A New Project to Address Run-Time Reconfigurable Hardware Systems

Jim Torresen and Dirk Koch

Department of Informatics, University of Oslo
P.O. Box 1080 Blindern, N-0316 Oslo, Norway
E-mail: {jimtoer,koch}@ifi.uio.no
Web: http://www.matnat.uio.no/forskning/prosjekter/crc/

Abstract. This paper introduces a new project named Context Switching Reconfigurable Hardware for Communication Systems (COSRECOS) which was started autumn 2009. In this project, reconfigurable hardware technology (Field Programmable Gate Arrays - FPGAs) will be applied for designing high performance computing systems for embedded communication systems. The overall goal of the project is to contribute in making run-time reconfigurable systems more feasible in general. This includes introducing architectures for reducing reconfiguration time as well as undertaking tool development. Case studies by applications in network and communication systems will be a part of the project. The paper describes how we plan to address the challenge of changing hardware configurations while a system is in operation.

1   Introduction

Before the introduction of multitasking operating systems around 1985, processors would run one program at a time. The program would be uploaded at startup and run until finished. There would be no swapping to other programs during execution of a given program. With today's multitasking operating systems, it would be the exception not performing multitasking for software. This is in contrast to hardware which normally is static at run-time even though reconfigurable hardware is programmable at run-time. However, in this project – called *Context Switching Reconfigurable Hardware for Communication Systems (COSRECOS)* and funded by the Research Council of Norway, architectures where the hardware configuration is dynamically changed (i.e. context switching) will be investigated. The main contributions of the project are expected to be:

– Develop new architectures and tools to make run-time reconfiguration easier to use.
– Develop platforms for fast and reliable reconfiguration.
– Undertake case studies with implementation of selected communication applications.

We regard that both architectures as well as tools would be needed to make run-time reconfigurable hardware more applicable. The next section gives some background information, followed by an overview of benefits and possible approaches in section 3. Finally, section 4 concludes the paper.

2   Background

Reconfigurable computing has grown to become an important and large field of research. The present target technology is Field Programmable Gate Arrays (FPGAs). The FPGA provides a massive parallel computational engine by the distributed organization of its logic blocks. Each logic block consists of configurable combinational logic together with one or a few flip-flops. The configuration of the function of each logic block and its connections to other blocks are given by the *configuration bitstream* loaded from outside the device and stored in distributed RAM on the FPGA. It is substituting the configuration bitstream at *run-time* we think of by context switching reconfigurable hardware – see details below.

Reconfigurable systems are designed either by using commercial FPGAs or by having new FPGA-like devices developed. A survey of can be found in [1]. The approach of applying reconfigurable logic for data processing has been demonstrated for a number of years ago in areas such as video transmission, image recognition and various pattern-matching operations (handwriting recognition, face identification) [2]. Another area of interest is wireless systems where tremendous computational capabilities are needed to allow for high data rates in the future [3]. Automotive electronic systems are expected to gain large benefit from adaptive reconfigurable hardware [4].

A platform for network applications – called The Field-programmable Port Extender (FPX) (see http://www.arl.wustl.edu/projects/fpx/), has been implemented by a group led by professor Lockwood. It is a generic platform with network interfaces that has been used in a wide variety of applications: route Internet packets; compress, encrypt, and buffer data; transcode motion JPEG images; and process multiple flows of video. By using FPGA hardware, rather than a microprocessor, the packet processor can perform full processing of packet payloads at Gigabit rates. The development of the FPX platform demonstrates a valuable use of FPGA technology in routers and other network equipment. The hardware of the system will evolve over time as packet processing algorithms and protocols progress, as stated in [5].

For bus-based communication, an architecture called ReCoBus has been proposed that efficiently supports the integration of dozens of partial modules into a reconfigurable systems that can provide up to a few hundreds of resource slots [6]. In addition, I/O bars have been proposed for dedicated high speed streaming links between the static system and the reconfigurable modules [7].

We have been conducting research on several different application areas of reconfigurable logic. An architecture for providing a fast (in the Gigabit range) network security system (called Network Intrusion Detection Systems (NIDS))

has been proposed. The detection engine (rule matching) in the open source network intrusion detection system Snort has been implemented [8]. Further, stateful inspection is applied in NIDS in [9]. By checking the handshakes in a communication session, it provides a more advanced network security tool than firewalls.

