Indian Journal of Science and Technology, *Vol 9(1), DOI: 10.17485/ijst/2016/v9i1/85757, January 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Design and Analysis of RTOs based Reconfigurable Cots Design on FPGA

V. Gunasekaran*, K. Mohanasundaram, M. Dhandapani, S. Tamilselvan and S. Arivarasan

Department of Electrical and Electronics Engineering, Vel Tech Multitech Dr. Rangarajan Dr. Sakunthala Engineering College, Avadi, Chennai - 600062, Tamil Nadu, India; vgunasekaran@veltechmultitech.org, kmohanasundaram@veltechmultitech.org, dhandapani@veltechmultitech.org, tamilselvans@veltechmultitech.org, arivarasan@veltechmultitech.org

Abstract

Background: Due to greater potential to accelerate a wide variety of application, reconfigurable FPGA has become a subject to great deal of research. The FPGA has ability to perform computations in hardware to increase the system performance, while retaining much of the flexibility of software solutions. In recent development, the real-time embedded system are increasingly being built with COTS components namely mass-produced peripherals and by reducing cost through buses and performance improvement. But the available COTS systems do not guarantee any timeliness and do not implement any priority scheduling mechanism. **Methods/Statistical Analysis:** This article deals with a new approach to create an RTOS based reconfigurable COTS design for FPGA platform. Thereby RTOS kernel is designed for FPGA and COTS system designed on the FPGA. **Application/Improvements:** At runtime, the hardware tasks are scheduled and allocated system resources like I/O, memory, etc. to successfully export multiple virtual devices for a single physical device.

Keywords: FPGA, RTOS based Reconfigurable Commercial-off-the-Shelf

1. Introduction

FPGAs (Field Programmable Gate Arrays) are normally used to create digital circuits based on their design. In general FPGA is a blank slate and by itself nothing. A configuration file (bit file) is created by the designer for the FPGA. Once it is loaded the FPGA will function like the digital circuit as designed. The main advantages of FPGAs are over ASIC (Application Specific Integrated Circuit) is that the circuit design is not set and the designer can BORPH provides FPGA with conventional OS services such as file system support. It provides a unified HW/SW runtime environment with a familiar UNIX interface[1,2].

Reconfigurable computing has been accepted as vehicles for both achieving potentially much higher performance than software and maintaining a higher level of flexibility than hardware[3–6]. Utilization of FPGA in very efficiently and reconfiguring under any condition by the flow of control and the task management over the application[6–8].

Operating systems for reconfigurable devices which allow the development of embedded systems and particularly software tasks that are running on a CPU, can coexist with hardware tasks running on a reconfigurable hardware device (FPGA)[9,10].

FPGA has more potential than general purpose processor and also meet reliable requirements. But the need of standard tools and interfaces restricts FPGAs' to develop reconfigurable application and makes their programming not productive. R3TOS provides contribution to tackle this problem. It provides systematic OS support for FPGA and also is advantage due to the non-conventional way of exploiting on-chip resources. During runtime, the hardware tasks are scheduled and allocated with the dual objective of improving computation density and circumventing damaged resources on the FPGA.

*Author for correspondence

2. Materials and Methods

The proposed approach is to focus on providing RTOS on FPGA. As mentioned earlier FPGA is just a prototype and used to check the circuit functionality. The proposed method is to make the FPGA inheriting the proprieties of processor by in building RTOS in FPGA and check the functionality of RTOS using a simple application as a test model. Research is going on in implementing the reconfigurable FPGA as it is a complex task. By implementing RTOS the focus is to make the FPGA as to adopt the Intellectual Property (IP) and peripherals needed to quickly create a custom System on a Chip (SoC) tailored to fit any application. Consolidating two discrete devices into one and so reducing system power, cost and board size there by increasing the performance.

The working model involves by making the connections to be created virtually as object and achieved by using single implementation module. Modified RTOS kernel is adopted to build the RTOS along with the COTS framework. It is the standard frame work used for commercial market and this frame work is built under the user define application[11,12].

In this proposed method, RTOS mainly deals Inter Process Communication and has the following mechanisms:

- Semaphores
- Message Queues
- Shared Memory Segments

Among IPC it deals with the concepts of Task Management, Semaphores and Preemptive Scheduling.

2.1 Task Management

- Creating an RT task and memory without delay: this is not easy because memory has to be allocated and a lot of data structures, code segment must be copied/ initialized.
- The memory blocks for RT tasks must blocked in main memory to avoid access latencies due to swapping.
- Changing run-time priorities is dangerous. It may change the run-time behavior and predictability of the whole system

The Task management is done by using three tasks. The first task is to check the conditions base on the signal strength arriving from both the ADCs. Then the second task will responds to the semaphore1 arriving from Task 1. The third task will responds to the semaphore 2 arriving from Task 1 as shown in the Figure.1.

