Unit / Module Description:	SMT 339
Document Issue Number:	1
Issue Date:	25/09/07
Original Author:	Francois Godreau

CODESIGN, IMPLEMENTATION AND VALIDATION OF THE XILINX FPGA'S EMBEDDED DEVICES FOR SIGNAL PROCESSING SOLUTIONS

Sundance Multiprocessor Technology Ltd, Chiltern House, Waterside, Chesham, Bucks. HP5 1PS.

This document is the property of Sundance and may not be copied nor communicated to a third party without prior written permission.
© Sundance Multiprocessor Technology Limited 2006

Certificate Number FM 55022

Application note

Abstract

The design team develops rapid prototyping platforms for high-performance processing. These boards are independent modules populated by DSPs and FPGAs that can be inter-connected via communication links. Sundance boards are typically based on TI DSP and Xilinx FPGAs which embed processors like the PowerPC or Microblaze.
Sundance provides basic software support for their modules. To complement this software, Sundance works in partnership with software companies. These companies provide high-level design tools and methodologies that accelerate the design of applications based on a mixed DSP and FPGA systems. Although these companies provide solutions for the DSPs and FPGAs in the Sundance systems, no support for the PowerPC has yet been developed.

The Sundance SMT339 provides a TI DSP DM642 and a Xilinx FPGA-FX60 with two embedded PowerPCs which formed an ideal development platform for this application note. In addition to the SMT339, a SMT319 module was used to communicate with this design in the PowerPC. The SMT319 was connected to a host system on which debug information could be printed out. XILINX tools were used such as EDK to implement the embedded processor and XILINX ISE software to design the application.
Sundance FPGAs implements several types of communication resources that are used to communicate between modules. These are comport, SHB, SDB, RSL and SLB.

Keywords: FPGA; PowerPC, Microblaze, firmware, co-design, embedded processors

Table of contents

Table of figures

TABLE

INTRODUCTION

Sundance Multiprocessor Technology Limited is a British firm located in Chesham, to the North-west of London. As shows by its name, this company designs and produces boards, with processors on them, dedicated for high frequency acquisition and data processing application. These boards, built with DSPs and FPGAs, are designed to the Texas Instrument Module standard.
To communicate and synchronise the parallel processing on different targets, DSPs and FPGAs, the designers need methods and tools to fill these requirements. Sundance works in partnership with companies providing high level design tools to ease development of applications for the Sundance's systems. **3L** is a company that offers DIAMOND, a design tool which is optimised for the Sundance products.

Thanks to new technology, FPGAs now provide a hardware/software development target within a single programmable device. An FPGA embedded processor system offers many exceptional advantages compared to typical microprocessors. Because of this, Sundance's customers want to develop applications implementing embedded processors. As yet, Sundance has not provided solutions to implement an embedded processor such as PowerPC and Microblaze into its boards. It needs to develop reference designs providing solutions for its customers and application notes describing every design.

This design was split into three modular designs to make the implementation easier. The first design contained a PowerPC with Sundance communication resources. This design allowed an embedded processor to transfer data to another processor such as a DSP. The second design implemented a PowerPC with an external ZBT memory controller. This design provided 8MB external memory to the processor. The memory controller was connected onto both interconnect buses: On-Chip Peripheral Bus (OPB) and Processor Local Bus (PLB). The last design incorporated every element: external memory, two communication resources, GPIO LED, data On-Chip Memory-Bus, instruction On-Chip Memory-Bus, data-cache side, instruction-cache side and internal memory. Application notes are available to give some information about each phase of the project. Each design is based on the model 3L DIAMOND uses to describe a system (3L Diamond).

1 Project description

This chapter contains the aim of this design and its specification.

1.1 The main objective

Advances in FPGA technology allow manufacturers to insert more and more peripherals increasing the functionality of this chip into their FPGAs. In the latest FPGAs, you can find tri-mode Ethernet Media Acess Controller that operates at 10, 100 and 1000 Mb/s, 100 Mb/s to 3,2Gb/s serial transceivers, In Tegrated Endpoint Block for PCI Express Compliance delivering full PCI Express Endpoint functionality, high-speed clock management circuitry, dedicated DSP slices. Also, a wide array of FPGA hard-IP core blocks include embedded processors.

FPGA manufacturers like Xilinx spent a lot of money to promote their embedded systems. For example, Xilinx has been developing a tool to simply implementing Xilinx's IP cores and embedded processors. Thus, a lot of people are very interested by these embedded processors because they can develop a design with both hardware and software application within the same chip.

Sundance's customers are very interested to use the PowerPC. To answer customer demand, Sundance have to support these embedded processors which formed the basis of this application note.

1.2 Requirement

This project was split in several applications to make the implementation easier. But the whole of the following specifications should be enforced for every application.
All specifications are contained in the **Table 1**.

1.2.1 Processor choice

Sundance works in partnership with Xilinx foundation. Xilinx provides FPGA families that embed two kinds of processors: the Microblaze and the PowerPC. The Microblaze is a soft processor core optimized for Xilinx's FPGA, by contrast, the PowerPC, created by IBM, is a physical processor built into the FPGA silicon, which delivers several advantages. You can find more details about these processors in section that follow.
PowerPC is the architecture of choice in the embedded world. This kind of processor was used on game console such as Nintendo's GameCube and Wii, Sony's PlayStation 3 and Microsoft's Xbox 360 and in the personal computer market like Apple's Macintosh. The PowerPC architecture developed is very famous and the knowledge about this processor is quite important.
Contrary to the Microblaze, the PowerPC architecture is independent of the target and it is built from dedicated silicon. Its implementation into a design does not change the performances of the device.
Although the PowerPC provides features better than the Microblaze, only certain Xilinx FPGA families embed a PowerPC. Despite this, this application note implements a PowerPC because Sundance boards provide both co-design solutions: FPGA-DSP or FPGA-PowerPC. Thus, if the FPGA does not embed a PowerPC, the software application may be developed on the DSP. An application designed for a PowerPC can easily be moved to the Microblaze processor. In fact, these both processors provide the same communication resources. Similarly, IP blocs can be shared between the processor and the same programming language is used.

1.2.2 Communication resources

To allow the PowerPC to communicate with another processor such as a DSP, the host or FPGA, it has to implement the Sundance's communication resources. Typically, Sundance FPGAs implement four types of communication resources. These resources are implemented in firmware in the FPGA. The aim of this application note was not to implement all communication resources in the PowerPC. By using 3L DIAMOND the required communication resources are implemented automatically.
The following are types of communication resources:

- comport
- Sundance digital bus
- global bus
- rocket IO serial link

The Comport link was chosen among other links to connect the PowerPC to another processor for many reasons.
Firstly, the Comport is the simplest of the Sundance communication resources. Its implementation on an FPGA would be easier and quicker than rocket IO links for example.
A basic application was developed allowing a PowerPC to transfer data to another processor. This would be easier to validate and improve it.
The Comport is usually used to download programs to processor TIMs. This method is used by DIAMOND to load boot code into processor TIM. Thus, it was important to implement this link to simplify the development of the DIAMOND PPC software.
The implementation of communication resources follow design guidelines set out by Sundance.

1.2.3 External memory

On Sundance boards, some FPGAs are connected with an external memory. Fast memory is connected to FPGA allowing high speed data storage. Many kinds of memory may be connected with FPGA such as SDRAM, DDR, Flash. But a type of memory used on Sundance board provides interesting features.
Zero-bus turnaround (ZBT) SRAM with no bus latency memory is a synchronous-burst SRAM with a simplified interfaced that fully uses available bandwidth.

ZBT SRAM devices use the full bandwidth because they do not require turnaround cycles. A turnaround cycle is idle cycles between read and write or write and read operation.

An external ZBT memory controller was implemented to connect a ZBT SRAM with the PowerPC. According to PowerPC buses, data rate expected is 40MB/s.

This external memory is used to store data from the PowerPC. Thus, memory controller has to support several accesses like byte, half word (2 bytes) and word (4 bytes). And the memory map should be laid out to follow guidelines set out by DIAMOND. For example, internal memory must precede external memory.

Peripherals / Features	External ZBT memory controller	Comport link	PowerPC
Type	ZBT - K7N321801M	Com-port	PPC 405
Clock	100MHz	50MHz	300MHz
width	2x18bits (4Mbytes)	32bits	-
Number device	2(separate)	2(comport 3 et 2)	2
Bus interface	OPB/PLB s	OPB	OPB/PLB
Burst mode	Supported	-	-
Access mode	8bits – 16bits – 32bits read and write	-	-
mode	Pipeline	-	-
Implementation	IP block	IP block-connector block	-
Internal memory	Consecutive	-	64Kbytes
External memory	Consecutive	-	8Mbytes
instruction-cache	-	-	4Kbytes
data-cache	-	-	4Kbytes
Data on-chip memory(DOCM)	-	-	16Kbytes
Instruction on-chip memory(IOCM)	-	-	16Kbytes
Debug port	-	-	JTAG
Target	virtex4	Virtex2P- virtex4	Virtex2P- virtex4
Sundance board	SMT339, SMT398, SMT145FX	SMT339, SMT398, SMT145FX, SMT338VP	SMT339, SMT398, SMT145FX, SMT338VP

Table 1 specification

The following scheme describes the design expected for this application note.

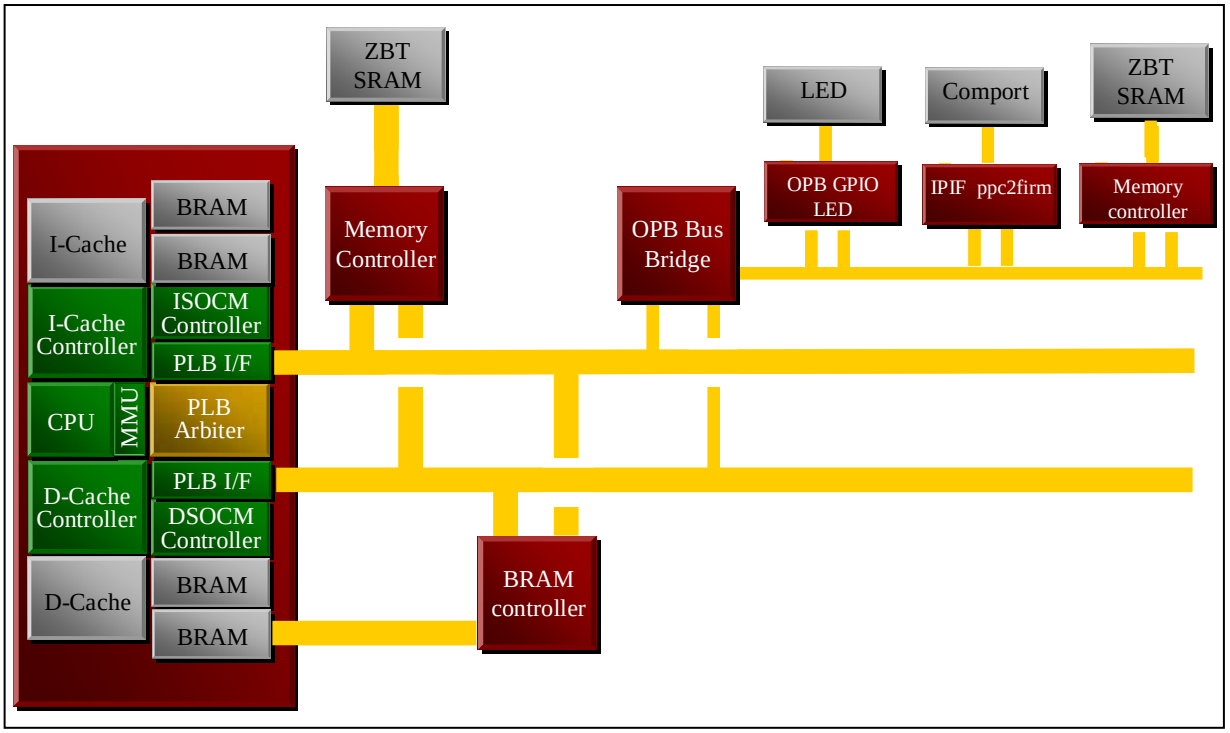

Figure 1 design expected

1.3 Background

This paragraph describes blocks implemented such as Sundance resources, external memory controller and PowerPC.

1.3.1 Embedded processor

As described before, there are two types of embedded processors. Hard and soft core.
A “soft” processor is built using the FPGAs general-purpose logic. The “soft” processor is described in a Hardware Description Language (HDL) or netlist. Unlike the hard processor, processor such as Microblaze must be synthesized and fit into the FPGA fabric.

Xilinx produces FPGA families that embed a physical processor core into the FPGA silicon.
A processor built from dedicated silicon is referred to as a “hard” processor. Such is the case for the PowerPC. 405 inside the Xilinx Virtex-II Pro and Virtex-4 families.

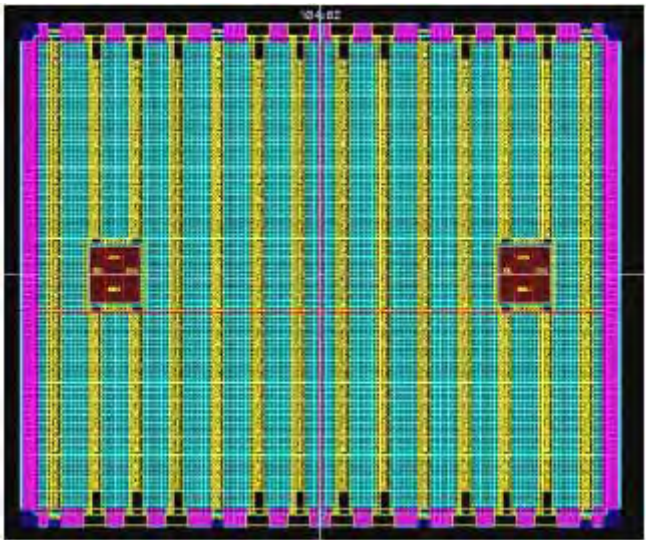

Figure 2 PowerPC hard processor

In both soft and hard processor systems, the local memory, processor busses, internal peripherals, controllers, and memory controllers must be built from the FPGA's general-purpose logic.

The PowerPC 405 processor is a 32-bit implementation of the PowerPC embedded environment architecture that is derived from the PowerPC architecture. Specifically, the PowerPC 405 processor is an embedded PowerPC 405D5 (for Virtex-II Pro) or 405F6 (for Virtex-4) processor core. The term processor block is used throughout this document to refer to the combination of a PPC405, on-chip memory logic (OCM), an APU controller (Virtex-4 only), and the gasket logic and interface.
The processor core consists of a 5-stage pipeline, separate instruction and data cache units, virtual memory management unit (MMU), three timers, debug, and interface to other functions. **Figure 3** illustrates the logical organization of the PPC405Fx.

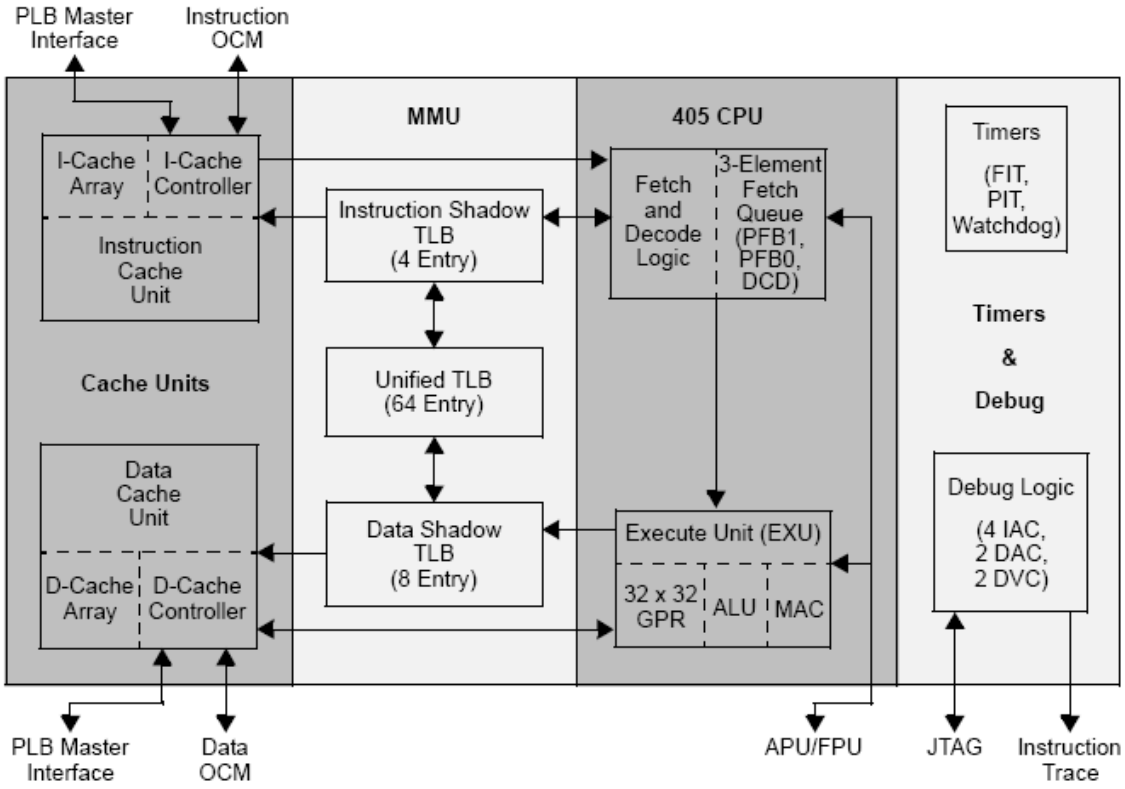

Figure 3 PowerPC405Fx block diagram

The PowerPC is designed along RISC principles, and allows for a superscalar implementation. The PPC405 supports both big-endian and little-endian byte ordering, but by default, the PowerPC Architecture is Big Endian. In Big Endian, the MSB of an instruction word is assumed to be at the lowest address.

LSB			MSB
0x03	0x02	0X01	0x00

Table 2 big Endian byte ordering

MSB			LSB
0x03	0x02	0X01	0x00

Table 3 little Endian byte ordering

In a Little Endian program the same instruction is arranged with the least significant byte (LSB) of the instruction word at the lowest numbered address.

CoreConnect™ is an IBM-developed on-chip bus-communications link that enables chip cores from multiple sources to be interconnected to create entire new chips. The CoreConnect technology eases the integration and reuse of processor, system and peripheral cores within standard product platform designs to achieve overall greater system performance. The CoreConnect bus architecture includes the Processor Local Bus (PLB), the On-chip Peripheral Bus (OPB), a bus bridge, two arbiters, and a Device Control Register (DCR) bus.

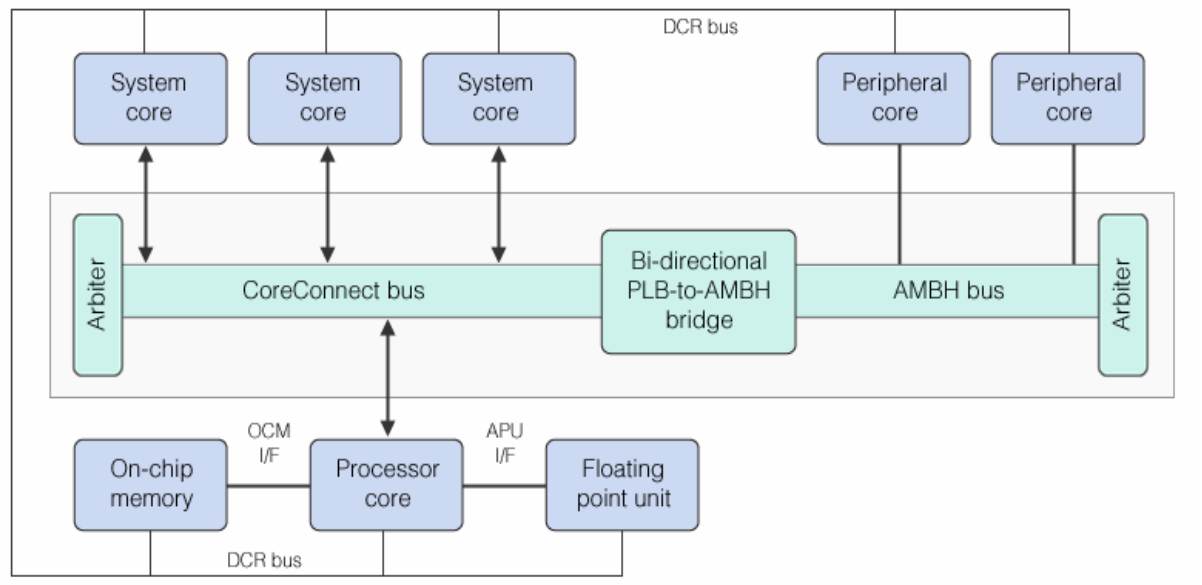

Figure 4 CoreConnect bus architecture

The 64-bit Processor Local Bus (PLB) consists of a bus control unit, a watchdog timer, and separate address, write and read data path units with a three-cycle-only arbitration feature. It contains a DCR slave interface to provide access to its bus error status registers. It also contains a power-up reset circuit to insure that a PLB reset is generated if no external reset has been provided. The PLB consists of a central bus arbiter, the necessary bus control and gating logic, and all necessary bus OR/MUX structures. The PLB provides the entire PLB bus structure and allows for direct connection for up to 16 masters and 16 slaves.

