Implantation of Dynamically Reconfigurable Systems on Chip with OS Support

Vaibhawa Mishra
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

Kota Solomon Raju
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

Pramod Tanwar
CSIR-CEERI
Digital Systems Group
Pilani, Rajasthan, India-333031

ABSTRACT
This work presents the implementation of dynamically reconfigurable system with operating system support specifically Linux. The presented work combines both HW and SW flows where the complex parts of the architecture are designed to HW modules. These HW modules can be reconfigured on the fly by using partial dynamic reconfiguration. In our work, we are using floating point computation unit as partial reconfiguration module. Our aim is to show the idea how an operating system can be involved in the area of reconfiguration computing. The application that manages the reconfiguration can be developed either as standalone software that is specific for the system or with an operating system support, to achieve code reusability and code portability. Finally, a prototype is implemented on Xilinx ML507 board, where a general Linux open source kernel has been used to handle dynamic reconfigurable hardware recourses.

Keywords
Partial reconfiguration, Embedded Linux, Floating point arithmetic units.

1. INTRODUCTION
The leading advantage of the FPGA based embedded system is that some part of the active circuit may be partially reconfigured externally or internally [1]. The internal reconfiguration techniques of various Xilinx devices basically involve internal configuration access port [2]. By using this feature of Xilinx platform FPGAs we propose a method of using a configuration controller to achieve reconfiguration application that has been supported by a running operating system especially for complex designs that benefit from partial reconfiguration by taking all arithmetic operation for floating point data as an example. The different modules for reconfiguration region are simply adder, multiplier divider and subtractor.

Linux may be a good option because many advantages are associated with it like portability, support for various processor architecture and open source [3]. A co design flow for reconfigurable computing system with RTOS support has been presented in [4]**.** Our main aim is to develop prototype for dynamically reconfigurable system with the Linux kernel support where reconfiguration application will be managed by OS. To demonstrate this idea, Xilinx ML507 FPGA board embedded with PowerPC 440 hard core processor has used as target platform [5]. Linux open source kernel versions 2.6.34 used as main operating system and ELDK 4.1 [6] tool chain has been used to compile the kernel as well as reconfiguration controlling application. Some configuration change is required using "make menuconfig" utility, in original kernel to use exact drivers for hard core device in Xilinx FPGA specifically Internal Configuration Access Port. We have followed the guidelines to develop a Dynamic partial reconfigurable system as in [7]**.** As soon the required hardware peripheral and configurable IP-core are ready, there is need to establishment the communication channel between the OS and module itself [8]**.** A well written device driver is required to access reconfigurable region as well as hardware module. All required device driver, partial bit files and software for application are bundled with final executable image [9]. Once the target system boots, the image will be transferred from the host to target via JTAG.

This paper is organized as follows. Section 2 presents brief description of proposed hardware architecture. The software part of the design such as cross compiling of the kernel source has been presented in Section 3. The experimental results and conclusions have been presented in Section 4 and 5 respectively.

2. HARDWARE IMPLELNTATION
Complete system hardware architecture is shown as in Figure 1. It shows the components of the constructed self-reconfiguring platforms targeting 32-bit embedded PowerPC hard processor core.

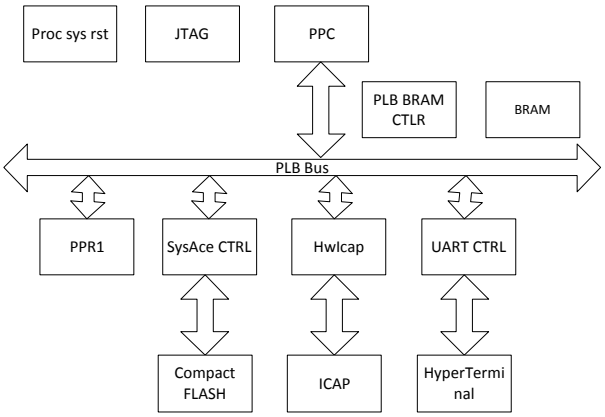

Fig 1: Proposed hardware architecture

The proposed system is designed on a ML507 board having xc5vfx70t-1ffg1136 with only required modules. Along with one processor, the system has DDR memory to store kernel image and partial bit files, UART terminal as input/output console, ICAP peripheral to achieve reconfiguration and

reconfigurable custom IPs that contains floating point arithmetic unit wrapper to use proposed functionality. All the peripheral are connected with the processor via Processor Local Buses. Xilinx EDK is used to design the system and to generate the memory map for the hardware platform [10]. We have followed the IEEE 754 32 – bit single precision data format to represent floating point numbers. All floating point arithmetic module are designed in such a way that they take operands in 32 bit hex format and produce the output in same pattern. The wrappers for these modules are combined with configurable IP core and connected to the system with Processor Local Bus and synthesized separately. The discussions of algorithms to implement these arithmetic modules are quite straight forward and beyond this paper's scope. The algorithms for arithmetic units have been adopted from as it is from [11].

