A Hardware Operating System based Approach for Run-time Reconfigurable Platform of Embedded Devices

Krishnamoorthy Baskaran and Thambipillai Srikanthan
Centre for High Performance Embedded Systems (CHiPES),
School of Computer Engineering,
Nanyang Technology University, Singapore.
{asbaskaran, astsrikan}@ntu.edu.sg

Abstract

Runtime reconfigurable embedded systems target an architecture consisting of a reconfigurable system-on-chip (RCSoC), which has hardcore or softcore general-purpose processor (GPP) and field programmable gate arrays (FPGAs). The architecture provides common execution semantics for software and hardware tasks. The partial reconfiguration abilities of the newest FPGAs are fully exploited in order to improve performance, cost, energy-efficiency, and time-to-market

In this paper, we describe the research issues and technology approaches for the development of a hardware operating system for runtime reconfigurable platform of embedded devices. We propose a specialized operating system, that will manage hardware/software task management, to allow run-time partitioning and allocation of reconfigurable FPGA area, efficient communication techniques and well-organized task scheduling methodology.

1 Introduction

Runtime reconfiguration of FPGAs is one of the widely researched areas in the field of computer architecture over the last decade. The reconfigurable platform based design makes use of novel technologies for improving performance, cost, energy-efficiency, and time-to-market. The embedded devices employ an FPGA that has the ability to be reconfigured in runtime to achieve the above-mentioned objectives. The latest FPGAs have several millions of gates, partial reconfiguration and readback features.

A run-time reconfigurable platform system would be able to decide - on the fly - what hardware to execute on the FPGA area and what software to run on the CPU. The use of reconfigurable system is continuously growing in the areas like wearable computing [1], mobile communication system [2], video communications [3], neural computing, and network processors.

The runtime reconfigurable platform has the following benefits.

- *Faster execution*: FPGA based implementations algorithms are much faster than software algorithms implemented on a general-purpose computer.
- *Low cost:* A reconfigurable computing system is much cheaper for each new application than an ASIC.
- *Area Utilization:* Reconfigurable approach also allows for full hardware utilization. Unused space on an FPGA can be dynamically allocated to another task, thus increasing performance of multiple tasks.
- *Low power, foot print:* One reconfigurable computing system can take over the nonconcurring functions of several dedicated, special-purpose peripherals, reducing the size and power consumption of the embedded system.

Run-time reconfiguration is the ability to rapidly change the functionality of an FPGA during the execution of a hardware task. The reconfigurable hardware can be used to execute designs, which are larger than the available hardware resources. The Hardware operating system for reconfigurable platform is a new line of research, which supports all the functionality of the real-time operating system and is extended to manage reconfigurable platform resources.

In this work we use a reconfigurable computing system composed of an embedded processor core running the software part of the hardware operating system, connected to an FPGA containing a group of

partitioned reconfigurable blocks, which can be individually reconfigured at run time to realize hardware circuits. We present the design and the implementation of the hardware operating system for runtime reconfigurable platform.

The rest of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 lists the issues and the requirements of an operating system for reconfigurable platforms. Section 4 attempts to address the technology approaches and describes the prototype implementation. Section 5 concludes the paper.

2 Related Work

Among numerous reconfigurable architectures, an architecture with single processor and reconfigurable resources (e.g. FPGA) has been widely studied and commercialized [14, 15]. In our work, we plan to use a processor/FPGA architecture to implement the runtime reconfigurable platform specification. With numerous resources and partial reconfiguration functionality, an FPGA is a perfect target for hardware-based operating system. A hardware operating system executes a set of hardware tasks on a reconfigurable platform in a parallel multitasking manner.

Brebner addressed the Virtual hardware operating system [17]. He explores some of the fundamental issues that will influence the construction of any operating system for FPGAs with dynamic reconfiguration. He proposes that application be designed into relocatable cores known as swappable logic units (SLU) [18]. Mignolet et al. [20] introduce relocatable tasks, which can be executed either in software or hardware, depending on the available resources and the performance required. Simmler addressed multitasking and task preemption on FPGA in [19]. Wigley et al discussed operating system functionalities like partitioning, placement and routing [16]. Previous research work has exposed various advantages of implementing a Real Time Operating System (RTOS) in hardware [21, 22]. Many research issues have not been addressed yet, so it is very difficult to compare the existing work as well as implementations aspect.

