EMULATION OF ASIC BASED NETWORK PROCESSOR ON A FPGA PLATFORM

by

SHARADA VAJJA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

AUGUST 2006

ACKNOWLEDGEMENTS

I am grateful to my thesis advisor Dr. Wendell A. Davis for giving me an opportunity to work under his guidance and support throughout the course of my thesis. I really respect all the time that you spent going through my reports and giving your valuable suggestions.

I am also thankful to my committee members, Dr. Wei-Jen Lee and Dr. Enjun Xiao, for their support and their precious time.

I take this opportunity to thank my mentor at Intel, Mr. James Kann without whose guidance it would have been a nightmare to understand the humungous documents and complex tools used in the project. It has been a pleasure and honor to work with him.

I am also very grateful to my manager at Intel, Mr. Chandra Vora for giving me the opportunity to work in his department, Infrastructure Processor Division.

I highly appreciate the support of all other colleagues in my department and other co-interns and friends at Intel for their encouragement and support.

July 21, 2006

ABSTRACT

EMULATION OF ASIC BASED NETWORK PROCESSOR

ON A FPGA PLATFORM

Publication No. _____

Sharada Vajja, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Wendell A. Davis

Exponential increase of device sizes and complexity of Application specific ICs (ASICs) mandate the need for design verification to meet timing specifications and functionality requirements. Verification techniques are broadly implemented in three categories viz., simulation, verification using commercial emulators and Field Programmable Gate Array (FPGA) based prototyping.

An Intel architecture based network processor, to be taped out in the near future, is being emulated on FPGA based platform. The basic blocks of the Network Processor, for instance, the Memory Controller Hub (MCH) and I/O controller Hub (ICH) were synthesized to be configured on FPGAs. The work involved synthesizing

the design, partitioning it on multiple FPGAs and pin multiplexing of I/Os. Each sub unit of MCH and ICH were analyzed to resolve gated clocks not supported in FPGAs but widely used by ASIC designers for reducing power consumption.

Synplicity's Certify tool is used for partitioning, pin multiplexing and for the gated clock fixes. FPGA compatible block RAMs were generated with Xilinx Coregen. The design is being configured on Xilinx Virtex-4 FPGAs which provide high performance and high density boards for prototyping ASICs with millions of transistors. Xilinx ISE 8.1 is used for place and route of the design and to implement it for generate bit files burnt on PROMs to be configured on FPGA boards.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Literature Review

Software simulation falls short of satisfying the productivity expectation of designers as today's typical design size grows past a few million gates [1]. It is difficult to accurately simulate behavior of such large designs using software solutions alone. Hardware assisted solutions, which include acceleration and emulation, can help bridge this gap. FPGA based emulation is one of the most promising solutions. It is also followed at certain processor development departments at Intel Inc [2].

The cost of designing a system on chip (SOC) has been increasing to such an extent that only high performance devices are implemented in Application Specific Integrated Circuits (ASICs), and design mistakes are unaffordable. This results in higher emphasis on design verification which usually takes 75% of the effort [1]. Design verification is widely achieved in three ways: by simulation, by commercial emulators or by FPGA based prototyping. Most high performance and complex devices cannot be verified by software simulators alone. The time required to simulate a large data processing application such as video with a traditional HDL simulator is no longer practical. Furthermore, many applications need to achieve specific speed targets for the prototype to be useful. For this type of application there is a strong movement in the

market toward prototyping ASICs using FPGAs [3]. In the programmable logic market, the latest, high-density Xilinx Virtex family offers an ideal solution.

We, here at Intel's Infrastructure Processor Emulation Department are working on emulating a network processor using Xilinx Virtex-4 FPGAs.

1.2 Network Processor Blocks

The processor essentially has a memory controller hub (MCH), an IO Controller hub (ICH), Micro engines, Gigabit Ethernet (GBE) and a Clock Reset Unit (CRU). The network processor's IA (Intel Architecture instruction set commonly known as "x86") core runs at 600-1200 MHz with an on-chip 100 plus MHz front-side bus (FSB) interface. The IA core is effectively a Pentium4 processor [4].

The network processor's MCH is derived from the P861 chipset and operates in lock step with the front side bus (FSB) at 100 plus MHz. The MCH provides the main path to memory for the IA core and all peripherals that perform coherent IO (e.g. the Peripheral Component Interconnect (PCI) Express, the ICH south bridge as well as coherent CPP (Command push pull) transactions).

The Network Processor's ICH provides a set of PC-platform compatible I/O devices that include Serial Advanced Technology Attachment (SATA), Universal Serial Bus (USB), and serial UART (Universal Asynchronous Receiver/Transmitter) interfaces. The ICH complex runs at 100 and odd MHz and interfaces to the MCH through the "NSI" (north-south interface) , a simplified on-chip interface.

The network processor's communications complex supports Gigabit Ethernet (GbE) media access controllers, high-speed serial Time Division Multiplexing (TDM

interfaces. Gigabit Ethernet (GbE) is a term describing various technologies for implementing Ethernet networking at a nominal speed of one gigabit per second defined by the IEEE standards. Recently, it has become a built-in feature in many motherboards [5].

Four ME (Micro Engines) and a cryptographic "security services module" (SSM) accelerate common security and packet processing tasks. The ME and SSM attach to a single 64-bit Command Push Pull (CPP) chassis with two command busses. A multi-ported "convergence bridge" allows CPP buses to access on-chip SRAM (Static Random Access Memory) and external DRAM (Dynamic Random Access Memory). Based on the memory map configuration, the convergence bridge routes coherent requests through the MCH (where they snoop the IA processor's caches), while others travel directly to the memory (and do not snoop the IA Processor caches) [4]. The Memory Complex operates at few hundred MHz, depending on the configuration.

The processor will contain one PLL (Phase Locked Loop) sourced by a few MHz clock. This PLL will output a few GHz clock that will feed a Clock Reset Unit (CRU) The CRU contains a divider block and a distribution block [4]. The emulation proposed here bypasses the PLL logic.

1.3 Verification Techniques

With the increasing complexity of the design and number of transistors, verification has become the most important step in the whole design process and needs greater efforts and emphasis. The task of verification is determining whether some

product conforms to its specifications and expected behavior. Verification techniques are aimed at verifying the so-called register-transfer-level (RTL) representation of a design. For most chips, that RTL description provides the input to a logic synthesis tool, which then automatically generates a correct transistor-level implementation. In order to achieve the highest and the most accurate performance in a microprocessor, manual design is preferable for large fractions of the design. That in turn makes the design very prone to errors; hence it is almost mandatory to verify that the RTL and the transistor-level representations of the design are functionally equivalent [6]. Verification is most popularly done in the following three ways:

1. Simulation
2. Commercial Emulators
3. FPGA Based Prototyping

Simulations usually run at 6 to 10 orders of magnitude slower than the actual ASICs [7]. Commercial emulators are usually much faster than simulations [8], but is a very expensive approach. FPGA based prototyping can run at speeds comparable to real ASICs, almost at 10-80 MHz [7]. Prototyping ASICs on FPGA based platforms can also be customized and more precisely controlled than commercial emulators.

1.4 ASIC Prototyping on FPGA Platform

ASICs are often prototyped on FPGAs and Synplicity provides ideal tools for the implementation [3].Prototyping ASICs with high density FPGAs offer an efficient verification solution by reducing the design risks and making the verification much

faster [1]. Xilinx FPGAs have an increasing demand in the market since they are high density FPGAs, and they exhibit internal performance of over 100 MHz [9].

The initial Virtex-4 family (used in my project) includes three platforms; Virtex-4 LX for logic, Virtex-4 SX for very high performance signal processing, and Virtex-4 FX for embedded processing and high-speed serial connectivity. Each version has a different mix of special features and comes in a range of density to cover a variety of application sizes [10].

Nowadays, more designs can fit into these high density, high performance FPGAs. Previously, the biggest FPGA was $1/100^{th}$ the gate count of ASICs as against today's FPGAs which are $1/10^{th}$ the gate count [7]. In the recent past more and more systems are plug and play as IP vendors design to work in both ASICs and FPGAs. Despite the fact mentioned above HDL code is never 100% compatible between ASICs and FPGAs [7].

Most designs these days are huge enough and would require more than just a single FPGA for implementation [11]. For very large ASICs the design needs to be partitioned across multiple FPGAs and the time and expertise required to do so is very high [3].

Designers usually make multiple time-consuming iterations between synthesis and partitioning to find a "legal" partition for the design. Synthesis is typically done without understanding the partitioning, resulting in a prototype which may not run at the target speed. The high capacity and high performance of Xilinx Virtex FPGAs combined with the unique partition-driven synthesis algorithms in Synplicity's Certify

tool means that even the largest multi-million gate ASIC can be prototyped with a manageable number of Virtex devices. Each Virtex-4 device used in the project namely, XC4VLX200, can accommodate up to 200,000 logic cells [12].

FPGAs are difficult to partition, place and route if they are utilized above about 80% of all the CLBs (Configurable logic blocks) [13]. Hence, it is recommended to use more FPGAs than trying to crowd any one, which then leads to the problem of signals crossing more chip boundaries. Each Xilinx Virtex-4 XC4VLX200 can use a maximum of 960 pins across FPGAs, but after reserving certain pins for power pins, clocks and system control one would be left with only 700 plus pins for interconnection among FPGAs [12, 14].Each FPGA pin (physical wire) is dedicated to a single emulated signal (logic wire) and hence only a fraction of potential communication bandwidth is used. These logical wires can only switch at emulation clock speeds and all of them may not be active simultaneously. Hence we can introduce time multiplexed pin usage. Virtual wires overcome this pin limitation by multiplexing each physical wire among multiple logic wires and pipelining these logic connections at the maximum frequency of the FPGA. Virtual wires represent a connection from logical output of one FPGA to the input of the other FPGA. This results in the improvement in the bandwidth thus reducing the global interconnects. This allows usage of low dimension inter-chip connections (such as nearest neighbor). Nearest neighbor topologies and the virtual wire concepts can improve the efficiency of emulations [15, 16].

Synplicity has enabled a powerful prototyping method through a new tool called Certify. Certify takes a fresh approach to the prototyping problem by integrating

partitioning and logic synthesis. By performing synthesis during the partitioning phase, accurate area and pin utilization information is presented so that partitioning decisions can be made in a fast, interactive environment. A second benefit is that Certify performs partition-driven synthesis. In other words, the physical partitioning data is used to affect synthesis along with timing constraints. Certify also has an option of Quick Partitioning Technology (QPT) which allows the user to automatically partition the RTL [15].Certify is the only tool that combines multi-chip partitioning with RTL synthesis.

For pin multiplexing in the project another Certify feature called Certify Pin Multiplier (CPM) is used to reduce the number of I/O pins required between FPGAs. CPM applies a Time Division Multiplexing (TDM) technique to reduce pin count [17]. In this project mostly 8:1 or 16:1 multiplexing is used. Not all nets qualify for CPM and the major criteria for qualification are the existence of a sequential element on the destination FPGA. Other CPM qualifications are discussed in further detail in Chapter 4. Some of the unqualified signals can be worked on and forced to qualify for CPM by using certain techniques which are also described in Chapter 4. The use of CPM may result in additional timing overhead and some area penalty.

The partitioned design is then synthesized again and the generated netlist file is given as an input to the Xilinx ISE (Integrated Synthesis Environment) tool. ISE software from Xilinx helps meet design performance in the shortest possible time. And to make the most of the Virtex-4 performance advantage, Xilinx offers Plan Ahead software, which allows designers to analyze floor-plan and improve placement and

timing of even their most complex designs. It allows the designers to meet timing budgets and reduce design iterations in an easy-to-use design environment [18]. This tool generates PROM (Programmable Read Only Memory) image/bit files and configures flash. Xilinx iMPACT software of ISE programs and verifies. The PROM (flash) will automatically configure the FPGA on power up.

While doing place and route one of the greatest of emulation challenges namely gated clocks is faced which is usually foreseen and resolved at earlier stages. In ASIC design gating off clocks to conserve power is a common design practice. Unfortunately, when prototyping an ASIC design with multiple clock trees on a FPGA, precisely controlling these clock trees implementation is not feasible.

Most of the gated clocks issues, if they fall in the eligibility criteria for fixing, can be resolved by Synplicity's gated clock fix enable option. For those that do not satisfy the criteria need to be fixed manually and can prove to be one of the most time consuming tasks in the whole design process. Unfortunately, in this project there is a lot of legacy code and hence a lot of gated clocks some of which are outputs from multiplexers and more complex to resolve.

1.5 Summary

With the increase in the transistor density and decrease in die size the complexity of the design and its cost to manufacture has increased. A design error has become too expensive to commit and hence it has become mandatory to verify the design for the functionality and timing analysis.

Conventional methods of verification like simulation are less accurate to implement on complex designs since their speeds are not comparable to ASICs. Prototyping ASICs on FPGAs on the other hand can be done at much higher speeds (very close to real ASICs) than conventional techniques. Prototyping on FPGAs can offer the following advantages:

- High logic capacity—FPGAs can fit more than 2 million ASIC gates. More than 50 percent of ASICs designed today can fit into a single FPGA.
- High performance— FPGAs can run at speeds up to 450 MHz, meeting real-time testing requirements.
- Flexible I/Os with high bandwidth— FPGAs support memory interfaces such as DDR2 (Double Data Rate). FPGAs also support high-speed LVDS (Low Voltage Differential Signal) and serializer/deserializer (SERDES) to optimize chip-to-chip communication, simplifying board interconnects for ASIC designs requiring multiple FPGAs.
- Flexible on-chip memory blocks—FPGAs have flexible embedded RAM blocks that can be configured for a wide range of applications such as on-chip cache and video frame buffers.
- Design security—FPGAs include encryption features to protect IP when the prototype is used in the field during market testing [19].