The On-chip Peripheral Bus (OPB) implements a 32-bit address bus and a separate 32-bit data bus. Transaction widths can be full-word, half-word, or byte-size. The bus supports 8-, 16-, and 32-bit wide device interfaces (aligned on the left-most byte).

It's a synchronous bus, data transactions can take a single cycle (for matched clocks), and burst operations are supported. Designed to support slower peripherals, the OPB is implemented as a straightforward multimaster, arbitrated bus.

The DCR bus provides an alternative path to the system for setting the individual device control registers. With it, the host CPU can set up the device-control-register sets without loading down the main PLB. This bus has a single master, the CPU interface, which can Read or Write to the individual device control registers. The bus employs a ring implementation to connect the CPU interface to the devices, which are addressed via a 10-bit address bus. A separate 32-bit data bus transfers register data. This is a synchronous bus.

A bridge is a synthesizable core that permits transfers of code and data between two different buses as such the Advanced Microcontroller Bus Architecture (AMBA) advanced high-performance bus (AHB) and the CoreConnect PLB (see the **Figure 4**).

A typical system-on-a-chip (SOC) implementation based on the PowerPC 405 CPU and CoreConnect, uses a three level bus structure for system level communication, configuration and control functions. High bandwidth memory and system interfaces are tied to the PowerPC 405 Core via the Processor Local Bus (PLB). Less demanding peripherals share the On-chip Peripheral Bus (OPB) and communicate to the PLB through the OPB bridge.
This three level bus architecture provides common interfaces for the IP cores. This application note is according to this architecture. **Figure 5** illustrates a representative SOC configuration.

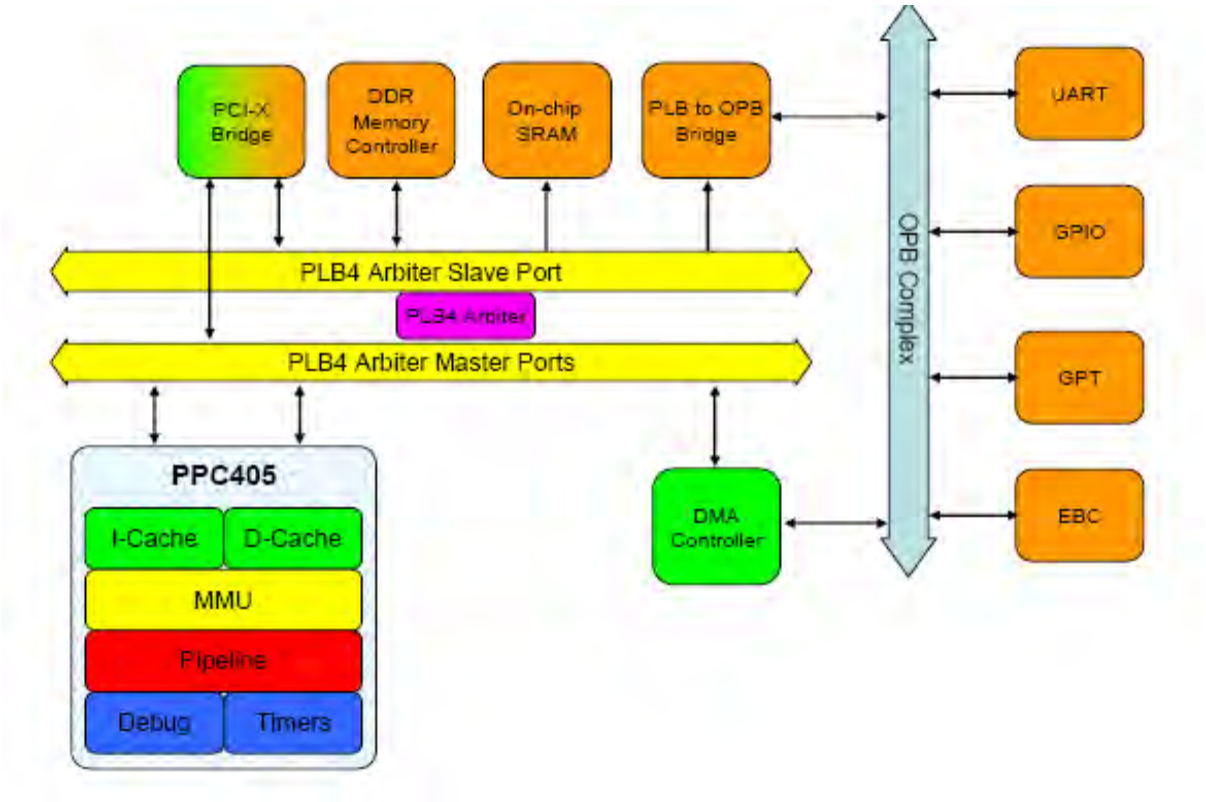

Figure 5 PowerPC405 CPU and CoreConnect based SOC example

1.3.2 External ZBT memory

This design was developed on the SMT339. On this board, 8 Mbytes of ZBT SRAM is provided as a FPGA memory resource. The memory is based on two separate Samsung K7N321801M-20 devices which are each 2M by 18-bit devices, allowing independent access of each device.

The K7N321801M is Synchronous Static SRAMs. The N*t*RAMTM, or No Turnaround Random Access Memory utilises all the bandwidth in any combination of operating cycles.

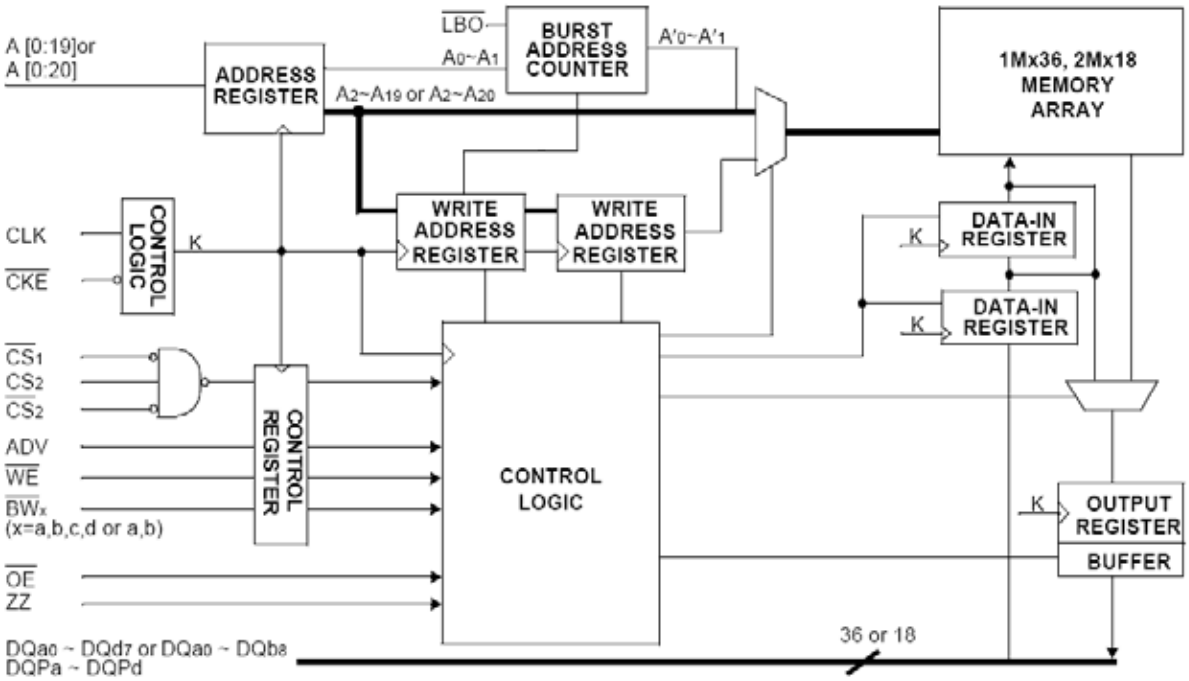

Figure 6 ZBT memory block diagram

The following figure explains the timing feature for this memory.

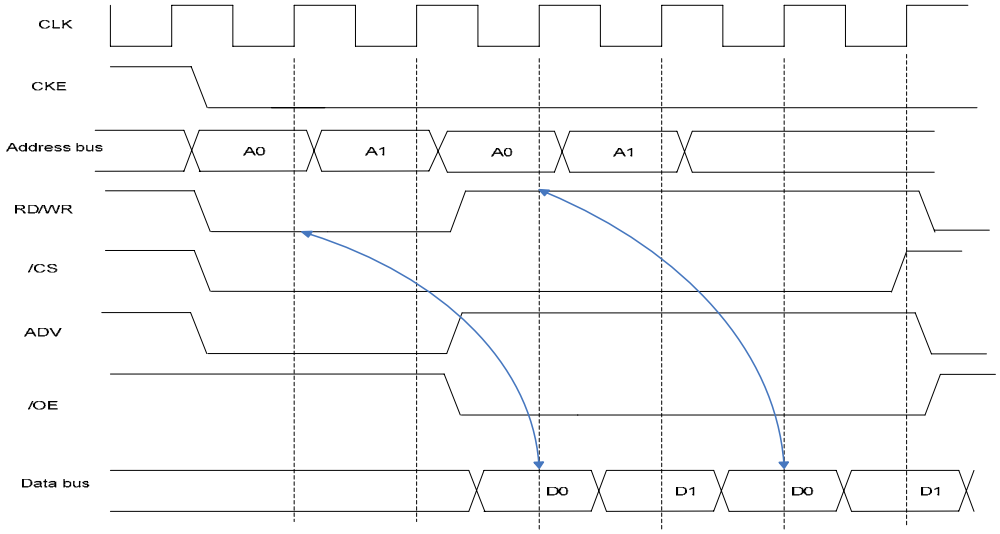

Figure 7 timing feature of the ZBT memory

Xilinx provides external memory controllers (Xilinx EMC PLB). These modules support data transfer between the Processor Local Bus or On-chip Peripheral Memory and external synchronous and asynchronous memory devices. These kinds of controller receive control signals from the PLB or OPB to read and write to external memory devices.

The PLB/OPB EMC supports PLB data bus widths of 8, 16 ,32, 64bits and OPB data bus widths of 8,16,32 bits, and memory subsystem widths of 8, 16, 32 (64bits for PLB). However, this ZBT control does not support sleep mode, burst mode (memory mode), parity checking or parity generating. “This controller does not support burst mode” means that the user can read and write to the ZBT memory in burst mode but the burst sequence provides by the memory is not used; a signal is used to load a new external address or increment the initial burst counter. In this case, the ZBT memory loads a new external address every read or write access.

Some features of the OPB/PLB EMC can be parameterized such as number of separate memory banks, memory type (synchronous or asynchronous), data width of each memory bank, pipeline delay of each memory, read and write access times for each memory bank, enabling of data-width matching per memory bank, enabling burst support and enabling negative edge IO registers enabling of PLB burst and cache line transaction support. The descriptions of every OPB/PLB EMC parameters are described in the following section.

1.4 Test bench configuration

This section explains hardware and software required for this application.

1.4.1 Hardware set up

This application note was developed on the SMT339. This board is a dedicated high speed image processing module for use in a wide range of image analysis systems. The module can be plugged into a standard TIM single width slot and can be accessed by either a standard Comport, or Rocket Serial Link (RSL) Interface. The image processing engine is based upon the 'Texas Instruments' TMS320DM642 Video Digital Signal Processor. It is fully software compatible with C64x using Code Composer Studio.

The DM642 runs at a clock rate of 720MHz. It features two levels cache based architecture. There are 16K Bytes of level one program cache (Direct mapped), 16K Bytes of level one data cache (2-Way Set-Associative) and 256K Bytes of level two cache that is shared program and data space (Flexible RAM/Cache Allocation). TheDM642 can perform 4, 16 x 16 Multiplies or 8, 8 x 8 Multiplies per clock cycle.

A powerful Vitrex-4 FPGA (XC4VFX60-11) is used onboard as the FPGA processing unit for image data. 8 Mbytes of ZBT SRAM is provided as a FPGA memory resource. Processing functions such as Colour Space Conversion (CSC), Discrete Cosine Transforms (DCT), Fast Fourier Transforms (FFT) and convolution can be implemented, without using any of the DSP's resources. The Virtex 4FX60 has 2 Power PC hardware cores that can be incorporated into this design.

The basic block diagram of the SMT339 and its components is illustrated **Figure 9**.

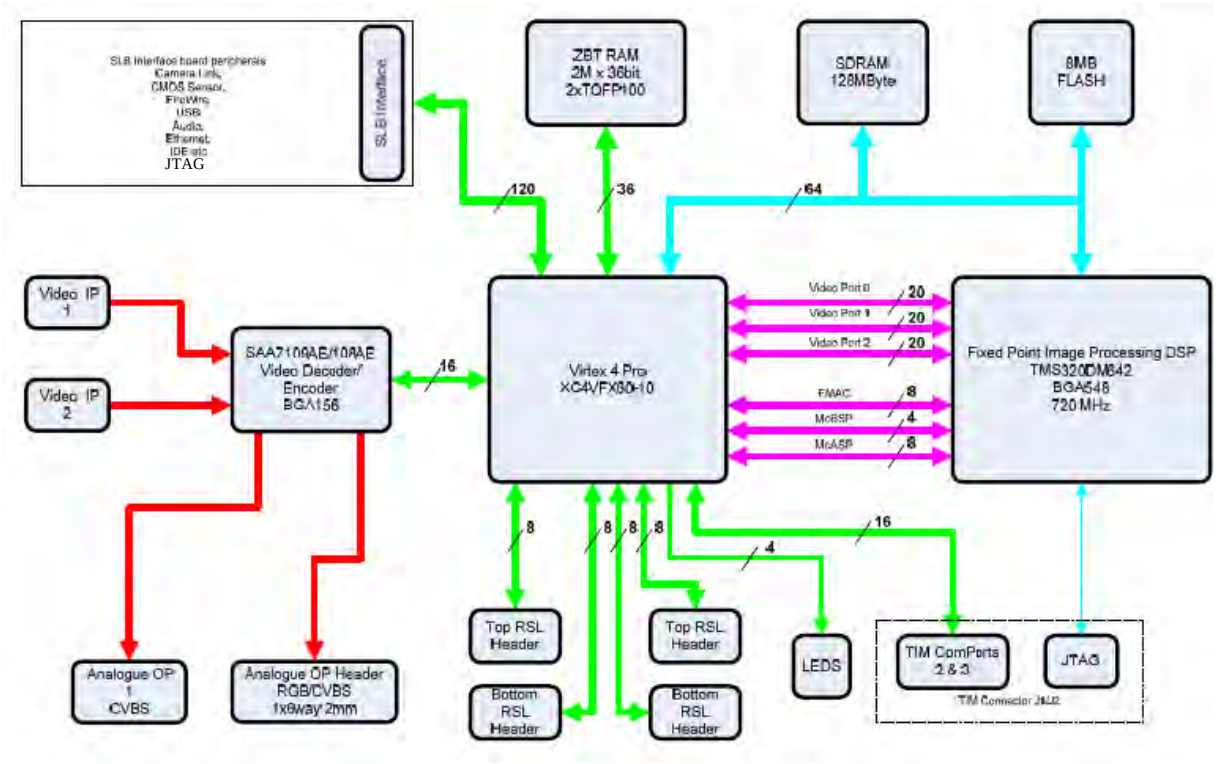

Figure 8 SMT339 diagram block

There are two separate JTAG chains on the SMT339 module. One allows the DSP chain to be accessed while the other allows the Virtex 4 to be configured. The last one was used to debug the PowerPC with Xilinx Micro-Processor debug (XMD).

The SMT 319 was connected to the SMT339 via comport link to print the result on the host.

The SMT319 is a Graphics and Imaging Module based on Texas Instruments (TI) high-performance, TMS320C6414 digital signal processor (DSP), that allows original equipment manufacturers (OEM) designers to easily capture, process and display images without any host computer involvement and as an embedded solution if necessary.

SMT319 is suitable for a wide range of image processing applications including, graphics, rendering of pictures and general-purpose display sub-systems in larger multiDSP processing systems. This solution is supported on major platforms like PCI, CPCI, PXI and VME.

The module also includes a Xilinx Virtex-II FPGA, which is configured to provide “C4x” style Com-Ports, or alternatively the Sundance Digital Link (SDL), a TIM compatible enhanced global bus, two Sundance High Speed Bus (SHB) and other control functions.

1.4.2 Software set up

To simplify the design process, Xilinx offers several sets of tools. The Xilinx Embedded Development Kit (EDK) is a suite of tools and IP that allows to design a complete embedded processor system for implementation in a Xilinx Field Programmable Gate Array (FPGA) device.

EDK includes XPS and SDK software. Xilinx Platform Studio (XPS) is the development environment or GUI used for controlling the hardware and software development of the embedded processor system.

Software Development Kit (SDK) is an integrated development environment, complimentary to XPS, that is used for C/C++ embedded software application creation and verification. SDK is built on the Eclipse™ open-source framework.

The Integrated Software Environment (ISE) is the foundation for Xilinx FPGA logic design. Because FPGA design can be an involved process, Xilinx has provided software development tools that allow the designer to circumvent some of this complexity. Various utilities such as constraints entry, timing analysis, logic placement and routing, and device programming have all been integrated into ISE.

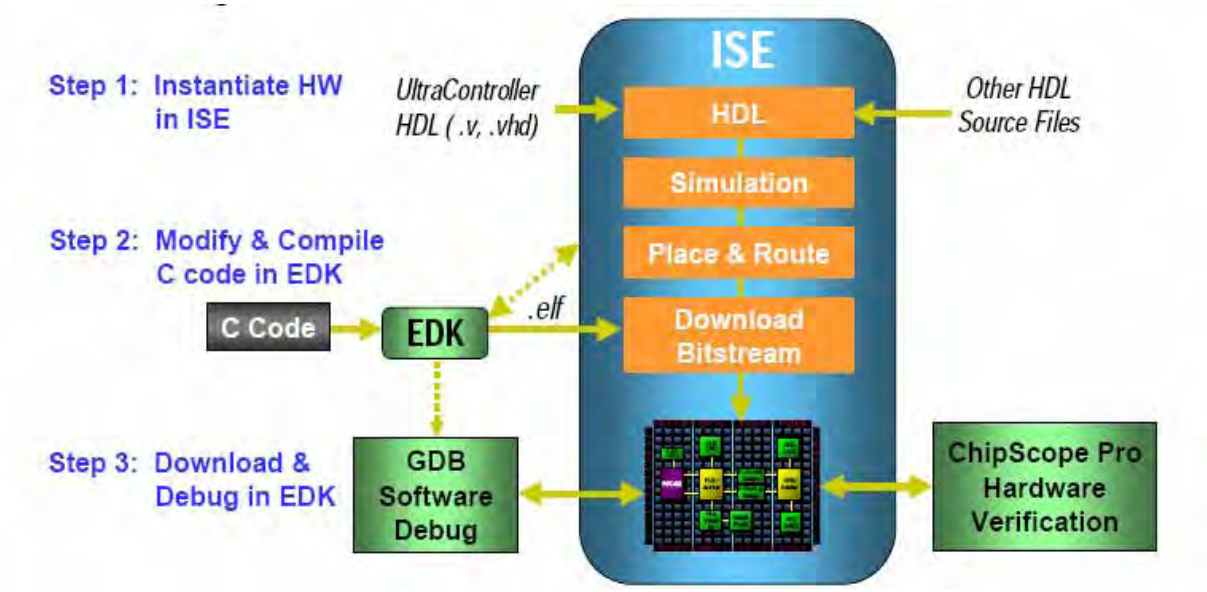

Figure 9 Simplified ISE / EDK Design Flow

Like Xilinx, Texas Instrument provides several sets of tools. Code Composer Studio™ integrated development environment includes host tools and target software that slashes development time and optimizes the performance for all real-time embedded DSP applications.

