FPGA BASED DESIGN OF ON-CHIP PERIPHERAL (OPB) BUS BRIDGES FOR MULTIMEDIA APPLICATIONS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

PARVEEN KUNDU

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

JUNE, 2005

ID No. MT-300/2005-31/[illegible]

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation entitled **"FPGA BASED DESIGN OF ON-CHIP PERIPHERAL (OPB) BUS BRIDGES FOR MULTIMEDIA APPLICATIONS"** submitted in partial fulfillment of the requirements for the award of degree of **Master of Technology in Electrical Engineering** with specialization in **System Engineering and Operations Research,** in the Department of Electrical Engineering, Indian Institute of Technology Roorkee, Roorkee, is an authentic record of my own work carried out from July 2004 to June 2005 under the guidance of **Prof.M.K.Vasantha**, Professor and **Dr.Indra Gupta,** Asstt Professor, Department of Electrical Engineering, Indian Institute of Technology Roorkee, Roorkee.

The matter embodied in this report has not been submitted by me for the award of any other degree or diploma.

Date: 29 JUNE 2005
Place: Roorkee

Parveen Kundu
(PARVEEN KUNDU)

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

[illegible] 29/6/2005
Prof. M.K.Vasantha
Professor
Department Of Electrical Engineering
Indian Institute of Technology Roorkee.
Roorkee.

Indra 29/6/05
Dr.Indra Gupta
Asst Professor
Department of Electrical Engineering
Indian Institute of Technology Roorkee
Roorkee.

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere gratitude to **Prof.M.K.Vasantha**, Professor, and **Dr.Indra Gupta,** Asstt Professor, Department of Electrical Engineering, Indian Institute of Technology Roorkee, Roorkee for encouraging me to undertake this dissertation as well as providing me their valuable guidance and inspirational support without which this work would not have been in present shape.

I consider myself extremely fortunate for having got the opportunity to learn and work under their able supervision. They provided me the latest technology to work on and state of the art facilities and equipments at the lab. Their able guidance ensured me that I should not have any lack of resources while carrying out my thesis work. I have deep sense of admiration for their innate goodness and inexhaustible enthusiasm. The valuable hours of discussions and suggestions that I had with them have undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective. Working under their guidance will always remain a cherished experience in my memory and I will adore it throughout my life.

I am also thankful to **Dr. H.O Gupta,** Professor and Head of Department of Electrical Engineering, IIT Roorkee for helping me in my dissertation work. I am especially thankful to him for providing us the technical training in M/S Mechtaronics, Pune which was pivotal in completion of my work.

My heartfelt gratitude and indebtedness goes to all the teachers of SEOR group who, with their encouraging and caring words, constructive criticism and suggestions, have contributed directly or indirectly in a significant way towards completion of this report.

I am highly grateful to **Mr.Rahul Dubey,** Research scholar, Department of Electrical Engineering, Indian Institute of Technology Roorkee, and Roorkee for his keen interest and generous encouragement during this venture.

I am especially thankful to **Major Seby Thomas** for his feedback in the class. His ideas and discussions not only helped in technical areas but also in personality development.

I am also thankful to **Mr.Vishal Saxena** and **Mr.Vijender Singh** Research Scholar for their valuable guidance in the hour of need.

I am thankful to **Mr. Kalyan Singh** and **Mr. C.M Joshi**, Laboratory staff of Micro Processor & Computer Lab for providing the required facilities and co-operation during this work.

Special, sincere and heartfelt gratitude goes to my parents and my friends whose sincere prayers, best wishes, support and encouragement have been a constant source of assurance, guidance, strength and inspiration to me.

Parveen Kundu

PARVEEN KUNDU

ABSTRACT

With the recent advancements in silicon densities it is now possible to integrate numerous functions onto a single silicon chip. With this increased density, the peripherals that were formerly attached to the processor at the card level are now integrated onto the same chip as the processor. Even it is possible now that the chip may contain several processors. Now a days this technology is popularly known as System-on-Chip (SoC).

As large numbers of peripherals are present on the single chip, on–chip buses are required to connect peripherals and the processor. With this aim in mind on-chip peripheral bus (OPB) and Processor Local Bus (PLB) were designed to integrate the different components as a complete system.

On-chip Peripheral Bus (OPB) and Processor Local Bus (PLB) have different functions to perform.OPB is used for connecting slower peripherals and PLB for connecting faster peripherals that operate almost at the same speed as that of the processor .For carrying out the transaction between slower and faster components attached to the different buses bridges are designed called PLB to OPB bridge and OPB to PLB bridge.

Xilinx has designed a tool called Embedded Development Kit (EDK) 6.3 that eases the designing of the complete embedded system .This dissertation uses this tool to design application. OPB, PLB and bus bridges have been used to carry out the connectivity of the peripherals and the processor. Virtex-II Pro FPGA, one of the most sophisticated FPGA available till date is used to realize the design in real time.

Virtex-II Pro-PCI Video Card has been used in this dissertation for implementing the Multimedia Application. Power PC that is embedded into the Virtex-II Pro FPGA has been used as processing unit of the embedded design.PLB to OPB Bridge has been used to connect the multimedia application to the Power PC.These bridges acts as a link between devices connected to different buses.

Xilinx design tool ISE 6.3 has been used to actually Place and Route the design on the Virtex-II Pro FPGA.

CONTENTS

	Page No.
CANDIDATES'S DECLARATION	I
ACKNOWLEDGEMENTS	II
ABSTRACT	IV
LIST OF FIGURES	VII
ABBREVIATION AND ACRONYMS	VIII
INTRODUCTION	1
CHAPTER 1: TECHNOLOGY AND BACKGROUND	3
1.1. System on Chip (SoC)	3
1.2. Application Specific Integrated Circuits (ASIC's)	3
1.3. Programmable Logic Devices (PLD's)	4
1.4. Field Programmable Gate Arrays (FPGA's)	4
1.5. Hardware Description Languages (HDL's)	4
1.6. Active HDL.	4
1.7. Processors Cores	4
1.7.1. Soft Processor.	5
1.7.2. Hard Processor	5
1.8. IBM Core connect Bus Architecture	5
1.9. Embedded Development Kit	6
1.10. ISE.	7
1.11. Bus Bridges	7
CHAPTER 2: EMBEDDED SYSTEM TOOLS	8
2.1. Embedded development Kit (EDK)	8
2.1.1. Xilinx Platform Studio	9
2.1.2. Base System Builder	9
2.1.3. Create/Import IP Wizard	10
2.1.4. Platform Generator	10
2.1.5. Simulation Model Generator	10
2.1.6. Library Generator	10
2.1.7. Bitstream Initializer	10
2.1.8. GNU Compiler Tools	11
2.2. Tools Flow In the creation of Embedded System	11
2.2.1. Hardware Platform Creation	11
2.2.2. Verification Platform Creation	12
2.2.3. Software Platform Creation	12
2.2.4. Software Application Creation and Verification	13

CHAPTER 3: BUS BRIDGES IN EMBEDDED SYSTEMS 15
3.1. Introduction 15
3.2. Bridges in Embedded system design. 15
3.3. PLB to OPB Bridge 17
3.3.1. Interfaces in PLB to OPB Bridge. 19
3.4.OPB to PLB Bridge. 20
3.4.1. Interfaces in OPB to PLB Bridge. 22

CHAPTER 4: DESIGN OF EMBEDDED SYSTEM USING BUS BRIDGES 23
4.1. Introduction 23
4.2. Microblaze 23
4.3. Power PC 24
4.4. Design of embedded system using MicroBlaze 27
4.5. Design of embedded system using Power PC 28
4.6. On-chip Peripheral (OPB) Bus. 29
4.7. General Purpose Input Output 29
4.8. Virtex-II Pro FPGA. 30
4.9. Digital to Analog Converter (DAC) THS8133. 32
4.10. Multimedia Application using Virtex-II Pro and bridges 37

CONCLUSIONS 40
FUTURE SCOPE 41
REFERENCES 42

APPENDIX
A. Virtex-II Pro –PCI Video Board
B. Program Code (VHDL)
C. Program Code (C)
D. RESULTS

LIST OF FIGURES

Figure No.	Title	Page no.
1.1	IBM Core Connect Structure	6
2.1	Embedded Software Tool Architecture	9
2.2	Hardware Platform Creation	11
2.3	Verification Platform	12
2.4	Software Platform	13
2.5	Software Application Creation and Verification	14
3.1	On-chip bus structure	16
3.2	Block Diagram of PLB to OPB Bridge	18
3.3	Block Diagram of OPB to PLB Bridge	21
4.1	MicroBlaze Core Block Diagram	23
4.2	Power PC organization	26
4.3	Embedded System Design using MicroBlaze	27
4.4	Embedded System Design using Power PC	28
4.5	Block Diagram of DAC THS 8133	34
4.6	THS8133 DAC Pin Diagram	36
4.7	Multimedia Application using bus bridge	37
4.8	IPIF Module connecting IP core	39

ABBREVIATIONS

ADC	Analog to Digital
ASIC	Application Specific Integrated Circuits
BRAM	Block Random Access Memory
BSB	Base System Builder
CLB	Configurable Logic Blocks
DAC	Digital to Analog
DCM	Digital Clock Manager
DCR	Device Control Register Bus
DLMB	Data Local Memory Bus
DMA	Direct Memory Access
DOPB	Data On-Chip Peripheral Bus
EDIF	Electronic Design Interchange Format
EDK	Embedded Development Kit
EMAC	Ethernet Media Access Control
FIFO	First In First Out
FPGA	Field Programmable Gate Arrays
FSL	Fast Simplex Link
GPIO	General Purpose Input Output
GUI	General User Interface
HDL	Hardware Description Language
HW	Hardware
IBM	International Business Machines
IOB	Input Output Blocks
IP	Intellectual Property
IPIF	Intellectual Property Interface
JTAG	Joint Test Action Group
LUT	Look Up Tables
MB	MicroBlaze

INTRODUCTION

In near future, the microelectronics industry will face the reality of a billion transistors on a chip. Now it is not economical to use the chip for a single function, as large resources will remain unutilized. Therefore these days there is a growing trend towards designing the complete system on a single chip. This is popularly known as System-on-Chip (SoC) Technology. This technology allows the designer to implement a complete system on a single chip dedicated to particular application.

In an effort to implement the system from scratch the costs incurred are enormous and only a few companies can build the needed competency in all the design areas. In most cases, designers have to use intellectual property (IP) blocks, possibly originating from external vendors. IP blocks are predefined, large grained blocks, such as filters, peripherals, memories, and processors, whose function has been precisely specified. Designing a system based on reusable IP blocks challenges industry to develop new design methodologies and tools.

Xilinx has developed a tool called Embedded Development Kit (EDK) [6] to provide a solution to the problem of placing a complete system on the single chip using IP blocks from different sources. It provides the platform to integrate the IP's from different vendors with the resources available within the tool to design the system according to particular application. This approach is known as a plug and play environment.

Such an integration environment as provided by the Xilinx is typically a design platform for a specific application domain. The IP blocks are standard building blocks that can be easily integrated within the application domain. The integration platform consists of components such as a target hardware and software architecture, a portfolio of virtual components (IP blocks), and a design validation methodology.

A simple general-purpose processor core is the basic component in the integration platform. Xilinx have included two processor cores into the design, Microblaze [8] which is a soft processor and Power PC [11] core which is hard processor. These processor cores are integrated into the fabric of the chip. All the IP blocks are glued together through buses that communicate with the processor.

Two kinds of buses are introduced for the purpose of connecting different peripherals on the chip: the processor local bus (PLB) and the on-chip peripheral bus (OPB) [10]. The OPB is connected to the PLB through a module interface called Bus Bridge. There are two bus bridges available PLB to OPB Bridge[2] and OPB to PLB Bridge[4]. The PLB arbiter controls the PLB communications among the processor, memory and OPB bridge. The OPB arbiter controls the OPB communications among the IP blocks.

The IP blocks can be user-defined logic blocks or third party IP blocks. Other company which is also working in this area is ARM and they have designed there own on-chip bus known as AMBA.

FPGA's are the programmable logic devices that are used for real time testing for the system designed for the chips.Virtex-II Pro FPGA's [14] are the latest in the family of Xilinx devices which has almost all the features for designing the complex embedded system .

TECHNOLOGY AND BACKGROUND

1.1. System-on chip (SoC)

System-on chip is referred to devices where whole of the computing system is integrated on the one chip. Main features of SoC are:

1. Reduced chip interconnect.
2. Reduced power dissipation.
3. Reduced device size.

A typical SoC may contain one or many processor, on-chip memory, different types of bus structures, bridges to connect the different buses, an arbitrary number of peripherals connected to each other to form a complete circuit. The system on chip aims at providing only one circuit for any particular application. A SoC contains a large number of I/O interfaces to connect to the other circuits.

A SoC are designed with limited applications in mind and need less processing power then general purpose computer .An SoC processor works on only few MHz range while modern PC run on 500 MHz –GHz range. Frequency required to operate the SoC is just sufficient to run the design. Due to low frequency the power consumption and chip temperature is reduced and the system operates with less cooling devices and better battery utilization.

1.2. Application Specific Integrated Circuits (ASIC's)

Application Specific Integrated Circuits (ASIC) is one of the common types of chip types ASIC can implement designs ,ranging from very simple to complex which are dedicated to specific application.ASIC are therefore customized for reduced power dissipation, less area and greater clock frequencies.ASIC have low mass production costs.

Main disadvantages of ASIC's are long design phases, low configurability and high start up costs. ASIC's are mainly used for large scale manufacturing of verified designs but not for prototypes.

1.3. Programmable Logic Device (PLD's)

This are also used to design systems whose complexity varies from simple combinatorial design to complex SoC's.

1.4. Field Programmable Gate Arrays (FPGA's)

Field Programmable Gate Arrays (FPGA) are a type of Programmable Logic Devices.FPGA is a general architecture consisting of configurable logic blocks and programmable interconnections. Several FPGA has enough logic to implement complex SoC's. FPGA's are not optimized for a particular design and therefore consume more power or implement a design less efficient then the ASIC's. Price per chip is high.

Advantages of FPGA's is that they are easy to reprogram, which shortens design cycles and allow early real world tests. This makes FPGA's well suited for prototypes and small production volumes.

1.5. Hardware Description Languages (HDL's)

Hardware Description Language (HDL) provides ways to design hardware .VHDL and Verilog are two popular hardware description Languages

1.6. Active HDL

This is the simulation software used to simulate VHDL design.Version 6.1 of this software has been used.

1.7. Processor Cores.

Processor cores refer to a processor excluding any peripherals that are connected to it .In SoC one or more processors are connected to peripherals in a single chip.

There are three different types of processor cores:

1. Soft Processor core: Cores delivered as technology dependent gate-level netlist or HDL source code.
2. Firm processor cores: Cores delivered as library elements.

3. Hard processor cores: Cores which has a fixed physical layout and which are incorporated into the design as standard cell.

1.7.1. Soft Processor:

Soft Processor Core: These are the cores delivered as technology dependent gate-level *netlist* or HDL source code for synthesis. Soft processors have recently gained wide popularity .This popularity appears to more amongst the FPGA developers.

This popularity is mainly due to several factors as:

1. Increase in performance.
2. Increased in performance/price ratio on FPGA's.
3. Increased availability of both commercial and academic cores.
4. Free soft processors have been released by teams consisting of professionals, academics and enthusiasts (FPGACPU, OPENCORES).

Microblaze [8] is such a soft processor core provided by the Xilinx.

1.7.2 Hard Processor:

Power PC[11] is the hard processor .Xilinx have embedded the these hard processor into the FPGA Fabric from Virtex-II Pro FPGA families onwards.

1.8. IBM Coreconnect Bus Architecture:

IBM Coreconnect Bus[1] Architecture is a small set of buses intended for SoC designs. The block diagram IBM Coreconnect structure is shown above in the Figure 1-1:

IBM Coreconnect features three buses:

1. Processor Local Bus (PLB): High performance bus for connecting fast processor cores and high performance peripherals (PCI interface, memory controllers) etc.
2. On-Chip Peripheral Bus (OPB): This is a simple bus for connecting slow peripherals like UART are connected to this bus.
3. Device Control Register bus (DCR): Control Bus that links to all of the devices, controllers and bridges.

By using these buses the overall SoC can be optimized for performance while simple peripherals may be optimized for simplicity.

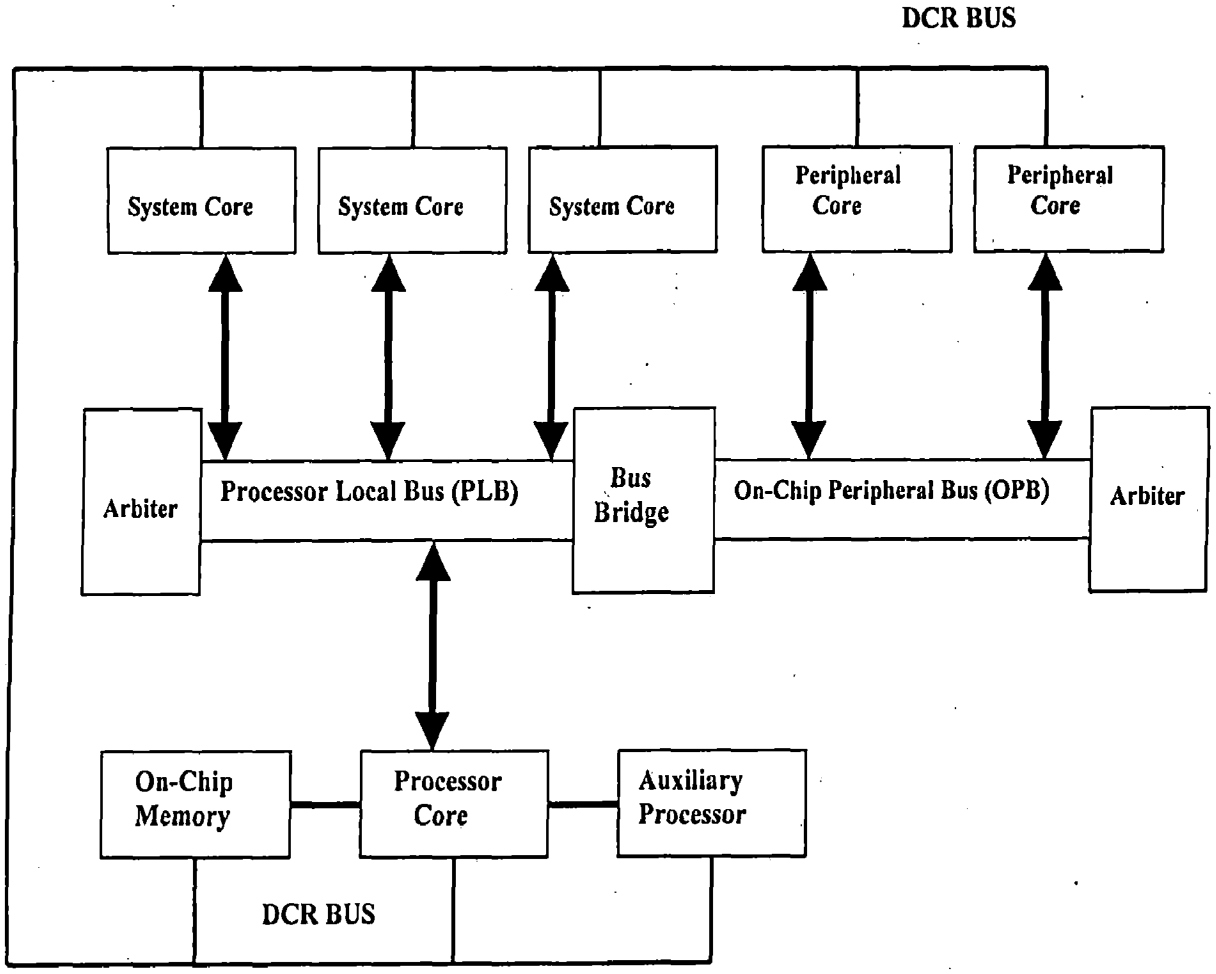


Figure1.1: IBM Core Connect Structure

Xilinx has developed the IBM Core connect OPB with most configurable OPB parameters set to specific values. Xilinx OPB is based on OR-Gates and does not use enable inputs which are used in IBM OPB which are used to assert when the bus is not used.

