Performance Enhancement Techniques for Modern Reconfigurable Computing Systems

M. Aqeel Iqbal
Dept. of CE, College of Electrical & Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan

Farooque Azam
Dept. of CE, College of Electrical & Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan

Uzma Saeed Awan
Dept. of CS, COMSATS, Institute of Information Technology (CIIT), Abbottabad, Pakistan

Saifullah Hammad
Advanced Engineering Research Organization (AERO) Islamabad, Pakistan

ABSTRACT
Reconfigurable computing has become an essential part of research in the domain of modern computing paradigms. Reconfigurable computing approach integrates both, the performance and flexibility gaining aspects on a single computing system. The computational performance of such kind of systems is crucially dependant on the configuration overheads caused by configuration management unit. Performance of the configuration management unit greatly accelerates the computational power of reconfigurable computing system. There are a large number of control and management techniques which can be used to improve this technology. This research paper presents a comprehensive analysis of existing performance enhancement methodologies in practice. The paper also point outs the different aspects of configuration management for critical analysis and further optimization.

Keywords
Configuration Management, Configuration Streams, Configuration Overheads, Performance Management, Reconfigurable Computing.

1. INTRODUCTION TO RECONFIGURABLE COMPUTING
Consider the Fig.1 which shows the basic computing themes of ASICs, GPPs and the position of Reconfigurable Computing in between them. In order to formalize the concept reconfigurable computing paradigm, such kind of computing devices or computing systems are required which should demonstrate the expected computational functionality. The reconfigurable computing systems have varying characteristics to demonstrate the worth for modern applications. The reconfigurable computing platforms consist of a reconfigurable computing accelerator which provides the expected programmable logic resource for a configuration management unit [1]. Configuration management unit controls the execution of the required tasks on reconfigurable computing resources.

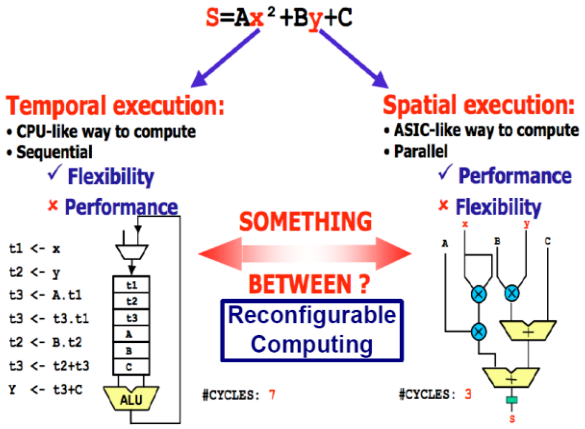


Figure courtesy of "Eduardo Sanchez" (Lecture slides).

Figure-1: Concept of Reconfigurable Computing

The design and control complexity of the configuration management unit is widely varying depending upon the desired application and the computational approach being used to achieve the desired goals. Consider the Figure-2 which shows the composition of basic reconfigurable computing system.

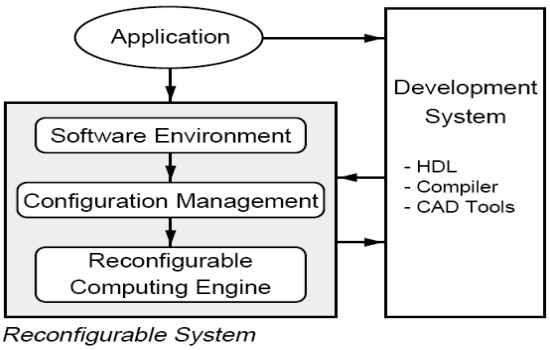
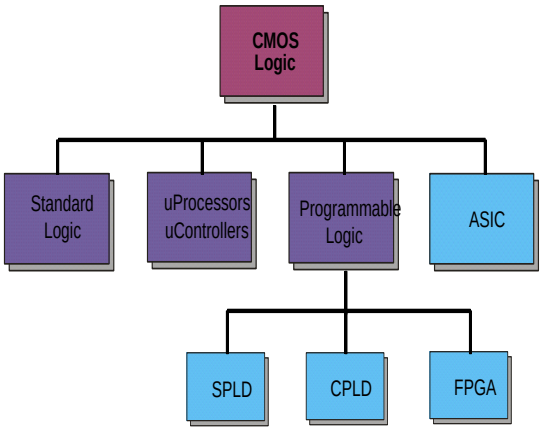


Figure-2: Composition of a Reconfigurable Computing System

The most critical elements of the development system have a well defined design and implementation flow that has been adapted typically for modern reconfigurable computing systems [1], [2]. The compilers used in reconfigurable computing systems allow easy and fast compilation facility of modern high-level software descriptions into hardware circuits. Also the CAD tools now fully support the emerging reconfigurable computing mechanisms including partial and run-time reconfiguration.