The CCStudio IDE includes DSP/BIOS support, real-time analysis capabilities, debugger and optimization tools, C/C++ Compiler, Assembler, Linker, integrated CodeWright editor, visual project manager, and a variety of simulators and emulation drivers.

Figure 10 development cycle with Code Composer Studio

Software required for this reference design is described in the following table:

Software		version
ISE		9.1 SP3
EDK		9.1 SP2
Code Composer Studio		3.3

Table 4 Software version

1.5 Design implementation

The main objective explained and every specification defined, the next step is the implementation of this application.

1.5.1 PowerPC block

This paragraph demonstrates process of creating and testing a PowerPC system design using the Embedded Development Kit (EDK).

1.5.1.1 Overview

This design implemented the PowerPC block and General Purpose Input/Ouput (GPIO). This peripheral was used to flash LEDs on the board. The PowerPC was running at 100MHz like its communication links. By a basic example, this design allowed users to understand the PowerPC architecture and use Xilinx's software and hardware. In this example, the design flow was generated by EDK.

1.5.1.2 Constraints

According to the requirement, all system was running at 100 MHz to make easier the implementation. This constraint can be checked in the MHS file. The Microprocessor Hardware Specification (MHS) file describes the embedded processor: either the soft core MicroBlaze processor or the hard core PowerPC, peripherals and associated address spaces, buses and overall connectivity of the system.
The MHS file is a readable text file that is an input to the Platform Generator; the hardware system building tool. Conceptually, the MHS file is a textual schematic of the embedded system.

For the Sundance board, the bitgen file needs to be changed. It contains every setting for JTAG configuration. By default, the device is synchronised to the clock provided by JTAG.

This option specifies the start-up sequence for the device synchronisation. But Sundance's TIMs need clock system to download the bitstream before connecting to the JTAG. Thus, the default clock startup needs to be replaced by the clock provided in the FPGA device: "-g StartUpClk:CCLK".
If you don't change this parameter, you cannot download the bitstream into the FPGA with the Sundance package (SMT6001).

For design entry in VHDL, only a few constraint options can be embedded in the source code. In this case, designers have the option to use a UCF. User constraint files allow designers to manually control some of the following attributes in a FPGA design: pin and node assignments, slew rate control, fit options, input pin characteristics, initial register state, global clock use. In this design, the user constraint file contains only five pin assignments: Clk, Rst, LED(0 to 2).

This implementation includes both hardware and software. In fact, this design needs components such as PLB controller, BRAM…and a software application as well.

1.5.1.3 Hardware implementation

Figure 12 shows the system assembly view created with the Base System Builder.

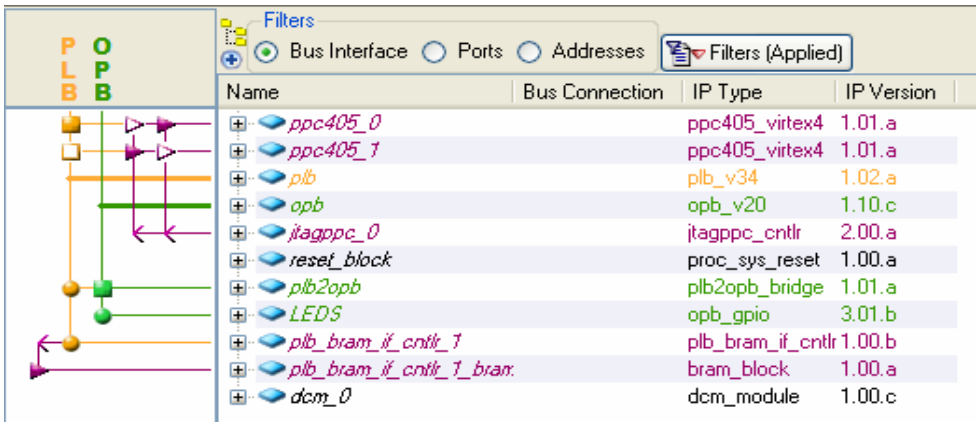

Figure 11 system assembly view

Both PowerPC cores were implemented: ppc405_0 and pp405_1. But only the ppc405_0 was used. This basic example includes both CoreConnect bus architecture the Processor Local Bus (PLB) and the On-chip Peripheral Bus (OPB). The ports tab gives further information about the connectivity between the blocks.

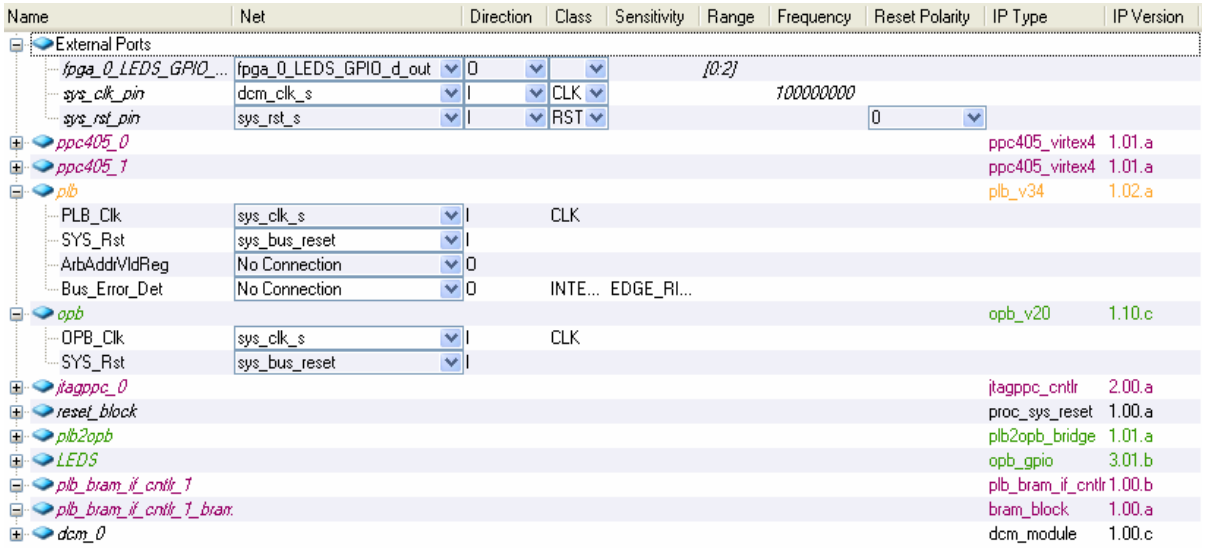

Figure 12 blocks connexions

Blocks connexions allowed user to check the frequency, the reset polarity and every connexion between blocks. It is the graphic interface of the MHS file.

The following table shows the memory map for this design as created by Base System Builder.

Device	Address		Size	Comment
	Min	Max		
PLB_BRAM_CNTLR	0xFFFFC000	0xFFFFFFFF	16K	PLB Memory
OPB_GPIO	0x40000000	0x4000FFFF	64K	LED output
PLB2OPB	0x40000000	0x4000FFFF	64K	Bridge

Table 5 design memory map

It is important to check the address of every component. In fact, a peripheral can be controlled by the PowerPC, if it is part of PowerPC address range.

Figure 14 describes hardware component implemented by the wizard.

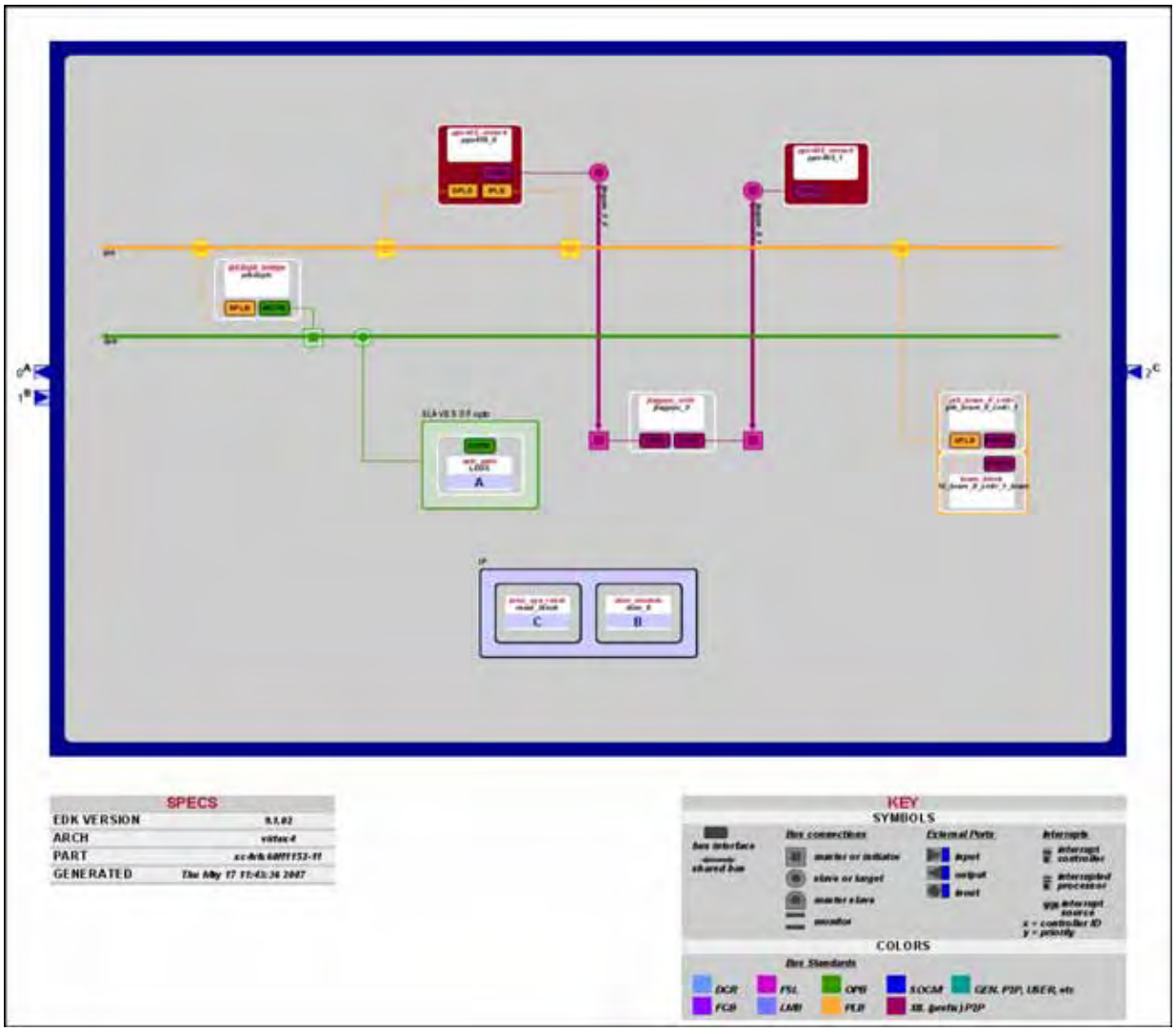

Figure 13 Block diagram PowerPC - GPIO

After that the hardware had been completely specified in the MHS file, Platform Generator was able to run. Platform Generator elaborates the MHS file into a hardware system consisting of NGC files that represent the processor system. Then it can generate a netlist and create the bit file.

1.5.1.4 Software implementation

The next step is defining the software design. There are two major parts to software design, configuring the Board Support Package (BSP) and writing the software applications. The configuration of the BSP includes the selection of device drivers and libraries.

Software application code development for this design is shown in **Figure 15**.

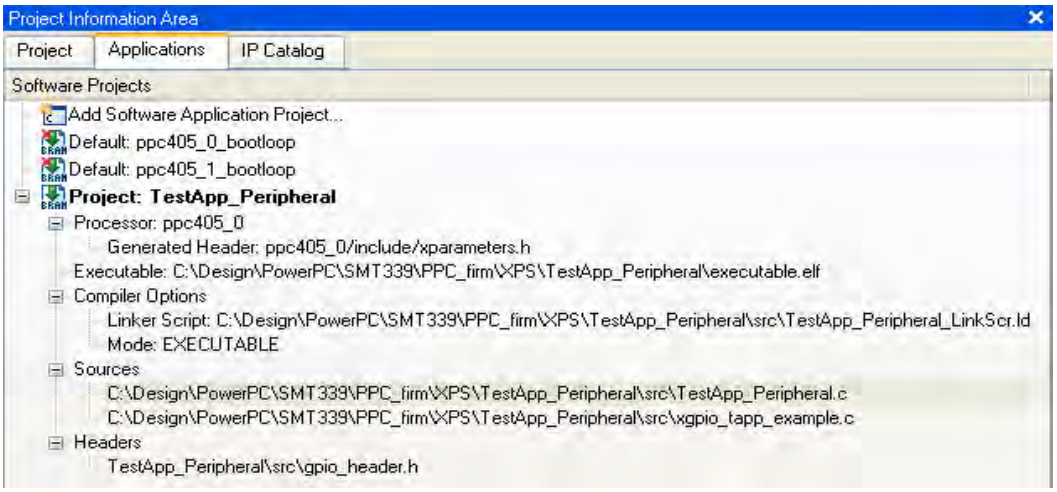

Figure 14 application tab

The Base System Builder (BSB) generates a sample application which tests a subset of the peripherals included in the design.

Both sources code TestApp_Peripheral.c and xgpio_tapp_example.c were added to this project. They were generated by BSB. This application set the Led to high one following the other. These sources code were compiled using the GNU GCC Compiler.

The compiled software routines are available as an Executable Format (ELF) file. The ELF file is the binary ones and zeros that are run on the processor hardware.

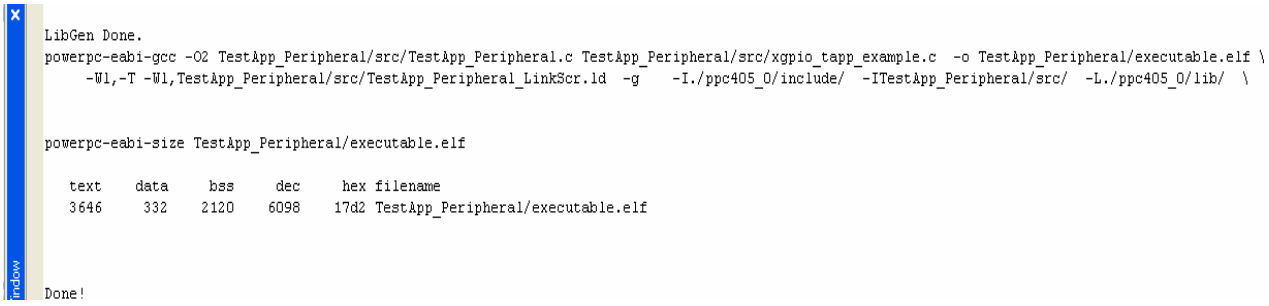

Figure 15 XPS output widows - software compiled

1.5.1.5 Test and validation

After that the hardware and software designs were completed, device can be configured. The Host was connected to the target board via the Parallel-JTAG cable IV.

The figure below shows the test scheme.

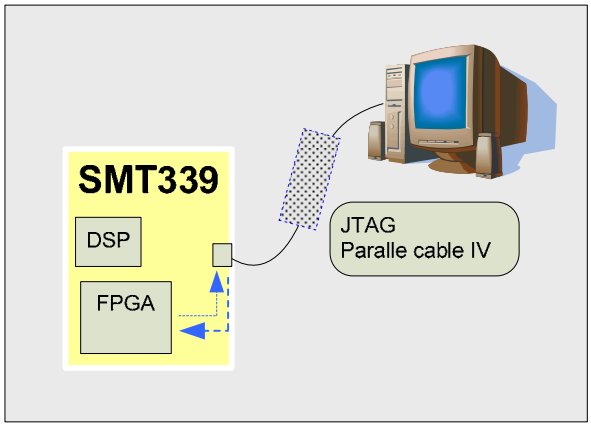

Figure 16 test scheme

The bitstream was downloaded by SMT6001 package. The main program of the SMT6001 package, the Flash Programming Utility, manages the Flash ROMs on Sundance TIMs. The Flash Programming Utility uses Texas Instruments' Code Composer™ to allow the user to analyse the contents of a ROM and program new data. For added security, the Flash Programming Utility will also allow the user to erase the ROM completely.

Both files are generated by EDK: PPC.bit and download.bit. For software downloading: software can be initialized into the bitstream if it fits inside FPGA internal block RAM (BRAM) memory. Else the software debug tools can be used, such as the XPS Software Development Kit (SDK), to download a program to the board.
Download.bit file merges the FPGA bitstream and ELF files into a single bitstream file. When this bitstream is downloaded into the FPGA, application runs in standalone after a reset board. Opposed to PPC.bit file that only configures the device. This bitstream is used to debug the software directly via the JTAG PPC connections. GDB connects to the PowerPC core through the JTAGPPC and the Xilinx Microprocessor Debug (XMD) engine utility as shown in **Figure 18**. XMD is a program that facilitates a unified GDB interface and a TCL (Tool Command Language) interface for debugging programs and verifying microprocessor systems.

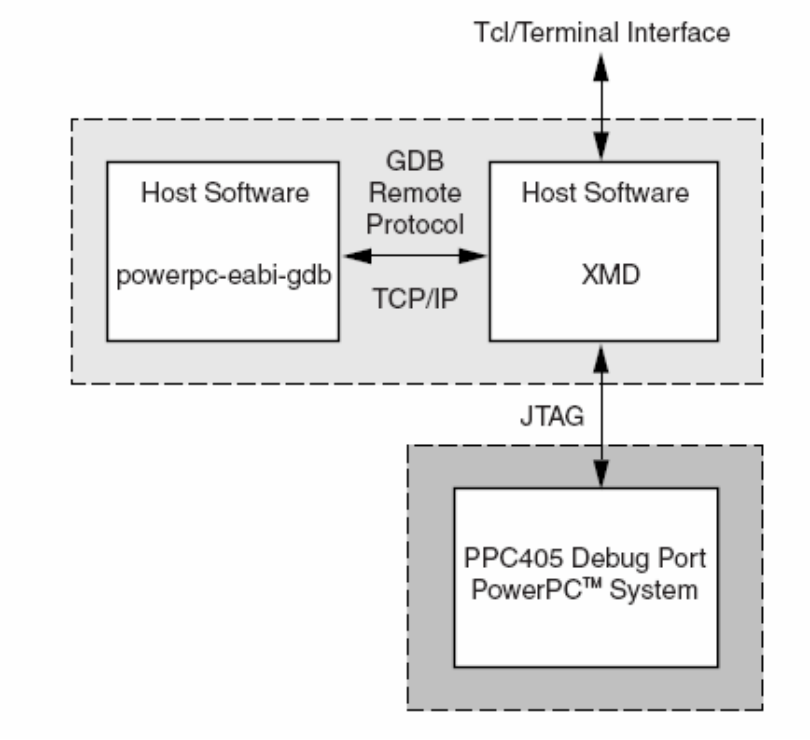

Figure 17 XMD PowerPC system connexion

This design was validated with these both methods. Before starting the next design, it was important to check features and the resources used by this application.

This design used 4 percents slices into the FPGA. Overview about the embedded processor shows both PowerPC implemented used no resource. That is according to the paragraph describing the embedded processor into FPGA. Indeed, this kind of processor is built from dedicated silicon. Thus, its implementation “doesn’t use” FPGA resources.

The following **Table 6** explains the program size.