Xilinx OPB devices share the same parameters because of which they are compatible. Xilinx devices include Xilinx Microblaze Soft processor core, Xilinx Picoblaze Soft processor core and a large set of OPB peripherals.

1.9. Embedded Development Kit:

EDK[6] is a series of software tools for designing embedded programmable systems, and supports designs of processor sub-systems using the IBM PowerPC hard processor core and the Xilinx Microblaze soft processor core. It has graphical user interface Platform Studio that integrates all the processes from design entry to design debug and verification. This is used to design both simple and complex designs .EDK 6.3 has been used in this dissertation work though EDK7.1 has been launched in the market.

1.10. ISE:

This software is used to place and route the design into the FPGA.

1.11. Bus Bridges:

There are two bridges core recently introduced:

1. PLB to OPB Bridge.
2. OPB to PLB Bridge.

These two bridges PLB to OPB Bridge[2] and OPB to PLB[4] Bridge allow the PPC 405 to be connected to OPB devices. These bridges are used in the recently built Virtex -2 Pro boards.

CHAPTER-2
EMBEDDED SYSTEM TOOLS

2.1. Embedded Development Kit (EDK)

Embedded Development Kit (EDK)[6] 6.3 contains the rich set design tools and a wide selection of standard peripherals required to build embedded processor systems using MicroBlaze, the industry's fastest soft processor solution, and the new and unique feature in Virtex-II Pro, the IBM PowerPC CPU.

These tools, contains processor platform for particular application, software application development tool, a full featured debug tool chain and device drivers and libraries, which allows the developer to develop a System-On-Chip(SoC) design with the help of MicroBlaze[8] and Virtex-II Pro[14] based Power PC[11].

Figure 2.1 below shows embedded software tool architecture. Multiple tools based on a common framework allow the user to design the complete embedded system. System design consists of the creation of the hardware and software components of the embedded processor system, and optionally, a verification or simulation component as well. The hardware component consists of an automatically generated hardware platform that can be optionally extended to include other hardware functionality specified by the user.

The software component of the design consists of the software platform generated by the tools, along with the user designed application software. The verification component consists of automatically generated simulation models targeted to a specific simulator, based on the hardware and software components.

A typical embedded system design project involves the following phases:

1. Hardware platform creation,
2. Hardware platform verification (simulation),
3. Software platform creation,
4. Software application creation, and
5. Software verification (debugging).

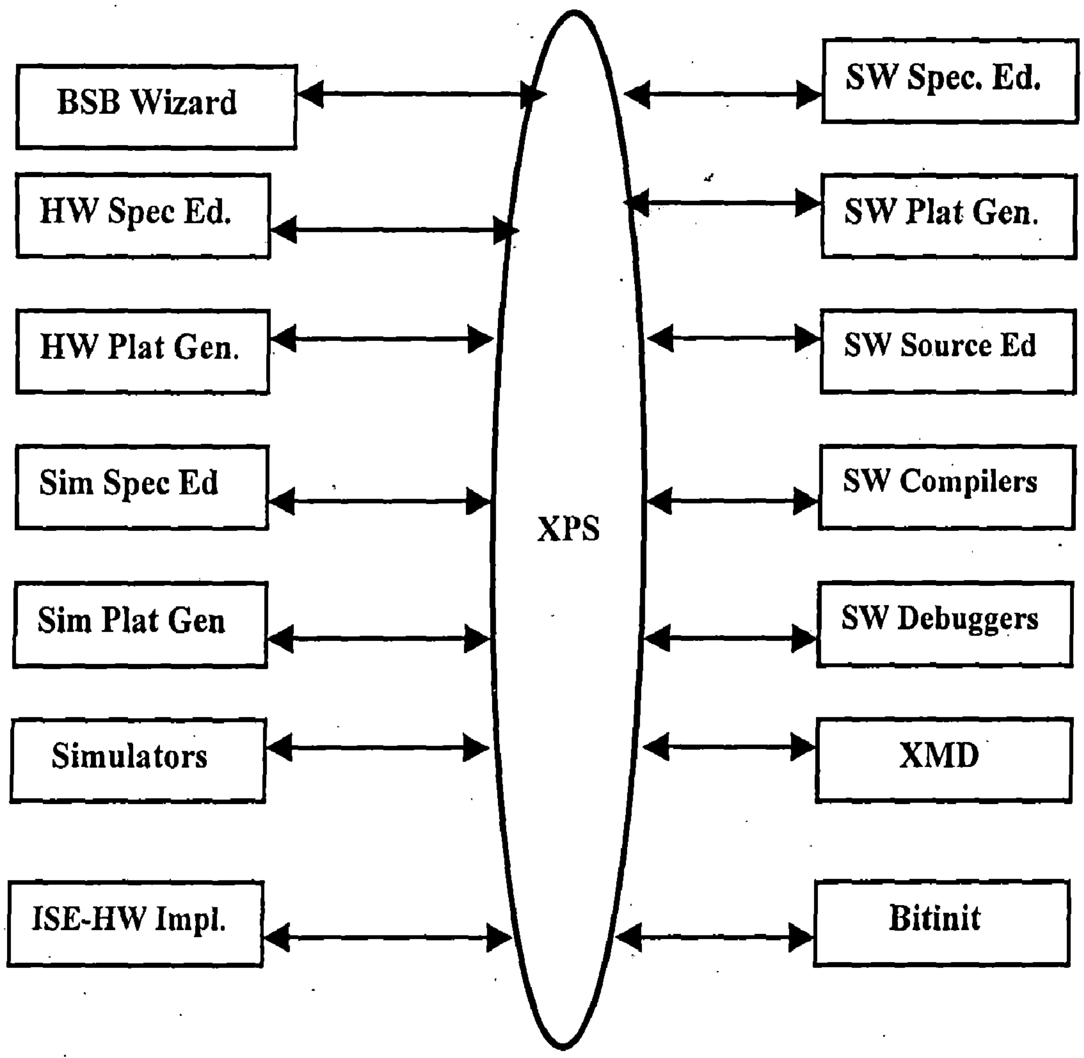


Figure 2.1 Embedded Software Tool Architecture

Embedded Development Kit contains the following tools in the design:

2.1.1. Xilinx Platform Studio

The Xilinx Platform Studio (XPS)[9] tool provides a GUI (General User Interface) for creating the MHS and MSS files for the hardware and software flow. XPS also provides source file editor capability and project and process management capability. XPS is used for managing the complete tool flow, that is, both hardware and software implementation flows.

2.1.2. Base System Builder

The Base System Builder (BSB) wizard is a software tool that helps users quickly build a working system targeted at a specific development board. BSB is invoked by XPS when the user wants to create a new system.

2.1.3. Create/Import IP Wizard

The Create/Import Peripheral[13] Wizard helps to create own peripherals and import them into EDK compliant repositories or Xilinx Platform Studio (XPS) projects. This wizard uses the Psf Utility tool to create the necessary Platform Specification files.

2.1.4. Platform Generator

The embedded processor system in the form of hardware net lists (HDL and EDIF files) is customized and generated by the Platform Generator (PlatGen).

2.1.5 Simulation Model Generator

The Simulation Platform Generation tool (simgen) generates and configures various simulation models for the hardware. It takes a Microprocessor Hardware Specification (MHS) file as input.

2.1.6. Library Generator

XPS calls the Library Generator tool for configuring the software flow. The Library Generator (LibGen) tool configures libraries, device drivers, file systems and interrupt handlers for the embedded processor system. The input to LibGen is an MSS file.

2.1.7. Bitstream Initializer

The Bitstream Initializer tool initializes the instruction memory of processors on the FPGA. The instruction memories of processors are stored in BlockRAMs in the FPGA. This utility reads an MHS file, and invokes the Data2MEM utility provided in ISE to initialize the FPGA BlockRAMs.

2.1.8. Platform Generator

The embedded processor system in the form of hardware net lists (HDL and EDIF files) is customized and generated by the Platform Generator (PlatGen).

2.1.9. Simulation Model Generator

The Simulation Platform Generation tool (simgen) generates and configures various simulation models for the hardware. It takes a Microprocessor Hardware Specification (MHS) file as input.

2.1.10. Library Generator

XPS calls the Library Generator tool for configuring the software flow. The Library Generator (LibGen) tool configures libraries, device drivers, file systems and interrupt handlers for the embedded processor system. The input to LibGen is an MSS file.

2.1.11. GNU Compiler Tools

XPS calls GNU compiler tools for compiling and linking application executables for each processor in the system.

2.2. Tools Flow In the creation of Embedded System:

2.2.1. Hardware Platform Creation

Xilinx Platform Studio[9] provides the Base System Builder Wizard for creating the Hardware Platform.Details of hardware platform creation are depicted in Figure 2.2

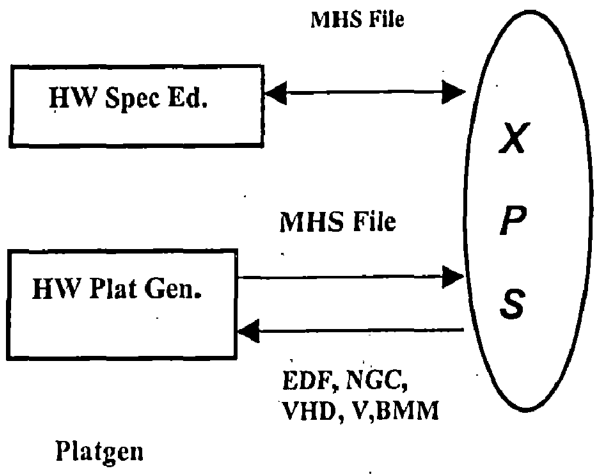


Figure 2.2 Hardware Platform Creation

Microprocessor Hardware Specification (MHS):

The hardware platform is defined by the MHS (Microprocessor Hardware Specification) file. The hardware platform consists of one or more processors and peripherals connected to the processor buses. Several peripherals are provided with the software. Peripherals can be designed according to any particular applications and can be included into the MHS. The MHS file is a simple text file and any text editor can be used to create this file. The XPS tool provides graphical means to create the MHS file.

The MHS file defines the system architecture, peripherals and embedded processors. The MHS file also defines the connectivity of the system, the address map of each peripheral in the system and configurable options for each peripheral. Multiple processor instances connected to one or more peripherals through one or more buses and bridges can also be specified in the MHS.

Platform Generator tool (PlatGen):

The Platform Generator tool (PlatGen) creates the hardware platform using the MHS file as input. PlatGen creates netlist files in various formats (NGC, EDIF), as well as support files for downstream tools, and top-level HDL wrappers to allow designers to add other components to the automatically generated hardware platform.

2.2.2. Verification Platform Creation

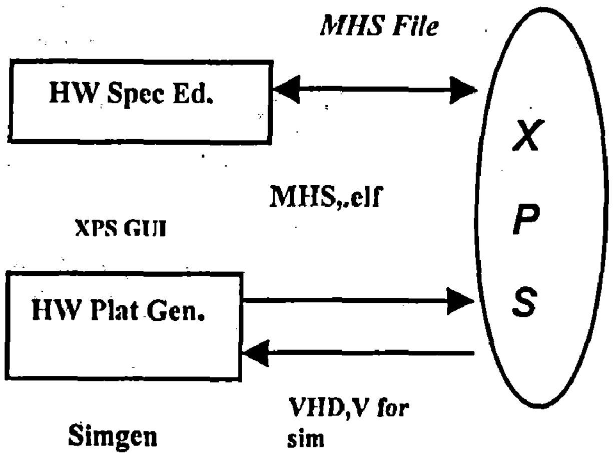


Figure 2.3 Verification Platform

The verification platform is based on the hardware platform. The MHS file is processed by the Simgen tool to create simulation files (VHDL, Verilog or various compiled models) along with some command files for specific simulators supported by the tool. As in the case of the hardware platform, edit these simulation files can be edited to add other components to the automatically generated verification platform. If the software application that runs on the hardware platform is available in executable format, it can be used to initialize memories in the verification platform. The processor of verification is shown in Figure 2.3.

2.2.3. Software Platform Creation:

Microprocessor Software Specification (MSS) File:

The software platform is defined by the MSS (Microprocessor Software Specification) file. The MSS file defines driver and library customization parameters for peripherals, processor customization parameters, standard input/output devices, interrupt handler routines, and other related software features. The MSS file is a simple text file and any text editor can be used to create this file. The XPS tool provides a graphical user interface for creating the MSS file.

The MSS file is an input to the Library Generator tool (LibGen) for customization of drivers, libraries and interrupt handlers. The entire process of creating the software platform is shown in Figure 2-4.

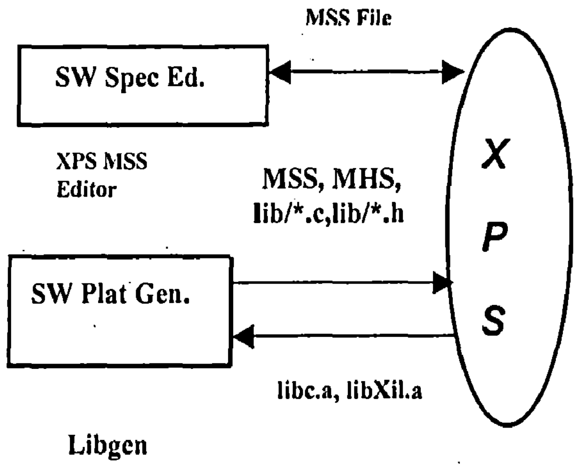


Figure 2-4: Software Platform

2.2.4. Software Application Creation and Verification:

Software Application:

The software application is the code that runs on the hardware and software platforms. The source code for the application is written in a high level language such as C or C++, or in assembly language. XPS [9] provides a source editor for creating these files, but any other text editor may be used here. Once the source files are created, they are compiled and linked to generate executable files in the ELF (Executable and Link Format) format. GNU compiler tools for PowerPC and MicroBlaze are used by default.

Verification:

XMD (Xilinx Microprocessor Debugger) and the GNU debugger (GDB) are used together to debug the software application. XMD provides an instruction set simulator, and optionally connects to a working hardware platform to allow GDB to run the user application. This process of Software Application Creation and Verification is depicted in Figure 2-5.

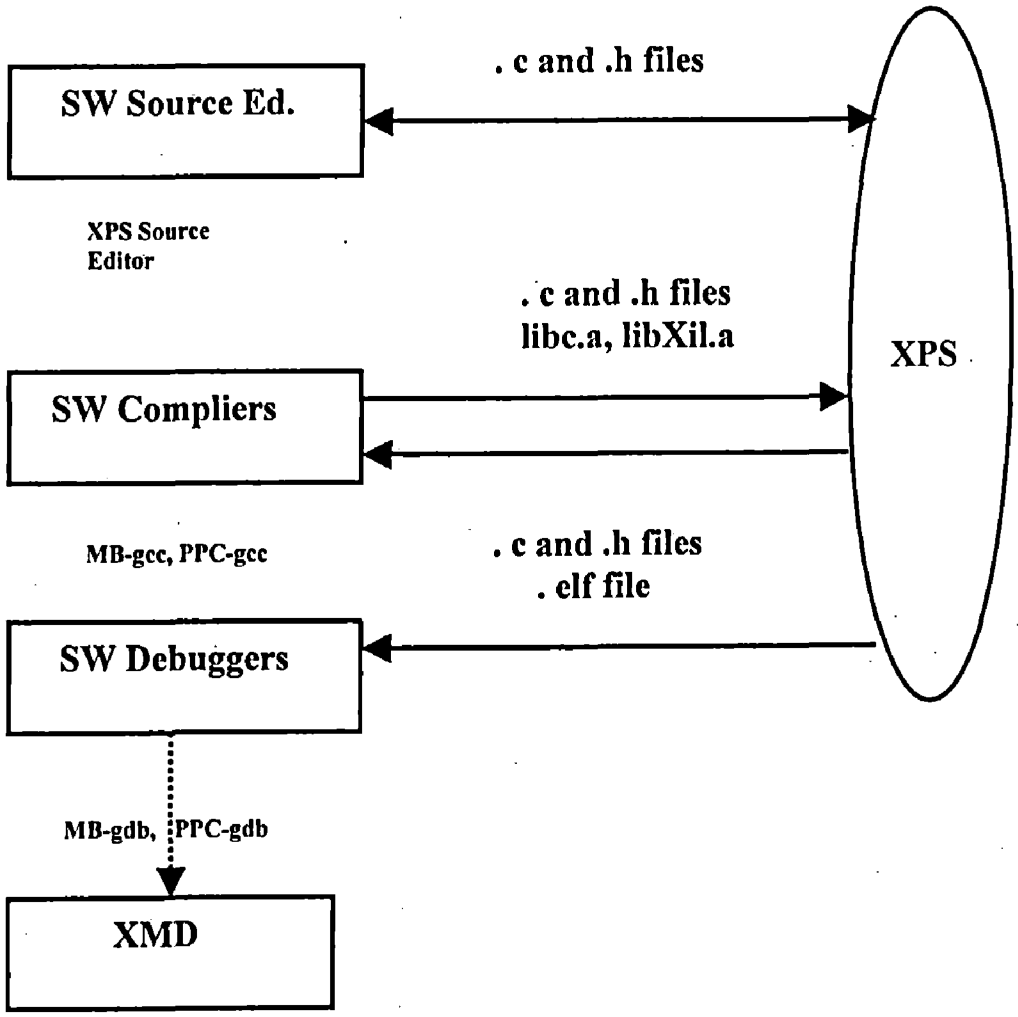


Figure 2-5 Software Application Creation and Verification

These processes help in the hardware and software co-design.

