IMPLEMENTATION AND TESTING OF TRAFFIC GENERATOR FOR A HIGH SPEED SERIAL DATA (HSSD) PROTOCOL

Thesis submitted in partial fulfillment of the requirement for the award of degree of

MASTER OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

Submitted By

Pankaj Kumar Pipersaniya

80761018

Under the Guidance of

Mr. Manoj Kumar Bansal
Project Leader
WMM
ST Ericsson Pvt. Ltd.

Dr. Rajesh Khanna
Professor
ECED Department
Thapar University

Electronics and Communication Engineering Department

Thapar University, Patiala-147004 (INDIA)

June 2009

DECLARATION BY THE INDUSTRY

We hereby declare that the contents of the project work entitled "**Implementation and Testing of Traffic Generator for A High speed serial data protocol**" have been subjected to numerous modifications. This was a protective action and was necessary to preserve the confidentiality of the actual matter and maintain security over the company's proprietary content.

Mr. Manoj Kumar Bansal
Project Leader
WMM
ST Ericsson Pvt. Ltd.

CERTIFICATE

I hereby declare that the work, which is being presented in the thesis, entitled "**Implementation and Testing of Traffic Generator for a High speed serial data protocol**" in partial fulfillment of the requirements for the award of degree of Master of Engineering in Electronics & Communication branch at Electronics & Communication Engineering Department of Thapar University, Patiala is an authentic record of my own work carried out under the guidance of Dr. Rajesh Khanna.

This thesis work has been carried out in ST Microelectronics, Greater Noida, under the guidance of Mr. Manoj Kumar Bansal (Project Leader).

I have not submitted the matter presented in the thesis for award of any other degree of this or any other university.

Pankaj

(Pankaj Kumar Pipersaniya)

This is to certify that the above statement made by the student is correct and true to best of our knowledge.

Mr. Manoj Kumar Bansal
Project Leader
WMM
ST Ericsson Pvt. Ltd.

Dr. Rajesh Khanna
Professor
ECED Department
Thapar University

Countersigned by

Head & Professor
ECED, Thapar University
Patiala

Dean of Academic Affairs
Thapar University,
Patiala

ii

ACKNOWLEDGEMENT

Any fruitful effort in a new work needs a direction and guiding hands that shows the way. I take this opportunity to express my sincere gratitude to D**r Rajesh Khanna (Professor, ECED)**, Thapar University, Patiala for his suggesting new ways for implementing my ideas by his expert guidance throughout my work.

It is my proud, privilege and pleasure to bring my indebtedness and warm gratitude to **Mr. Manoj Bansal** (Project Leader), ST Ericsson Pvt. Ltd. for his support during my project work. I have warm regards for **Mr. Rajiv Kumar, Mr. Vinay Pandey, Mr. Mahendra Singh, Mr. Suman Kumar** and **Mr**. **Praveen Gupta**, ST Ericsson Pvt. Ltd., who have obliged me with their time to time guidance.

I express my sincere gratitude to Head of Department **Dr A.K Chatterjee** for giving me the opportunity to accomplish the project at ST Ericsson Pvt. Ltd. and for his efforts and encouragement throughout the work. He has been a continuous source of inspiration all the time.

Finally, my thanks to everyone who has in some way or other helped me in completing this project successfully. I should not fail to mention my parents who have always been a source of inspiration. I am grateful to my friends for their valuable support and help.

ABSTRACT

As 3G became the general tendency of mobile industry, new mobile handsets are challenged to show more and more functions. In order to process high resolution images, the high-speed data transfer technology that was originally used in communication system or server system started to be adopted in mobile handsets.

This thesis describes High speed serial data protocol for high speed data transfer between camera and ISP (Image Signal Processor). Serial interface shows excellence in terms of power, bandwidth and EMI in comparison to the conventional parallel interface.

A Traffic Generator has made for real time implementation of protocol. The traffic generator can generate and transmit all types of image data. High speed serial data transmission is done by using High speed Serial Data Protocol as protocol and PHY IC as high speed serial interface between transmitter and receiver.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

EMI	Electromagnetic Interference
3G	Third Generations
PCs	Personal Computers
ISP	Image Signal Processor
HSSD	High Speed Serial Data
PHY	Physical Layer
SoT	Start of Transmission
EoT	End of Transmission
LLP	Low Level Protocol
FPGA	Field Programmable Gate Array
PLD	Programmable Logic Devices
CLB	Configurable Logic Block
RAM	Random Access Memory
CPLD	Complex Programmable Logic Device
ASIC	Application Specific Integration Chip
SRAM	Static Random Access Memory
FF	Flip Flop
RTL	Register Transfer Logic
HDL	Hardware Description Language
VHDL	Very High Speed Integrated Chip Hardware Description Language
ISE	Integrated Software Environment
XST	Xilinx Synthesis Technology
EDIF	Electronic Data Interchange Format
AHB	Advanced High Performance Bus
CSR	Control & Status Registers
PPI	PHY Protocol Interface
PLL	Phase Locked Loop

CHAPTER 1
INTRODUCTION

1.1 Introduction

Entering the era of the 3G (Third Generation) cellular handset, mobile handset design engineers are faced with the high bandwidth requirements for camera and display. When it comes to 3G handsets, (not only for the particular 3G, but also for specific handsets which can support moving pictures or high-resolution camera functions) consumers expect the functionality just the same as what multimedia PCs have. Multimedia mobile handset is required to support high-speed transfer of content rich multimedia. In addition, ISP (Image Signal Processor) or Multimedia Chipset demands high-speed clocks, low-EMI and high bandwidth in order to meet the expectations. The demand for low power and low EMI high-speed bus interfaces has exponentially increased. Also, the amount of data is increasing exponentially as the resolution of camera and the size of video increase. Therefore, high-speed interfaces between camera module and ISP are emerging. Since the LVCMOS technology which is currently used in the parallel bus is inefficient in terms of EMI and Power efficiency, chipset designers need to think about the ways to transfer high-speed data on serial buses with lower power consumption. In [1, 2] comparison between conventional Parallel Interface and various high speed Serial Interfaces is given and it is shown that Serial Interfaces consume lower power and generate lower EMI in short distances. There are several open standards (SMIA, MIPI, MDDI etc) that aim to be the next generation camera and display data transfer standard for mobile handset devices. Especially these serial interfaces are most suitable for mobile devices because they consume lower power and generate lower EMI in short distances.

High Speed Serial Data protocol (HSSD) described in this Thesis is an emerging serial solution of higher bandwidth with low power consumption for mobile applications. The implementation of this standard protocol is used to transfer the image data from sensor to host processor via high speed serial links. The aim of the thesis was to create a Traffic

Generator for the transmitter side of the HSSD protocol. The Traffic Generator would be able to produce all supported data formats and stimulus that a Camera Sensor can typically produce. The generator is to be used for interoperability testing for HSSD receiver devices ensuring their capability to work with various compatible sensor devices.

It will also cover the compliance check of each stages of protocol with respect to defined standard. HSSD Protocol description is defined and standardized by a working group**.**

1.2 Objective of Thesis

The objective of thesis is divided in five parts:

- **Analysis of High Speed Serial Data protocol:** Analysis of High Speed Serial Data protocol includes analysis of functional layers of High Speed serial data protocol. It will also include the analysis of the hardware requirements, implementation platform understanding and validation aspects.

- **Understanding & Hands-on with EDA tools & FPGA devices:** Understanding of EDA tools flow for logic synthesis, simulation and implementation. These tools will include SynplifiyPro, NC-Sim and Xilinx ISE and Altera Quartus.

- **Defining RTL architecture for Traffic generator:** It consists of design and development of RTL modules for Traffic Generator in VHDL language. This RTL will have the provision for generation of different types of supported image data used in mobile application. This RTL flow will be described through the State Machines and Flow diagrams.

- **RTL Verification:** Verification of developed RTL using test-benches.

- **Mapping RTL to FPGA and execution of application:** Mapping of RTL over FPGA and system debugging. It will also include execution of software test cases developed in application environment.

1.3 Organization of Thesis

This thesis is divided into six chapters and organized as follows:

- Chapter 2 describes High Speed Serial Data protocol overview and layer definition of protocol.
- Chapter 3 describes the work flow of Traffic Generator RTL development. It is also gives information of FPGA devices, and EDA tools used in this thesis.
- Chapter 4 describes the design architecture of Traffic Generator. It is also describes each module of Traffic Generator RTL.
- Chapter 5 describes the resulting waveform, generated by simulation of RTL.
- Chapter 6 deals with conclusion.

1.4 About ST Microelectronics

STMicroelectronics was created in 1987 by the merger of SGS Microelectronica of Italy and Thomson Semiconductors of France. Since its formation, ST has grown faster than the semiconductor industry as a whole and it has been one of the world's Top Ten semiconductor suppliers since 1999. The group totals approximately 50,000 employees, 16 advanced research and development units, 39 design and application centers, 15 main manufacturing sites and 78 sales offices in 36 countries. Corporate Headquarters, as well as the headquarters for Europe and for Emerging Markets, are in Geneva. The Company's U.S. Headquarters are in Carrollton (Texas); those for Asia-Pacific are based in Singapore and Japanese operations are headquartered in Tokyo.

STMicroelectronics was one of the first global industrial companies to recognize the importance of environmental responsibility, its initial efforts beginning in the early 1990s. Since then ST has made outstanding progress; for example, energy consumption per product unit was reduced by 47% between 1994 and 2006 and CO2 emissions have been reduced by 61% over the same timescale. In addition, ST has gone far beyond existing legal requirements in almost completely eliminating the use of hazardous substances such as lead,

cadmium, and mercury. Since 1991, the Company's sites have received more than 100 awards for excellence in all areas of Corporate Responsibility, from quality to corporate governance, social issues and environmental protection. Since December 8, 1994, when ST completed its initial public offering, the Company's shares have been traded on the New York Stock Exchange (NYSE: STM) and on Euronext Paris; since June 1998, ST has also been listed in Milan on Borsa Italiana. The Company now has around 900 million outstanding shares, 71.1% of which are publicly traded on the various stock exchanges. The balance of the shares is held by STMicroelectronics Holding II B.V. (27.5%), a company whose shareholders are Cassa Depositi e Prestiti and Finmeccanica of Italy, anwqqd Areva of France, and treasury shares (1.4%) held by STMicroelectronics NV.

STMicroelectronics is a global independent semiconductor company and is a leader in developing and delivering semiconductor solutions across the spectrum of microelectronics applications. An unrivalled combination of silicon and system expertise, manufacturing strength, Intellectual Property (IP) portfolio and strategic partners positions the Company at the forefront of System-on-Chip (SoC) technology and its products play a key role in enabling today's convergence trends. ST is one of the world's largest semiconductor companies.

According to the latest industry data, ST is the world's fifth largest semiconductor company with market leadership in many fields. For example, ST is the leading producer of application-specific analog chips and power conversion devices. It is also the number #1 supplier of semiconductors for the Industrial market and for set-top box application, and occupies leading positions in fields as varied as discrete devices, camera modules for mobile phones and automotive integrated circuits.

The Company's products are manufactured and designed using a broad range of fabrication processes and proprietary design methods. To complement this depth and diversity of process and design technology, the company also possesses a broad intellectual property portfolio that it has used to enter into cross-licensing agreements with many other leading semiconductor manufacturers.

CHAPTER 2
High Speed Serial Data Protocol

2.1 Introduction

Demand for increasingly higher image resolutions is pushing the bandwidth capacity of existing host processor-to-camera sensor interfaces. Common parallel interfaces are difficult to expand, require many interconnects and consume relatively large amounts of power. Emerging serial interfaces address many of the shortcomings of parallel interfaces while introducing their own problems.

High Speed Serial Data protocol provides the mobile industry a standard, robust, scalable, low-power, high-speed, cost-effective interface that supports a wide range of imaging solutions for mobile devices. It defines standard data transmission and control interfaces between transmitter and receiver. Data transmission interface is unidirectional differential serial interface with data and clock signals.

2.2 High Speed Serial Data protocol Layer Definition

Figure defines the conceptual layer structure used in High Speed Serial Data protocol. The layers can be characterized as follows:

• **PHY Layer:** The PHY Layer specifies the transmission medium (electrical conductors), the input/output circuitry and the clocking mechanism that captures “ones” and “zeroes” from the serial bit stream. This layer defines the characteristics of the transmission medium, electrical parameters for signaling and the timing relationship between clock and data Lanes. The mechanism for signaling Start of Transmission (SoT) and End of Transmission (EoT) is specified as well as other “out of band” information that can be conveyed between transmitting and receiving PHYs. Bit-level and byte-level synchronization mechanisms are included as part of the PHY.

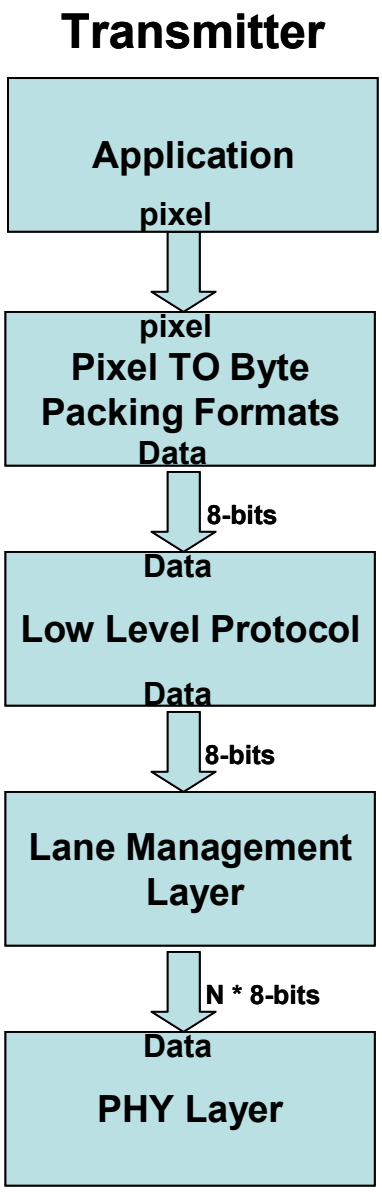


Figure 2.1 High Speed Serial Data Protocol Layer Definition

• **Pixel/Byte Packing/Unpacking Layer.** The High Speed Serial Data protocol supports image applications with varying pixel formats from six to twenty-four bits per pixels. In the transmitter this layer packs pixels from the Application layer into bytes before sending the data to the Low Level Protocol layer. In the receiver this layer unpacks bytes from the Low Level Protocol layer into pixels before sending the data to the Application layer. Eight bits per pixel data is transferred unchanged by this layer.

• **Low Level Protocol.** The Low Level Protocol (LLP) includes the means of establishing bit level and byte-level synchronization for serial data transferred between SoT (Start of Transmission) and EoT (End of Transmission) events and for passing data to the next layer. The minimum data granularity of the LLP is one byte. The LLP also includes assignment of bit-value interpretation within the byte.

• **Lane Management.** High Speed Serial Data protocol is Lane-scalable for increased performance. The number of data Lanes may be multiple depending on the bandwidth requirements of the application. The transmitting side of the interface distributes ("distributor" function) the outgoing data stream to one or more Lanes. On the receiving side, the interface collects bytes from the Lanes and merges ("merger" function) them together into a recombined data stream that restores the original stream sequence.

• **Application Layer.** This layer describes higher-level encoding and interpretation of data contained in the data stream. The High Speed Serial Data protocol specification describes the mapping of pixel values to bytes.

CHAPTER 3

Work Flow, FPGA AND EDA TOOLS

3.1 Work Flow

The Figure below shows the wok flow for creating the High Speed Serial Data protocol Traffic Generator.

