Indian Journal of Science and Technology, *Vol 7(12), 2015–2019, December 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Implementation of Multi-threading on Hybrid ARM Cortex Dual Core A9–FPGA architecture for Energy Efficiency

V. Jean Shilpa* and P. K. Jawahar

Department of Electronics & Communication, B. S. Abdur Rahman University, India;
jeanshilpa@bsauniv.ac.in, hodece@bsauniv.ac.in

Abstract

One of the greatest challenges of FPGA is the crave for a platform, which runs a system, handles a software and co-ordinates exchange with peripherals. To explore this advantage on FPGA with energy efficiency, this work aims at developing multithreading on ARM cortex dual core A9 processor present in Zedboard FPGA. To enhance the speed of execution on this processor and to allow concurrency, multithreading is developed using intel thread building blocks(TBB) library which aids in creating task based parallelism on FPGA being the first of its kind. In the first step towards this process, the processor is booted with Ubuntu 12.04 linux operating system making it as a standalone processor. This system is enabled with graphical user interface using xillinux IP library FPGA code kit. To test the efficiency of multihreading on the processing system of Zedboard, an application known as 2D – raytracer is developed using the parallel-for loop scheduling method which aids in running all the iteration loops in the code into chunks and runs each chunk as a separate thread under intel TBB. In this application an image was parallelized by speculating each pixel running in parallel resulting in excellent speedup. The newly implemented multithreading achieves a speedup of 53% compared to sequential execution on the same processor. This method explores the flexibility of parallel processing on FPGA.

Keywords: Dual Cortex A9 ARM(Advanced RISC machines) Processor, Field Programmable Gate Array(FPGA), Intel TBB(Thread Building Blocks), Software Development Kit(SDK), Xilinx Platform Studio(XPS), ZedBoard(Zynq Evaluation and Development Board)

1. Introduction

Traditional CPU designs employed methods such as increased clock frequency and silicon size to enhance the performances of CPU designs but at the expense of limitations on power wall, memory wall, instruction level parallelism and time to market[1]. To solve this issue the next generation of multi-core processors hit the market to achieve extremely high computational requirements and to aid many computation intensive applications such as multimedia, physical simulation and financial modeling[2], but at the expense of time to market. Additionally embedded and real time system designers are continuously challenged to provide systems to satisfy price/performance ratios[3]. To reduce time to market, with good amount of price/performance ratio, this work aims at developing new techniques of energy efficiency and parallelization on the Zedboard FPGA chip[4]. The demand of today's technology craved for a system that has a fusion of processing system and a programming logic on a single board with less cost, which is full-filled by Zynq – 7000 all programmable system on board popularly known as zedboard[11,7]. Zedboard is a Zynq evaluation and development board which uses ARM cortex dual A9 processor[12] in the processing system of FPGA, configurable by the user. Hence as the

*Author for correspondence

processors capabilities have increased, the pressure on software developers began to invent methods to always keep the processor busy, since the processors wasted much of the time running single tasks and waiting for certain events to complete[5], but as the cost and time involved in designing multi core processors with multithreading are huge. We go for a new design on Zedboard aiming at developing multithread processing on Zedboard using Intel thread building blocks which is a library enabling task based parallelism to improve the processing speed and the utilization of FPGA as a multi core processor with less time to market and aiding hardware developers to go with thread level parallelism design on the re-configurable processors like FPGA for the first time.

2. Design Methodology

2.1 ZedBoard – Processing System

This work is developed on Zedboard which is known as a programmable system on chip[13].One of the important peripherals of Zed board is 28-nm ARM cortex dual – A9 processing system[8], this processor runs at 1GHz programmable logic. The two cores are hard IP components embedded on the Zynq processing system[9]. The Figure 1. shows the architecture of ARM cortex dual A9 processor which is hard core on the Zedboard, as a first step in our project an OS will be booted on this processor whose image will be loaded on the SD card.