3 Issues In Run-Time Reconfigurable Architecture

3.1 Hardware/Software Codesign Issues

Recently, embedded processors integrated on the FPGA chip have introduced a new boundary for hardware/software co-design with software running on the embedded core as well as hardware tasks running on the FPGA. The computational sequence of tasks in an embedded device is represented by task flow graphs (TFG). For many applications, these tasks can be distributed both as hardware area as well as software resources. Resources in hardware side can be Application Specific Integrated Circuits (ASIC) or reconfigurable logic device like FPGA. ASICs are not suitable for runtime reconfigurable platform because of the lack of reconfiguration function and very large design times. The FPGA devices play a major role in reconfiguration to utilize hardware resources on account of their run-time reconfiguration functionalities. In this platform, is commonly used for implementing software operation a general-purpose processor (GPP). The architecture for runtime reconfigurable platform consists of an embedded processor system for software operations, reconfigurable logic resources for hardware functions integrated in an FPGA and shared memory as illustrated in Figure 1.

In the proposed run-time reconfigurable platform, we describe the fundamental difference between the hardware and software resources as follows. The tasks in the software process are executed in a sequential way. In the FPGA, the hardware tasks can be executed concurrently. At run-time the hardware tasks take time to configure the FPGA resources. This is not applicable to software operations. The number of tasks executed in the FPGA resources depends on the number of hardware units that can be configured on the device at run-time. The execution of the hardware tasks is faster compared to the software tasks. The power consumption of the software resources is much less than hardware and there is no concurrency in software.

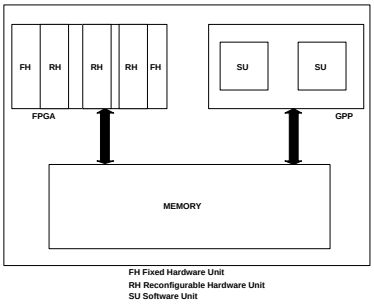

FIGURE 1: *Codesign Issues*

3.2 Platform issues

3.2.1 Operating System Level Issues

In the run-time reconfigurable platform, we intend to develop an RTOS-like system known as Hardware

OPErating System [HOPES], which will manage the different hardware/software task, FPGA area management, memory handling, and placement loader running on the platform. In the run-time reconfigurable platform, each hardware task is a circuit, i.e., a task can consist of a combination of logic circuits and memory. This task is loaded and executed in an FPGA area. Our proposed operating system performs the following functions:

- Partitioning the reconfigurable hardware area depending upon the task size
- Placement of tasks in the configurable area.
- Scheduling of the tasks
- Loading, executing and removing tasks

3.2.2 Partitioning Issues:

The latest commercial FPGAs have several millions of reconfigurable gates, and are capable of running with clock speeds in the hundreds of MHz range. The FPGAs can be configured in two ways: complete and partial reconfiguration [4]. Partial reconfiguration is the ability to reconfigure a portion of the FPGA while the remainder of the design is still in operation. This configuration method is useful for applications that require the loading of different designs into the same area of the device. To make use of this technology, we need to handle the partition issues.

In the runtime reconfigurable platform, a hardware resource is partitioned into blocks. While the vertical dimension of each block is fixed, the horizontal dimensions can be varied. Each partitioned block accommodates one task at a time. In order to obtain good resource utilization and to reduce configuration times, the partition size of the block should be based on the hardware task size. When we partition the FPGA fabric, we consider the configuration and readback time. The hardware tasks require the blocks to be configured before they are executed in the available hardware blocks. The configuration time various depending upon the block size. Figure 2 shows the TGF application and the partitioned Reconfigurable FPGA. Each task in the TGF is instantiated through partial reconfiguration and its instantiation will not affect the other existing tasks. In Figure 2, tasks D, E, and F, can be instantiated by reconfiguring the partitioned hardware resources. This operation will not affect the configured existing tasks B and C.

