DYNAMIC PARTIAL FPGA RECONFIGURATION IN A PROTOTYPE MICROPROCESSOR SYSTEM

Kai Schleupen[1], Scott Lekuch[1], Ryan Mannion[2], Zhi Guo[3], Walid Najjar[2], Frank Vahid[2]
[1]IBM System & Technology Group; {kais, scottl}@us.ibm.com
[2]Department of Computer Science & Engineering, University of California, Riverside; {rmannion, najjar, vahid}@cs.ucr.edu
[3]Brocade Communications System; zguo@brocade.com

ABSTRACT

Modern FPGAs' parallel computing capability and their ability to be reconfigured make them an ideal platform to build accelerators for supercomputing systems. As a multi-core processor, the recently announced Cell Broadband EngineTM1 offers tremendous computing power. In this paper, we introduce a prototype system that combines these two types of computing devices together in a reconfigurable blade and we describe its architecture, memory system and abundant interfaces.

On the reconfigurable blade it is desirable that the FPGA devices can be partially reconfigured at run-time. This paper presents the dynamic partial reconfiguration (DPR) technique and its design flow for the reconfigurable blade. We report our experimental results of the blade doing partial reconfiguration. DPR allows the reconfigurable blade to be a powerful, run-time changeable computing engine. A sample application is presented that was both simulated for the Cell processor and dynamically loaded to run on the FPGA.

1. INTRODUCTION

A Configurable System-on-a-Chip (CSoC) has one or more microprocessors integrated with a field programmable gate array (FPGA). These CSoC devices' high transistor density FPGA fabrics and their embedded microprocessors provide a configurable accelerator platform for supercomputing systems.

The recently announced Cell Broadband Engine™ can achieve 256 GFlops peak performance at 4 GHz (single precision). Cell is a multi-core device comprising a 64-bit Power™[2] processor core and multiple synergistic processor cores [2][3]. A system that combines both the Cell processor and FPGAs together would be a platform on which these two types of devices serve as a complement to each other. In this paper we present the design of a reconfigurable Cell-based blade system that includes two FPGAs that support dynamic partial reconfiguration.

Normally, the configuration process applied to a configurable device flushes the preexisting configuration, if a configuration exists, and fully programs the whole device. During the configuration process, the system does not function. However, under some circumstances, reconfiguring part of the device at run-time while the remainder continues working is desirable. One such example is shown in Figure 1, in which a data stream goes through a configurable system. One of the modules working on the data stream is a string matching module searching for particular strings. The strings being searched for need to be changed or updated, but the link to the data stream has to be preserved, necessitating - dynamic partial reconfiguration (DPR)

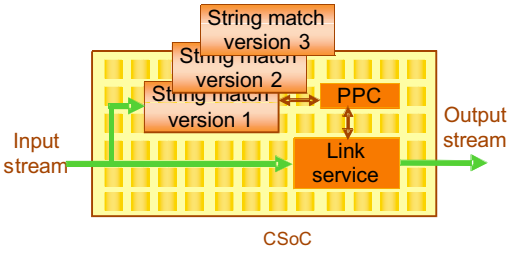

Figure 1 - A system with a string matching module. The string matching module needs to be dynamically changed at run-time but the link has to be preserved.

Both the Xilinx Virtex and Virtex-4 architectures support dynamic reconfiguration. Xilinx Virtex, Virtex-2, and Virtex-2 Pro devices can be reconfigured in a column-based way [4]. The authors of [5] developed the "merge dynamic reconfiguration" technique in the Virtex-4 architecture. In this paper, we present our approach to harness the dynamic partial reconfiguration technique in a reconfigurable blade. We introduce the reconfigurable blade's system overview in section two. Section three presents our approach to integrating the dynamic partial reconfiguration technique with our reconfigurable blade system. We report the experimental results in section four. Section five concludes the paper.