Text(Kb)	Data	bss	dec	hex
3646	332	2120	6098	17d2

Table 6 program size

Post Synthesis Device Utilisation			
Resource Type	**Used**	**Available**	**Percent**
Slices	1073	25280	3
Slice Flip flops	1024	50560	1

Table 7 device utilisation

More than 3 percents slices were used by the PLB controller and the PLB/OPB bridge. Thus, to optimise the resources used by a design, it would be more interesting to implement every peripheral onto the PLB bus.

Just to check this remark, a design without OPB bus interface was created. **Figure 19** shows the memory map and components attached to the PLB.

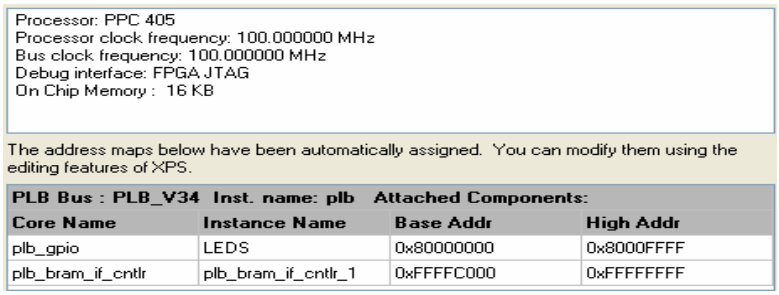

Figure 18 memory map without OPB bridge

The following **Figure 20** explains bus interfaces. None OPB bus was implemented.

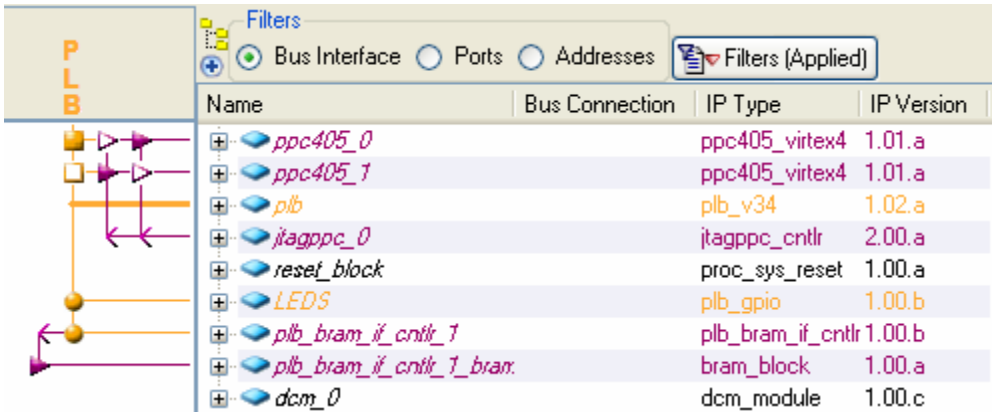

Figure 19 bus interface without OPB bridge

Only one percent slices was used by this design. This feature is quite important when the area is critical.

1.5.2 Comport link implementation

The first design allowed you to create and test a PowerPC system design using the Embedded Development Kit (EDK). In this section, the Comport link was implemented to improve the previous design. The first step defined the requirements.

1.5.2.1 Overview

This design implemented an external peripheral: Comport. The Comport was instantiated like an IP core. A Xilinx's module was used to provide an interface between the OPB bus and the Comport IP core. This design was exported to ISE for the Comport block integration. Indeed, ISE describes every step of the design flow and control the hardware implementation in the FPGA. The Comport link should transfer data to 20MB/s. Every bus and PowerPC had to run at 100MHz according to the specifications.

1.5.2.2 Constraints

The Comport link must be implemented according to Sundance concept of view. All of the Sundance TIMs provide mechanisms to transfer data between modules. The mechanisms are implemented on the TIMs using FPGAs.
The FPGA (Field Programmable Gate Array) is a device that can be configured by software to implement hardware functions. Sundance TIMs use FPGAs to create peripheral devices such as Comports.
Before a TIM can access the mechanisms that will allow it to communicate with other modules, the DSP must load the FPGA with appropriate firmware. This firmware is held in the FPGA data segment of the flash, and will be installed in the FPGA each time the bootloader executes.

Sundance uses modularity in its designs to improve reusability and maintainability. This approach creates a number of blocks which are used to create the firmware. The role of the firmware is to allow these peripherals to exchange data.
The FPGA is connected to the peripherals via physical wires and these peripherals may be zero or more processors or connectors or other components on the module.

The firmware includes **processor blocks**, **interface blocks**, and **connector blocks**. All blocks are interconnected with pre-defined signals. The connector and processor blocks are connected to the physical pins of the FPGA. This is a standard case of figure. Some modules do not have a processor connected to the FPGA. Usually the firmware for these modules makes use of the standard connector blocks, but the processor and interface blocks may be replaced by a custom controller block.

The next section explains how the Comport link was implemented.

1.5.2.3 Hardware implementation

One of the key advantages of building an embedded system in an FGPA is the ability to include customer IP and interface that IP to the processor. The processor system by EDK is connected by OPB and/or PLB bus, so this custom peripheral: Comport, must contain a set of bus ports that is compliant to OPB or PLB protocol.

To connect the custom peripheral, Comport, to the PowerPC, the IP core must be attached the system OPB or PLB bus as shown the **Figure 21**.

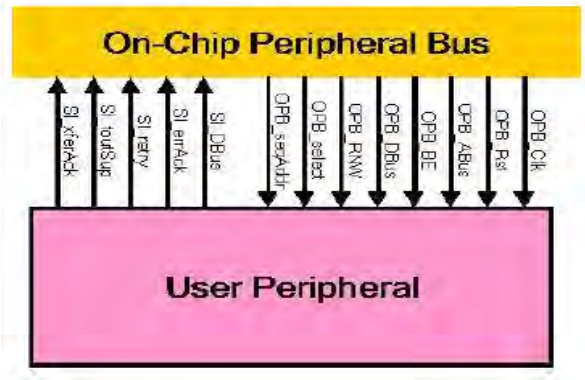

Figure 20 OPB bus protocol example

Meaning the top-level module of the custom peripheral must contain a set of bus ports that is compliant to OPB or PLB protocol. Another constraint adds to this one.

To use the custom peripheral in XPS, the peripheral must be stored in a place (directory structure) that is accessible by various EDK tools; and it must have the correct Platform Specification Format interface files: MPD and PAO files. But Xilinx provides Intellectual-Property Interface (IPIF) library to implement common functionality among various processor peripherals.

Both OPB/PLB IPIF are designed to provide a bidirectional interface between a user IP core and the IBM OPB or PLB Bus.

The back end interface standard, the Xilinx IPIC, is common in most aspects between IPIF modules in various versions for the OPB and the PLB for the features supported. This allows IP blocks using the IPIC to be adapted for either the OPB or the PLB. **Figure 22** shows a block diagram of the OPB IPIF.

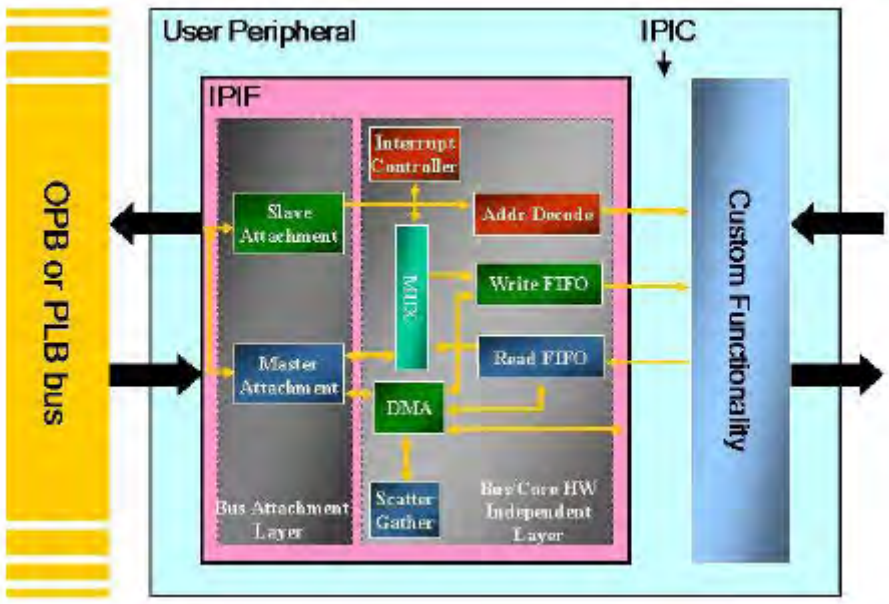

Figure 21 IPIF module

According to the specification, the custom peripheral was connected to the OPB bus. The IPIF was generated by “Create and Import Peripheral Wizard” functionality. It was called “ppc2fir”. It means PowerPC to firmware. The wizard automatically created corresponding OPB peripheral templates to add the custom peripheral. Two HDL template files were generated: peripheral top-level and user-logic HDL templates.

The top-level file, ppc2fir.vhd, is the template file for the peripheral's top design entity. It configures and instantiates the corresponding IPIF unit in the way you indicated in the wizard GUI and connect it to the stub user logic where the user logic should get implemented.

The recommended design using wizard is added the user port mapping in the top-level file and incorporated the custom IP functionality in the file user_logic. In this case, the software can access to the peripheral thanks to the software addressable registers. It means the user logic component of the peripheral has registers that are addressable through software. But the Comport IP was not implemented according to this recommendation.
The Comport implementation needed high constraint onto these control signals. It was easier to define these constraints in ISE. This software describes every step of the design flow allowing designers to check every step to see if the constraints were met. So the Comport IP functionality was not implemented in the user_logic file.

Indeed, the user port mapping was just added in the top-level file as shown **Figure 23** and the user_logic component instantiation was deleted. The ppc2firm core was connected to the Comport link in Xilinx ISE.

```
port
(
  -- ADD USER PORTS BELOW THIS LINE ------------------
  --USER ports added here
  Bus2IP_Clk                     : out  std_logic;
  Bus2IP_Reset                   : out  std_logic;
  Bus2IP_Addr                    : out  std_logic_vector(0 to C_AWIDTH-1);
  Bus2IP_RNW                     : out  std_logic;
  IP2Bus_Ack                     : in std_logic;
  IP2Bus_Retry                   : in std_logic;
  IP2Bus_Error                   : in std_logic;
  IP2Bus_ToutSup                 : in std_logic;
  Bus2IP_ArData                  : out  std_logic_vector(0 to C_MAX_AR_DWIDTH-1);
  Bus2IP_ArBE                    : out  std_logic_vector(0 to C_MAX_AR_DWIDTH/8-1);
  Bus2IP_ArCS                    : out  std_logic_vector(0 to C_NUM_ADDR_RNG-1);
  IP2Bus_ArData                  : in std_logic_vector(0 to C_MAX_AR_DWIDTH-1);
  -- ADD USER PORTS ABOVE THIS LINE ------------------
```

Figure 22 map user port

Figure 24 shows the system assembly view in EDK implementing the “ppc2firm” core.

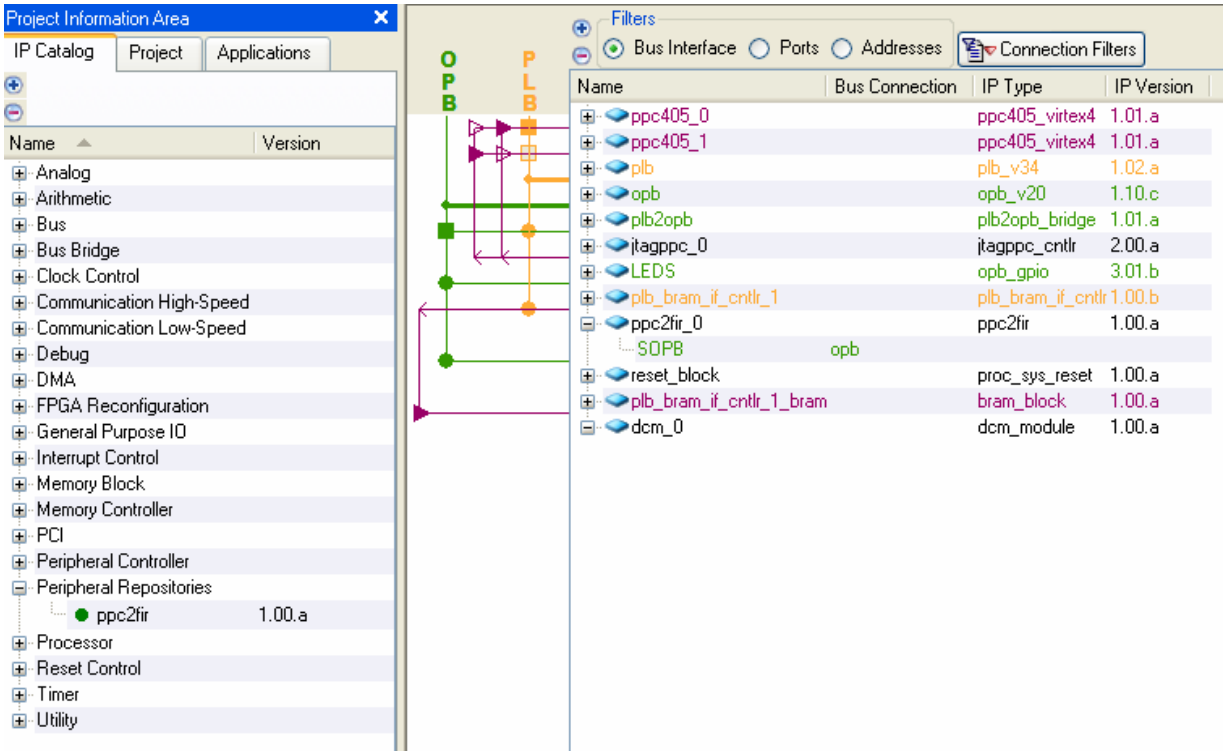

Figure 23 system assembly view - ppc2firm core

The following figure shows the ports for the ppc2firm bloc implemented. These were defined by the wizard. They are always present. These signals correspond to the interface provided from the IPIC bloc.

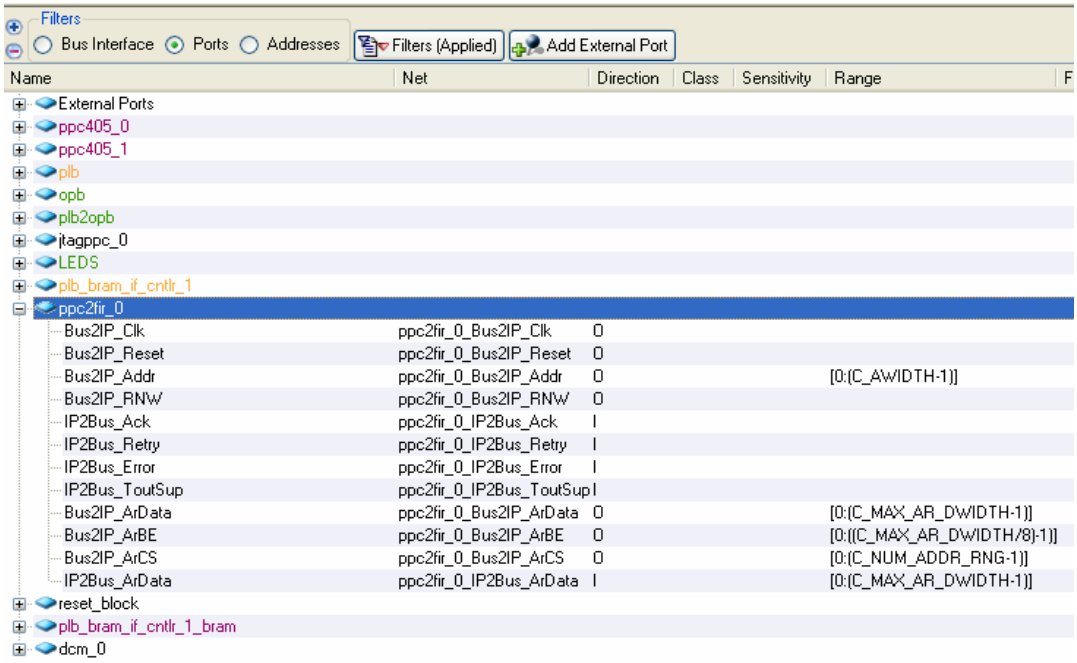

Figure 24 port ppc2firm

An address can be defined for the newly added ppc2firm peripheral by entering the Base Address. The following table describes the memory map of this application.

Device	Address		Size	Comment
	Min	Max		
PLB_BRAM_CNTLR	0xFFFFC000	0xFFFFFFFF	16K	PLB Memory
OPB_GPIO	0x40000000	0x4000FFFF	64K	LED output
PLB2OPB	0x40000000	0x4000FFFF	64K	Bridge
ppc2firm	0x70000000	0x7007FFFF	512K	Custom peripheral

Table 8 memory map - ppc2firm

The following figure describes the hardware implemented. Compare to the previous design, a new block appear onto the OPB. This block is the custom peripheral: ppc2firm.

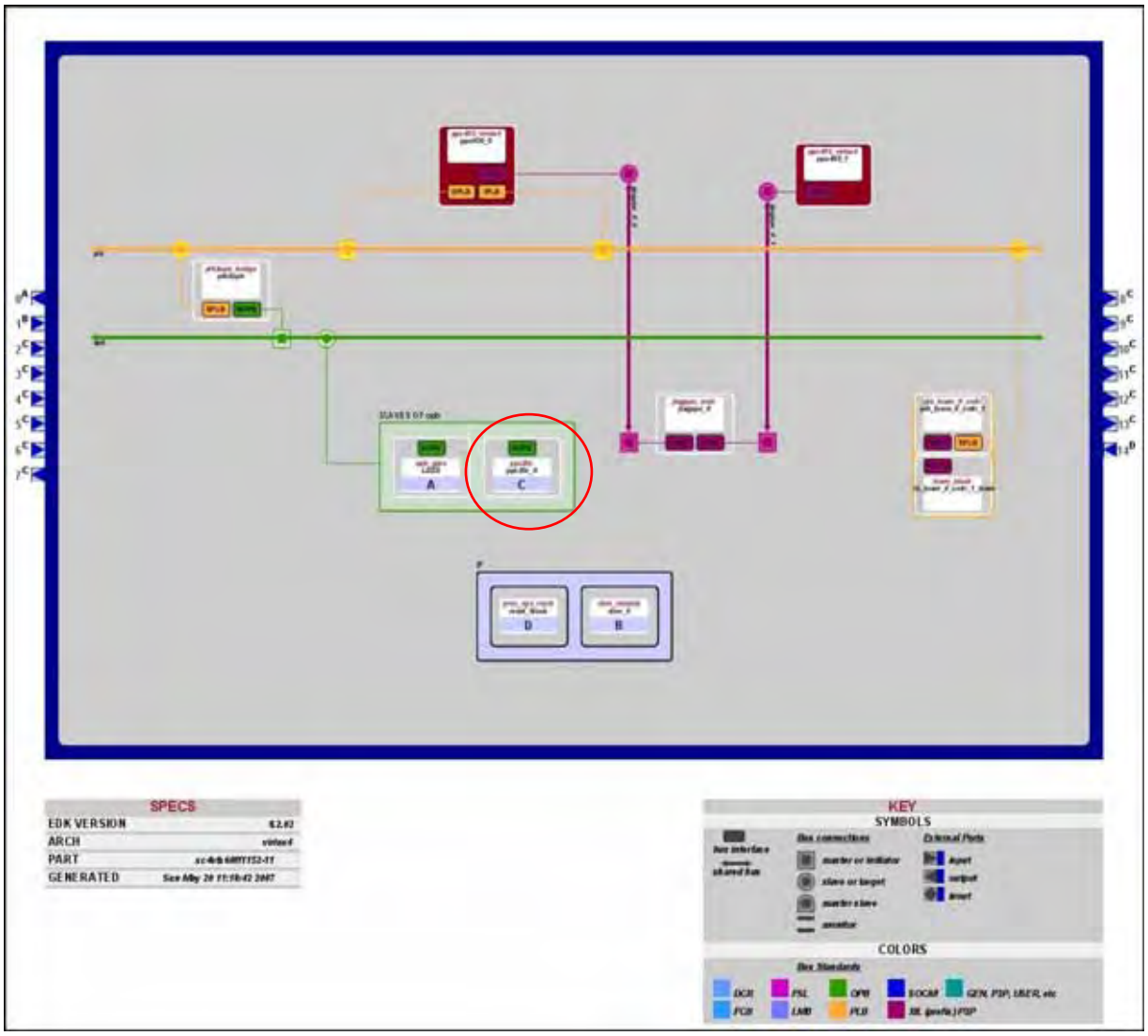

Figure 25 block diagram ppc2firm

As soon as all hardware is defined in the MHS file, Platform Generator can run. Platform Generator elaborates the NGC files that represent the processor system. Then, Xilinx ISE tools are called to implement the design for the target board. It generates a netlist and creates the bit file. So, the EDK project needs to be exported into ISE.