We have demonstrated the benefits of undertaking data processing in reconfigurable logic for applications like string matching [10] and image filtering [11]. An FPGA implemented processor architecture with adaptive resolution has been introduced in [12]. This allows for a variable resolution in data variables at run-time.

3   FPGA Based Architectures

This section includes a brief overview of how FPGAs can be applied in adaptive systems [13]. We can distinguish between three different degrees of FPGA reconfiguration providing configurable computing:

- **Static:** The configuration within the FPGA is the same throughout the lifetime of the system. This means no adaptivity at run-time.
- **Upgrade:** The configuration is changed from time to time for bug fixes or functional upgrades. This represents rare reconfiguration.
- **Run-time:** A set of configurations (multi-context) are available which the FPGA switch between at run-time. This could provide several benefits as described below.

Most applications are implemented by applying the *static* approach – i.e. no reconfiguration. However, *upgrading* of systems have recently become more common. This allows the configuration to be upgraded when bugs are found *or* when the functionality of the system is to be changed. In the future, automatic dynamic products will probably arrive. These could *autonomously* upgrade the hardware as the environment (or data) changes or when bugs are detected in the system. One promising approach based on this idea is evolvable hardware [14].

The application areas for *run-time* reconfigurable systems are:

- Space/cost/power reduction
- Speeding up computation
- Incorporating new data/patterns realized in reconfigurable logic

If not all functions in a system are needed at the same time, we can substitute a part of the configuration at run-time as seen in Figure 1. Function A contains the parts of the system that always need to be present. However, part B and C are not needed concurrently and can be assigned to the *same* resources (location) in the FPGA. An example of such an application can be a multi-functional handheld device with e.g. mobile phone, MP3 player, radio and camera. For most purposes, a user would normally not apply more than one of these functions at a time. Thus, instead of having custom hardware for

each function, it could be efficient having a reconfigurable system where only the *active* function is configured. This would allow for a smaller hardware device which leads to reduced cost and for some systems reduced power consumption. Such benefits are important in a competitive market.
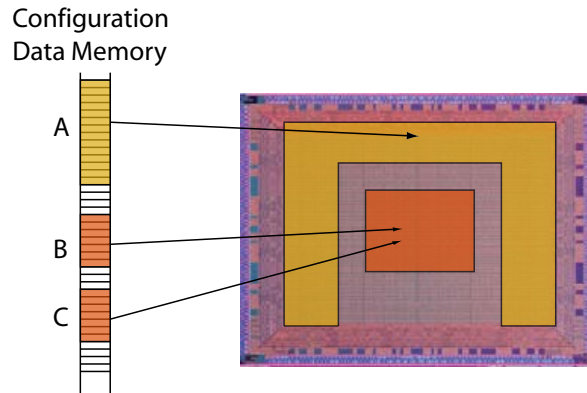
Fig. 1. Illustration of a run-time reconfiguration of FPGA.

The application area for run-time reconfiguration for computational *speedup* is depicted in Figure 2. Swapping between successive configurations can give a hardware system a considerable throughput compared to having a general *static* FPGA configuration. If a task A can be partitioned into a set of separate subtasks (A.1, A.2 and A.3 in the example in the figure) to be executed one after the other, an FPGA configuration can be designed for each of them. Thus, each configuration is optimized for one *part* of the computation. During run-time, context switching is undertaken and the total execution time for the task in the given example is reduced. The context switching time would have to be short compared to the computation time, to reduce the overhead of switching between the different configurations.

Since commercially available FPGAs do not yet provide configuration switching in one or a few clock cycles, download time is often the main obstacle against effective run-time reconfiguration. There has been undertaken some work based on run-time reconfiguration of FPGAs. The main experience seems to be that FPGAs are requiring a (too) long reconfiguration time, and we regard this as the key challenge when designing context switching systems. A task should be partitioned into *coarse* grained parts to reduce the overhead of switching between different configurations [15]. Many devices require the *complete* configuration bitstream to be downloaded in one operation. The download time then increases

4
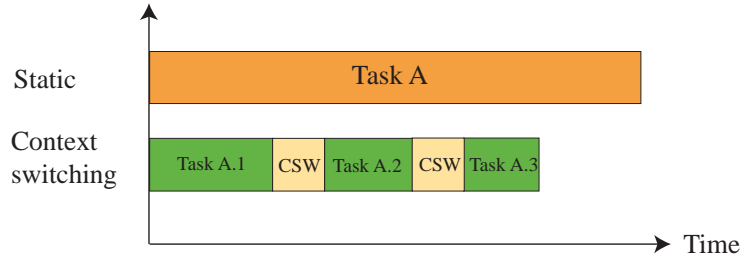
Fig. 2. Illustration of a run-time reconfigurable FPGA compared to a static FPGA.

with the size of the device. The challenge with long download time is further addressed in the next section.