We are currently in the process of developing an efficient partitioning algorithm for runtime reconfigurable platforms to achieve a better hardware resources utilization.

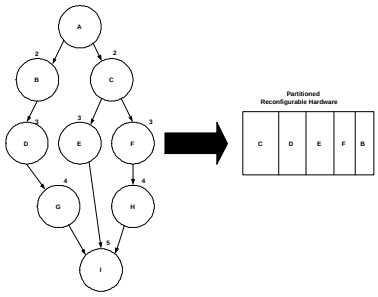

FIGURE 2: *Executing TGF On A Partitioned Reconfigurable Hardware*

3.2.3 Scheduling Issues

Task scheduling is an important problem in HOPES. A scheduler is a module that allocates processor and resources to various tasks. Tasks are scheduled and allocated resources according to a chosen set of scheduling algorithms and the availability of resources. Scheduling hardware tasks is different from ordinary software task scheduling because in the case of hardware task, there is a need to reconfigure the partitioned FPGA area. A good hardware scheduler satisfies the following conditions:

- One hardware block is assigned one hardware task at any given instant
- Hardware task size should be less than or equal to the size of the block to which it is assigned.
- No hardware task is scheduled before its release time
- Depending on the scheduler, the configuration and readback time is added to the task execution time
- Placement handler handles the placement constraints.
- Resource utilization constraints should also be satisfied

3.2.4 Communication Issues

The next major issue in HOPES is inter-process communication. In reconfigurable platform, there are different hardware and software mechanisms that could communicate with each other. The user applications running under the HOPES and the HOPES itself are implemented in hardware, so we need standard communication interface between hardware (FPGA) and software (CPU). The hardware tasks

are stored in external memory in the form of configuration bitstreams. Interface between external memory and the FPGA is one of the communication issues. The hardware inter-task communication is also one of the major issues.

3.3 Target Architecture for reconfigurable computing

The proposed reconfigurable computing target architecture is shown in Fig. 3. The architecture consists of an embedded processor running HOPES. Reconfigurable hardware blocks, memory and application specific hardware modules as well as reconfiguration manager are attached to the communication block. The ReConfigurable System-on-Chip (RCSoC) is used for this platform. It consists of one soft-core CPU, HOPES (Hardware OPErating System), external memory, and communication bus. The embedded processor is implemented in the RCSoC as a fixed (non-reconfigurable) core. HOPES manages the RTOS facilities, hardware task allocation, and the placement engine. The communication block serves to communicate among the hardware blocks, HOPES, and the embedded processor.

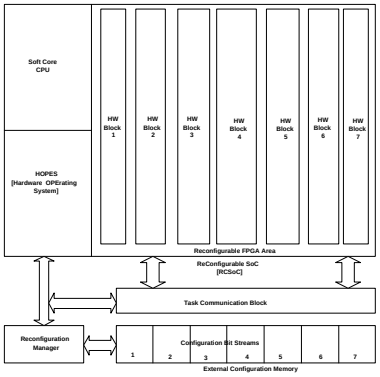

FIGURE 3: *Proposed Run Time Reconfigurable Architecture*

4 Technology approaches

4.1 ReConfigurable System-on-Chip (RCSoC)

ReConfigurable System-on-Chip (RCSoC) has become the reality now, driven by fast development of CMOS VLSI technologies. The RCSoCs are sometimes called platform FPGAs. The basic concept for RCSoC is using reconfigurable resources along with a conventional microprocessor. The main goal is to take advantage of the capabilities and features of both resources. ReConfigurable SoCs (RCSoC), consist of processor, memory, and on-chip reconfigurable hardware parts for customization to a particular application. The RCSoC device consists of an FPGA that incorporates a CPU located in the fixed logic area. The hardware tasks are executed in the reconfiguable logic area, so that CPU is not interrupted. The CPUs used in the reconfigurable computing systems can be characterized as either hard cores or soft cores. Hard cores are designed into the FPGA fabric by the vendor as custom VLSI layouts and are connected to the FPGA through predefined wiring. Soft cores are designed like any other FPGA-based circuit. For the proposed run-time reconfigurable platform, we are planning the use of a soft core CPU.

