Design Of User Level Threads For Multi-Core Processors Implemented On FPGA

V Jean Shilpa, Dr P K Jawahar

1. Research Scholar, Department of Electronics and Communication, B.S.Abdur Rahman University, India
2. Professor and Head, Department of Electronics and Communication, B.S. Abdur Rahman University, India

Abstract— The era of Energy efficient high performance computing seems to be increasing at a very high rate. Multi core processors play a key role for high performance computation. Multi core processors employ the idea of spreading concurrency through out the system. Multi – core and Multi – threaded processors both together exploit concurrency by executing multi threads on multiple processors. Hence an operating system provides multi threads for parallel applications. Since the scheduling process is hard coded in the operating system, it is unknown to the user. A different category of threads known as user level threads which reside in the user mode and the one which avoid kernel context switching will be implemented on FPGA soft core processor known as microblaze to effectively schedule threads based on the inputs from the application loaded by the user based on priority based scheduling using Open MP API(application program interface) to enhance the scheduling capability of user threads and allow user level threads to effectively utilize the advantage of concurrency in multi core processors and multi core embedded processors. Implementation of user level thread parallelism on FPGA can also be effectively used for embedded. processors which employ multi core processors.This Open MP user level multi threads will be designed using the tool XPS(Xilinx platform Studio) and SDK(software development kit) provided by Xilinx 14.5 and implemented on microblaze soft core processor embedded in sparten 3E starter FPGA kit[4] and interfaced with the kernel to effectively utilize the advantages of multi core processors and multi core embedded processors

Index Terms— API(Application program interface), SDK(Software development kit), TLP(thread level parallelism), XPS(Xilinx platform studio.

1 INTRODUCTION

Parallel computing achieved its high performance when implemented on multi core processors[1]. Performance of a multi core processors mainly depends on time it takes to execute a given task.

Perforamce = Instruction executed per clock cycle × Frequency

Hence to increase the performance increasing the instruction executed per clock cycle and frequency concurrently is an trivial idea but developing new architectures that support high level of parallelism is essential and architectures that aid parallelism efficiently to adapt to the increasing number of core in multi processor system has become essential.

Especially designers working with real time application running on multi core processors are required to provide high computational capabilities. FPGA is treated as an component or accelerator that can be reused for wide range of applications[2]. It posses the characteristic of both specialization as well as generalization to any application. FPGA's are embedded with processors known as embedded processor in FPGA[3] .

Primarily ILP(instruction level parallelism technique was applied to achieve pipelining, then the era shifted to thread level parallelism which encouraged parallelism to be implemented on processor. Instead of implementing TLP on one single core this was adapted to multi core architecture because multiple streams of instruction can be executed on multiple processors[4]. The creation , management and scheduling of threads which is defined as the smallest sequence of pre- programmed instruction that is independently managed by the operating system scheduler. Threads always resides in side a process, its a light weighted process. These threads which are created inside an operating system are executed concurrently on multi core processors. Such kind of threads are know as kernel level threads[5]. The next category of threads are known as user level threads. These threads work with the support of user thread library which reside in user application address space. The user level threads do not interfere with the kernel. All the threads that are created inside a user level mode are put together as process, and they make a system call to the operating system.This research mainly deals with developing an user level threads on FGPA soft core processor called microblaze to effectively schedule threads based on the inputs from the application loaded by the user based on priority based scheduling.
This design on FPGA gets certain inputs from OS which includes, the total number of cores available, and the current status

- *V Jean Shilpa is currently pursuing PhD program in embedded system design inB S Abdur Rahaman University,India, PH-9655537103. E-mail: jeanshilpa@mail.com*
- *Dr P K Jawahar Professor and Head of the department inelectronics and communication engineering inB.S. Abdur Rahman University,India.Email: HOD_ECE@bsauniv.com*

of the core and based on the status it will effectively schedule the user level threads so that work has to be equally distribute to the multiple cores to increase the over all speed and give the output to the OS .

