RECONFIGURABLE ARCHITECTURES – THE FUTURE OF THE DIGITAL DESIGN

Blagomir Rosenov Donchev, Anna Severinova Kuncheva

Department of Microelectronics, ECAD Laboratory, FETT, Technical University of Sofia, 8, Kliment Ohridski St, 1000 Sofia, Bulgaria, e-mail: donchev@ecad.tu-sofia.bg;

Reconfigurable Computing is emerging as an important new organizational structure for implementing computations. It combines the post-fabrication programmability of processors with the spatial computational style most commonly employed in hardware designs. The result changes traditional "hardware" and "software" boundaries, providing an opportunity for greater computational capacity and density within a programmable media. This paper is an overview of conception for reconfigurable architectures where is represented advantages of using them and problem with there implementation. At applications point of view a few examples are given.

Keywords: FPGA, CCM, RTR, Partial Reconfiguration, MOLEN

1. INTRODUCTION

System performance is the most important factor that designers must consider while designing their systems. As much as dedicated optimized circuits for specific problem are considered, the system performance increases. The maximum performance can be achieved when the circuits are optimized for only single problem. As a result, the flexibility of changing any parameter of the system is lost. General purpose systems can be built to accommodate this problem and to increase system flexibility by means of software. This reduces the system performance because the hardware has no dedicated resources that can be used by specific algorithms. The configurable computing can achieve new technique that can be optimized for special circuits while keeping system flexibility. Dedicated circuits can be built on FPGAs and these circuits can be easily modified by reprogramming the chips again.

FPGA systems are divided into three main categories:

- *FPGA Emulators* - These systems are built with no prior knowledge of the characteristics of the application that is going to use the system. The system is composed of FPGA chips connected together through *Field Programmable Inter-Connect* chips (FPIC). The main advantage of this model is that larger systems can be implemented but it suffers from high cost and low clock performance.
- *Custom Computing Machines* (CCM) - These machines are built from and array or matrix of FPGA ICs that are connected directly to each others. These FPGAs are located on a single board. These boards have built in memory modules to store the FPGA configurations and the intermediate results. These machines are generally built for specific applications so the inter-connections between FPGAs are optimized to get the best performance while reducing some of the system flexibility.

• *Run-time reconfigurable "RTR" system* - They are used to get best performance with maximum hardware utilization. The circuits are loaded into the hardware and unloaded from it dynamically during the operation of the system. So the area used by such systems will be less than static systems due to unloading idle circuits. RTR systems can be built on single FPGA chip or on multi-IC systems like CCMs. The flow of operation of RTR systems is different from the traditional flow of the general computing systems, where as the circuits configurations are loaded into the system's memory instead of the traditional instructions. These configurations are loaded and unloaded according to the application needs. Refer to fig. 1a,b,c for a small comparison between different systems.

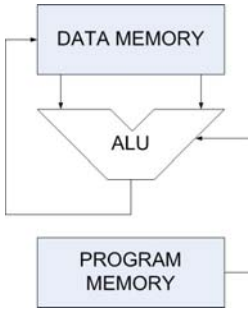

Fig.1a FPGA Emulator

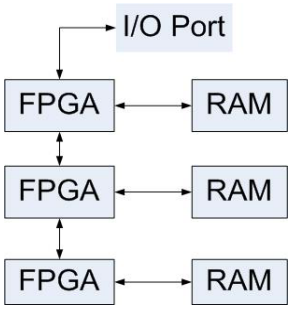

Fig.1b CCM

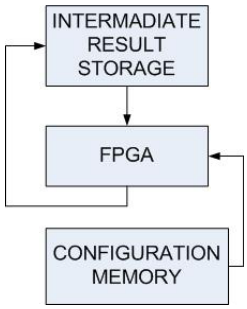

Fig. 1c RTR

2. ADVANTAGE OF RECONFIGURABLE SYSTEMS

- **Performance:** Configurable computing can be highly optimized for specific data set or specific applications.
- **Cost Effectiveness:** Configurable computing can be used to reduce system costs through *hardware reuse*. Any new features or can be easily updated to the system and due to the field programmability, any problems with the design can be corrected without any changes to the on board resources.
- **System prototyping:** Large systems can be built on a single or multiple FPGAs that makes system testing and debugging more easier than testing the real system and even cheaper.
- **Custom I/O:** FPGAs provides flexible set of programmable I/O signals. That gives the designer the opportunity to reuse existing hardware and to add new changes to it easily. New FPGAs supports different types of IOs levels and standards. On important use of this feature is matching different families.
- **System density:** Configurable computing especially "Run-time configurable logic" increases the system density and can deliver functionality of the device many times more than its size by dynamically reconfiguring the system and increase resource utilization through loading and unloading different system modules.
- **Fault-Tolerance:** System reliability can be increased through redundancy by duplication or building some testing circuits.

3. PARTIAL RECONFIGURATION

There are two main implementation approaches for RTR applications, total or partial reconfiguration. In the first method, all FPGA resources are reconfigured or deleted between different configurations. While in the partial reconfiguration approach only the differences between the configurations are modified.