2. RESOURCE MANAGEMENT

Since the modern applications support the multiprogramming and multi-threading for performance enhancement and better user service availability, the consideration of multiple running applications on a single system demands that there should be some kind of controlling mechanisms and supporting policies that should manage such type of resource competition among different running applications and should be able to resolve any kind of resource allocation conflict [3]. Consider the Figure-3 which shows the basic categorization of programmable logic devices.



10

Figure-3: Categorization of Programmable Devices

In a conventional computing system, these resource management activities include the data storage management and I/O resource management to communicate with external world. Allocation of one active application at a time to a single field programmable gate array device is the most simple and easiest way to manage the resource allocation problem. This kind of strategy of resource allocation and management has already been implemented with positive and negative aspects in dynamically reconfigurable computing system. Consider the Figure-4 which shows the internal architecture of a typical FPGA.

Later on researchers proposed operating system based approaches for managing configuration overheads for partially reconfigured computing devices or systems. The basic idea behind this operating system based approach was to design the swappable hardware configuration modules. These modules are in fact some kinds of position independent tasks which can be immediately swapped in and swapped out by the active operating system being interfaced with the reconfigurable device [2], [4].

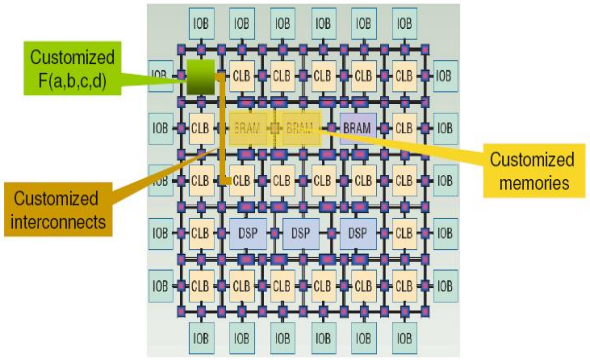


Figure-4: Reconfigurable Logic Device

These reconfigurable modules are in fact logic segments of FPGA. These logic modules are supposed to be of equal size and can be allocated to active application. This assumption of having equal sized blocks within an application has very limited uses since it is mostly very difficult to divide a running application into the segments of equal size. Hence later on the researchers also improved the concept and proposed the reconfigurable computing hardware modules having un-equal sizes and dimensions [4]. Consider the Figure-5 which shows the fundamental logic of internal design of FPGA.

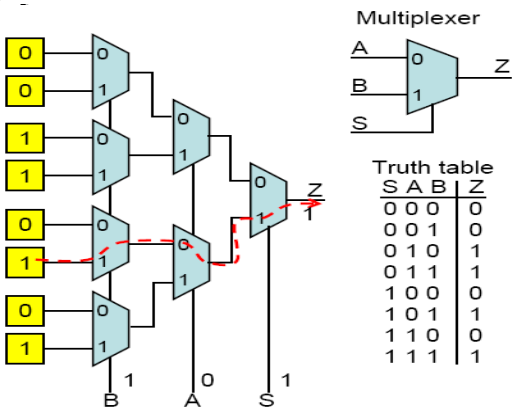


Figure-5: Basic Logic in side FPGA

The internal routing between the logic blocks being available within the reconfigurable logic device is most important aspect. This kind of internal routing in fact greatly contributes to the overall performance of the reconfigurable computing device [5]. When the overall usage of the logic blocks in field programmable gate array becomes very high, then the automatically routing tools frequently come across the difficulty of achieving the necessary internal connections between the successive and adjacent blocks.

Therefore effective routing structures are essential to ensure that a design can be successfully placed and routed onto the reconfigurable computing logic hardware. Also the arbitrarily relocated computing modules need to internally communicate with each other and also with I/O devices of the system. Hence in this regard an online place and routing mechanism is required to establish and enable this internal to external communication. Consider the Figure-6 which shows the target parameters for making reconfigurable computing to be successful.