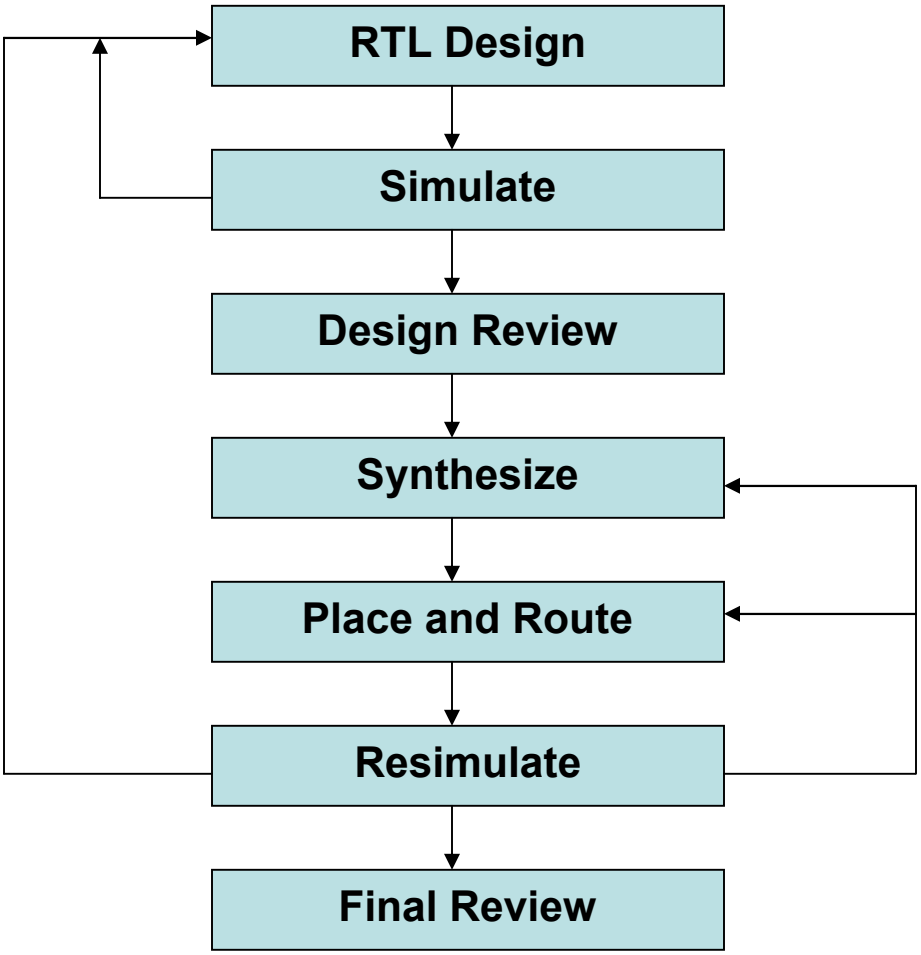


Figure 3.1 Work Flow

First of all an RTL level model of Traffic Generator design and behavioral test benches that model the other systems that surround the Traffic Generator design are written. In the RTL Design various modules are created. VHDL is used as hardware description language to write the RTL of these modules. Then a series of simulation testcases are run, the issues found are fixed and the new design is again subjected to the simulation testcases. This is done until it is decided that the RTL design has been sufficiently tested. The simulation tool used here is NC Launch.

The RTL is then synthesized into a gate-level netlist which can feed into a place-and-route tool to produce a final chip layout or to generate an FPGA programming file. Synthesis tool used here is Xilinx's ISE.

After layout, the chip must be resimulated with the new timing parameters produced by the actual layout. If everything has gone well up to this point, the new simulation results will agree with the predicted results. Otherwise, there are three possible paths to go in the design flow. If the problems encountered here are significant, sections of the FPGA may need to be redesigned. If there are simply some marginal timing paths or the design is slightly larger than the FPGA, it may be necessary to perform another synthesis with better constraints or simply another place and route with better constraints. At this point, a final review is necessary to confirm that nothing has been overlooked.

3.2 FIELD PROGRAMMABLE GATE ARRAY (FPGA)

FPGA's are programmable devices just like PLD's etc, which are used to implement complex digital logic. It is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities. The individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The arrays of logic cells and interconnect form a fabric

of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit.

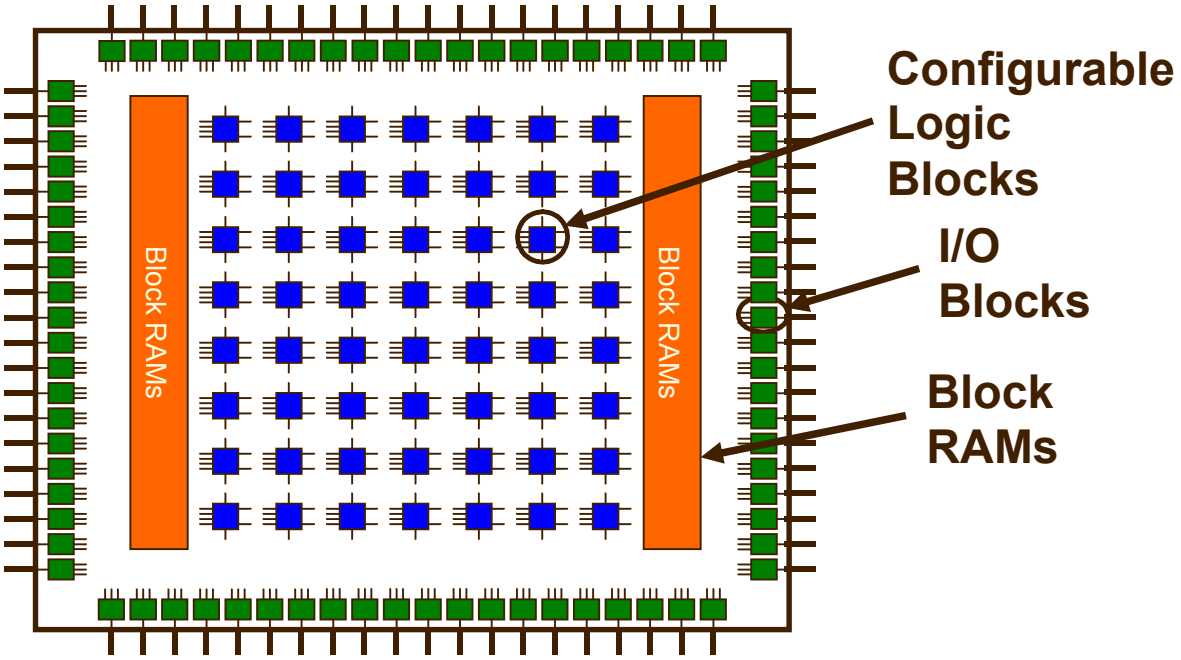


Figure 3.2 FPGA Architecture [7]

Each FPGA vendor has its own FPGA architecture, but in general terms they are all a variation of that shown in the above figure. The architecture consists of configurable logic blocks, configurable I/O blocks, and programmable interconnect. Also, there will be clock circuitry for driving the clock signals to each logic block, and additional logic resources such as ALUs, memory, and decoders may be available. The two basic types of programmable elements for an FPGA are Static RAM and anti-fuses.

3.2.1 Configurable Logic Blocks

Configurable Logic Blocks contain the logic for the FPGA. In large grain architecture, these CLBs will contain enough logic to create a small state machine. In fine grain architecture, more like a true gate array ASIC, the CLB will contain only very basic logic. The diagram in the figure would be considered a large grain block. It contains RAM for creating arbitrary combinatorial logic functions. It also contains flip-flops for clocked storage elements, and

multiplexers in order to route the logic within the block and to and from external resources. The multiplexer also allow polarity selection and reset and clear input selection.

3.2.2 Configurable I/O Blocks

A Configurable I/O Block is used to bring signals onto the chip and send them back off again. It consists of an input buffer and an output buffer with three state and open collector output controls. Typically there are pull up resistors on the outputs and sometimes pull down resistors. The polarity of the output can usually be programmed for active high or active low output and often the slew rate of the output can be programmed for fast or slow rise and fall times. In addition, there is often a flip-flop on outputs so that clocked signals can be output directly to the pins without encountering significant delay. It is done for inputs so that there is not much delay on a signal before reaching a flip-flop which would increase the device hold time requirement.

3.2.3 Programmable Interconnect

Interconnects of an FPGA is very different than that of a CPLD, but is rather similar to that of a gate array ASIC. In Figure a hierarchy of interconnect resources can be seen. There are long lines which can be used to connect critical CLBs that are physically far from each other on the chip without inducing much delay. They can also be used as buses within the chip. There are also short lines which are used to connect individual CLBs which are located physically close to each other. There are often one or several switch matrices, like that in a CPLD, to connect these long and short lines together in specific ways. Programmable switches inside the chip allow the connection of CLBs to interconnect lines and interconnect lines to each other and to the switch matrix. Three-state buffers are used to connect many CLBs to a long line, creating a bus. Special long lines, called global clock lines, are specially designed for low impedance and thus fast propagation times. These are connected to the clock buffers and to each clocked element in each CLB. This is how the clocks are distributed throughout the FPGA.

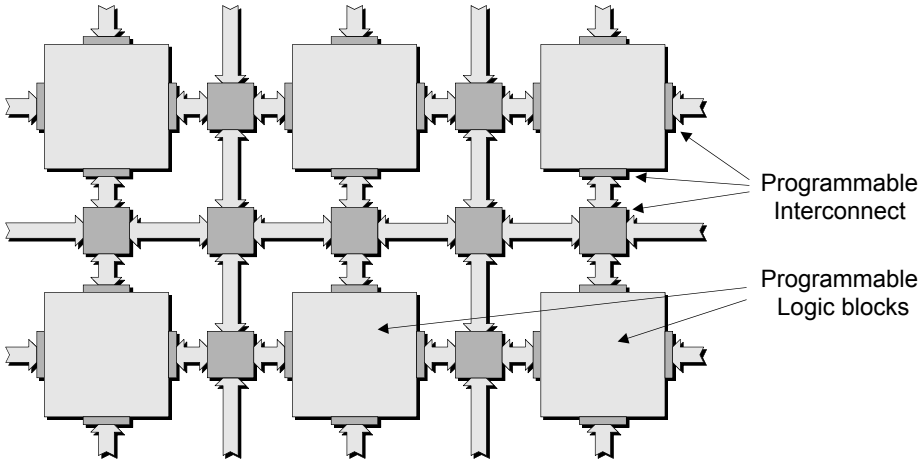


Figure 3.3 FPGA Programmable Interconnect [7]

3.2.4 Clock Circuitry

Special I/O blocks with special high drive clock buffers, known as clock drivers, are distributed around the chip. These buffers are connected to clock input pads and drive the clock signals onto the global clock lines described above. These clock lines are designed for low skew times and fast propagation times. Synchronous design is a must with FPGAs, since absolute skew and delay cannot be guaranteed. Only when using clock signals from clock buffers can the relative delays and skew times are guaranteed.

3.2.5 Small vs. Large Granularity

Small grain FPGAs resemble ASIC gate arrays in that the CLBs contain only small, very basic elements such as NAND gates, NOR gates, etc. The philosophy is that small elements can be connected to make larger functions without wasting too much logic. In a large grain FPGA, where the CLB can contain two or more flip-flops, a design which does not need many flip-flops will leave many of them unused. Unfortunately, small grain architectures require much more routing resources, which take up space and insert a large amount of delay which can more than compensate for the better utilization.

3.2.6 SRAM vs. Anti-fuse Programming

There are two competing methods of programming FPGAs. The first, SRAM programming, involves small Static RAM bits for each programming element. Writing the bit with a zero turns off a switch, while writing with a one turns on a switch. The other method involves anti-fuses which consist of microscopic structures which, unlike a regular fuse, normally make's no connection. A certain amount of current during programming of the device causes the two sides of the anti-fuse to connect. The advantages of SRAM based FPGAs is that they use a standard fabrication process that chip fabrication plants are familiar with and are always optimizing for better performance. Since the SRAMs are reprogrammable, the FPGAs can be reprogrammed any number of times, even while they are in the system, just like writing to a normal SRAM. The disadvantages are that they are volatile, which means a power glitch could potentially change it. Also, SRAM based devices have large routing delays. The advantages of Anti-fuse based FPGAs are that they are non-volatile and the delays due to routing are very small, so they tend to be faster. The disadvantages are that they require a complex fabrication process, they require an external programmer to program them, and once they are programmed, they cannot be changed.

Some Important Desired Properties of FPGAs

- Low power consumption.
- Minimum area consumption.
- Low ON resistance and high OFF resistance.
- Low parasitic capacitance to attached wires.
- Reliability in volume production.

3.3 THE DESIGN FLOW FOR A FPGA DEVICE

In a typical design flow, the design team will first create an RTL level model of their design and write behavioral test benches that model the other systems that surround their design. They will then perform a series of simulations, fixing problems found during simulation, until they decide that the RTL design has been sufficiently tested. The RTL is then synthesized into a gate-level Verilog and VHDL netlist which can feed into a place-and-route tool to produce a final chip layout or to generate an FPGA programming file.
The design flow for A FPGA Device consists of the following steps:

3.3.1 Writing a Specification

The importance of a specification cannot be overstated. This is an absolute must, especially as a guide for choosing the right technology and for making our needs known to the vendor. As specification allows each engineer to understand the entire design and his or her piece of it. It allows the engineer to design the correct interface to the rest of the pieces of the chip. It also saves time and misunderstanding. There is no excuse for not having a specification.
A specification should include the following information:

- An external block diagram showing how the chip fits into the system.
- An internal block diagram showing each major functional section.
- A description of the I/O pins including
    1. output drive capability
    2. input threshold level
- Timing estimates including
    1. setup and hold times for input pins
    2. propagation times for output pins
    3. clock cycle time
- Estimated gate count
- Package type
- Target power consumption

- Target price
- Test procedures

It is also very important to understand that this is a living document. Many sections will have best guesses in them, but these will change as the chip is being designed.

3.3.2 Choosing a Technology

Once a specification has been written, it can be used to find the best vendor with a technology and price structure that best meets our requirements.

3.3.3 Choosing a Design Entry Method

We must decide at this point which design entry method we prefer. For smaller chips, schematic entry is often the method of choice, especially if the design engineer is already familiar with the tools. For larger designs, however, a hardware description language (HDL) such as Verilog or VHDL is used because of its portability, flexibility, and readability. When using a high level language, synthesis software will be required to "synthesize" the design. This means that the software creates low level gates from the high level description.

3.3.4 Choosing a Synthesis Tool

We must decide at this point which synthesis software we will be using if we plan to design the FPGA with an HDL. This is important since each synthesis tool has recommended or mandatory methods of designing hardware so that it can correctly perform synthesis. It will be necessary to know these methods up front so that sections of the chip will not need to be redesigned later on. The synthesis tool is used in this Project is Xilinx's ISE, which is described later in section 3.4 in this chapter. At the end of this phase it is very important to have a design review. All appropriate personnel should review the decisions to be certain that the specification is correct, and that the correct technology and design entry method have been chosen.

3.3.5 Designing the chip

It is very important to follow good design practices. This means taking into account the following design issues.

- Top-down design
- Use logic that fits well with the architecture of the device we have chosen
- Macros
- Synchronous design
- Protect against metastability
- Avoid floating nodes
- Avoid bus contention

3.3.6 Simulating - design review

Simulation is an ongoing process while the design is being done. Small sections of the design should be simulated separately before hooking them up to larger sections. There will be much iteration of design and simulation in order to get the correct functionality. Once design and simulation are finished, another design review must take place so that the design can be checked. It is important to get others to look over the simulations and make sure that nothing was missed and that no improper assumption was made. This is one of the most important reviews because it is only with correct and complete simulation that we will know that our chip will work correctly in our system. The Simulation tool is used in this Project is NCLaunch, which is described later in section 3.5 in this chapter.

3.3.7 Synthesis

If the design was entered using an HDL, the next step is to synthesize the chip. This involves using synthesis software to optimally translate our register transfer level (RTL) design into a gate level design that can be mapped to logic blocks in the FPGA. This may involve specifying switches and optimization criteria in the HDL code, or playing with parameters of the synthesis software in order to insure good timing and utilization.

3.3.8 Place and Route

The next step is to lay out the chip, resulting in a real physical design for a real chip. This involves using the vendor's software tools to optimize the programming of the chip to implement the design. Then the design is programmed into the chip.

3.3.9 Resimulating - final review

After layout, the chip must be resimulated with the new timing numbers produced by the actual layout. If everything has gone well up to this point, the new simulation results will agree with the predicted results. Otherwise, there are three possible paths to go in the design flow. If the problems encountered here are significant, sections of the FPGA may need to be redesigned. If there are simply some marginal timing paths or the design is slightly larger than the FPGA, it may be necessary to perform another synthesis with better constraints or simply another place and route with better constraints. At this point, a final review is necessary to confirm that nothing has been overlooked.

3.3.10 Testing

For a programmable device, we simply program the device and immediately have our prototypes. We then have the responsibility to place these prototypes in our system and determine that the entire system actually works correctly. If we have followed the procedure up to this point, chances are very good that our system will perform correctly with only minor problems. These problems can often be worked around by modifying the system or changing the system software. These problems need to be tested and documented so that they can be fixed on the next revision of the chip. System integration and system testing is necessary at this point to insure that all parts of the system work correctly together. When the chips are put into production, it is necessary to have some sort of burn-in test of our system that continually tests our system over some long amount of time. If a chip has been designed correctly, it will only fail because of electrical or mechanical problems that will usually show up with this kind of stress testing.

3.4 XILINX ISE (INTEGRATED SOFTWARE ENVIRONMENT)

Xilinx ISE controls all aspects of the design implementation flow on the Xilinx FPGA. It consists of a number of built in tools which are integrated together to form one complete package. It is the ideal software for FPGA and CPLD design offering HDL synthesis and simulation, implementation, device fitting, and JTAG programming. There is a Text Editor, which is used to write RTL of Design and Xilinx Synthesis Tool, which is used for synthesize of this RTL Design.

3.4.1 XILINX ISE text editor

Figure 3.4 Xilinx's ISE Text Editor [3]

Xilinx ISE text editor is used to write the Verilog and VHDL source codes. ISE Text Editor enables us to edit source files and user documents. We can access the Language Templates, which is a catalog of ABEL, Verilog, VHDL and User Constraints File templates that we can use and modify in our own design. All the Verilog and VHDL codes were written in the Xilinx ISE editor.