Prototyping advantages come with challenges especially for huge designs like partitioning the design into multiple FPGAs. That is made feasible by Synplicity's Certify tool. A good insight into the RTL design and various blocks in the design is

required for partitioning as one has to keep into account the routing of signals across various FPGAs which should be kept minimal in order to meet timing requirements. Also, the signals in most cases need to be multiplexed since they cross the maximum usage limit in VLSI designs which is attained by Certify Pin Multiplier.

Xilinx ISE tool is used to place and route and implement the design and at this stage most designs suffer from gated clock issues which are usually predicted and fixed in advance. Unlike ASICs one cannot precisely control the clocks in FPGAs so all the clock gating used for reducing power consumption need to be resolved before taking the design through place and route. The Xilinx ISE tool generates the program files that can be burnt on PROMs and configured on FPGAs [20].

Thus, verification by prototyping can be implemented to save designers from bearing the cost of errors before the chip is fabricated.

1.6 Contributions and Future Research

The majority of the time was spent in the biggest units namely ICH (I/O Controller Hub) and MCH (Memory Controller Hub) of the processor. The RTL was converted to emulation specific RTL to be compatible with FPGAs and the design was synthesized. Converting RTL consumes a substantial amount of design efforts and time since the code was not designed originally to be compatible with FPGAs. Now an effort is being made towards back annotating these modifications so that in the future releases of the RTL the same work need not be done again.

ASIC memories are not synthesizable on FPGAs so there arises a need to convert them into FPGA block RAMs by using Xilinx Coregen tool. The netlist of the

Xilinx Coregen software created memories is passed during Place and Route stage [21]. All the block RAMs included in all the sub units of the network processor were created.

For the MCH and ICH a lot of work is done towards design partition and pin multiplexing the I/Os. Also, a lot of the I/Os were forced to qualify for CPM manually.

Gated clock conversion for the whole of ICH was done for all the sub blocks. Most of the clock gating was due to the test mode signals which were disabled for the purpose of this project. Also some multiplexed clocks were fixed.

Future Research would include working on NEX (Next Generation Expansion Buses) only release in the next few days which will include taking the latest release of NEX and taking it all the way through place and route and pin assignment, implementation and creating bit files to configure the FPGAs.

Also, further work has to be done to implement ICH and MCH all the way towards creating the program files to configure those sub units on FPGAs. This could be more challenging as MCH and ICH is more of the legacy codes which will face more obstacles during the Xilinx ISE place and route stage. MCH needs to be further fixed for gated clock issues.

Similarly, the next step would be to take the whole of the processor that needs to be routed and implemented through ISE tool to be configured on FPGAs.

1.7 Thesis organization

This thesis presents the emulation technique of an ASIC based network processor on a FPGA platform. The remainder of it is structured as follows.

Chapter 2 details the various blocks in a generic network processor and some special features of the processor used in the project. Each block's functionality and its generic architecture is also discussed.

Chapter 3 presents the various techniques which are used for verification by laying emphasis on the techniques used in the project, namely prototyping. The various advantages and some disadvantage of the FPGA based prototyping technique are also discussed.

Chapter 4 provides in detail the method followed for FPGA based prototyping. It describes each step in the emulation design flow and the tool used by each of them. The chapter presents various problems faced, and their solutions that were applied during the course of the thesis.

Chapter 5 summarizes the whole project by mentioning the advantages of the technique used. It talks about the contributions and challenges faced. It also discusses the future work that can be done.

CHAPTER 2

PROCESSOR BLOCKS

Intel Corporation offers network processors with high speed, flexibility that enable a broad range of services from the customer premises to the core of the network. Featuring robust, flexible packet processing on a single chip, the Intel® IXP2XXX product line of network processors handles complex algorithms and performs deep packet inspection, traffic management, and forwarding at line rate [22].

Intel provides an extended family of network and communications processors that meet the increasing processing demands created by faster line speeds, the deeper packet inspection requirements of content-based services and the need to support multiple protocols and evolving industry standards.

The network processor being emulated opens doors to new convergence markets by enabling new IPD products by IP re-use. It is a single chip (Intel Architecture) IA with balanced power, performance and price. It is primarily targeted to be used in telecom, storage industrial automation and print imaging. It is aimed at increasing IA scalar performance, improve power and cost efficiency, increase packet throughput. The target customers include industry leaders in the field of security, voice, routers, print and wireless applications. It would include commercial giants like Nokia, Cisco, Nortel and Cannon [23].

2.1 Network Processor's Block Summary

The Network Processor's core which is a Pentium 4 processor runs at 600-1200 MHz. The Processor typically has a Memory Controller Hub (MCH), an I/O Controller Hub (ICH), PCI-Express, a security service module (SSM), Gigabit Ethernet (GBE), Microengines and Memory cluster. Described below is each block with its detailed functionality.

2.1.1 Memory Controller Hub

The Network Processor's *MCH* operates in lock step with the FSB at 100 plus MHz. The MCH provides the main path to memory for the IA core and all peripherals that perform coherent IO (e.g. PCI Express, the ICH south bridge as well as coherent CPP (Command push pull) transactions). The *Northbridge*, also known as the Memory Controller Hub (MCH), is traditionally one of the two chips in the core logic chipset on a PC motherboard, the other being the Southbridge. Rarely, these two chips have been combined onto one die when design complexity and fabrication processes permit it. In general, however, core logic chipsets are divided into these two main parts. In the processor being emulated we are coming up with both MCH and ICH on the same chipset which increases the complexity of the design.

MCH typically handles communication between RAM, CPU, PCI-Express and the Southbridge. Because different processors and RAM require different signaling, a Northbridge will typically work with only one or two classes of CPUs and generally only one type of RAM. There are a few chipsets that support two types of RAM Generally these are available when there is a shift to a new standard). For example, the

Northbridge from the NVIDIA nForce2 chipset will only work with Duron, Athlon and Athlon XP processors combined with DDR SDRAM, the Intel i875 chipset will only work with systems using Pentium 4 processors or Celeron processors that have a clock speed greater than 1.3 GHz and utilize DDR SDRAM, and the Intel i915g chipset only works with the Intel Pentium 4 and the Intel Celeron, but it can use DDR or DDR2 memory.

The name is derived from drawing the architecture in the fashion of a map. The CPU would be at the top of the map at due north. The CPU would be connected to the chipset via a fast bridge (the Northbridge) located north of other system devices as drawn. The Northbridge would then be connected to the rest of the chipset via a slow bridge (the Southbridge) located south of other system devices as drawn in the Figure 1 below.

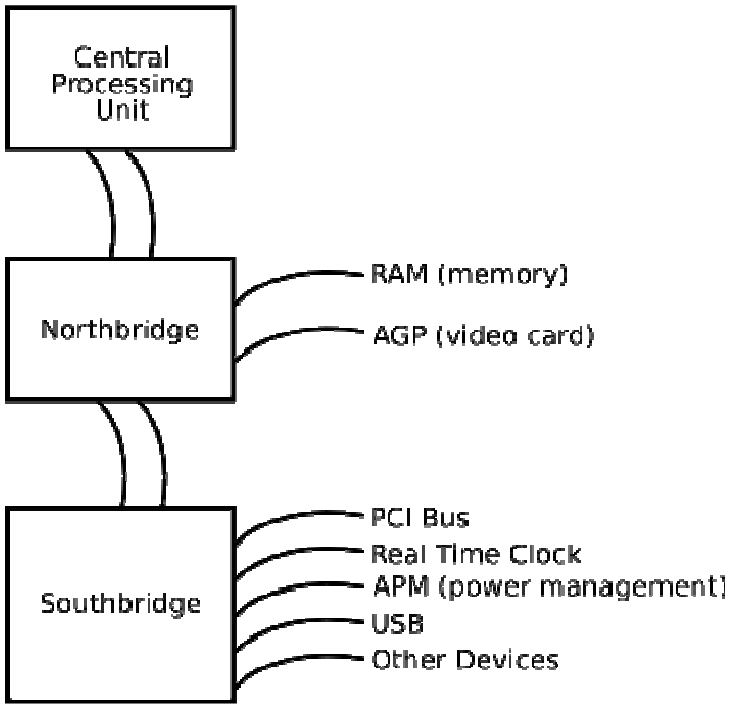

Figure 2.1: Block diagram of MCH and ICH and their interface.

The Northbridge plays an important part in how far a computer can be over clocked, as its frequency is used as a baseline for the CPU to establish its own operating frequency. In today's machines, the chip is becoming increasingly hotter as computers become faster.

2.1.2 Processor's I/O Controller Hub

The Processor's *ICH* provides a set of PC-platform compatible I/O devices that include SATA, USB, and serial UART (Universal Asynchronous Receiver/Transmitter) interfaces. The ICH complex also runs at 100 plus MHz and interfaces to the MCH through the "NSI" (north-south interface) interface, a simplified on-chip interface.

The Intel ICH6 family, for desktop systems, consists of the Intel ICH6, Intel ICH6R, Intel ICH6W, and the Intel ICH6RW I/O controller hubs and is an integral part

of Intel chipsets in conjunction with the (Graphics) Memory Controller Hub (G)MCH. Connected to the (G)MCH through the Direct Media Interface (DMI), a new chip-to-chip interface, the ICH6 provides platform features and capabilities for quality, robustness, and longevity [24].

The I/O Controller Hub (ICH)is also known as the Southbridge, implements the "slower" capabilities of the motherboard in a Northbridge/Southbridge chipset computer architecture. The Southbridge can usually be distinguished from the Northbridge by not being directly connected to the CPU. Rather, the Northbridge ties the Southbridge to the CPU. Traditionally this interface between the Northbridge and Southbridge was simply the PCI bus. However since this created a performance bottleneck, most current chipsets use a different (often proprietary) interface with higher performance. In this case the interface is called the NSI (North-South Interface).

The functionality of the contemporary ICH usually includes PCI bus, SMBus, DNA controller, Interrupt Controller, SATA (Serial ATA), USB (Universal Serial Bus), Power management, real time clock, LPC (Low Pin count) Bus and the BIOS ROM (flash). Basic Integrated Operating System or BIOS is the software run by a computer when first powered on. The LPC Bridge provides a data and control path to the serial I/O and BIOS RAM. LPC bus is used in PCs to connect low bandwidth devices to the CPU like the boot ROM, and I/O devices like keyboard, mouse etc. LPC's main advantage is that it requires only seven signals. It is therefore easy to route on modern motherboards, which are often quite crowded. An integrated circuit using LPC will need fewer pins.

The SMBus (System Management Bus) is used to communicate with other devices on the motherboard (e.g. system fans). The DMA (Direct Memory Access) controller allows LPC devices direct access to main memory without needing help from the CPU.

2.1.3 PCI Express

PCI Express is a high performance, general purpose I/O Interconnect defined for a wide variety of computing and communication platforms. A PCI Express link consists of dual unidirectional differential links, implemented as a transmit pair and a receive pair. A data clock is embedded using an encoding scheme to achieve very high data rates. PCI Express is intended to be used as a local interconnect only. As it is based on the existing PCI system, cards and systems can be converted to PCI Express by changing the physical layer only – existing systems could be adapted to PCI Express without any change in software. The higher speeds on PCI Express allow it to replace almost all existing internal buses.

2.1.4 Other Processor Blocks

The Network Processor's Communications Complex supports three Gigabit Ethernet media access controllers, high-speed serial TDM interfaces. Gigabit Ethernet (GbE) is a term describing various technologies for implementing Ethernet networking at a nominal speed of one gigabit per second defined by the IEEE standards. Recently, it has become a built-in feature in many motherboards. Gigabit Ethernet allows network transfers up to 1.000 Mbps using standard Cat 5 UTP (unshielded twisted pair) cabling [5].

Four ME microengines and a cryptographic “security services module” (SSM) accelerate common security and packet processing tasks. The ME and SSM attach to a single 64-bit Gen3 CPP chassis with two command busses. A multi-ported convergence bridge” allows CPP and XSI buses to access on-chip SRAM and external DRAM. Based on the memory map configuration, the convergence bridge routes coherent requests through the MCH (where they snoop the IA processor’s caches), while others travel directly to the memory (and do not snoop the IA processor caches).

CPP and XSI memory requests specify whether they need to snoop the IA processor’s caches. The bridge routes coherent requests through the MCH, while others travel directly to the memory. The Memory Complex operates at few hundred MHz, depending on configuration.

The Processor will contain one PLL sourced by a few MHz clock. This PLL will output a few Ghz clock that will feed a Clock Reset Unit (CRU) The CRU contains a divider block and a distribution block. For emulation purposes the PLL logic is bypassed. CRU is the clock reset unit which controls the state of the external reset (resetn), and usually if it has a value ‘0’ for at least a few say three clock cycles, then it sets the internal synchronous reset signal (rst) to ‘1’ [25]. The number of clock cycles can vary from one design to another.

CHAPTER 3

VERIFICATION TECHNIQUES

It is well known to the ASIC designers that verification prior to the chip fabrication is an effective but a time consuming process taking almost 75% of the design effort [1, 26]. As today's typical design size grows past few million gates, software simulation falls short of satisfying the productivity expectation of designers. It's difficult to accurately simulate behavior of such large designs using software alone. Hardware assisted solutions, which include acceleration and emulation, can help bridge this gap. FPGA based emulation is one of the most promising solutions also followed at certain processor development departments at Intel Inc.