CHAPTER-3
BUS BRIDGES IN EMBEDDED SYSTEMS

3.1. Introduction:

There are large numbers of buses following different standards for communicating with th peripherals attached to them. If a peripheral attached to a particular bus wants to carry out an transaction with any other peripheral attached to different bus then it will cause a problem i carrying out the desired operations. To solve this issue bridges are designed so that they take care o all the factors to make the communication work.The bridge takes care of all the issues of differen buses operating at different frequencies, and having different data width. There are large number o bridges like PCI bridge that is used for providing the compatibility between the Processor and th outer peripherals. Similarly when a system is designed on the chip then processor will have its ow bus system and it will work at higher speed then the other peripherals. So a bridge becomes necessity to connect on chip peripherals and also off chip peripherals.

2. Bridges in Embedded system design.

In embedded system there are two different types buses .The PLB and the OPB.So there are two bridges that are designed for carrying out the transactions between the devices connected to the either bus.

The two bridges available are:

1. PLB to OPB Bridge.
2. OPB to PLB Bridge.

These bridges and the Processor Local Bus (PLB) are supported only with PowerPC, which are available with the Xilinx Virtex-2 Pro Devices onwards. Xilinx Microblaze soft processor core is used for designing simple systems with OPB.Microblaze does not support bridges architecture and Processor Local Bus (PLB). Microblaze is attached directly with OPB bus. Figure 3-1 demonstrates how the PLB to OPB Bridge and OPB to PLB Bridge are connected for the purpose of development of system-on-a-chip design.

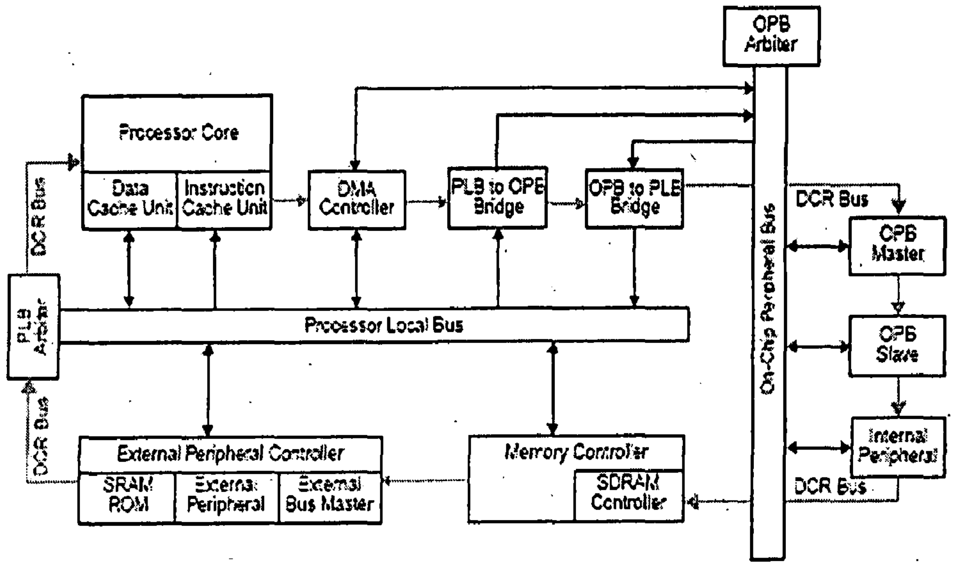


Figure 3-1 On-Chip Bus Structure

As shown in Figure 3-1, the on-chip bus structure provides a link between the processor core and other peripherals which consist of PLB and OPB master and slave devices.

The processor local bus (PLB) is the high performance bus used to access memory through the bus interface units. The two bus interface units shown above: external peripheral controller and memory controller are the PLB slaves. The processor core has two PLB master connections, one for instruction cache and one for data cache. Attached to the PLB is also the direct memory access (DMA) controller, which is a PLB master device, used in data intensive applications to improve data transfer performance.

Lower performance peripherals (such as OPB, master, slave, and other internal peripherals) are attached to the on-chip peripheral bus (OPB). A bridge is provided between the PLB and OPB to enable data transfer by PLB masters to and from OPB slaves. In the Figure 1 we have two bridges, a PLB to OPB Bridge which is a slave on the PLB and a master on the OPB and an OPB to PLB Bridge which is a slave on the OPB and a master on the PLB. OPB peripherals may also comprise DMA peripherals.

The device control register (DCR) bus is used primarily for accessing status and control registers within the various PLB and OPB masters and slaves. It is meant to off-load the PLB from the lower performance status and control read and writes transfers. The DCR bus architecture allows data transfers among OPB peripherals to occur independently from and concurrent with, data transfers between the processor and memory, or among other PLB devices.

3.3. PLB to OPB Bridge:

The Processor Local Bus (PLB) to On-chip Peripheral Bus (OPB) Bridge translates PLB transactions into OPB transactions. It functions as a slave on the PLB side and a master on the OPB side. The bridge is required in those system designs which have OPB slave devices, which must be accessed by the processor.

Features of PLB to OPB Bridge[2]:

1. PLB slave device and OPB master device
2. External programmable address space via address decode pin (this allows the PLB to OPB bridge to be located anywhere in the address map.
3. Supports 8 PLB masters.
4. 64-bit PLB slave interface supports.
   - Double Word (64-bit) writes or (32-bit writes) and Word (32-bit) reads
   - All partial transfers
5. Supports word, halfword, and byte burst reads and writes, including fixed-length bursts.
6. Supports pipelining for read transfers.
7. Compliant with quad-, and octal-word bursts for upward compatibility.
8. Supports 4-, 8-, and 16-word line transfers.
9. OPB master performs dynamic bus sizing for varying width slave devices
10. Bus error log accessible through device control registers

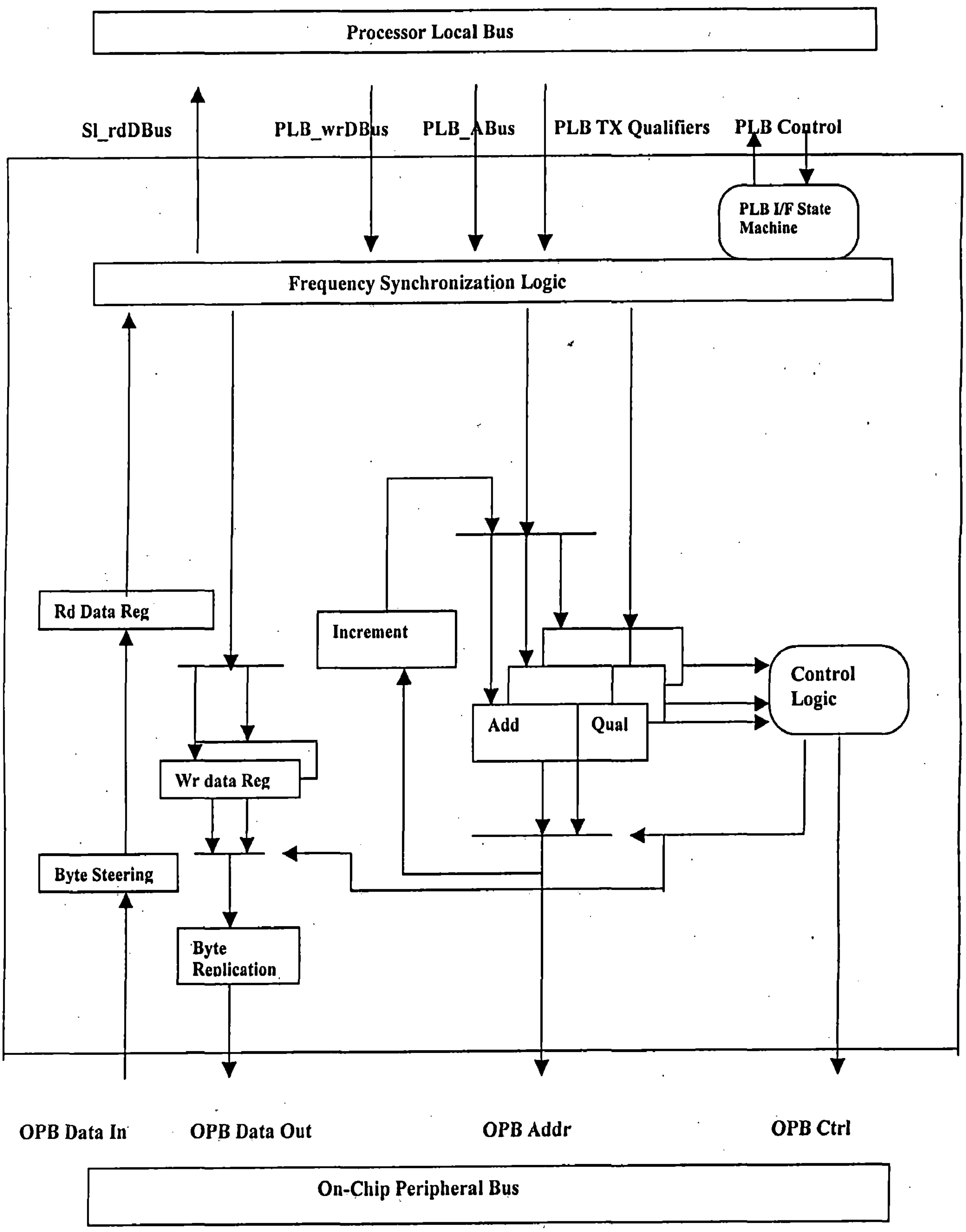


Figure 3-2 Block Diagram of PLB to OPB Bridge

11. 66+ MHz OPB clock frequency.

12. Support for PLB bus-speeds at 1x, 2x, 3x, or 4x the frequency of the OPB.

13. Watchdog timer for implementations omitting OPB arbiter.

3.3.1. Interfaces in PLB to OPB Bridge

PLB Interface:

The PLB to OPB Bridge [2] interfaces to the PLB as a 64-bit slave device for write operations, and as a 32-bit slave for reads. It has a single input for address decode so that the OPB slaves may be relocated in the system address map. Thc PLB to OPB bridge has the logic which steps down the PLB frequency by an integer amount (2:1, 3:1, etc.), allowing the OPB side of the bridge to run at the same (slower) frequency as the OPB. The PLB side always runs at the PLB bus frequency. All PLB to OPB bridge operations on the OPB are performed in the order accepted by the PLB. This insures that coherency is preserved.

OPB Interface:

The PLB to OPB Bridge interfaces to the OPB as a 32-bit master device. It fully implement the OPB architecture on the OPB side, performing dynamic bus sizing transfers as necessary. In cas when the PLB to OPB Bridge is the only master device on the OPB, and the arbiter need not b attached.

PLB to OPB Bridge Buffering:

The PLB to OPB bridge buffering has three sections called data buffering, address and transfer qualifiers buffering and error registers.

Data Buffering:

The PLB to OPB Bridge contains an 8 byte write data buffer, dynamically configurable as a single 64-bit, or two independent 32-bit data registers, and a separate 4 byte read data register. The write buffer resources are dynamically configured to the width of the requested write data transfer.

Address and Transfer Qualificrs Buffering:

The PLB to OPB Bridge contains three sets of address and PLB transfer qualificr registers. These FIFO address and qualificr registers are dynamically allocated between reads and writes. The PLB to OPB Bridge may accept a secondary read request or a second primary write request, depending on the current allocation of data buffer resources and the state of the current transfer.

Error Registers:

The PLB to OPB Bridge also contains a set of error reporting registers accessed through the device control register (DCR) bus. There are two 32-bit registers, one for the address that the error occurred at - Bridge Error Address Register (BEAR), and one that contains what type of error occurred and for which master - Bridge Error Status Register (BESR).

Address Registers:

Separate 32-bit PLB addresses are registered by the PLB to OPB Bridge for each operation. Byte addresses are incremented as necessary to implement dynamic bus sizing on the OPB. Word addresses are incremented only for PLB burst and line transfers.

3.4. OPB to PLB Bridge:

The On-Chip Peripheral Bus (OPB) to Processor Local Bus (PLB) Bridge module translates OPB transactions into PLB transactions. It functions as a slave on the OPB side and a master on the PLB side. The OPB to PLB Bridge [4] is necessary in systems where an OPB master device, such as an OPB based coprocessor, requires access to PLB devices (i.e. high speed memory devices, etc.).

Features of the OPB to PLB Bridge Include:

1. Serves as OPB slave device and/or PLB master device.
2. Can be mapped to any OPB address space.
3. As a 64-bit PLB master interface it supports.
   – Doubleword (64-bit) reads, and
   – Doubleword, word, halfword, and byte write
4. Provides data packing on writes, up to 4-doublewords
5. Operates at 66+ MHz OPB clock frequency
6. Supports PLB at 1, 2, 3, or 4 times the frequency of the OPB
7. Utilizes clock and power management

The block diagram of the OPB to PLB Bridge [4] is shown below in Figure 3-3

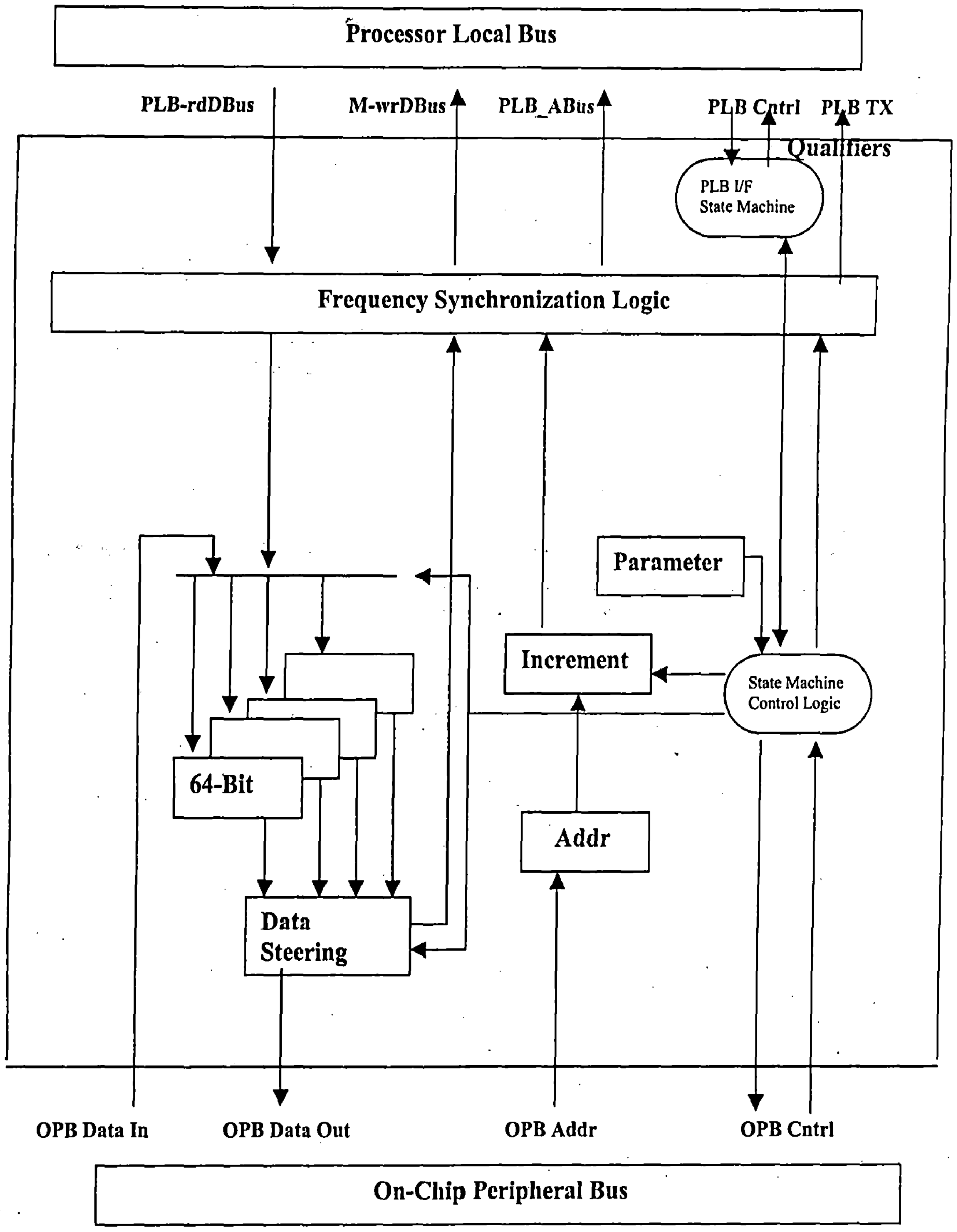


Figure 3-3 Block Diagram of OPB to PLB Bridge

Any application that has an OPB and a PLB can operate in one of two modes:

1. The XBM (external bus master) mode.
2. The OPB mode.

In the XMB mode the OPB to PLB Bridge uses four byte enable signals to decode requested data sizes and the core accepts unaligned half words and three-byte transfers.
In the OPB mode the OPB to PLB Bridge uses the OPB_fwXfer, OPB_hwXfer, and OPB_ABus (30:31) to decode requested data sizes. The core will only respond properly to byte, aligned halfword, and fullword transfers, as defined by the OPB architecture.

3.4.1. Interfaces in OPB to PLB Bridge

PLB Interface:

The OPB to PLB Bridge interfaces to the PLB as a 64-bit master device. It supports all single write transfers, and fixed-length two, three, or four doubleword write bursts. It performs a doubleword read to service all single read requests, and fixed length read bursts and four doublewords to service read burst requests.

OPB Interface:

The OPB to PLB Bridge interfaces to the OPB as a 32-bit slave device. It is selected by an externally decoded select line, so that the PLB slaves may be mapped anywhere in the OPB address map. The OPB to PLB Bridge supports 64-bit and 32-bit slaves. Conversion cycles are performed where necessary to support 32-bit slaves.

Address Registers

The OPB to PLB bridge contains a single 32-bit OPB address and transfer qualifier register. Addresses are incremented as necessary to affect transfers on the PLB.

Internal Data Buffer Structure

PLB Bridge contains a 32-byte data buffer, used for both read and write operations. During write operations, the buffer is organized as a 4-doubleword FIFO .During read operations; the buffer is organized as a fully associative cache with a line size of one fullword. Operation of the buffer differs for both read and write requests depending on the state of the OPB_seqAddr signal.

4-1 INTRODUCTION

In the design of embedded system processors are considered to be main unit. Xilinx have provided both the soft processor core and hard processor cores.Microblaze[8] is the soft processor and Power PC is the hard processor core. This section gives the brief introduction of design of embedded systems using MicroBlaze and Power PC. The architectural details of the two processors, OPB bus and peripherals are discussed in brief.

4-2 MICROBLAZE

The Microblaze[8] is an embedded soft core RISC (Reduced Instruction Set Computing) optimized for implementation in Xilinx FPGA's and it operates at 125 MHz.
The architecture of MicroBlaze is given in figure 4-1.