You have to create a new Project Navigator project in ISE and add the ppc.xmp in the New Project Wizard - Add Existing Sources dialog window. In this project, two files have to be created. A file that instantiated the PowerPC with its ports and a top level file that implemented the com-port architecture: interface block and connector block.

ISE offers functionality to generate the PowerPC instantiation template. This option, called “View HDL Instantiation Template” generates a vhi file.

The file ppc_wrapper instantiated the PowerPC and provided the interface between the IPIC and the Comport IP core. This file delivered the following entity:

```
use Smt.CB_pkg.all;

entity ppc_wrapper is
    generic(
        C_AWIDTH          : integer := 32;
        C_MAX_AR_DWIDTH : integer := 32;
        C_NUM_ADDR_RNG   : integer := 1
        );

    port(
        clk  : in std_logic;
        nrst  : in std_logic;
        Leds : out std_logic_vector(0 to 2);
        i_ib : in t_PB_IB_Y;
        o_ib : out t_PB_IB_X
        );
end ppc_wrapper;
```

Figure 26 ppc_wrapper entity

Both signals i_ib and o_ib were used to connect the processor block to the interface. Signals provided by processor block to interface block are grouped in the type **t_PB_IB_X**. The signals provided by interface block to processor block are gathered in the type **t_PB_IB_Y**.

Table 9 gives the description of the signals of the Comport interface block.

Signals coming from processor block (type t_PB_IB_X)		
I_pb.clk	In	Processor block clock. All signals constituting bus i_pb are synchronous to this clock
I_pb.rst	In	General reset for interface block
I_pb.addr	In	Address bus
I_pb.data	In	Data bus
I_pb.read	In	Read request
I_pb.write	In	Write request
Signals going to processor block (type t_PB_IB_Y)		
O_pb.clk	Out	Tri-stated
O_pb.rst	Out	Tri-stated
O_pb.addr	Out	Tri-stated
O_pb.data	Out	Data bus
O_pb.read	Out	Tri-stated
O_pb.write	Out	Tri-stated

Table 9 description of the signals Comport

Figure 28 shows the connection between processor block and interface block

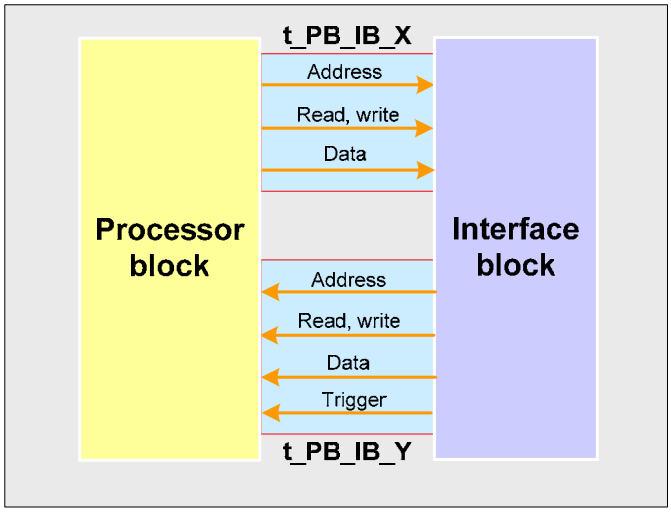

Figure 27 interface between PB and IB

The ppc_wrapper file instantiated the PPC block with these ports. Indeed, every external port defined in EDK project had been declared in the PowerPC component.

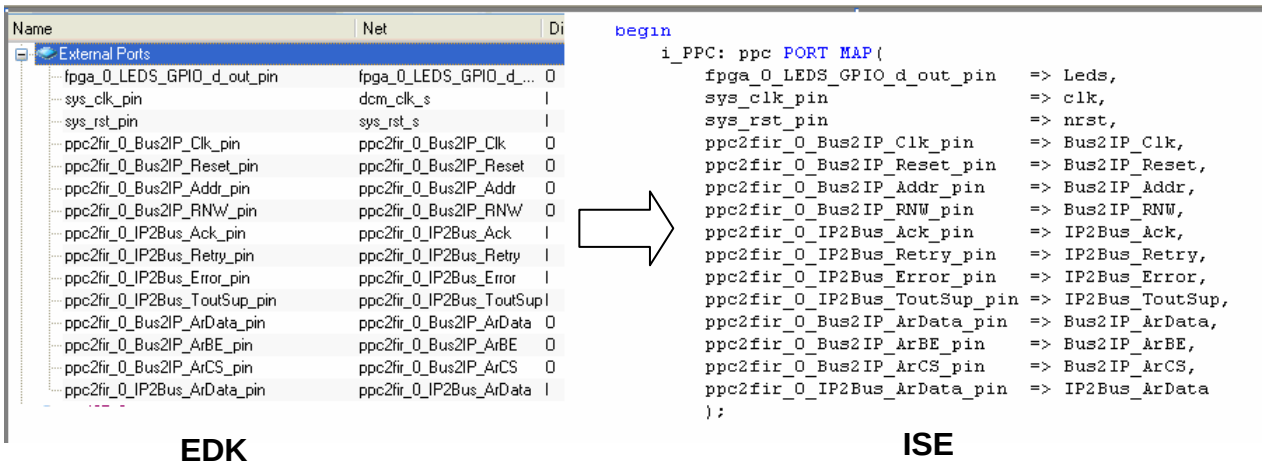

Figure 28 external ports

The VHDL program following the PowerPC instantiation drives signals between Comport and PowerPC external ports (bus signals).

Both data bus and address bus needed processing because the PowerPC provided signals in big Endian and the comport protocol worked in little Endian. This processing is descried by the **Figure 30**.

```
g_data_bus_rev: for i in 0 to 31 generate
    o_ib.data(31-i)    <= Bus2IP_ArData(i);
    ar_data_out(0)(i) <= i_ib.data(31-i);
end generate;

g_add_bus_rev:for i in 12 to 19 generate
    o_ib.Addr(31-i)     <= Bus2IP_Addr(i);
end generate;
```

Figure 29 conversion big Endian - little Endian

The second file, top.vhd, instantiated the connector block, the interface block and provided signals to the PowerPC component.

This file was the top level of this design. So, it provided signals to drive the physical wires. In its entity, only four signals were defined: the clock system, the reset, Leds and the comport link.

```
entity top is  port(
        clk     : in std_logic;
        nrst     : in std_logic;
        Leds    : out std_logic_vector(0 to 2);
        Comport3: inout t_Comport_Pins
        );
end top;
```

Figure 30 top level entity

The “t_Comport_Pins” type describes the comport bus. This signal formed the interface to the pins of the FPGA. Other signals were used such as “t_TriggerVector” and “t_TriggerControl”. The signal “t_TriggerControl” was used to signals go from event block. Contrary to “t_TriggerVector” that was used to signals go to event block. **Table 10** gives the description of these signals.

Signals coming from event block (type t_TriggerControl)		
i_trig_ctl(1..0)	In	Triggers generation control. When i_trig_ctl(i) is high, trigger generation is enabled. When i_trig_ctl(i) is low, trigger generation is disabled and o_trigger(i) stays low
Signals going to event block (type t_TriggerVector)		
O_trigger(0).trigger	Out	Input condition asserted event (pulse)
O_trigger(0).flag	Out	Input FIFO not empty
O_trigger(0).trigger_latch	Out	Latched version of O_trigger(0).trigger
O_trigger(1).trigger	Out	Output condition asserted event (pulse)
O_trigger(1).flag	Out	Output FIFO not empty
O_trigger(1).trigger_latch	Out	Latched version of O_trigger(1).trigger

Table 10 event block signals

The program instantiated both components “IB_Comport_3” and “cb_comport_rcx”. These blocks provided signals according to comport specifications.

To use these specials types, a new library must be created. It called SMT. This library was defined in the libraries tab into ISE. **Figure 32** shows the library created.

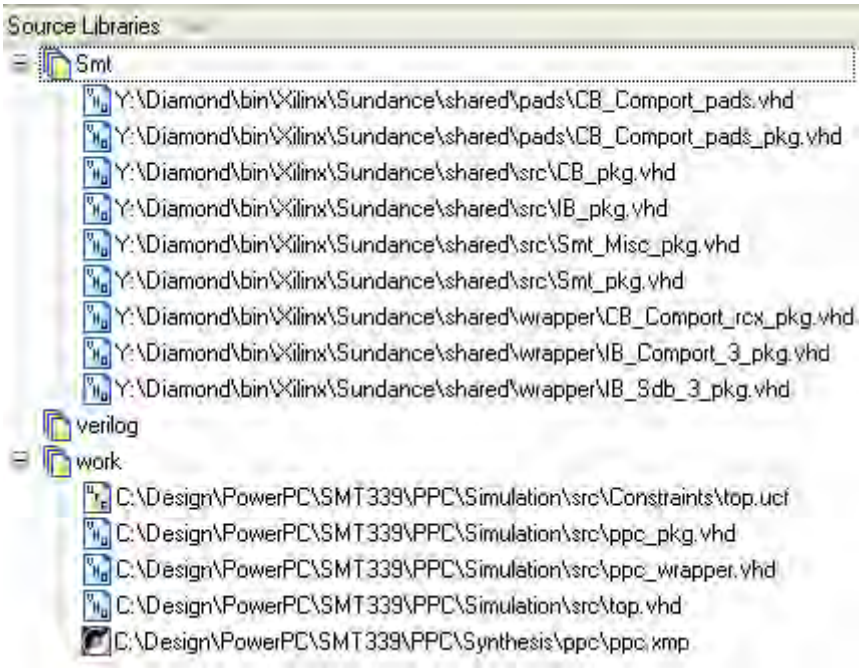

Figure 31 SMT library

Into VHDL files, this new library may be used thanks to the following sentence: library Smt. Here was every library and package defined in this design.

```
-- (c) sundance multiprocessor technology
Library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

library Smt;
use Smt.Smt_pkg.all;
use Smt.Smt_misc_pkg.all;
use Smt.IB_pkg.all;
use Smt.CB_pkg.all;
use Smt.cb_comport_rcx_pkg.all;
use Smt.CB_Comport_Pads_pkg.all;
use Smt.IB_Comport_3_pkg.all;
use work.ppc_pkg.all;
```

Figure 32 libraries used

By adding ppc_wrapper.vhd and top.vhd to the Project Navigator project the hierarchy was updated as shown in **Figure 34**.

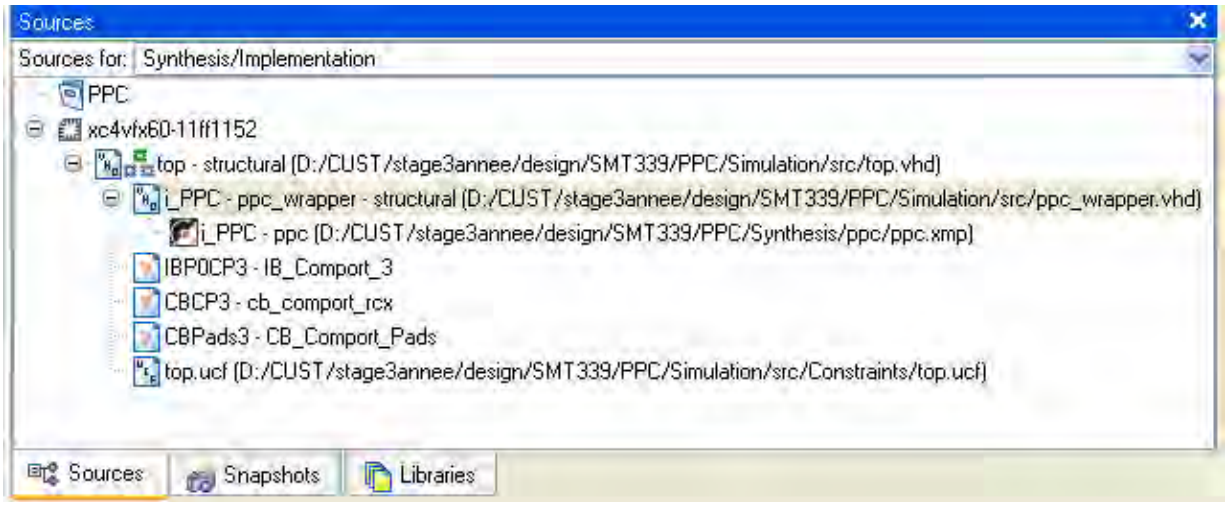

Figure 33 project hierarchy into ISE

Constraints have to be implemented onto the comport link. They are defined in the ucf file. The hierarchy has changed now that the EDK system is instantiated inside the ppc_wrapper module so the PPC reset pins are no longer available in the top level module. You need to add a */ in front of signals C405RSTCORERESETREQ, C405RSTCHIPRESETREQ, C405RSTSYSRESETREQ. That allows the tools to “wildcard” the hierarchy preceding the PPC reset pins.

In order to implement the Comport according to the protocol, several constraints must be defined such as timing constraint and pin assignments. These constraints are always required when the Comport link is implemented into a design. This ucf file also included the Led, clk and reset pin assignments.

After this step, the design was ready to be implemented. When the bitstream was generated by ISE, it was called XPS to generate the VHDL file that defined the processor system, the NGC files and the finally bitstream. Every step of the design flow can be checked thanks to the following windows

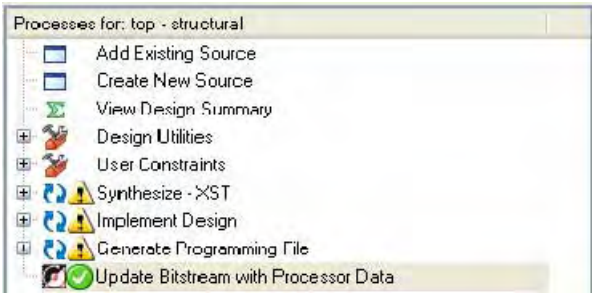

Figure 34 implementation processes into ISE

This step allowed users to check the resources required by the IP core. The ppc2firm block required 78 slices. This block included only the IPIF because the com-port was not implemented into this block.
The following synthesis options have been chose such as optimisation strategy, perform timing-driving packaging and placement that improved place and route into the FPGA. Yet, the full design used 2 percents of slices. After bitstream generation, the software design must be defined.

1.5.2.4 Software implementation

A software program must be developed to use the Comport link. This link is controlled by four registers:

- Event Control Register (CP_ECR)
- Global Status Register (CP_GSR)
- Control and Status Register (CP_CSR)
- Data Register (CP_DAT)

Using these registers, it is possible to transfer data using a Comport with the following techniques:

- Polling
- Interrupts
- EDMA

But the block IPIF instantiated did not implement interrupt source and develop an example with the EDMA would have been too difficult. So, this application note polls the registers of the comport.
The program implemented was easy. Only one com-port was designed, the PowerPC received data from the com-port and displayed it.
This program used the function “cpread” defined in the "SmtTim.h" file. Sundance provides this package to its customers to make the Comport interface easier.
The function "cpread" polled the status and control register of the Comport (bit 26: IFE). And, it read back one word to the data register of the Comport 3 when there is one word available to read in the data register.

EDK provided libraries to use the custom peripheral implemented. Here was the function provided by EDK to read a value from Comport:

XIo_In32((BaseAddress) + (RegOffset))

BaseAddress was the base address of the PPC2FIR device and RegOffset was the register offset from the base to write to. This function took as parameter the address register and not the comport number such as the Sundance's macros.

To simplify this implementation for the Sundance's customers, a function "cpread" and "cpwrite" was defined. They are defined in the header file "SmtTim.h". If these functions are used with a PowerPC, only the variable SMTPPC should be defined in the main program.

Figure 36 explains the improvements in the "SmtTim.h" file.

```
#if defined SMTPPC

#if defined __BASE

#define _FPGA(X) (volatile unsigned int *)((__BASE)+(X))

#define CPN_DAT(N)                _FPGA(CPN_DAT_OFFSET(N)) /* comport data register */
#define CPN_CSR(N)                _FPGA(CPN_CSR_OFFSET(N)) /* comport control and status register*/
.
.
.

#else  // SMTPPC

#define CPN_DAT(N)               XIo_In32( _FPGA(CPN_DAT_OFFSET(N)))
#define CPN_CSR(N)               XIo_In32(_FPGA(CPN_CSR_OFFSET(N)) )/* comport control and status register */
.
.
.
#endif // __BASE

#endif // SMTPPC
```

Figure 35 update SmtTim.h file

In which time the CPN_DAT or CPN_CSR macros are called by the main program, if SMTPPC was defined, the function XIo_In32() provided by EDK is executed to read a data from com-port.

Figure 37 shows the user application with EDK.

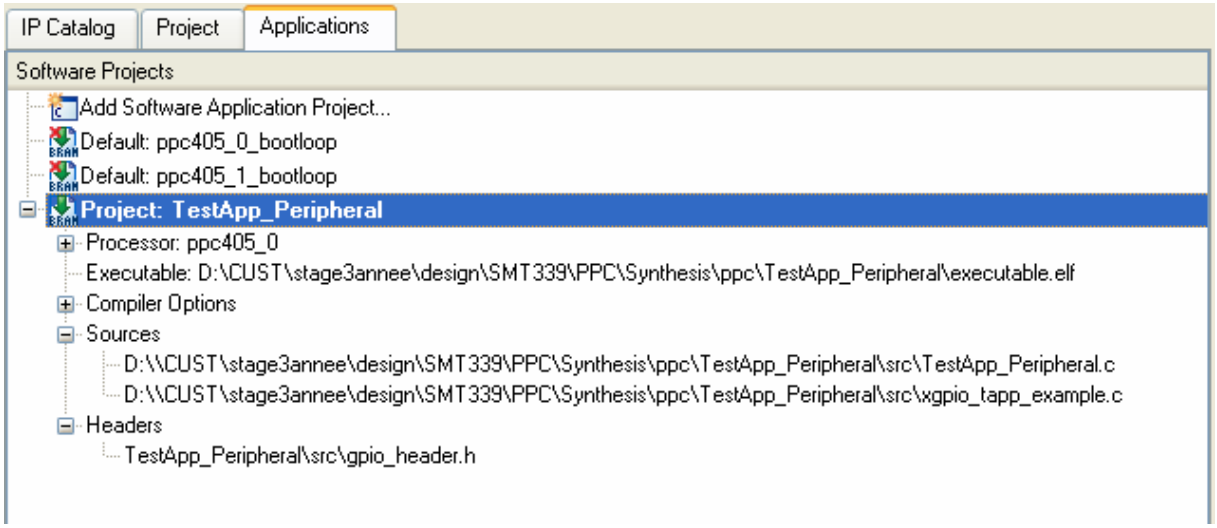

Figure 36 user application

1.5.2.5 Test and validation

For this application, another board was required to test the comport link. The same method and the same tools were used in this application and the first design.

Testes were split in two parts. The first part tests the comport link between the SMT339 and the 319 with no processing. It means the program described above was not used in this step. Only the comport registers were tested in this first parts. The second part ran the polling program. The following figure shows the scheme used in the first step.