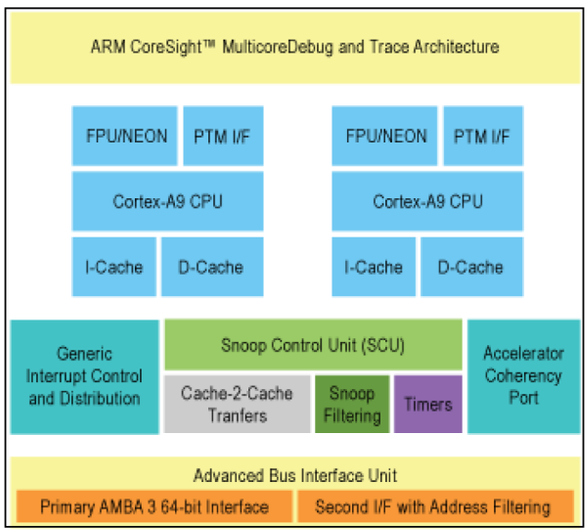

Figure 1. Cortex-A9 Dual MP Core Architecture.

2.1 Booting OS from Zync – 7000 series

To boot the operating system for the processor, here we use SD(Secure Digital) card slot which essentially requires four items to be placed on the SD card namely linux kernel image, the linux file system, a BOOT.BIN file and compiled device tree[11]. To complete this function we require a PC running with Linux distribution. To begin this work we have installed Ubuntu 12.04.To format the SD card for the purpose of booting the dual cortex processor with embedded operating system, we first partition the SD card into two portions, one for having the FAT file system which atleast requires 1GB and the second partition contains a ext4 file system, which requires atleast 3GB. Figure 2. shows the partition made on the SD card. As the first step in formatting the SD card device , the device node is identified, after which all the previous partitions are removed. This new method avoids wired connection from PC to boot the processor on the Zedboard.

The next process is to install the custom drivers essential for the zedboard peripherals. Prior to this process ARM GNU tools chain will be installed and then the BOOT.BIN file, which configures the programmable logic and the processing systems in both the partition. After which the ZedBoard Linux Hardware Design is developed which enables linux to communicate with the peripherals on the Zed board. Finally the device tree which is a data structure of the hardware peripherals is installed for the linux kernel to understand the drivers for every node, hence during the boot up this device tree initializes the drivers for all the nodes. The SD card is now inserted into the Zedboard. And the jumper settings are made as specified MIO 6: set to GND ,MIO 5: set to 3V, MIO 4: set to 3V, MIO 3: set to GND, MIO 2: set to GND, VADJ Select: Set to 1V, JP6: Shorted ,JP2: Shorted and all other jumpers should be left unshorted. To configure the terminal settings as Baud is set to 115200.Since we're using a Linario

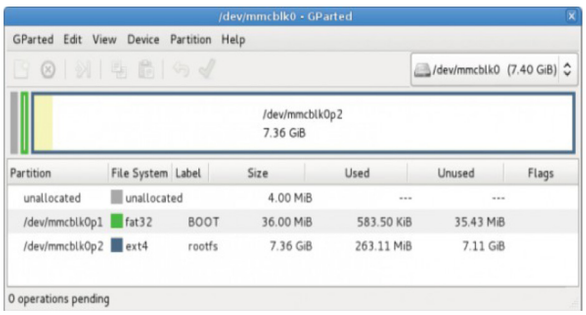

Figure 2. Partition of SD Card.

system we connect peripherals such as HDMI, mouse, keyboard and USB to USB OT hub. Waiting for the boot process to complete, now the linux system will be running on the Zedboard.

To enable the FPGA board with graphical user interface, xillinux software which is an FPGA code kit is installed with the linux. Figure 3. shows the booting of Ubuntu with Xillinux from SD card.

2.3 Configuring the Design of Processing System using Xilinx Plan Ahead

This work is developed with Xilinx14.2 version. Initially the board support package for the zedboard is loaded into the too[15]. The specification for the processor, interfaces, timing and address ranges are generated by XPS. The tool outputs the HDL netlist of the processing system which is fed as input to the ISE(Integrated software environment) to generate the processing system bit file. Then the Plan ahead generates the RTL netlist for the processing system. Now the XPS runs software application on the processing system by employing the software development kit of Xilinx EDK. In the process of development of multithreading applications, language of programming is selected as C++. Figure 4. clearly indicates the image of processing system ARM cortex dual A9 processor generated by plan ahead tool of Xilinx 14.2.

2.5 Parallelism by MultiThreading on Zedboard with Intel TBB

Multithreading on FPGA is developed with the principle of coding the program written for an application in a

Figure 3. Booting of Xilinux from SD card.