2. USER LEVEL THREADS ON FPGA

2.1 User Level threads

To implement parallel application operating system has provided with kernel level threads. The burden on operating system is increasing day by day since more number of distributed system , multi media processing etc has to be managed by the operating system. The demand of an operating system to get adapted to the various hardware platforms connected to it is the need of the hour. Also the performance of any application being executed in a parallel environment like multi core architectures mainly depends on the capability of the scheduler present in the multi threading unit. The different categories of scheduling mainly are categorized as follows: First, Multi media applications which demands real time scheduling and the quality of service of the scheduling maily relies on the parameters of the appilications availability. Second, Threads scheduling for servers because fairness metrics of different jobs have to be taken into consideration for efficiently scheduling different jobs on the server. Third, Thread scheduling for distributed user application which maily deal with shared memory, message passing, because these kind of applications block the kernel until the applications responds .Foue application threads which involve heavy work load, a scholar which can avoid page faults and page thrashing is nessecary.Five, threads have have very less work load ,the nature of these threads are they are too many in number, hence they have to efficiently occupy the space in the cache.Hence the task of managing all the above functions requires a good user level thread which does not interfere with the kernel and perform scheduling by carefully monitoring the application behaviour in the system and effieciently interact with the applications to kep track of its states and the tasks the threads are supposed to full fill. Implementing thread at the kernel level is costlier compared to user level because user level threads are created at user level mode which is operating system independent, language independent, application independent. These user level threads are created and maintained at the user level threads[5][6], there is no intervention with the kernel.

2.2 Open MP

For Open MP is a portable application program interface for parallel programming model in multi processors with shared memory. Open mp is an industry standard for shared memory multi programming[7]. Open MP is a scalable and portable model that gives multi core processors a flexible interface for the development of parallel applications in high performance computing areas. Open Mp works on the basic concept of Pragmas which is otherwise known as active components which effieciently aids in parallelizing the code. This Open MP can is not processor specific. Hence open MP is defined as a set of run time library routines, compiler directives and environment variables which is used to program shared memory multi core using C/C++ and Fortran. The programming model it employs is thread based shared memory .The variables in Open Mp can be both shared as well as used as private variables. Parallelism in multi programming works on the principle of fork join model.The number of threads to be executed on the parallel region depends on the entry to the parallel region which resides in FPGA[8]. A parent thread creates a team of chidren thread which resides in the thread pool for reuse in parallel programming.

2.3 FPGA FOR USER THREAD LEVEL PROGRAMMING

Field Programmable gate array are used as accelerators whose application support can be retargeted or reconfigurable. Recently emerging CPU and FPGA hybrid design have significant economics to scale according to the technology needs, were significant amount of work can be pushed to FPGA since it supports retargetable applications, since the number of processor cores are increasing day by by, the support given by operating system for equally distributing the threads to multiple processor increases and hence the burden lies on the operating system. There for to simplify the tasl the user level threads are implemented on the FPGA to support and provide fast threads to the processors for computing.

2.4 FPGA EMBEDDED PROCESSORS

Embedded processors in FPGA are capable of running a full fledged real time operating system environment[9]. With regard to communication on board, FPGA embedded processors are well equipped with high speed data processing, high speed converters interfaces etc. Embedded processors in FPGA are classified as hard core and soft core processors. The well known hard core processor is power PC and the soft core processor is microblaze. Hard core's processor's architecture are fixed that cannot be reconfigured by the user were as soft core processors known as microblaze in Xilinx FPGA can be reconfigured according to the preference of the user. It posses a 32 bit Harvard RISC architecture.Micro blaze is a virtual microprocessor that is built by combining cores inside Xilinx FPGA. The speciality of the architecture of microblaze is , it can be tailored up to our needs. It has a separate 32 bit instruction and data bus use to access program and data from the memory. It has 32 general purpose registers,32 – bit instruction word with three operand and two addressing modes. Special 32 bit instruction and data bus that has direct connection to on – chip block RAM through LMB, 32 – bit address bus,single issue pipeline, instruction and data cache, hardware debug logic, fast simulink uni directional bus support. Also it has the advantage of pipelining. The user is given full freedom to decide on the size of cache memory, type of memory interface and execution units. The performance of the micro blaze processor mainly depends on the processor's configuration , speed grade and FPGA target architecture. Tools that aid to design these embedded processors are provided with push button board support packages which can be generated both from XPS as well as SDK.