Like other applications of programmable logic, we don't have to re-invent emulation from scratch. Depending on your complexity and performance requirements, you can find solutions starting with single-FPGA boards and ranging up to sophisticated, scalable emulators with multi-million ASIC gate capability.

Unfortunately, FPGA-based emulators in the past have had a bad reputation due to long setup times, poor reliability, and high cost - especially for systems supporting large design capacity. And, those criticisms were certainly appropriate in previous generations of these tools. Long setup times stemmed from the painful process of

partitioning a design into a suitable array of blocks, while simultaneously fighting the bottle neck of limited I/O bandwidth.

Nowadays, an FPGA-based emulator offers several critical advantages over its processor-based counterpart, advantages that significantly outweigh its few drawbacks. From the hardware standpoint, an FPGA-based emulator can perform at speeds unmatched by other technologies, short of going to real silicon. Hitting 10MHz or higher is now quite possible, even on large designs, thus providing the levels of performance needed to support software validation in advance of silicon.
When properly designed and supported by a robust software package, the FPGA-based emulator can both debug the hardware and validate the embedded software in a systemon-chip (SoC) design. It all hinges on the hardware architecture and the software support, both of which are very vendor specific.

ASIC designers do verification in most popularly three ways:

1. Simulation
2. Emulation
3. FPGA Based Prototyping

3.1 Simulation

Simulation is one of most popular and oldest ways of verifying a design. This technique of verifying runs at much lesser speed when compared to the real world ASICs so it is not the most efficient way of verification.

Software-based simulation is widely used, but even when running on a really high-end (and correspondingly expensive) computer platform, it runs six to ten orders

of magnitude slower than the actual ASIC hardware, which makes it an extremely time consuming and inefficient technique. To provide a sense of scale, software simulation of the entire system can typically achieve equivalent speeds of only a few Hz (that is, a few cycles of the design's system clock for each second of real time). In practice, this means that extensive software verification can be performed on only small portions of the design.

3.2 Emulation

Hardware-based emulation is another alternative, but it is still at least three orders of magnitude slower than the actual ASIC hardware, because the massive amounts of multiplexing involved slows the verification speed down to only 500 KHz to 2 MHz. Furthermore, this approach is extremely expensive, both in terms of budget and resources (depending on the size of the emulator, the cost can be anywhere from 25 cents to $1 per equivalent gate) [27]. What designers need is an alternative that will allow them to get to market quickly with low risk and at low cost.

3.3 FPGA Based Prototyping

FPGA-based Prototypes: In many cases, it is necessary to verify the design "at-speed." In the case of a video processing chip, for example, part of the verification may involve evaluating the subjective quality of the video output stream. Similarly, verifying the hardware in the context of the embedded software mandates extreme speed. The answer is to use multi-FPGA prototyping boards running at speeds from 10 to 80 MHz, which is equal (or comparable) to real-time ASIC speeds ("real stimulus in, real responses out"). When it comes to designing a custom board versus using an off-

the-shelf board, the latter – when coupled with the appropriate design tools – can shave weeks if not months of verification time and (at a typical value of under 1 cent per equivalent gate) save tens of thousands of dollars in NRE charges.

Also of interest is that fact that, in addition to providing a platform for software development and hardware-software verification, the company designing the ASIC may simply require access to a fully-functional implementation of the design as soon as possible; for example, demonstration hardware may be required to take to a tradeshow.

Even though ASIC designs are increasing in size and complexity, recent advances in the capacity and performance of modern FPGAs means that 2/3 of these designs can be modeled using a single FPGA (this topic was discussed in an earlier whitepaper entitled FPGA-based Prototyping: Why all ASICs Should be Prototyped using FPGAs).

However, this still leaves 1/3 of these designs (that is, one in nine of all ASIC designs) requiring a multi-FPGA based prototyping board [27].

Our project falls in that smaller probability of 1/3rd of the designs that need multiple FPGAs typically in our case 11 FPGAs. This demands for partitioning and pin multiplexing expertise discussed in the next chapter.

CHAPTER 4

PROTOTYPING ON FPGA PLATFORM

In the previous chapters the advantages of prototyping techniques over other verification techniques were discussed. This chapter gives the detailed design flow involved in prototyping.

System designers can nowadays emulate huge sections or even the whole SOC (System on Chip) designs on FPGA boards using the high capacity, high performance FPGAs provided by many vendors like Xilinx, Altera etc. The FPGA board used for our project is the Xilinx Virtex -4 boards.

4.1 Xilinx Viretx-4 Boards

Virtex-4 devices incorporate up to 200,000 logic cells, 500 MHz performance, and unrivaled system features to deliver twice the density, twice the performance, and half the power consumption of previous-generation FPGAs [28].

The Virtex-4 family offers three platforms with a total of seventeen devices tailored to the requirements of different application domains. The multiple platforms enable the selection of the device that most cost-effectively implements each unique application.

4.1.1 High Performance logic (Xilinx Virtex-4 LX)

The Xilinix Virtex-4 LX series has the highest logic-to-feature ratio and can have a maximum of 200,000 LUTs (Look up tables). The popular FPGA LUT architecture consists of a function generator typically constructed out of a 16x1-bit RAM. The 4-bit input selects data in one of the 16 entries in the table. That method can easily produce any function out of the four inputs. The LUT output is then routed to a flip-flop to form a programmable logic cell. FPGA architectures are measured by how many logic cells they contain [29].

It also has the highest I/O-to-feature ratio since the maximum IO utilization is as high as 960 in a Xilinx LX device.

4.1.2 Ultra-high-performance signal processing (Xilinx Virtex-4 SX and FX devices)

Xilinx Virtex-4 SX series has highest DSP-to-feature ratio and highest memory-to-feature ratio. Xilinx FX devices have Embedded Processing and high-speed serial connectivity which include Embedded IBM PowerPC processor.

Virtex-4 FPGAs achieve new levels of performance and value by combining the revolutionary ASMBL (Advanced Silicon Modular Block) architecture with 90 nm process and 300mm wafer technologies. Xilinx has proven these technologies in 90 nm devices shipping in high volumes since 2003.

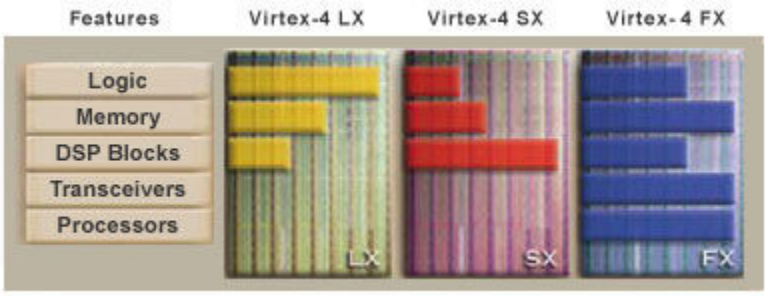

Figure 4.1: Xilinx Virtex-4 family devices and their features.

The innovative ASMBL (Advanced Silicon Modular Block) architecture enables Xilinx to assemble FPGA platforms with varying feature mixes suitable for different application domains.

The Xilinx Virtex-4 LX device used in the project namely XC4VLX200 has 200,448 logic cells and 1392 distributed RAMs and these together make the CLBs (Configuration Logic Blocks) for the device. It has 6048 block RAMs. The maximum user I/O is 960 [12]. It exhibits up to 40% speed improvement over previous generation devices. Out of 200,000 Logic cells 178,000 are LUTs and others are logic expanding multiplexers and I/O registers. It also has a 18 bit x 18 bit multiplier, multiply accumulator or multiply adder blocks.

4.2 Synplicity Tools for Synthesis and Partitioning

Synplicity provides tools like Synplify Pro which can support single FPGA based prototyping and a versatile tool called Certify which supports multiple FPGAs. Certify has several advantages over any other synthesis tool. The most prominent of the advantages is that it allows partitioning and pin multiplexing that are inevitably needed for large designs and can not be attained by other tools.

The Certify ASIC prototyping solution is the leading product for ASIC prototyping using multiple FPGAs. Certify software combines RTL multi-chip partitioning with best-in-class FPGA synthesis. Using the Certify product makes ASIC prototyping significantly easier, shortens prototype development time, improves prototype performance, and enables faster time-to-market. Shown below is a very user friendly graphical interface window of Certify which even allows the user to drag and drop the modules in whichever FPGA. It also shows the pin utilization and area estimation of each FPGA.

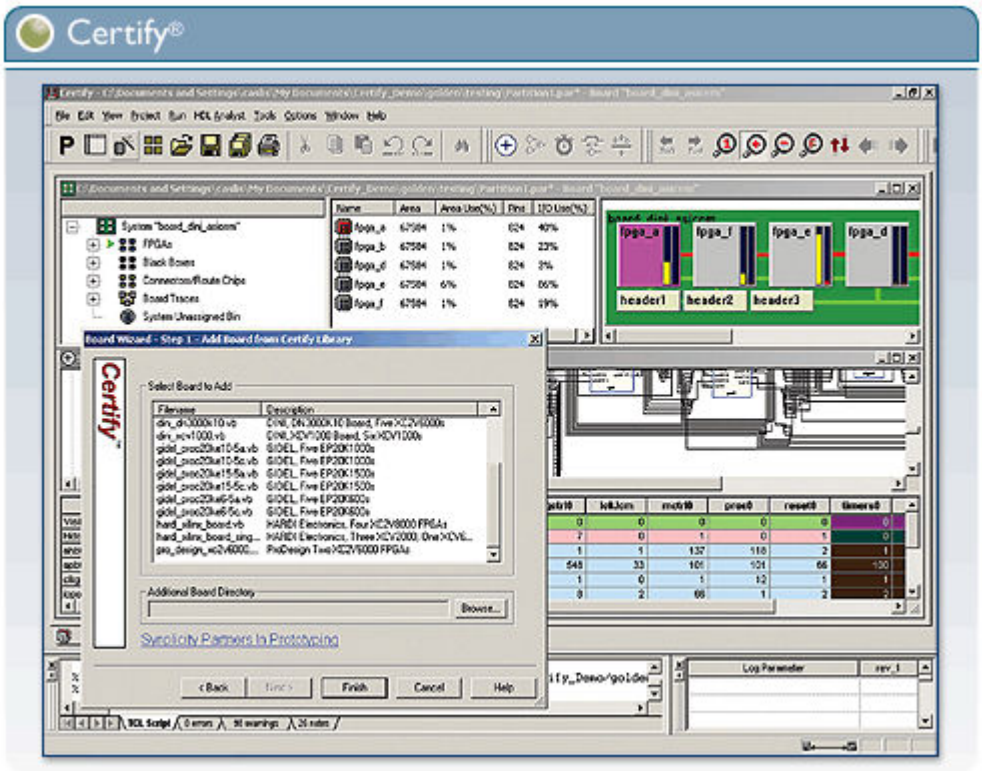

Figure 4.2: GUI window of Symplicity's Certify tool illustrating partitioning feature on multiple FPGAs.

4.3 Steps involved in Prototyping

Prototyping includes taking an RTL and synthesizing it in order to implement it on FPGA platforms so that it can be verified. RTL taken from the ASIC designers is not 100% compatible with FPGAs and need to be converted to an emulation specific RTL namely the eRTL. Shown below in Figure 4.3 is a typical flow chart for emulation [20].

It starts with extracting the design that one needs to emulate followed by preparing eRTL. This eRTL needs to be synthesized using synplify Pro or Certify tools. Synthesis can take anywhere from a few minutes to several hours depending on the design size.

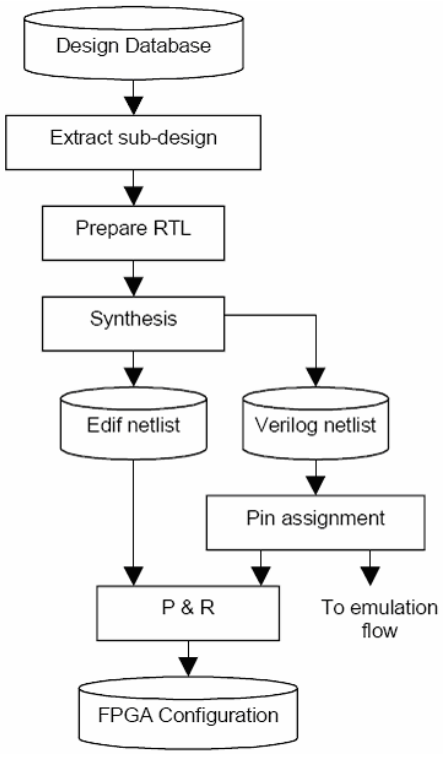

Figure 4.3: A typical design flow in FPGA based prototyping.

4.3.1 Synthesizing MCH

MCH Synthesis usually takes around 7-8 hours and needs a lot of RTL modifications to be made as lot of it is legacy code being carried over from previous generation MCH. It had around 252,000 LUTs.

ICH synthesis typically takes 4-5 hours in our design. It also needs several RTL changes to be made. The number of LUTs was around 145,000. Although this is less than maximum allowable 178,000 but trying to fit it one might lead to problems at place and route due to high density.

Both MCH and ICH are huge and were partitioned on multiple FPGAs. After synthesizing the MCH, there was a need to partition the design on two FPGAs since the whole design was too big almost 252000 LUTs (look up tables) which translate to a gate count of almost 4-5 times more than the LUT figure mentioned. Each FPGA Xilinx Virtex4 Device can accommodate around 178000 or less LUTs so the design had to be partitioned in a way that it gives a good balance between the LUT count and the pins connecting the two FPGAs. The maximum pins that an FPGA offers is 960 and for MCH there were around 1411 across the FPGAs. In order to fit in all these pins we have to use a feature in the tool called CPM (Certify Pin multiplexing) which is equivalent to the multiplexing on the source and demultiplexing these signals on the destination side.