2.1.1 Task 1

- It receives the analog signals from the ADCs.
- Checks the signal strength from both the ADCs
- After comparing, the highest signal strength will be selected and through which the analog data is sent to the corresponding UART (Task 2/Task 3).
- Semaphores (xSemGive) are created for each task separately and delivers it to them respectively.

2.1.2 Task 2 and Task 3

- If the signal strength is greater with respect to the Task 2 then semaphore 1 is selected.
- If the signal strength is greater with respect to the Task 3 then semaphore 2 is selected.
- Then the analog data is transferred through that particular UART.

2.2 Semaphores

A semaphore is an object with an integer value two routines; namely are semwait () and sempost (). Because the initial value of the semaphore determines its behavior, before calling any other routine to interact with the semaphore.

In this paper, two semaphores are used to select the particular task. In background both the tasks will be running buy selection of UART will be made through the semaphore selection.

2.3 Preemptive Scheduling

Scheduling is classified into preemptive and non-preemptive scheduling. In this paper preemptive

Figure 1. Task management.

scheduling is used where at real time; if Task 2 receives higher signal strength then obviously UART1 is selected to transfer analog data. If signal strength is higher for Task 3 then under the condition of preemptive scheduling, the signal is scheduled to UART2 and from this analog data is transferred.

2.3.1 Off-the-Shelf Configuration

In this proposed approach, COTS is provided as an addition part in order to provide the adaptability to the varying or changing interface and protocol standards. The COTS frame work is used in our application end. The structural flow of application describes that RTOS has the control over the system which is played by the FPGA as shown in the Figure 2.

The application end is a COTS framework based design with specific conditions to make the product commercial. Our application is to transfer the general analog data through ADC and UART under the influence of RTOS. This application can be used as the commercial product in the market so the designer should be able to remake this application with some changes in functionality.

3. Result and Discussion

In order to verify the proposed approach a circuit is designed with SPARTAN TYRO PLUS. In this a PIC controller is used to distribute UART uniformly among the 3 modules. Three sensors are used and each will play the role of receiving audio signals.

Free RTOS used will schedule the transformation of audio data through the one which has the greatest signal strength. Here we use the module Tera Term for ZIGBEE, Hyper Terminal for Bluetooth and Command Prompt WiFi. Each module is discussed below with each execution. Here in order to trigger the signal among each module TTL signal is used and data is transferred and received accurately. For each and every module an extra RS322 is made to execute and is connected to the PIC controller where it is triggered accordingly. The signal strength is made to change among the three modules and can be seen through the LED light blinking. (Figure 3)

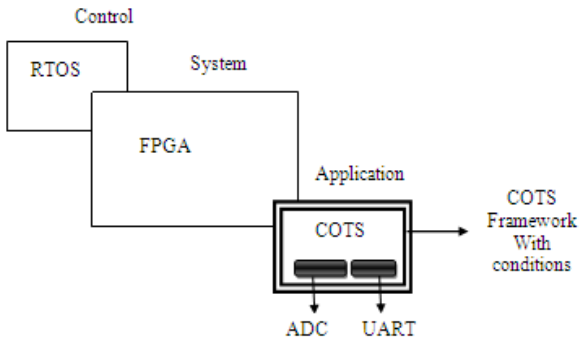

Figure 2. Structural flow of application.

3.1 Hyper Terminal

Hyper Terminal is the one where the Blue Tooth signal strength is measured. The transmitter and receiver signal strength is distributed via the PIC Controller for controlling the respective UART and the signal is triggered using the triggering board attached to it. (Figure 4)

3.2 Result

At run time the execution is tested by varying the signal strength at each module. The condition of our system is that making signal variation among each module and to verify the selection of analog data flow through the respective module. The execution of our application is made and verified efficiency of RTOS functionality. (Figure 5)

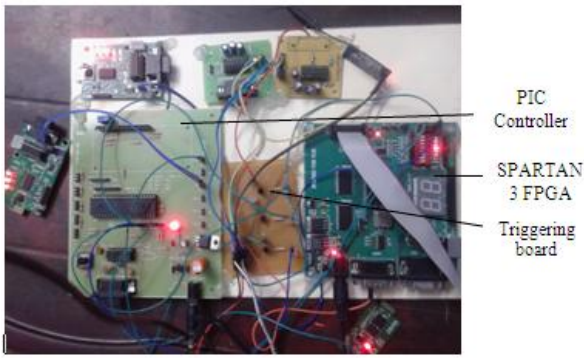

Figure 3. Hardware design.