2.4 Xilinx Platform Studio and Software Development Kit

XPS and SDK provided by Xilinx are very efficient tool for hardware and software integration. The product version of Xilinx used in our project is EDK 14.5, a new board support package environment is created, in which PLB(Processor local bus) which is a bus standard used with FPGA, the target board that is choose is Xilinx Sparten 3E Starter kit[10], the tool gives an option to

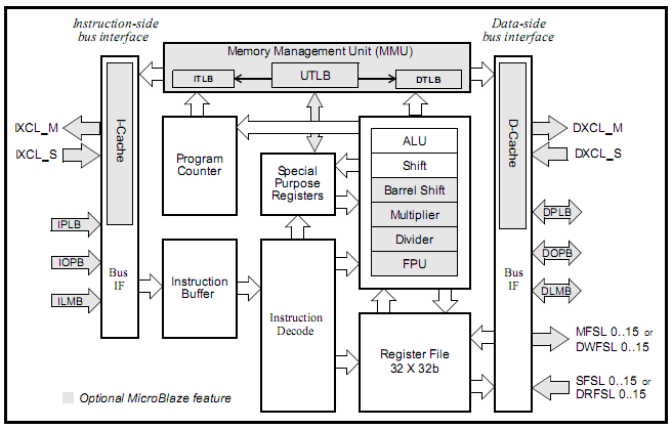

Fig 1: Block Diagram of Micro Blaze embedded processor

select between single core or dual core processors, we have choose sinlge core processor to implement the user thread unit., the frequency we have selected is 50 Hz and the processor is microblaze, the tool gives us the flexibility to vary the size of cache memory which supports the microblaze processor, hence 32KB is initially selected, the tool gives us the option of selecting other peripheral devices if required to be connected to the processor, here we have selected LMB(local memory bus).

Xilinx platform studio is a tool provided by Xilinx to design hardware portion of the embedded system. The system is built from the predefined libraryfor the cores. Tool provides the ability to integrate IP cores. The sytem constructed here includes a microblaze processor, local memory bus, fast simplex link, processor local bus. The lmb bus is used to control the dlmb controller and the ilmb control modules here microblaze processor acts as master and other modules acts as slave. The PLB bus aids in connecting the RS232 component to microblaze processor. Once the netlist and bitstreams are generated directly the software development kit(SDK) of Xilinx is launched from XPS which takes the hardware platform created. Three main files are exported from XPS, system.xml contains the details of the IP cores, system .bit caontains the programming file for FPGA and system .bmm file contains the information need by SDK to launch memory in the target board. In the software development kit of Xilinx a C project to create threads is developed, the header files included are as follows

```
#include<stdio.h>
#include <omp.h>
#include <xparameters.h>
```

Stdio.h and omp.h are the header files included to develop the threads creating program where in openmp is used to develop an API for writing multi threaded applications, xparameters.h file contains system parameters for the Xilinx device drivers environment.It gives the over all information about the number of devices in the system, memory map and parameters for each devices. The board support package (BSP) is exported from XPS to SDK which contains the nessecary library and the drivers form the lowest layer of application software stack. Hence the software application developed for threads creation and management runs on the board support package using the provided application program interface. After the software is developed again the board support package settings can be modified by system.mss.

Design Flow :

The Embedded Development kit provided by Xilinx enables to integrated hardware and software co- design using C/C++ languages or hardware description languages given in Fig 2.From the hardware side first the platform kit along with its board support package is selected. The peripherals, cache memory, external memory , the bus architecture etc are selected to support the hardware design and the corresponding hdl file is generated which contains the hdl code for the platform designed. The hdl code is furthered synthesized into gate level netlist and then matched to primitives mapped on to the specific device resources such as flip flops, look up tables and block memories. The interconnections and location and followed int he placement and routing step. The .bit file downloaded file or file which is exported to software development kit is generated. The software code written in C is compile along with the required header files which was previously mentioned and converted into executable and linkable format known as elf file. The other two files known as microprocessor hardware specification. Mhs and microprocessor software specification .mss file specify the hardware connections and software structure of the system.The whole system runs on a real time operating system.