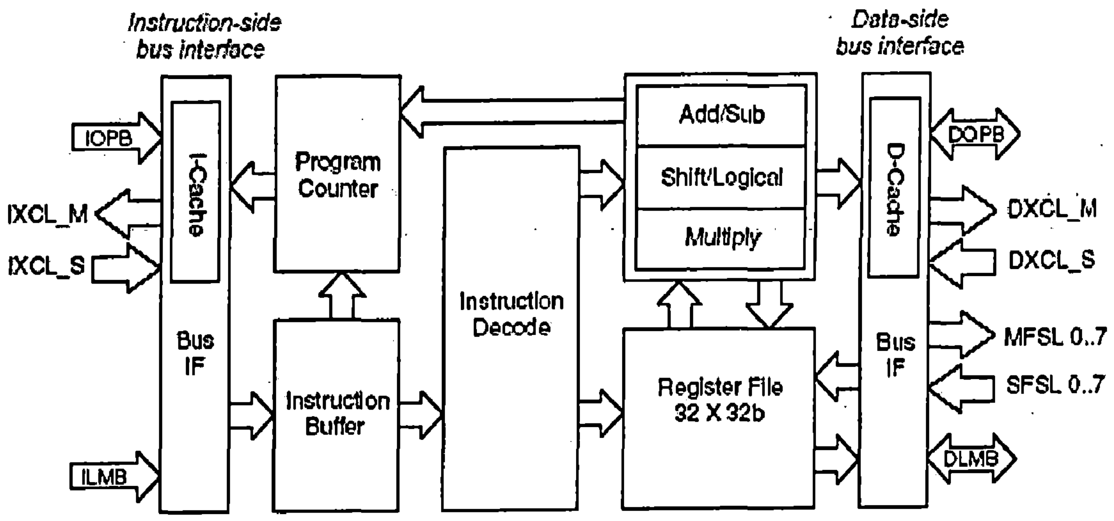


Figure 4.1 Microblaze core Block Diagram

MicroBlaze core implements separate buses for instruction fetch and data access, denoted by the I-side and D side buses, respectively.
These buses are split into two buses types:

1. OPB buses for OPB peripherals and memory controllers.

2. Local Memory Bus used exclusively for high speed access to internal block RAM (BRAM).

DOPB: Data interface, On-chip peripheral bus.

DLMB: Data interface, Local Memory bus (BRAM only).

IOPB: Instruction interface, On-chip peripheral bus.

IOPB: Instruction interface, Local Memory bus (BRAM only).

Features of Microblaze:

Thirty-two 32-bit general purpose registers

- 32-bit instruction word with three operands and two addressing modes.
- Separate 32-bit instruction and data buses that conform to IBM's OPB (On-chip Peripheral Bus) specification.
- Separate 32-bit instruction and data buses with direct connection to on-chip. Block RAM through a LMB (Local Memory Bus).
- 32-bit address bus
- Single issue pipeline
- Instruction cache
- Data cache
- Fast Simplex Link (FSL) support.

4-3 POWER PC

The IBM Power PC [11] is a RISC processor which can provide performance of up to 300+MHz. The PPC405 RISC CPU can execute instructions at a sustained rate of one instruction per cycle. On-chip instruction and data cache reduce design complexity and improve system throughput.

The features of Power PC are:

- Implements the PowerPC User Instruction Set Architecture (UISA) and extensions for Embedded applications.
- Thirty-two 32-bit general purpose registers (GPRs).
- Hardware multiply/divide for faster integer arithmetic (4-cycle multiply, 35-cycle divide).

- Big/little endian operation support.
- Separate instruction and data cache units.
- Virtual mode memory management unit (MMU) which can translate the 4 GB logical address space into physical addresses.
- Debug support which can be in internal mode or external debug mode or real time debug support.
- Two hardware interrupt levels support.
- Low Power Consumption 0.9mW/MHz.
- Dedicated On Chip Memory (OCM) Interface

The organization of Xilinx Power PC is given below in Figure 4-2

Central Processing Unit (CPU)

- Five Stage Instruction pipeline consisting of fetch, decode, execute, write back and load write back stages.
- Fetch Queue.

Memory Management Unit (MMU)

- Supports 4GB Address Space.
- Provides Address Translation, protection functions and storage attribute control for the address space.
- TLB used to control memory translation and protection, avoids TLB contention.

Instruction and Data Caches

- Each array is of 16KB size.
- Each is 2 way set Associative.
- ICU supplies two instructions every cycle to the fetch and decode unit.

64 Bit incrementing Time Base Counter.

- Programmable Interval Timer (PIT) - 32 Bit register that is decremented at time base incrementing frequency. PIT interrupt occurs when PIT Count reaches 0
- Fixed Interval Timer - Causes an interrupt when a selected bit in the time base register changes from 0 to 1.

- Watchdog Timer - Causes a hardware reset when a selected bit in the time base register changes from 0 to 1.

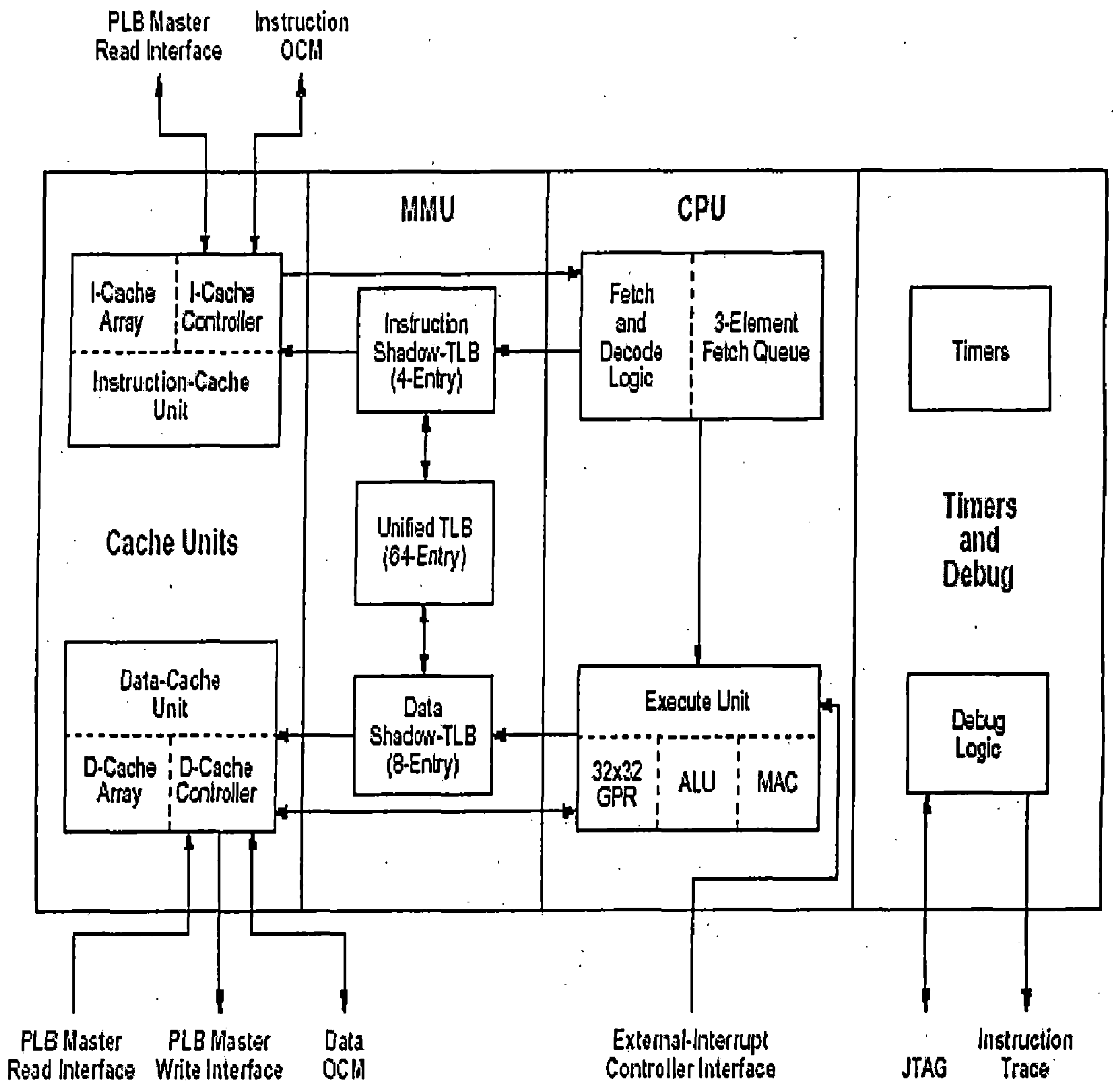


Figure 4-2 Power PC Organization

Special Debug modes that support various types of debugging.

- Internal Debug mode for use by ROM monitors and software debuggers.
- External Debug mode for use by JTAG debuggers.

- Debug wait mode, which allows the servicing of interrupts while the processor appears to be stopped.
- Real time trace mode, which supports event triggering for real time tracing.

Internal Debug and External Debug mode can be enabled simultaneously

4-4 Design of embedded system using Microblaze

Figure 4-3 shows the overall idea of how to design the embedded system using Microblaze. Microblaze is soft processor that acts as a master on the OPB bus. There can be other masters on the OPB bus as well like DMA controller. Peripherals like UART, Interrupt controller, GPIO (General Purpose Input Output) acts as slaves on the OPB bus.
These cores are provided by the Xilinx IP's.The whole design can implement the complete system on chip (SoC) design.

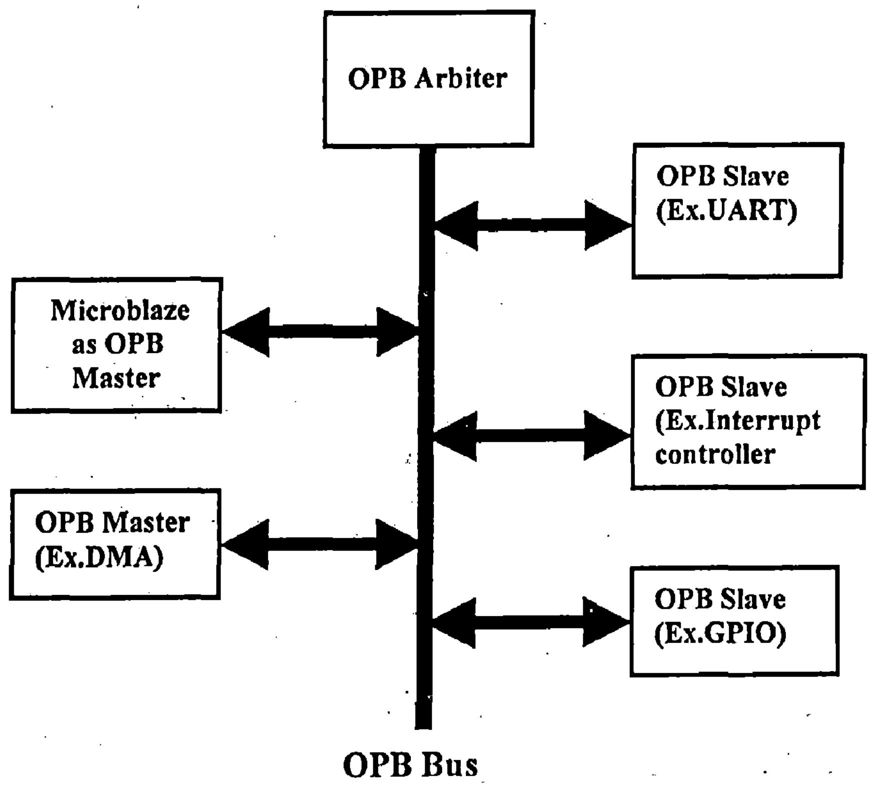


Figure 4-3 Embedded system design using Microblaze

4-5 Design of embedded system using Power PC

The design of embedded system using the Power PC is shown below in the Figure 4-4. The block diagram shows that more resources can be used when an embedded system is designed with Power PC[11].

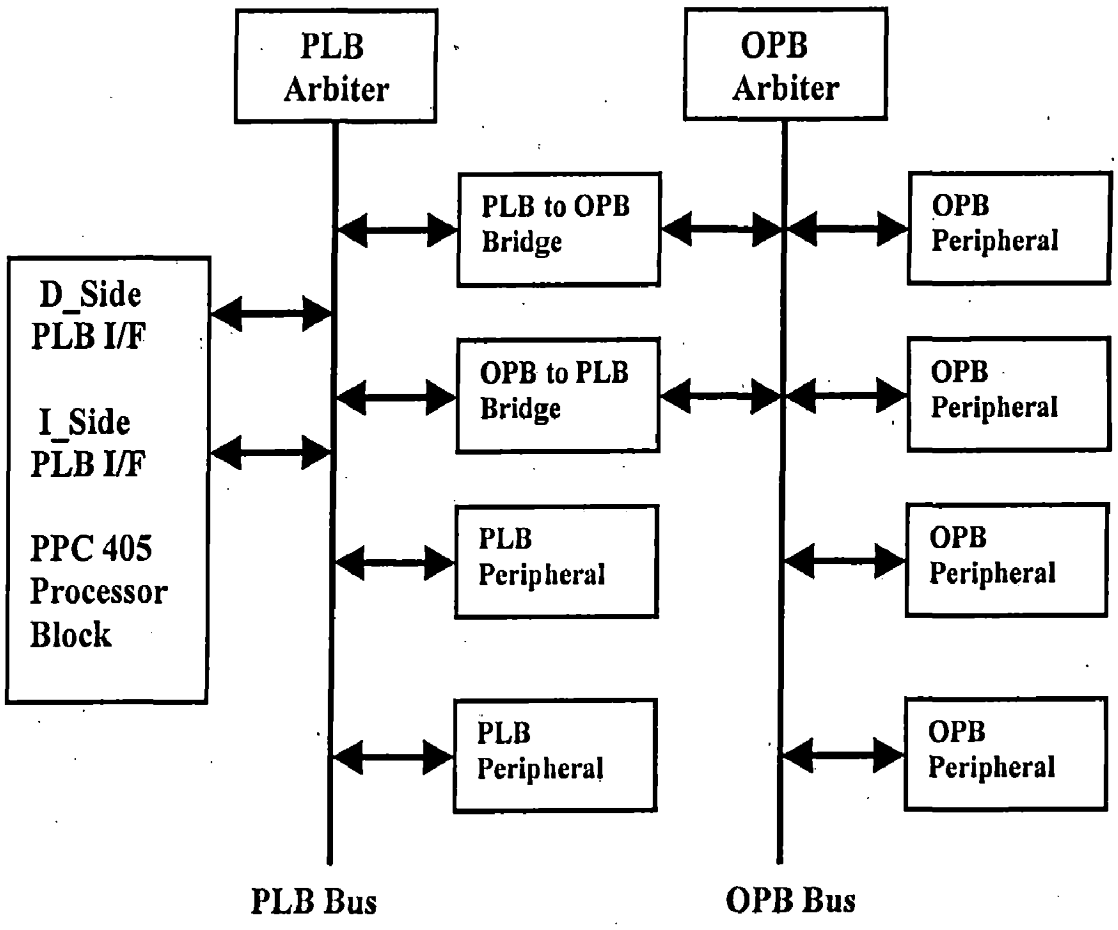


Figure 4-4 Embedded System using Power PC

Figure 4-4 shows the block diagram an embedded system that can be developed using PLB to OPB bus bridges and OPB to PLB bus bridges .These bridges facilitate the transfer of data between the peripherals of different speeds. With the help of these bridges it is possible to carry out the transaction between the processor and the fast peripherals that operate at system frequency and the slower peripherals. The processor and fast peripherals are connected to the Processor Local Bus (PLB) and slower peripherals that operate at much lower frequency like UART, USB, Ethernet card etc. are connected to the on-chip peripheral bus

(OPB)[10].MicroBlaze does not support these bridges as it is connected to the OPB only. Bridges help in designing the system that has multiple processing units.

4-6 On-Chip Peripheral (OPB) Bus

The OPB[10] is full featured bus architecture and is one element of the IBM Coreconnect architecture. It is general–purpose synchronous bus designed for easy connection of on-chip peripheral devices. The OPB interface connects both on-chip and off-chip peripherals and memory. Most of the features of OPB map well to the FPGA architecture ,however some can lower system clock rates.So,Xilinx uses an efficient subset of the OPB for Xilinx developed OPB devices.

Features of OPB:

1. Fully synchronous.
2. 32-bit address bus, 32-bit data bus.
3. Single transfer of data between OPB master and OPB slave.
4. Supports master byte enables.
5. Supports slave timeout suppress.
6. Supports slave retry.
7. No tri-state drivers required.

4-7 General Purpose Input Output (GPIO)

The GPIO is a 32-bit peripheral that attaches to OPB.

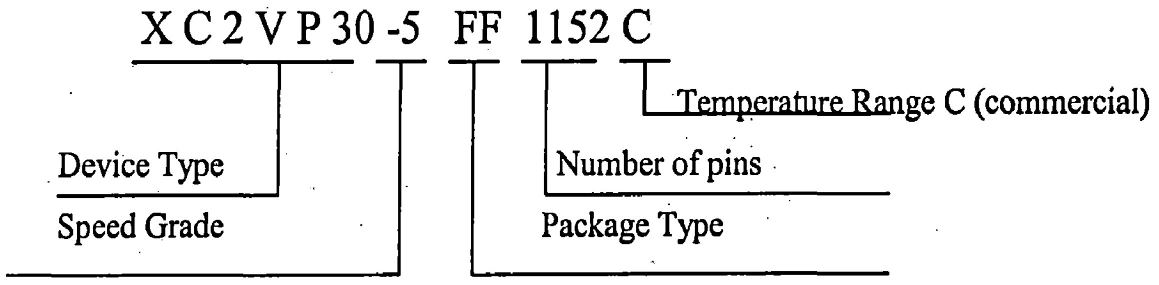
The features of GPIO are given below:

1. Configurable as single or dual GPIO channel(s).
2. Each GPIO bit dynamically programmable as input or output.
3. Number of GPIO bits configurable from 1-32 bits.
4. Can be configured as inputs-only to reduce resource utilization.
5. Ports for both three-state and non-three-state connections.
6. Optional Interrupt request generation.
7. Independent reset values for each bit of all registers.

4-8 Virtex-II Pro FPGA

The Virtex-II Pro Platform FPGA [14] is said to be the most technically advanced silicon and software product development in the history of the programmable logic industry. The Virtex-II Pro family FPGA has changed the technology focus from programmable logic to programmable systems, with enormous applications in leading areas of system architectures in networking applications, multimedia applications embedded systems, and digital signal processing systems. It allows custom user-defined system architectures to be synthesized and complex hardware and software systems to be co-developed rapidly. It also helps in-system debugging at system speeds.

Virtex-II Pro ordering Information:



Virtex-II Pro devices are user-programmable gate arrays with configurable elements and embedded blocks optimized for high density and high performance system designs. A brief overview of the components of Virtex-II Pro is:

1. **Embedded High-Speed Serial Transceivers:**

   These devices have RocketIO Multi-Gigabits Transceivers which is a flexible parallel-to-serial and serial-to-parallel embedded transceiver used for high bandwidth interconnections between buses, back-planes and other subsystems. Multiple user instantiations are possible, with a speed of upto 120 Gb/s of full-duplex raw data transfer. Each channel can be operated at a maximum data transfer rate of 3.125 Gb/s.

2. **Power PC:**

   The Power PC is the hard processor core that is embedded into the FPGA fabric. The architectural details of Power PC are discussed above.