3.1 Total vs. Partial reconfiguration

One of the features of the partially reconfigured FPGAs, is that the size of the bit-stream is proportional to the size of the reconfigured resources. This means that as the size of the bit-stream decreases, the speed of the reconfiguration increases. As a result, the configuration of the partially reconfigured systems is faster than the total reconfigurable systems. The algorithm is divided into time-exclusive segments "time independent partitions" in the totally reconfigured systems. While when the partial reconfiguration is used the it is divided into functional groups that can be loaded or removed independently. The first method of division requires that the partitions must be of equal sizes because each partition will occupy the whole resources of the FPGA. This may increase the difficulty of the partitioning process unless there is well defined partitions in the algorithm. This means that the density of the partially reconfigured systems is increased due to the extra utilization of the idle circuitry.

The total reconfiguration scheme uses external memory to save these information. The other scheme needs less or no memory to store these results because most of their driving circuits remain on the hardware. One of the most advantages of partially reconfigured systems is that the partitions can be located on the FPGA resources during the run-time to provide extra performance and to increase the utilization of the resources that may be reduced due to multiple loading and unloading of hardware modules.

3.2 Partial reconfigurations considerations

Algorithm partitioning is one of the important issues that must be considered while designing an RTR application. In partially reconfigured applications the location of the hardware module "partition" is unknown at the design time so this may cause some problems that may reduce the performance of the system.

Inter-communications and interfacing between hardware modules is more problematic in the partially reconfigured systems than in the totally reconfigured ones. The interfacing between circuits is unknown at the design time and also the sub-circuits need to interact to different sub-circuits dynamically.

Special kind of architectures must be introduced to support the partial re-configuration technique. For example there must be some predictable circuits timing to ensure that the circuits will behave according to the specifications wherever they are located. Also there must an easy interface to the whole internal resources of the FPGA.

3.3 Partial reconfiguration methods

To get the best performance of fully dynamic partial reconfigurable systems, the partitions must be so fine grained so as to reduce the configuration over head and to

increase the system density. It is difficult to partition the system to such small circuits. Instead "semi-static" circuits (the circuit is almost used as is but only small changes can be applied to it) can be used to achieve good performance while maximizing system utilization.

4. RUN TIME RECONFIGURABLE APPLICATIONS

Although RTR applications provides much enhancements over the static designs, but they are still in the research area. RTR is suitable for many applications specially those need much computations and depend on parallel methods such as DSP applications, image processing, neural networks, encryption, compression, pattern matching, motion and target tracking and even general purpose processors.

• *General computing Processor "The MOLEN Polymorphic Processor"* - In this project is presented the general concept of transforming an existing program to one that can be executed on the reconfigurable computing platform we propose and hints to the new mechanisms, intended to improve existing approaches.

The conceptual view of how program P (intended to execute only on the general-purpose processor (GPP) core) is transformed into program P' (executing on both the GPP core and the reconfigurable hardware) is depicted in Fig. 2.

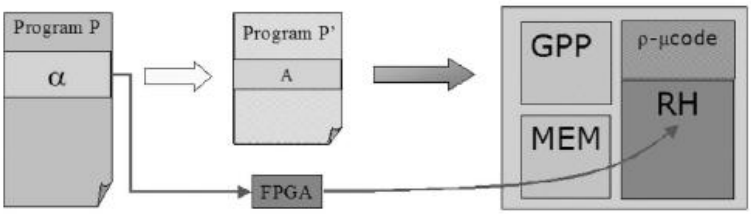

Fig. 2

The purpose is to obtain a functionally equivalent program P' from program P which (using specialized instructions) can initiate both the configuration and execution processes on the reconfigurable hardware. The steps involved in this transformation are the following:

1. Identify code *"a"* in program P to be mapped in reconfigurable hardware.
2. Show that *"a"* can be implemented in hardware in an existing technology, e.g., FPGA, and map *"a"* onto reconfigurable hardware (RH).
3. Eliminate the identified code *"a"* and add "equivalent" code (A) assuming that code A "calls" the hardware with functionality *"a."* Code A is comprised of the following:
   - Repair code inserted to communicate parameters and results to/from the reconfigurable hardware from/to the general-purpose processor core.
   - "HDL"-like hardware code and emulation code inserted to configure the reconfigurable hardware and to perform the functionality that is initialized by the "execute code".
4. Compile and execute program P' with original code plus code having functionality A (equivalent to functionality "a") on the GPP/reconfigurable processor.