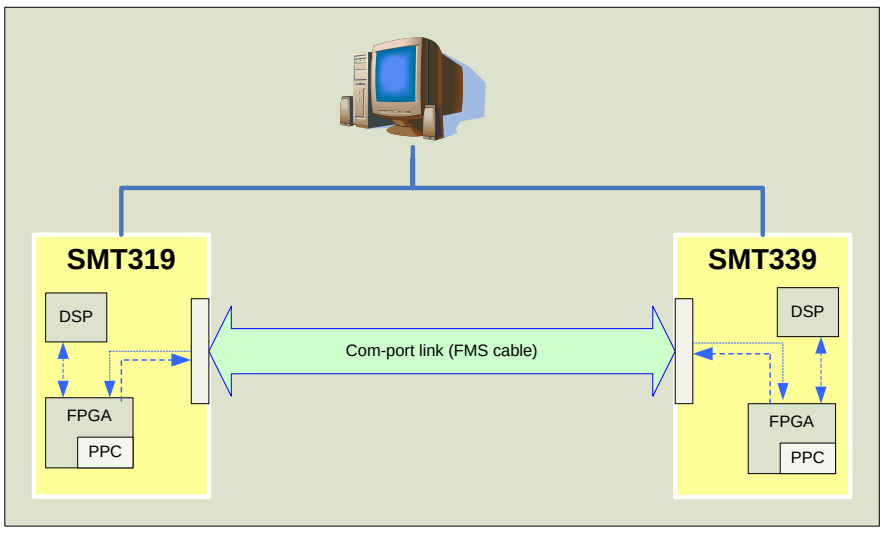

Figure 37 test scheme for the com-port link

As shown this figure the SMT319 was used to test this design. Only the DSP on this board was used. It allowed user to transmit data from SMT 319 to the SMT339 via Comport.
For this test, you need to connect physically the Comport 1 of the module 1 (SMT319) to the Comport 3 of the module 2 (SMT339).

The SMT319 module was used to transmit and check data receive by the module 2. For that, Code Composer Studio (CCS) was used to transfer a data to the PowerPC via the Comport link. The data received by the PowerPC could be checked thanks to the XMD debugger.

The base address of the Comport 1 is reauired to check the Comport registers in CCS. It is available in the user guide. For the SMT319, the base address of the Comport 1 is 0x90008000. And for the SMT339, the base address of the comport 3 is 0x70018000.
Figure 39 shows an example sending a data from the module 1 with CCS to the module 2. A data “1” was sent onto the comport 1 (address 0x90008000). The XMD memory windows displayed the data received by the PowerPC via the Comport 3 link.

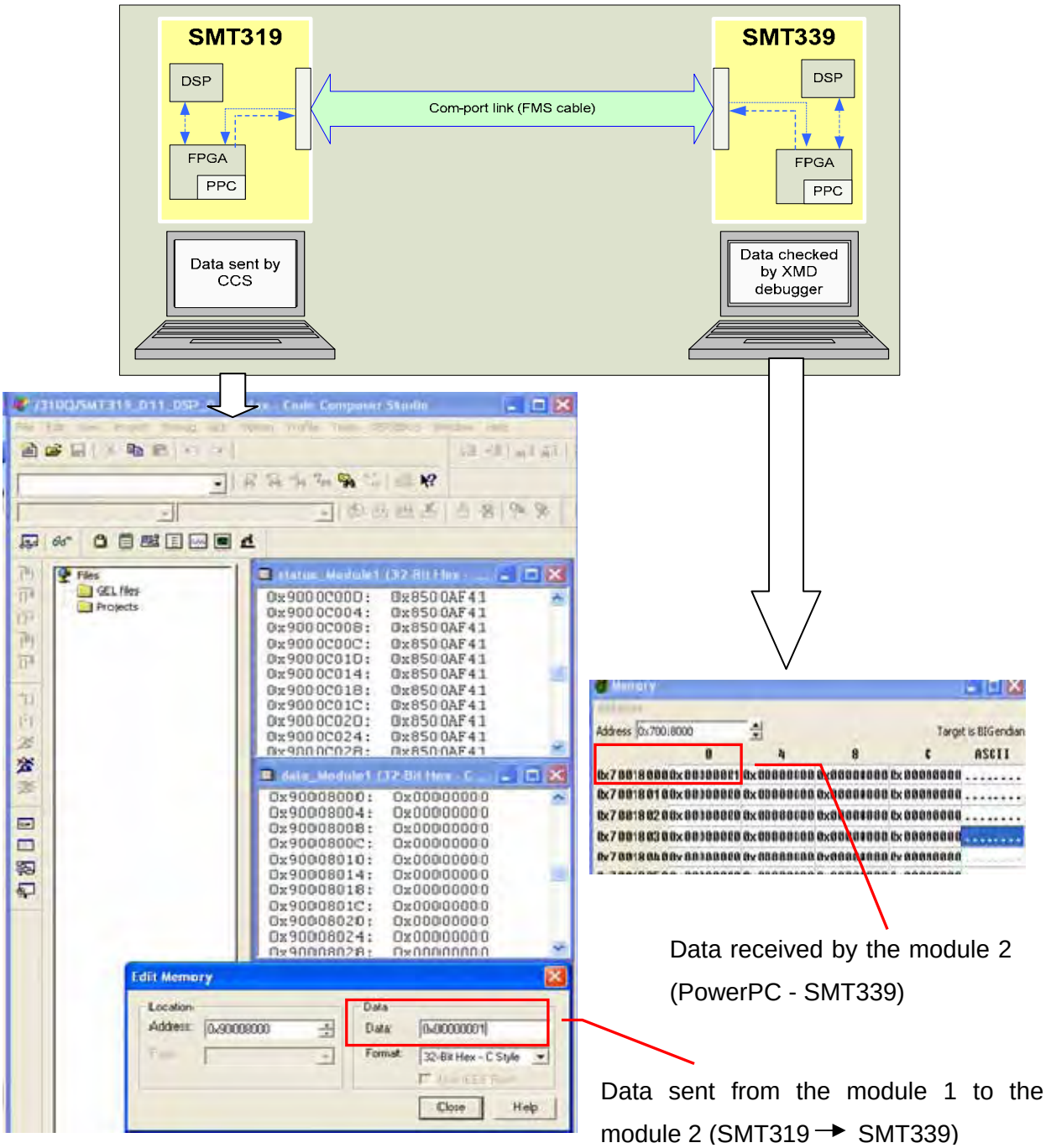

Figure 38 test com-port link CCS - XMD debugger

With this method, you are able to check every comport register. You can improve this test with the polling application.

The second part of this test uses polling application described in the previous section.

The value read from the comport 3, by the PowerPC, could be checked with the variable “read” in the local variable widows. **Figure 40** shows an application with the polling program.

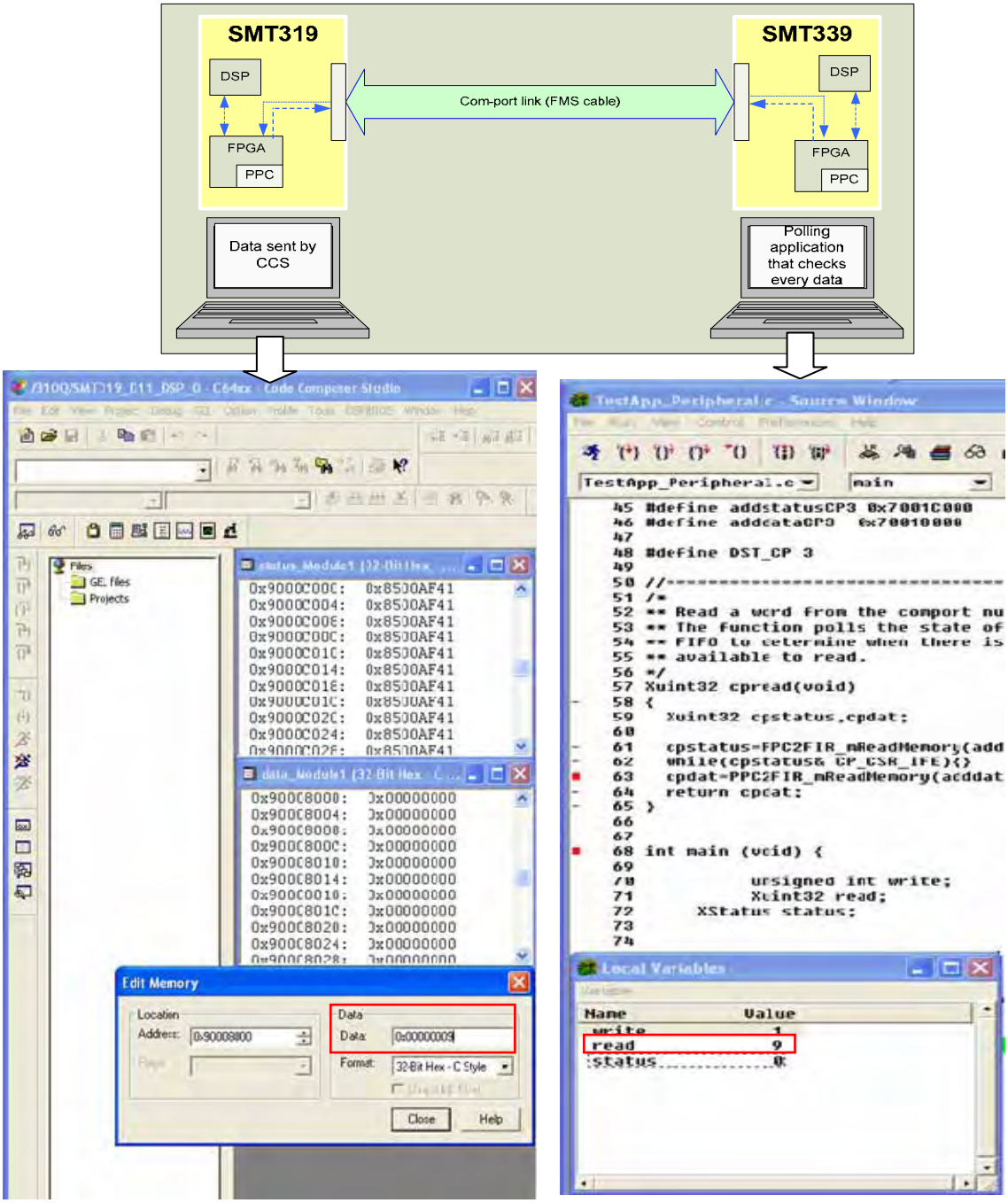

Figure 39 polling example

Application note

With this examples, the PowerPC can communicate with another processor according to the Sundance resources. The link implemented can sustain a data rate of about 20MB/s according to Sundance's requirements. PowerPC and its peripherals run at 100MHZ.

1.5.3 External ZBT memory controller

1.5.3.1 Overview

For the third design, an external ZBT memory controller was implemented to the PowerPC with the first design by beginning. This controller was chosen in the Xilinx's IP cores library. The design flow was generated by EDK. Both controllers: OPB EMC and PLB EMC were implemented in two separated design in order to provide several hardware configurations for Sundance's customers. A design with the two ZBT banks is also available.
In this design, ZBT memory was running at 100MHz like every peripheral connected to the PowerPC. The data rate expected was 40MB/s between the PowerPC and the ZBT memory.

1.5.3.2 Constraints

Every design was implemented onto the SMT339. This board provides two ZBT SRAM memory banks directly connected to FPGA pins. The PLB EMC memory controller provided by Xilinx was implemented. It is available is the IP Catalog tab. The IP Catalog tab shows all of the IP that are available to use in the EDK project.

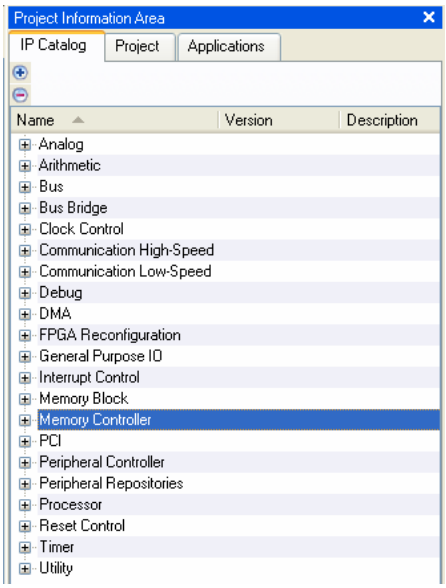

Figure 40 IP catalog

The PLB_EMC controller is available in the Memory controller directory. This module supports data transfers between the Processor Local Bus (PLB) and external synchronous and asynchronous memory devices.

The PLB EMC provides an interface between the PLB and one to four external banks of memory components.

The EMC supports PLB data bus widths of 8, 16, 32 & 64 bits, and memory subsystem widths of 8,16, 32 & 64 bits. It supports the PLB V3.4 byte enable architecture. Any access size up to the width of the PLB data bus is permitted. This kind of controller can perform data-width matching meaning when the width of the memory is less than the width of the PLB, multiple memory cycles are performed to transfer the data width of the bus if data-width matching has been enabled for that memory bank.

The PLB EMC provides basic read/write control signals and the ability to configure the access times for read, write, and recovery times when switching from read to write or write to read.

Figure 42 illustrates the top Level block diagram of the PLB EMC.

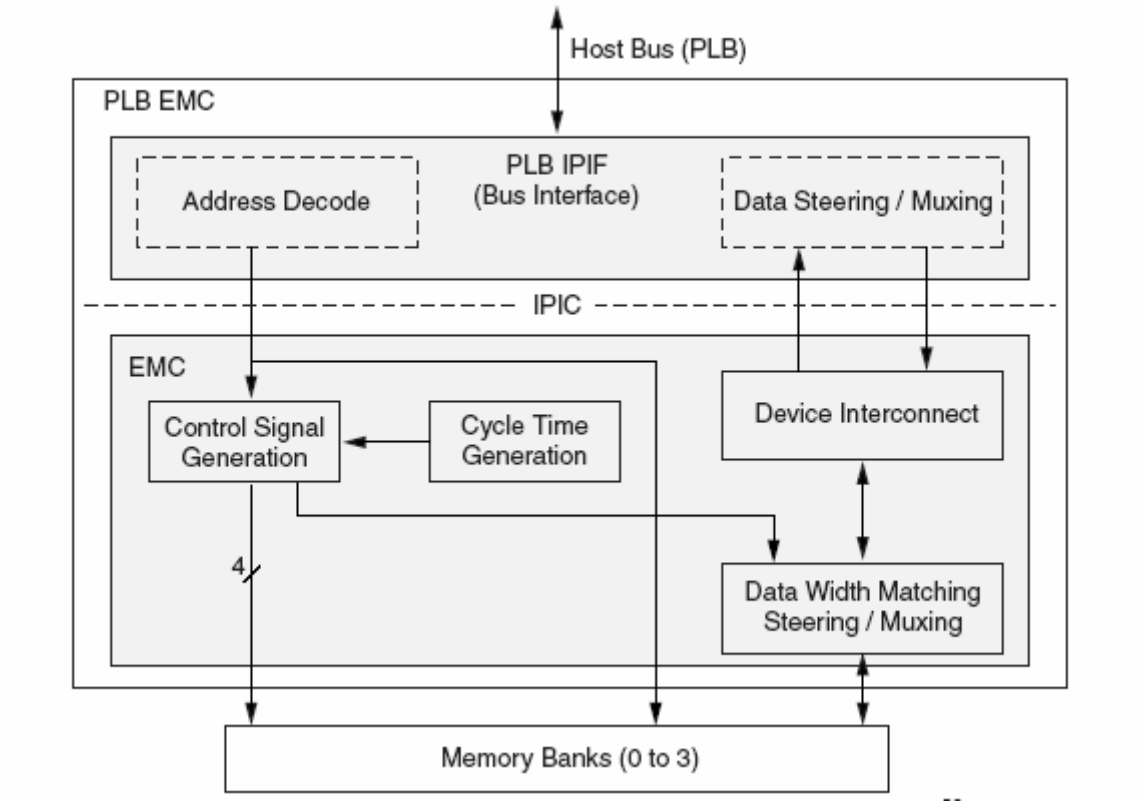

Figure 41 PLB top-level block diagram

The PLB EMC supports up to 4 banks memory. Each bank of memory has its own independent base address and address range. The address range of a bank of memory is restricted to be a power of 2. If the desired address range is represented by 2*n*, then the *n* least significant bits of the base address must be 0. For example, a memory bank with an addressable range of 4Mbytes (2^{22}) could have a base address of 0x00400000 and a high address of 0x7FFFFF. In this design, base address of the PLB EMC was 0x00000000.

Memory can be accessed through the EMC as one of four types: byte (8 bits), half word (2 bytes), word (4 bytes), and double word (8 bytes). From the point of view of the PLB, data is organized as big-endian.

The PLB EMC does not provide a clock output to any synchronous memories. The PLB clock should be routed through an output buffer to provide the clock to the synchronous memories. To synchronise the synchronous memory clock to the internal FPGA clock, the FPGA system design should include a DCM external to the PLB EMC core that uses the synchronous memory clock input as the feedback clock as shown in **Figure 43**.

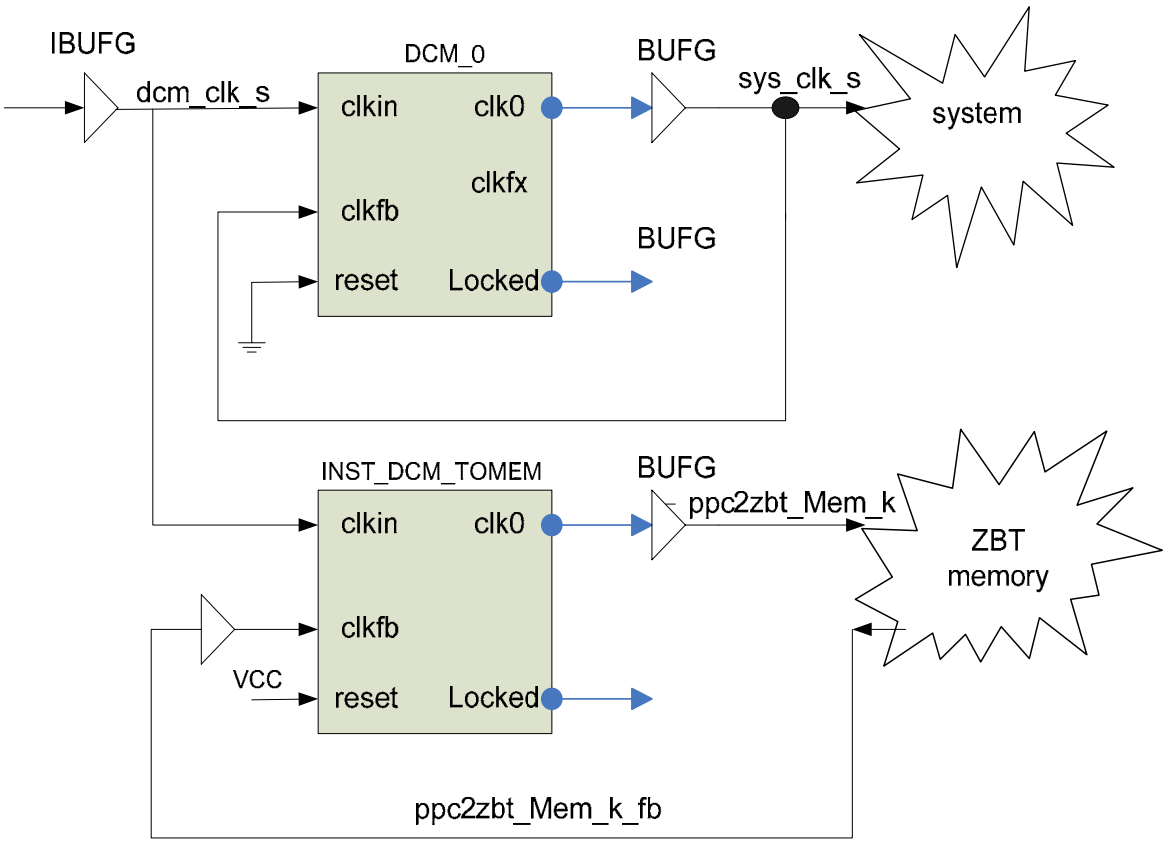

Figure 42 synchronous memory clocked by FPGA output with feedback

The three primary considerations for connecting the controller to memory devices were the width of the PLB data bus, the width of the memory subsystem, and the number of memory devices used. The width of the memory subsystem was simply the maximum width of data that was able to be read from or written to the memory subsystem.

The data and address signals at the memory controller are labelled with big-endian bit labelling (for example, D(0:31), D(0) is the MSB), whereas most memory devices are either Endian agnostic (they can be connected either way) or little-endian D(31:0) with D(31) as the MSB.

