STUDY AND FPGA BASED DESIGN OF BUS PROTOCOLS

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF TECHNOLOGY

IN

VLSI DESIGN AND EMBEDDED SYSTEM

Submitted by:

Deepak Chauhan

(2K19/VLS/05)

Under the supervision of

Ms. Kriti Suneja

Assistant Professor, Dept. of ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

August, 2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Deepak Chauhan, Roll No. 2K19/VLS/05 student of M.Tech (VLSI Design & Embedded system), hereby declare that the project dissertation titled "**Study and FPGA based design of BUS protocols"** which is submitted by me to the Department of Electronics and Communication Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Chauhan

Place: Delhi

Deepak Chauhan
(2K19/VLS/05)

Date: 31st August, 2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "**Study and FPGA based design of BUS protocols**" which is submitted by **Deepak Chauhan**, 2K19/VLS/05 (Department of Electronics & Communication Engineering), Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Kriti

Place: Delhi

Date: 31st August, 2021

Ms. Kriti Suneja

Department of ECE

Delhi Technological University

ACKNOWLEDGEMENT

I would like to express my deep gratitude and appreciation to all the people who have helped and supported me in the process of dissertation. Without their help and support, 1 would not have been able to reach this level of satisfaction with what 1 have learnt and accomplished during my Master's dissertation. First and foremost, I would like to express my deep sense of respect and gratitude towards my supervisor **Ms. Kriti Suneja**, Assistant Professor, Electronics and Communication Dept., DTU, for giving me opportunity to do my Major project of master's dissertation under her guidance. I am very thankful to her for giving me the opportunity to choose such an interesting topic by my own. I would also like to thanks the NPTEL Lectures for their valuable thoughts and knowledge, which motivated me to do better. Finally, none of this would have been possible without incredible support of my friends. They were always supporting me and encouraging me with their best wishes.

Chauhan

DEEPAK CHAUHAN
VLSI DESIGN & EMBEDDED SYSTEM
4th Semester
Delhi Technological University
(Formerly Delhi College of Engineering)

ABSTRACT

The ARM Advanced Microcontroller Bus Architecture (AMBA) is an open-standard interconnect protocol for connecting and managing functional blocks in a system on a chip design. With this bus architecture, it is easier to construct multiprocessor architectures with a high number of controllers and components. ARM gave various IPs such as, high performance microprocessor, high level cache, memory management unit, decoders, arbiters, controllers, etc. But to connect them, there needs to be an interconnect standard which is easy to design and built for low power applications. This de facto standard is known as the AMBA architecture. Later, the standard was made open for others to build and integrate with their own IPs. On chip price, latency, bandwidth, and number of IPs can be connected to the bus are taken as key design considerations while developing an interconnection standard.

For our work, target device used for obtaining synthesis results is ZYNQ-7000 FPGA, which is increasingly becoming popular among the FPGA engineers due to its advanced features that make it stand out among all the boards in the presence of an ARM cortex A9 chip which is the main reason for its usage as a system on chip (SoC). Having an integrated support for PCI Express also helps it to persuade its dominance over other FPGAs known to us.

For simulations and synthesis, XILINX VIVADO 2019.2 tool has been used. The implemented designs have been analysed in terms of hardware utilization (number of slices, which is comprised of Look Up Tables and Registers), timing (delay in ns) and power consumption (in Watts).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. SOC - System on Chip

2. AMBA - Advance Microcontroller Bus Architecture

3. APB - Advance Peripheral Bus

4. AHB - Advanced High Performance Bus

5. AXI – Advanced Extensible Interface

6. ASIC - Application Specific Integrated Circuit

7. IP – Intellectual Property

8. IC - Integrated Circuit

9. FPGA –Field Programmable Gate Array

10. RTL – Register Transfer Level

11. CPU - Central Processing Unit

12. GPU - Graphics Processing Unit

13. Gbps - Giga bits per second

14. PS - Processing System

15. PL - Programmable Logic

16. IOP - Input Output Peripherals

17. BW - Bandwidth

18. APU - Application Processing System

19. STA - Static Timing Analysis

CHAPTER 1

INTRODUCTION

The Advanced Microcontroller Bus Architecture (AMBA) is a well-known open standard, on-chip communication standard, and on-chip interconnect for the alternate organizing and courting of squares in a design. AMBA as a structure aids a wide range of designs, regardless of how many controllers and peripherals they include. System on-chip designs (SOCs), Application-Specific Integrated Circuits (ASICs), and Anomalous state embedded tiny scale controllers all use ABMA for on-chip delivery [1].