To partition, the PCI-Ex (PCI-Express) was placed in the FPGA board, say U2, and the rest of the MCH is placed in the FPGA board, say U1. Where, U1 and U2 are FPGA boards as named by Certify tool. Table 4.1, and

Table 4.2 illustrate the signal counts before and after multiplexing the interface between the FPGAs respectively.

Table 4.1: Signal count before multiplexing the interface between FPGAs.

Signals Between	No. of Signals
U1 - U2	1411
U1 – elsewhere	1578
U2 – elsewhere	53

Table 4.2: Signal count after multiplexing the interface between FPGAs.

Signals Between	No. of Signals
U1 - U2	373

The number of signals between FPGAs U1 and U2 after multiplexing was further brought down from 373 to 242 by converting more signals that qualify for pin multiplexing. To do so, it is required to replicate or move some instances since in certain gate logic it has to be either copied across FPGAs or moved from one to the other.

Replicating and trying to convert these signals to qualify for the CPM process is a tedious task. Investigation was done towards automating this process, but it could not succeed. The reason for the failure was lack of an adequate text file that would give sufficient information to automate the process. The file would have to contain text for each gate. This would be impossible because there may be more than million gates.

The MCH was partitioned and multiplexed. The MCH has to have an interface with the Memory Cluster, IO controller hub and PCI-Ex. The pins have to be multiplexed across all these FPGAs and only then can the place and route procedure eventually generate a bit file for each of these units. To do so all units need to be synthesized first and so all PPC cluster units including NEX, Gigabit Ethernet, CAS, ME were synthesized.

4.4 Problems faced during multiplexing

A signal does not qualify for CPM (certify pin multiplexing) if it goes to a combinational unit at the destination FPGA as shown in the figure below.

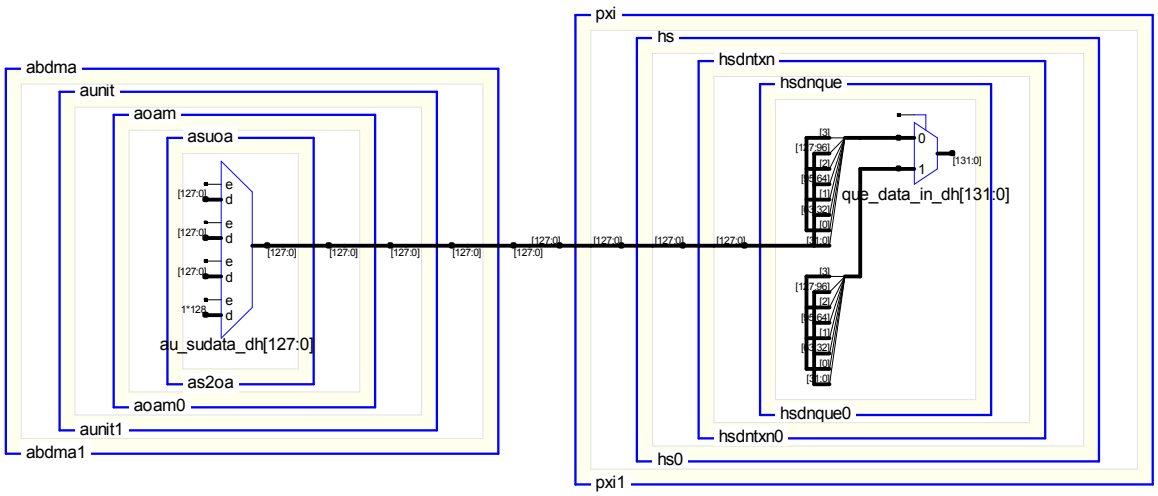

Figure 4.4: Example of net which is non-CPMable across two FPGAs U1 and U2.

In such cases expand the signal by clicking on the signal and trace its path forward to succeeding logic. This would allow seeing if the signal goes to a register so that the combinational unit can be moved across FPGAs to have a register/sequential

unit on the receiving end. Sometimes it is required to expand the signal deeper into the logic to see where one can hit a sequential element.

The signal shown in the above example Figure 4.4 when expanded goes to another mux and that mux outputs goes to RAMs as shown in the figure below:

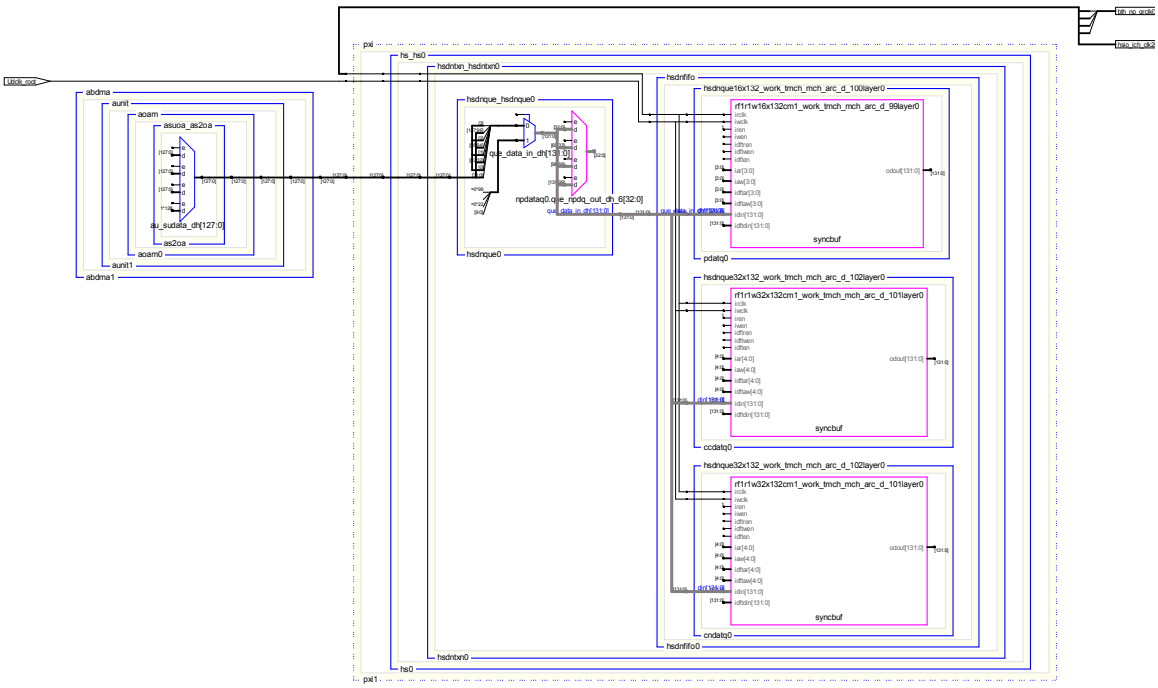

Figure 4.5: Expanded signal going to the RAM at destination FPGA.

The red high lighted path (the signal going from MUX to the RAMs) shows the signal eventually hitting a sequential element, a RAM (rectangular boxes in the figure).

Before moving the other two muxes into source FPGA make sure it does not adversely affect the pin count by expanding in all directions and then move the elements.

Initially the bus not capable of being multiplexed was a 128 bit data bus. This means 128 pins are going out of the FPGA by just this one bus. Moving the

combinational elements to the source results in a 132 bit bus crossing the interface. Since this bus qualifies for multiplexing the number of pins used in the 8:1 TDM is just 132/8 which is approximately 16 pins.

Similarly work was done on the other signals and the pin count dropped drastically from 483 non-CPM signals to 44 which significantly brings down the pin usage across the FPGAs.

Although RAMs are sequential the tool still does not recognize them to be sequential. This is because Synplicity tools are designed in such a way that at the partition stage the tool looks only at the RTL view which is generated from compilation and not at the synthesized view. Only during the synthesis stage does the tool look at the netlist files and without which it does not know if the coregen generated memories are sequential. So the memories had to be black boxed and forced to be sequential by using commands from Synplicity as depicted in Table 4.3.

Table 4.3: Synplicity commands to black box and make memories sequential.

```
object /* synthesis syn_force_seq_prim = 1 */ ;
where object is the module name of the black box.
module bbe (ena, clk, data_in, data_out)
/* synthesis syn_black_box */
/* synthesis syn_force_seq_prim="clk" */
;
input clk
/* synthesis syn_isclock = 1 */
/* synthesis syn_gatedclk_clock_en="ena" */;
input data_in,ena;
output data_out;
endmodule
```

These black boxed memories are later replaced with the actual ones while running the Xilinx ISE tool for place and route.

4.5 Quick Partitioner Technology (QPT) by Certify

QPT is the first automatic RTL partitioning technology. This provides tremendous ease of use and speed for the designer. Once the design compiled, the designer can choose to use QPT via a simple menu selection. The designer can also optionally set target goals for I/O and area usage for each FPGA. If no user targets are set, QPT uses the default setting of 100% I/O utilization and 80% area utilization. QPT is then run via a single menu selection. The performance of QPT is very fast. One example, a 600K gate design partitioned on a Pentium III computer in 44 seconds [30]. Another 2 million-gate design was partitioned in less than two minutes using QPT. Aside from the time saving benefits of this level of performance, the speed of QPT allows designers to try "what if" analysis of different board configurations for a design. With manual partitioning, the effort to redo partitioning on different board configurations was enough effort that most projects would just settle on a large enough board for the design. With QPT, a designer can, within minutes, try different board configurations to get a board with just enough capacity in order to reduce cost or target a board for whatever goal is desired.

In addition to the speed and ease of use, QPT also delivers excellent quality of results. Interestingly, QPT has delivered comparable results in I/O and area utilization to an experienced ASIC designer familiar with the design and the prototyping process. This allows a designer not familiar with the design or with prototyping methods to get results via QPT that would normally require an experienced senior designer.

4.6 Quad-Port Memories in Virtex devices

All the quad-port memories created were simulated and tested. The performance was as expected and now they are ready to be implemented and used for the processor wherever required.

Below is a brief description of how the existing dual port block memories in Virtex4 (FPGA board used in the design) can be used as Quad-port memories. This essentially halves the access time but doubles the functionality. The overall bandwidth of the block memory remains the same in terms of bits per second.

The dual port blocks are fully synchronous, true dual port structures. The user can read and write to and from each port independently with the exception of simultaneous read and write to the same address. A block diagram of the dual port block RAM is shown in Figure 4.6 below which has two ports for data input DataINA and DataINB and two for address input. The two clocks are the clocks to the dual port memory and the control logic is implemented by enabling the port A write enable WEA or port B write enable WEB.

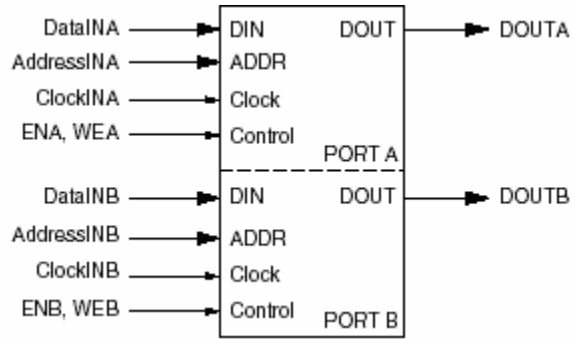

Figure 4.6: Basic block RAM structure.

By adding some extra logic and a clock running twice as fast, one can implement a quad-port memory by using the existing dual port memory function.

There are now four ports available to the designer, A, B, C, and D. Each can be used for independent Read or Write, with two provisos. The same clock (CLKA) should be shared for port A and port C and similarly, port B and port D should also share the same clock (CLKB).

The operation of the circuit is shown in Figure 4.7. A block diagram of the four-port functionary is shown in Figure 4.8.

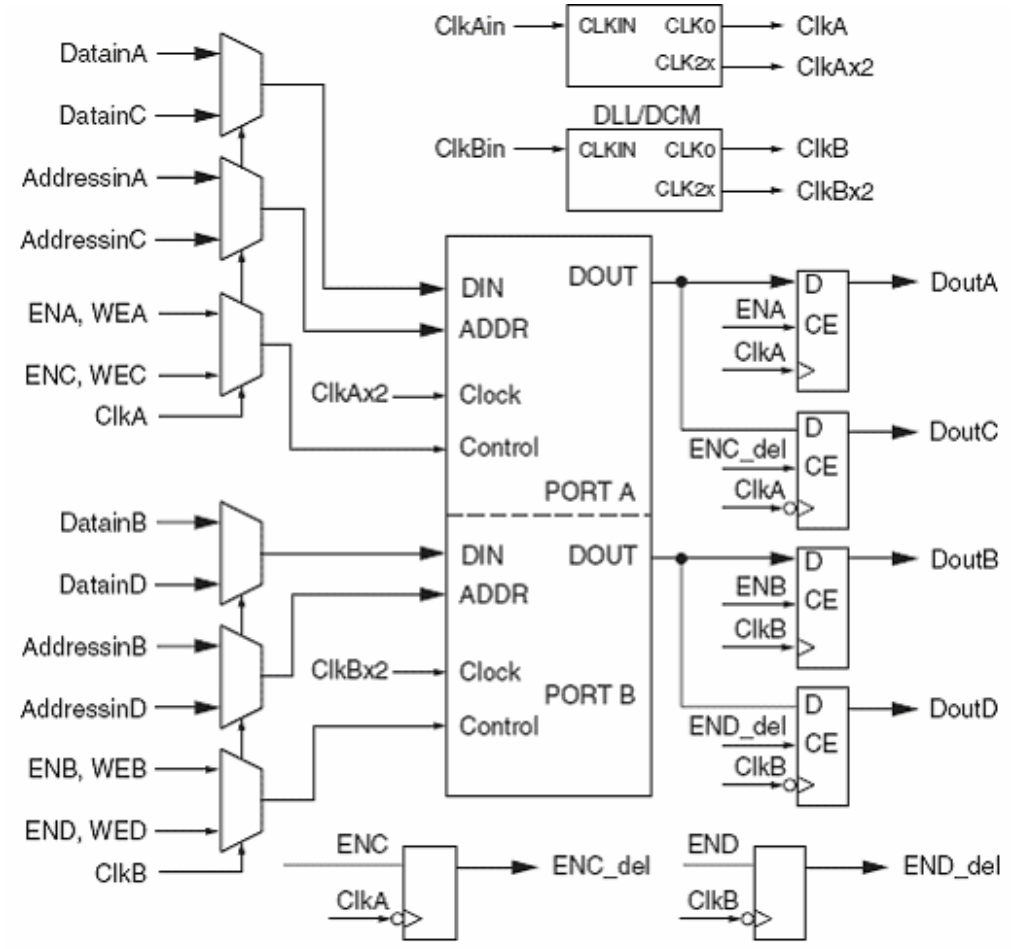

Figure 4.7: Quad-Port Functionality.