3.1 Approaches and Methods

The project will focus on research for developing new architectures that can reduce the problems of applying commercial FPGA technology to run-time reconfigurable computing. Rather than focusing on FPGA only as a tool for speeding up computation – as in many research projects, we will also emphasize on switching configurations at run-time. That is, replacing parts of the user logic inside the FPGA while other parts operate uninterrupted. By this approach, we would like to have a focus on both reduced cost and power consumption as well as additional computational speedup. However, some architectures would be limited to one of these priorities. By developing new architectures and algorithms, we will try to come up with systems making run-time reconfigurable hardware more usable. Having several leading international research groups as collaborators in this project will be a great strength for achieving these goals.

Challenges of run-time reconfiguration in FPGA to be addressed in the project are as follows:

- Reducing the long time required for reconfiguration
- Avoiding the system from being inactive during reconfiguration (safe and robust reconfiguration)
- Interfacing between modules belonging to different configurations
- Predictability (reliability and testability) of system operation

As introduced earlier, the main problem with switching configurations is the long reconfiguration time. Overcoming this would be one of the main objectives in the project. At the moment there seem to be three possible approaches; smaller devices, virtual FPGA and partial reconfiguration.

Smaller Devices Since the full reconfiguration time is less for smaller devices, reconfiguration time can be reduced by applying smaller devices. Moreover, by applying context switching, we may be able to implement a full system in a smaller device with the benefit of reduced cost and power consumption. The drawback would be that the system would have to be inactive during reconfiguration. In this project, we would like to include some analyzis on these aspects.
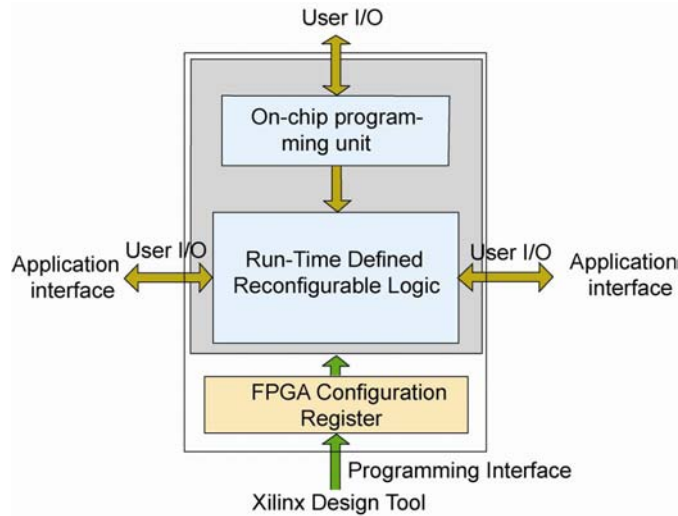

Fig. 3. Virtual FPGA.

Virtual FPGA Virtual FPGA is based on designing a multi-context "virtual" FPGA inside an ordinary FPGA [16] – see Figure 3. We have earlier introduced an architecture for context switching based on this idea that has been published in [17, 18]. In these papers, we report about our design of an architecture for switching between 16 different configurations in a *single* clock cycle. Such a system would never achieve as high clock frequency as a leading edge processor. However, by applying massive parallel processing, the execution time can still be less [12]. We have published a number of papers where virtual FPGA is combined with evolvable hardware [19–22, 13]. The developed architectures also include a soft (MicroBlaze) or hard (PowerPC) processor core. Even though a fast processing can be achieved, the context switching architecture requires much reconfigurable resources (in that way, this architecture is prioritizing speed before cost and power consumption). Therefore, there is still a large potential for improving these systems and this would be one of the challenges in this project.

Partial Reconfiguration As FPGA devices are getting bigger, the configuration bitstream becomes longer and programming time increases. Thus, runtime reconfigurable designs would benefit from having only a limited part of the FPGA being context switched by *partial reconfiguration*. This feature is available in some FPGAs where a selected number of neighboring columns are programmed. This requires detailed considerations for having no interruption at context switching [23].