3.4.2 XST (XILINX SYNTHESIS TECHNOLOGY)

Xilinx Synthesis Technology (XST) is a Xilinx application that synthesizes Hardware Description Language (HDL) designs to create Xilinx-specific net list files called NGC files. The NGC file is a net list that contains both logical design data and constraints. The NGC file takes the place of both Electronic Data Interchange Format (EDIF) and Net list Constraints File (NCF) files. All the RTL modules were checked for synthesis using the Xilinx Synthesis Technology i.e. XST. Errors if generated were taken care of and the procedure was repeated till an error free synthesis report was generated.

XST performs the following steps during FPGA synthesis:

- **Analyze / Check Syntax**

  It checks the syntax of the source code.

- **Compile**

  It Translates and optimizes the HDL code into a set of components that the synthesis tool can recognize.

- **Map**

  It translates the components from the compile stage into the target technology's primitive components.

❖ **Synthesizing the design using XST**

The following processes are available for synthesis using XST:

• **View Synthesis Report**

Gives a synthesis mapping and timing summary as well as optimizations that took place

• **View RTL Schematic**

Generates a schematic view of the RTL net list

• **View Technology Schematic**

Generates a schematic view of our Technology net list

• **Check Syntax**

Verifies that the HDL code is entered properly

• **Generate Post-Synthesis Simulation Model**

Creates HDL simulation models based on the synthesis net list

❖ **The RTL / Technology Viewer**

XST can generate a schematic representation of the HDL code that is entered. A schematic view of the code helps to analyze the design by displaying a graphical connection between the various components that XST has inferred. There are two forms of the schematic representation:

• **RTL View** - Pre-optimization of the HDL code.

• **Technology View** - Post-synthesis view of the HDL design mapped to the target technology.

XST supports User Constraint File (UCF) style syntax to define synthesis and timing constraints. This format is called the Xilinx Constraint File (XCF), and the file has an .xcf file extension. XST uses the .xcf extension to determine if the file is a constraints file.

The synthesis report that is generated after synthesis consists of the following four sections:

• "Compiler Report"
• "Mapper Report"
• "Timing Report"
• "Resource Utilization"

- **Compiler Report**

The compiler report lists each HDL file that was compiled, names which file is the top level, and displays the syntax checking result for each file that was compiled. The report also lists FSM extractions, inferred memory, warnings on latches, unused ports, and removal of redundant logic.

- **Mapper Report**

The mapper report lists the constraint files used, the target technology, and attributes set in the design. The report lists the mapping results of flattened instances, extracted counters, optimized flip-flops, clock and buffered nets that were created, and how FSMs were coded.

- **Timing Report**

The timing report section provides detailed information on the constraints that are entered and on delays on parts of the design that had no constraints. The delay values are based on wire load models and are considered preliminary.

- **Resource Utilization**

This section of the report lists all of the resources that were used for the given target technology.
Further steps in the HDL flow consist of "Behavioral Simulation," i.e. to perform a pre-synthesis simulation of this design.

This is followed by "Design Implementation," i.e. to place and route the design and then do post place and route simulation.
The overall goal when placing and routing the design is fast implementation and high quality results. We may not always accomplish this goal.

• Early in the design cycle, run time is usually more important than quality of results. Later in the design cycle, the reverse is usually true.

• If the targeted device is highly utilized, the routing may become congested, and our design may be difficult to route. In this case, the placer and router may take longer to meet our timing requirements.

• If design constraints are rigorous, it may take longer to correctly place and route our design, and meet the specified timing.

❖ **Timing Simulation**

Timing simulation is important in verifying circuit operation after the worst-case placed and routed delays are calculated. In many cases, we can use the same test bench that we used for functional simulation to perform a more accurate simulation with less effort. Compare the results from the two simulations to verify that our design is performing as initially specified. The Xilinx tools create a VHDL or Verilog simulation net list of the placed and routed design, and provide libraries that work with many common Hardware Description Language (HDL) simulators.

Timing-driven PAR is based on TRACE, the Xilinx timing analysis tool. TRACE is an integrated static timing analysis, and does not depend on input stimulus to the circuit. Placement and routing are executed according to the timing constraints that we specified at the beginning of the design process. TRACE interacts with PAR to make sure that the timing constraints we imposed are met. If there are timing constraints, TRACE generates a report based on those constraints. If there are no timing constraints, TRACE can optionally generate a timing report containing:

• An analysis that enumerates all clocks and the required Offsets for each clock
• An analysis of paths having only combinatorial logic, ordered by delay

❖ **Meeting Design Parameters**

The design must:

• Function at the specified speed
• Fit in the targeted device

After the design is compiled, we use the reporting options of the synthesis tool to determine preliminary device utilization and performance. After the design is mapped by the Xilinx tools, we can determine the actual device utilization.

At this point, we should verify that:

• Our chosen device is large enough to accommodate any future changes or additions
• Our design performs as specified

❖ **Estimating device utilization and performance**

We use the area and timing reporting options of the synthesis tool to estimate device utilization and performance. After compiling, use the report area command to obtain a report of device resource utilization. Some synthesis tools provide area reports automatically. The device utilization and performance report lists the compiled cells in the design, as well as information on how the design is mapped in the FPGA. Some reports specify the minimum number of CLBs required, while other reports specify the "unpacked" number of CLBs to make an allowance for routing. For an accurate comparison, compare reports from the Xilinx mapper tool after implementation. Any instantiated components, such as CORE Generator modules, Electronic Data Interchange Format (EDIF) files, or other components that our synthesis tool does not recognize during compilation, are not included in the report file. If we include these components, we must include the logic area used by these components when estimating design size. Sections may be trimmed during mapping, resulting in a smaller design. We use the timing report command of our synthesis tool to obtain a report with estimated data path delays. The timing report is based on the logic level delays from the cell libraries and estimated wire-load models. While this report estimates how close we are to our timing goals, it is not the actual timing. An accurate timing report is available only after the design is placed and routed.

3.5 NCLAUNCH

As a GUI (Graphical User Interface) NC LAUNCH was used to compile, elaborate and simulate the VHDL, Verilog and mixed source codes. Various components from the NCLAUNCH TOOLS were used for the below purpose:

1) NCVLOG was used for compiling Verilog source Codes
2) NCELAB was used for elaboration.
3) NCSIM was used for simulation to generate the various waveforms.

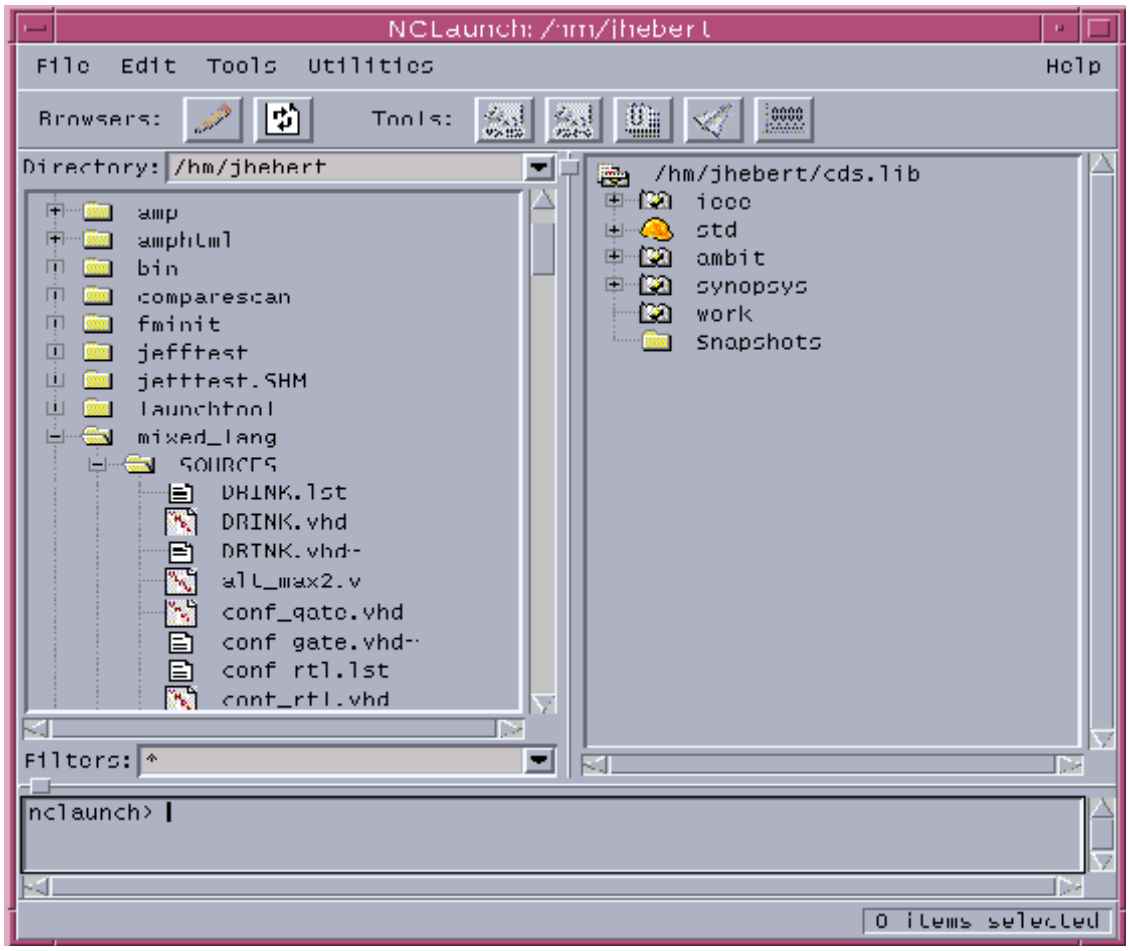


Figure 3.5 NC Launch Main Window [4]

NCLaunch is a graphical user interface that gives a unified view of the files and libraries in the design. The tool provides an easy and consistent way to configure and launch the Cadence simulation tools.

There are several components that need to be considered before, during, and after simulation:

1. The HDL files that make up the design
2. The libraries into which the files are compiled
3. The NC based tools that are available to help with the simulation process.

NCLaunch helps us to manage our files and libraries, and makes us aware of the different NC tools that are available to us.

As the HDL designs grow more complex, we need to manage larger design hierarchies and interact with more extended tool flows. This can make it difficult to design, verify, and deliver products with the maximum number of features possible in a minimal amount of time. As a consequence, increasing our design productivity and efficiently managing and verifying our HDL projects become an important design goal. NCLaunch can help us get our designs into simulation faster, so that we can find the maximum number of problems in the least amount of time.

The NCLaunch tool consists of a single main window that contains multiple browsers, which are integrated with the suite of NC tools. The integrated tools include the following:

1. Compilers: NCVHDL, NCVLOG

The ncvlog or ncvhdl compiler analyzes and compiles Verilog source files or VHDL source files respectively.

2. Elaborators: NCELAB

The elaborator takes the Library Cell and views the name of the top-level HDL design unit(s) as input. The elaborator then constructs a design hierarchy based on the instantiation and configuration information in our design, establishes the connectivity of the design, and computes the initial values for all of the objects in the design.

3. Simulator: NCSIM

The Simulator menu option launches either the ncvlog or ncvhdl simulator, depending on the type of source file that has been selected. The simulators simulate the snapshot file that the elaborator has generated.

CHAPTER 4

Design Architecture of Traffic Generator RTL

In this chapter first of all system architecture is described. The FPGA board in system architecture is used for mapping RTL of the Traffic Generator. The block diagram of Traffic Generator describes overview of Traffic Generator RTL, in which Datastreaming Logic block is main controlling block for controlling the transmission. Then design hierarchy of RTL is explained and Brief detail of each module is also given.

4.1 HSSD Protocol END TO END System Architecture:

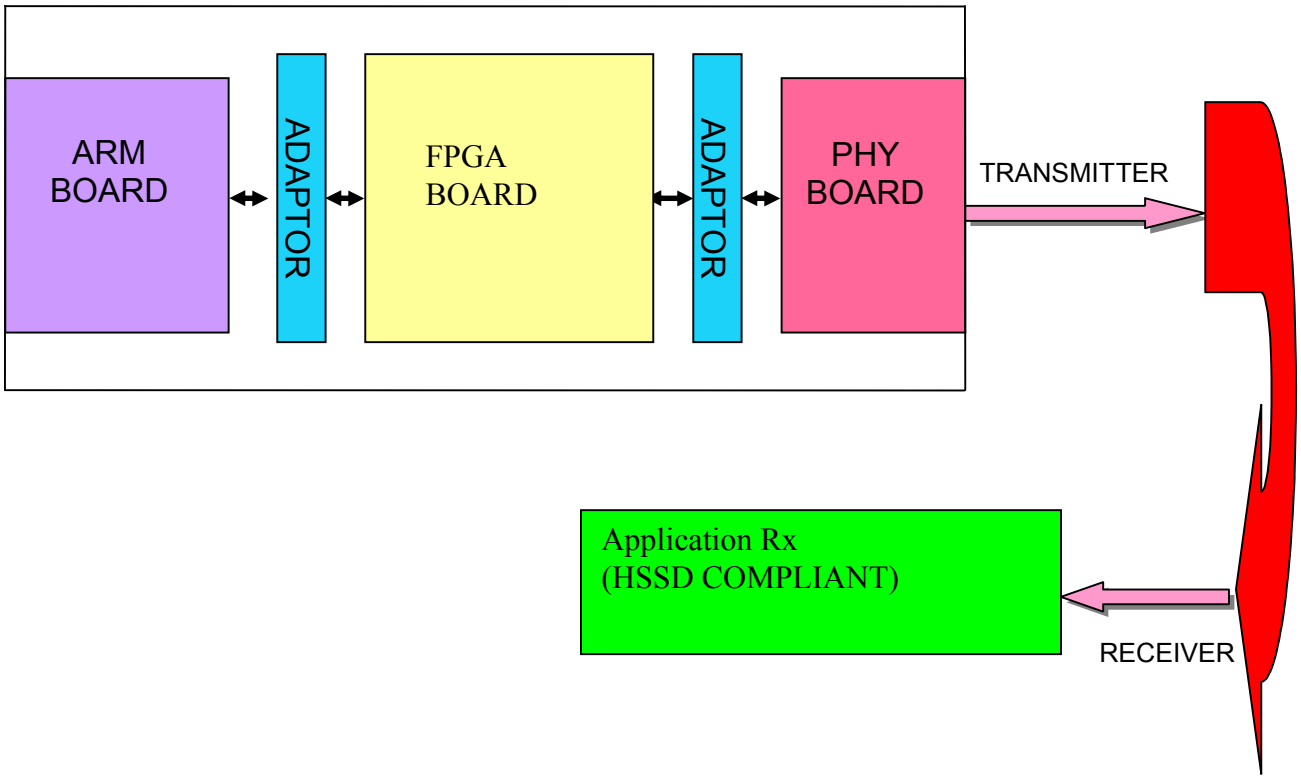


Figure 4.1 HSSD Protocol END TO END System Architecture

The diagram shows a pictorial representation of the HSSD protocol End to End System Architecture.

Figure 4.1 consists of one PHY card configured to transmit the high speed serial data, the FPGA board to map the RTL design of HSSD Protocol Traffic Generator module and ARM Board to control the data stream in end to end system. This setup also consists of two adaptor boards, the PHY adaptor board and the ARM adaptor board. The PHY adaptor board is required to make interconnections between the PHY card and FPGA board. ARM adaptor board is needed to make interconnections between the ARM Board and FPGA board.

In the end to end setup first control signals are sent, after that the memories of Traffic Generator mapped in FPGA Board are written. The control and programming is done by ARM Board using an AHB Interface. Traffic Generator starts transmission of data when it gets control signal start from ARM Board. Transmission is done over a high speed serial link using HSSD Protocol PHY IC. At the Receiver side data should be received correctly.

