Hardware IP for Scheduling of Periodic tasks in Multiprocessor Systems

HABIBULLAH JAMAL and ZEESHAN A. KHAN
Department of Electrical Engineering
University of Engineering and Technology, Taxila
Pakistan
drhjamal@uettaxila.edu.pk, zakhanpk@yahoo.com

Abstract: - The article presents an Intellectual Property (IP) for scheduling of multiprocessor systems that is designed using FPGA. The purpose of the IP is to minimize the processor time for scheduling activity by implementing the functionality in hardware. The algorithm implemented clusters the similar tasks on the same processor. Processors not in use are switched to power saving mode and they are only turned on if there is no other processor to take on the required activity.

Key-words: - FPGA, Hardware scheduler, multiprocessor systems, real-time systems, scheduling algorithm.

1 Introduction

A Real-Time Operating System (RTOS) allows processing of real-time applications in an optimal and timely manner. In an embedded multiprocessor system, RTOS is required to schedule the activities of the processors. Real time systems are categorized as hard real time and soft real time systems. Hard real time is a property related to the timeliness of a computation in the system. A hard real time constraint in the system is one for which there is no value to a late computation and the effects of a late computation can be catastrophic to the system. A soft real time system can tolerate some late answers to soft real time computations [1].

A real-time operating system (RTOS) must guarantee that scheduling is done feasibly given sufficient capacity is available [2]. A hardware Intellectual Property (IP) can be used for implementing the scheduling, inter-process communication (such as semaphores) and time management control (such as time ticks and delays) from the software OS-kernel to hardware unit. [3]. In this way, overhead can be significantly reduced by migrating kernel services to hardware. This will significantly improve the response time by increasing the CPU utilization [2].

In the area of real-time systems, scheduling algorithms have been implemented in hardware [4, 5]. Most of the hardware implementations have used only one scheduling criterion [6] while few of them have proposed dynamic reconfiguration to include more than one scheduling criteria implementation [7].

Here, our target is to design and implement multiprocessor hardware scheduler to be used in VoIP applications where several IPs for voice encoders/decoders have been developed and we schedule different tasks so that similar tasks are clustered onto one processor/IP increasing its average utilization.

The paper consists of four sections. In the second section, multiprocessor hardware

scheduler architecture is presented. In the third section, implementation and experimental results for the coprocessor are discussed. Finally, the fourth section concludes the paper.

2 Multiprocessor Hardware Scheduler

Multiprocessor hardware scheduler (MHS) is designed to handle the scheduling of non-resident tasks in multiprocessor systems. Non-resident task is the one whose code is fetched from the specified memory location. Our scheduler is designed for scheduling onto four homogenous processors with similar computational capacity.

The architecture consists of a processor selection controller which selects the processor such that tasks of similar type are scheduled on same processor so that code fetches are minimized and memory does not becomes the bandwidth bottleneck.

The block diagram of the architecture is shown in Figure 1. Main components of the scheduler are:

1) Task Table
2) Processor Selection Controller
3) Processor's Task Queue
4) Processor Status Table

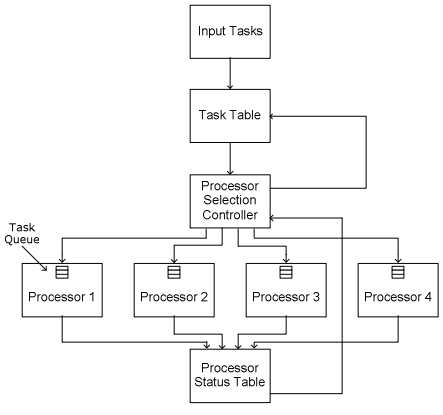

Fig. 1: Block Diagram of Multiprocessor Hardware Scheduler

2.1. The Task Table

The architecture consists of a task queue that contains the input tasks with respect to their deadline time. Each task table entry is of 32 bits and architecture can support a maximum of 500 task entries in its table. This determines the maximum number of task entries that will be processed by the scheduler. The input task table entry contains all the information necessary during the scheduling process such as worst case execution time (WCET), task type and memory location of non-resident tasks. The format is shown in Figure 2. Our scheduler supports up to five types of tasks. That is why, each task is specified by 3 bits and 5 bits specify worst-case execution time in number of clock cycles. All the tasks are non-resident tasks so the code is to be fetched from the 24 bit memory location specified in the input task entry.

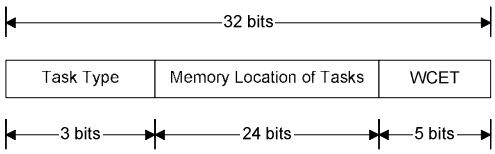

Fig. 2: Input Task Format

2.2. The Processor Selection Controller

Once a task table entry is given as an input to the processor selection controller, the controller selects the optimal processor queue. If the current task is specified by the type PRESENT, then the controller prioritizes the processors with respect to descending number of PRESENT tasks present in their queues. All the processors are searched for the required time slot with respect to the mentioned priority. If no slot is found in the highest priority processor queue, it abandons the search on that processor queue and the next processor queue is searched for the time slot. Once the required slot on a processor queue is found, it abandons the search and schedules

the task on that time slot. The main theme of the algorithm is to minimize the number of code fetches by scheduling the tasks of same type onto one processor.