Most asynchronous memory devices only use Mem_CEN, while most synchronous memory devices use both Mem_CEN and Mem_CE. Mem_CEN is a function of the address decode while Mem_CE is a function of the state machine logic.

The following **Table 12** shows the correct mapping of memory controller pins to memory device pins. Only one ZBT bank was implemented in this design but all of them were tested.

EMC signals (MSB LSB)	Memory device signal (MSB LSB)	Comment
ppc2zbt_Mem_DQ_pin (0 : 15)	DQ(15 : 0)	Data Input and Output pin
ppc2zbt_Mem_Mem_A_pin (0 : 20)	A(20:0)	Address
ppc2zbt_Mem_BEN_pin (0 to 1)	BW(1:0)	Byte write inputs
ppc2zbt_Mem_CEN_pin (0 to 0)	/CS1	Chip select (/CS1)
ppc2zbt_Mem_OEN_pin (0 to 0)	/OE	Output enable
ppc2zbt_Mem_CE_pin	CS2	Chip select (CS2)
ppc2zbt_Mem_ADV_LDN_pin	ADV	Address advance
ppc2zbt_Mem_LBON_pin	/LBO	Burst mode control
ppc2zbt_Mem_WEN_pin	/WE	Read/Write control input
ppc2zbt_Mem_mem_Zz_pin	ZZ	Power sleep mode
ppc2zbt_Mem_CE2_pin	/CS2	Chip select (/CS2)
ppc2zbt_Mem_CKEN_pin	/CKE	Clock enable

Table 11 connexion between EMC and ZBT memory

A timing constraint should be placed on the system clock, setting the frequency to meet the bus timing requirements. No external pullups/pulldowns were available on the MEM_DQ signals onto the board, so pullup or pulldown resistors in the UCF file must be specified.

1.5.3.3 Hardware implementation

One of the key advantages of using the Xilinx IP core is to have a design that only utilizes the resources required by the system and runs at the best possible performance.

Before testing the external memory controller, certain feature must be configured such as the memory data width, the memory address width, the base address, the external port connexion.

Figure 44 shows every peripheral implemented in this design. The PLB_EMC called "ZBT_2Mx16bit" was connected onto the PLB bus and two others blocks were added to this XPS project: busAddr_split_0 and INST_DCM_TOMEM.

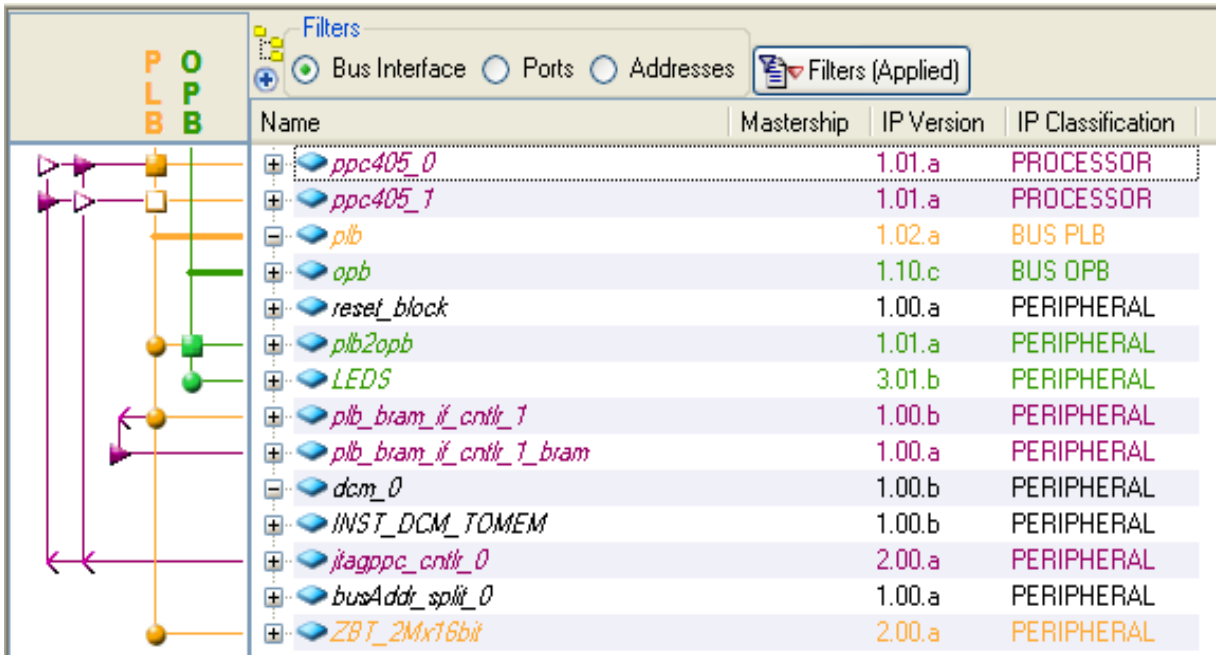

Figure 43 system assembly view for ZBT design

Only one bank was enabled (C_NUM_BANKS_MEM=1) in this external memory controller with a maximum width of the memory device equal to 16 bit (C_MAX_MEM_WIDTH=16). This example did not include support for PLB burst and cacheline transactions. The data-width matching option was checked in order to execute multiple memory access cycles to match width of Memory Bank 0 data bus to PLB data bus. The width of memory Bank0 data bus was set to 15 bit.

The ZBT memory provided a width of data bus of 18 bit but only 15 bit was used in this example. Finally, clock period was defined to 10000ps.
A DCM was configured according to the clock management **Figure 44**. This Digital Clock Manager DCM was called “INST_DCM_TOMEM”.
This DCM provided an output clock called “ppc2zbt_Mem_k” that was running at 100MHz. This signal was linked to ZBT clock. The feedback clock input was connected to the synchronous memory clock that was provided by the memory” ppc2zbt_Mem_k_fb”.

The EMC provided width of memory Bank0 address bus of 32 bit. But the ZBT memory had only 21 bit address inputs. The **Table 13** .

EMC signals(MSB LSB)	Memory device signal (MSB LSB)
ppc2zbt_Mem_Mem_A(10 to 30)	ppc2zbt_Mem_Mem_A(20 to 0)

Table 12 address bus connexion

A Utility Bus Split core provided by Xilinx was used. It allowed user to split a bus into smaller buses. This component was called “busAddr_split_0”.
This core took one input bus and splits it into two output buses and serves as glue logic between peripherals. **Figure 45** illustrates the block diagram of the “busAddr_split_0” component.

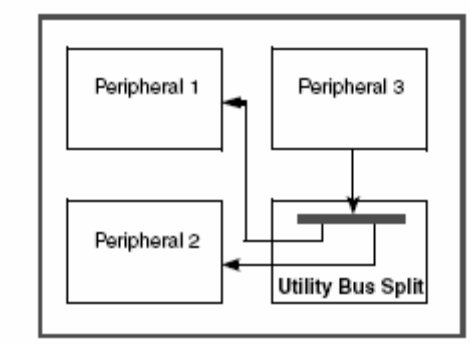

Figure 44 utility bus split in a system

This block took the 32-bit signal from the EMC (ppc2zbt_Mem_Mem_A_split) and provided a 21-bit signal to external memory (ppc2zbt_Mem_Mem_A).

The PowerPC was connected to the ZBT SRAM saccording to the following **Table 14**.

EMC signals(MSB LSB)	Memory device signal (MSB LSB)
ppc2zbt_Mem_DQ(0 to 15)	pppc2zbt_Mem_DQ(15 to 0)

Table 13 data bus connexion

So, the most significant bit of the memory device (pppc2zbt_Mem_DQ(17 to 16)) have not been connected. **Figure 46** illustrates the system assembly view of this design.

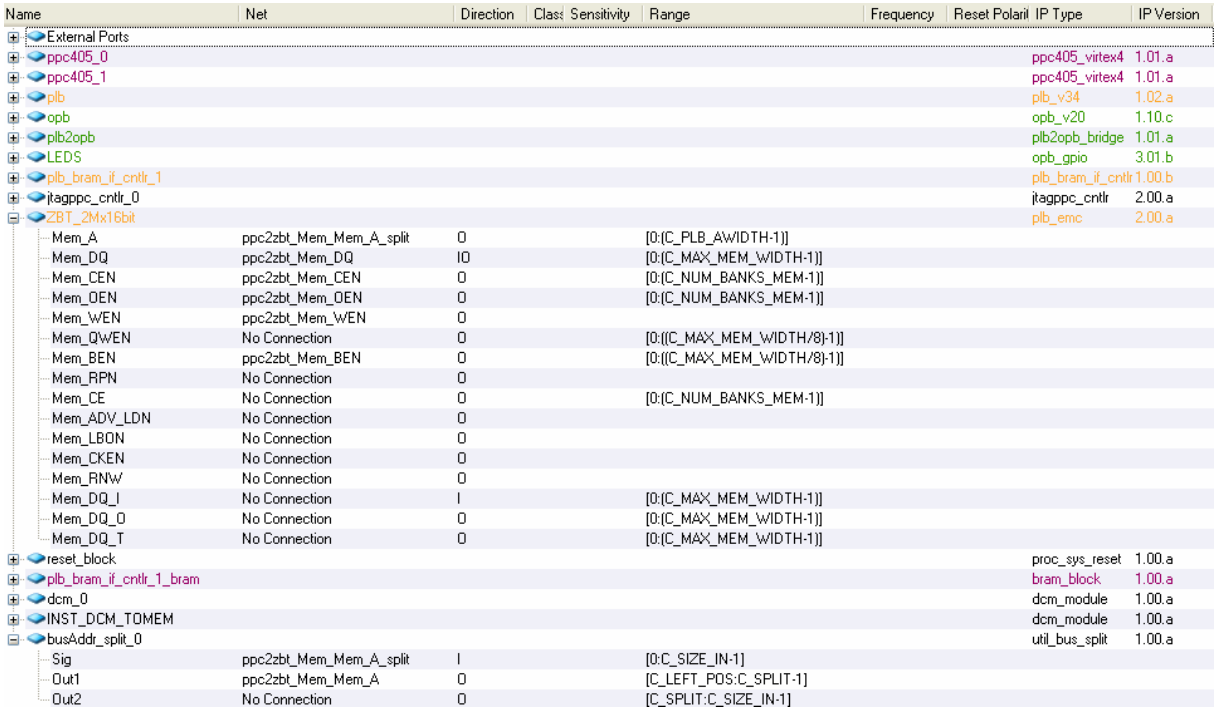

Figure 45 system assembly view for the ZBT design

Table 15 describes the memory map of this application.

Device	Address		Size	Comment
	Min	Max		
PLB_BRAM_CNTLR	0xFFFFC000	0xFFFFFFFF	16K	PLB Memory
OPB_GPIO	0x10000000	0x100003FF	1K	LED output
PLB2OPB	0x10000000	0x100003FF	1K	Bridge
ZBT_2Mx16bit	0x00000000	0x003FFFFF	4Mb	PLB_EMC

Table 14 memory map for the ZBT design

Figure 47 describes the hardware implemented. The block PLB EMC is available onto the PLB bus.

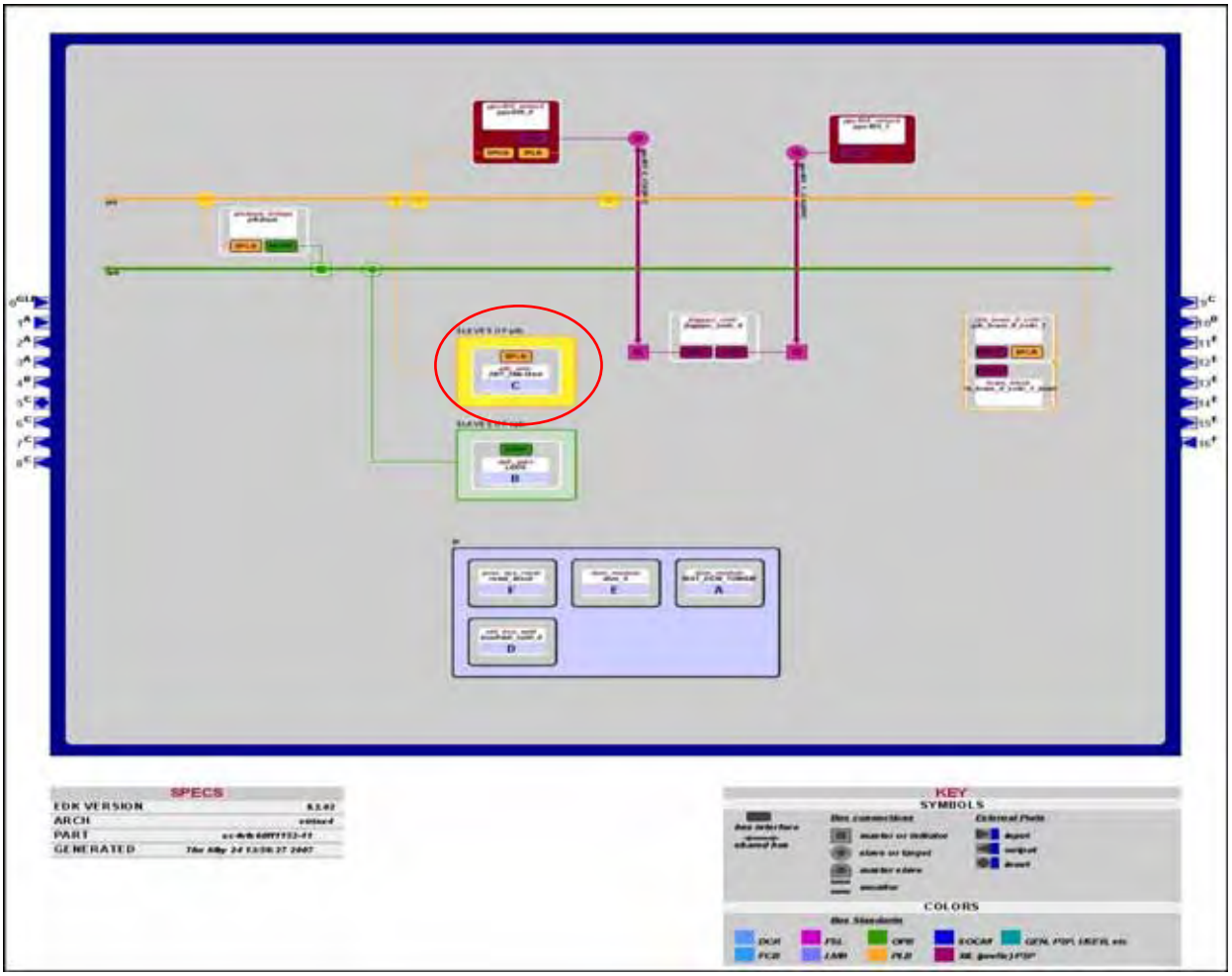

Figure 46 block diagram ZBT design

With this configuration, the PLB_EMC required 370 slices in spite of the PLB_ EMC block included the external ZBT memory controller and the PLB IPIF as shown the **Figure 15**.

The next section describes the software application.

1.5.3.4 Software implementation

The configuration of the BSP was the same that defined in the first design. Software application was implemented to test the external memory.

The Library Generator tool, Libgen, generated several folders which contain C header files needed by drivers, libraries and MAKE files needed to compile the OSs, libraries, and drivers.

In this project, the include file xutil.h was also created through Libgen. This file contained function prototypes such as memory test function. They were used to test the ZBT memory. Here are three function prototypes used in this application.

```
XStatus XUtil_MemoryTest32(Xuint32 *Addr, Xuint32 Words, Xuint32 Pattern,Xuint8 Subtest);
XStatus XUtil_MemoryTest16(Xuint16 *Addr, Xuint32 Words, Xuint16 Pattern, Xuint8 Subtest);
XStatus XUtil_MemoryTest8(Xuint8 *Addr, Xuint32 Words, Xuint8 Pattern, Xuint8 Subtest);
```

Figure 47 Xutil_memtest prototypes

These functions allowed users to access the memory as one of three types: byte (XUtil_MemoryTest8), half word (XUtil_MemoryTest16), word (XUtil_MemoryTest32). Every function took four parameters. “Addr” was a pointer to the region of memory to be tested. The “Words” parameter was the length of the block. The next one was the constant used for the constant pattern test and “Subtest” parameter was the test selected. A subset of the memory tests could be selected or all of the tests could be run in order. The detailed description of the subtest is available **Table 16**.

If there was an error detected by a subtest, the test stopped and the failure code was returned. Further tests were not run even if all of the tests were selected.

Type Definitions	description of the subtest
XUT_ALLMEMTESTS	Ran all of the following tests
XUT_INCREMENT	Incrementing Value Test. This test starts at 'XUT_MEMTEST_INIT_VALUE' and uses the incrementing value as the test value for memory.
XUT_WALKONES	Walking Ones Test. This test uses a walking '1' as the test value for memory. location 1 = 0x00000001 location 2 = 0x00000002 ...
XUT_WALKZEROS	Walking Zero's Test. This test uses the inverse value of the walking ones test as the test value for memory. location 1 = 0xFFFFFFFE location 2 = 0xFFFFFFFD ...
XUT_INVERSEADDR	Inverse Address Test. This test uses the inverse of the address of the location under test as the test value for memory.
XUT_FIXEDPATTERN	Fixed Pattern Test. This test uses the provided patters as the test value for memory. If zero is provided as the pattern the test uses '0xDEADBEEF".

Table 15 memory test descrption

The software application specified in Application tab. The TestApp_pheripheral.c file used Xilinx's libraries to test the memory and the GPIO peripherals. The file "xgpio_tapp_example.c" contained an example for using GPIO hardware and driver. The source code "testApp_ZBT.c" used GPIO hardware and driver to display the result of this test on the GPIO LED.

Figure 49 shows the software application selected for this implementation.

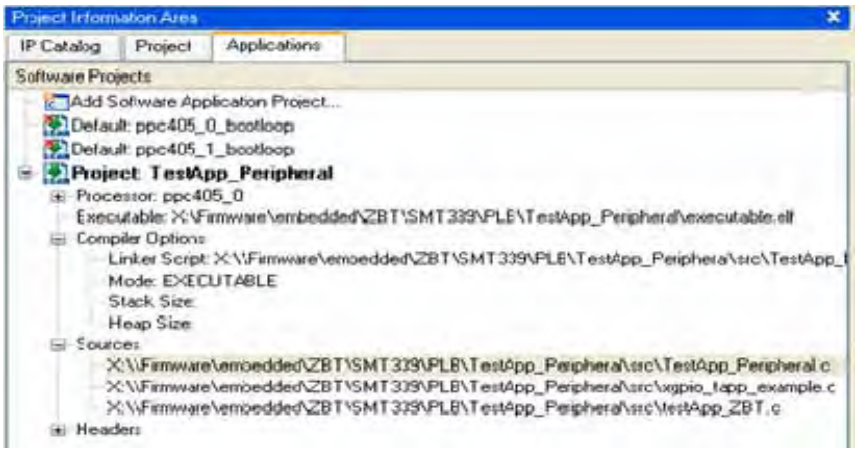

Figure 48 software application design ZBT

Next step was the validation of this implementation using the application code described above.

1.5.3.5 Test and validation

Before running the application, every parameter of function prototypes have to be initialised. The “Addr” parameter to the base address of the ZBT_2Mx16bit controller. All subtests can be run by setting “subtest” parameter to XUT_ALLMEMTESTS type. The “Words” parameter defined length of the memory. Thus, this element is different for every function. One of three types could be performed: 8bitx4MB (Words=4x1024x1024), 16bitx2MB (Words=2x1024x1024) and 32bitx1MB (Words=1x1024x1024). Finally, the “pattern” parameter was initialised to 0xA5.

This test was performed in in debug mode as shown the **Figure 50**.