2. RECONFIGURABLE BLADE

The reconfigurable blade's main devices consist of one Cell processor and two high-end FPGAs. The Cell Broadband Engine™ is composed of one 64-bit PowerPC (Power™ Processor Element, or PPE) and eight Synergistic Processing Elements (SPEs) [2][3]. The PPE has 32 Kbytes of L1 instruction and data cache, and has 512 Kbytes of L2 cache. Each SPE has 256 Kbytes of dedicated on-die

[1] Cell Broadband Engine is a registered trademark of Sony Computer Entertainment, Inc.
[2] Power is a registered trademark of IBM Corporation

1-4244-1060-6/07/$25.00 ©2007 IEEE.

memory, known as a Local Store (LS). Both the PPE and the SPEs support Single-Instruction-Multiple-Data (SIMD). Figure 2 depicts the system overview of the reconfigurable blade, on which there is one Virtex-4 LX160 FPGA ("FPGA #1") and one Virtex-4 FX100 FPGA ("FPGA #2"). The XDR™ DRAM memory system is connected to one end of the Cell processor. The XDRAM memory system delivers high memory bandwidth. The FlexIO™[3] [6] interface provides the connection between the other end of Cell and FPGA #1. The Rambus FlexIO Interface is a high-speed bus supporting a wide range of operating speeds. Capable of data rates from 400 Mbps to 8.0 Gbps, FlexIO interfaces are designed for low-cost consumer systems utilizing cost-effective packages and boards.

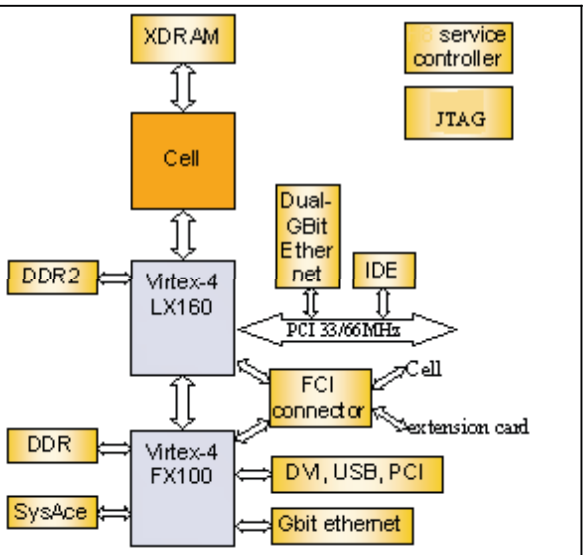

Figure 2 - The reconfigurable blade system overview.

FlexPhase circuit technology is a major departure from traditional circuit technologies. It enables precise, per bit, on-chip alignment of data and clock, eliminating the need for PCB trace length matching and PCB timing constraints. This results in logic systems capable of multi-GHz data rates that are simpler, more compact, and lower in cost. FlexIO interface offers a high bandwidth between the Cell processor and the FPGA chip.

The FPGAs each have their own local memory connections, either to DDR1, DDR2 or SRAM. A multi lane high-speed SERDES bus forms the interconnection between the two FPGAs and can achieve up to 5 GB/s bidirectional bandwidth. The reconfigurable blade has an abundant array of interfaces, including PCI express, PCI, an IDE hard-disk interface, DVI, dual high speed Ethernet channels, an high speed expansion connector, a Xilinx SystemACE compact flash card interface, USB, etc. The blade has 4,606 components in total.

[3] XDR and FlexIO are registered trademarks of Rambus, Inc

The two configurable devices, FPGA #1 and FPGA #2 are both Xilinx's high-capacity FPGAs. The former suites high-performance logic-intensive applications, while the latter (with two PowerPC405 hardware macros) suites embedded applications. These two devices are connected in the same boundary scan (JTAG) chain, as well as a SystemACE device (the Xilinx compact flash solution peripheral). FPGA #1's SelectMap interface is connected with FPGA #2.

3. DPR ON THE RECONFIGURABLE BLADE

In this paper, we only discuss the dynamic partial reconfiguration in the FPGA #2 FPGA. The same technique can be used to partially reconfigure either one FPGA or both.