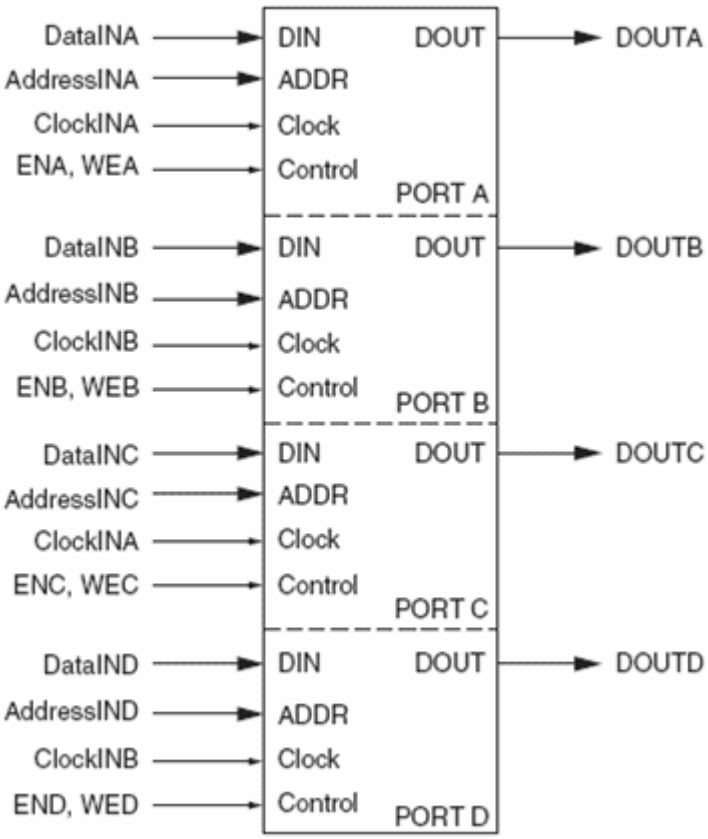

Figure 4.8: Four-Port Functionality.

A timing diagram is shown in Figure 4.9 which shows the output behavior at every clock cycle. Explained below are the outputs for ports A and C (B and D are identical) in further detail. At the rising edge of the system clock, the data, address and control for ports A and C will all change (point 1, in Figure 4.9). The information for port A is now applied to the RAM via a multiplexer. As the RAM is running at twice the rate of the system clock, this information will be put into the memory at the falling edge of the system clock, i.e., the next rising edge of the 2x clock (point 2, in Figure 4). Output data for port A will be available after the clock-to out time for the RAM, and as the system will expect this data to be valid for a whole cycle, it is then re-registered on

the rising edge of the system clock (point 3, in Figure 4.9), so as to be valid when necessary. Meanwhile the MUX changes state at position 2, and the data address and control for port C pass through the multiplexer on the second half of the clock cycle. They are registered by the RAM on the rising edge of the system clock in the normal way (point 3, in Figure 4.9).

The output data for port C will be valid following the clock-to-out time of the RAM, and is then reregistered by the falling edge of the system clock (point 4, in Figure 4.9) so as to be valid when necessary. The system will therefore see valid data for both ports at point 5 in Figure 4.9.

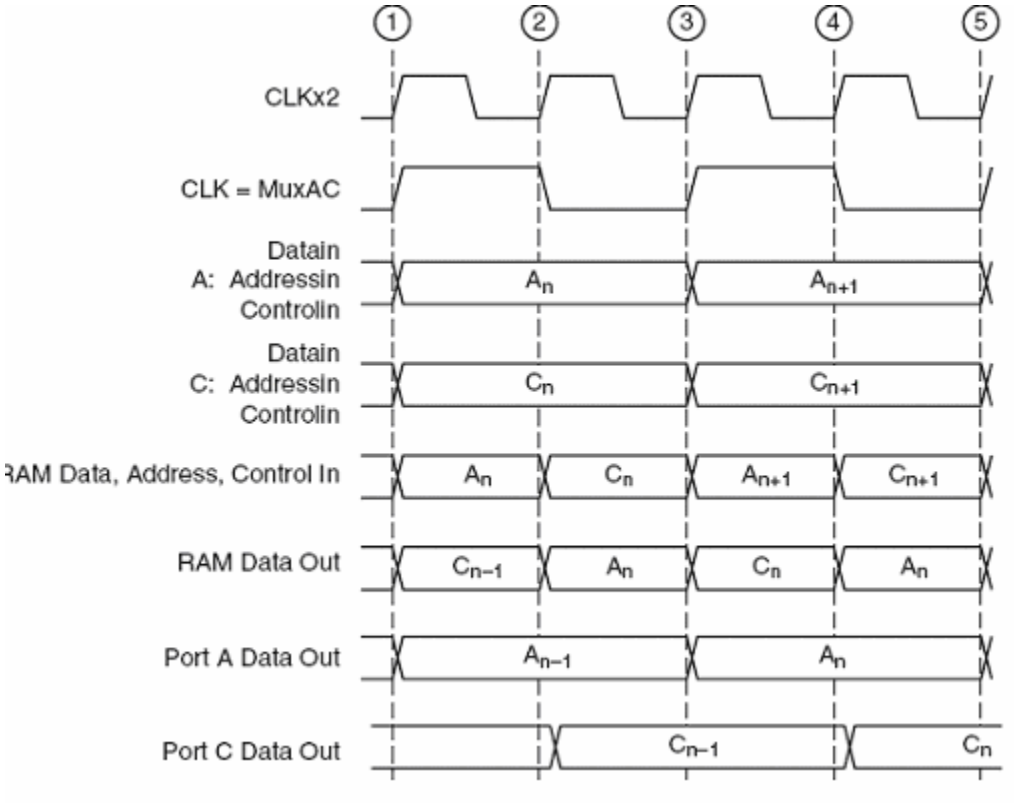

Figure 4.9: Timing Diagram.

4.7 Gated Clocks Assessment

The ability of Synplicity's Certify and Synplify Pro products to automatically convert ASIC-gated clocks to an FPGA-compatible form is a powerful tool for facilitating the prototyping of ASICs in FPGAs.

This report discusses the use of this feature. The ASIC designers usually use many gated clocks in their design. The ASIC design practice of gated clocks is summarized and contrasted with the FPGA design practice. Described below are the conditions that ASIC gated-clock structures must satisfy in order for Synplicity software to convert them to an FPGA-compatible form.

4.7.1 ASIC gated clocks vs FPGA global clocks

In ASIC designs, gating off clocks to conserve power is a common practice. Typically, an ASIC design is divided into modules, and the clock for a module that is not being used is gated off. In some cases, instead of entirely gating off the clock, the clock frequency is reduced. If current system loading does not require that the module run at full speed, the operating clock frequency of the module can be reduced by clock gating that results in a corresponding savings in power consumption. In a large ASIC design, this practice can result in hundreds of separate clock trees. ASIC designers have their design divided into modules with each module requiring different frequencies. In ASIC designs, a custom clock tree can be defined for each individual clock tree. When signals cross between different clock trees, the delays at these boundaries are adjusted to ensure that there are no set-up or hold-time violations.

Unfortunately, when prototyping an ASIC design with multiple clock trees into an FPGA, implementing a large number of customized clock trees is not feasible. Creating custom clock trees with unique requirements is a prohibitively difficult task and would require using the FPGA's programmable interconnect. In many cases, the FPGA vendors do not characterize the devices for worst-case minimum delays. In the absence of guaranteed minimum delays, attempting to use static timing analysis to prove that the FPGA does not have any hold-time violations is futile. Also, the FPGA's programmable routing interconnect is a finite resource. Using this resource to implement custom clock trees can lead to routing congestion and potentially an unroutable design.

4.7.2 Overcoming FPGA inability to implement custom clock trees

FPGAs provide a small number of dedicated global clock trees. These clock trees are routed to every sequential device on the die and are designed with such low skew that hold-time violations are guaranteed to be impossible. Using these global clock trees allows the programmable routing resources of the FPGA to be used primarily for logic interconnect and simplifies static timing analysis because checks for hold-time violations based on minimum delays are unnecessary. Placement and routing of the FPGA are likewise expedited. These types of trade-offs and simplifications allow the entire back-end design flow - placement, routing, and successful static timing analysis - for a large FPGA-based design, to be completed in a fraction of the time required to complete the equivalent flow for an ASIC device. Without this time savings, the ability to reprogram an FPGA would be of little advantage. The key to a

successful FPGA prototype implementation is to combine most, if not all, of the individual ASIC clocks trees onto the dedicated global clock trees of the FPGA by removing the gating from the clock inputs and moving it to the enable inputs.

4.7.3 Converting gated clocks to FPGA global clock trees

To implement a large number of gated clocks with a much smaller number of global clock trees, the gating is removed from the clock input and moved to the enable input on the sequential devices using the programmable routing resources of the FPGA; the ungated or 'base' clock is routed to the clock inputs of the sequential devices using the global clock resources. Because many gated clocks are normally derived from the same base clock, separating the gating from the clock allows a single global clock tree to be used for all of the gated clocks that reference that base clock.

Each sequential device in an FPGA includes a feedback multiplexer controlled by a clock enable as illustrated in Figure 4.10 of a D flip-flop. The Clock Enable (CE) input to the feedback multiplexer, when asserted, allows the contents of the flip-flop (the sequential device) to be updated with the D input. When CE is not asserted, the flip-flop retains its value. The feedback multiplexer emulates the functionality of the gated clock.

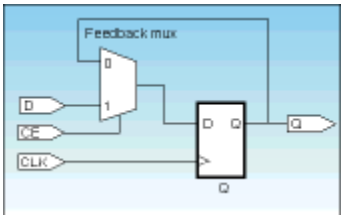

Figure 4.10: A D Flip-Flop.

Figure 4.11 illustrates how a gated-clock implementation is converted to one using clock enable. The RTL implementation with a gated clock is converted to a clock enable implementation in the 'Gated Clock Conversion' mode. In this mode, the implementation of the feedback mux is not shown. Also, the original base clock, 'clk', connects directly to the clock input of the clock-enabled flip-flop. In an actual design, the clk signal would probably clock many other sequential primitives. Likewise, the gated clock would also clock many other sequential primitives. After the gated clock conversion, all of these sequential primitives would be clocked by the base clock, clk. This base clock would be implemented using the FPGA's global clock which would eliminate clock skew and potential hold-time violations.

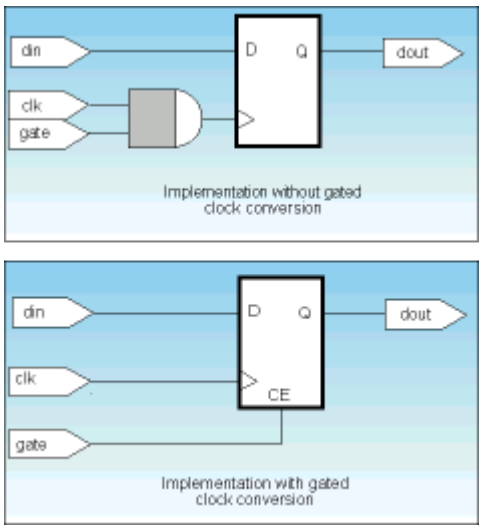

Figure 4.11: Illustration of how a gated-clock implementation is converted to one using clock enable.

4.7.4 General Requirements for Gated Clock Conversion

For a gated clock to be converted successfully, the following requirements must be met:

• Combinational Logic Only – the gated-clock logic must consist only of combinational logic. A derived clock that is the output of a register is never converted.

• Correct Logic Format – the combinational logic for the gated clock must have the correct logic format. Specifically, the combinational logic for a gated clock that drives edge-triggered sequential elements must satisfy the following two conditions:

– For at least one set of gating input values, the value output for the gated clock must be constant and not change as the base clock changes.

– For at least one value of the base clock, changes in the gating input do not change the value output for the gated clock.

• Single Base Clock – one, and only one, input to the combinational logic for the gated clock must be identified as a base clock. Nets are identified as clocks by specifying a period or frequency constraint in a constraint (.sdc) file.