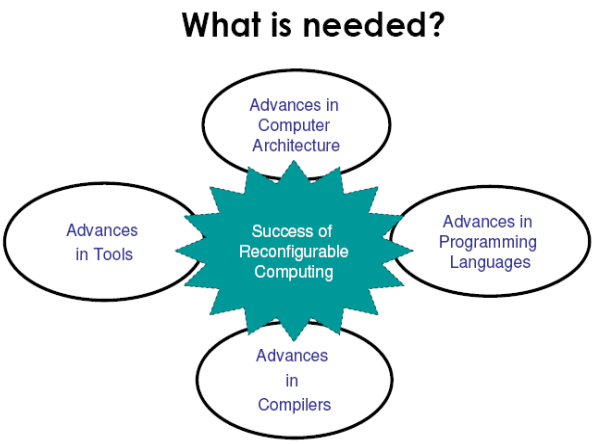


Figure-6: Success of Reconfigurable Computing

In order to support high performance reconfigurable computing systems that are mostly composed of several Field Programmable Gate Array (FPGA) chips integrated on a single processing board have additional hardware concerns as compared to the single-chip reconfigurable computing systems [6]. For such kind of systems there is always a critical need for an efficient interconnection scheme between the arrays of chips, as well as the connection of these chips to external memory devices and the system bus [7].

Such kind of huge reconfigurable computing systems are used for those circuits which are too heavy and cannot fit within a single FPGA device. Such circuits are then partitioned to run on the multiple FPGAs systems available. There is a need of efficient communication between the multiple FPGAs and hence the determining of the inter-chip routing is one of the most important steps being involved in the design of a multi-FPGA based systems [8], [9]. Consider the Figure-7 which shows the programming flow while working for a typical FPGA.

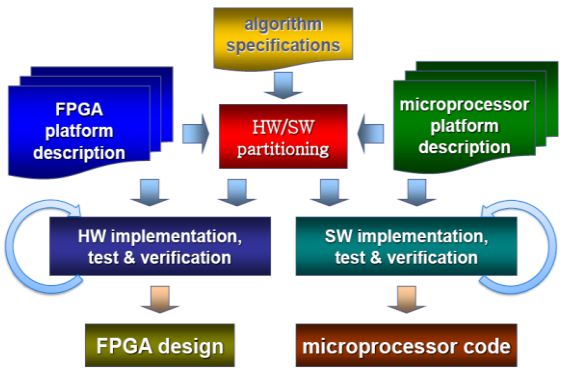


Figure-7: FPGA Based Programming Flow

Figure Source: Lecture slides "Introduction to Reconfigurable Computing" By Volodymyr Kindratenko, NCSA Craig Steffen, NCSA

3. Performance Management Techniques

In reconfigurable computing devices such as modern FPGAs, the reconfigurable computing logic and programmable routing resources are managed by reprogrammable internal memory locations, such as found in Static RAM or Flash memory. Binary bits available in the memory locations control the connectivity of certain wires and also define the functionality which has been implemented by a particular piece of computing logic. These binary bits are typically known as configuration Bit Streams [10]. The process of loading these configuration bit streams into the memory locations is known as *reconfiguration*. A set of specific bits sequence for certain memory locations in reconfigurable device like FPGA basically defines a specific hardware circuit. This set of streams defining the hardware circuits is commonly known as a *Configuration* for a given piece of hardware module [11]. The modern practices of loading and unloading configuration streams back and forth like swapping in and swapping out operating systems processes, is known as the Run-time Reconfiguration (RTR).

These configurations are basically created by the Computer Aided Design (CAD) software tools. The configuration streams are dependent on the type of the circuit to be designed and the internal architecture of the reconfigurable computing device. After generation of the configurations by the CAD tools, they are generally stored in a non volatile memory available external to the reconfigurable device. But this is not the only way to store the configurations. In certain cases the configurations are stored in the main memory (SRAM or DRAM) and it is the responsibility of microprocessor of the system to keep transferring them from the memory to the reconfigurable device as per the requirements of the running application. There is another technique for managing the configurations. In this technique the configurations are stored in a Programmable Read Only Memory (PROM) and a configuration controller which is a piece of hardware loads this stream directly from this PROM to the reconfigurable device.

