UACEE International Journal of Advances in Electronics Engineering
Vol:1 Issue:1 ISSN 2278 - 215X

Design of Dual-Core Embedded System using LEON3 Processor

Rahul K. Hiware
Research Scholar: Department of Electronics Engineering
G. H. Raisoni College of Engineering, Digdoh Hills,
Nagpur, India 440016
rahulhiware@gmail.com

Dinesh V. Padole
Assistant Professor: Department of Electronics Engineering
G. H. Raisoni College of Engineering,Digdoh Hills,
Nagpur, India 440016
dv_padole@rediffmail.com

***Abstract*— A multi-core processor is a single integrated circuit in which two or more processors have been attached for enhanced performance, reduced power consumption and more efficient simultaneous processing of multiple tasks.**
This paper presents dual-core embedded system designed with two LEON3 processors. Dual-core system has developed with centralized shared memory for process sharing, shared bus and tightly-coupled homogeneous scalable multi processor architecture.

Keywords*— *Multi-Core, Processor, LEON3, Leon Processor, Process Management in Multi-Core System, Embedded System, Controller

I. INTRODUCTION

To meet the growing needs of computing power, communication speed and performance requirements demanded by today's applications, processor clock speed has to be increased. However, increasing clock speed is not viable anymore due to heat dissipation and power consumption constraints. Hence Instead of trying to increase the clock speed, multi-core processor architectures with the lower frequency can be used.

A multi-core processor is a single integrated circuit in which two or more processors have been attached for enhanced performance, reduced power consumption and more efficient simultaneous processing of multiple tasks. A processing system is composed of two or more independent cores. An individual Processor is called as Core. The cores are integrated onto a single integrated circuit die or multiple dies in a single chip package. Multi-core system implements multiprocessing in a single physical package.

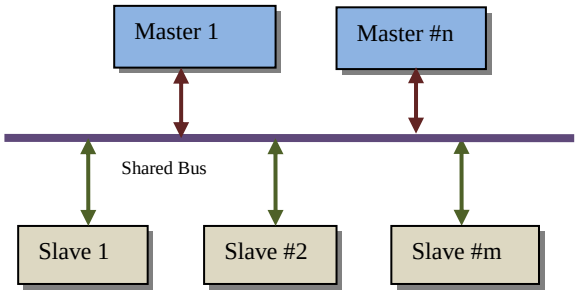


Fig.1 Multi-core System

Fig.1 shows a multi-core system uses *n* numbers of processors (core). There are *m* numbers of slaves which are nothing but various peripherals required in system. This system uses shared bus for data transfer and communication.

LEON is a 32-bit CPU microprocessor core, based on the SPARC-V8 RISC architecture and instruction set. It was originally designed by the European Space Research and Technology Centre, part of the European Space Agency, and after that by Gaisler Research. It is described in synthesizable VHDL. LEON has a dual license model: A LGPL/GPL FLOSS license that can be used without licensing fee, or a proprietary license that can be purchased for integration in a proprietary product. The core is configurable through VHDL generics, and is used in system-on-a-chip (SOC) designs both in research and commercial settings. The LEON3 is a synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. The model is highly configurable, and particularly suitable for SOC designs.

II. LEON3 ARCHITECTURE

The LEON3 core has the following main features:

- Implements 32-bit Scalable Processor Architecture(SPARC-V8)
- 7-stage pipeline with Harvard architecture i.e. Separate Instruction and Data caches
- AMBA-2.0 AHB bus interface
- New modules can easily be added using the on-chip AMBA AHB/APB buses.
- Hardware multiplier and divider
- Synthesizable VHDL model
- Highly configurable
- Suitable for system-on-a-chip (SOC) designs
- Availability of Full Source Code
- on-chip debug support and multiprocessor

LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance with low complexity and low power consumption. Fig.2 shows the internal functional blocks of LEON3. These blocks are explained as follows.

Integer unit

The LEON3 integer unit implements the full SPARC V8 standard, including hardware multiply and divides instructions. The number of register windows is configurable

within the limit of the SPARC standard (2 - 32), with a default setting of 8. The pipeline consists of 7 stages with a separate instruction and data cache interface (Harvard architecture).

Cache sub-system

LEON3 has a highly configurable cache system, consisting of a separate instruction and data cache. Both caches can be configured with 1-4 sets, 1–256 Kbyte/set, 16 or 32 bytes per line. Sub-blocking is implemented with one valid bit per 32-bit word. The instruction cache uses streaming during line-refill to minimize refill latency. The data cache uses write-through policy and implements a double-word write-buffer. The data cache can also perform bus-snooping on the AHB bus. A local scratch pad ram can be added to both the instruction and data cache controllers to allow 0-waitstates access memory without data write back.

