Volume : 4 | Issue : 6 | June 2015 • ISSN No 2277 - 8179

Implementation of Process Scheduling Strategies of Operating Systems Using Fpga

Engineering

KEYWORDS : VHDL, FPGA, MULTITASKING, ROUND ROBIN

Mr. Sagar. D. Bhopale Electronics Department T.K.I.E.T Waranangar

Prof. A. S. Mali Electronics Department T.K.I.E.T Waranangar

ABSTRACT

FPGA-based hardware implementation of multitasking strategies provide high-performance computing that rival ASICs for performance yet provide programmability and ease of development closer to traditional software systems. Because of their usefulness, such systems are high in demand and have limited availability. Multitasking can be used to increase the utilization of reconfigurable FPGAs. In this proposed system a flexible and up-gradable architecture is provided for using the commercially available reconfigurable FPGA'S. In this system different Multi Tasking Strategies are implemented using VHDL on FPGA. Because it can take advantage of an application parallelism, by using methods to context switch reconfigurable computing resources, multitasking reconfigurable computing systems can be constructed that provide most of the features available in traditional reconfigurable computers. This report describes methods to context switch the resources of a reconfigurable computers, as well as a prototype multitasking reconfigurable FPGA that has been created.

INTRODUCTION

When they were first introduced, FPGAs served as "glue-logic". However with their increasing density, they began to take other responsibilities. First they were used to implement larger logic functions and simple arithmetic. As their density grew, they started to be used for ALUs, DSP, and application-specific functions. Soon, the density of FPGA became larger enough that they were put to use in systems where the FPGAs themselves were the main components [1]. Although precursor devices were actively being studied in the research field, the first commercial FPGA emerged in 1986, developed by XILINX [2] .Today's bleeding edge commercial FPGAs, such as SPARTAN-2, contain two hundred thousand of gates and run at 4MHZ. Future devices will have even greater capacity, with more specialized features, available at even higher clock rates [1]. Multitasking can be used to increase the utilization of reconfigurable FPGA'S, because it can take advantage of application parallelism, Reconfigurable hardware solutions generally deliver higher performance than comparable software executing on a general purpose CPU. Multitasking can be used to increase the utilization of reconfigurable FPGA'S [4]. The big wins with multi-context FPGA are that the loading of configurations can be done while the current configuration is actively running and that configuration can be switched in a small number of clock cycles. Having a multi-context FPGA, it could greatly reduce the latency of context switching in multitasking reconfigurable systems. Similar to CPU support for context switching found in today's processors [3].

Multitasking refers to the concurrent processing of two or more tasks through a process of context switching. Today's Field Programmable Gate Arrays (FPGAs) can be reconfigured partially, which makes it possible to share resources between various functional modules (hardware tasks) over time [4].

Although the per unit cost for FPGAs is much greater than for ASICs in mass-produced systems, FPGAs still have several advantages. In situations where a system will not be mass-produces, purchasing several FPGAs is less Expensive than producing a small number of ASICs. In addition, FPGAs are infinitely reconfigurable, meaning that they may be used to do several types of computation, whereas ASICs can implement only a single function. Finally since FPGAs are easily reprogrammable, they can be developed and debugged faster than ASICs, and can have their configuration updated even after they are embedded in complete system [1].

Although per unit cost for FPGAs is much greater than for ASICs in mass-produced systems, FPGAs still have several advantages. In situations where a system will not be mass-produces, purchasing several FPGAs is less Expensive than producing a small number of ASICs. In addition, FPGAs are infinitely reconfigurable, meaning that they may be used to do several types of computation, whereas ASICs can implement only a single function. Finally, since FPGAs are easily reprogrammable, they can be developed and debugged faster than ASICs, and can have their configuration updated even after they are embedded in complete system [1].

Recent research has expanded to encompass a view of the reconfigurable logic where a resource is shared among multiple applications or users. Runtime reconfiguration is a method of computing by dynamically changing hardware configurations from phase to phase of a computation. In real-time multitasking system design, task deadlines play an important role as systems not only need to support sharing of the resources in space, but also need to guarantee execution of the tasks. At the operating system level, sharing logic gates, wires, and I/O pins among multiple tasks needs to be managed, and, from the high level standpoint, access to the resources needs to be scheduled according to task deadlines [9].

Exploring the efficiency of reconfigurable computing, one of the dimensions is the possibility for multitasking on an FPGA-based processor. Conventional computers and operating systems have demonstrated the many advantages of sharing computational hardware by several tasks over time. Design of computing systems is facing an interesting challenge with the opportunity to include runtime reconfigurable (RTR) devices in them. Operating systems (OS) need to be extended with functionalities that allow to efficiently manage such devices [11].