3. **Input/Output Blocks (IOBs):**

   I/O blocks provide the interface between package pins and the internal configurable logic. Most popular and leading-edge I/O standards are supported by the programmable IOBs

4. **Configurable Logic Blocks (CLBs):**

   Configurable Logic Blocks (CLBs) provide functional elements for combinatorial and synchronous logic, including basic storage elements. CLB resources include four slices and two 3-state buffers.

   Each slice is equivalent and contains:

   - Two function generators (F & G)
   - Two storage elements
   - Arithmetic logic gates
   - Large multiplexers
   - Wide function capability

The function generators F & G are configurable as 4-input look-up tables (LUTs), as 16-bit shift registers, or as 16-bit distributed Select RAM+ memory. In addition, the two storage elements are either edge-triggered D-type flip-flops or level-sensitive latches. Each CLB has internal fast interconnect and connects to a switch matrix to access general routing resources.

5 **Block Select RAM+ Memory:**

Block Select RAM+ memory modules provide large 18 Kb storage elements of True Dual-Port RAM. Block Select RAM+ memory is cascadable to implement large embedded storage blocks.

6 **Embedded 18-bit x 18-bit dedicated multiplier blocks:**

18-bit x 18-bit help in making Read/multiply/accumulate operations and DSP filter structures are extremely efficient.

7 **Digital Clock Manager (DCM):**

Digital Clock Manager (DCM) blocks provide self-calibrating, fully digital solutions for clock distribution delay compensation, clock multiplication and division. The DCM and global clock multiplexer buffers provide a complete solution for designing high-speed clock schemes. Up to eight DCM blocks are available. To generate deskewed internal or external clocks, each DCM can be used to eliminate clock distribution delay. The DCM also provides 90-, 180-, and 270-degree phase-shifted versions of its output clocks.

Virtex-II Pro Family has 10 members. In the dissertation XC2VP30 device is used which has following resources:

Number of Rocket IO Transceivers Blocks: 10

Power PC Processor Blocks: 2

Logic cells: 30816

Slices:13696

18 X 18 Multipliers Blocks:136

Block Select RAM+(18Kb blocks):136,Max.BlockRAM(kb):2448

DCM:8

Maximum User I/O Pads:644

4-9 DIGITAL TO ANALOG CONVERTER (DAC) THS8133-

A brief description of how DAC THS8133 [5] works is given below .It also shows how different signals are used to generate some pattern on the TV signal.

The features of DAC THS8133 are:

- Triple 10-bit D/A Converters
- Minimum 80 MSPS Operation.
- Direct Drive of Doubly-Terminated 75-W load into Standard Video levels.
- 3×10 Bit 4:4:4, 2×10 Bit 4:2:2 or 1×10 Bit 4:2:2 (ITU-BT.656) Multiplexed YPbPr/GBR Input Modes.
- Bi-Level (EIA) or Tri-Level (SMPTE) Sync Generation With 7:3 Video/Sync Ratio.
- Integrated Insertion of Sync-On-Green/Luminance or Sync-On-All channels.
- Configurable Blanking Level.
- Internal Voltage Reference.

The THS8133 is a general-purpose triple high-speed D/A converter (DAC) used in video/graphics applications. The device operates from a 5-V analog supply and a 3-V to 5-V range digital supply. The THS8133 has a sampling rate up to 80 MSPS. The device consists of three 10-bit D/A converters and additional circuitry for bi-level/tri-level sync and blanking level generation in video applications.

THS8133 is suited for applications High-Definition Television (HDTV) Set-Top Boxes/Receivers, High-Resolution Image Processing and in communication systems.
Its output drivers are specifically designed to produce standard video output levels when directly connected to a single-ended doubly-terminated 75 W coaxial cable.

The THS8133 can generate both a bi-level sync or a tri-level sync signal, as per the SMPTE standards, via a digital control interface. The sync signal is inserted on one of the analog output channels (sync-on-green/luminance) or on all output channels. Also, a blanking control signal sets the outputs to defined levels during the nonactive video window.

The input format can be either 3×10 bit 4:4:4, 2×10 bit 4:2:2 or 1 × 10 bit 4:2:2. This enables a direct interface to a wide range of video DSP/ASICs including parts generating ITU-BT.656 formatted output data. The block diagram of THS8133 is given below in Figure 4-5.

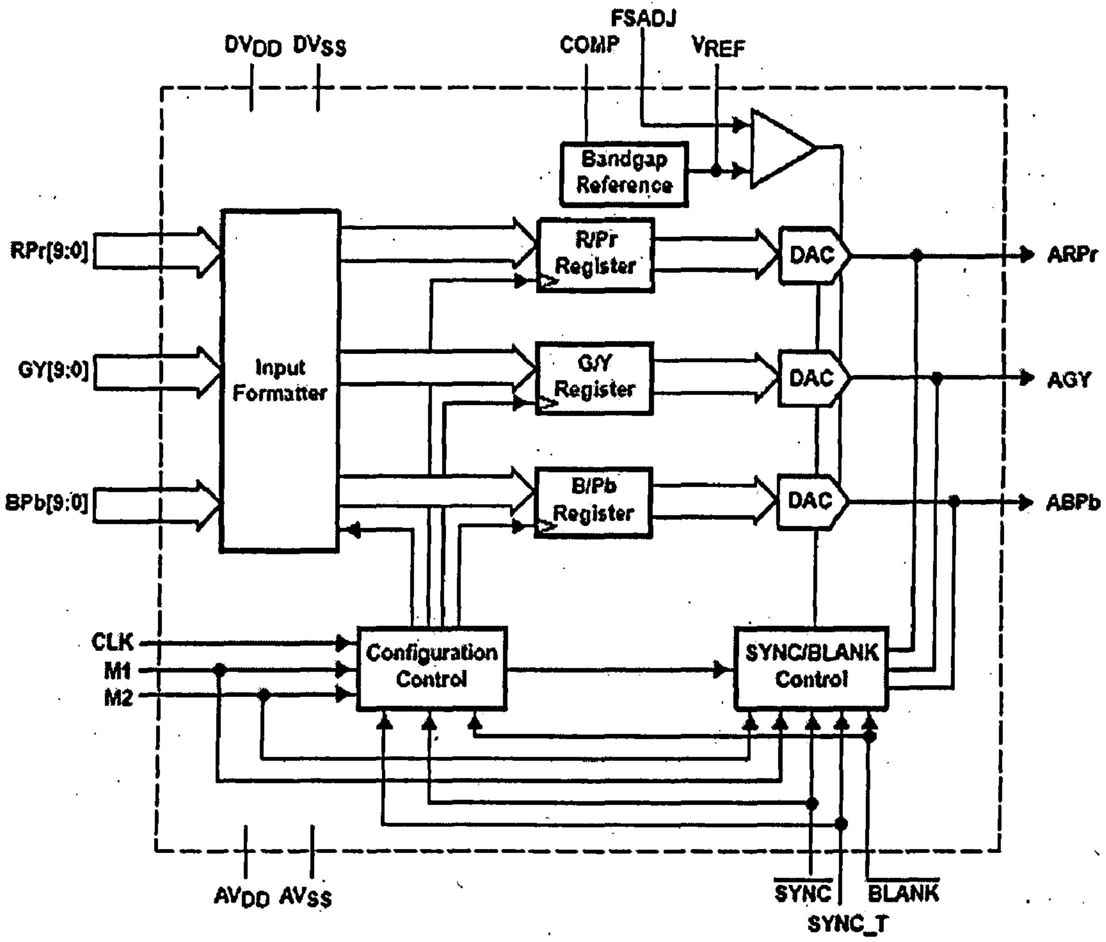


Figure 4-5 Block Diagram of DAC THS8133

The brief descriptions of various signals used are:

BPb0-BPb9, GY0-GY9 and RPr0-PPRr9 are pixels Input corresponding to Blue,

Green and Red signals respectively. ABPb, AGY, ARPr: These are the analog red, green and blue respectively Pr, Y and Pb current outputs, capable of directly driving a doubly terminated 75-Ω coaxial cable.

$\overline{BLANK}$: It is active low Blanking control input, and it is latched on the rising edge of CLK. When asserted, the ARPr, AGY and ABPb outputs are driven to the blanking level, irrespective of the value on the data inputs.

$\overline{SYNC}$: It is also the active low Sync control input which is latched on the rising edge on

CLK .When asserted, only the AGY output or ARPr, AGY and ABPb outputs are driven to the sync level, irrespective of the values on the data or $\overline{BLANK}$ inputs. $\overline{SYNC}$ takes precedence over $\overline{BLANK}$, so asserting $\overline{SYNC}$ (low) while $\overline{BLANK}$ is active (low) will result in sync generation.

SYNC_T: This signal is Sync tri-level control, active high. It is latched on rising edge of the CLK. When asserted, a positive sync (higher than blanking level) is generated when $\overline{SYNC}$ is low. When disabled, a negative sync (lower than blanking level) is generated when $\overline{SYNC}$ is low. When generating a tri-level (negative-to-positive) sync, a L to H transition on this signal positions the start of the positive transition. The value on SYNC_T is ignored when $\overline{SYNC}$ is not asserted (high).

CLK: It is the Clock input. A rising edge on CLK latches RPr0-9, GY0-9, BPb0-9, $\overline{BLANK}$, $\overline{SYNC}$, and SYNC_T. The M2 input is latched by a rising edge on CLK also, but only when additional conditions are satisfied.

M1: This is the configuration signal used for the mode control 1.It is directly interpreted by the device and is not latched by the CLK.

M2: This is also the configuration signal. It is latched when the second rising edge on CLK takes place after the transition on $\overline{SYNC}$.

The interpretation of M2 is dependent upon the polarity of the last $\overline{SYNC}$ transition. When $\overline{SYNC}$ L (Low) goes to H (High) M2 is latched as M2_INT.When $\overline{SYNC}$ goes H (High) to L (Low) M2 is latched as INS3_INT.The Pin Diagram of THS8133 DAC is given below in the Figure 4-6

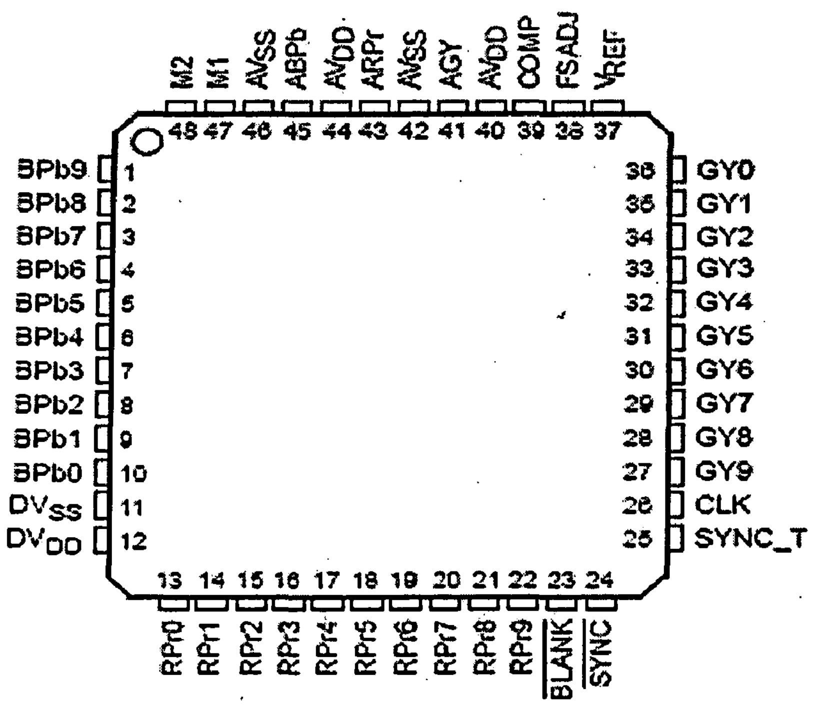


Figure 4-6 THS8133 DAC Pin Diagram

The device is configured using M1, M2_INT as shown in Table 4.1.

M1	M2_INT	CONFIGURATION	DESCRIPTION
L	L	GBR 3x10b-4:4:4	GBR mode 4:4:4. Data clocked in on each rising edge of CLK from G, B, and R input channels.
L	H	YPbPr 3x10b-4:4:4	YPbPr mode 4:4:4. Data clocked in on each rising edge of CLK from Y, Pb and Pr input channels.
H	L	YPbPr 2x10b-4:2:2	YPbPr mode 4:2:2 2x10 bit. Data clocked in on each rising edge of CLK from Y & Pr input channels. A sample sequence of Pb–Pr–... should be applied to the Pr port. At the first rising edge of CLK after $\overline{BLANK}$ is taken high, Pb should be present on this port
H	H	YPbPr 1x10b-4:2:2	YPbPr mode 4:2:2 1x10 bit (ITU-BT.656 compliant). Data clocked in on each rising edge of CLK from Y input channel.

Table 4.1 THS8133 Configuration

One channel of Video DAC is used for monochrome image processing. In case of INS3_INT signal if this signal is high, the sync output is inserted on all DAC outputs and when it is low sync output will be inserted only on the AGY output. The value of M2 at power up is undetermined. Therefore at least 1 L->H transition on SYNC is required to set M2.

Virtex-II Pro FPGA and THS8133 DAC are used for multimedia application. The program is stored in the FPGA and then it controls the interfacing of the DAC THS8133. It resembles the complete system-on–chip design (SoC).The whole design is developed with the help of EDK 6.3 with the help of PowerPC and bus Bridges. Here as there is no slave on the PLB bus therefore OPB to PLB Bridge is not used.

4.10. Multimedia Application using the Virtex-II Pro

FPGA based board is used for TV interface according to CCIR/PAL[19] standard. The blanking and sync signal are generated to display the pattern on the TV monitor. A pattern is generated in FPGA which is displayed in the TV monitor with the help of THS8133 DAC.The general block diagram of the application is shown in the Figure 4-7 below:

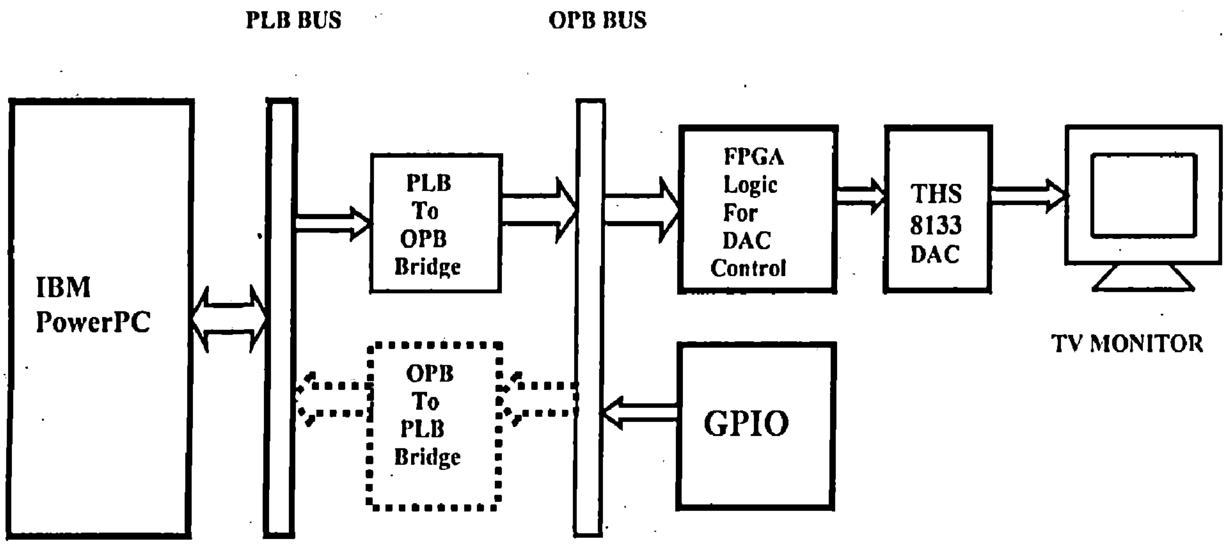


Figure 4-7 Multimedia Application using Bus Bridge

The figure 5-4 give the general block diagram of the Multimedia application using bus bridge .The function of different blocks is given below.

GPIO: It is General Purpose Input Output Block which is used for giving the input to the processor and transmits the output. It can be configured as input, output or both. It is configured as an output block to send the instruction to the processor .The instructions are written in the C code which are converted into the assembly level by the GNU C cross compiler .

Power PC: This is the processor core which is inserted in the FPGA fabric to carry out the instructions given by the user.
PLB Bus: This is the bus connected to the Power PC core. No peripheral is attached to this Bus as it is used for connecting the faster peripherals that operate at system frequency.

PLB to OPB Bridge: The link has been designed which connects the PLB and OPB Bus. This PLB to OPB bus bridge is the soft core available with in the Xilinx library .The parameters can be set according to the design requirements and the system requirements. This bridge acts as the slave on the PLB and master on the OPB bus. The OPB to PLB Bridge is used to map a range of PLB addresses onto OPB addresses.

OPB to PLB Bridge: This Bridge is not used in this application so it is shown dotted.
This bridge is used when there is some master on the OPB and it want to communicate with the peripheral attached to the PLB.
The create and import peripheral block has IPIF (Intellectual Property Interface) block which help in attaching the IP created in the design to the OPB.

FPGA Logic for DAC control: This is logic designed to control the DAC THS8133 .The Code for this is written in VHDL and it is attached to the OPB with the help of Create/Import Wizard of the EDK tool.
The block diagram of IPIF[13] connecting IP Core is shown in the figure 4-8. Base System Builder (BSB) is used to design the basic design using Xilinx Platform Studio (XPS).Then IP core is attached to the system using IPIF.