3.1. The blade's reconfiguration system

Figure 3 shows the reconfiguration system on the reconfigurable blade. In FPGA #2, the dynamically

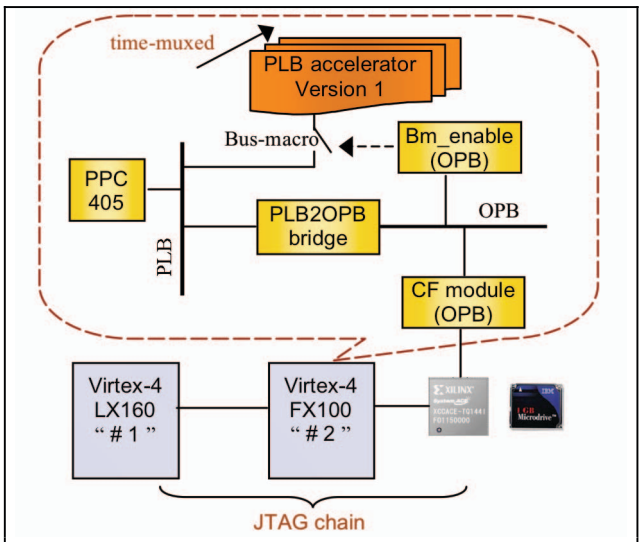

Figure 3 - The reconfiguration system on the reconfigurable blade. The two FPGAs are in the same JTAG chain. The on-chip PowerPC controls the dynamic partial reconfiguration process.

reconfigurable portion is a *PLB accelerator*, which has multiple implementation versions but only one of the implementations is instantiated in hardware at one time. The *PLB accelerator* is connected with one of the two PowerPC 405s through PLB (processor local bus). Notice that the accelerator is interfaced to the PLB through the bus-macros. The bus-macros are disabled/enabled by *Bm_enable*, which is an OPB (on-chip peripheral bus) module. The OPB SystemACE module, *CF module,* provides an interface between the PPC405 and the microprocessor in the SystemACE device.

3.2. DPR design flow

Figure 4 shows the reconfigurable blade's dynamic partial reconfiguration design flow. Bus-macros are inserted between the accelerator (the dynamic module) and the

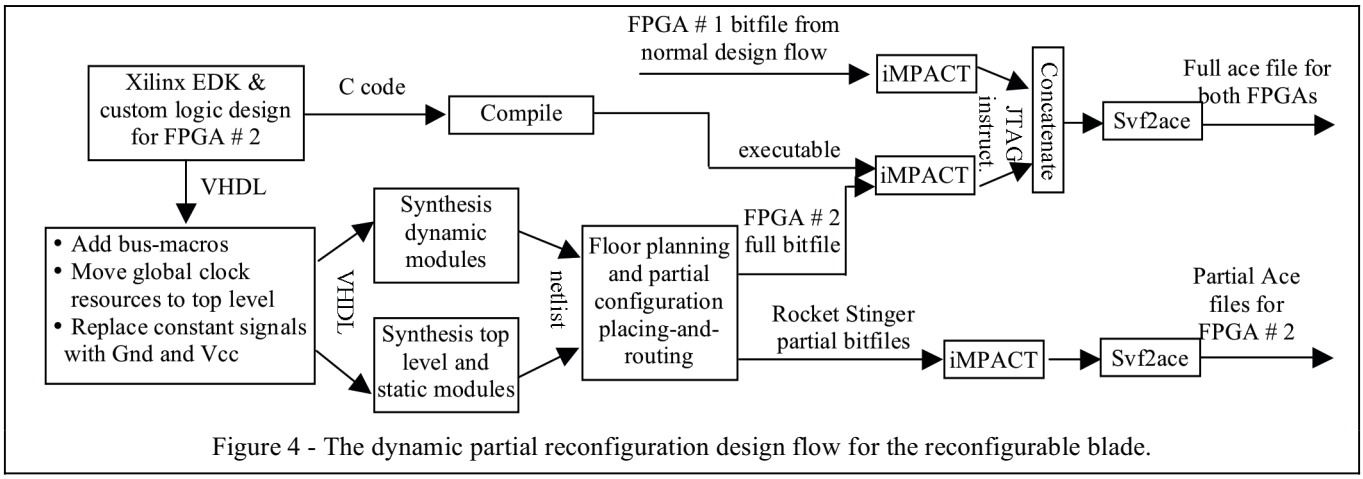

Figure 4 - The dynamic partial reconfiguration design flow for the reconfigurable blade.