Mostly in such kind of systems the configuration controller and the PROM both are integrated into the same chip or device. In these systems the configuration controller is implemented in the form of a finite-state machine that generates the sequences of memory addresses needed to read an appropriate bit stream sequence from the configuration memory. There are many methods which are used to minimize the time delays required by the system to load the required configurations. In the following some most prominent configuration overhead optimization techniques have been presented.

3.1 Single-context Configuration

The Virtex family of field programmable gate arrays provided by the Xilinx Corporation has addressable internal configuration locations inside the configuration memory. These devices support a single-context configuration mode. In these devices, the configuration bit streams are divided into addressable blocks commonly known as Frames. Each one of these frames is in fact corresponding to part of a vertical column of reconfigurable hardware resource. In the process of device reconfiguration, the configuration bit streams are shifted block wise into the frame input register and from there it is written back to a configuration memory location which is being specified by the frame address register.

Those devices which support the single context configuration mode, this configuration stream address begin at zero and are automatically incremented gradually for each of the new arriving frame [12]. Due to this characteristics it allows the FPGA device to appear as a single-context device as viewed externally. In order to make the configuration process fast the

number of configuration cycles can be minimized in single context devices by simply extending the configuration path [13].

3.2 Multi-context Configuration

For run-time reconfigurable systems, the configuration overhead of serial loading the bit streams may be harming for high speed applications. In order to solve this problem it is an emerging practice to provide a set of local storages in the FPGA device for holding temporary multiple configurations simultaneously. In this way it becomes quite feasible to facilitate the concepts of configuration pre-fetching and fast reconfiguration. An FPGA device which supports the *multi-context capability* contains a set of multiple active contexts of configuration bit streams. Depending upon the application requirements each configuration block of the FPGA device is internally controlled by a digital multiplexer circuit which chooses among the active contexts.

In general the multi-context FPGA devices have two major advantages as compared to the single-context FPGA devices. First advantage is that they allow the loading of configuration bit streams in the background meanwhile maintaining the operation of the system and hence demonstrating the configuration overlapping with computation of the running system [13], [14]. The second main advantage is that as these devices support the multiple stored configurations, hence they can be switched between internally stored configurations quickly. Most of these devices can switch between the multiple active configurations within a very short interval of almost equal to one cycle.

In this way these devices dramatically reduce the reconfiguration overhead of the device if the next required configuration is already available in one of the other active contexts. However, on the other hand if the next required configuration is not available, there would be a fraction of penalty faced for loading the configuration. In order to manage this issue it is recommended that either all of the required contexts must fully fit in the available hardware device or some kind of internal control mechanism should be provided that should determine when contexts should be loaded in the device in order to efficiently minimize the overall number of wasted cycles during the reconfiguration process completes [14]. Consider the Figure-8 which shows the concept of partial reconfiguration.

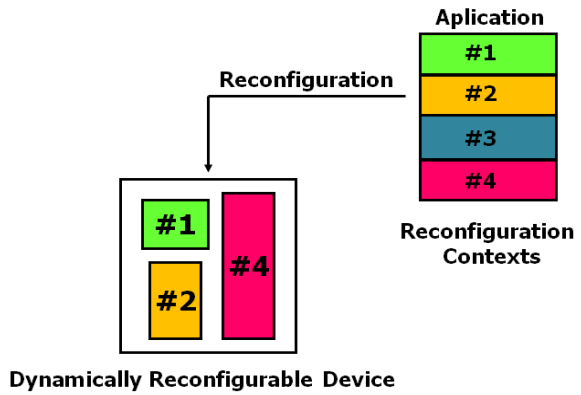


Figure-8: Partial Reconfiguration

3.3 Partial Reconfiguration

It is experimented that while configuring the device it is not always required to configure the whole device because not all kinds of configurations require the entire chip area. Hence it is possible to reduce the configuration overhead if we reload only those streams which are new and reuse those streams which match the next incoming streams. In those devices which support the process of partial reconfiguration, the configuration memory is addressable. In a similar way as that of traditional RAM memory structures. If required configurations are smaller as compared to full reconfigurable device, the partial reconfiguration can be used to actively decrease the reconfiguration overhead. The process of partial reconfiguration can also permit a set of multiple independent configurations to be swapped in and swapped of the reconfigurable device quite independently. Hence one of the configurations can be selectively replaced on the device while another one is left intact.