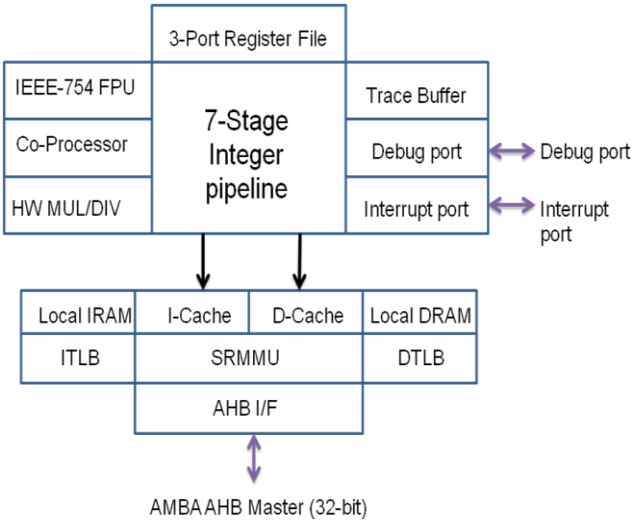


Fig.2 LEON3 processor core block diagram

Floating-point unit and co-processor

The LEON3 integer unit provides interfaces for a floating-point unit (FPU), and a custom co-processor. Two FPU controllers are available, one for the high-performance GRFPU (available from Gaisler Research) and one for the Meiko FPU core (available from Sun Microsystems). The floating-point processors and co-processor execute in parallel with the integer unit, and does not block the operation unless a data or resource dependency exists.

Memory management unit

A SPARC V8 Reference Memory Management Unit (SRMMU) can optionally be enabled. The SRMMU implements the full SPARC V8 MMU specification, and provides mapping between multiple 32-bit virtual address spaces and 36-bit physical memory. A three-level hardware table-walk is implemented, and the MMU can be configured to up to 64 fully associative TLB entries.

On-chip Debug support

The LEON3 pipeline includes functionality to allow non-intrusive debugging on target hardware. To aid software debugging, up to four watch-point registers can be enabled. Each register can cause a breakpoint trap on an arbitrary instruction or data address range. When the (optional) debug support unit is attached, the watch-points can be used to enter debug mode. Through a debug support interface, full access to all processor registers and caches is provided. The debug interfaces also allows single stepping, instruction tracing and hardware breakpoint/watch-point control. An internal trace buffer can monitor and store executed instructions, which can later be read out over the debug interface.

Interrupt interface

LEON3 supports the SPARC V8 interrupt model with a total of 15 asynchronous interrupts. The interrupt interface provides functionality to both generate and acknowledge interrupts.

AMBA interface

The cache system implements an AMBA AHB master to load and store data to/from the caches. The interface is compliant with the AMBA-2.0 standard. During line refill, incremental burst are generated to optimize the data transfer.

Power-down mode

LEON3 processor core implements a power-down mode, which halts the pipeline and caches until the next interrupt. This is an efficient way to minimize power-consumption when the application is idle, and does not require tool-specific support in form of clock gating. To implement clock-gating, a suitable clock-enable signal is produced by the processor.

Multi-processor support

LEON3 is designed to be use in multi-processor systems. Each processor has a unique index to allow processor enumeration. The write-through caches and snooping mechanism guarantees memory coherency in shared-memory systems.

III. DESIGN APPROACH

This section explains various steps and design strategies for designing multi core system using LEON3 processor core.

Multi core system is designed using shared resources like Shared memory, shared bus and shared peripherals.

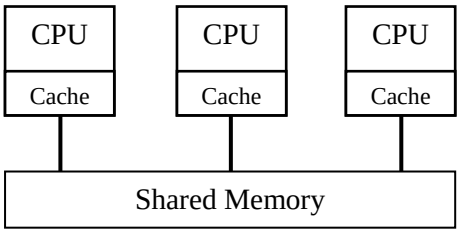
A. Shared memory

Two different possibilities of a tightly coupled (i.e. Shared memory system) multi core system exist. Shared memory model enable simple data sharing through a uniform mechanism of reading and writing shared structures in the common memory. Here all processors have symmetric access to the shared memory. In contrast, the distributed shared memory model implements a physically distributed-memory system. It consists of multiple independent processing nodes with local memory modules, connected by a general interconnection network like switches or meshes. Communication between processes residing on different nodes involves a message-passing model that requires extensive additional data exchange. The messages have to take care of data distribution across the system and manage the communication.