SDC File Clock Definitions example:

define_clock -name {clk} -freq 10.000 -clockgroup clk

• Supported Primitives – the sequential primitive clocked by the gated clock must be supported by Synplicity for gated clock conversion. Synplicity supports gated clock conversion for a majority of sequential primitives. Even black box modules driven by gated clocks can be converted if special synthesis directives are used to define the black box.

• Option Enabled – the Fix Gated Clock option must be enabled for the project. The corresponding TCL command is available for gated clock conversion. The command syntax is: set_option -fixgatedclocks *value* where value is 0, 1, 2, or 3

Thus the conclusion is that gated clock conversion not possible on certain logic and some of which are described below.

A lot of the design in my project had logic where Synplicity's built in converter could not convert the gated clocks. For instance, the figure shown below illustrates a design where conversion is not possible:

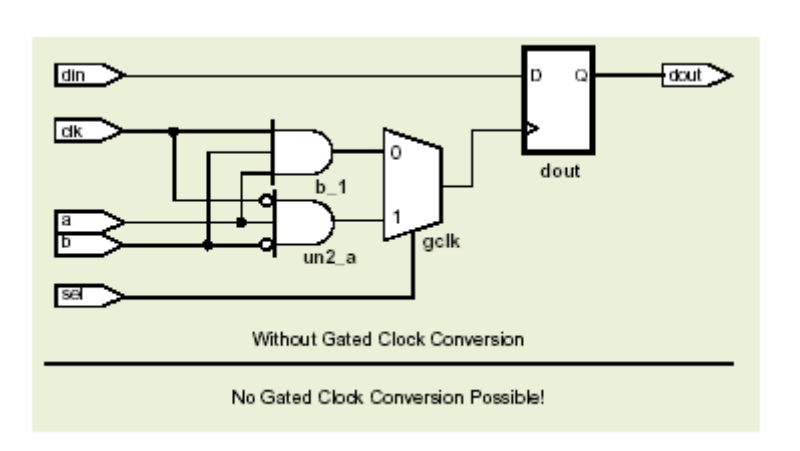

Figure 4.12: Example from [17]showing logic when gated clock conversion is not possible.

If the above example is modified so that the base clock, clk, has the same polarity in each product term, then both conditions are satisfied, and the gated clock is successfully converted. This is illustrated below.

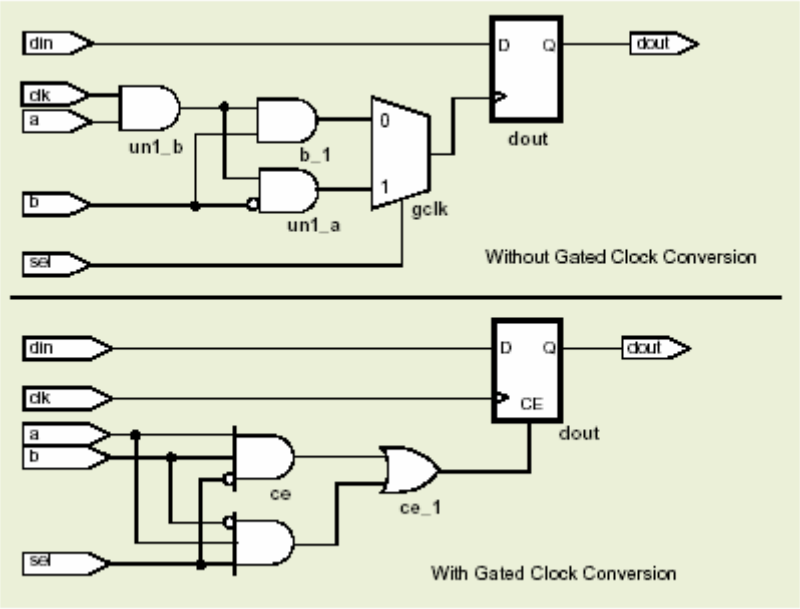

Figure 4.13: CLK with same polarity makes gated clocks possible.

Below is an example of a logic on which gated clock conversion is not possible at all:

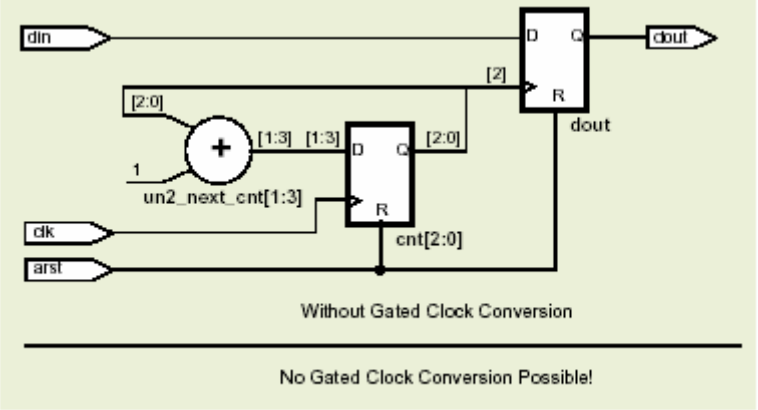

Figure 4.14: Figure illustrating circuitry on which gated clock conversion is not possible.

Also, when the clock is derived from a cascade of Muxes inbuilt conversion does not work. A temporary workaround for such cases is to trace back that signal till the original clock and change the logic as required.

This solution involves a lot of changes to be made at the RTL level. This is highly risky since one has to be very careful about not changing any functionality in this huge design. The whole design has to be simulated again after these changes are incorporated.

4.7.5 Gated Clock Assessment for ICH (manually resolving)

Gated clock assessment still remains the biggest challenge while emulating ASICs. Discussed further in brief are the various types of gated clocks that are present in the I/O controller hub (ICH), what are the measures taken to un-gate them and hence fix them so that the design is made usable for place and route.

Also described is another technique used to un-gate the clocks by disabling the test mode signals. Some scan/test mode signals are present just for testing purposes and they do so by toggling the flops.

In one of the cases shown below the clock is fed to the data input to avoid metastable states which is discussed in further details.

While working in resolving the gated clock issues it was found that every unit has an auto generated wrapper which has no logic but just a list of inputs and outputs and this is the ideal level of hierarchy where it is recommended to work in fixing gated clocks.

At this level one can look for all clocks that are used in the unit and define them explicitly in the constraint file namely the .sdc file so that the tool knows it is a clock. There are some signals like scan mode signals which when tied off to zero would completely optimize out the test mode logic and prevent several gated clocks. Only those scan signals that cause gated clocking were disabled thereby disabling the scan chain and un-gating the clocks.

The design has to be synthesized and to analyze the clock network a TCL (Tool Control Language) script is used. The report of the script is analyzed which helps in un-gating any left over clocks.

Major categories of gated clocks in ICH used in the design fall in the following categories:

- Some units have one global clock and is not gated; it goes directly to sequential elements.
- One of the units had around 60 generated clocks (mostly gated). By tying of the scan mode signal to value '0' resulting in disabling the logic derived from the signal, all gated clocks are converted to ungated clocks.
- Below is a case where tying off the *dt_scanmode* to zero to get rid of gated clocks is illustrated.

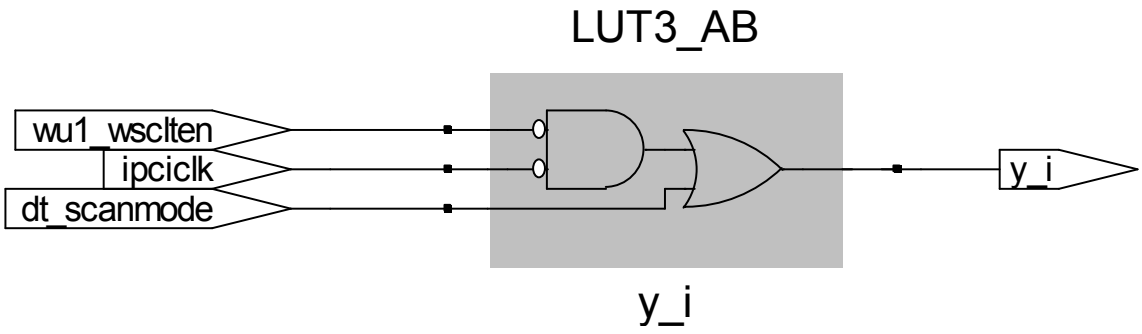

Figure 4.15: Circuit showing clock gating due to scan mode signals.

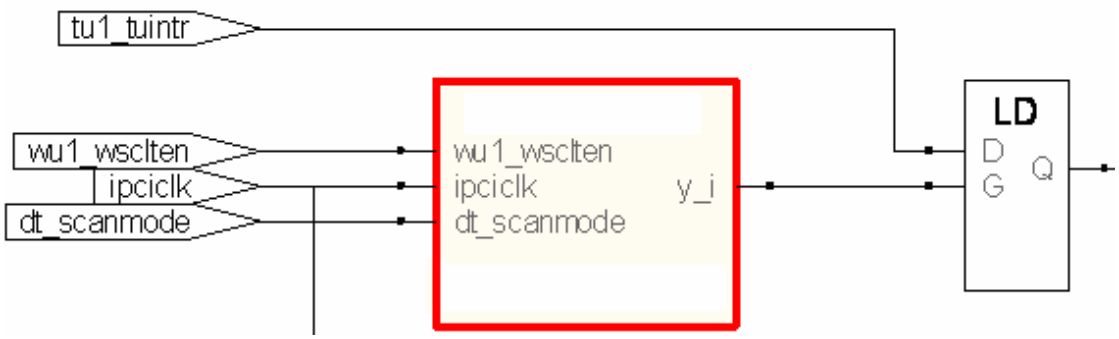

Figure 4.16: Module mapping dt_scanmode test signal.

Dt_scanmode is tied off to '0' in the case mentioned in Figure 4.16.

By tying off the signal to zero it is equivalent to not having the OR gate at all in the circuit shown in Figure 4.15.

- dunit_wrapper:

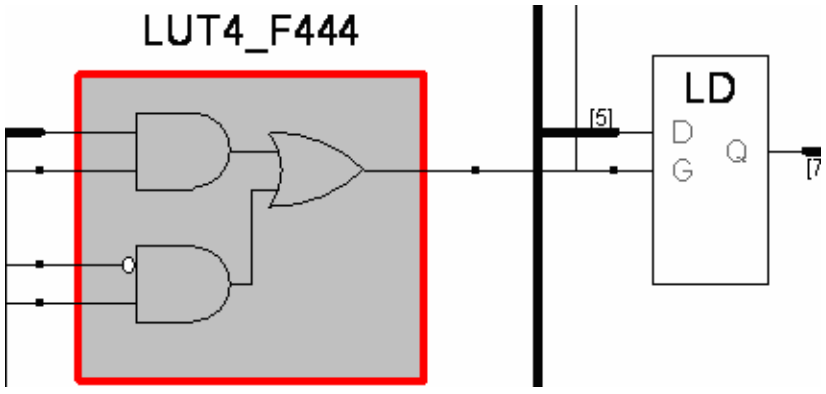

Figure 4.17: Instance of gated clocks from OR gates and AND gates.

In the above case shown in Figure 4.17 the gated clock comes from an OR gate that has both its inputs coming from AND gates and no scan mode signals present in the path which could be turned off.

So in this case the next step is to expand the AND gate inputs and see if they have any test enable signals which could be turned off.

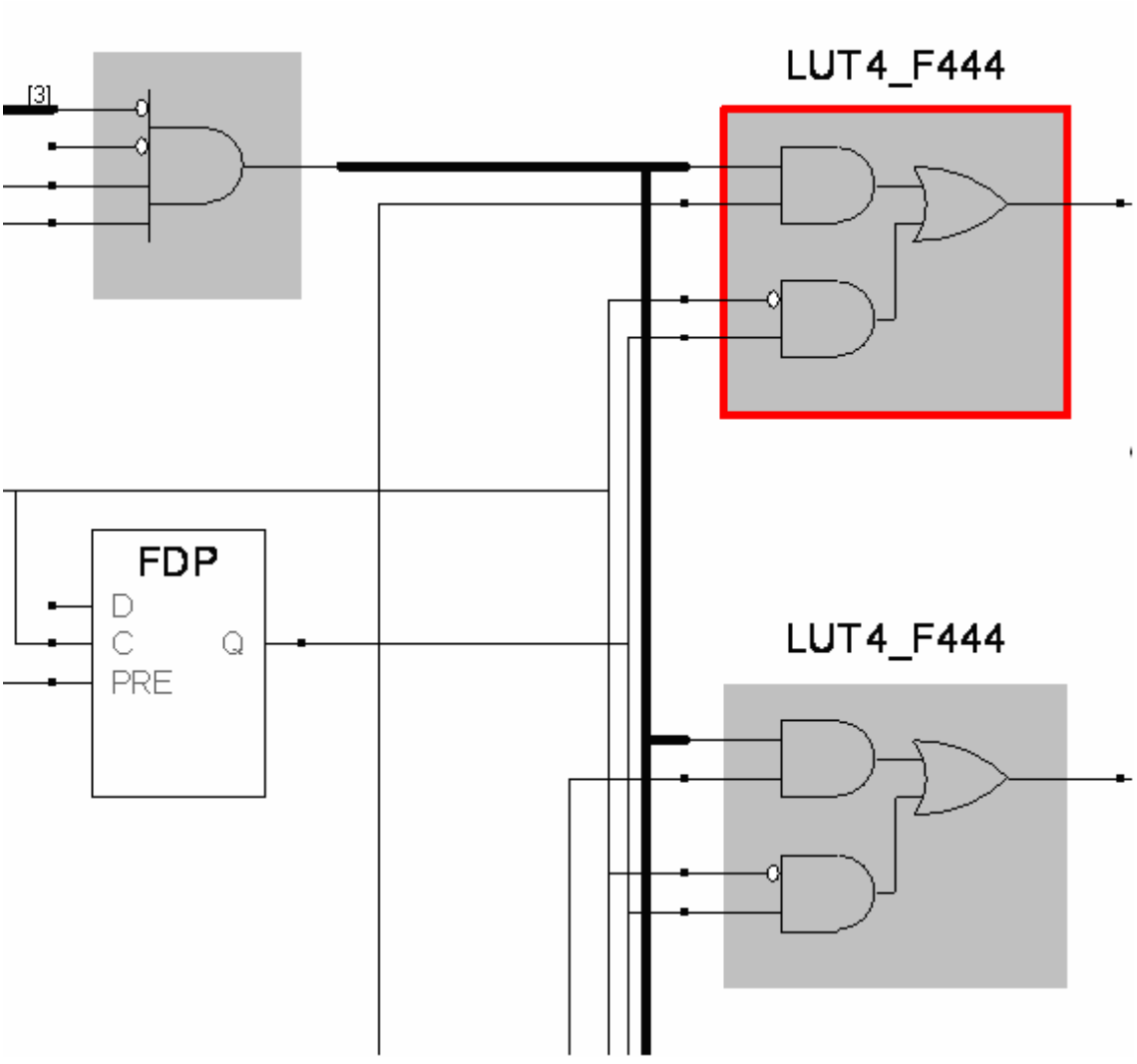

Figure 4.18: Instances where gated clocks are un-resolved.