Another challenge is to limit the inter partition data transfer. That is, efficient communication between context switched tasks. While the first FPGAs offering partial reconfiguration required *complete columns* of the device being programmed, the more recent ones – including Xilinx Virtex-4/5, require only a *part* of each column being programmed. This makes interfacing between tasks and having uninterrupted operation easier since some rows can be used for permanent configurations. The smallest Virtex-5 device (LX30) consists of 4 rows while the largest (LX330) consists of 12 rows. Further, there have been introduced tools like PlanAhead that make partial reconfiguration easier. It is possible to reconfigure the Virtex devices *internally* using the Internal Configuration Access Port (ICAP). This will be applied in this project where research will be undertaken on efficient data routing and data storing between context switching tasks [24]. We have already undertaken various successful work with partial reconfiguration including change of look-up table content [25] and internal reconfiguration with ICAP by use of PlanAhead [26].

Platform and Tools The work at the moment on reconfigurable systems are usually based on problem specific coding. Thus, a goal of this project is to come up with a tool and some general platforms that could make context switching systems more accessible for a larger number of users. That is, develop a general tool and architectures that can be applied by others in future run-time adaptable systems.

If the context switching is *not* deterministic, an operating system may be needed to schedule hardware tasks [27]. However, for many of the applications, it would be possible with a deterministic context switching. This would be our first approach since online scheduling of tasks including control of reconfigurable logic fragmentation would be much more difficult. However, to have more general computing platforms, this would become more necessary. We believe a general computing platform will make reconfigurable technology more accessible than it is today, and make it into a viable complement to processor technology. The success of multitasking in software is probably much because of the introduction of widely used operating systems like Windows and Linux. If an equivalent could be found across FPGA vendor technologies by e.g. virtual FPGA, it would probably be an important step towards more widely use of run-time reconfigurable hardware.

A part of the research will be on analyzing HW/SW partitioning and how this can be undertaken in the most efficient way. Much research related to communication technology is *either* related to implementing software *or* hardware.

However, few projects are concerned about the *integration* of application software, low level software and hardware. In the industry on the other hand, much focus is given to this integration. Thus, we would in this project like to address how hardware should be designed to most effectively execute the software to be implemented. More details about the project can be found in [28].

4   Conclusion

This paper has described a new project focusing on context switching reconfigurable hardware. It will target to make such technology more applicable by introducing software as well as hardware architectures to make it more feasible. Challenges include addressing reconfiguration time and making the context switching robust.

Acknowledgment

References

1. R. Hartenstein. A decade of reconfigurable computing: A visionary retrospective. In *Proc. of Int. Conference on Design Automation and Testing in Europe - and Exhibit (DATE)*, 2000.
2. J. Villasenor and W.H. Mangione-Smith. Configurable computing. *Scientific American*, (6), 1997.
3. J.M. Rabaey. Silicon platforms for the next generation wireless systems - What role does reconfigurable hardware play?. In R.W. Hartenstein et al., editors, *10th International Conference on Field Programmable Logic and Applications (FPL-2000)*, Lecture Notes in Computer Science, vol. 1896, pages 277–285. Springer-Verlag, 2000.
4. J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka. Dynamic and partial fpga exploitation. *Proceedings of the IEEE*, 95(2):438–452, Feb 2007.
5. J.W. Lockwood. Evolvable internet hardware platforms. In *Proc. of the Third NASA/DoD Workshop on Evolvable Hardware*, pages 271–279, 2001.
6. D. Koch, C. Beckhoff, and J. Teich. ReCoBus-builder - a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. In *Proceedings of Int. Conf. on Field-Programmable Logic and Applications*.
7. D. Koch, C. Beckhoff, and J. Teich. Minimizing internal fragmentation by fine-grained two-dimensional module placement for runtime reconfigurable systems. In *Proceedings of 17th IEEE Symp. on Field-Programmable Custom Comp. Machines (FCCM*.
8. S. Li, J. Torresen, and O. Soraasen. Exploiting reconfigurable hardware for network security. In *11th Annual IEEE Symp. on Field Programmable Custom Computing Machines (FCCM'03)*. IEEE, 2003.