3.4 Pipelined Reconfiguration

This is another modern technique used to effectively reduce the configuration overhead. In such type of systems a series of physical pipeline stages are implemented to provide the concept of the virtual pipeline stages of configurations. In this method any virtual pipeline stage can actively be relocated to any physical pipeline stage available. Also the number of virtual pipeline stages is not limited by the number of physical available pipeline stages. PipeRench is one of the most prominent systems having this concept being implemented. This system has been designed to implement pipelined configurations, which have been subdivided into a set of virtual pipeline stages. During the active operation of the device, these virtual pipeline stages are assigned to one of the available physical pipeline stage. Over the time the pipeline stages may be implemented in different physical locations. Hence the virtual pipeline appears quite fixed to its internal own pipeline stages, with each pipeline stage receiving its inputs from its predecessor stage and hence generating the output to its successor stage. In the PipeRench the pipeline stages can be reconfigured in a single clock cycle to speed up the device execution.

3.5 Configuration Cloning

Regularity and locality is exploited in the process of configuration cloning while the device is under reconfiguration. The configuration cloning is performed in FPGA device by copying the single or multiple bit-streams from one region of device to one or multiple other regions. Hence without loading entire configuration bit-stream, the active device can be reconfigured at an instant. The process of configuration cloning can be used to greatly reduce the device configuration overhead. But overall this method of configuration cloning is not very much realistic [12], [14]. This method requires the FPGA device to send all bits of the configuration stream from multiple cells of the device in a column/ row to several other available cells in the same column/row concurrently.

Such kind of configuration copying process from one region of FPGA chip to another region requires a large number of internal routing resources and very complex control circuitry [15]. It may also require a set of large switch matrixes, all of which can impose a significant amount of area overhead. Also this kind of configuration process may require very high level of regularity. Hence it might be best suited only to hand-mapped circuits and those kind of circuits which exist in the form of arrays of similar replicated cells. Due to all of these technical limitations, this configuration method has a very limited utility.

4. CLASSIFICATION OF EXISTING SYSTEMS

The main criteria of the classifications of the existing reconfigurable computing systems are based on the implementation of the reconfigurable logic and its placement inside computing system. The reconfigurable logic can be placed at four different levels with respect to the placement of the standard processing logic in the system. Consider the Figure-11 which shows the different coupling approaches for reconfigurable logic inside a reconfigurable computing system.

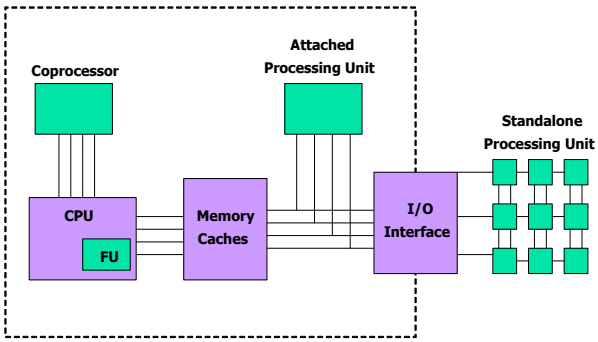


Figure-11: Coupling Approaches for Reconfigurable Unit

The characteristics of each of the four types are given as under.

4.1 Placement as a Functional Unit

- Commonly known as Tightly Coupled Systems
- FPGAs are working as main functional Units
- Internal registers hold input/output data streams
- They have very less communication overhead
- Reconfigurable hardware cannot operate alone for long periods of time
- Amount of reconfigurable logic being implemented is limited
- Examples of such systems include OneChip, PRISC and Chameleon etc

4.2 Placement as a Co-Processor

- They perform computations without the constant supervision of host system
- Host system initializes the reconfigurable hardware
- They can perform the Independent parallel computations
- They have less communication overhead
- Examples of such systems include PRISM and GARP etc

4.3 Placement as a Processing Unit attached to Memory

- These systems does not share cache
- They behave as an additional processor
- They exhibit DMA-type overlap interface
- Higher delay to communicate with host processor
- They can perform independent computations
- Examples of such systems include SPLASH and SPLASH-2 etc

4.4 Placement as an External Processor Attached to I/O

- They are commonly known as loosely Coupled systems
- They are most loosely coupled to host processor
- They have infrequent Communication with host processor
- They can perform independent computation for very long periods of time
- They have higher communication overheads
- Examples of such systems include FPGA on a PCI bus

5. CONCLUSION

Reconfigurable computing is becoming an important part of emerging research. Reconfigurable computing systems can be used to provide the benefits of high speed ASICs and programmable processors. The main issues related to reconfigurable computing include the minimization of device configuration overheads. Configuration overheads can be greatly reduced by using the emerging concepts of multiple context switching, partial reconfiguration, configuration cloning and configuration pipelining. In this regard there are many dimensions of further research to fully exploit the power of the reconfigurable computing systems.