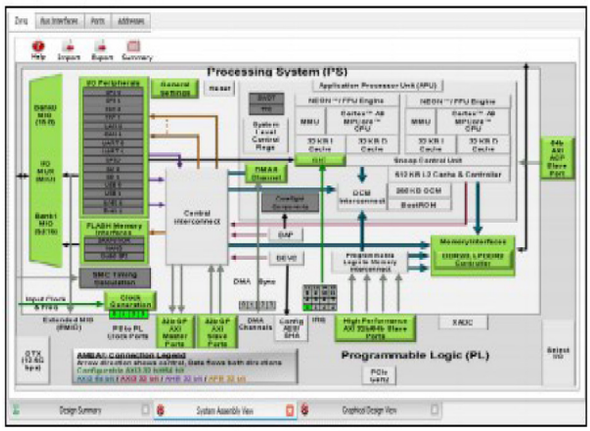

Figure 4. XPS generated ZedBoard Processing system assembly view.

such a way were in instances of its routine called threads can execute concurrently. Instead of creating threads and manipulating them, this burden is left to the multithreading library. In this design intel thread building blocks library is used for the first time on FPGA providing a rich and complete approach to explore parallelism in hard core processors[17]. Intel TBB is not just a multithreads library rather, it represents task based parallelism that abstracts platform details and threading mechanisms for performance and scalability. Intel TBB is a library that supports scalable parallel programming using standard C++ template. Additionally, it fully supports nested parallelism, so you can build larger parallel components from smaller parallel components. To use the library, tasks are specified in the application developed, not threads, and let the library map tasks onto threads most efficiently reducing the burden of the designer. In this process of implementing multi threading on Zedboard and testing its efficiency, we have developed an application known as 2D raytracer, using parallel_for loop scheduling method available in intel TBB. The syntax of the parallel_for employed to design the 2D raytracer is given below

#include “tbb/parallel_for.h”
Func parallel_for(Index first,Index_type last,Index step,
constFunc& f[, partitioner[, task_group_context& group]]);

This parallel_for loops recursively splits the range into subranges to the point such that is divisible. When worker threads are available, parallel_for executes iterations is

non-deterministic order. This application takes an image input and it is parallelized by speculating each pixel running in parallel resulting in excellent speedup. The output screens are show along with time of execution of single thread, dual threads and multi threads. Figure 5 shows the execution of sequential thread without adding any parallelism techniques. Sequential the image 2D raytracer is executed with thread level parallelism on ARM cortex dual A9 processor. This specific image contains 7386 objects.

Next the application is designed to split itself and run on the available number of processors. Being ARM cortex dual A9 processor, number of threads created are dual. Figure 6. shows the parallel version that uses intel TBB's parallel_for loop and block_range to parallelize tasks. This develops group of scan lines to run on the available number of processors. The figure clearly indicates two threads running simultaneously on the processors. For this application in the program the TBB_NUM_THREADS environment variable sets the desired number of threads and the TBB_GRAINSIZE sets the number of grains in the image. Grain size refers to number of scan lines running in parallel.

In the third process instead of parallelizing the scan line, the number of tasks are increased by employing intel tbb with blocked_2D functions to parallelize the tasks into rectangular block.Figure 7 shows the parallel version code with increased grain size and number of threads.

The graph in Figure 8 shows that with single thread implementation, the time take to open the image is higher

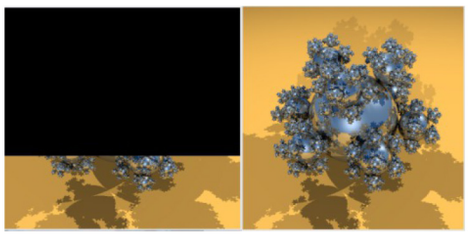

Figure 5. Sequential thread Execution.

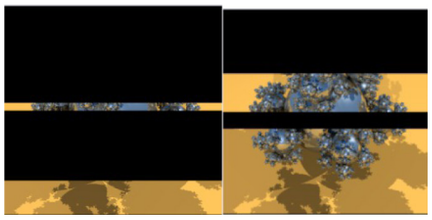

Figure 6. Dual thread Execution.