4.2 Block Diagram of HSSD Protocol Traffic Generator:

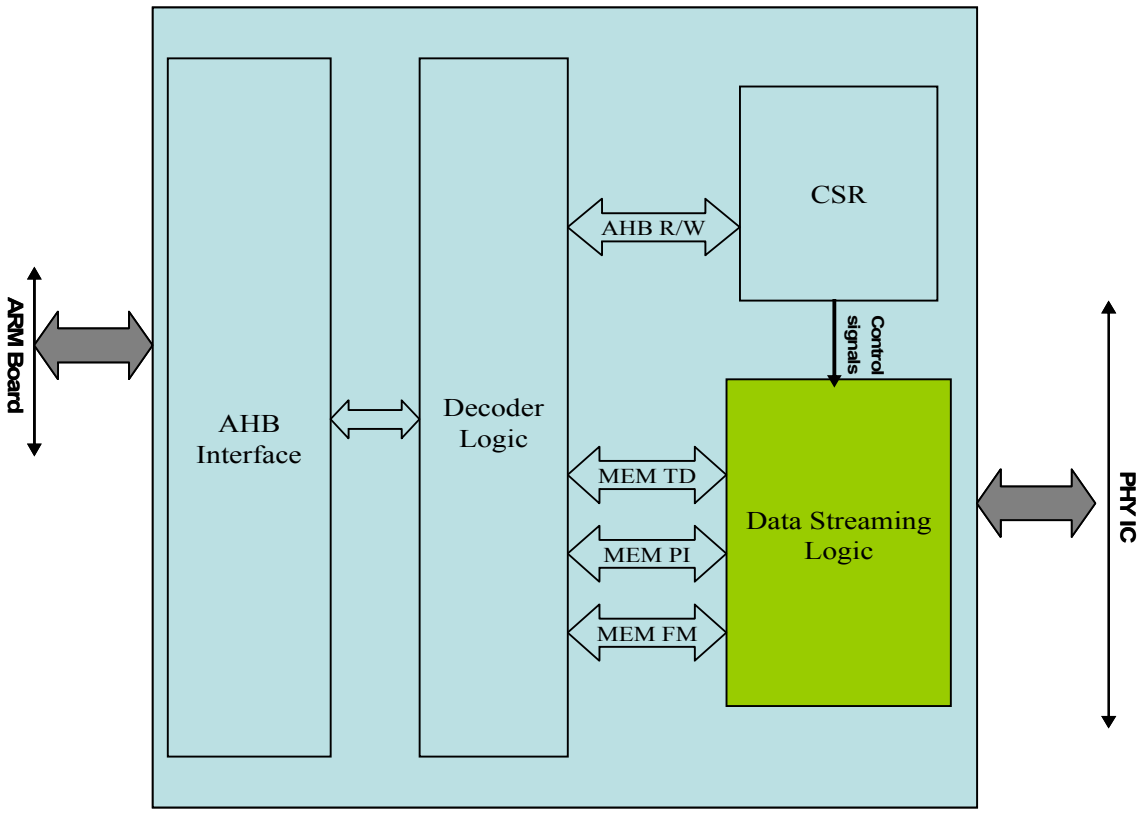


Figure 4.2 Block Diagram of HSSD Protocol Traffic Generator

As shown in figure 4.2 Traffic Generator mainly has following Parts:

1. AHB Interface
2. Decoder Logic
3. Control & Status Registers
4. Data Streaming Logic Top

AHB Interface works as an interface between ARM processor and Traffic generator's modules. It is used to read and write the memories, CSR registers and other control information by ARM Processor. Decoder Logic is used to selecting the registers or memories. Control & Status Registers is the controlling block of the Traffic Generator, It gives the control signals required for transmission to Data streaming Logic block and have the status of various events like start of Transmission, end of Transmission and many more. Data Streaming Logic block is used for organization of packets in frame and transmission of frame over a PHY IC.

4.3 Block Diagram of Data Streaming Logic Top

As shown in figure 4.3 the Data Streaming Logic Top has following blocks:

1. Data Streaming Logic
2. PPI Tx

Data Streaming Logic block also has following blocks:

1. Streaming Flow Controller
2. Traffic Descriptor Memory
3. Packet Index Memory
4. Frame Memory

Data Streaming Logic block uses these three memories to make desired frame format. The Flow controller block controls the frame formation using memories and also controls the data transmission process. PPI Tx block is used for transmission of data over high speed physical interface.

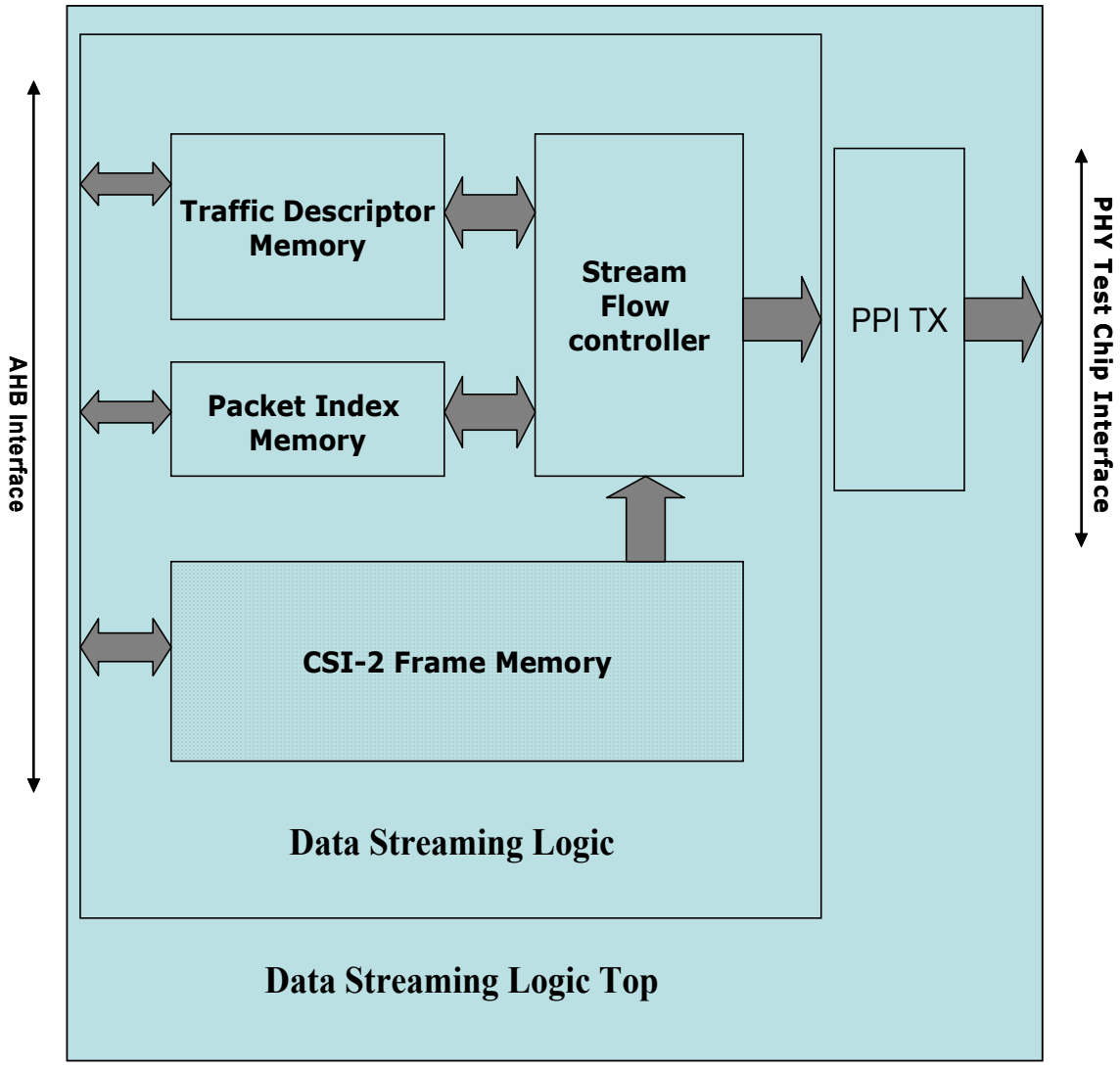


Figure 4.3 Block Diagram of Data Streaming Logic Top

4.4 Design Description of HSSD Traffic Generator

Traffic Generator is top module of HSSD Traffic Generator RTL design .Traffic Generator interfaces with AHB Master Bus Interface and PHY chip. The components of Traffic Generator module are given below:

Components: It has following components

1. AHB Interface
2. Decoder Logic
3. Control & Status Registers Block
4. Data Streaming Logic Top

Figure 4.4 describes the ports of Traffic Generator module (the primary inputs, outputs and inouts of top module).

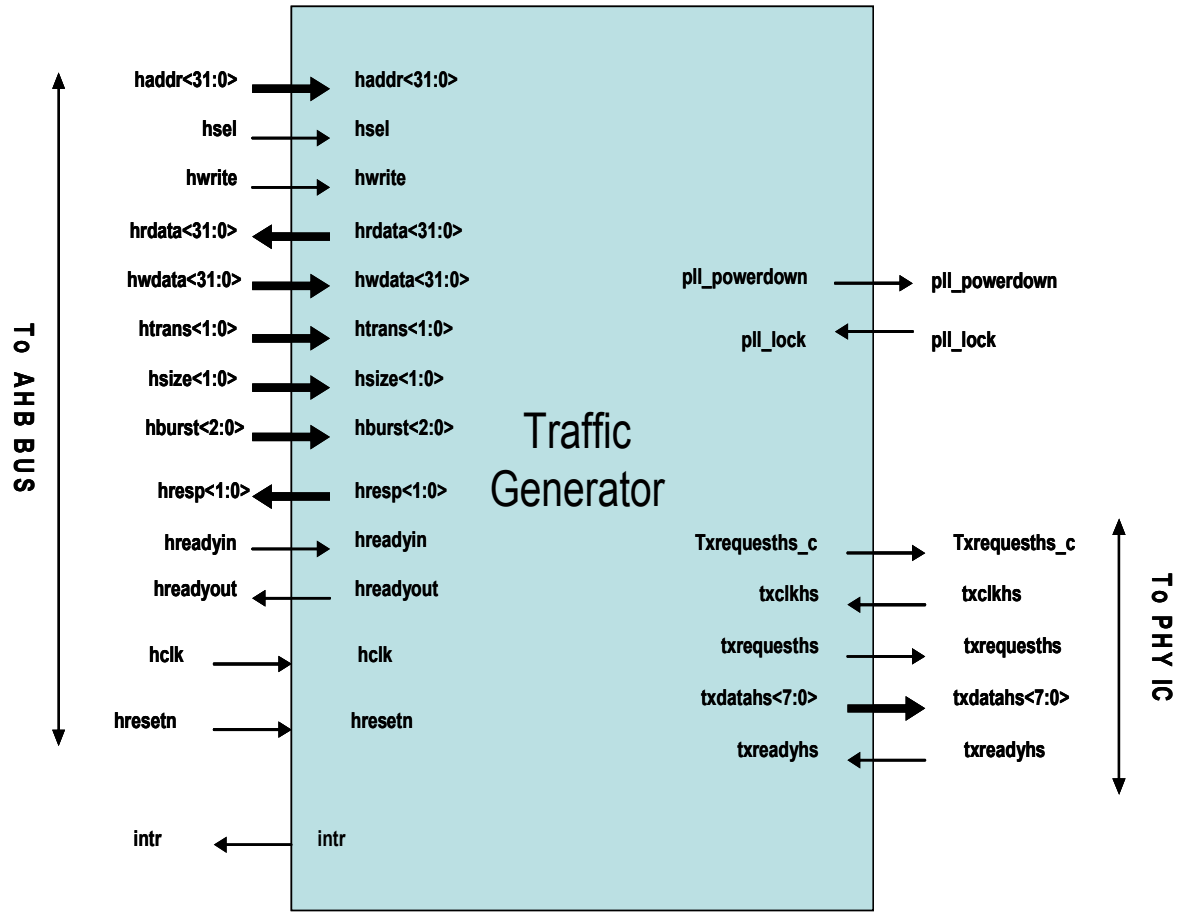


Figure 4.4 Pin Diagram of Traffic Generator

Port list of Traffic Generator is given in Table 4.1.

Port Name (with Width)	Direction	Description
ARM Versatile Board Interface		
haddr<31:0>	Input	AHB Address
Hsel	Input	AHB Peripheral Select
Hwrite	Input	AHB write/read signal
hrdata<31:0>	Output	AHB read data
hwdata<31:0>	Input	AHB Write data
htrans<1:0>	Input	AHB Transfer Type
hsize<1:0>	Input	AHB Data access width
hburst<2:0>	Input	AHB Burst type
hresp<1:0>	Output	AHB slave response
Hreadyin	Input	AHB ready signal (input)
Hreadyout	Output	AHB ready signal (output)
System Interface		
Hclk	Input	AHB/ System Clock
Hresetn	Input	AHB Reset
Intr	Output	Interrupt out
PHY chip Interface		
Txrequesths	Output	HS DATA transmit request
Txrequesths_c	Output	HS Clock request
Txclkhs	Input	Transmit byte clock
txdatahs<7:0>	Output	HS transmit data
Txreadyhs	Input	HS Data transmit ready signal

Table 4.1 Traffic Generator Port List

4.4.1 AHB Interface Module

AHB Interface will provide interconnection with ARM Versatile baseboard. The Control and Status Registers and dual port memories are accessed through software running on ARM Versatile baseboard. It has input and output ports necessary to connect with ARM board and to read/write operation on Control and Status Registers and memories.

Figure 4.5 describes the ports of AHB Interface Module (the primary inputs, outputs and inouts).

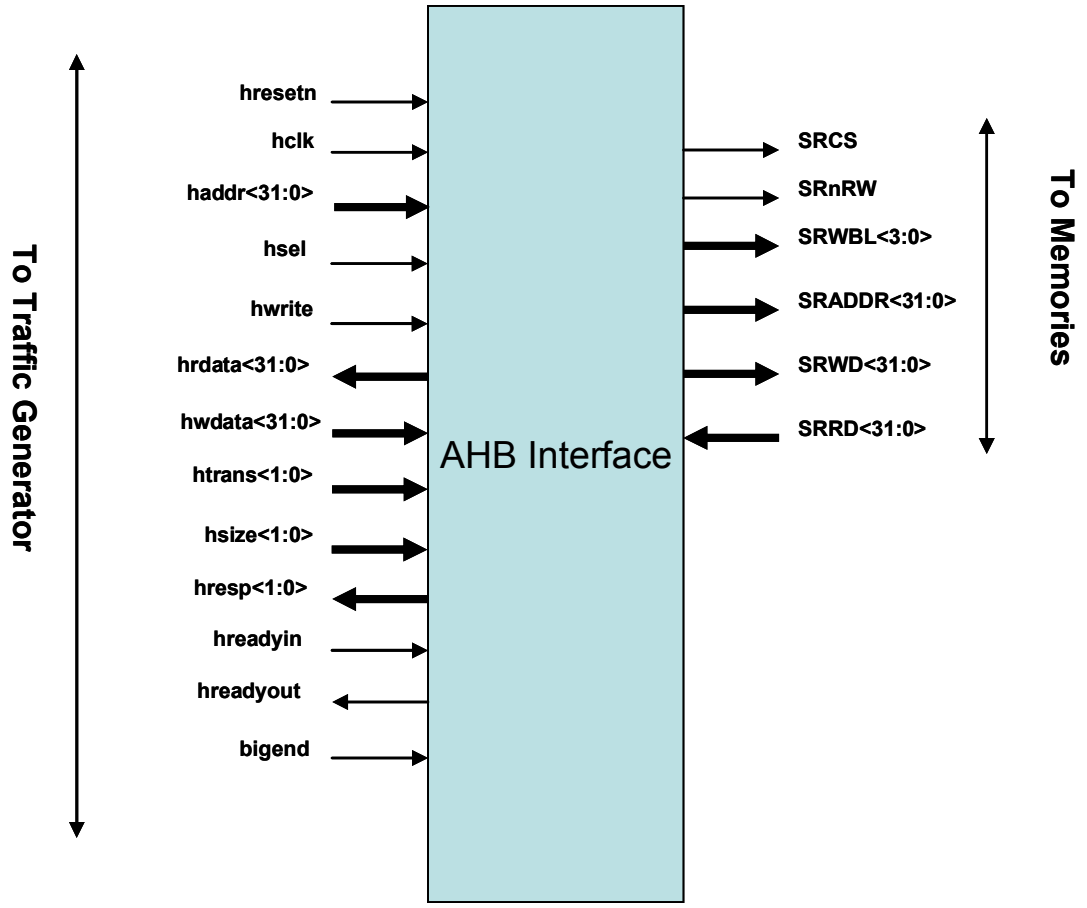


Figure 4.5 Pin Diagram of AHB Interface Module

Port list of AHB Interface is given in the Table 4.2.

Port Name (with Width)	Direction	Description
AHB Interface		
haddr<31:0>	Input	AHB Address
Hsel	Input	AHB Peripheral Select
Hwrite	Input	AHB write/read signal
hrdata<31:0>	Output	AHB read data
hwdata<31:0>	Input	AHB Write data
htrans<1:0>	Input	AHB Transfer Type
hsize<1:0>	Input	AHB Data access width
hresp<1:0>	Output	AHB slave response
Hreadyin	Input	AHB ready signal (input)
Hreadyout	Output	AHB ready signal (output)
BIGEND	Input	AHB bigendian
System		
Hclk	Input	AHB/ System Clock
Hresetn	Input	AHB Reset
Memory interface		
SRCS	Output	RAM chip select
SRnRW	Output	RAM Read/Write
SRADDR<31:0>	Output	RAM Address
SRWD<31:0>	Output	RAM Data Write to Memory
SRWBL<3:0>	Output	RAM Write Byte Enable
SRRD<31:0>	Input	RAM Data Read From Memory

Table 4.2 AHB Interface Port List

4.4.2 Decoder Logic Module

Decoder module will decode the address provided by AHB Interface module and will generate the chip selection signals for control and status registers provided at CSR module and also the chip selection signals for dual port RAMs of Datastreaming Logic module.