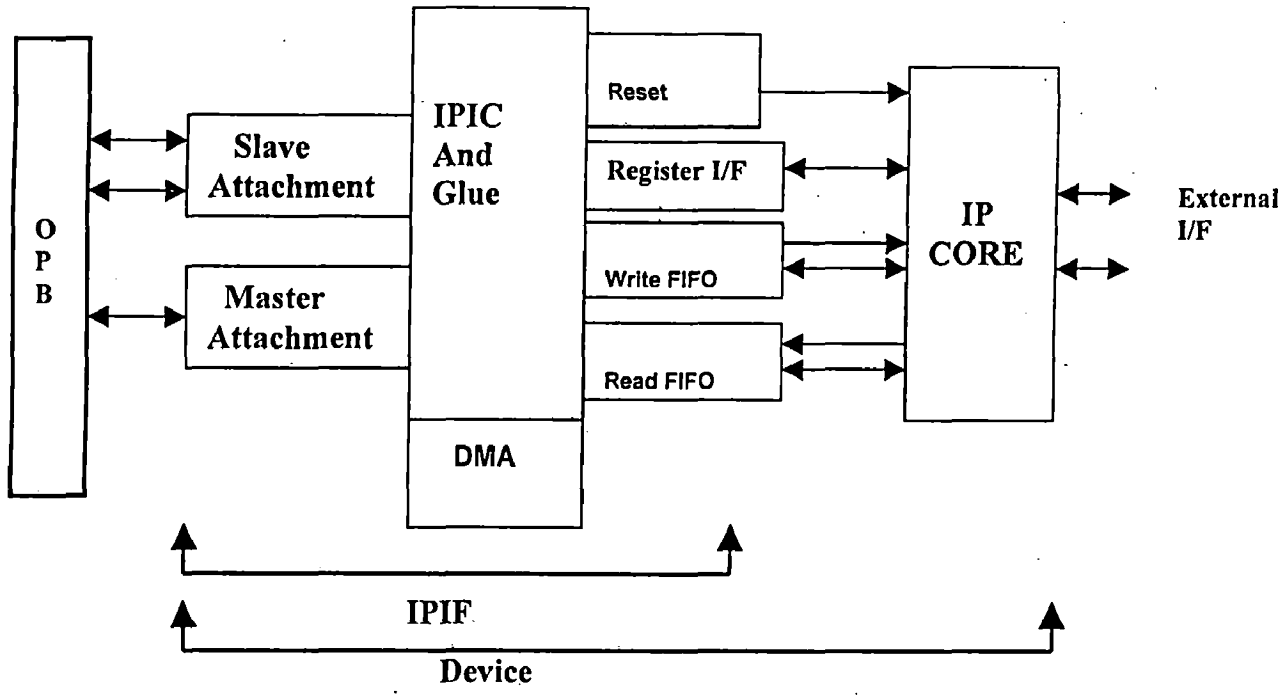


Figure 4-8 IPIF module connecting IP Core

The output from the design is used to control the output of the DAC so that a desired pattern can be generated. The output signals sent to DAC are used for configuring the various modes that are used to control the DAC.

DAC THS8133: This is the 10 bit Digital to analog converter used for video and graphics applications. The operation of this DAC THS8133 [5] is explained above.

The complete embedded system is generated and compiled using the EDK Version 6.3 tool. The different files that are generated during compilation along with the brief overview is given below .The design was tested successfully using Virtex-II Pro-PCI Video card.

FUTURE SCOPE

Virtex-II Pro is the first platform FPGA solution capable of implementing ultra high bandwidth SoC designs that were previously in the exclusive domain of ASIC's.The Virtex-II Pro presents all the capabilitiesof the ASIC's and still maintain the flexibilities and low development costs of programmable logic devices. The Virtex-II Pro solution enables the high performance programmable systems especially in the areas of wired and wireless networking, storage systems, professional broadcast, embedded systems, and digital signal processing systems.

With the help of EDK IP cores we can develop the complete embedded systems in any of the areas and carry out the real time test using the Virtex-II Pro FPGA's.The flexibility

Of the EDK to add IP cores from the third party or from the particular user helps in completely design the particular application with the minimum cost and less time. The future scope in this area is unlimited and it even depends relatively less on the market conditions because these are upward compatible of the tools and their flexible nature.

This approach helps in codesign in which software and hardware are connected simultaneously which is improvement of the earlier approach in which hardware and software platform were developed independently and there was problem in the integration of the platforms.

This thesis work can be extended to develop any video game. The PCI slot can also be used to send the data from the PC to the board through the PCI slot.

REFERENCES

1.IBM Coreconnect Architecture
http://www-03.ibm.com/chips/products/coreconnect/

2.User Guide 64-Bit PLB to OPB Bridge Core (SA-27E and Cu-11)Version 3.1 2nd Edition 2002
http://www306.ibm.com/chips/techlib/techlib.nsf/techdocs/Virtex 2 pro.docD5B8805C808F81E287256A2B00616A49

3.Product Brief -64-Bit PLB to OPB Bridge Core(SA-12)
http://www306.ibm.com/chips/techlib/techlib.nsf/techdocs/E2A66770 67F64EE287256B9E0 05ACBEB

4. User Guide 64-Bit OPB to PLB Bridge Core(C27E318_um.pdf)Version 3.3 2nd Edition 2002
http://www306.ibm.com/chips/techlib/techlib.nsf/techdocs/460F4792AC4DFE1C87256A2B 0063611B/$file/C27E318_um.pdf

5.Texas Instruments -THS 8133 Video DAC Data Sheet 08 September 2000
http://focus.ti.com/lit/ds/symlink/ths8133b.pdf

6.EDK 6.3 Getting started with EDK 6.3 August 10, 2004
http://www.xilinx.com/ise/embedded/edk6_3docs/edk_getstarted.pdf

7.Embedded System Tools Reference Guide August 20,2004
http://www.xilinx.com/ise/embedded/edk6_3docs/est_rm.pdf

8.Microblaze Reference Guide June 14,2004
http://www.xilinx.com/ise/embedded/edk6_3docs/mb_ref_guide.pdf

9.Platform Studio Tools User Guide August 24,2004
http://www.xilinx.com/ise/embedded/edk6_3docs/ps_ug.pdf

10.On-chip Peripheral Bus Architecture Specification Version 2.1 April 2001
http://www1.cs.columbia.edu/~sedwards/classes/2005/4840/opb_ibm_spec.pdf

11. Power PC 405 Processor Block Reference Guide August 20,2004
http://www.xilinx.com/bvdocs/userguides/ug018.pdf

12. Xilinx Device Driver Documentation 24 June,2004
http://www.xilinx.com/ise/embedded/edk6_2docs/xilinx_drivers.pdf

13. Usercore Templates Reference Guide January 2004
http://www.xilinx.com/ise/embedded/edk6_2docs/usercore_templates_ref_guide.pdf

14. Virtex-II Pro Platform FPGA Handbook October 24,2002
http://www.xilinx.com/bvdocs/userguides/ug012.pdf

15. J.Bhaskar, VHDL Primer 3rd Edition, Addision Wesley Longman Singapore Pte.Ltd.

16. Douglas L.Perry, Programming by Example 4th Edition, Tata McGraw Hill.

17. Ben Cohen, VHDL Coding Styles and Methodologies,2nd Edition Kluwer Academic Publishers.

18 Ram Kumar,Rakesh Agarwal, Programming inANSI C,Tata McGraw Hill

19. R R Gulati,Monochrome and Colour Television,New Age International (P)Limited, Publishers.

APPENDIX

Virtex-II Pro-PCI Video Board

Virtex-II Pro–PCI Video Board is used for developing multimedia application using Power PC with the help of Embedded Development Kit (EDK).

The applications where this card can be used is:

- Image Processing.
- Digital video Processing
- Digital TV
- Video capture
- Video editing
- Multimedia
- Embedded systems
- Embedded microprocessor
- Data storage.

Specifications of Virtex-II Pro-PCI Video board are :

Analog Input: ADC – 14 bit, 10 MSPS single channel is available using AD9240

Video Input: Video ADC – 3 channel, 8 bit, 30MSPS single Video Input channel is available using Video ADC TLV5734.

Video Output: DAC - Two Video Output channels provides NTSC, PAL compliant component video signal using 10 bit Video DAC THS8133.

User IO's:

1. 16 IO's are available with XC2VP30 device (Maximum 32 XC2VP40 & XC2VP50).
2. IO's are provided on two 20 pin box type FRC connectors.

Memory

1. 5 fully independent banks of 1M X 16 SRAM (NEC make uPD4416016).
2. 3 fully independent banks of 512K X 16 Flash PROM (LH28F800)

Serial Interface –

1. Two RS-232 channels (MAX3223)
2. One MIL-STD 1553 channel. (optional)

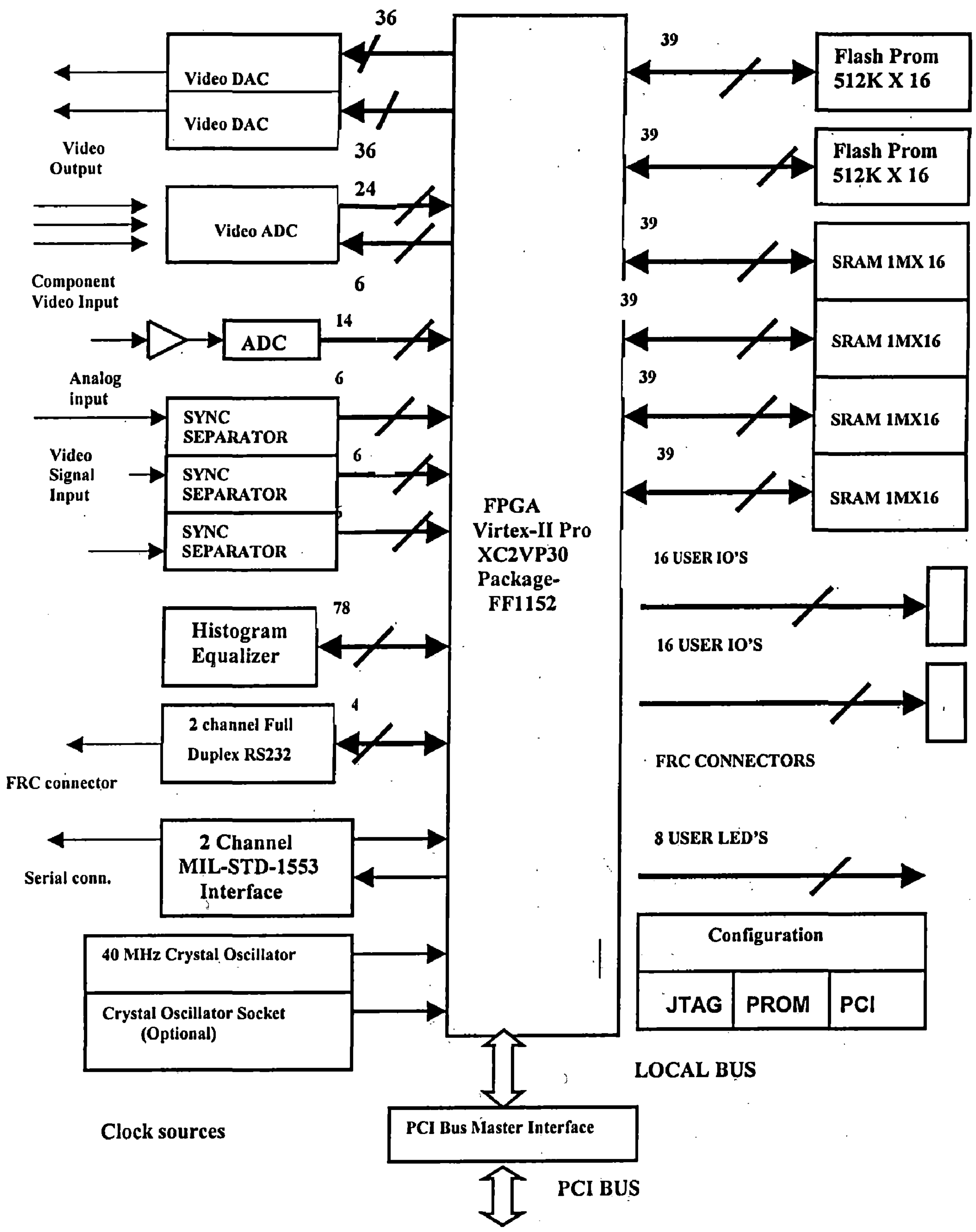


Figure A.1 Virtex-II Pro based video processing board with PCI Interface

PCI Interface:

32 bit 33 MHz master interface, with facility for DMA transfers.

Block Diagram of the Virtex-II Pro–PCI Vidco Board is shown above in the figure 5-2.

APPENDIX B

Program Code (VHDL)

```
--The VHDL code for the control of DAC THS8133 is given below.
--GIVEN BELOW IS THE PROGRAM FOR GENERATING
--ANY PATTERN USING THE THS8133 DAC.
--THE PROGRAM WAS TESTED SUCCESSFULLY ON VIRTEX-II PRO
--FPGA'S.HERE MOVING I I T PATTERN IS GENERATED
--SIGNALS gen_1,gen_2 and gen_t are used
--respectively for generating
--the I,I and T RESPECTIVELY.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity pattern is
   Port (
        rst,clk,gen_1,gen_2,gen_t:in std_logic;
   dac_clk, sync_dac, blank_dac:out std_logic;--DAC SIGNALS
        mode_1, mode_2:out std_logic;    -- DAC SIGNAL FOR MODE SELECTION Y
CHANNEL IS SELECTED
        sync_t:out std_logic;
   dac_data:out std_logic_vector(9 downto 0));-- DATA FOR PATTERN
       end pattern;

architecture Behavioral of pattern is

signal sync,m2_s1,m2_s2,rst_en:std_logic; -- signal for dac intialization for selecting particular
channel
```

```
signal Config_Over,master_en:std_logic;
signal  clk_8m :std_logic;
signal clk_div  :std_logic_vector(5     downto 0);
signal     hblank,hblank_1,hblank1,vblank,vblank1,frame_marker:std_logic;         --signals     for
horizontal blanking vertical blanking ,composite blanking
signal count_h :std_logic_vector(15   downto 0);--counter for horizontal pixel counting 320 pixel
trace period and 192 retrace period total 512 pixel
signal count_v :std_logic_vector(25 downto 0);
signal hsync,serr1:std_logic; --signals for horizontal and vertical syncronization. these signal is
generated in between the blanking period
signal en_p1,en_p2,en_p3,en_p4,en_i1,en_i1v,en_i2,en_i2v          :std_logic_VECTOR(9
        DOWNTO 0);
signal add_en,sync_dac1,sync_dac_m:std_logic;      --
type state is (start,sync_h_l,dly1,dly2,dly3,sync_l_h,dly4,dly5,dly6,dly7);
signal  ps,ns: state;
begin

--===CLOCK DIVIDER FOR 8MHz CLOCK ========

process(clk,rst)
begin
  if rst='1' then
    clk_div<= "000000";
    elsif clk'event and clk ='1' then
    clk_div<= clk_div+ '1';
  end if ;
 end process;
clk_8m        <= clk_div(1);

--===FSM FOR CONFIGURATION SELECTION======
process(rst,clk_8m)
 begin
   if rst='1' then
   ps<=start;
      elsif clk_8m'event and clk_8m='0' then
    ps<=ns;
   end if ;
            end process;

process(ps,rst)
        begin
               case ps is
                      when start          => ns<=sync_h_l;
                      when sync_h_l       =>    ns<=dly1;
                      when dly1           =>    ns<=dly2;
                      when dly2           =>    ns<=dly3;
                      when dly3           =>    ns<=dly4;
                      when dly4           =>    ns<=dly5;
                      when dly5           =>    ns<=dly6;
                      when dly6           =>    ns<=dly7;
                      when dly7           =>    ns<=sync_l_h;
                      when sync_l_h       =>    ns<=sync_l_h;
                      when others         =>    ns<=sync_h_l;
               end case;
        end process;

process(ps)
 begin
  case ps is
   when start       =>      sync              <='1';
                                   Config_Over  <='0';
```

```
                                        master_en     <='0';
   when sync_h_l => sync                       <='0';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly1      => sync                         <='0';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly2      => sync                         <='0';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly3      => sync                         <='1';
                                 Config_Over  <='0';
                                        master_en     <='0';
       when dly4      => sync                         <='1';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly5      => sync                         <='1';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly6      => sync                         <='1';
                                        Config_Over  <='0';
                                        master_en     <='0';
       when dly7      => sync                  <='1';
                                        Config_Over  <='1'; -- DAC configuration over in this
cycle
                                        master_en     <='1';
       when sync_l_h =>    sync                       <='1';
                                        Config_Over  <='1';
                                        master_en     <='0';
      end case;
  end process;
```

```
 ---==end of mode selection for dac=========


 process(rst,clk_8m,sync)
  begin
   if rst='1'  then
     m2_s1 <='1';
       m2_s2 <='1';
         elsif clk_8m'event and clk_8m='1' then
       m2_s1 <= sync;
    m2_s2 <= m2_s1;
   end if ;
 end process;
--==GENERATE H-blanking===================


process(clk_8m,master_en,Config_Over,rst)
begin
  if  master_en ='1'  or rst = '1'  then
          count_h<=(others =>'0');
        elsif clk_8m'event and clk_8m = '1' then
                if Config_Over='1' then
                        if count_h = 511 then
                          count_h <= (others =>'0');  --        H line period counter
                        else
                      count_h <= count_h +'1';
              end if;
             end if;
   end if;
 end process;


-----------------------------------------------

 process(rst, clk)
 begin
   if rst='1' then
                    hblank <= '1' ;
                                vblank <= '1' ;
                                sync_dac_m <= '1' ;
               elsif clk'event and clk='1' then
                                hblank <= hblank1 ;
                                vblank <= vblank1 ;
                                sync_dac_m <= sync_dac1 ;

     end if;
 end process;

process(rst, clk_8m)
 begin
 if rst='1' then
                 sync_dac <= '1' ;
              elsif clk_8m'event and clk_8m='1' then
             if Config_Over='1' then

           sync_dac <= sync_dac_m ;
                 else
                             sync_dac <= sync ;
end if ;
   end if ;
 end process;
 ----------------------------------------------------------

hblank1 <= '1' when (count_h < 416 ) and (master_en='0') else
   '0';
```

```
--==========I.I.T PATTERN=============

--generation of T --
en_p1 <= "1111111111"       when (count_h <= 280 )       else
        "0000000000"        WHEN (count_h > 280 AND count_h <= 370) else
                    "1111111111";

en_p2 <= "1111111111"       when COUNT_V <= 60000   else
      "0000000000"  WHEN COUNT_V > 60000 AND COUNT_V <= 75000 else
                    "1111111111" ;
en_p3  <=     "1111111111"        when (count_h <= 310 )       else
        "0000000000"        WHEN (count_h > 310 AND count_h <= 340) else
                    "1111111111";
en_p4 <=      "1111111111"        when COUNT_V <= 75000   else
      "0000000000"  WHEN COUNT_V > 75000 AND COUNT_V <= 100000 else
                    "1111111111" ;
-- Generation of  1st I --

en_i1 <= "1111111111"       when (count_h <= 60 )        else
        "0000000000"        WHEN (count_h > 60 AND count_h <= 85) else
                    "1111111111";
en_i1v <="1111111111"       when COUNT_V <= 60000   else
      "0000000000"  WHEN COUNT_V > 60000 AND COUNT_V <= 100000 else
                    "1111111111" ;
--Generation 0f 2nd I --
en_i2 <= "1111111111"       when (count_h <= 170 )       else
        "0000000000"        WHEN (count_h > 170 AND count_h <= 200) else
                    "1111111111";
en_i2v <= "1111111111"      when COUNT_V <= 60000   else
```

```
    "0000000000"   WHEN COUNT_V > 60000 AND COUNT_V <= 100000 else
                   "1111111111" ;


--generation of I.I.T --
dac_data <= (en_p1 or en_p2) and (en_p3 or en_p4) when (gen_t='1') else
        (en_i1 or en_i1v) when (gen_1='1') else
                   (en_i2 or en_i2v) when (gen_2='1') else
                    "1111111111";
--=====================================================
sync_t          <= '0';
mode_2                 <= m2_s2;
sync_dac1              <= hsync and serr1 ;
mode_1                 <= '0';
dac_clk                <= clk_8m;
blank_dac <= hblank and vblank when config_over = '1' else
               '0'     ;
end Behavioral;
```

APPENDIX C

Program Code (C)

C code for controlling the processor.