3. RECONFIGURATION MANAGEMENT

Since the proposed work is intended to use Linux kernel source to manage reconfiguration. So for each and every device, a unique and well examined driver needs to be implemented. We have used Xilinx provided driver for ICAP. For our custom reconfigurable IPs, we have implemented one character based device driver which implements simple I/O control based read and write system calls. In read operation data is copied from the kernel space to user space as our reconfiguration application is running in user space. On the other hand, in write operation data is copied from user space to kernel space. A device can be identified by its major number and its minor number. On system boot, the reconfiguration IP is registered in the Linux device subsystem. The IP may be accessed using standard system calls such as open, read and write. All the needed steps about cross compiling of Linux kernel source code and kernel configuration has been adopted from [12]. The application that performs reconfiguration is written in C. It has been cross - compiled to generate object code using ELDK 4.1 for PowerPC by executing the commands *"ppc_4xx-gcc –static icap.c –o icap"*. The program is bundled with the image when final image is created using "*ramdisk"* image creation method by simply using make command *"make simpleImage.initrd.virtex440-ml507"*. The reconfiguration application "icap" is user interactive program. The menu displayed on the hyper-terminal directs the user to reconfigure the region by pressing keys from the keyboard like '2' for addition. The data can be provided to the arithmetic floating point modules by another application "math".

4. RESULTS

The proposed system has been successfully implemented and tested with only the required components. We have used HyperTerminal to see the information related to booting of Linux OS on PowerPC440 as well as to give commands to reconfiguration controller and to view results from the floating point arithmetic IP core. It should be noted that in the Virtex-5 FXT device, PowerPC cores are the part of the FPGA fabric with no resource usage even though hard core processors in the FPGA fabric reduce the available area for logic in general. Table 1 provides the device utilization summary of all individual modules. The given resource utilization is only for the PRRs in the design. As it has been discussed in the previous section, we have implemented floating point arithmetic unit as reconfigurable IP core. The simulation results for addition, subtraction, multiplication and division are shown in Figure 2, 3, 4 and 5 respectively.

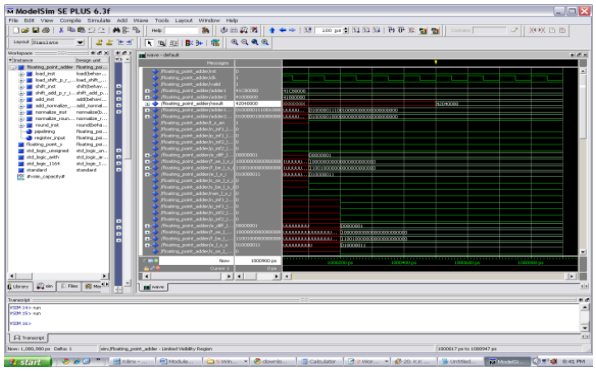

Fig 2: Simulation result of floating point adder

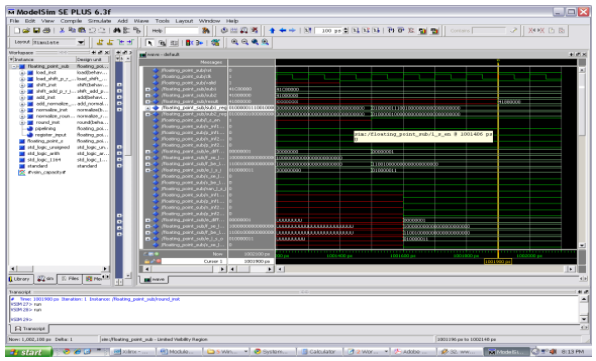

Fig 3: Simulation result of floating point subtractor

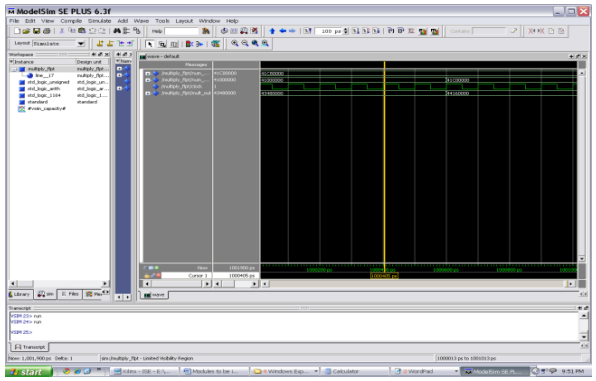

Fig 4: Simulation result of floating point multiplier

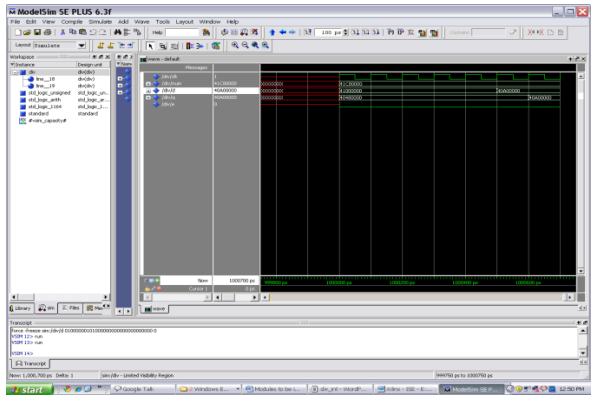

Fig 5: Simulation result of floating point divider

The device utilization is close for both static and reconfigure region. The PowerPC system used lesser amount of resources even though it required the use of extra PLB bus but it should be considered that the resource usage for the PRMs increases the utilization of resources.