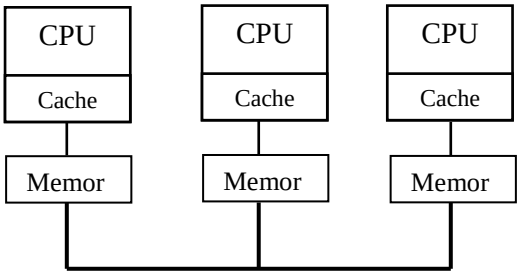
For a machine with shared address space, that address space can be used to communicate data implicitly via load and store operations. The advantage of shared memory

communication is Ease of programming. The program is loaded in the shared memory and the control flow can be maintained with simple programming constructs without too much of OS multiprocessor support.

In the Shared memory architecture, each memory request takes the same time independent of the CPU. Additionally, no message-passing communication system that



(a)



(b)

Fig.3 (a) Shared Memory model (b) Distributed Memory model

could limit the bandwidth of the interconnection is needed. Therefore, the shared memory architecture is the best choice for analyzing tight bounds of a memory access.

B. Arbiter based shared AMBA bus

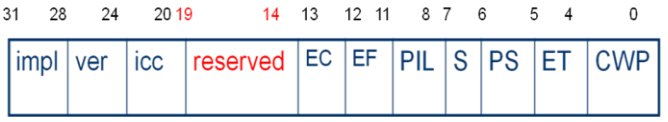
Proposed architecture is based on AMBA AHB/APB shared buses, to which new module of processor or peripherals can easily be added. It is a single resource which can carry only one transaction at a time. Hence to resolve the issue of bus sharing among multiple masters arbitration policies are to be used**.** Round robin and fixed arbitration policies are included in the design. Peripherals are also shared among various maters.

Following steps are taken for building Multi core embedded controller.

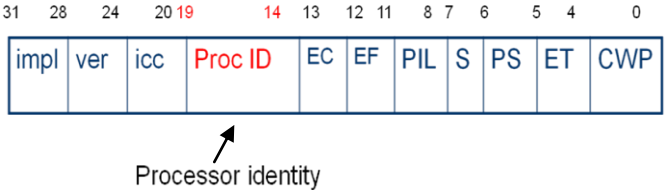
C. Assigning Unique Identity to Each Processor

For a multi core system, there has to be some means to uniquely identify each processor in order to have the control to execute a process on the desired processor and also to differentiate each processor. Therefore, a unique integer starting from '0' can be assigned to each processor and that is hardwired in the Processor Status Register (PSR) of each processor. The format for PSR register is given in fig.4.

In Multi core system one processor is designated as the BP (Boot Processor) because it is capable of controlling all system hardware, including AP (Application Processor) startup and shutdown. The BP is not necessarily the first processor, especially in fault tolerant multiprocessor systems in which any available processor can be designated as the BP.



(a)



(b)

Fig.4 (a) Initial PSR (b) Modified PSR

D. Embedded Controller with multiple masters

The AHB bus can connect up to 16 masters. The LEON processor '0' is connected as master '0' with processor id '0' and processor '1' as master '1' with processor id '1'. Each processor has its own instruction and data cache controllers inter-faced through the AHB interface to communicate to the AHB bus. Each processor has its corresponding Cache Controller register sitting on the APB bus as slaves at different addresses. The Cache Control registers of both the processors and the other registers like PSR, Floating unit and Coprocessor unit registers are initialized in the boot code. The AHB controller controls the bus and implements the bus arbitrator. The Boot Processor (processor 0) is by default put on to the lowest index.

E. Default master setting

Every system must include a default bus master which is granted the bus if all other masters are unable to use the bus. When granted, the default bus master must only perform *IDLE* transfers. If no masters are requesting the bus, then the arbiter may either grant the default master or alternatively it may grant the master that would benefit the most from having low access latency to the bus. Granting the default master access to the bus also provides a useful mechanism for ensuring that no new transfers are started on the bus and is a useful step to perform prior to entering a low-power mode of operation. The default master must be granted if all other masters are waiting for *SPLIT* transfers to complete.

F. Adding Boot Register as slave to AHB Bus

The first problem with a multiprocessor kernel is waking up other processors. Other processors should not start

or captured elsewhere. So to achieve this, a Boot Register is added as an AHB slave at address '0xf0000000'.