The mentioned steps illustrate the need for a new programming paradigm in which both software and hardware descriptions are present in the same program. It should also be noted that the only constraint on *"a"* is implementability, which possibly implies complex hardware. Consequently, the microarchitecture may have to support emulation via microcode [3]. This reconfigurable microcode (p/u-code) is different from the traditional microcode. The difference is that such a microcode does not execute on fixed hardware facilities. It operates on facilities that the p/u-code itself "designs" to operate upon. The methodology of the transformation described previously for the reconfigurable computing platform is depicted in Fig. 3. First, the code to be executed on the reconfigurable hardware must be determined. This is achieved by high-level to high-level instrumentation and benchmarking. This results in several candidate pieces of code. Second, we must determine which piece of code is suitable for implementation on the reconfigurable hardware. The suitability is solely determined by whether the piece of code is implementable (i.e., "fits in hardware"). Those parts can then be mapped into hardware via a hardware description language (HDL). In case the HDL corresponds to "critical" hardware in terms of, for instance, area, performance, memory, and power consumption, the translation will be done manually (see Fig. 3). Otherwise, the translation can be done automatically or be extracted from a library.

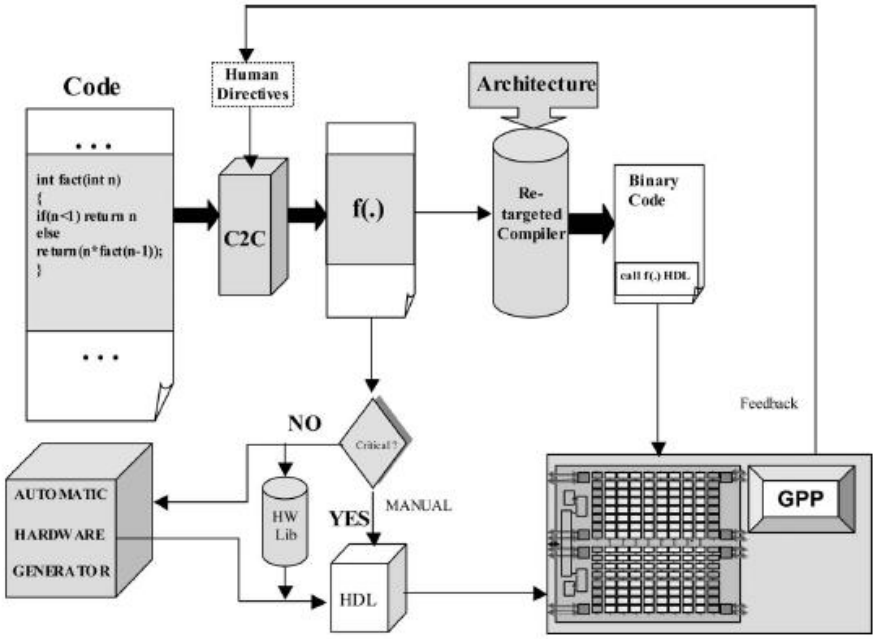

Fig. 3

• *Automatic Target Recognition "ATR"*- The challenge to the ATR is the rapid comparison of an input image to thousands of templates with large number of pixels per image. Dedicated hardware for special templates can be used to provide higher performance but without the flexibility of changing the templates. RTR logic can be used to combine between the advantages of the both approaches by building custom circuits optimized to each template and reconfigured on the fly to different templates.

• *Data encryption:* The Data Encryption Standard "DES" algorithm showed good performance when implemented by RTR logic. When the receiver gets the key of the sender, the system reconfigure the hardware to the new key and decodes the incoming messages. At the end, the system removes this key waiting for a new key. In this case, we can get maximum performance of the system for any key.

4. CONCLUSION

There are three major factors that limits the evolution of the RTR applications:

•FPGA gate capacity is some how small compared to the ASICs. This increases the complexity of the system because designers have to use more than one FPGA and they have to partition their designs be tween the on-board FPGAs.

• Configuration speed is the most important factor that limits the number of RTR applications. New FPGA architectures must be developed to reduce the configuration overhead.

• RTR applications need highly skilled developers who can deal with the low level design, so these applications are developed in laboratories under the supervision of qualified designers. The only solution to this problem is to develop CAD tools that can reduce the design time.

• Benchmarks for RTR applications is one of the important issues that must be introduced to enable designers to discuss the performance of their designs versus the performance of other designs techniques.

To prevent this limitations an extra features must be added to the FPGAs to increase their usability, such as:

• On-chip memory must be increased.

• Special-purpose blocks such as Arithmetic units are needed to increase FPGA usability in the arithmetic computation.

• Memory and micro-processors interfacing support must be included in the FPGA chips to reduce the overhead needed to interface to these devices.

• Enhanced partial reconfigurable FPGA must be built to reduce the configuration time.

To conclude, reconfigurable logic will become more considerable for real world applications after high speed configuration FPGA architectures are developed with intelligent software that ease both the system use and design.

ACKNOWLEDMENTS

This research is supported by, program "Young scientists" of the Bulgarian Ministry of Education and Science.

5. REFERENCES

[1] Khatib J., *Configurable Computing*, 1998.

[2] Vassiliadis S., S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov, E. Moscu Panainte, *The MOLEN Polymorphic Processor basic Electronics*, IEEE Transactions On Computers, November 2004, Vol. 53, No. 11, pp 1363-1375

[3] Vassiliadis S., S. Wong, S. Cotofana, *Microcode Processing: Positioning and Directions*, IEEE computer Society, July-August 2003, pp 21-30.