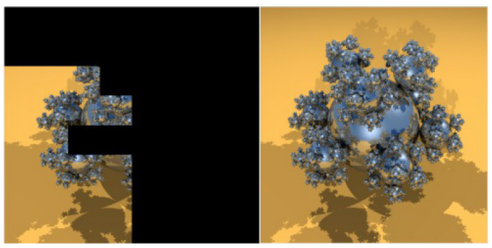

Figure 7. Multi thread Execution.

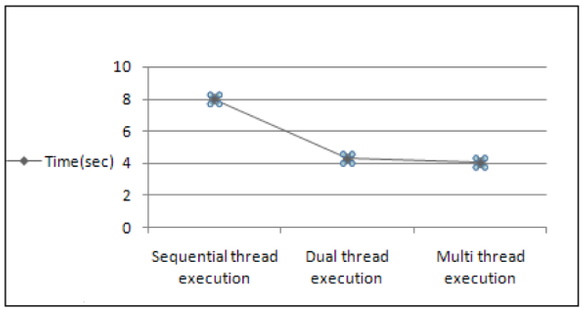

Figure 8. Graphical view Time Taken for Execution with multi threads.

than multithreads. Calculating from the graph the time saved by imlementing multi threads is 53% using intel TBB instead of using the conventional multithreading libraries. Hence the processing time is reduced more than double the time, which is a very great advantage of the hybrid ARM – FPGA unit. The execution time for running the applications is shown in Table 1.

4. Conclusion and Future Work

In this article, the efficiency for multi threading on Zedboard is explored very efficiently. This can be further extended to image processing applications using extensive parallelism on FPGA. Secondly this design was only implemented on the processing system portion of zedboard, but the programmable logic portion of the Zedboard is left free, hence forth to increase the number of processors on Zedboard, multiple softcore processors can be invoked with the programmable logic part of FPGA and communicate with the hardcore ARM cortex dual A9 processor. Thirdly with linux installed any C++ application which can make extensive use of parallelism can be easily implemented on Zedboard.

Table 1. Execution time of threads

Type of Execution	Time taken for execution in (sec)
Sequential thread execution	7.969
Dual thread execution	4.268
Multi thread execution	4.022

5. References

1. Asanovic K, et al. The landscape of parallel computing research: A view from Berkeley. Technical Report, EECS Department, University of California, Berkeley, 2006 Dec. Report no: UCB/EECS-2006-183.
2. Tsoi KH, Axel WL. A heterogeneous cluster with fpgas and gpus. acm 978-1-60558-911-4, fpga'10, 2010 Feb 21–23.
3. Andrews D, et al. Programming models for hybrid FPGA-CPU computational components: A missing link. IEEE Computer Society, 2004.
4. Fletcher BH. FPGA Embedded Processors: Revealing True System performance. Embedded Systems Conference; 2005; San Francisco.
5. Intel® Hyper-Threading Technology. Technical User's Guide. 2003 Jan.
6. Kingyens J, Steffan JG. The Potential for a GPU-Like Overlay Architecture for FPGAs. International Journal of Reconfigurable Computing. 2011.
7. Russell M, Fischaber S. OpenCV Based Road Sign Recognition on Zynq. 11th IEEE conference on industrial informatics; 2013.
8. Pham KD, Jain AK, Cui J, Fahmy SA, Maskell DL. Microkernel Hypervisor for a Hybrid ARM-FPGA Platform. IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP); 2013.
9. Daul-Core ARM cortex MPCore Processor. 2012 Oct.
10. Available from: www.xillybus.com
11. Zync Evaluation and Development.Hardware user's guide. Version 2.2, 2014.
12. Zynq-7000 All Programmable SoC Technical Reference Manual. Xilinx Inc; 2013.
13. Getting Started With Embedded Linux – ZedBoard, 2013.
14. Vincke R, Messiaen A, Boydens J. Hybrid FPGA/Multi-core CPUs for Industrial Applications. Annual journal of electronics. 2013.
15. Xilinx. Plan ahead user guide(UG632) 2012.
16. Ababei C. Speeding Up FPGA Placement via Partitioning and Multithreading. International Journal of Reconfigurable computing. 2013; 2009:1–9. Article ID 514754.
17. O'Reilly, Intel thread building Block, Outfitting C++ for multicore processor parallelism, 2007