Dynamically and partially reconfigurable field-programmable gate arrays (FPGAs) allow to swap in and out tasks without interrupting the execution of other tasks. The FPGA controller can decide on-line where to place new tasks onto the FPGA. Rearranging a subset of the tasks executing on the FPGA may allow the next pending task to be processed sooner. When tasks are rearranged, the arriving input data have to be buffered while the execution is suspended [8].Partial reconfiguration allows for mapping and executing several tasks on an FPGA during runtime. Multitasking on FPGAs raises a number of questions on the management of the reconfigurable resource which leads to the concept of a reconfigurable operating system. A major aspect of such an operating system is task placement. Online placement methods are required that achieve a high placement quality and lead to efficient implementations. This work presents placement methods that rely on efficient algorithms for the partitioning of the reconfigurable resource and a hash matrix data structure to maintain the free space [7].

2. IMPLEMENTATION OF FPGA BASED HARDWARE TASK MANAGEMENT STRATEGIES

2.1 Block Diagram of Proposed System

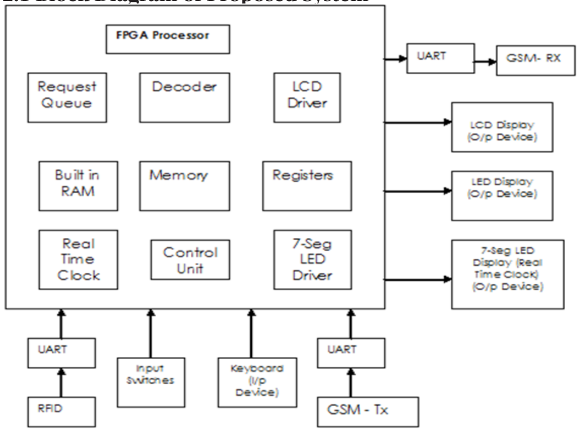
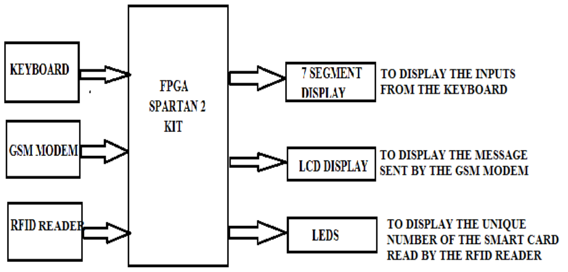


Figure 2.1 Block diagram of proposed systemFigure 2.2 Functional Diagram of Proposed System


KEYBOARD
GSM MODEM
RFID READER
FPGA SPARTAN 2 KIT
7 SEGMENT DISPLAY
TO DISPLAY THE INPUTS FROM THE KEYBOARD
LCD DISPLAY
TO DISPLAY THE MESSAGE SENT BY THE GSM MODEM
LEDS
TO DISPLAY THE UNIQUE NUMBER OF THE SMART CARD READ BY THE RFID READER

Working Principle

- The hardware tasks (applications), in the functional diagram run in different modes.
- There are three modes namely multitasking, Round-Robin & request/acknowledgement modes implemented in this proposed system.
- The inputs from the keyboard are displayed on 7-Segment display interfaced.
- The message received from the GSM MODEM is displayed on LCD display.
- The unique identification number of the smart card is read by the RFID reader and is displayed on LEDS on the FPGA board.
- The different modes of operations are explained in the section 2.3.

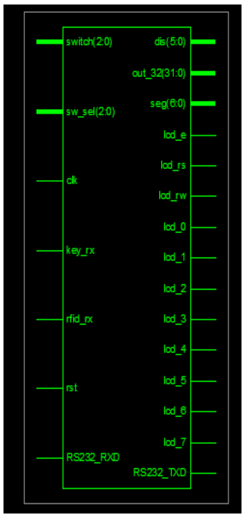


Figure 2.3 RTL Schematic of Hardware Task Management

2.2 Functionality of Modules

2.2.1 GSM Modem

The working of GSM MODEM contains two sections, UART receiver and the UART transmitter section, the UART receiver, receives the message from the user mobile. UART transmitter sends an acknowledgement to the user mobile.