9. S. Li, J. Torresen, and O. Soraasen. Exploiting stateful inspection of network security in reconfigurable hardware. In *Field-Programmable Logic and Applications: 13th International Conference on Field Programmable Logic and Applications (FPL-2003)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

10. G. Nilsen, J. Torresen, and O. Soraasen. A variable word-width content addressable memory for fast string matching. In *Proc. of 22nd Norchip Conference*, pages 214–217. IEEE, 2004.

11. J. Torresen, J. W. Bakke, and L. Sekanina. Efficient image filtering and information reduction in reconfigurable logic. In *Proc. of 22nd Norchip Conference*, pages 63–66. IEEE, 2004.

12. J. Torresen and J. Jakobsen. An FPGA implemented processor architecture with adaptive resolution. In *Proc. of 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2006)*. IEEE, 2006.

13. J. Torresen and K. Glette. Improving flexibility in on-line evolvable systems by reconfigurable computing. In *Evolvable Systems: From Biology to Hardware. Seventh International Conference, ICES'07*, volume 4684 of *Lecture Notes in Computer Science*, pages 391–402. Springer-Verlag, 2007.

14. J. Torresen. An evolvable hardware tutorial. In *Proc. of the 14th International Conference on Field Programmable Logic and Applications (FPL 2004)*, pages 821–830. Springer Verlag, LNCS 3203, 2004.

15. Y. Agarwal et al. Solving fracture mechanics problems using reconfigurable computing. In *Proc. of Int. Conf. on Reconfigurable Computing and FPGAs, ReConFig'04*, 2004.

16. L. Sekanina and R. Ruzicka. Design of the special fast reconfigurable chip using common FPGA. In *Proc. of Design and Diagnostics of Electronic Circuits and Systems - IEEE DDECS'2000*, pages 161–168, 2000.

17. J. Torresen and K.A. Vinger. High performance computing by context switching reconfigurable logic. In *Proc. of the 16th European Simulation Multiconference*, pages 207–210. SCS Europe, June 2002.

18. K. A. Vinger and J. Torresen. Implementing evolution of FIR-filters efficiently in an FPGA. In *Proc. of the 2003 NASA/DoD Workshop on Evolvable Hardware*, 2003.

19. K. Glette, J. Torresen, M. Yasunaga, and Y. Yamaguchi. A flexible on-chip evolution system implemented on a Xilinx Virtex-II Pro device. In *Proc. of Evolvable Systems: From Biology to Hardware. Sixth International Conference, ICES 2005. Volume 3637 of Lecture Notes in Computer Science*, pages 66–75. Springer-Verlag, 2005.

20. K. Glette, J. Torresen, M. Yasunaga, and Y. Yamaguchi. On-chip evolution using a soft processor core applied to image recognition. In *Proc. of the First NASA /ESA Conference on Adaptive Hardware and Systems (AHS 2006)*, pages 373–380. IEEE Computer Society, 2006.

21. K. Glette, J. Torresen, and M. Yasunaga. An online EHW pattern recognition system applied to face image recognition. In M. Giacobini et al., editor, *Applications of Evolutionary Computing, EvoWorkshops2007: EvoCOMNET, EvoFIN, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC, EvoTransLog*, volume 4448 of *Lecture Notes in Computer Science*, pages 271–280. Springer-Verlag, 2007.

22. K. Glette, J. Torresen, and M. Yasunaga. Online evolution for a high-speed image recognition system implemented on a Virtex-II Pro FPGA. In *The Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*. IEEE, 2007.

23. D. Lim and M. Peattie. *Two flows for partial reconfiguration: module based or small bit manipulation, Application Note 290*. Xilinx, 2003.

24. N.P. Sedcole, P.Y.K. Cheung, G.A. Constantinides, and W. Luk. On-chip communication in run-time assembled reconfigurable systems. In *Proc. of IC-SAMOS*. IEEE, 2006.

25. H. Kawai, M. Yasunaga, K. Glette, and J. Torresen. An adaptive pattern recognition hardware with dynamic partial reconfiguration. 2008. In preparation.

26. G.A. Senland. *Design of an Architecture for Evolvable Hardware based on Internal Reconfiguration of an FPGA (in Norwegian)*. University of Oslo, 2008. Master thesis.

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

28. D. Koch and J. Torrresen. Advances in component-based system design and partial run-time reconfiguration. In *Proc. of Dagstuhl Seminar 10281*, Lecture Notes in Computer Science, 2010.