static portion to provide the locking routing. The only exception is the global clock signals, which go into the dynamic module directly, since the configuration of clock resources is separate from the configuration of logics and therefore partial configuration does not change the global clock signals' routing. At the time of this paper's writing, Xilinx's partial reconfiguration place-and-route tool requires that all the global clock resources need to be at the top level of the design. Any wrapped global clock resources, such as DCMs (digital clock manager), should be moved to the top level of the design. Another issue is that the constant signals (zeros and ones, commonly seen in bus signals) passing the static-dynamic boundary through bus-macros cannot always be successfully recognized as "connections" by the partial configuration tool. One solution is manually assigning the constant signals to Gnd/Vcc rather than wiring the signals to bus-macros. We use Xilinx PlanAhead to perform floor planning and manage the partial reconfiguration implementation flow to generate the full and partial bit files.

The two FPGAs in our system are configured by the JTAG chain, as shown in Figure 3. The compact flash card provides the storage for the full and partial configurations. FPGA #2's full configuration file and executable, and FPGA #1's full configuration file, are formatted to JTAG instructions separately, and then are concatenated together. The concatenated full configuration for both the two FPGAs is converted to an ACE file. Each version of the partial configuration file is also converted to an ACE file. When generating the partial configuration ACE file, we set the option that prevents the JTAG chain from flushing FPGA #1's configuration.

3.3. DPR work flow

A subroutine running in one of FPGA #2's PowerPC405s controls the partial reconfiguration flow. During partial reconfiguration, the bus-macros from the dynamic portion to the static portion have to be disabled. Otherwise, the unpredictable signal toggling during partial reconfiguration would drive the system, including the PowerPC405, to an unknown status.

4. EXPERIMENTAL RESULTS

We take the string matching example in Figure 1 as an experimental application and report the experimental results in one of the system's FPGAs and the simulation results for the Cell processor.

4.1. Hardware accelerator performance

We set up a system as shown in the dashed region in Figure 3 in FPGA #2. The input data stream fed into the dynamic accelerator module consists of two 32-bit words per cycle. For each 32-bit input stream, the accelerator checks the current word, the word shifted by one, two, and three bytes with a 32-bit string pattern. The calculation is equivalent to four 32-bit comparators. 64 32-bit patterns are being checked simultaneously. Therefore, for one input word, the accelerator performs 256 (4 x 64 = 256) comparisons, as shown in Figure 5. For both the two 32-bit input streams, the whole computation is equivalent to 512 32-bit comparators. We have two versions of the accelerators, differing by patterns.

In FPGA #2, a 13x120 array of CLBs is dedicated to the dynamic accelerator. The overall system area cost is 5,455 slices, while the accelerator costs 3,264 slices. The throughput of the accelerator is two words per clock cycle.

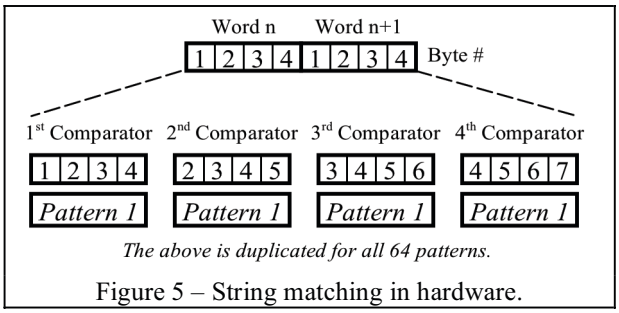

Figure 5 – String matching in hardware.

Table 1 - Reconfiguration experimental results

Partial bit file size (KByte)	Partial ace file size (KByte)	JTAG clock (MHz)	DPR time (ms)
434	435	33	290

The size of the generated partial reconfiguration ACE file is 435 Kbytes [Table 1]. We use the PPC405's on-chip timer to measure the number of the clock cycles that the PPC405 has to wait for the DPR process. The PPC405 runs at 100MHz and the DPR time is 290 milliseconds for this example.