G. Cache Coherency

The data cache uses write-through policy and implements a double-word write-buffer. The data cache can also perform bus-snooping on the AHB bus. The 23rd bit of Cache Control Register (CCR) when asserted enables the AHB-bus snooping. Changes are made in the cacheable areas which are checked before Cache Snooping. The address on the bus is checked if it belongs to the PROM area or RAMS area. The address of AHB Boot-Register is also added to the list to make the address to be snooped and the changes made at the AHB Boot Register be notified to all the processors that have the snooping enabled. During snooping, if the data transfer is noticed by the master who issues it, then that data transfer is not considered by that master to avoid redundancy.

III. Design of Embedded System using LEON3 CORE

A. Single Core System

Embedded system consists of processor with memory and Peripherals. The processor could be single (in single core) or multiple (in Multi core). Peripheral on chip can be Timer, serial communication interface, Interrupt controller and programmable IO devices. The system design presented here have used shared bus (AMBA Bus is utilized for core interface) and shared resources (peripherals and Memory). Fig.5 shows block diagram for embedded controller design using single Leon core.

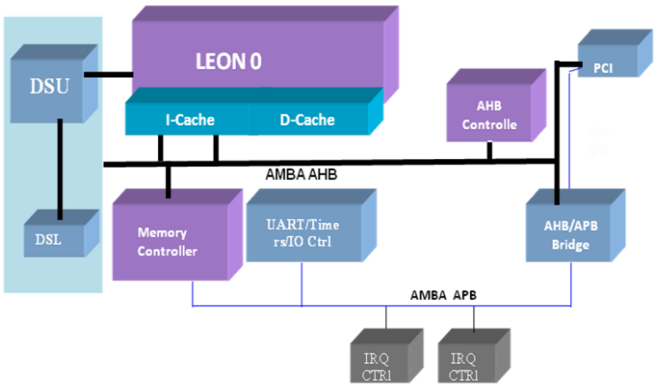


Fig.5 Block diagram of Single core LEON3 System

Fig.6 shows structure for embedded controller design using single Leon core.

Embedded controller using LEON processor is synthesized with the help of Xilinx 9.1 ISE synthesis software tool. Timer & Interrupt controller is added as peripheral as well as shared memory is build up on chip. Leon processor will be working as master, whereas Peripherals (Timer, Interrupt controller) & memory is configured as slave. AMBA is used as shared bus for all the cores.

B. Dual Core System

Fig.7 shows block diagram of embedded controller using two LEON3 processors. In the diagram L-0 and L-1 An

indicate LEON3 processor core. Each processor has their individual cache memory. Both cores (processor L0, L1) are connected using AMBA shared bus. Leon '0' is configured as boot processor.