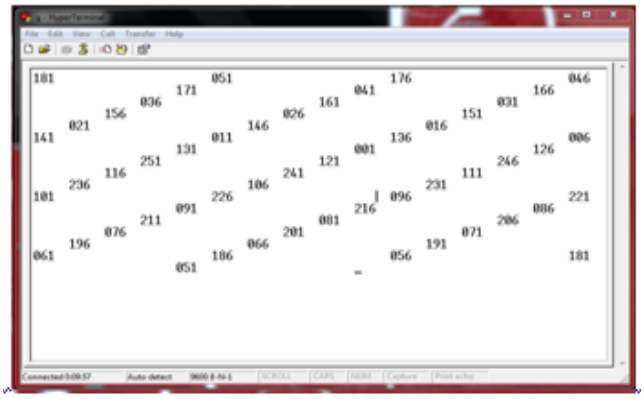

Figure 4. Hyperterminal.

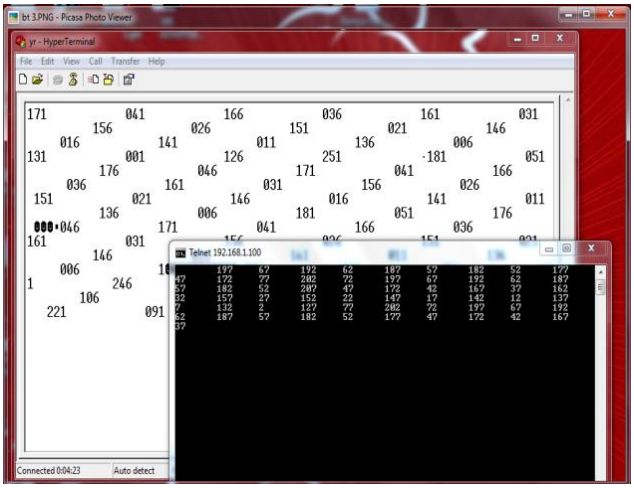

Figure 5. Run time execution window.

4. Conclusion

In this paper the efficient functionality of RTOS is tested. Each and every module is tested and verified experimentally that signal strength variation leads to efficient transformation of the audio data that send. As this is the initial stage of successful data transfer and further this method can be made to build large applications.

5. Acknowledgement

The authors would like to thank the fund from the DST-FIST, Govt of India, vide Ref.: SR/FST/College-189/2013, Dated: 6th August 2014

6. References

1. So HKH. BORPH: An operating system for FPGA-based reconfigurable computers. [PhD thesis]. Univ. of California at Berkeley; 2007.
2. Wigley G, Kearney D. Research issues in operating systems for reconfigurable computing. Proc Int'l Conf Eng Reconfigurable Systems and Algorithms. 2013 Aug; 62(8):1542–56.
3. Kari B, Muthulakshmi S. Real time implementation of speaker recognition system with MFCC and Neural Networks on FPGA. Indian Journal of Science and Technology. 2015 Aug; 8(19).
4. Zhou B, Qiu W, Peng C. An operating system framework for reconfigurable systems. Proc Int'l Conf Computer and Information Technology; 2005 Sep 21-23. p. 781–7.
5. Iturbe X, Benkrid K, Hong C, Ebrahim A, Torrego R, Martinez I, Arslan T, Perez J. R3TOS: A novel reliable reconfigurable real-time operating system for highly adaptive, efficient, and dependable computing on FPGAs. IEEE Transactions on Computers. 2013 Aug; 62(8):1542–56.
6. Iturbe X, Benkrid K, Arslan T, Torrego R, Martinez I. Methods and mechanisms for hardware multitasking: executing and synchronizing fully relocatable hardware tasks in Xilinx FPGAs. Proc Int'l Conf Field-Programmable Logic and Applications; Chania. 2011 Sep 5-7. p. 295–300.
7. Pellizzoni R, Caccamo M. Adaptive allocation of software and hardware real-time tasks for FPGA-based embedded systems. Proc IEEE Real-Time and Embedded Technology and Applications Symp. 2006 Apr 4-7. p. 208–20.
8. Gohringer D, Hubner M, Zeutebouo EN, Becker J. Operating system for runtime reconfigurable multiprocessor systems. Int'l J Reconfigurable Computing. 2011; 2011: article 121353.
9. Bak S, Betti E, Pellizzoni R, Caccamo M, Sha L. Real-time control of I/O cots peripherals for embedded systems. Proc IEEE 30th Real-Time Systems Symp; Washington DC. 2009 Dec 1-4. p. 193–203.
10. Pellizzoni R, Bui B, Caccamo M, Sha L. Coscheduling of CPU and I/O Transactions in COTS-based embedded systems. Proc Real-Time Systems Symp; Barcelona. 2008 Nov 30-Dec 3. P. 221–31.
11. Brebner GJ. A virtual hardware operating system for the xilinx XC6200. Proc Int'l Workshop Field-Programmable Logic. Smart Applications, New Paradigms and Compilers, London, UK: Springer-Verlag. 1996; 1142:327–36.
12. Walder H. Operating system design for partially reconfigurable logic devices. [PhD thesis]. Swiss Fed. Inst. of Technology, Zurich, Switzerland; 2005.