UART Receiver

The UART Receiver receives the messages sent by the mobile user. The GSM Modem to read that particular message AT command is needed, that AT command is AT+CMGR=loc↵

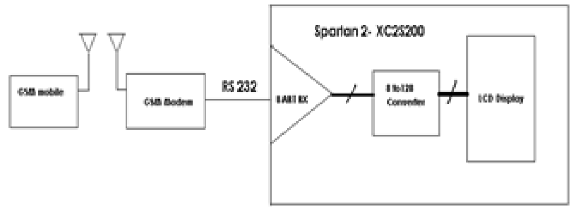


Figure 2.4 UART Receiver

The modem is interfaced to FGPA kit through UART (RS232 cable), The UART receiver on FPGA module receives the upcoming serial data and converts it into parallel 8-bit data. In this proposed system the upcoming data (message) is displayed on the LCD display. This LCD display is having the capability of displaying 16 characters at a time; it means up to 128 bits (16*8) can be displayed. For that 8-bit to 128 bit converter is required and it is implemented in this proposed system.

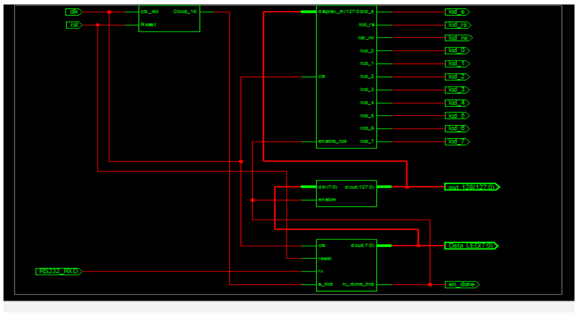


Figure 2.5 RTL Schematic of the GSM Modem Receiver

This RTL Schematic shows the entire working of GSM MODEM Receiver section. First baud generator is taking the available 4 MHz clock of spartan2 FPGA kit and converting it into 16 clock pulses and which is giving to UART receiver as clock. Then UART receiver takes serial data (message) from GSM MODEM through RS232 cable and converts it into 8-bit parallel data and which is applied to 8 to 128 converter. The 8 to 128 converter converts the upcoming 8-bit parallel data into 128 bits and then it is giving to LCD as input. Then LCD will display the result (message).

UART Transmitter

The AT command given AT+CMGR=loc↵ is of 11 characters (11*8= 88 bits), A enable generator is used to convert 88 bits to 8-bit parallel data. UART transmitter converts 8-bit parallel data to serial data, and then at the end of this process the user gets the delivery report as acknowledgement

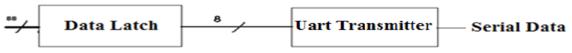


Figure 2.6 GSM MODEM Transmitter Section

Volume : 4 | Issue : 6 | June 2015 • ISSN No 2277 - 8179

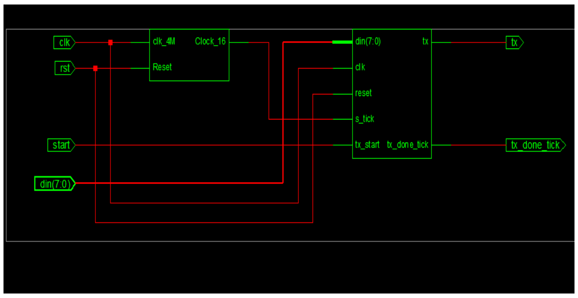


Figure 2.7 RTL Schematic GSM Modem Trasmitter

This RTL Schematic shows the entire working of GSM MODEM Transmitter section. First baud generator is taking the available 4 MHz clock of spartan2 FPGA kit and converting it into 16 clock pulses and which is giving to UART transmitter as clock. Then the transmitter takes the 8 bit parallel data and converts it into serial data then through this entire process mobile user will get acknowledgement.

2.2.2 RFID Reader

The RFID reader shown in the figure reads the unique identification number (UIN) of the RFID (Smart cards) tag, the RFID reader is interfaced to the FPGA module through UART, the UART receiver on the FPGA module converts the upcoming serial data into 8-bit parallel data. The RFID tag is used in this system has the 8 character UIN (8*8=64 bits), each character has a start bit and a stop bit, totally 80 bits. Decoder converts the 8-bit data into 80 bits and is displayed on LEDs on the FPGA board.