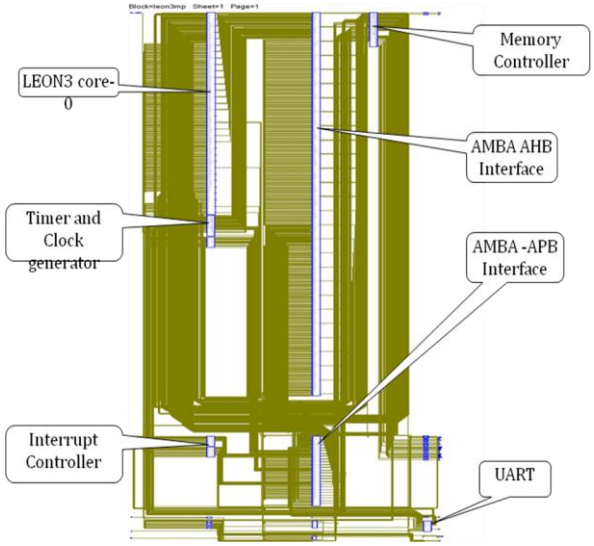


Fig.6 RTL Structure of Single core LEON3 System

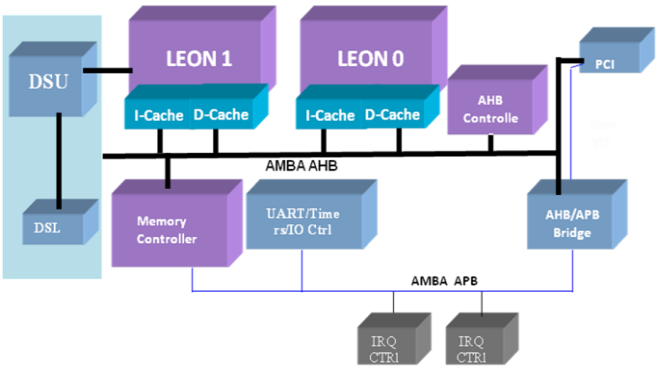


Fig.7 Block diagram of Dual core LEON3 System

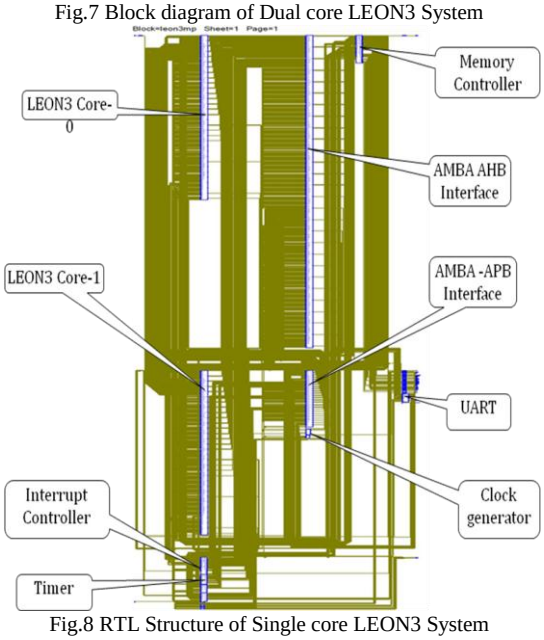


Fig.8 RTL Structure of Single core LEON3 System

C. Resource utilization by systems

TABLE 1 shows table of comparisons of total resources used by Single, Dual and Quad core systems. The device used for implementation is Xilinx FPGA Vertex5

Vol:1 Issue:1 ISSN 2278 - 215X

(XC5vlx30-3ft324). Numbers in circular bracket shows total availability of particular component of FPGA device used

TABLE 1 RESOURCE UTILIZATION BY SINGLE AND DUAL CORE EMBEDDED SYSTEMS

	Single core	**Dual core**
Xilinx FPGA, Vertex5 device used	XC5vlx30-3ft324	XC5vlx30-3ft324
Number of Slice Registers (19200)	3596	6213
Number of Slice LUTs (19200)	7070	11687
Number of fully used Bit Slices (14269)	2231	3631
Number of bonded IOBs (222)	113	113
Total Utilization	**36%**	**60%**

IV.CONCLUSION

The performance of multi core system can be calculated in terms of power consumption.
The comparison of percentage power consumption by Multi core controller can be calculated as –

% power Utilization = N * (Power by single core controller) *100 / Power by Multi core controller
Where, N is number of core in Multi core SoC.

The homogeneous dual core system using LEON soft core, shared memory and shared AMBA bus has been designed and results are verified. The design is made scalable where user has to specify the number of processor and arbitration policy. The performance comparison of single and quad core system is obtained in terms of resource utilization and power consumption. It is concluded that power consumption reduces by increasing number of on chip processor.

REFERENCES

[1] Jiri Gaisler. *The LEON-3 Processor User's Manual,* Version 1.0.20, February 2009 http://www.gaisler.com,

[2] *GRLIB IP Core User's Manual,* Version 1.0.22,April http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=156&Itemid=104/grip.pdf

[3] IBM, QuadCore, http://www-03.ibm.com/systems/x/quadcore.html

[4] Dual-Core Intel® Xeon® Processor, "Increasing Data Center Density While Driving Down Power and Cooling Costs" www.intel.com/go/xeon

[5] *Snehal Dongare, Dinesh Padole, Dr. Preeti Bajaj,* "Design of Shared Resource Based Multicore Embedded Controller Using LEON Processor", ICETET-10, Goa, India.

[6] *Dinesh Padole, Dr. Preeti Bajaj, "*Fuzzy Arbiter Based Multi Core System-On-Chip Integrated Controller For Automotive Systems: A Design Approach", IEEE CCECE08, Canada

[7] Xilinx Inc. *The Virtex VDCM — Digital Clock Manager*. www.xilinx.com/products/virtex/techtopic/vtt010.pdf

[8] RAMP – Research Accelerator for Multiple Processors, http://ramp.eecs.berkeley.edu/

[9] *Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta*, "The SPLASH-2 Programs: Characterization and Methodological Considerations", 22nd International Symposium on Computer Architecture, pages 24-36, Santa Margherita Ligure, Italy, June 1995 (http://www-flash.stanford.edu/apps/SPLASH/)

[10] *Timothy Wong* "LEON3 System-on-Chip Port for BEE2 and ASIC Implementation",