4.2 Soft Core CPU

The Soft Core CPU architecture provides the designer with the ability to customize the CPU architecture. There are several soft CPU cores currently available that can meet the performance demands of real time applications. Commercial CPU cores such as the NIOS [6] and Microblaze [7] are FPGA vendor supported and hence cannot be ported into other FPGA platforms. For our runtime reconfigurable platform, we have chosen to use an open source CPU that is capable of supporting future enhancements .The CPU needs a supporting assembler, simulator and C compiler. The LEON SPARC Processor in VHDL [8] and Opencores OR1200 in Verilog [9] are both open source CPUs. We have chosen to use the OR1200 as it fares better in terms of performance and FPGA resource requirement.

4.3 Hardware OPErating System (HOPES)

The reconfigurable hardware operating system is a new line of research and involves several challenges. HOPES manages the available hardware resources of the node, i.e., CPU, reconfigurable hardware unit, I/O, and memory. The main function of HOPES is to manage tasks. Generally, an application will contain a combination of tasks. While software tasks can be run on the CPU, hardware tasks execute on the reconfigurable hardware area. HOPES should keep track of available resources, meaning to find the location for incoming hardware task.

We propose a HOPES kernel running on a CPU within an FPGA. The rest of the resources on this RCSoC are available for custom computing circuits that can be reconfigured at run-time. The CPU has access to memory on the FPGA as well as an interface to larger off-chip memory. The CPU will have an off-chip communication interface.

4.4 Partitioning & Placement Technology

In runtime reconfigurable computing, we make use of task based partitioning method [10] for partitioning the reconfigurable hardware FPGA area. A hardware design unit is called a task; it is executable on the partitioned FPGA. The FPGA fabric is partitioned into blocks. The blocks are static with different widths, which allows running the hardware tasks. The latest FPGA devices from Xilinx [11] have partial reconfiguration functionality. To make use of this technique we need to partition the FPGA surface depending upon the hardware task size.

The next major step is to place the hardware task into the partitioned hardware surface. We have implemented the new placement handler in our platform. The placement handler determines where to place each hardware task on the FPGA.

4.5 Scheduling Technology

Scheduling is one of the primary issues in runtime reconfigurable computing. Partitioning and scheduling are known to be complex optimization problems [12]. The task scheduler decides which task has to be executed next, among all tasks ready to execute. Scheduling can be classified into offline and online. The proposed architecture is based on online scheduling; we develop non-preemptive and preemptive schedulers for hardware tasks. In non-preemptive scheduling, the currently executing task will not be interrupted until it decides on its own to release the allocated resources, normally after completion. For non-preemptive scheduler, we take the configuration time into consideration while scheduling the task. In preemptive scheduling, the currently executing task may be preempted, i.e., interrupted, if a more urgent task requests service. The preemptive scheduler takes configuration time and readback time into consideration while scheduling task.

4.6 Target Technology and Platform

We have chosen the Celoxica RC200 board [13] as the implementation platform. This board integrates a Virtex XC2V2000 SRAM based FPGA and a variety of different I/O devices, i.e., Ethernet, Audio, Video, SmartMedia, Parallel port, RS-232 and PS/2 keyboard and mouse. The Opencores OR1200 soft-core CPU is installed in the FPGA fabric as fixed logic. Xilinx ISE Foundation 5.1 in combination with the Modular Design Package [14] serves as the development environment for circuit/bitstream generation; all PC software is created with C or C++.

5 Conclusion

In this paper, we have examined research issues and technology approaches for run-time reconfigurable platforms. We then proposed an environment that describes operating system services and services in a hardware device. We also list a set of approaches to runtime reconfigurable computing. We want to have a proof of approaches for three main topics. The first issue to be tackled is the infrastructure that allows multitasking on an FPGA. For this we have to exploit the partial reconfiguration abilities of the newest FPGAs. Next, a good partitioning algorithm for FPGA is required. The third issue is the development of HOPES (Hardware OPErating System) that can manage hardware and software constraints.