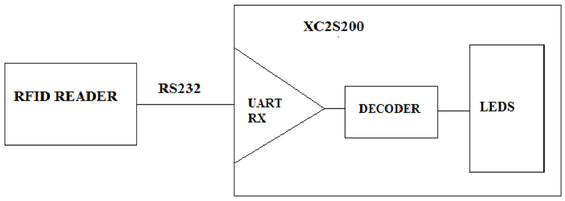


Figure 2.8 RFID Receiver Section

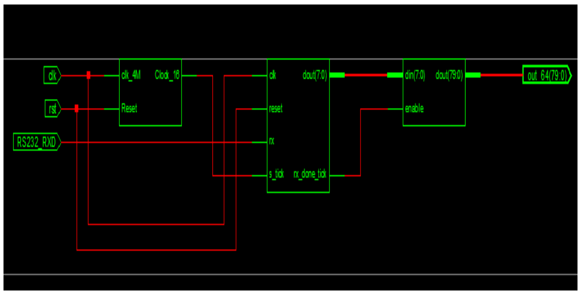


Figure 2.9 RTL Schematic of the RFID Receive

This RTL Schematic shows the entire working of RFID Receiver section .First baud generator taking the available 4 MHz clock of spartan2 FPGA kit and converting it into 16 clock pulses and which is giving to UART Receiver as clock. Then the UART Receiver takes the unique identification number (UIN) from the RFID Reader through RS232 cable in a serial way then converts it into 8 bit parallel data. Then the decoder converts the 8 bit data into 80 bit parallel data. Then the 64 bits data will be given to LEDs. Then unique identification number of RFID tag will be displayed on the LEDs on FPGA board.

2.2.3 Keyboard

The keyboard is interface d to FPGA module through UART, the inputs given from the keyboard is received by the UART Receiver and it converts into 8-bit parallel data and is displayed on the 7-segment LED display interfaced to FPGA kit.

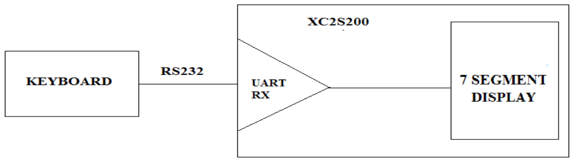


Figure 2.10 Keyboard Interfaced to FPGA

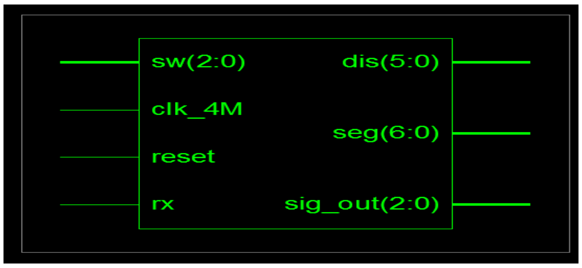


Figure 2.11 RTL Schematic of the Keyboard Interfaced to FPGA Board

2.3 Mode Selection

Table 2.1 Mode Selection

MODE SELECTION BITS	MODE SELECTED
001	Multi tasking
010 010 & 001 010 & 010 010 &100	Request/acknowledgement Keyboard RFID Reader GSM Modem
100	Round – Robin

2.3.1 Multitasking Mode

Multitasking is a method for sharing system resources so that multiple processes can appear to run simultaneously. Multitasking is a method where multiple tasks, also known as processes, share common processing resources such as a XC2SPQ208 processor.

When inputs from the dip switches of the FPGA board are 001, multitasking mode is selected. In this mode all the tasks (applications) run simultaneously i.e. the inputs from the keyboard is displayed on the seven-segment display, the UNI of the RFID tag which is read by the RFID reader is displayed on the LEDs of the FPGA board and the message received by the GSM modem is displayed on the LCD display of the board.

2.3.2 Request / Acknowledgement Mode

In this mode application (transmitter) to be served sends a request to the processor before transmitting the data. And if the resources are free and ready to receive the data then it sends an acknowledgement signal to the transmitter. Then the connection between transmitter and receiver will be established.

When the inputs from the Dip switches of the FPGA board are 010, request/ acknowledgement mode is selected.

When inputs are 010 & 001, the keyboard application is served, the inputs from the keyboard are displayed on the 7-Segment display, and all other applications are ignored.

When inputs are 010 & 010, the RFID Reader is requesting for the service, and it is served, the UNI of the smart cards which is read by RFID reader is displayed on the LEDS.

2.3.3 Round-Robin Mode

In a round-robin system several processes are executed sequentially to completion, often in conjunction with a cyclic executive. In round-robin systems with time slicing, each executable task is assigned a fixed-time quantum called a time slice in which to execute. A fixed-rate clock is used to initiate an interrupt at a rate corresponding to the time slice. The task executes until it completes or its execution time expires, as indicated by the clock interrupt. If the task does not execute to completion, its context must be saved. The task is the placed at the end of the executable list. The context of the next executable task in the list is restored, and it resumes execution. Round-robin systems can be combined with pre-emptive priority systems, yielding a kind of mixed system.