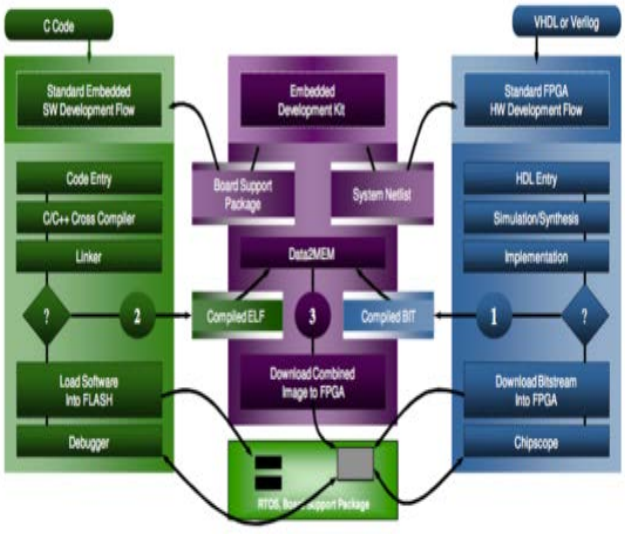

Fig2 : Xilinx EDK Flow Diagram

3 Experimental Results

An application to develop multi threads on microblaze was written in embedded C program with the supporting hardware platform created in XPS and application of creating these multi threads was designed and developed in SDK and the corresponding files generated as well as the netlist,the IP core and the output generated when the design was downloaded to FPGA are given below The overall summary of the microblaze processor memory system generated by the tool is given below. The list of IP blocks present in the design are microblaze, mb_plb, ilmb, dlmb, lmb_bram, rs232, clock_generator, proceesor_system_reset.um.

TABLE 1
Memory System file generated

Core Name	Instance name	Base Address	High Address
Processor 1	microblazr		
Lmb bram if cntlr	Dlmb cntlr	0x00000000	0x00007FFF
Lmb bram if cntlr	Ilmb cntlr	0x00000000	0x00007FFF
Xps bram if cntlr	Xps bram if cntlr 0	0x83A18000	0x83A19FFF
Xps_timebase_wdt	Xps_timebase_wdt 0	0x83A00000	0x83A0FFFF
Xps timer	Xps timer 0	0x83C00000	0x83C0FFFF

TABLE 2
Netlist and Bitstream file generated

Report	**XPS Synthesis Report**		
	Flip Flops Used	**LUT's used**	**BRAM's used**
System	1870	2568	8
System Clock generator wrapper	4	1	-
System mdm wrapper	152	382	
System plb wrapper	126	143	
System microblaze wrapper	1366	1834	
System rs232	147	137	

TABLE 3
The Board support package file generated

Configuration for OS : Standalone Name	**Board Support Package**		
	Value	**Type**	**Description**
Stdin	rs232	peripherals	Stdin peripheral
stdout	rs232	peripherals	Stdout peripherals
compiler	mb-gcc	string	Compiler used to compile both BSP/ library and applications

The output generated in the console window of Xilinx SDK is shown below

```
Hello World from thread = 0
Hello World from thread = 1
Hello World from thread = 2
Number of threads = 3
Hello World from thread = 4
Hello World from thread = 5
Hello World from thread = 6
Hello World from thread = 7
Hello World from thread = 8
```

The output clearly indicates that new threads to be created on multicore processor are created and ported to microblaze processor, futher the entire thread scheduling unit to support thread scheduling of multi core processors will be entirely designed and ported to microblaze and the same architecture will aid the multi core processors.

REFERENCES

[1] David A. Bader, Varun Kanade and Kamesh Madduri," A Parallel Programming Framework for Multicore Processors",Parallel and distributed Computer Architecture,IEEE Conference, 2007.

[2] Wadiyasoorya ,Hastikka "FPGA Implementation of heterogenous multi core platform with SIMD / MIMD custom accelerators" Circuits and System,IEEE,2012

[3] Xilinx Inc. MicroBlaze Reference Manual, version 10.1

[4] Hongtao Zhong, Steven A. Lieberman, and Scott A. Mahlke, "Extending Multicore Architectures to Exploit Hybrid Parallelism in Single-thread Applications",High performance Computer Architecture,IEEE,2007.

[5] Thomas Riechmann , Jürgen Kleinöder, "User-Level Scheduling with Kernel Threads "Citeseer,1996

[6] Multi-threading: concepts and application tuning, Technical white paper,Multi-threading: concepts and application tuning 2, HP OpenVMS technical journal V19

[7] "The OpenMP API specification", www.openmp.org .

[8] Daniel Cabrera, Xavier Martorell, Georgi Gaydadjiev, Eduard Ayguade, Daniel Jim´enez-Gonz´alez OpenMP extensions for FPGA Accelerators, Reconfigurable Computing, SAMOS'09 Proceedings of the 9th international conference on Systems, architectures, modeling and simulation pages17-24 IEEE Press Piscataway, ©2009

[9] Bryan H. Fletcher, "FPGA Embedded Processors Revealing True System Performance Embedded Systems Conference San Francisco 2005

[10] Spartan-3 Starter Kit Board User Guide, Xilinx, Inc.

[11] P. Waldeck, N. Bergmann, "Evaluating software and hardware implementations of signal-processing tasks in an FPGA", Proceedings - 2004 IEEE International Conference on Field-Programmable Technology, FPT '04, pp. 299-302.