This C code gives the instructions to the microprocessor to carry out the desired operations.This C code is converted into the assembly code by GNU GCC compiler.Here the drivers are given in the EDK which controls the given devices .In this program the driver programs for GPIO's.

```
#include<xparameters.h>
#include<xgpio_l.h>
#include<xio.h>
#include<xgpio.h>

void delay(void);

main()

{
int i;
/*
 * Routine to write a pattern out to a GPIO
 * which is configured as an output
 *    PARAMETER C_ALL_INPUTS = 0
 */
 while (1)
 {
      XGpio_mSetDataDirection(XPAR_IITDISPLAY_0_BASEADDR,1,0x00000000);    /*
Set as outputs */
      XGpio_mWriteReg(XPAR_IITDISPLAY_0_BASEADDR,0x00,0x00000001);
      delay();
      XGpio_mWriteReg(XPAR_IITDISPLAY_0_BASEADDR,0x00,0x00000002);
      delay();
      XGpio_mWriteReg(XPAR_IITDISPLAY_0_BASEADDR,0x00,0x00000004);
      delay();

}
}
void delay(void)
{
int i;
for(i=0;i<=400000;i++)
{}
}
```

RESULTS:

Microprocessor Hardware Specification (MHS) File:

These file gives the information about the peripherals and processors used while designing the hardware system. It shows that it has Power PC ,LED's ,DIP switches, Reset Block, Bram Block ,PLB-IF- controller ,PLB-to-OPB Bridge, PLB bus ,OPB bus, iitrdisplay,DCM clock module in this hardware design. This file also gives the Base Address, Higher Address, Version, clock to which the module is connected nad also the bus interface i.e on which bus the component acts as slave or as master.

In short MHS file give the general overview otf the complete hardware involved in the design.The MHS file generated is given below.

```
##############################################################################
# Created by Base System Builder Wizard for Xilinx EDK 6.3 Build EDK_Gmm.10
# Thu Jun 23 14:57:58 2005
# Target Board:  Custom
# Family:  virtex2p
# Device:  xc2vp30
# Package:          ff1152
# Speed Grade:   -5
# Processor: PPC 405
# Processor clock frequency: 32.000000 MHz
# Bus clock frequency: 32.000000 MHz
# Debug interface: No Debug
# On Chip Memory :  16 KB
##############################################################################
 PARAMETER VERSION = 2.1.0
 PORT fpga_0_LEDS_GPIO_d_out_pin = fpga_0_LEDS_GPIO_d_out, VEC = [0:31], DIR
= OUTPUT
 PORT fpga_0_DIP_Switches_GPIO_in_pin = fpga_0_DIP_Switches_GPIO_in, VEC =
[0:31], DIR = INPUT
 PORT sys_clk_pin = dcm_clk_s, DIR = INPUT, SIGIS = CLK
 PORT sys_rst_pin = sys_rst_s, DIR = INPUT
 PORT iitdisplay_0_dac_clk = iitdisplay_0_dac_clk, DIR = O
 PORT iitdisplay_0_sync_dac = iitdisplay_0_sync_dac, DIR = O
 PORT iitdisplay_0_blank_dac = iitdisplay_0_blank_dac, DIR = O
 PORT iitdisplay_0_mode_1 = iitdisplay_0_mode_1, DIR = O
 PORT iitdisplay_0_mode_2 = iitdisplay_0_mode_2, DIR = O
 PORT iitdisplay_0_sync_t = iitdisplay_0_sync_t, DIR = O
 PORT iitdisplay_0_dac_data = iitdisplay_0_dac_data, VEC = [9:0], DIR = O
```

```
BEGIN proc_sys_reset
 PARAMETER INSTANCE = reset_block
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT Ext_Reset_In = sys_rst_s
 PORT Slowest_sync_clk = sys_clk_s
 PORT Chip_Reset_Req = C405RSTCHIPRESETREQ
 PORT Core_Reset_Req = C405RSTCORERESETREQ
 PORT System_Reset_Req = C405RSTSYSRESETREQ
 PORT Rstc405resetchip = RSTC405RESETCHIP
 PORT Rstc405resetcore = RSTC405RESETCORE
 PORT Rstc405resetsys = RSTC405RESETSYS
 PORT Bus_Struct_Reset = sys_bus_reset
 PORT Dcm_locked = dcm_0_lock
END

BEGIN ppc405
 PARAMETER INSTANCE = ppc405_0
 PARAMETER HW_VER = 2.00.c
 BUS_INTERFACE IPLB = plb
 BUS_INTERFACE DPLB = plb
 PORT PLBCLK = sys_clk_s
 PORT C405RSTCHIPRESETREQ = C405RSTCHIPRESETREQ
 PORT C405RSTCORERESETREQ = C405RSTCORERESETREQ
 PORT C405RSTSYSRESETREQ = C405RSTSYSRESETREQ
 PORT RSTC405RESETCHIP = RSTC405RESETCHIP
 PORT RSTC405RESETCORE = RSTC405RESETCORE
 PORT RSTC405RESETSYS = RSTC405RESETSYS
 PORT CPMC405CLOCK = sys_clk_s
END

BEGIN bram_block
PARAMETER INSTANCE = plb_bram_if_cntlr_1_bram
PARAMETER HW_VER = 1.00.a
BUS_INTERFACE PORTA = plb_bram_if_cntlr_1_port
END

BEGIN plb_bram_if_cntlr
 PARAMETER INSTANCE = plb_bram_if_cntlr_1
 PARAMETER HW_VER = 1.00.b
 PARAMETER c_plb_clk_period_ps = 31250
 PARAMETER c_baseaddr = 0xffffc000
 PARAMETER c_highaddr = 0xffffffff
 BUS_INTERFACE SPLB = plb
 BUS_INTERFACE PORTA = plb_bram_if_cntlr_1_port
 PORT PLB_Clk = sys_clk_s
END

BEGIN plb2opb_bridge
 PARAMETER INSTANCE = plb2opb
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_DCR_INTFCE = 0
 PARAMETER C_NUM_ADDR_RNG = 1
 PARAMETER C_RNG0_BASEADDR = 0x7fffe000
 PARAMETER C_RNG0_HIGHADDR = 0x7fffefff
 BUS_INTERFACE SPLB = plb
 BUS_INTERFACE MOPB = opb
 PORT PLB_Clk = sys_clk_s
 PORT OPB_Clk = sys_clk_s
END

BEGIN plb_v34
 PARAMETER INSTANCE = plb
 PARAMETER HW_VER = 1.02.a
 PARAMETER C_DCR_INTFCE = 0
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT SYS_Rst = sys_bus_reset
 PORT PLB_Clk = sys_clk_s
END

BEGIN opb_v20
 PARAMETER INSTANCE = opb
 PARAMETER HW_VER = 1.10.b
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT SYS_Rst = sys_bus_reset
 PORT OPB_Clk = sys_clk_s
END

BEGIN iitdisplay
 PARAMETER INSTANCE = iitdisplay_0
 PARAMETER HW_VER = 3.03.a
 PARAMETER C_BASEADDR = 0x7fffe400
 PARAMETER C_HIGHADDR = 0x7fffe7ff
 BUS_INTERFACE SOPB = opb
 PORT dac_clk = iitdisplay_0_dac_clk
 PORT sync_dac = iitdisplay_0_sync_dac
 PORT blank_dac = iitdisplay_0_blank_dac
 PORT mode_1 = iitdisplay_0_mode_1
 PORT mode_2 = iitdisplay_0_mode_2
 PORT sync_t = iitdisplay_0_sync_t
 PORT dac_data = iitdisplay_0_dac_data
 PORT OPB_Clk = sys_clk_s
END

BEGIN dcm_module
 PARAMETER INSTANCE = dcm_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_CLK0_BUF = TRUE
 PARAMETER C_CLKIN_PERIOD = 31.250000
 PARAMETER C_CLK_FEEDBACK = 1X
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT CLKIN = dcm_clk_s
 PORT CLK0 = sys_clk_s
 PORT CLKFB = sys_clk_s
 PORT RST = net_gnd
 PORT LOCKED = dcm_0_lock
END

BEGIN opb_gpio
 PARAMETER INSTANCE = LEDS
 PARAMETER HW_VER = 3.01.a
 PARAMETER C_GPIO_WIDTH = 32
 PARAMETER C_IS_DUAL = 0
 PARAMETER C_ALL_INPUTS = 0
 PARAMETER C_IS_BIDIR = 0
 PARAMETER C_BASEADDR = 0x7fffe200
 PARAMETER C_HIGHADDR = 0x7fffe3ff
 BUS_INTERFACE SOPB = opb
 PORT OPB_Clk = sys_clk_s
 PORT GPIO_d_out = fpga_0_LEDS_GPIO_d_out
END

BEGIN opb_gpio
 PARAMETER INSTANCE = DIP_Switches
 PARAMETER HW_VER = 3.01.a
 PARAMETER C_GPIO_WIDTH = 32
 PARAMETER C_IS_DUAL = 0
 PARAMETER C_ALL_INPUTS = 1
 PARAMETER C_IS_BIDIR = 0
 PARAMETER C_BASEADDR = 0x7fffe000
 PARAMETER C_HIGHADDR = 0x7fffe1ff
 BUS_INTERFACE SOPB = opb
 PORT OPB_Clk = sys_clk_s
 PORT GPIO_in = fpga_0_DIP_Switches_GPIO_in
END
```

The Platform Generator tool (PlatGen) creates the hardware platform using the MHS file as input. PlatGen creates netlist files in various formats (NGC, EDIF), as well as support files for downstream tools, and top-level HDL wrappers to allow users to add other components to the automatically generated hardware platform.

The file created by the Platgen is given below:

```
Release 6.3i - platgen EDK_Gmm.10
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.

Command Line: platgen -p xc2vp30ff1152-5 -lang vhdl -lp E:/RAHUL/ -st xst
system.mhs
Parse system.mhs ...
Read MPD definitions ...
WARNING:MDT - Search path E:\RAHUL\ directly contains pcores directory. Search
   path should point to a directory two levels above pcores.Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/ppc405_v2_00_c/data/ppc405_v2_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb_bram_if_cntlr_v1_00_b/data/plb_bram_if
_cntlr_v2_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb2opb_bridge_v1_01_a/data/plb2opb_bridge
_v2_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb_v34_v1_02_a/data/plb_v34_v2_1_0.tcl
...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/opb_v20_v1_10_b/data/opb_v20_v2_1_0.tcl
...Done
Overriding IP level properties ...
bram_block (plb_bram_if_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.mpd:3
8 - overriding c_family value virtex2 to virtex2p dcm_module (dcm_0) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\dcm_module_v1_00_a\data\dcm_module_v2_1_0.mpd
:55 - overriding c_family value virtex2 to virtex2p opb_gpio (leds) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_gpio_v3_01_a\data\opb_gpio_v2_1_0.mpd:37    -
overriding c_family value virtex2 to virtex2p opb_gpio (dip_switches) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_gpio_v3_01_a\data\opb_gpio_v2_1_0.mpd:37    -
overriding c_family value virtex2 to virtex2p

Performing IP level DRCs on properties...
```

```
Running DRC Tcl procedures for OPTION IPLEVEL_DRC_PROC...
Address Map for Processor ppc405_0
  (0x7fffe000-0x7fffe1ff) DIP_Switches      plb->plb2opb->opb
  (0x7fffe200-0x7fffe3ff) LEDS      plb->plb2opb->opb
  (0x7fffe400-0x7fffe7ff) iitdisplay_0      plb->plb2opb->opb
  (0xffffc000-0xffffffff) plb_bram_if_cntlr_1plb

Check platform configuration ...
plb_v34 (plb) - E:\myiitr\system.mhs:94 - 2 master(s) : 2 slave(s)
opb_v20 (opb) - E:\myiitr\system.mhs:103 - 1 master(s) : 3 slave(s)

Check port drivers...
Check platform address map ...
Overriding system level properties ...
bram_block (plb_bram_if_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.mpd:3
4 - overriding c_memsize value 2048 to 16384 bram_block (plb_bram_if_cntlr _1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.mpd:3
5 - overriding c_port_dwidth value 32 to 64 bram_block (plb_bram_if_cntlr _1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.mpd:3
7 - overriding c_num_we value 4 to 8 plb_bram_if_cntlr (plb_bram_if_cntlr_1) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_bram_if_cntlr_v1_00_b\data\plb_bram_if
_cntlr_v2_1_0.mpd:39  -  overriding  c_num_masters  value  8  to  2  plb_bram_if_cntlr
(plb_bram_if_cntlr_1) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_bram_if_cntlr_v1_00_b\data\plb_bram_if
_cntlr_v2_1_0.mpd:46 - overriding c_plb_mid_width value 3 to 1 plb2opb_bridge (plb2opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb2opb_bridge_v1_01_a\data\plb2opb_bridge
_v2_1_0.mpd:47 - overriding c_plb_num_masters value 4 to 2 plb2opb_bridge (plb2opb)
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb2opb_bridge_v1_01_a\data\plb2opb_bridge
_v2_1_0.mpd:48 - overriding c_plb_mid_width value 4 to 1 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:38      -
overriding c_plb_num_masters value 4 to 2 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:39      -
overriding c_plb_num_slaves value 4 to 2 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:40      -
overriding c_plb_mid_width value 2 to 1 opb_v20 (opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_v20_v1_10_b\data\opb_v20_v2_1_0.mpd:39      -
overriding c_num_masters value 4 to 1 opb_v20 (opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_v20_v1_10_b\data\opb_v20_v2_1_0.mpd:40      -
overriding c_num_slaves value 4 to 3

Running DRC Tcl procedures for OPTION SYSLEVEL_DRC_PROC...

Performing System level DRCs on properties...
```

```
Runnung UPDATE Tcl procedures for OPTION PLATGEN_SYSLEVEL_UPDATE_
PROC
Modify defaults ...
Performing XLPP processing on licensed instances ...
Completion time: 0.00 seconds
Creating hardware output directories ...
Managing hardware (BBD-specified) netlist files ...
Managing cache ...
Elaborating instances ...
bram_block (plb_bram_if_cntlr_1_bram) - E:\myiitr\system.mhs:64 - elaborating IP
Writing HDL for elaborated instances ...
Inserting wrapper level ...
Completion time: 6.00 seconds
Constructing platform-level signal connectivity ...
Completion time: 15.00 seconds
Writing (top-level) BMM ...
Writing BMM - E:\myiitr\implementation\system.bmm
Writing (top-level and wrappers) HDL ...
Generating synthesis project file ...

Running XST synthesis ...
INFO:MDT - The following instances are synthesized with XST. The MPD option
  IMP_NETLIST=TRUE indicates that a NGC file is to be produced using XST
  synthesis. IMP_NETLIST=FALSE (default) instances are not synthesized. A batch
  file, synthesis.sh, has been created that allows you to synthesize those
  instances in your specified synthesis tool of choice.
reset_block_wrapper (reset_block) - E:\myiitr\system.mhs:33 - Running XST
synthesis ppc405_0_wrapper (ppc405_0) –
E:\myiitr\system.mhs:49 - Running XST synthesis plb_bram_if_cntlr_1_bram_wrapper
(plb_bram_if_cntlr_1_bram) -
E:\myiitr\system.mhs:64 - Running XST synthesis plb_bram_if_cntlr_1_wrapper
(plb_bram_if_cntlr_1) - E:\myiitr\system.mhs:70 -
Running XST synthesis
plb2opb_wrapper (plb2opb) - E:\myiitr\system.mhs:81 - Running XST synthesis
plb_wrapper (plb) - E:\myiitr\system.mhs:94 - Running XST synthesis
opb_wrapper (opb) - E:\myiitr\system.mhs:103 - Running XST synthesis
iitdisplay_0_wrapper (iitdisplay_0) - E:\myiitr\system.mhs:111 - Running XST
synthesis dcm_0_wrapper (dcm_0) - E:\myiitr\system.mhs:127 - Running XST synthesis
leds_wrapper (leds) - E:\myiitr\system.mhs:141 - Running XST synthesis
dip_switches_wrapper (dip_switches) - E:\myiitr\system.mhs:155 - Running XST
synthesis
Running NGCBUILD ...

Rebuilding cache ...
Total run time: 332.00 seconds
```

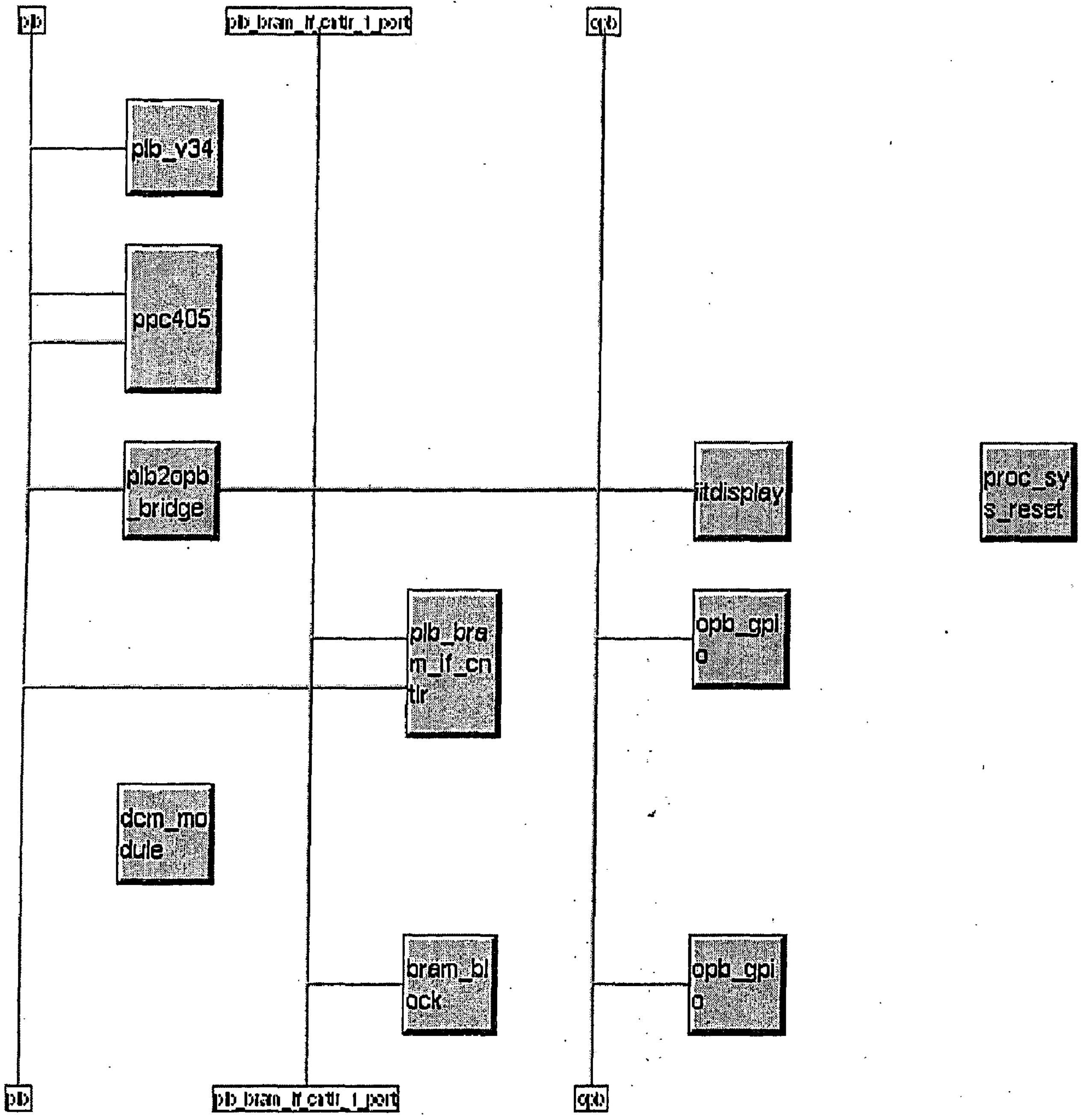


Figure 5-5 Microprocessor Peripheral Description

Microprocessor Peripheral Description gives the block diagram of the various components that are connected in the design. It is automatically generated and from here we can change the parameters of the various components which show the complete flexibility offered by the tool while designing the embedded tool.