References

[1] C. Plessl et al., "Reconfigurable Hardware in Wearable Computing Nodes. In Proceedings of the 6th International Symposium on Wearable", Computers (ISWC), pages 215-222. IEEE Computer Society, October 2002.

[2] IMEC Interuniversity Micro Electronic Center, T-ReCS Gecko, http://www.imec.be.

[3] J.Villasenor, C.Jones, and B.Schoner., "Video communication using rapidly reconfigurable hardware", IEEE Trans. Circuits Syst. Video Technol., 1995,5, (6), pp.565-567

[4] Xilinx Inc., Vertex Series Configuration Architecture User Guide, Application Note, http://www.xilinx.com/bvdocs/appnotes/xapp151.pdf

[5] Altera M-DS-EXCNIOS-01 "Nios Soft Core Embedded Processor Data Sheet", http://www.altera.com June 2000.

[6] Xilinx, "MicroBlaze Hardware Reference Guide", http://www.xilinx.com Mar. 2002.

[7] J. Gaisler, "LEON SPARC Processor", http://www.gaisler.com/leonmain.html, June 2001.

[8] D. Lampret, "Open RISC 1000 Project", http://www.opencores.org/projects/or1k/, Dec. 2002.

[9] Pedro Merino, Margarida Jacome, and Juan Carlos Lopez. "A Methodology for Task Based Partitioning and Scheduling of Dynamically Reconfigurable Systems", In Proceedings of the

IEEE Symopsium on FPGAs for Custom Computing Machines (FCCM), pages 324-325. IEEE CS Press, April 1998.

[10] Xilinx Inc., "Two Flows for Partial Reconfiguration: Module Based or Small Bit Manipulations", Application Note, http://www.xilinx.com/bvdocs/appnotes/xapp290.pdf

[11] J.W.S.Liu, "Real-Time Systems". Prentice Hall, 2001.

[12] Celoxica Limited, Celoxica RC200 Development Board, http://www.celoxica.com.

[13] Xilinx Inc., Advanced Design Techniques, Modular Design, http://www.xilinx.com

[14] Xilinx Inc., Virtex II Pro Series http://www.xilinx.com.

[15] Chameleon Systems, Inc. CS2000 Reconfigurable Communications Processor Family Product Brief, San Jose, CA, 2000.

[16] G. Wigley and D. Kearney. "Research Issues in Operating Systems for Reconfigurable Computing", In proceedings of the International Conference on Engineering of Reconfigurable System and Algorithms (ERSA), pp 10-16. CSREA Press, Junie2002.

[17] G. Brebner, "A Virtual Hardware Operating System for the Xilinx XC6200", In proceedings of the 6th International Workshop on Field Programmable Logic and Applications (FPL), pp. 327-336. Springer, 1996.

[18] G. Brebner and O. Diessel,"Chip-Based Reconfigurable Task Management", In Proceedings of the 11th International Workshop on Field Programmable Gate Arrays (FPL), pp 182-191. Springer, 2001.

[19] H. Simmler, L. Levinson, and R. Manner,"Multitasking on FPGA Coprocessors," In proceedings of the 10th International Workshop on Field Programmable Gate Arrays (FPL), pp 121-130. Springer, 2000.

[20] J.-Y. Mignolet et al,"Infrastructure for Design and Management of Relocatable Tasks in a Heterogeneous Reconfigurable System-on-Chip," In proceedings of Design, Automation and Test in Europe (DATE), pp 986-991. IEEE Computer Society, March 2003.

[21] J.Adomat et al, "Real-Time Kernel in Hardware RTU: A Step Towards Deterministic and High-Performance Real-Time Systems," Proceedings of EURWRTS '96, PP. 164-168, June 1996.

[22] T,Nakano et al," VLSI implementation of a Real Time Operating System," Proc. Of ASP-DAC '97, PP.679-680, January 1997.