Table 2 - Hardware performance results

System clock (MHz)	Accelerator thrghpt (input words/cycle)	Peak comparisons per second (Billions/s)
100	2	51.2

Table 2 shows the accelerator's computing performance. The FPGA #2 works at 100MHz. The string matching accelerator can achieve the peak performance of up to 51.2 billion 32-bit comparisons per second at this clock rate, or 200 million input words compared against 64 patterns per second.

4.2. Software performance

We ran the string matching example as software running on a simulation of the Cell processor's SPE as a basis for comparing the relative performance of the dynamic accelerator architecture. We utilized IBM's Full-System Simulator, also known as Mambo [1]. Mambo supports cycle-accurate performance modeling of code running on an SPE, but not for memory accesses outside an SPE's local store. Thus we limit the scope of our analysis to the throughput of the string-matching process assuming that the input data stream and string patterns are already present in the local store. This is a fair limitation since the performance figures discussed in 4.1 are isolated to an analogous process in hardware.

We implemented the string matching example as a C program running in a Linux environment on the simulator. We wrote the code in a manner that took full advantage of the SPE's SIMD support. As SPEs operate on 128-bit data, SPEs can process four 32-bit input streams simultaneously. Our implementation includes a loop used to iterate through the 64 string-matching patterns being checked. We unrolled said loop by a factor of 8 to minimize dependency stalls.

After running our software string-matching example in the simulator using an input data array of 1,024 4-word elements, the simulator reports a runtime of 1,019,960 cycles. Assuming a clock rate of 3.2GHz, this figure corresponds to 318.7375 μs to compare 4,096 32-bit elements to 64 patterns. This rate corresponds to a peak performance of matching approximately 12.851 million elements per second if a single SPE is utilized or 102.806 million elements per second for 8 SPEs, disregarding memory accesses.

5. CONCLUSION

The increasing capacity and the on-chip processors of CSoC devices make them suited to be an accelerator for a supercomputing system. Meanwhile, the recently announced Cell Broadband Engine™ delivers up to 256GFlop peak performance. We designed a reconfigurable blade to combine these two types of computing devices together. In this paper, we introduced the architecture and the interfaces of the reconfigurable blade.

As a computing system, for some applications, the reconfigurable blade has to keep running when reconfiguring its FPGAs. It is desired to be able to perform dynamic partial reconfiguration at run-time. In the paper we presented the design flow to accomplish this. We reported the experimental results of an application example in one of the system's FPGAs and the simulation results for the Cell processor.

It is still too early to measure either the Cell processor's performance on the reconfigurable blade or the overall performance of the blade. Our future work includes the improvement of the inter-chip interfaces and the evaluation of the system.

Acknowledgement

This work was supported in part by the Semiconductor Research Corporation (2005-HJ-1331).

References

[1] IBM Full-System Simulator for the Cell Broadband Engine™ Processor. http://www.alphaworks.ibm.com/tech/cellsystemsim.

[2] Introduction to the Cell multiprocessor. http://researchweb.watson.ibm.com/journal/rd/494/kahle.html , 2006.

[3] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, K. Yazawa The Design and Implementation of a First-generation Cell Processor. In Procs. of the IEEE Int. Solid-State Circuits Symposium, February 2005.

[4] D. Lim and M. Peattie. Two flows for partial reconfiguration: module based or small bit manipulation, Application Note 290, Xilinx, 2002.

[5] P. Sedcole, B. Blodget, T. Becker, J. Anderson and P. Lysaght. Modular dynamic reconfiguration in Virtex FPGAs, IEE Proceedings of Computers and Digital Techniques, May 2006, Volume 153, Issue 3, p. 157-164.

[6] K. Chang, S. Pamarti, K. Kaviani,E. Alon, X. Shi, T.J. Chin, J. Shen, G. Yip, C. Madden, R. Schmitt, C. Yuan, F. Assaderaghi, M. Horowitz, Clocking and Circuit Design for a Parallel I/O on a First-generation CELL Processor, International Solid-State Circuit Conference 2005, pp 526-527.