The above case shown in figure 4, could not be resolved since there were no scan mode signals that could be tied off.

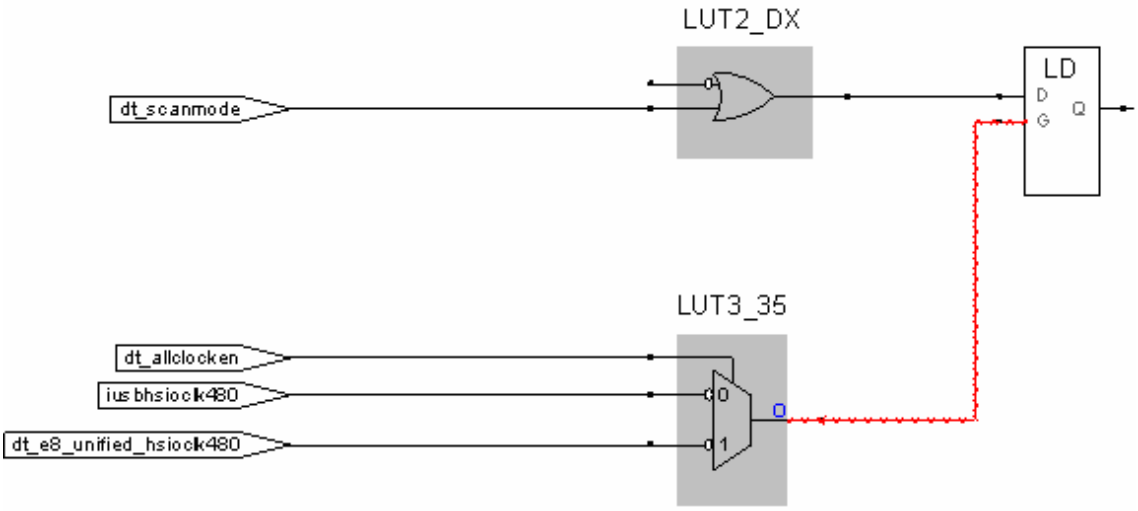

Figure 4.19: Example of clocks with test mode clocks.

Looking at the MUX above in Figure 4.19, its inputs look like both are clocks of which one is a dt_*, which is a clock to be used during test scan mode. Tying the enable input of the multiplexer to zero will always select the iusbhsioclk480 and never the test clock.

Some units are black boxed to avoid mixed driver errors so this report shows no sequential elements for such units.

4.7.6 Gated clocks unresolved in Edge detection circuits

The circuit shown below in Figure 4.20 has a clock going to the data input of the flop and yet another clock going to the clock input. These kinds of circuits are used as edge detection circuits to avoid metastability.

In general for the edge detection circuits the clock going to the clock pin should be typically 4-5 times as fast the clock going to the data pin in order to function correctly.

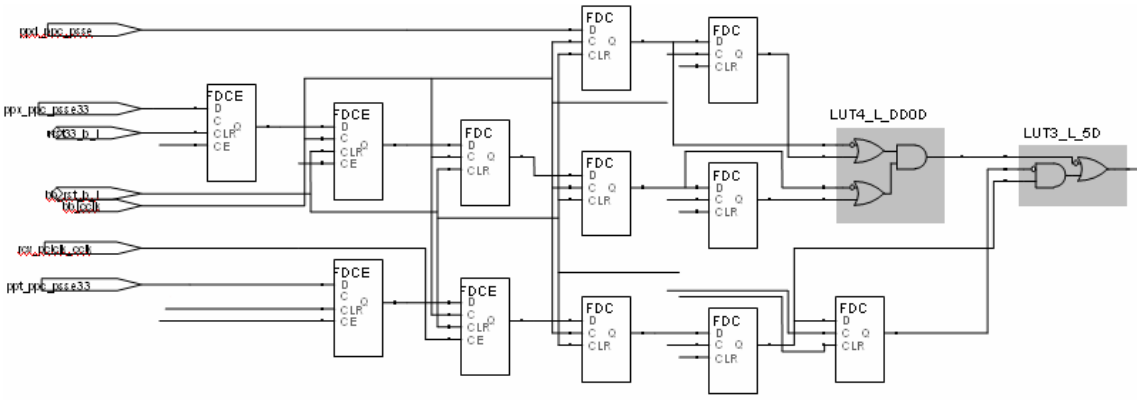

Figure 4.20: Circuit showing cascaded flops used in the design to solve metastability.

4.7.7 ICH Pars1 clocks

- ssusunit1_wrapper: CLOCK net wu1_cr_cd31_f0_extstmreg3 comes out of a MUX as illustrated in Figure 4.21. The gated clock had to be left unresolved since it is being used for edge detection.

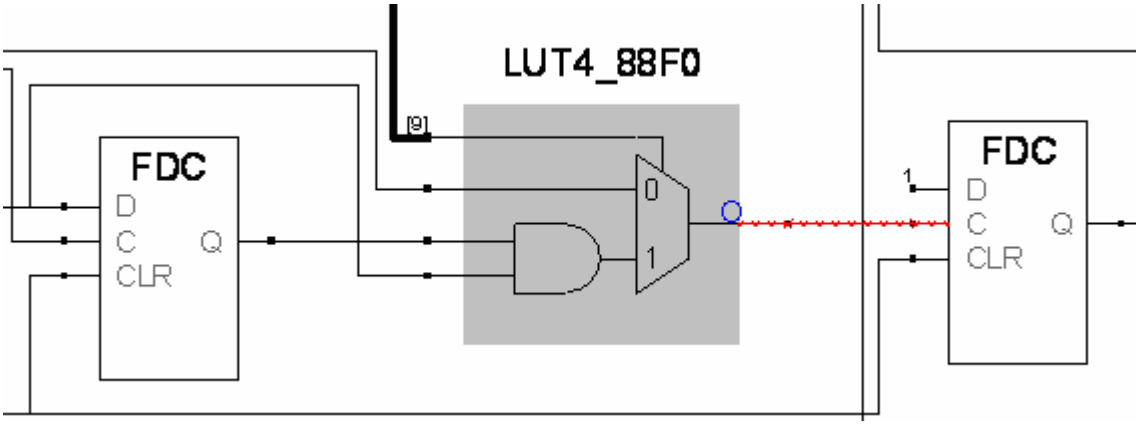

Figure 4.21: Unresolved muxed clock arising due to edge detection circuitry.

wsusunit: Net wrtcclk expands and looks like: such clock gating could be solved by select I0 of the BUFGMUX shown below Figure 4.22.

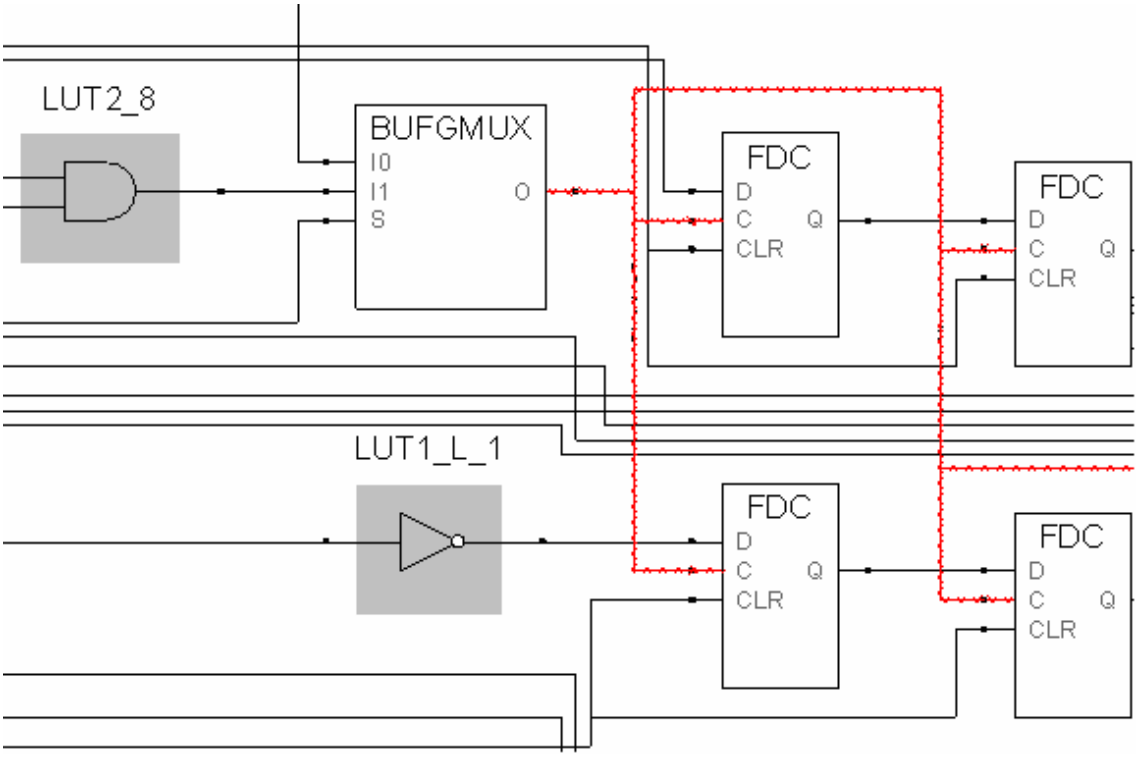

Figure 4.22: Muxed clocks resolved by tying "select" to one of the inputs.

xrrtcunit: Muxed clocks and the MUX is used to select one of the clocks between IOSC32S (actual silicon clock) and IOSC32F (testing clock) as shown in Figure 4.23.

Set wu1_wufastrst to 0, rgunit_irtestb to 1, dt_allclcokcen to 0 in order for the MUX to always select the IOSC32s clock.

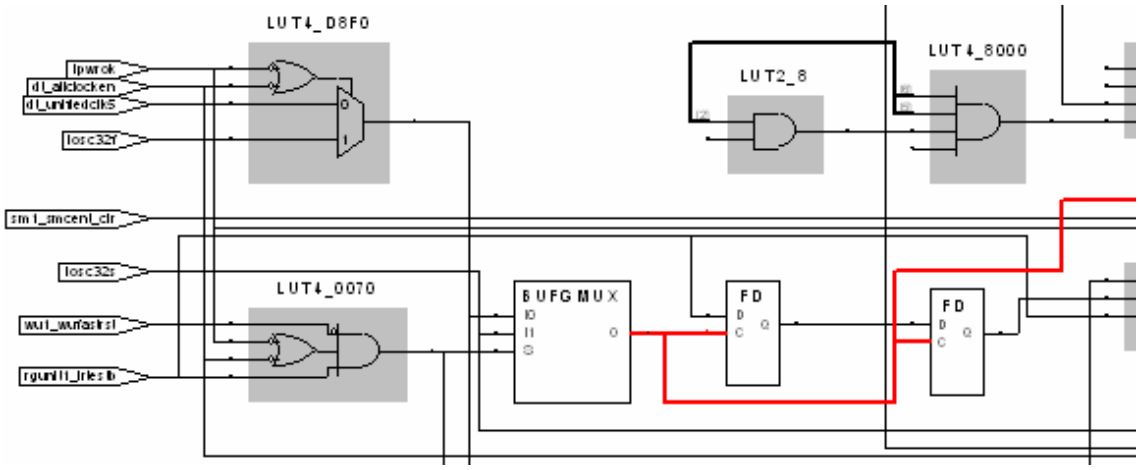

Figure 4.23: Another example of resolving muxed clocks by tying of the "select" pin.

Clock gating assessment leads to several observations including circuits used for resolving metastabilty. Some of the gated clocks are converted by the Synplicity's tool itself. Certain others are resolved by disabling the scan chain and hence clock gating is removed.

There are some more units that have gated clocks which are still unresolved partially since they have Muxed clocks, that is these clocks go to muxes and such clocks can not be converted by the tool. They do not have any test signals that could be disabled. Such clocks need further analysis and have to be resolved at either a higher level of hierarchy or by changing the RTL.