2.3. The Processor's Task Queue

Each processor task queue contains the tasks that are to be scheduled on that particular processor. We can have a maximum of 50 task entries in the queue of each processor. Once a processor completes the execution of a task, the next task in the queue is scheduled on the processor. This way the processing of all the tasks in the queue continues in a first-in, first-out manner.

2.4. The Processor Status Table

The processor status table helps in making the right decision for selecting the processor. It is basically a storage element that contains information for number of tasks of each type present in the queue of each processor. A typical entry of the table is shown in Figure 3. The data of status table is updated when a task is either removed or entered in the queue of the processor.

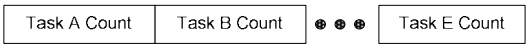

Fig. 3: Processor Status Entry Format

3 Implementation and Experimental Results

The architecture described in previous section is successfully synthesized and simulated using Xilinx ISE and ModelSim. The results obtained are briefly described in the following sections.

3.1. Synthesis Results

The prescribed MHS architecture is implemented using Xilinx ISE tool for Spartan III xc3s1000 device. For the sake of synthesis and simulation, five different types of tasks (namely type A, B, C, D and E) are supported by the multiprocessor hardware scheduler those were to be scheduled on four different processors. The outcomes of the synthesis are presented in table 1 and 2, respectively.

TABLE 1
DEVICE UTILISATION SUMMARY

Selected Spartan III Device 3s1000ft256-4			
Parameter	*Utilized*	*Total*	*%age Utilized*
#Slices	284	7680	3%
#Flip Flops	240	15360	1%
#4 input LUTs	545	15360	3%
#Bonded IOBs	33	173	19%
#GCLKs	1	8	12%

TABLE 2
MACRO STATISTICS FOR HDL SYNTHESIS REPORT

#FSM	1
#Registers	20
#Latches	575
#Counters	4
#Shift Registers	44
#Multiplexers	20
#Adders/Subtractors	74
#Comparators	1

3.1. Simulation Results

The synthesized architecture is simulated using ModelSim and the results show that similar type of tasks tends to be scheduled on one processor queue. This is evident from simulation waveform for 1st Processor Queue given in Figure 4. This figure shows that at the present moment most of the tasks present in the 1st processor queue are of type D while tasks of type C and E are also present in

minority. Similar pattern is seen on the queue of other three processors [8].

Figure 5 shows real-time simulation graph of number of tasks in the queue of each processor from 0-12.5µsec. The result clearly shows that fair amount of tasks are given to each processor so that queue of any processor does not become the bandwidth bottleneck.

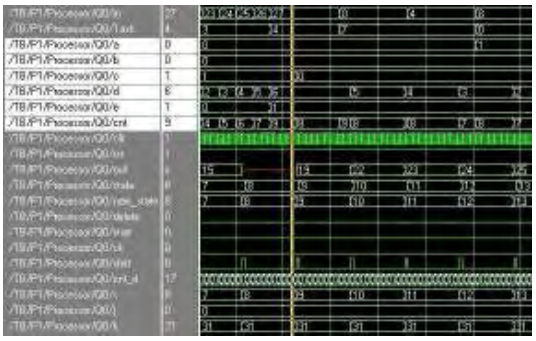

Fig. 4: Simulation for Input Queue of 1st Processor

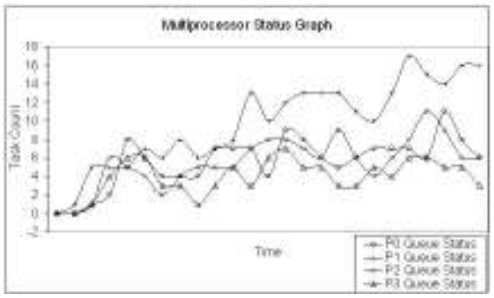

Fig. 5: Multiprocessor Status Graph

4 Conclusions

A hardware implementation of a scheduling coprocessor on an FPGA for multiprocessor system is presented. The hardware scheduler schedules the task so that code fetch overhead is minimized by clustering similar tasks onto one processor. The performance profile of the algorithm is promising enough to inspire more work in this field.

References:

[1] Steve Furr, "What Is Real Time and Why Do I Need It?" available online at http://www.qnx.com.

[2] Vincent J. Mooney III and Douglas M. Blough, "A Hardware-Software Real-Time Operating System Framework for SoCs," IEEE Design and Test of Computers, pp 44-51, November-December 2002.

[3] A. Daleby and K. Ingström, Technical Reference Manual for RTU Operating System Accelerator, Västerås, Sweden, 2002.

[4] S. Moon, J. Rexford and K. Shin, "Scalable hardware priority queue architectures for high-speed packet switches," in Proceedings of IEEE Transactions on Computer, vol. 49, no.11, pp.1215 –1227, November 2000.

[5] D. Picker and R. Fellman, "A VLSI priority packet queue with inheritance and overwrite," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 3 no. 2, pp. 245–253, June 1995.

[6] Byung Kook Kim and Kang G. Shin, "Scalable Hardware Earliest-Deadline-First Scheduler for ATM Switching Networks," in Proceedings of 9th Real-Time Systems Symposium, December 1997, pp 210-218.

[7] P. Kuacharoen, M. Shalan and V. Mooney, "A Configurable Hardware Scheduler for Real-time Systems," in Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms, 2003.

[8] Zeeshan Ali Khan, "A Hardware Implementation of Scheduling Scheme for Multiprocessor Systems," M.Sc. thesis, Department of Electrical Engineering, University of Engineering and Technology, Taxila, Pakistan, 2007.