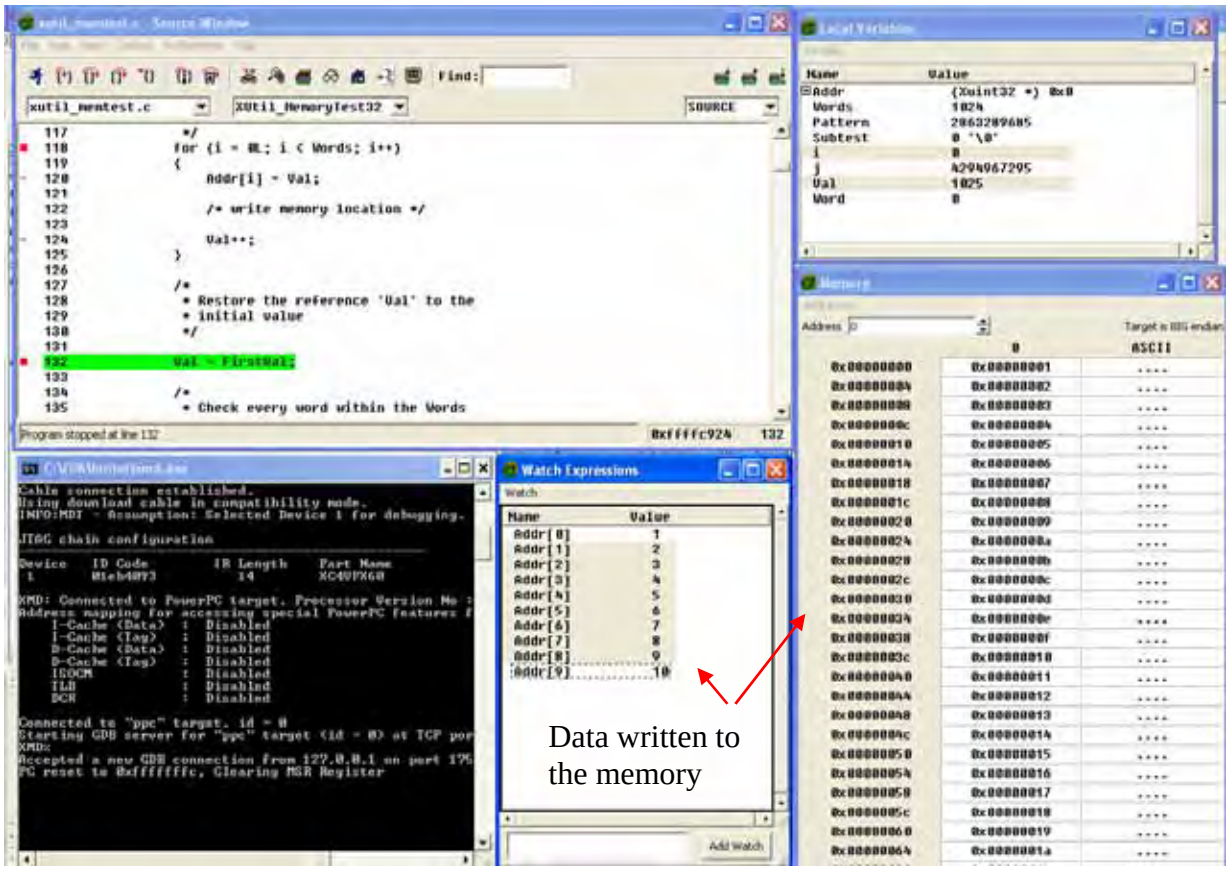

Figure 49 XMD debugger - ZBT design

An application is also available in stand alone. The following paragraph illustrates an example implementing a memory cache, com-port link and increasing the PowerPC clock frequency.

1.5.4 Global implementation

The data on-chip memory (DOCM), the instruction on-chip memory (IOCM), data and instruction cache and internal memory were connected to the PowerPC according the requirements. The PowerPC clock frequency was increased as well. In order to provide a full firmware, two Comport links were added to this design.

1.5.4.1 Hardware implementation

According to the requirements, different kinds of memories were implemeted. These memories used a specific communication resource.

Figure 51 shows every peripheral implemented in this design. Both of ZBT memories were implemented in this design.

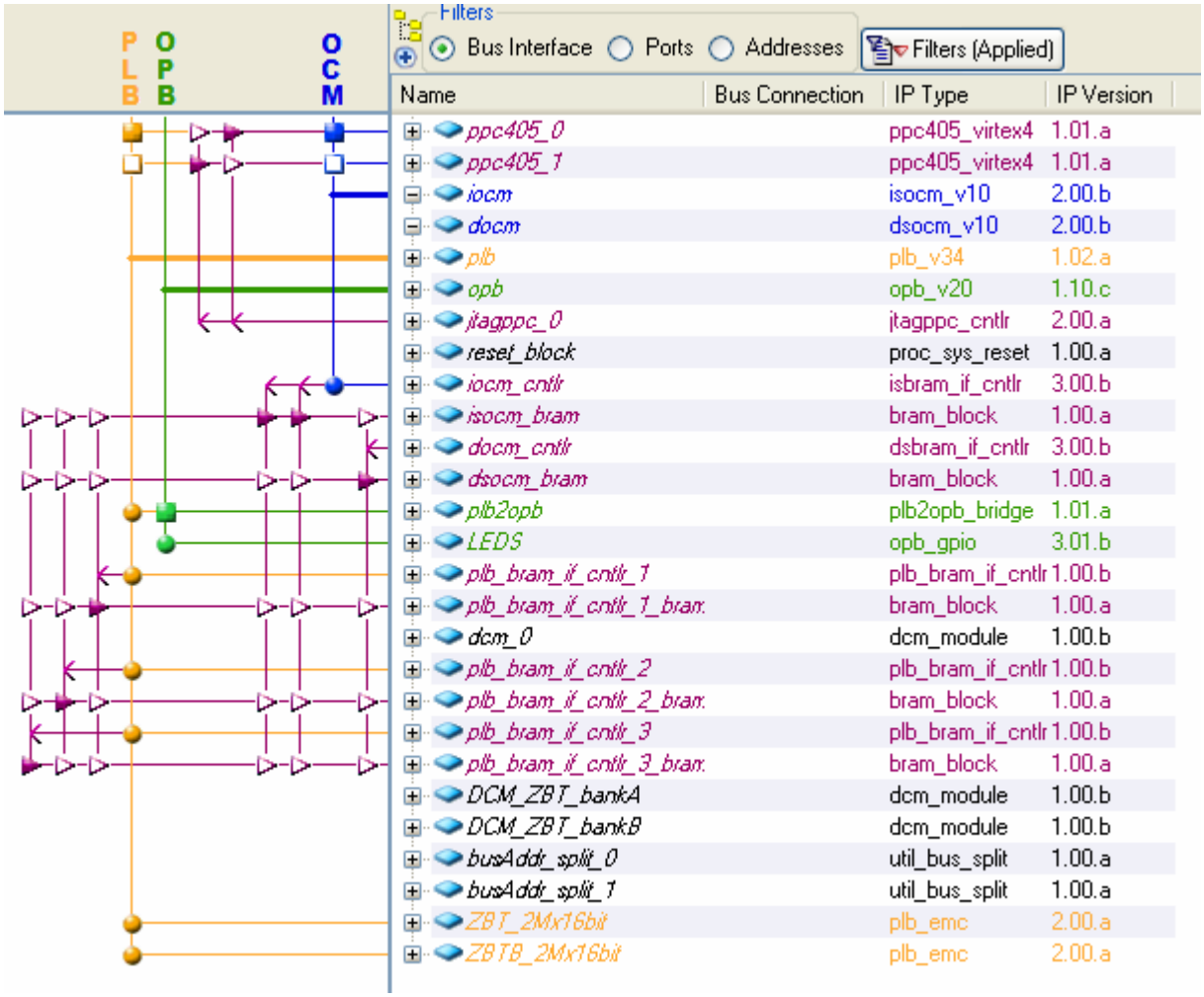

Figure 50 system assembly view - two ZBT memory banks

The docm_cntlr connected BRAM block to the data side on chip memory (DSOCM) bus in PowerPC 405 based embedded systems. The DOSCM block was a data-side On-chip Memory (OCM) bus interconnect core. The core connected the PowerPC data-side OCM interface to data-side OCM BRAM controller (docm_cntlr). The same architecture was implemented for the instruction-side on chip memory. The iocm_cntlr connected BRAM block to the instruction side on chip memory (ISOCM) bus. The IOCM component was an instruction side On-chip Memory bus interconnect core.

In this design the PowerPC was running at 300MHz and every peripheral was clocked at 100MHz. **Figure 52** shows the clock management. The “DCM_0” component generated two clocks: the system clock (sys_clk_s) at 100 MHz and the processing clock (proc_clk_s) was running at 300 MHz.

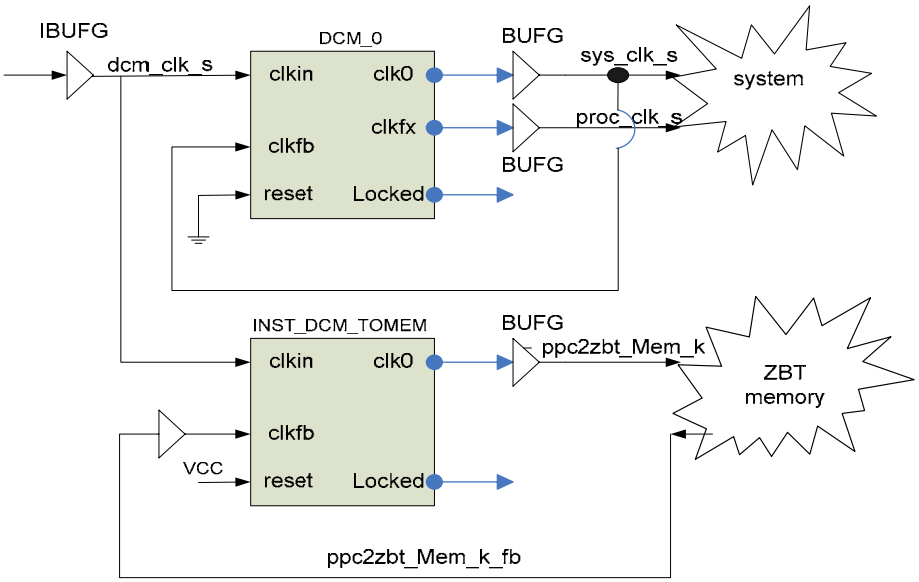

Figure 51 clock management global design

Table 18 describes the memory map of this design. In this design, 108Kb on chip memory were connected to the PowerPC. This processor also supported data transfer between the PLB bus and the 8Mb of external ZBT memory. The memory map shows that the internal memory and the external memory have consecutive addresses according to DIAMOND specification.

Device	Address		Size	Comment
	Min	Max		
PLB_BRAM_CNTLR	0x00FF0000	0x00FFFFFF	64Kb	PLB Memory
OPB_GPIO	0x40000000	0x4000FFFF	64Kb	LED output
PLB2OPB	0x40000000	0x4000FFFF	1Kb	Bridge
ZBT_2Mx16bit	0x01000000	0x013FFFFF	4Mb	PLB_EMC
ZBTB_2Mx16bit	0x01400000	0x017FFFFF	4Mb	PLB_EMC
IOCM_CNTLR	0xFFFFc000	0xFFFFFFFF	16Kb	ISOCM
DOCM_CNTLR	0xA0008000	0xA000BFFF	16Kb	DSOCM
PLB_BRAM_CNTLR_2	0x00000000	0x00000FFF	4Kb	Dcache
PLB_BRAM_CNTLR_3	0x00001000	0x00001FFF	4Kb	Icache

Table 16 memory map global design

Figure 53 describes the hardware implemented. Both external ZBT memory controllers were connected onto the PLB bus. Both memories connected onto the On-chip memory bus appear above the PowerPC. This Figure allowed you to check every memory connected onto the PLB bus

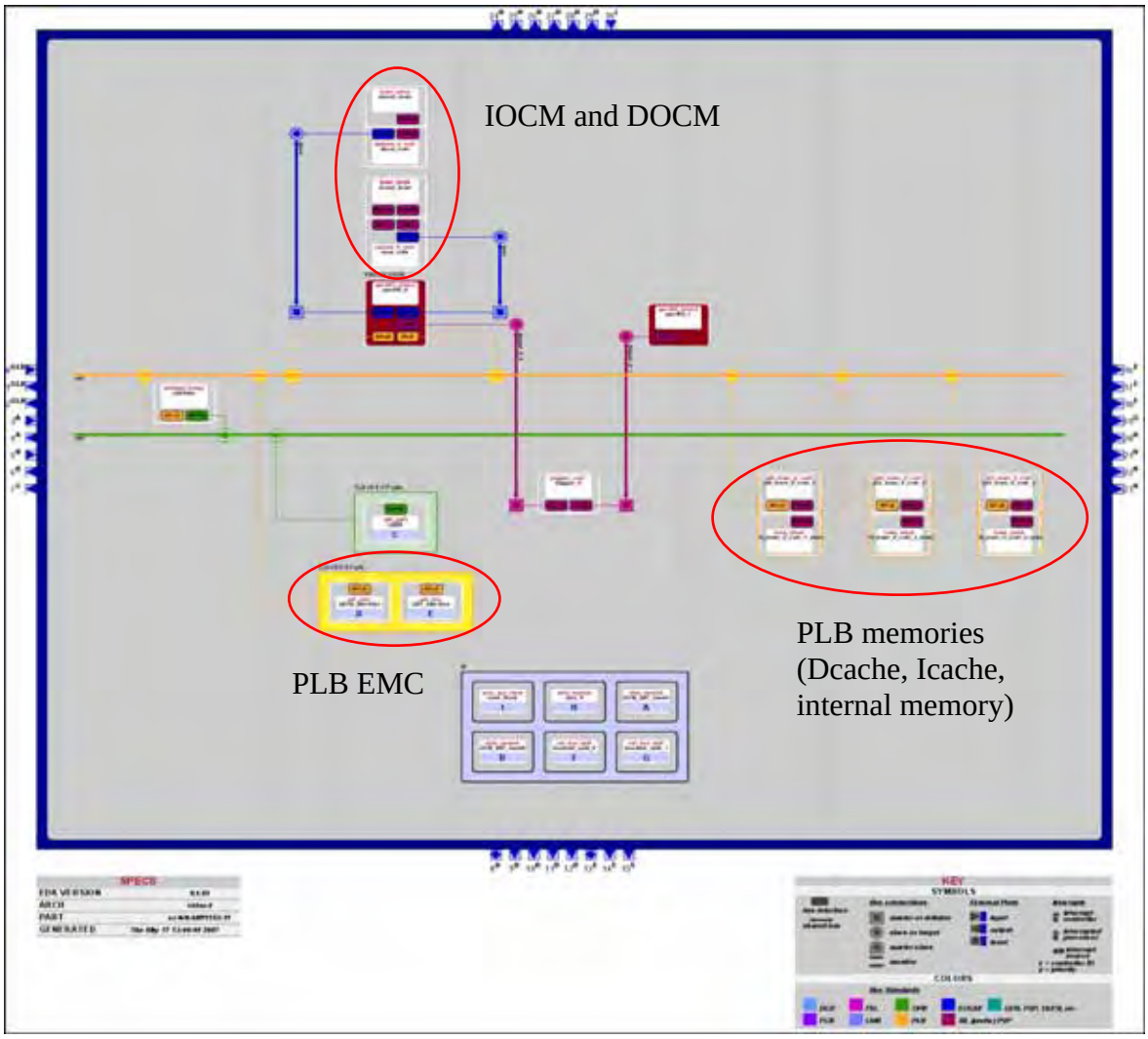

Figure 52 block diagram global design

This design used 5 percents of slices. More than 4 percents slices are used by the PLB controller and the PLB/OPB bridge. The design flow was generated by XPS.

1.5.4.2 Software implementation

The software application developed in the previous design was improved to test every memory.

In this application, every memory was tested singly and then all memories were tested in the same time as shown in **Figure 54**.

```
#include "xutil.h"
//(plb_bram_if_cntlr_1) - internal memory
#define MemorySize_I1 (XPAR_PLB_BRAM_IF_CNTLR_1_HIGHADDR - XPAR_PLB_BRAM_IF_CNTLR_1_BASEADDR)+1
//external Memory (ZBT_2Mx16bit) - external memory
#define MemorySize_EA (XPAR_ZBT_2MX16BIT_MEMO_HIGHADDR - XPAR_ZBT_2MX16BIT_MEMO_BASEADDR)+1
#define MemorySize_EB (XPAR_ZBT_2MX16BIT_MEMO_HIGHADDR - XPAR_ZBT_2MX16BIT_MEMO_BASEADDR)+1
//all of them
#define MemorySize (MemorySize_I1 + MemorySize_EA + MemorySize_EB)
//=====================================================

  // Testing internal and external Memory (plb_bram_if_cntlr_1 - ZBT_2Mx16bit - ZBTB_2Mx16bit) - external memory
  Size=MemorySize;
  status=XUtil_MemoryTest32((Xuint32*)XPAR_PLB_BRAM_IF_CNTLR_1_BASEADDR, (Size/4), 0xAAAA5555,XUT_ALLMEMTESTS);
 if(status==0)value=0x1; else value=0x0;
  status=XUtil_MemoryTest16((Xuint16*)XPAR_PLB_BRAM_IF_CNTLR_1_BASEADDR,(Size/2) , 0xAA55,XUT_ALLMEMTESTS);
  if(status==0)value=(0x2 | value); else value=value;
  status=XUtil_MemoryTest8((Xuint8*)XPAR_PLB_BRAM_IF_CNTLR_1_BASEADDR,(Size), 0xA5, XUT_ALLMEMTESTS);
  if(status==0)value=(0x4 | value); else value=value;
  status = displayLed(XPAR_LEDS_DEVICE_ID,3,value);
  value =0;
```

Figure 53 test internal - external memories

Another project implementing the Comport 2 to the PowerPC was created. The architecture was the same with the Comport 3 but just the top level into Xilinx ISE changed in order to reset this link to the output. Thus, the PowerPC can be connected to another processor via two comport links starting off in two different electrical states: Comport 3 in input and Comport 2 in output.

CONCLUSION

Sundance provides a design that implemented a PowerPC with different kind of memories such as data and instruction memories connected onto On-chip memory bus, data and instruction cache and internal memory interfaced to the PLB bus. Two separate ZBT memory banks were also connected to the PowerPC. This design contained an internal memory and external memories with consecutive addresses. The processor had more than 8MB of memories available.

This design allows users to interface the PowerPC to another processor such as a DSP. For this design, another board was required.

Several improvements could be done such as interrupt support and DMA in the user peripheral. These options can be added to the user peripheral in the IPIF service.

References

[1] Sundance Multiprocessor Technology Ltd, URL: *http://www.sundance.com*

[2] 3L DIAMOND, URL:*http://www.3l.com*

[3] Xilinx Inc, URL: http://www.xilinx.com

[4] Texas Instruments, URL: *http://focus.ti.com/dsp/docs/dsphome.tsp?sectionId=46*

[5] IBM – CoreConnect architecture, URL: *http://www-03.ibm.com/chips/products/coreconnect/*

[6] Samsung – ZBT memory, URL: *http://www.samsung.com/*

[7] Texas Instrument, URL: *http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html*

[8] Xilinx application notes, URL: *http://www.xilinx.com/xlnx/xweb*

Glossary

Acronym	Definition
BRAM	Block Random Access Memory
DCM	Digital clock Manager
DMA	Direct Access Memory
DSP	Digital Signal Processor
FIFO	First In First Out (memory)
FIRMWARE	Firmware is data that sets programmable logic into a state where it implements a particular hardware design. It starts as a textual description of an electronic circuit which is processed to produce a binary file.
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
Gb	Gigabits
GB	Gigabytes
ISE	Integrated System Environment
LSB/MSB	Less Significant Bit / Most Significant Bit
MIPS	Mega Instructions Per Second
MULTIPROCESSOR	A multiprocessor machine uses two or more CPUs for routine processing
NTSC	National Television Standard Committee
OEM	Original Equipment Manufacturer
OPB	On-chip Peripheral Bus
PAL	Phase Alternation Line
PCI	Peripheral Component Interconnect
PPC	PowerPC
RAM/ROM	Read Access Memory / Read Only Memory
RTOS	Real Time Operating Software
SDB	Sundance Digital Bus
SHB	Sundance High-speed Bus
SDRAM	Synchronous Dynamic Random Access Memory
SMT	Sundance Multiprocessor Technology
TI	Texas Instruments
TIM	Texas Instruments Module: a form factor board format
VHDL	Very High-speed integrated circuit Hardware Description Language
YCrCb	luminance, red chrominance, blue chrominance (video format)
ZBT	Zero Bus Turnaround