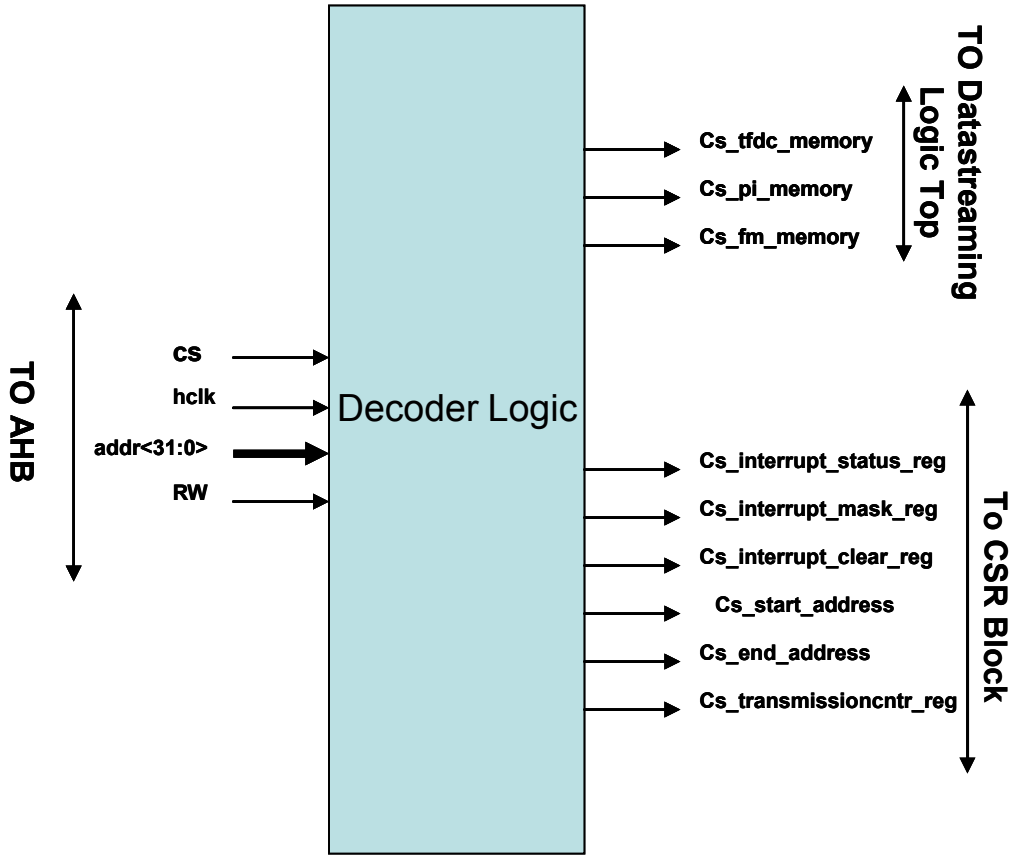


Figure 4.6 Pin Diagram of Decoder Logic Module

Port list of Decoder Logic is given in Table 4.3.

Port Name (with Width)	Direction	Description
Hclk	Input	System Clock from ARM
Addr<31:0>	Input	Address from ARM AHB
Cs	Input	Chip select for decoder block
RW	Input	Read/Write
Cs_tfdc_memory	Output	Chip select for traffic descriptor memory
Cs_pi_memory	Output	Chip select for packet index memory
Cs_fm_memory	Output	Chip select for frame memory
Cs_interrupt_status_reg	Output	Chip select for interrupt status register
Cs_interrupt_mask_reg	Output	Chip select for interrupt mask register
Cs_interrupt_clear_reg	Output	Chip select for interrupt clear register
Cs_start_address	Output	Chip select for start address of traffic descriptor memory
Cs_end_address	Output	Chip select for end address of traffic descriptor memory
Cs_transmissoncntr_reg	Output	Chip select for transmission control register

Table 4.3 Decoder Logic Port List

4.4.3 Control & Status Register Block:

CSR module will provide the control and status registers for Traffic Generator module. These registers will be used to generate the control signals. Status registers have information about errors, end of transmission and other information.

These registers are written and read by using AHB interface module by user, so the data transmission is user programmable and controllable.

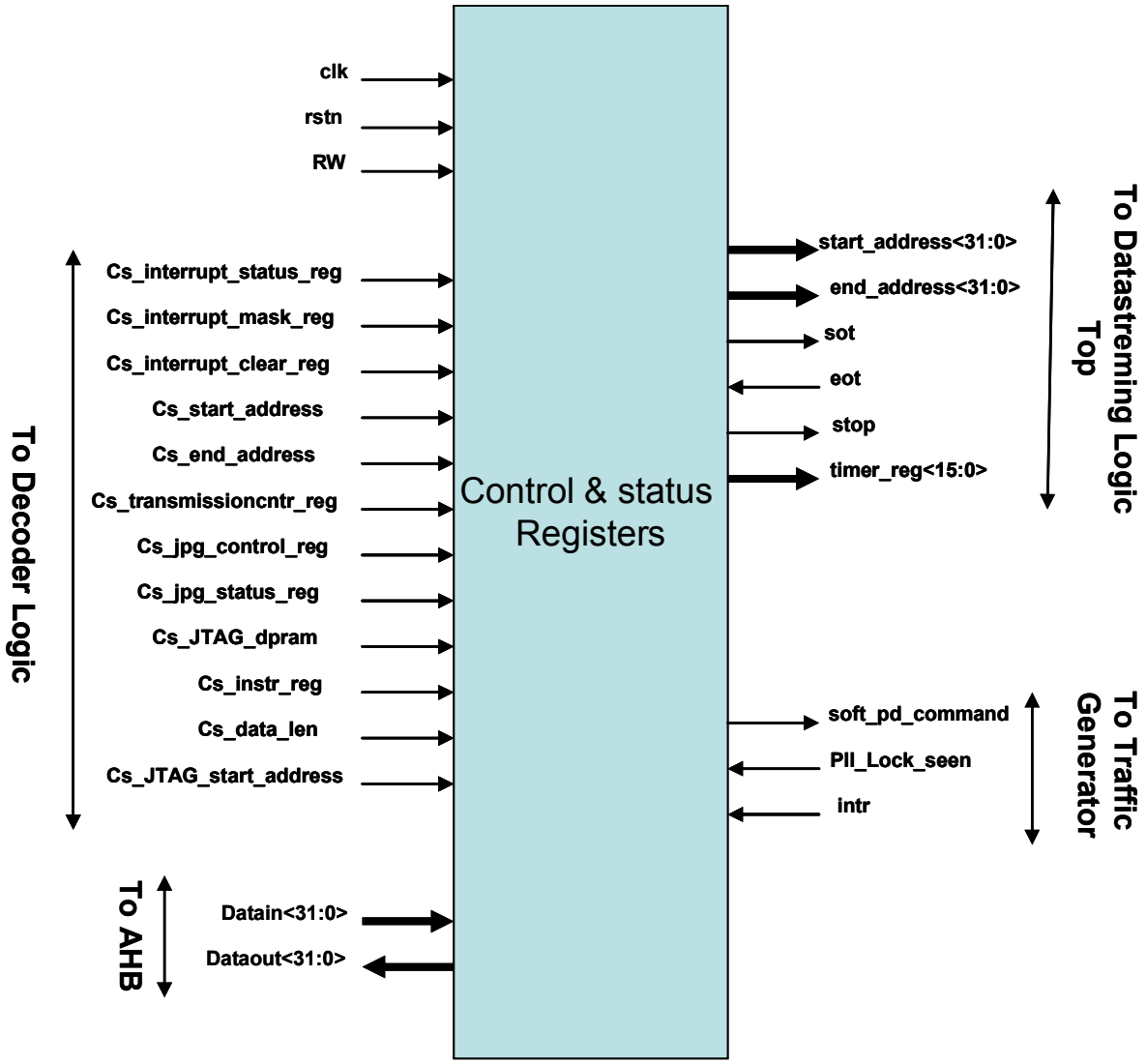


Figure 4.7 Pin Diagram of Control & Status Registers Module

Port list of Control & Status Registers Block is given in Table 4.4.

Port Name (with Width)	Direction	Description
Decoder Logic Block		
Cs_interrupt_status_reg	Input	Chip select for interrupt status register
Cs_interrupt_mask_reg	Input	Chip select for interrupt mask register

Cs_interrupt_clear_reg	Input	Chip select for interrupt clear register
Cs_start_address	Input	Chip select for start address of traffic descriptor memory
Cs_end_address	Input	Chip select for end address of traffic descriptor memory
Cs_transmissoncntr_reg	Input	Chip select for transmission control register
System		
Clk	Input	AHB/ System Clock
Rstn	Input	AHB/system Reset
RW	Input	Read/ write
DATA Streaming Logic Top Block		
Start_address<31:0>	Output	Start address for traffic descriptor memory
End_address <31:0>	Output	End address for traffic descriptor memory
Sot	Output	Start of transmission signal for ppi tx
Eot	Input	End of transmission signal from flow controller block
Stop	Output	Stop signal to stop transmission manually
Timer_reg<15:0>	Output	Hold value to provide sufficient delay between packets
AHB Interface		
Datain <31:0>	Input	AHB slave input data
Dataout <31:0>	Output	AHB slave output data
Traffic Generator		
Soft_pd_command	Output	Soft power down command
Pll_lock_seen	Input	Pll lock
Intr	Input	Interrupt from events

Table 4.4 Control & Status Registers block Port List

❖ **Control and Status Registers:**

The Control and Status Registers detail is given below:

- **Transmission control register**

It is 32 bit read-write register. Some control signals are in this register

Transmissioncntr_reg	**0x00005000**		**RW /32 bit**
Bit	Bit Field	Reset value	Description
Sot	0	0	Start of data transmission, signal for data streaming logic block
Stop	1	0	Manual stop of data transmission, signal for data streaming logic block
Reserved for future	2-15	0	Reserved for future
Timer_reg	31-16	0	Value for delay between packets

- **Traffic descriptor memory end address register**

It is 32 bit read-write register. End address of Traffic descriptor memory is in this register.

End address register	**0x00005004**		**RW /32 bit**
Bit	Bit Field	Reset value	Description
end address	31-0	0	end address of traffic descriptor memory

- **Traffic descriptor memory start address register**

It is 32 bit read-write register. Start address of Traffic descriptor memory is in this register.

Start address register	**0x00005008**		**RW /32 bit**
Bit	Bit Field	Reset value	Description
Start address	31-0	0	Start address of traffic descriptor memory

- **Interrupt Status Register**

It is read only register. It is used for show the interrupt status of various blocks.

IntStatusReg	**0x0000500c**		**Read only /32 bit**
Bit	Bit Field	Reset value	Description
PLL lock seen interrupt	0	0	Interrupt for pll lock seen
Pattern done interrupt	1	0	Interrupt for pattern done
start interrupt	2	0	Interrupt for start
Error_0 interrupt	3	0	for some error in future
Error_1interrupt	4	0	for some error in future
Error_2nterrupt	5	0	for some error in future
Error_3nterrupt	6	0	for some error in future
Error_4nterrupt	7	0	for some error in future
Error_5nterrupt	8	0	for some error in future
End of tx interrupt	9	0	Interrupt from end of tx
Other Interrupt	10-31	0	TBD (Reserved for future use)

- **Interrupt Mask Register**

It is 32 bit read-write register. For each interrupt a corresponding mask bit is assigned in this register.

IntMaskReg	**0x00005010**		**RW /32 bit**
Bit	Bit Field	Reset value	Description
PLL lock seen interrupt	0	0	Interrupt mask(enable/disable) bit for pll lock seen 1: to mask the interrupt, 0: to unmask the interrupt
Pattern done interrupt	1	0	Interrupt mask(enable/disable) bit for pattern done

			1: to mask the interrupt, 0: to unmask the interrupt
start interrupt	2	0	Interrupt mask(enable/disable) bit for start 1: to mask the interrupt, 0: to unmask the interrupt
Error_0 interrupt	3	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
Error_1interrupt	4	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
Error_2nterrupt	5	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
Error_3nterrupt	6	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
Error_4nterrupt	7	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
Error_5nterrupt	8	0	Interrupt mask bit for some error in future 1: to mask the interrupt, 0: to unmask the interrupt
End of tx interrupt	9	0	Interrupt mask bit for some error in

			future 1: to mask the interrupt, 0: to unmask the interrupt
Other Interrupt	10-31	0	TBD (Reserved for future use)

- **Interrupt Clear Register**

It is 32 bit read-write register. For each interrupt, there will be a corresponding clear bit assigned in this register.

IntClearReg	**0x00005014**		**RW /32 bit**
Bit	Bit Field	Reset value	Description
PLL lock seen interrupt clear	0	0	Interrupt clear bit for pll lock seen
Pattern done interrupt clear	1	0	Interrupt clear bit for pattern done
start interrupt clear	2	0	Interrupt clear bit for start
Error_0 interrupt clear	3	0	clear bit for some error in future
Error_1interrupt clear	4	0	clear bit for some error in future
Error_2nterrupt clear	5	0	clear bit for some error in future
Error_3nterrupt clear	6	0	clear bit for some error in future
Error_4nterrupt clear	7	0	clear bit for some error in future
Error_5nterrupt clear	8	0	clear bit for some error in future
End of tx interrupt clear	9	0	Interrupt from end of transmission
Other Interrupt clear	10-31	0	TBD (Reserved for future use)

4.4.4 Data streaming Logic Top

It is the top module for data streaming logic. It consist two sub modules Datastreaming_logic and ppitx. It has ports necessary for AHB read/write operation on memories, generation of request for PHY chip, data transmission, start and end of transmission, delay between packets and some other ports. It is used to form the packets and frames and transmit these frames on PHY chip Interface. Datastreaming_logic module has four modules; the flow controller and three dual port memories. Datastreaming_logic module is used for frame formation and Ppitx is used for data transmission. It transmits 8 bit data to PHY chip.

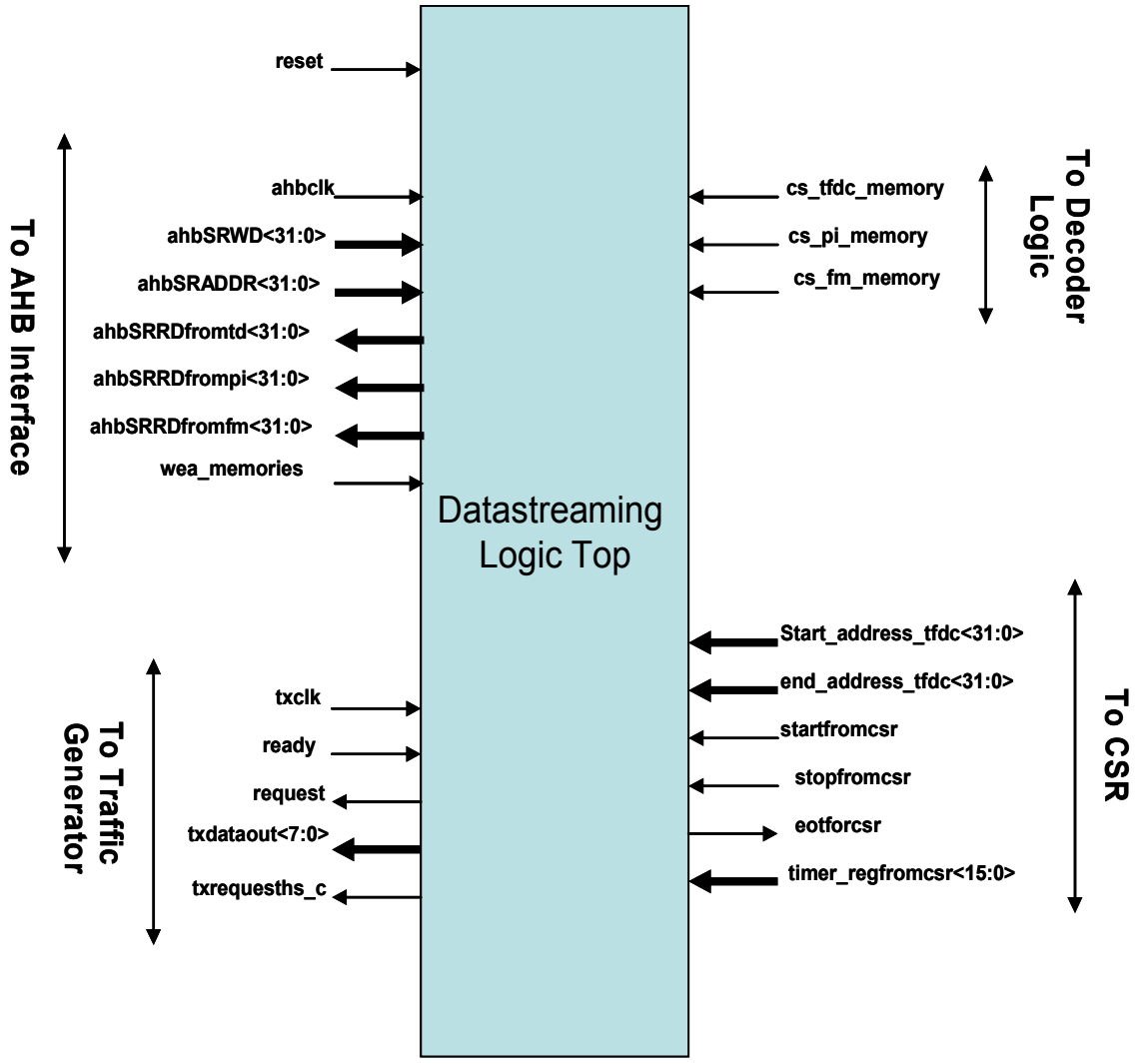


Figure 4.8 Pin diagram of Data Streaming Logic Top

Port list of Data streaming Logic Top is given in Table 4.5.