6. REFERENCES

[1] M. Aqeel Iqbal, Asia Khannum, Saleem Iqbal and M. Asif, "Emerging Requirements of Reconfigurable Computing Systems for Performance Enhancement", Published in the International Journal on Computer Science and Engineering (IJCSE), Volume-02, No-05, PP 1572-1579, August-2010, Published by Engineering Journals Publications. ISSN: 0975-3397.

[2] M. Aqeel Iqbal, Shoab A. Khan and Uzma Saeed Awan, "Computational Unit Design For High Speed Reconfigurable Processors", Published in International Journal of Intelligent Information Technology Application (IJIITA), Volume-02, No-05, PP 229-236, October-2009, Published by Engineering Technology Press. ISSN: 1999-2459.

[3] M. Aqeel Iqbal, Shoab A. Khan and Uzma Saeed Awan, "Reconfigurable Computing Systems Related Hardware and Software Perspectives", Published in International Journal of Intelligent Information Technology Application (IJIITA), Volume-02, No-05, PP 209-217, October-2009, Published by Engineering Technology Press. ISSN: 1999-2459.

[4] M. Aqeel Iqbal, "RFU Based Computational Unit Design for Reconfigurable Processors", Published in Journal of World Academy of Science, Engineering and Technology (WASET). Volume-50, PP 950-957, February-2009, Published by Waset. ISSN: 2070-3724.

[5] Benkrid, Khaled. "High Performance Reconfigurable Computing: From Applications to Hardware" IAENG International Journal of Computer Science, vol. 35, issue 1. February, 2008.

[6] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," ACM Computing Surveys, vol. 34, no. 2, pp. 171–210, 2002.

[7] Compton K. and Hauck S. "An introduction to reconfigurable computing", IEEE Computer Society, April 2000.

[8] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in Proceedings of the ACM/SIGDA 14th International Symposium on Field-Programmable Gate Arrays (FPGA '06), pp. 21–30, Monterey, Calif, USA, February 2006.

[9] R. Hartenstein. "A Decade of Reconfigurable Computing: A Visionary Retrospective". In Design, Automation, and Test in Europe Conference (DATE), pages 642{649, Munich, Germany, 13-16 March 2001.

[10] A. DeHon, J. Adams, M. DeLorimier, et al., "Design patterns for reconfigurable computing," in Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04), pp. 13–23, Napa Valley, Calif, USA, April 2004.

[11] Katherine Compton, "Reconfiguration Management" in Reconfiguration Computing, (ed.) S. Hauck & A. Dehon, Morgan Kaufman, 2008, pp. 65-86.

[12] R. Hartenstein, "Trends in reconfigurable logic and reconfigurable computing," in Proceedings of the 9th IEEE International Conference on Electronics, Circuits, and Systems (ICECS '02), pp. 801–808, Dubrovnik, Croatia, September 2002.

[13] T. J. Todman, G. A. Constantinides, S. J. E.Wilton,O.Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: architectures and design methods," IEE Proceedings: Computers and Digital Techniques, vol. 152, no. 2, pp. 193–207, 2005.

[14] L. Bauer, M. Shafique, J. Henkel, "A Computation and Communication-Infrastructure for Modular Special Instructions in a Dynamically Reconfigurable Processor"; Proc. of IEEE International Conference on Field Programmable Logic and Applications (FPL 2008), Heidelberg, Germany, pp. 203–208, Sept. 8–10, 2008.

[15] W. Peck, E. Anderson, J. Agron, J. Stevens, F. Baijot, and D. Andrews. Hthreads: A computational model for reconfigurable devices. In 16th International Conference on Field Programmable Logic and Applications, Madrid, Spain, August 2006.