a Heart Disease Diagnosing System in FPGA', International Journal of Reconfigurable Computing Volume 2010 (2010)
[31] R. Souissi, M. Ben-Ammar, 'An Intelligent Wireless Sensor Network Temperature Acquisition System with an FPGA', International Journal of Reconfigurable Computing Volume 2010 (2010)
[32] Trimberger, S. A Time-Multiplexed FPGA Proceedings FPGAs for Custom Computing Machines,FCCM 97, IEEE Cmputer Society, 1997.
[33] Xilinx. XC6200 Field Programmable Gate Arrays. Xilinx, Inc., April 1997
[34] Gunther, B. SPACE 2 as a reconfigurable stream processor. Proceedings of PART'97 the 4th Australasian Conference on Parallel and Real-Time Systems, pages 286 – 297, Singapore, September 1997. Springer- Verlag.

[35] Hartenstein, R. W. and Kress, R. and Nageldinger, U. An Operating System for Custom Computing Machines based on the Xputer Paradigm. Proceedings of the 7th International Workshop on Field Programmable Logic,FPL 97, September 1997.
[36] Brebner, G. A virtual hardware operating system for the Xilinx XC6200. In Field-Programmable Logic: Smart Applications, New Paradigms and Compilers, 6th International Workshop, FPL 96 Proceedings, pages 327 – 336, September 1996.
[37] Reconfigurable Computing. Proceedings of IPPS/SPDP'99 Parellel and Distributed Processing, pages 579 – 587, Germany, 1999. Springer-Verlag.
[38] Wigley & David, "The Development of an Operating System for Reconfigurable Computing", Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)
[39] Kearney, D, and Kiefer, R. A High Bandwidth I/O Architecture for a Reconfigurable Computer. In Proceedings 4th Australasian Computer Architecture Conference, January 1999.