Table 1. Resource usage of PRMs for Virtex 5 FXT

Floating Point Arithmetic Modules		Resource Used							
		Slice LUTs		Full Used LUT FF pairs		Slice Registers		DSP48Es	
		Min	Max	Min	Max	Min	Max	Min	Max
PRR1	Add.	956	44800	253	1183	480	44800	-	128
	Multi.	127	44800	90	247	215	44800	2	128
	Div.	2355	44800	91	2565	300	44800	2	128
	Subt.	951	44800	177	1183	485	44800	-	128

When the system boots, an initial bit file is loaded to FPGA to invoke all the required hardware. The FPGA is configured with Xilinx impact tool and once system is ready, the final kernel image is transferred via JTAG. UART terminal will show the booting process of Linux kernel as in Figure 6. The execution mode of the application need to change with "chmod" command before running the software and the output of the code is as in Figure 7.

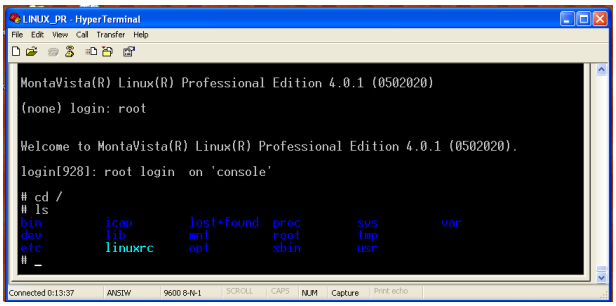

Fig 6: HyperTerminal output window

Fig 7: Application output

5. CONCLUSION

This paper attempts to provide the proof of concept for the implementation of Dynamically Reconfigurable Systems with Linux Operating System. We have tried to use the simple algorithmic blocks to demonstrate this. This approach can be extended for more complex systems like having CDMA & GSM blocks on a single chip *etc*. All the domains having a requirement of Low power & On-demand computation are best suited for the applications of these systems.

6. ACKNOWLEDGEMET

The authors would like to thank Dr. Chandra Shakher, Director, CSIR-CEERI and Dr. P. Bhanu Prasad, Group Leader, Digital Systems Group, CSIR-CEERI for allowing to utilize the resources of the institute.

7. REFERENCES

[1] Scott Hauk, Andr'e DeHon, "Reconfiguration Computing: The theory and practice of FPGA- based computation" Elsevier Inc. 2008.

[2] Xilinx, Inc., "Partial Reconfiguration User Guide", User Guide UG702, Version 12.3, October-5, 2010.

[3] Zhou Qingguo; Yao Qi; Li Chanjuan; Hu Bin;, "Port embedded Linux to XUP Virtex-II Pro development board," IT in Medicine & Education, 2009. ITIME'09. IEEE international Symposium on , vol. 1, no., pp.165-169, 14-16 Aug. 2009doi: 10.1109/ITIME.2009.5236439

[4] Xiao-Wei Wang; Wei-Nan Chen; Ying Wang ; Chen-Lian Peng; , " A Co-design Flow for Reconfigurable Embedded Computing System with RTOS Support, "Embedded Software and Systems, 2009. ICESS '09. International conference on , vol., no., pp.467-474, 25-26 May 2009 doi: 10.1109/ICESS.2009.84.

[5] Xilinx, Inc., "Virtex-5 FPGA Configuration User Guide", User Guide UG191, Version 3.9.1, August 20, 2010.

[6] www.denx.de/en/News/PressReleaseELDK41.

[7] Xilinx, Inc., "PlanAhead Software Tutorials: Partial Reconfiguration of a processor Peripheral" User Guide UG744, Version 12.3, September 21, 2010.

[8] Santambrogio, M.D.; Rana, V.; Sciuto, D.; , "Operating system support for online partial dynamic reconfiguration management," Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on , vol., no., pp.455-458, 8-10 Sept. 2008 doi: 10.1109/FPL.2008.4629982.

[9] J. W. Williams and N. Bergmann, "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip," in Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '04), T. P. Plaks, Ed., pp. 163–169, CSREA Press, Las Vegas, Nev, USA, June 2004.

[10] Xilinx Inc., "Embedded System Tools Reference Manual," Version 3.0, 2004.

[11] http://www.ece.uvic.ca/~elec499/2004a/group05/html/math.html.

[12] http://wilki.xilinx.com/powerpc-linux.