Port Name (with Width)	Direction	Description
Reset	Input	Reset for block
Txclk	Input	High speed txclk from PHY chip
Ahbclk	Input	System clock from AHB
AhbSRWD<31:0>	Input	Data input from AHB
ahbSRADDR<31:0>	Input	Address for memories
ahbSRRDfromtd<31:0>	Output	Data to AHB Output from traffic descriptor memory
ahbSRRDfrompi<31:0>	Output	Data to AHB Output from packet index memory
ahbSRRDfromfm<31:0>	Output	Data to AHB Output from frame memory
Ready	Input	High speed ready signal from PHY chip
Request	Output	High speed data transmit request to PHY chip
Txdataout<7:0>	Output	Data output to PHY chip
Cs_tfdc_memory	Input	Chip select for traffic descriptor memory
Cs_pi_memory	Input	Chip select for packet index memory
Cs_fm_memory	Input	Chip select for frame memory
Wea_memories	Input	Write enable for port A of memories
Start_address_tfdc<31:0>	Input	Start address for traffic descriptor memory
End_address_tfdc<31:0>	Input	End address for traffic descriptor memory
Startfromcsr	Input	Start signal for transmission begin comes from CSR
Stopfromcsr	Input	Stop signal for manually stop of data transmission
Eotforcsr	Output	Signal shows end of transmission of data
Timer_regfromcsr<15:0>	Input	Holds data for delay between packets
Txrequesths_c	Output	High speed request signal for clock lane

Table 4.5 Datastreaming Logic Top Port List

4.4.4.1 Datastreaming_logic

Its function is formation of data stream according to control signals and traffic description given in the memory. Users gives control signal by writing control registers of CSR Block using the AHB interface. This module takes control signals from control registers of CSR Block i.e. start, stop, start address for traffic descriptor memory, end address for traffic descriptor memory, eot etc. It has four sub modules flow controller, traffic descriptor memory, packet index memory and frame memory. The contents of three memories have information of packet formation and delay between packets. This module's output is 32 bit data, which goes to ppitx module.

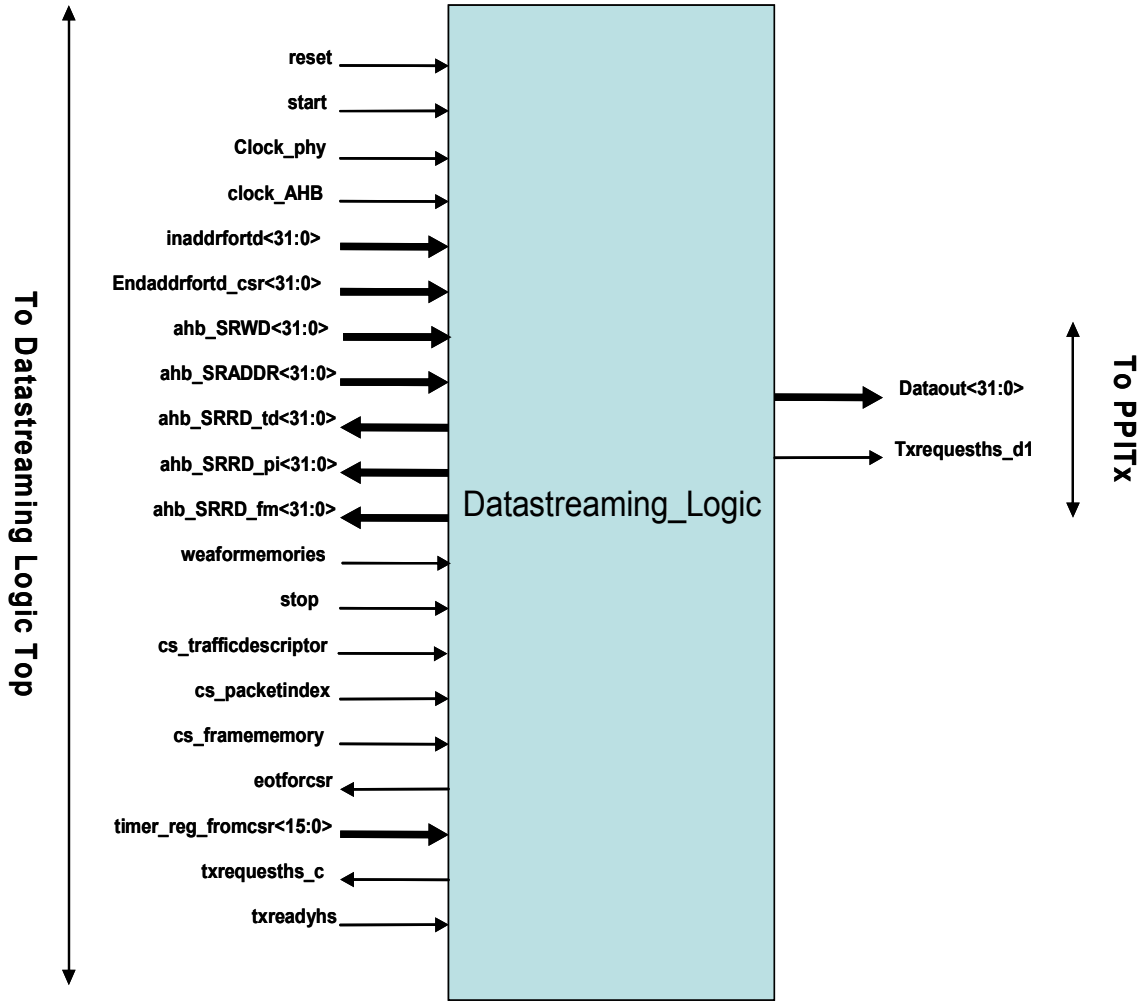


Figure 4.9 Pin Diagram of Datastreaming_logic Module

Port list of Datastreaming_logic is given in Table 4.6.

Port Name (with Width)	Direction	Description
Reset	Input	Reset for block
Clock_phy	Input	High speed txclkhs from PHY chip
Clock_ahb	Input	System clock from AHB
Start	Input	Start signal for transmission begin
Inaddrfortd<31:0>	Input	Start address for traffic descriptor memory
Endaddrfortd_csr<31:0>	Input	End address for traffic descriptor memory
dataout<31:0>	Output	Data output to ppi tx block
Ahb_SRWD<31:0>	Input	Data input from AHB
Ahb_SRADDR<31:0>	Input	Address for memories
Ahb_SRRD_td<31:0>	Output	Data to AHB Output from traffic descriptor memory
Ahb_SRRD_pi<31:0>	Output	Data to AHB Output from packet index memory
Ahb_SRRD_fm<31:0>	Output	Data to AHB Output from frame memory
Weaformemories	Input	Write enable for port A of memories
Stop	Input	Stop signal for manually stop of data transmission
Cs_trafficdescriptor	Input	Chip select for traffic descriptor memory
Cs_packetindex	Input	Chip select for packet index memory
Cs_framememory	Input	Chip select for frame memory
Eotforcsr	Output	Signal shows end of transmission of data
Timer_reg_fromcsr<15:0>	Input	Holds data for delay between packets
txRequesths_d1	Output	High speed data transmit request to PHY chip data lane
Txrequesths_c	Output	High speed request signal for clock lane
txReadyhs	Input	High speed ready signal from PHY chip

Table 4.6 Datastreaming_logic Port List

4.4.4.1.1 Flow Controller

Flow controller is main control module for controlling of data streaming and data transmission. It reads frame configuration detail from traffic descriptor memory. It has ports which have control information taken from control and status registers of CSR Block.

It has a state machine, which has 12 states as follows; idle, waitforstart, addrfortd, readfromtd, readfrompi, bufferfromfm, addrforfm, state_request, readfromfm, delayforppitx, checkstate and delaystate. Start and end of data transmission are decided by start and end address of traffic descriptor memory which is takes from CSR registers.

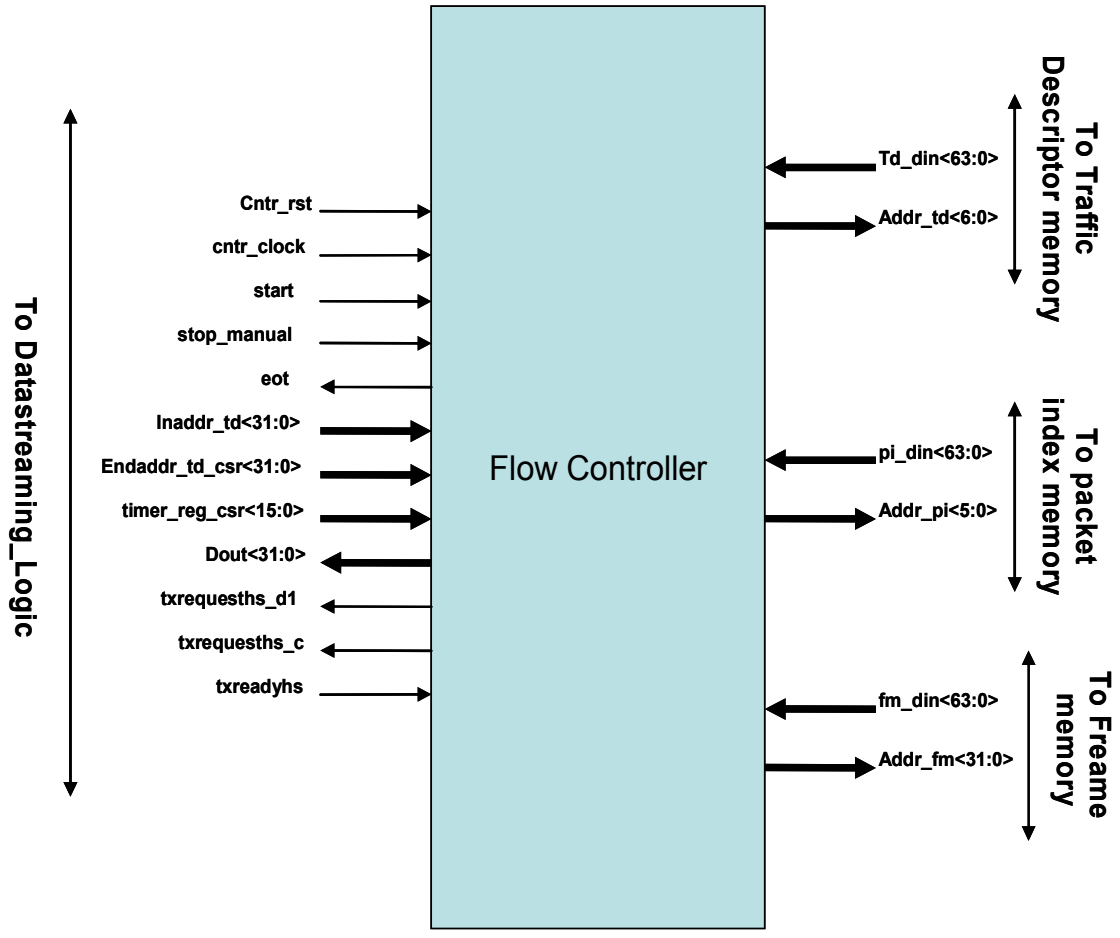


Figure 4.10 Pin Diagram of Flow Controller Module

Port list of flow controller is given in Table 4.7.

Port Name (with Width)	Direction	Description
Cntr_rst	Input	Reset (active low)
Cntr_clock	Input	PHY CHIP clock
Start	Input	Signal for start of transmission of data
Stop_manual	Input	Signal for manually stop transmission of data
Eot	Output	Signal shows end of data transmission
Inaddr_td<31:0>	Input	Start address for traffic descriptor memory
Endaddr_td_csr<31:0>	Input	End address for traffic descriptor memory
Timer_reg_csr<15:0>	Input	Holds data for give delay between packets
Td_din <63:0>	Input	Input data from traffic descriptor memory
Pi_din<63:0>	Input	Input data from packet index memory
Fm_din<63:0>	Input	Input data from frame memory
Dout<31:0>	Output	Output data for ppitx
Addr_pi<5:0>	Output	Output address for packet index memory
Addr_td<6:0>	Output	Output address for traffic descriptor memory
Addr_fm<31:0>	Output	Output address for frame memory
Txrequesths_d1	Output	High speed transmit request for data lane
Txrequesths_c	Output	High speed transmit request for clock lane
Txreadyhs	Input	High speed ready signal

Table 4.7 flow controller Port List

4.4.4.1.2 Traffic Descriptor Memory

This memory is a dual port RAM. It consist two ports A and B. port A is read/write port with 32 bit width. Port B is read only port with 64 bit width. There are two clocks AHB clock connected with port A and PHY CHIP clock connected with port B. This memory has

information of frame configuration, packet index memory address, and delay between packets, number of retransmission of same packet and other information.

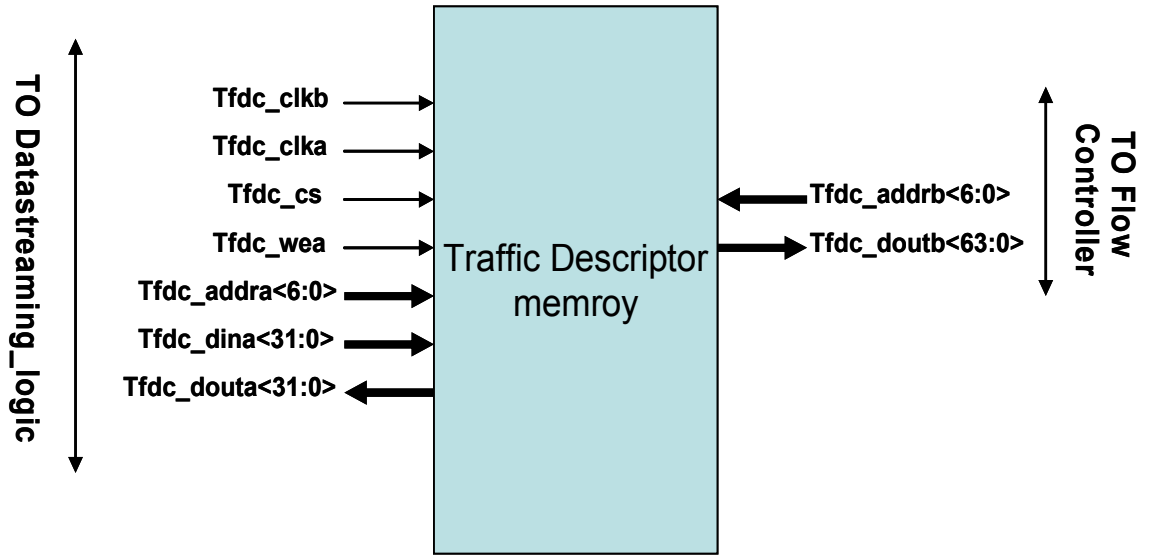


Figure 4.11 Pin Diagram of Traffic Descriptor Memory

Port list of Traffic Descriptor Memory is given in Table 4.8.

Port Name (with Width)	Direction	Description
Tfdc_clka	Input	Clock for port A
Tfdc_clkb	Input	Clock for port B
Tfdc_cs	Input	Chip select from decoder block
Tfdc_addra<6:0>	Input	Address for port A
Tfdc_addrb<6:0>	Input	Address for port B
Tfdc_dina<31:0>	Input	Data input from Port A
Tfdc_douta<31:0>	Output	Data output from Port A
Tfdc_doutb<63:0>	Output	Data output from Port B
Tfdc_wea	Input	Write enable for port A

Table 4.8 Traffic descriptor memory Port List

4.4.4.1.3 Packet Index Memory:

It is a dual port RAM, which consist two ports A and B. port A is 32 bit width read/write port and port B is 64 bit width read only port. There are two clocks, first AHB clock connected with port A and second PHY chip clock connected with port B. This memory stores start and end addresses of frame memory.

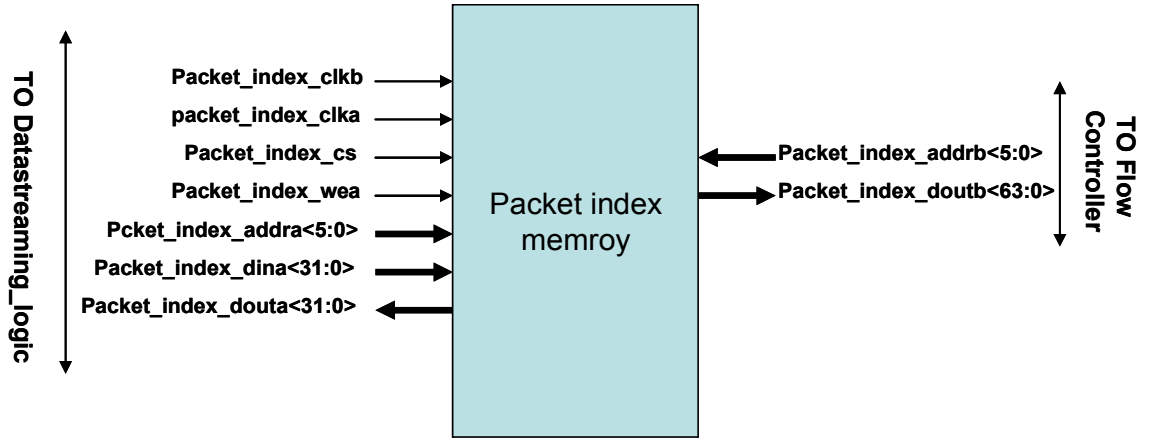


Figure 4.12 Pin Diagram of Packet index Memory

Port list of Packet Index Memory is given in Table 4.9.