In addition to the processor ,buses and user blocks there are addition blocks called BRAM blocks. One Bram block is attached to the PLB and the other to the OPB. Two controllers control access from the OPB to its BlockRAM ('opb_bram_if_cntlr') and PLB to its BlockRAM ('plb_bram_if_cntlr'). This controller acts as an interface between the bus and the BlockRAM.These are shown in the PBD diagram shown below in Figure 5-5.

Microprocessor Software Specification (MSS) File:

The MSS file defines driver and library customization parameters for peripherals, processor customization parameters, standard input/output devices, interrupt handler routines, and other related software features. The MSS File generated for the design is given below.

```
PARAMETER VERSION = 2.2.0


BEGIN OS
 PARAMETER OS_NAME = standalone
 PARAMETER OS_VER = 1.00.a
 PARAMETER PROC_INSTANCE = ppc405_0
END
BEGIN PROCESSOR
 PARAMETER DRIVER_NAME = cpu_ppc405
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = ppc405_0
 PARAMETER COMPILER = powerpc-eabi-gcc
 PARAMETER ARCHIVER = powerpc-eabi-ar
 PARAMETER CORE_CLOCK_FREQ_HZ = 32000000
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = plbarb
 PARAMETER DRIVER_VER = 1.01.a
 PARAMETER HW_INSTANCE = plb
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = opb
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = plb2opb
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = plb2opb
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = gpio
 PARAMETER DRIVER_VER = 2.00.a
 PARAMETER HW_INSTANCE = DIP_Switches
END
```

```
BEGIN DRIVER
 PARAMETER DRIVER_NAME = gpio
 PARAMETER DRIVER_VER = 2.00.a
 PARAMETER HW_INSTANCE = LEDS
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = bram
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = plb_bram_if_cntlr_1
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = generic
 PARAMETER DRIVER_VER = 1.00.a
 PARAMETER HW_INSTANCE = dcm_0
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = iitdisplay
 PARAMETER DRIVER_VER = 3.03.a
 PARAMETER HW_INSTANCE = iitdisplay_0
END
```

The MSS file is an input to the Library Generator tool (LibGen) for customization of drivers, libraries and interrupt handlers. This process is used for creating the software platform. The Library tool (LibGen) creates the following report file.

```
Release 6.3i - libgen Xilinx EDK 6.3
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.

Command Line: libgen -mhs system.mhs -p xc2vp30ff1152-5 -lp E:/RAHUL/system
.mss
Output Directory (-od)          : E:\myiitr\
Part (-p)                       : virtex2p
Software Specification file     : system.mss
WARNING:MDT - Search path E:\RAHUL\ directly contains pcores directory. Search  path
should point to a directory two levels above pcores.Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/ppc405_v2_00_c/data/ppc405_v2_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb_bram_if_cntlr_v1_00_b/data/plb_bram_if_cnt
lr_v2_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb2opb_bridge_v1_01_a/data/plb2opb_bridge_v2
_1_0.tcl ...Done
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/plb_v34_v1_02_a/data/plb_v34_v2_1_0.tcl...Done
```

```
Sourcing tcl file
F:/EDK/hw/XilinxProcessorIPLib/pcores/opb_v20_v1_10_b/data/opb_v20_v2_1_0.tcl...Do
ne
Overriding IP level properties ...
bram_block (plb_bram_if_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.m
pd:38 - overriding c_family value virtex2 to virtex2p dcm_module (dcm_0) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\dcm_module_v1_00_a\data\dcm_module_v2_1_0.
mpd:55 - overriding c_family value virtex2 to virtex2p opb_gpio (leds) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_gpio_v3_01_a\data\opb_gpio_v2_1_0.mpd:3
7 - overriding c_family value virtex2 to virtex2p opb_gpio (dip_switches) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_gpio_v3_01_a\data\opb_gpio_v2_1_0.mpd:3
7 - overriding c_family value virtex2 to virtex2p

Performing IP level DRCs on properties...
Running DRC Tcl procedures for OPTION IPLEVEL_DRC_PROC...
Address Map for Processor ppc405_0
  (0x7fffe000-0x7fffe1ff) DIP_Switches plb->plb2opb->opb
  (0x7fffe200-0x7fffe3ff) LEDS  plb->plb2opb->opb
  (0x7fffe400-0x7fffe7ff) iitdisplay_0    plb->plb2opb->opb
  (0xffffc000-0xffffffff) plb_bram_if_cntlr_1    plb

Check platform configuration ...
plb_v34 (plb) - E:\myiitr\system.mhs:94 - 2 master(s) : 2 slave(s)
opb_v20 (opb) - E:\myiitr\system.mhs:103 - 1 master(s) : 3 slave(s)

Check port drivers...
Check platform address map ...
Overriding system level properties ...
bram_block (plb_bram_if_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.m
pd:34 - overriding c_memsize value 2048 to 16384 bram_block (plb_bram_if
_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.m
pd:35 - overriding c_port_dwidth value 32 to 64 bram_block (plb_bram_if_cntlr_1_bram) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\bram_block_v1_00_a\data\bram_block_v2_1_0.m
pd:37 - overriding c_num_we value 4 to 8 plb_bram_if_cntlr (plb_bram_if_cntlr_1) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_bram_if_cntlr_v1_00_b\data\plb_bram_if
_cntlr_v2_1_0.mpd:39 - overriding c_num_masters value 8 to 2 plb_bram_if_cntlr
(plb_bram_if_cntlr_1) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_bram_if_cntlr_v1_00_b\data\plb_bram_if
_cntlr_v2_1_0.mpd:46 - overriding c_plb_mid_width value 3 to 1 plb2opb_
bridge (plb2opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb2opb_bridge_v1_01_a\data\plb2opb_bridge
_v2_1_0.mpd:47 - overriding c_plb_num_masters value 4 to 2 plb2opb_bridge
(plb2opb) -
```

```
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb2opb_bridge_v1_01_a\data\plb2opb_bridge
_v2_1_0.mpd:48 - overriding c_plb_mid_width value 4 to 1 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:38 -
overriding c_plb_num_masters value 4 to 2 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:39 -
overriding c_plb_num_slaves value 4 to 2 plb_v34 (plb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\plb_v34_v1_02_a\data\plb_v34_v2_1_0.mpd:40 -
overriding c_plb_mid_width value 2 to 1 opb_v20 (opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_v20_v1_10_b\data\opb_v20_v2_1_0.mpd:39
- overriding c_num_masters value 4 to 1 opb_v20 (opb) -
F:\EDK\hw\XilinxProcessorIPLib\pcores\opb_v20_v1_10_b\data\opb_v20_v2_1_0.mpd:40
- overriding c_num_slaves value 4 to 3

Running DRC Tcl procedures for OPTION SYSLEVEL_DRC_PROC...

Performing System level DRCs on properties...
WARNING:MDT - Search path E:\RAHUL\ directly contains pcores directory. Search
path should point to a directory two levels above pcores.INFO:MDT - List of peripherals
addressable from processor instance ppc405_0 :
  - plb_bram_if_cntlr_1
  - iitdisplay_0
  - LEDS
  - DIP_Switches

Building Directory Structure for ppc405_0
Generating platform libraries and device drivers ...
Running CopyFiles ...

Copying files for os standalone_v1_00_a from
F:\EDK\sw\lib\bsp\standalone_v1_00_a\src\ to
E:\myiitr\ppc405_0\libsrc\standalone_v1_00_a\ ...

Copying files for driver iitdisplay_v3_03_a from
E:\myiitr\drivers\iitdisplay_v3_03_a\src\ to
E:\myiitr\ppc405_0\libsrc\iitdisplay_v3_03_a\ ...

Copying files for driver gpio_v2_00_a from
F:\EDK\sw\XilinxProcessorIPLib\drivers\gpio_v2_00_a\src\ to
E:\myiitr\ppc405_0\libsrc\gpio_v2_00_a\ ...

Copying files for driver cpu_ppc405_v1_00_a from
F:\EDK\sw\XilinxProcessorIPLib\drivers\cpu_ppc405_v1_00_a\src\ to
E:\myiitr\ppc405_0\libsrc\cpu_ppc405_v1_00_a\ ...

Running DRCs for OSes, Drivers and Libraries ...
Running generate for OS'es, Drivers and Libraries ...
Running post_generate for OS'es, Drivers and Libraries ...
Running make for Drivers and Libraries ...
Configuring make for target include using:
make -s include "COMPILER=powerpc-eabi-gcc" "ARCHIVER=powerpc-eabi-ar"
"COMPILER_FLAGS= -O2 -c" "EXTRA_COMPILER_FLAGS=-g"
Configuring make for target libs using:
make -s libs "COMPILER=powerpc-eabi-gcc" "ARCHIVER=powerpc-eabi-ar"
"COMPILER_FLAGS= -O2 -c" "EXTRA_COMPILER_FLAGS=-g"
Libraries generated in E:\myiitr\ppc405_0\lib\ directory
Running execs_generate for OS'es, Drivers and Libraries ...
LibGen Done.
```

User Constrained File (UCF) File:
This file is used to assign the pin numbers to connect the FPGA to the outer world.

```
#########################################################################
## This system.ucf file is generated by Base System Builder based on the
## settings in the selected Xilinx Board Definition file. Please add other
## user constraints to this file based on customer design specifications.
#########################################################################

 Net sys_clk_pin LOC="D18";
 Net sys_rst_pin LOC="AD3";
## System level constraints
Net sys_clk_pin PERIOD = 31250 ps;
Net sys_rst_pin TIG;
Net "RSTC405RESETCORE" TPTHRU = "RST_GRP";
NET "RSTC405RESETCHIP" TPTHRU = "RST_GRP";
NET "RSTC405RESETSYS" TPTHRU = "RST_GRP";
NET "C405RSTCORERESETREQ" TPTHRU = "RST_GRP";
NET "C405RSTCHIPRESETREQ" TPTHRU = "RST_GRP";
NET "C405RSTSYSRESETREQ" TPTHRU = "RST_GRP";
TIMESPEC "TS_RST1" = FROM CPUS THRU RST_GRP TO FFS  TIG;
TIMESPEC "TS_RST2" = FROM FFS  THRU RST_GRP TO FFS  TIG;
TIMESPEC "TS_RST3" = FROM FFS  THRU RST_GRP TO CPUS TIG;

Net iitdisplay_0_dac_data<0> LOC="AE30";
Net iitdisplay_0_dac_data<1> LOC="AE27";
Net iitdisplay_0_dac_data<2> LOC="AE28";
Net iitdisplay_0_dac_data<3> LOC="AD29";
Net iitdisplay_0_dac_data<4> LOC="AD30";
Net iitdisplay_0_dac_data<5> LOC="AD27";
Net iitdisplay_0_dac_data<6> LOC="AB32";
Net iitdisplay_0_dac_data<7> LOC="AB31";
Net iitdisplay_0_dac_data<8> LOC="AC32";
Net iitdisplay_0_dac_data<9> LOC="AC31";
Net iitdisplay_0_sync_t LOC="AD31";
Net iitdisplay_0_mode_2 LOC="AC29";
Net iitdisplay_0_mode_1 LOC="AD26";
Net iitdisplay_0_blank_dac LOC="AA32";
Net iitdisplay_0_sync_dac LOC="AA31";
Net iitdisplay_0_dac_clk LOC="AD32";
```

Implementation Details:

Here I am giving the implementation details of the design.Here I am only incorporating only some of the details about the device utilization.

The section below gives the details of the wrappers used for different components in the design.

```
#----------------------------------------------#
# Starting program ngdbuild
# ngdbuild -p xc2vp30ff1152-5 -nt timestamp -bm system.bmm
E:/myiitr/implementation/system.ngc -uc system.ucf system.ngd
#----------------------------------------------#
Release 6.3i - ngdbuild G.35
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.
PM_SPEC -- Xilinx path component is <F:/EDK>

Command Line: ngdbuild -p xc2vp30ff1152-5 -nt timestamp -bm system.bmm -uc
system.ucf E:/myiitr/implementation/system.ngc system.ngd

Reading NGO file "E:/myiitr/implementation/system.ngc" ...
Reading component libraries for design expansion...
Loading design module "E:/myiitr/implementation/plb_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/opb_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/iitdisplay_0_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/dcm_0_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/leds_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/dip_switches_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/reset_block_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/ppc405_0_wrapper.ngc"...
Loading design module
"E:/myiitr/implementation/plb_bram_if_cntlr_1_bram_wrapper.ngc"...
Loading design module
"E:/myiitr/implementation/plb_bram_if_cntlr_1_wrapper.ngc"...
Loading design module "E:/myiitr/implementation/plb2opb_wrapper.ngc"...
```

This section gives the details of the logic utilized in the FPGA's.

```
#-------------------------------------------#
# Starting program map
# map -o system_map.ncd -pr b system.ngd system.pcf
#-------------------------------------------#
Release 6.3i - Map G.35
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.
PM_SPEC -- Xilinx path component is <F:/EDK>
Using target part "2vp30ff1152-5".
Removing unused or disabled logic...
Running cover...
Writing file system_map.ngm...
Running directed packing...
Running delay-based LUT packing...
Running related packing...
Writing design file "system_map.ncd"...

Design Summary:
Number of errors:     0
Number of warnings: 107
Logic Utilization:
  Number of Slice Flip Flops:       1,231 out of  27,392    4%
  Number of 4 input LUTs:           1,137 out of  27,392    4%
Logic Distribution:
  Number of occupied Slices:        1,284 out of  13,696    9%
  Number of Slices containing only related logic:  1,284 out of   1,284  100%
  Number of Slices containing unrelated logic:         0 out of   1,284    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:          1,474 out of  27,392    5%
  Number used as logic:           1,137
  Number used as a route-thru:       94
  Number used for Dual Port RAMs:   210
    (Two LUTs used per Dual Port RAM)
  Number used as Shift registers:    33

  Number of bonded IOBs:              82 out of     644   12%
    IOB Flip Flops:                 67
  Number of PPC405s:                  1 out of       2   50%
  Number of Block RAMs:               8 out of     136    5%
  Number of GCLKs:                    2 out of      16   12%
  Number of DCMs:                     1 out of       8   12%
  Number of GTs:                      0 out of       8    0%
  Number of GT10s:                    0 out of       0    0%

Total equivalent gate count for design:  578,947
Additional JTAG gate count for IOBs:  3,936
Peak Memory Usage:  149 MB
```

NOTES:

Related logic is defined as being logic that shares connectivity - e.g. two LUTs are "related" if they share common inputs. When assembling slices, Map gives priority to combine logic that is related. Doing so results in the best timing performance.

Unrelated logic shares no connectivity. Map will only begin packing unrelated logic into a slice once 99% of the slices are occupied through related logic packing.

Note that once logic distribution reaches the 99% level through related logic packing, this does not mean the device is completely utilized. Unrelated logic packing will then begin, continuing until all usable LUTs and FFs are occupied. Depending on your timing budget, increased levels of unrelated logic packing may adversely affect the overall timing performance of your design.

Mapping completed.

This section gives the details of how much of the available devices are utilized.

Device utilization summary:

```
  Number of External IOBs          82 out of 644    12%
    Number of LOCed External IOBs   18 out of 82     21%

  Number of PPC405s                 1 out of 2      50%
  Number of RAMB16s                 8 out of 136     5%
  Number of SLICEs               1284 out of 13696   9%

  Number of BUFGMUXs                2 out of 16     12%
  Number of DCMs                    1 out of 8      12%
All signals are completely routed.
```

```
**************************
   Generating Clock Report
**************************

+------------------------+--------------------+--------+-------+---------------+---------------+
|      Clock Net         | Resource           |Locked|Fanout|Net Skew(ns)|Max Delay(ns)|
+------------------------+--------------------+--------+-------+---------------+---------------+
|plb_bram_if_cntlr       |                    |        |       |               |               |
|_1_port_BRAM_Clk | BUFGMUX2S          |  No    |  912  |  0.732        |  1.954        |
+----------------------+-------------------+--------+------+--------------+--------------+
|iitdisplay_0_dac_     |                   |        |      |              |              |
|     Clk_OBUF         | Local             |        | 36   | 0.716        |    3.046     |
+----------------------+-------------------+--------+------+--------------+--------------+
```

Timing Score: 0

Asterisk (*) preceding a constraint indicates it was not met.
  This may be due to a setup or hold violation.

```
-------------------------------------------------------------------------------------
     Constraint                                     | Requested  | Actual   | Logic
                                                    |            |          | Levels
-------------------------------------------------------------------------------------
  NET "bufgp_64/IBUFG" PERIOD = 31.250 nS           | N/A        | N/A      | N/A
  HIGH 50.000000 %                                  |            |          |
-------------------------------------------------------------------------------------
 PERIOD analysis for net "dcm_0/dcm_0/CLK0          | 31.250ns   | 15.182ns | 5
 _BUF" derived from NET "bufgp_64/IBUFG"            |            |          |
  PERIOD = 31.250 nS   HIGH 50.000000 %             |            |          |
-------------------------------------------------------------------------------------
PATH "FROM CPUS THRU RST_GRP TO FFS" TIG            | N/A        | 4.715ns  | 0
-------------------------------------------------------------------------------------
PATH "FROM FFS THRU RST_GRP TO FFS" TIG             | N/A        | 4.656ns  | 0
-------------------------------------------------------------------------------------
PATH "FROM FFS THRU RST_GRP TO CPUS" TIG            | N/A        | 2.844ns  | 0
-------------------------------------------------------------------------------------
```

All constraints were met.
INFO: Timing: 2761 - N/A entries in the Constraints list may indicate that the
  constraint does not cover any paths or that it has no requested value.
Total REAL time to PAR completion: 1 mins 48 secs
Total CPU time to PAR completion: 1 mins 31 secs

Peak Memory Usage: 205 MB

Placement: Completed - No errors found.
Routing: Completed - No errors found.
Timing: Completed - No errors found.
Writing design to file system.ncd.
PAR done.