In this proposed system a time slice of sixty seconds is implemented, each tasks run for sixty seconds later they are preempted. Priority has to be assigned, keyboard is assigned higher priority, RFID reader second and GSM modem the lower priority.

When this mode is selected, all the tasks run sequentially for the assigned time interval based on the assigned priority.

4. CONCLUSION AND FUTURE SCOPE

This paper has demonstrated hardware implementation of multitasking strategies on commercially available FPGAs. It has shown how three modes of operations namely multitasking mode, request\acknowledgement mode and round-robin mode are implemented and tested and verified with the theoretical results, performance characteristics of the system have been described.

This paper concretely shows that multitasking can be applied to reconfigurable hardware and that is does in fact result in increased utilization, Not only does this work demonstrate such a system, but shows that it can work with today's commercial available FPGAs, without requiring any hardware modifications.

The multitasking FPGAs described would provide an excellent basis for complete reconfigurable system, further research would be needed to design a complete multitasking reconfigurable computer that could take advantage of these FPGAs while providing additional board features, such as memories, FIFOs, configurable controllers etc., that lead themselves to easy multitasking.

When inputs are 010 & 100, the GSM Modem is requesting for the service, and it is served, the message received by the modem is displayed on the LCD display.

REFERENCE

1) Using Hardware Context- Switching to enable a multitasking reconfigurable computer system. Wesley.J.Landaker. Department of Electrical and Computer Engineering Master of Science. | 2) W.S. Carter, K.D.R.H Freemam, H.c.Hsich, L.Y.Ngo and S.L.Sze, "A User programmable reconfigurable logic array "in proceedings of the IEEE 1986 custom integrated circuits conference, May 1983. | 3) S.M.Scalera and J.R.Vazquez "The design and implementation of context-switching FPGA" in proceedings of IEEE workshop in FPGAs for custom computing machines. | 4) S.Hauck "The roles of FPGAs in reprogrammable systems" in preceedings of the IEEE, April 1998. | 5) S. Brown & J. Rose "Architecture of FPGAs & CPLDs: A tutorial "In IEEE Design & test of computers, 1996. | 6) T.C.Waugh, "Field programmable gate array key to reconfigurable array outperforming supercomputers" in processing of the IEEE 1991 custom Integrated circuits conference. | 7) Fast online task placement on FPGAs: free space partitioning and 2D-hashing Walder, H. Steiger, C. Platzner, M. Comput. Eng. & Networks Lab., Swiss Fed. Inst. of Technol., Switzerland. | 8) An Evolutionary Approach to Dynamic Task Scheduling on FPGAs with Restricted Buffer Martin Middendorf, Bernd Scheuermann Hartmut Schmeck and Hossam ElGind. | 9) An Approach to Task Allocation for Dynamic Scheduling in Reconfigurable Computing Systems M. Ashraf Chughtai, Arjumand Yaqoob. | 10) J.D.Hadley and B.L.Hutchings "Design methodologies for partially reconfigured systems" in proceedings of IEEE workshop on FPGAs for custom-computing. | 11) M. Simmler Levinson, L. Manner, R. Sessler "Multitasking on FPGA coprocessors"in FPL,2000. | 12) Partition Based Dynamic 2D HW Multitasking Management Roman, S. Mecha, H. Mozos, D. Septien, J. Dept. Arquitectura de Computadores y Automatica, Univ. Complutense de Madrid. | 13) J. Vuillemin , D.Berlin, Dr.Roncin, M.Shand, H.Touati, and D.Bouccrd "Programmable active memories reconfigurable systems come of age "IEEE transactions on VLSI system. | 14) Context saving and restoring for multitasking in reconfigurable system Kalte, H. Porrmann, M. Sch. of Comput. Sci. & Software Eng., Univ. of Western Australia, Crawley, WA, Australia. | 15) M.Gokhale,W.Holmes,A.KopserS.Lucas,R.Minnich,D.Sweety and D.Lopresti "Building and Using a highly parallel programmable logic array" IEEE Computer. | 16) A Highly Integrated and Flexible Joint Test System Based on DSP/FPGA FPGA Jin Dang, Fenglei Ni, Yikun Gu, Minghe Jin, Hong Liu. | 17) R.Amerson, R.Carter, W.Culbertson, P.Kuckes, G.Snider and A.L."Plasma: An FPGA for million gates system,"in ACM\ SIGPA International Symposium on Field Programmable Gate arrays, Monterey,CA.Feb 1996. | |