Port Name (with Width)	Direction	Description
Packet_index_clka	Input	Clock for port A
Packet_index_clkb	Input	Clock for port B
Packet_index_cs	Input	Chip select from decoder block
Packet_index_addra<5:0>	Input	Address for port A
Packet_index_addrb<5:0>	Input	Address for port B
Packet_index_dina<31:0>	Input	Data input from Port A
Packet_index_douta<31:0>	Output	Data output from Port A
Packet_index_doutb<63:0>	Output	Data output from Port B
Packet_index_wea	Input	Write enable for port A

Table 4.9 Packet Index memory Port List

4.4.4.1.4 Frame Memory

It is a dual port RAM, which consist two ports A and B. Port A is 32 bit width read/write port. Port B is 32 bit width read only port with. There are two clocks, first AHB clock connected with port A and second PHY chip clock connected with port B. This memory stores the data of packets.

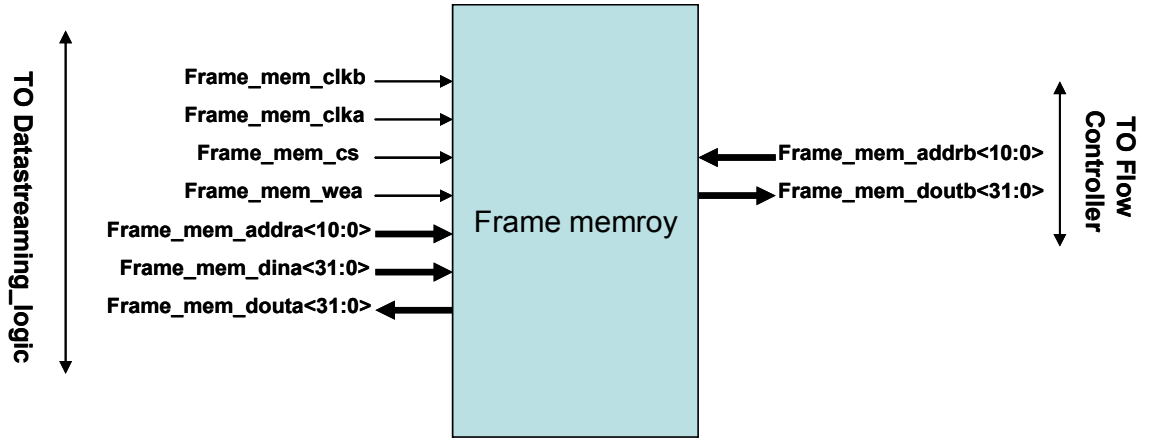


Figure 4.13 Pin Diagram of Frame Memory

Port list of Frame Memory is given in Table.

Port Name (with Width)	Direction	Description
Frame_mem_clka	Input	Clock for port A
Frame_mem_clkb	Input	Clock for port B
Frame_mem_cs	Input	Chip select from decoder block
Frame_mem_addra<10:0>	Input	Address for port A
Frame_mem_addrb<10:0>	Input	Address for port B
Frame_mem_dina<31:0>	Input	Data input from Port A
Frame_mem_douta<31:0>	Output	Data output from Port A
Frame_mem_doutb<31:0>	Output	Data output from Port B
Frame_mem_wea	Input	Write enable for port A

Table 4.10 Frame memory Port List

4.4.4.2 PPI Transmitter Block

It has 5 states idle, waitforstart, waitforrequest, waitforready, and hstransmission. It has start and stop signals and high speed data transmission signals i.e. txrequesths, txreadyhs, txclkhs, txdatahs and datain. When txreadyhs signal is high, It takes 32 bit data from flow controller by datain port and transmits 8 bit high speed data for PHY chip from txdatahs port.

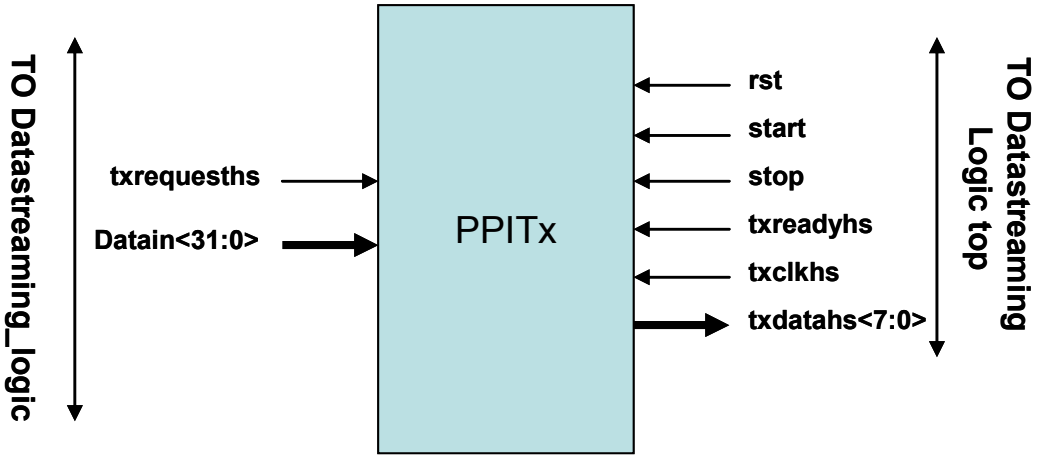


Figure 4.14 Pin Diagram of PPI Transmitter Module

Port list of ppitx is given in Table 4.11.

Port Name (with Width)	Direction	Description
Rst	Input	Reset (active low)
Start	Input	Signal for start of transmission of data
Stop	Input	Signal for manually stop transmission of data
Txreadyhs	Input	High speed ready signal from PHY CHIP
Txclkhs	Input	High speed clock from PHY CHIP
Datain<31:0>	Input	Data input from Datastreaming_logic block
Txrequesths	Input	High speed data transmission request
Txdatahs<7:0>	Output	High speed data transmission

Table 4.11 Ppitx Port List

CHAPTER 5
RESULTS

A use case of single frame transmission and packet data flow in a frame is shown using various tables provided below. It is used in this project for testing of traffic generator and real time implementation of HSSD protocol.

Traffic Descriptor Memory

Each entry is 64 bit wide and continuously placed in memory.

Packet Index (16 bits)	**No. of Iterations (12 bits)**	**Inter Packet Delay counter (12 bits)**	**Reserved bits (24 bits)**
0	1	26	X
1	479	1848	X
2	1	26	X
3	1	0	X

Packet Index Memory

The entries indicate where the different packets are located in the Frame Memory.

Packet Index	0	1	2	3
Start address	0	4	652	1300
End address	3	651	1299	1303

Frame Memory

The memory containing the actual packets to be transmitted.

0	….	3	4	….	651	652	….	1299	1300	….	1303
Frame Start			Long Packet 1			Long Packet 2			Frame end		

Data Flow

The representation below shows the sequence of packets transmitted according to the above programming.

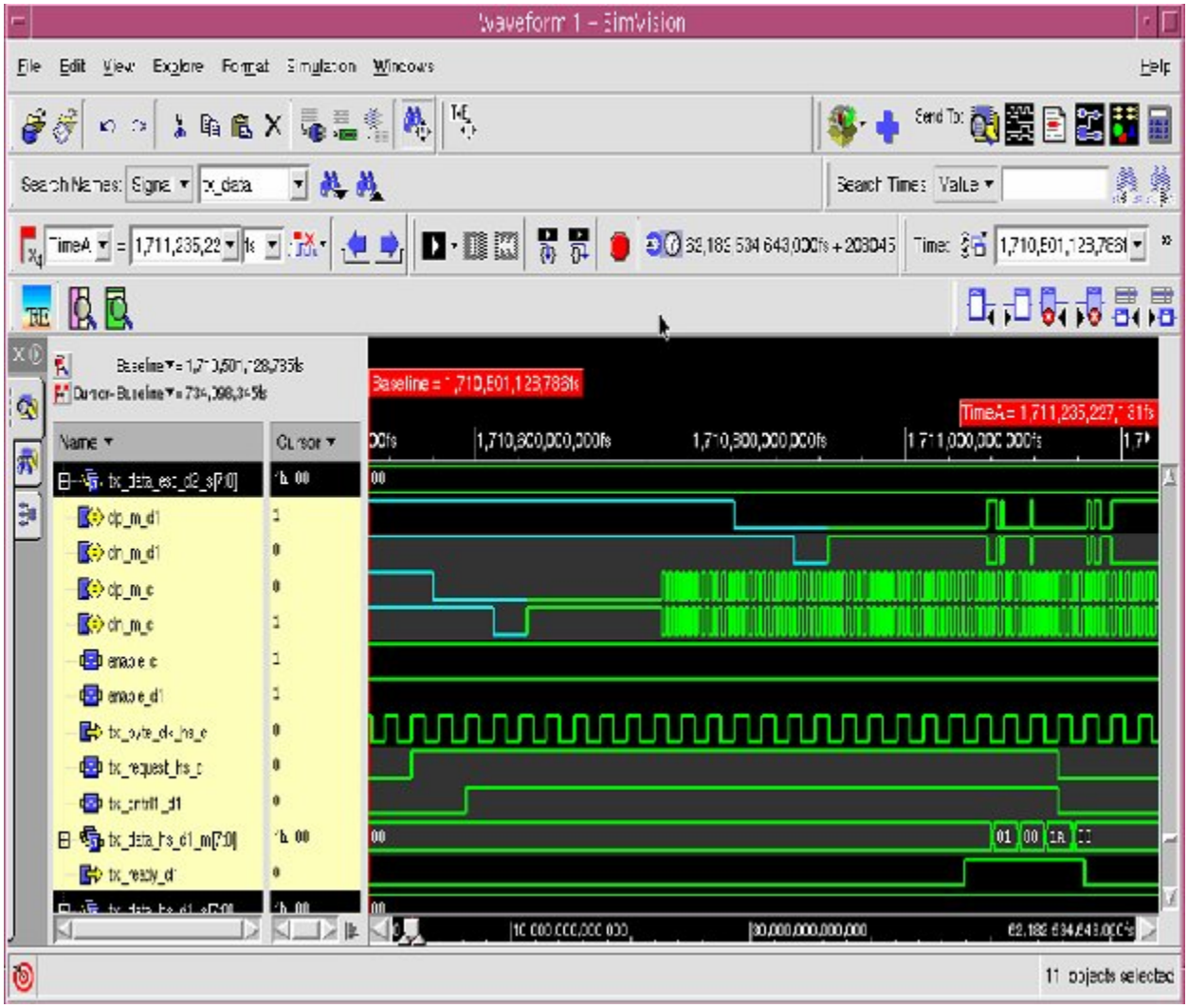
Frame Start	LPS (26 Unit)	Line 1	LPS (1848 Units)	Line2	LPS (1848 Units)	- - -	Line 480	LPS (26 Units)	Frame End

The waveforms show the successfully transmission of a single frame by traffic generator using HSSD as Protocol. Successful transmission of 2, 10, 40 and other multiple numbers of frames has also been verified.

The resulting waveform for above case of single frame transmission is shown below:

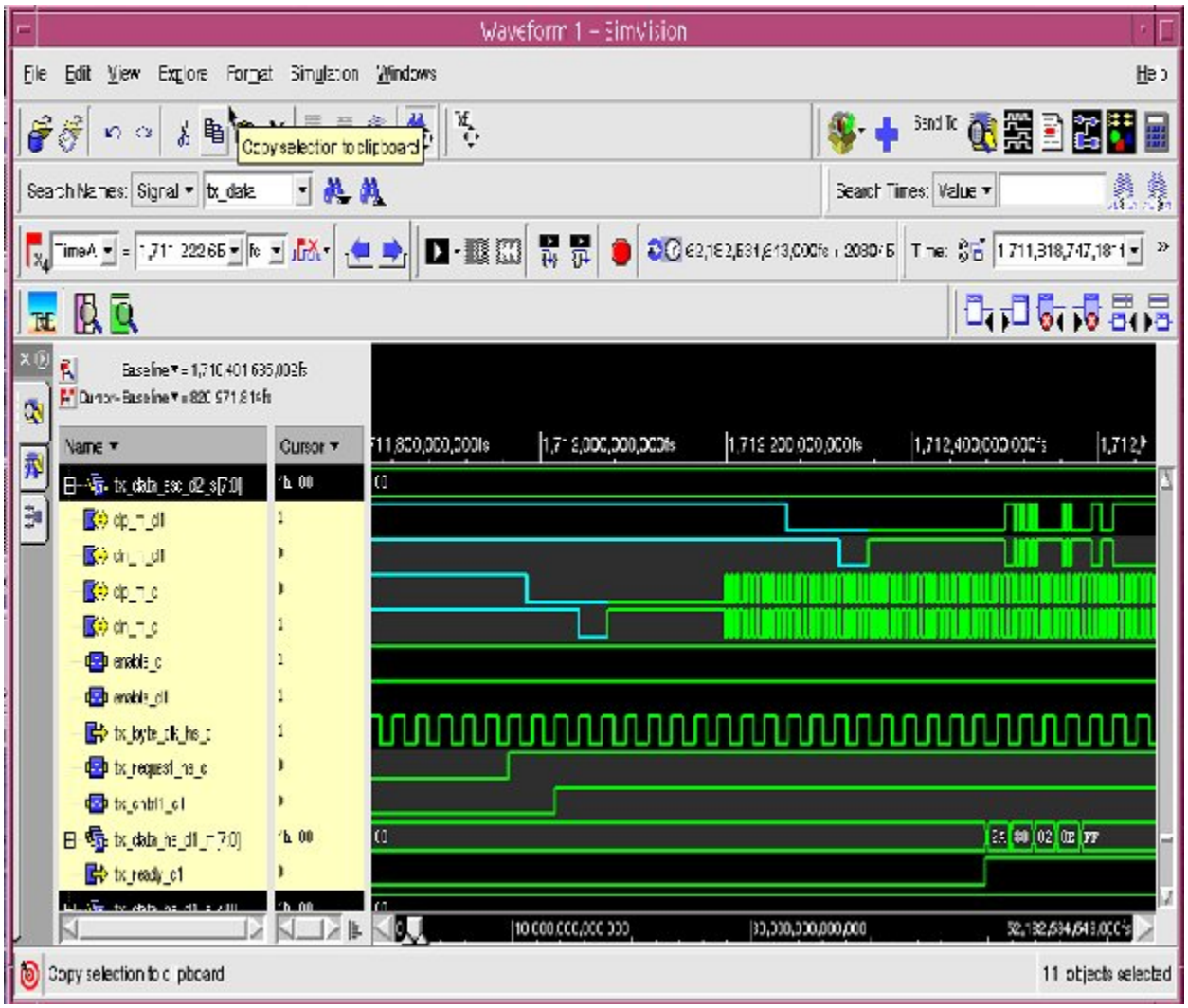
Frame Start Packet waveform:

In HSSD protocol first a frame start packet is transmitted, which has a count value to show the no. of frame being transmitted. Waveform below shows the transmission of Frame start packet. As shown in waveform first a request is send and then logic waits for the ready signal. When ready is received, the traffic generator transmits the frame start packet. After packet transmission, the request becomes low.