4.8 Metastabilty and its Solutions

Flip Flops have two well defined states, 0 and 1. Under certain circumstances the state can be between these two, and more than one transition in the output can occur in one clock cycle, then, the flop is said to be in metastable state. It typically occurs when the hold time or setup time is violated or when the transition occurs in small

windows where the flop still accepts new inputs. Cascaded flops with a common clock or synchronizer circuits can resolve metastabilty to a certain extent.

4.8.1 Multi-Stage Synchronizer

Adding a second flip flop to the design will reduce the chance of the output going metastable. The output from the first flip flop may go valid, before the second flip flop is clocked. Adding yet another flip flop will reduce the probability that its output will be even more unstable as shown in Figure 4.24 and Figure 4.25.

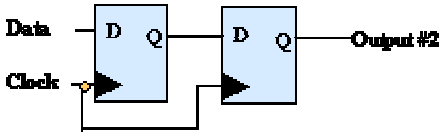

Figure 4.24: Circuits resolving metastability.

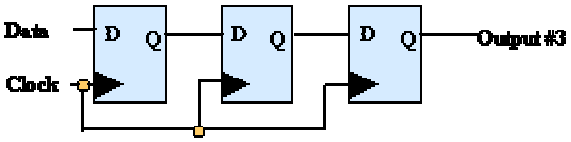

Figure 4.25: A more stable edge detection circuit to avoid metastability.

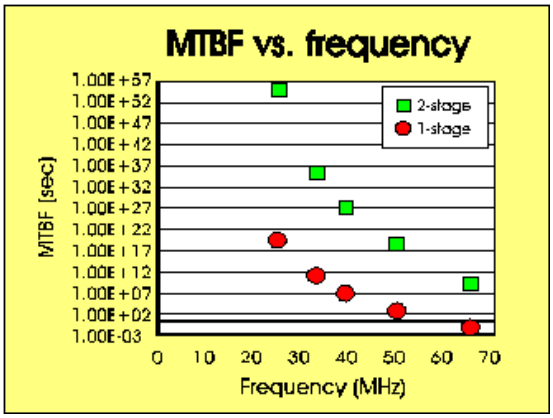

Figure 4.26: Graph depicting MTBF vs Frequency for a one and two stage synchronizers .

The graph shown in Figure 4.26 above details the difference in MTBF between a single and dual stage synchronizer.

4.8.2 Metastability Equations

This section discusses the equations governing metastability namely the mean time between failures.

To determine how often a flip flop will go to an undefined state, plug the data into one of these equations [31] to calculate the MTBF.

$$MTBF = \frac{\exp\left(tr / \Gamma\right)}{F_D \cdot F_C \cdot T_P} \tag{4.1}$$

$$MTBF = \frac{\exp\left(\dfrac{(tr+1)}{(F_c - 1)} \Big/ \Gamma\right)}{F_D * F_C{}^2 * T_P{}^2} \tag{4.2}$$

where, $MTBF$ is the Mean Time Between Failure, F_D is the Data Frequency, F_C is the Clock Frequency, T_P is the Flip Flop Propagation Delay, tr is the Resolve Time, dt is the Delay Time between Clocks, tsu is the Device Setup Time, and Γ is the Flip FLOP Resolution Time.

Usually, the faster the flip flop used, the better the MTBF for a given circuit. The faster device families have lower set-up and hold times. This reduces the window of occurrence and hence reduces metastabilty.

CHAPTER 5

CONCLUSIONS

The large number of transistors and the increasing high complexity in ASICs has made designs vulnerable to errors. The tremendous increase in NRE (Non Recurring Engineering) expense has led to errors being unaffordable at the fabrication stage. This leads to an increased emphasis being laid on design verification to check for the timing requirements and functionalities. Conventional verification techniques such as simulations have become less efficient and accurate since they run at speeds much less than original real world ASICs. Commercial emulators are too expensive and complex, although they run at higher speeds than simulations do but still not comparable to ASICs. FPGA based prototyping is the best answer to these problems since they can run at much higher speeds than any other verification technique available today.

With the advent of high density, high performing FPGAs, prototyping has become the best way to verify the design at very close to real time speeds. Many FPGA vendors like Xilinx , Altera etc provide the designers with high density FPGAs which can accommodate millions of transistors. In this project used Xilinx Virtex-4 family FPGAs to attain maximum I/O availability and high density of 200,000 LUTs. Major blocks of the network processor namely MCH, ICH, Gigabit Ethernet, SSM (security module), NEX were synthesized by converting their RTL to eRTL. Some units are

small enough to fit on a single FPGA like the CRU, which is instantiated on all FPGAs. Certain others like MCH and ICH are huge and need to be partitioned. By Using Synplicity's advanced tools like Certify one can partition the design either manually or by using the automatic partitioner called Quick Partitioner. The automatic one does not give the designer enough liberty to partition the design according to their specific requirements. So the design was manually partitioned by looking at the areas of each unit of MCH and the pins crossing the interface. Each sub unit was synthesized and through Certify their areas and pin counts were estimated. After doing an intelligent judgment on partition the pin counts still exceed the maximum allowed limit. Pin count can be brought down by time multiplexing the signals with Certify's feature called Certify Pin Multiplier. Most of the multiplexing was either 8:1 or 16:1.

Multiplexing can not be enforced on all signals, only certain signals can qualify for CPM. The qualification criteria is that the signal should not go to a black box, it should not be a top level port of the design and it must go to a sequential element at the destination FPGA. If the third criterion is not met it can be worked around to get satisfied whereas if the signal falls in the first two categories it would fail to get multiplexed. A non qualified signal not reaching a sequential element is made qualified by moving some logic such that the signals at the interface have sequential elements at the receiving port of the FPGA.

The design is then ready to be re-synthesized for placement and routing. If the design has any gated clocks which is very likely then they need to be fixed before place and route otherwise the design would fail in the routing stage. In FPGAs gated clocks

can not be handled and need to be resolved. Synplicity's tools offer gated clock fixing but they can only handle certain clock gating like AND and OR gates but not MUXes and flip flops on the clock gating. To handle such complex gating one needs to manually fix the RTL. In this project some MUXed clocks were resolved by disabling the scan chain in the test mode signals. Also by selecting the actual silicon clock over test mode clocks can help fix this issue.

5.1 Contributions

The majority of the work focused on the MCH and ICH, including conversion to eRTL, synthesizing, partitioning, pin multiplexing and fixing the gated clocks. All other units in the design were also synthesized namely CAS, NEX, ME, GBE etc. A lot of work was done in manually pin multiplexing the interface across FPGAs and fixing the gated clocks. Clock gating is totally resolved in the ICH except for the units which have edge detection circuitry. Furthermore, gated clocks have to be fixed in some sub units of the MCH. All dual port memories used in the processor were translated to block RAMs in order to make it compatible with FPGAs.

5.2 Future Work

The ICH and MCH have to be run through XILINX ISE in order to place, route and implement the design for configuration of FPGAs. More work can be done towards fixing the unresolved gated clock issues especially in the units where there is no logic but only I/Os are defined. Also, lot of analog blocks of the ICH has gated clocks for which we need to find a solution.

The next version of NEX cluster needs to be synthesized. This requires converting some logic to eRTL especially the user defined primitives (UDP). The UDPs which are in tabular format are non-synthesizable and have to be re-written in verilog to be synthesized. Further, the I/O pad logic has to be made synthesizable. The unit then has to be placed and routed and configured on FPGAs.

Thus the full chip emulation would be implemented and configured on multiple FPGAs for verification purposes. The successful prototyping will be a major milestone in achieving the verification stage of the processor's functionality.

REFERENCES

[1] J. Tai, Protyping ASICs with high Density PLDs. *The Syndicated-- From Synplicity* [Online]. 3*(1),* Available: http://www.synplicity.co.jp/literature/syndicated/pdf/syndicated_v3_i1.pdf

[2] T. Pasquino, "Tolapai: Kodiak Island Emulation System Hardware Design Verification/Test/Bring-up Plan," 2005.

[3] J. Garrison, Prototyping ASICs Using Xilinx FPGAs and Certify. *New Products Software--From Xilinx* Available: http://www.xilinx.com/xcell/xl32/xl32_24.pdf

[4] Z. Rumi and T.E. Willis, "Tolapai External Architecture Specifications," Intel Inc., Intel Inc, Hudson, MA, May, 2005.

[5] G. Torres, October 10, 2005) How Gigabit Ethernet Works. Available: http://www.hardwaresecrets.com/article/231

[6] R. Dordick, Chip Verification On a Large Scale. *Featured Concept--From Think Research by IBM.* Available: http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/verify296.html

[7] J. Gallagher, (2006, Verification Techniques: Going Beyond Simulation. *Chip Design Magazine* Available: http://www.chipdesignmag.com/display.php?articleId=333

[8] S. Walters, "Computer-aided Protyping for ASIC-based Systems," *IEEE Design and Test of Computers,* vol. 8, pp. 4-10, 1991 June.

[9] R. Fulton, System Emulation and Rapid Prototyping Using Xilinx FPGAs. *Design Hints--From Xilinx.* Available: http://www.xilinx.com/xcell/xl30/xl30_24.pdf

[10] K. Morris, June 8, 2004.) Virtex-4 Xilinx Details its Next Generation. *TechFocus Media--An FPGA and Programmable Logic Journal* Available: http://www.fpgajournal.com/articles/20040608_virtex4.htm

[11] C. Chang, K. Kuusilinna, B. Richards and R.W. Brodersen, "Implementation of BEE: a real-time large-scale hardware emulation system," in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays* 2003, pp. 91-99.

[12] Anonymous "Virtex-4 Family Overview," in *Virtex-4 FPGA Handbook,* San Jose, CA, USA: Xilinx Inc., 2004, pp. 21-22, 23,24, 25, 26.

[13] P.H. Kelly, K.J. Page and P.M. Chau, "Rapid Prototyping of ASIC based Systems," in *Proceedings of the 31st annual conference on Design automation,* June 1994,

[14] J. Varghese, M. Butts and J. Batcheller, "An Efficient Logic Emulation System," *IEEE Transactions on VLSI Systems,* vol. 1, pp. 171-174, June 1993.

[15] J. Babb, R. Tessier and A. Agarwal, "Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators," in *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines,* 1993, pp. 7.

[16] M.L. Dahl, "An Implementation of the Virtual Wires Interconnect Scheme," M.S. Dissertation/Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, Feb 1994.

[17] Anonymous "Certify User Guide- ASIC protyping with FPGAs," Synplicity Inc., Sunnyvale, CA, USA, Nov., 2005.

[18] *Tech Online--From Xilin Inc.* Available: http://www.techonline.com/community/member_company/member/896/content_39093

[19] Accelerating ASIC Verification Through FPGA Prototyping. *From The Syndicated--A Synplicity Web Resource.* Available: http://syndicated.synplicity.com/Q206/altera.html

[20] J. Ray and J.C. Hoe, "Protyping, verification, and test: High-level modeling and FPGA prototyping of microprocessors," in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays,* 2003,

[21] M. Wannemacher, M. Munteanu, P. Sacha and R. Singer, "Taking the Best Out of Two worlds: Prototyping and Hardware Emulation," *Philips Semiconductors,*

[22] Intel Network Processors- For Communications and Networked Embedded Applications. Available: http://www.intel.com/design/network/products/npfamily/index.htm

[23] J. Kim Y., "Processor Silicon Implementation Plan Approval," Intel Massachusetts Inc.,., Hudson, MA, USA, May 4, 2005.

[24] Intel I/O Controller Hub 6 (ICH6) Family. (2004, June) Available: http://whitepapers.techrepublic.com.com/abstract.aspx?promo=50002&docid=91563

[25] 8-bit microprocessor mega function. *CAST--From Altera.* Available: http://www.cast-inc.com/cores/cz80cpu/cast_cz80cpu-a.pdf

[26] J. Gallagher, (2006, Verification Techniques: Going Beyond Simulation. *Chip Design Magazine* Available: http://www.chipdesignmag.com/display.php?articleId=333

[27] ASIC Prototyping Using Off the Shelf FPGA Boards. Available: http://www.synplicity.com/literature/whitepapers/pdf/proto_wp06.pdf

[28] Xilinx Virtex-4 Devices. Available: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/overview/index.htm

[29] H. Vince, (1999, Programmable Device or Gate Array. *System Design--By EE Times* Available: http://www.eetimes.com/editorial/1999/asicfpga9901.html

[30] B. Caslis, Quick Partitioning Technology. *The Syndicated--From Synplicity* [Online]. 1*(3),* Available: http://www.synplicity.com/syndicated/pdf/QPT_v1_3.pdf

[31] C. Wellheuser, "Metastability Performance of Clocked FIFOs," Texas Instruments Inc., Dallas, Texas, USA, Tech. Rep. SCZA004A, 1996.

BIOGRAPHICAL INFORMATION

Sharada Vajja received her dual degree in Master of Science Physics and Bachelor of Engineering Electrical and Electronics from The Birla Institute of Technology and Science, Pilani, Rajasthan in the year 2003 where she worked on various projects in the digital VLSI design and synthesis. In partial fulfillment of that integrated dual degree program she worked as a Co-op and Research Scholar at Infineon Technologies AG, Munich, Germany. There she developed a high speed multiplier using VHDL. She was also an intern for another six months at ST Microelectronics, Noida. She then moved on to pursue her Master of Science in Electrical Engineering from The University of Texas at Arlington in August 2004. She is awarded the graduate school fellowship and has been a teaching assistant for undergraduate classes. She began working as a Graduate Technical Intern at Intel Inc. at Hudson, MA since May 2005 till August 2006. Her current research interests include low power VLSI design, interconnect capacitances at small die size and emulation of ASIC on FPGAs.