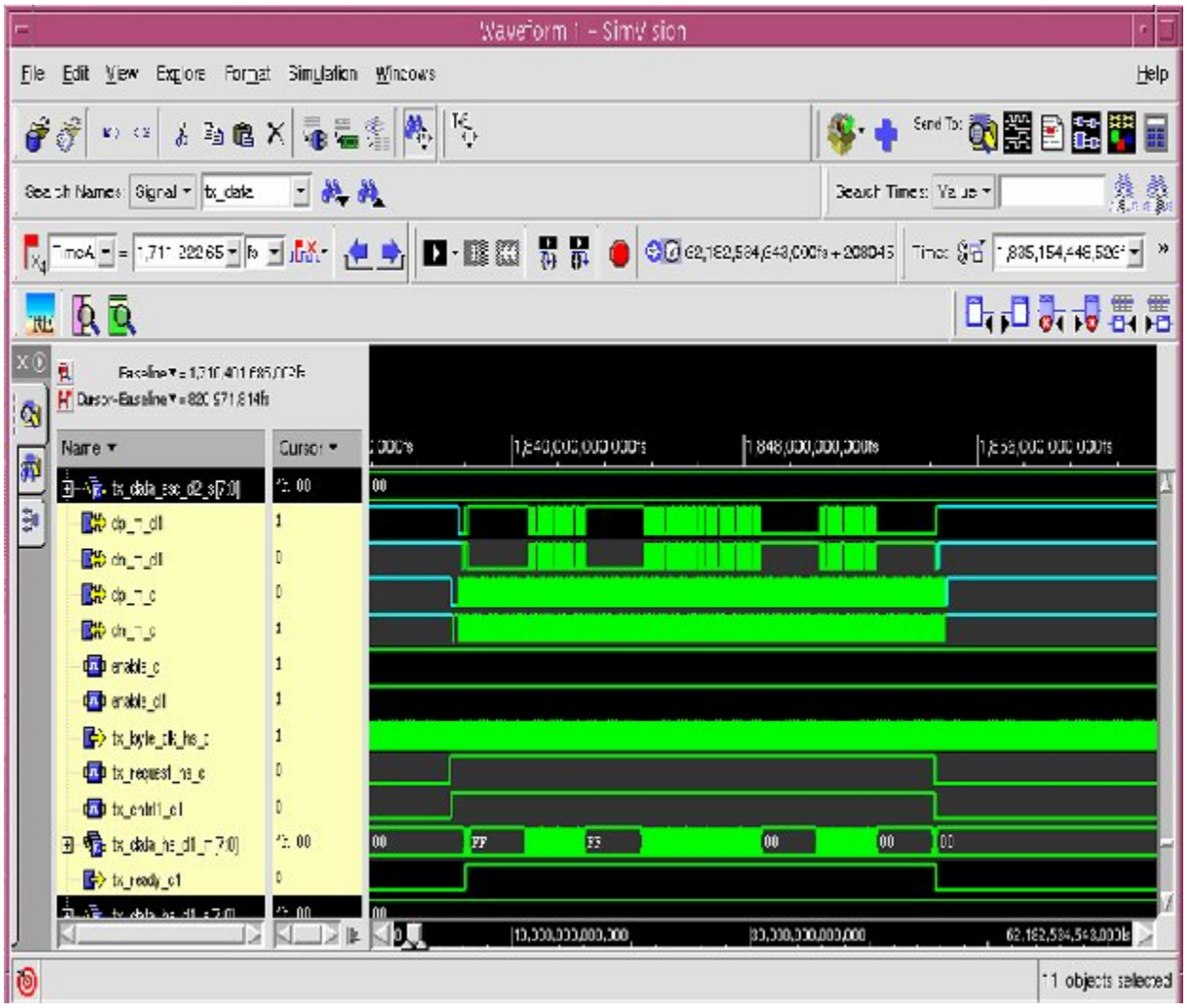
Waveform for Packet header of long Packet:

Every long packet has packet header and packet footer. This waveform shows packet header of a long packet. Packet header provides information about data type and number of bytes in the packet.



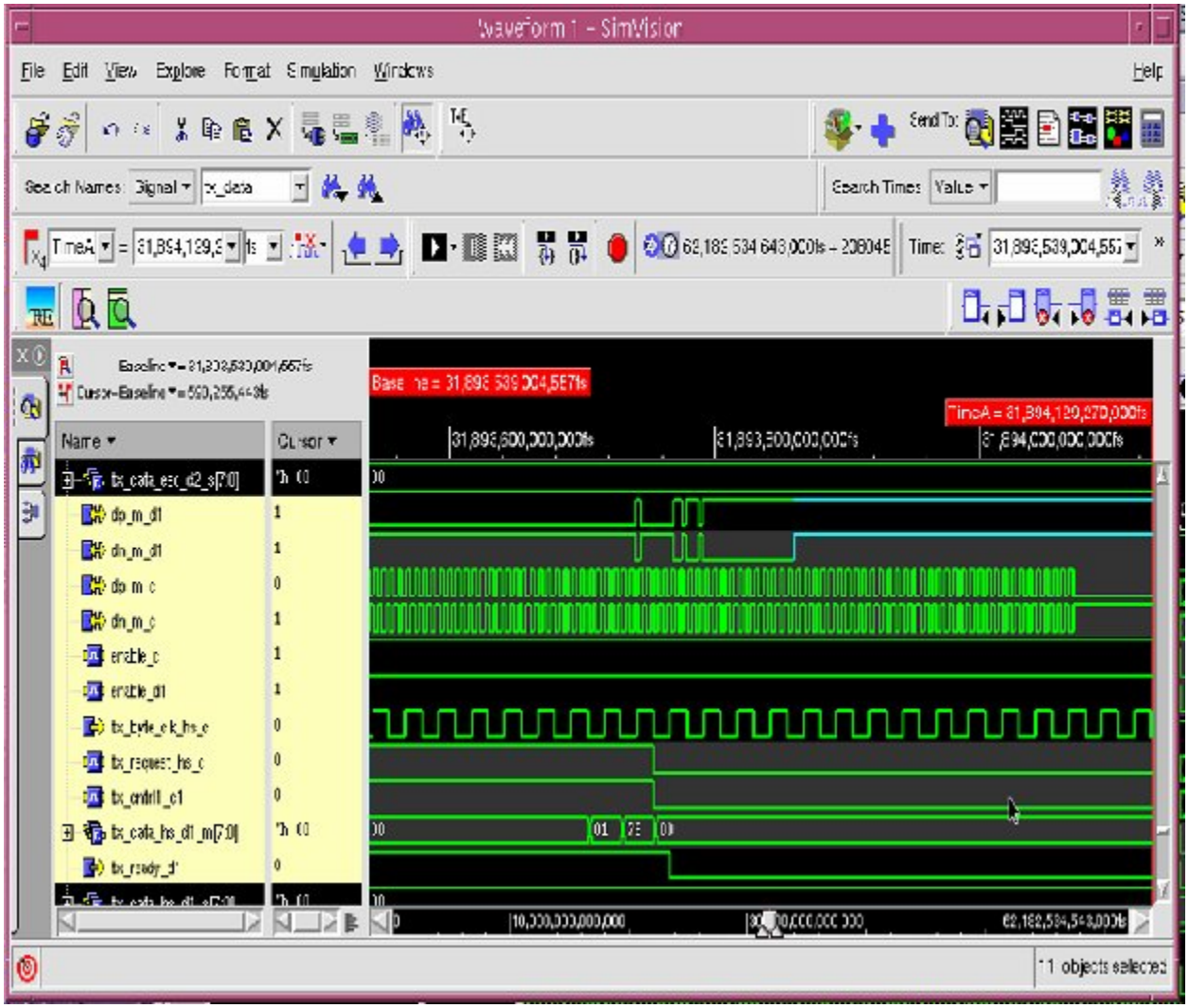
Waveform of a long packet in the frame:

This waveform shows the transmission of a long packet in the frame



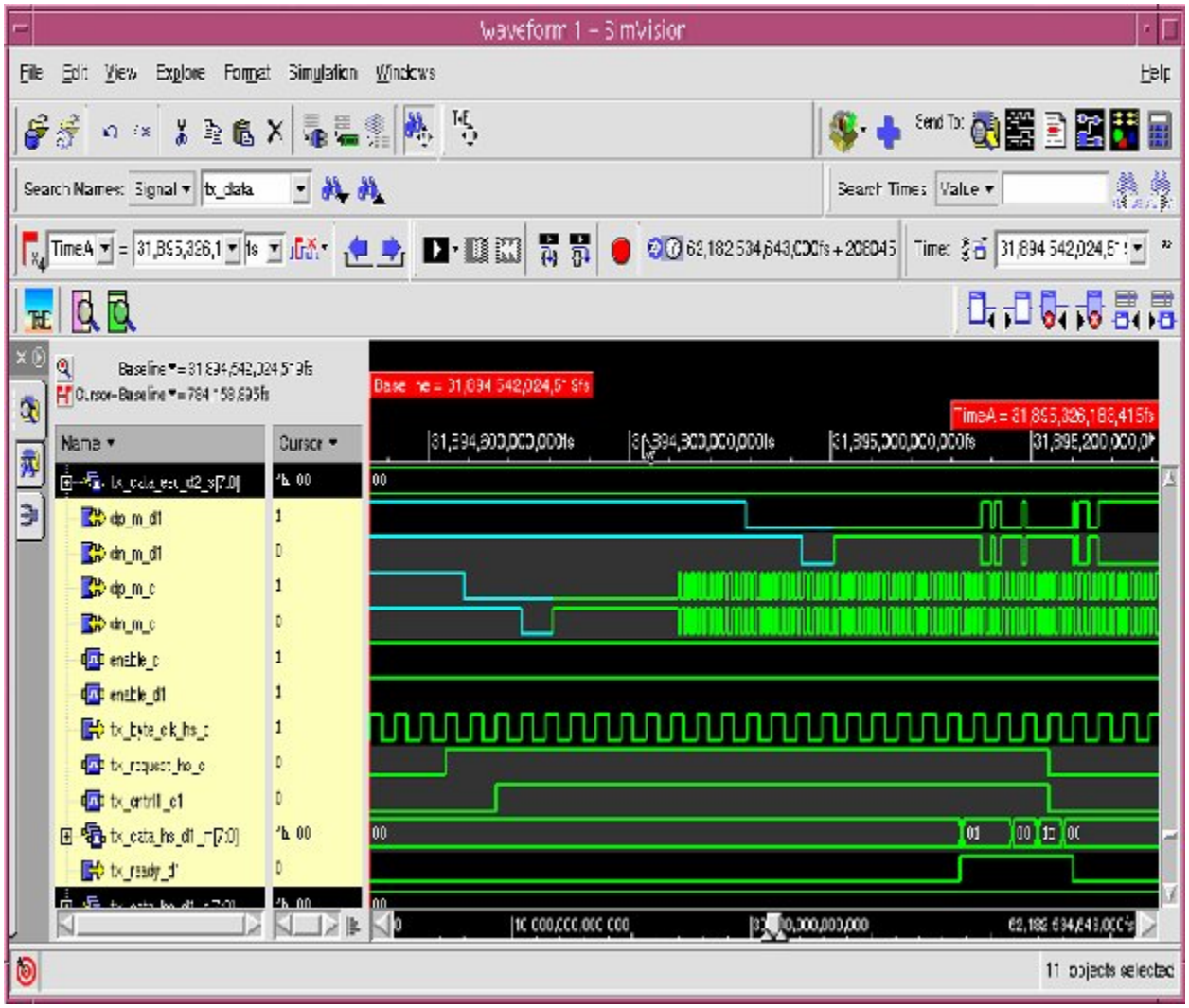
Waveform of Packet footer of a long packet:

This waveform shows the packet footer of a long packet. Packet footer is a CRC code to check the error in the data of packet.



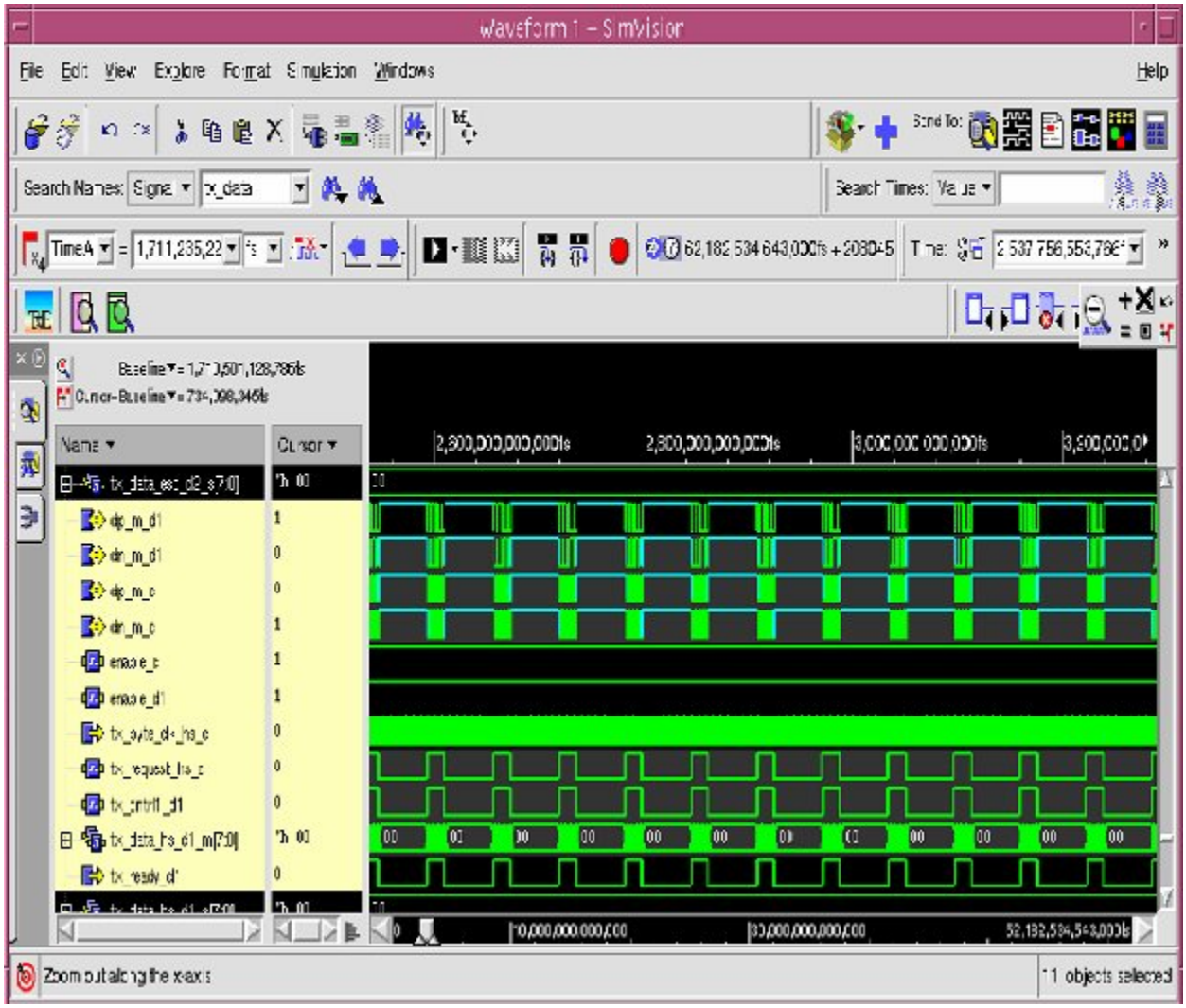
Waveform of Frame End packet:

Waveform below shows the transmission of Frame End packet for the current Frame. The Frame End packet, similar to the Frame Start packet, also has a frame number field. Frame number of Frame Start and Frame End packets should be same for correct frame synchronization.



Waveform showing transmission of multiple long packets in the frame:

The waveform below shows multiple long packets being transmitted in the frame one after the other.



CHAPTER 6
CONCLUSION

The HSSD Protocol Traffic Generator is made in this thesis. The HSSD Protocol Traffic Generator can transmit all type of image data format i.e. RAW, RGB, YUV, JPEG etc. It can Transmit High resolution Images at high speed over physical layer. It has 1 clock lane and 1 data lane. Data lane can transmit data up to 1 GHz data rate. It is lane scalable, so that it can be made for two, three and more data lane as per requirement of user. The HSSD Protocol Traffic Generator uses high speed serial interface in place of conventional parallel interface. Serial Interface can transmit data at high speed with consuming low power and lower EMI. It means Traffic generator consumes low power while transmitting data at high speed as compare to camera's which uses parallel interface to transmit data.

Cellular handset, mobile handset design engineers are faced with the high bandwidth requirements of camera and display as 3G services are provided on handsets. In 3G Handsets multimedia functionality are provided, this type of mobile handset is required to support high-speed transfer of high-resolution images. In addition, ISP (Image Signal Processor) or Multimedia Chipset demands high-speed clocks and high bandwidth in order to meet the expectations. The HSSD Protocol Traffic Generator can transmit high-resolution images at high speed and HSSD PHY IC can meet the requirement of high speed clock and high bandwidth transfer, Because of Traffic Generator meet the requirement of 3G handsets, It can be the emerging solution for camera of 3G handsets.

Traffic Generator is also used for interoperability testing for HSSD receiver devices ensuring their capability to work with various compatible sensor devices of different vendors. It is also check each stages of protocol with respect to defined standard of protocol. It is the real time implementation of HSSD Protocol.

It can also be used in future for making cameras with high image resolution, which can be transmit at high speed, so that camera can meets the requirement of high data bandwidth and high image resolution for 3G applications like video calling. Now this Traffic Generator is made for 1 clock lane and 1 data lane, which can support up to 1 GBPS data rate. Traffic Generator is lane scalable, In future as the user requirement increases; data lane of Traffic Generator can be scale to 2, 3 and more data lanes for meet the data rate requirement of the application.

REFERENCES

[1] Minyoung Eom, Yeong-Kyu Lim and Tae Ik Kang, "Performance Comparison of Camera interface methods in a mobile handset," 7th IEEE International Conference on Computer and Information Technology, Fukushima, pp. 835-840, 2007.

[2] Minyoung Eom, Jaegeun Oh and Seon Wook Kim, "Camera interface method in mobile handset and its performance comparison," International Conference on Parallel Processing Workshops, Xian, pp. 33-33, 2007.

[3] Draft Xilinx's ISE User Guide.

[4] Draft NC Launch User Guide.

[5] Draft HSSD Protocol Specification.

[6] Draft HSSD Physical Interface Specification.

[7] Steve Kilts, "Advanced FPGA Design: Architecture, Implementation, and Optimization," Wiley-IEEE Press, 2007.

[8] J. Bhaskar, "A VHDL Primer," 3e, Prentice Hall, 2002.

[9] Douglas L Perry, "VHDL Programming by Example," 4e, McGraw Hill, 2002.

[10] Peter J Ashenden, "The System Designer's Guide To VHDL-AMS," Morgan Kaufman Publishers, 2005.