AMBA makes architectures with several processors and a large number of controllers and peripherals easier to manage. However, AMBA's reach has grown over time, and it now encompasses far more than just microcontrollers. AMBA is now widely used in a wide range of ASIC and SOC products. Application processors, for example, are used in devices such as IoT subsystems, smartphones, and networking SOCs.

A modern SOC design incorporates a high level of integration of several design components, often known as Intellectual Property (IP), which is possible with shrinking technique technologies. To put it another way, a SOC is an integrated circuit that implements most or all of the capabilities of a completely digital system. Figure 1.1 shows a block diagram of an AMBA-enabled SoC.

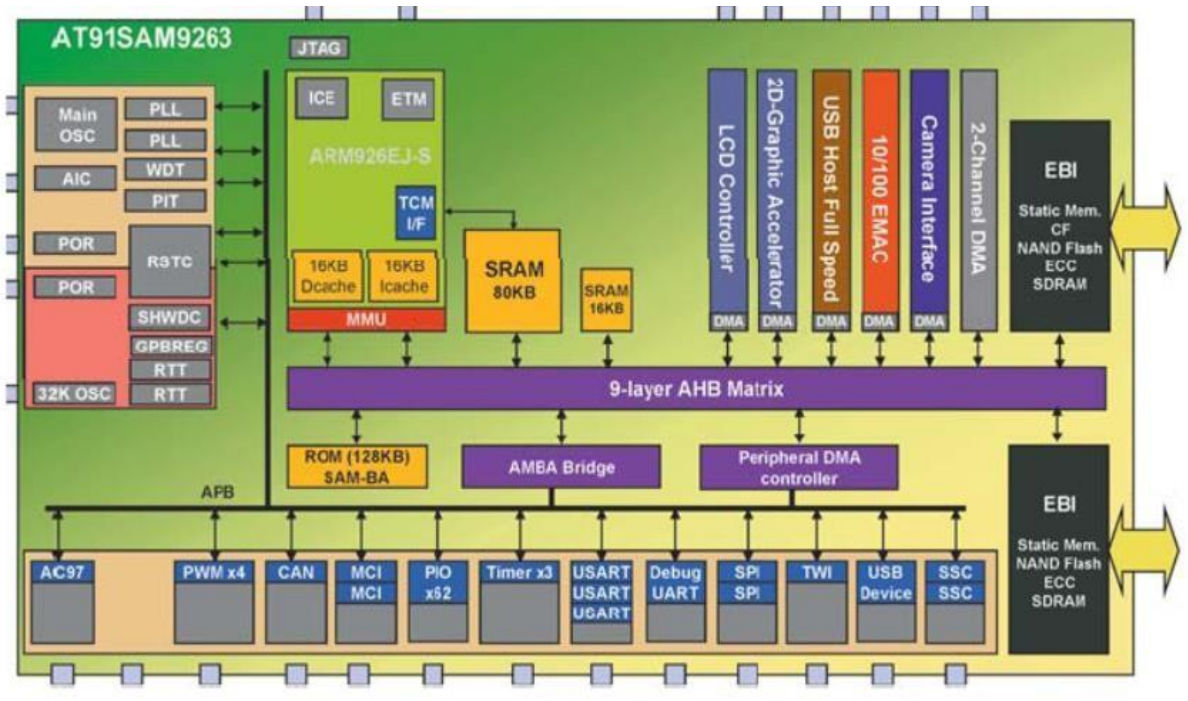


Fig 1.1 SOC System Block Diagram [2]

A SOC design may consist of one or more programmable components consisting of general-purpose processor cores, digital signal processor cores, or application-specific intellectual property (IP) cores and an analog the front end on-chip memory, IO devices, and numerous different application-specific circuits.

One of the biggest challenges in SOC design is the on-chip communication among the specific components. The specific bus protocols used for interconnection have a big impact on the overall performance of the SOC design.

Most of the time, the IP cores are designed with many different interfaces and communication protocols and this may be a problem while integrating into a SOC. To keep away from this problem, standard on-chip bus systems and protocols had been developed.

AMBA provides several benefits:

Efficient IP reuse: IP reuse is a vital aspect in reducing SoC improvement costs and timescales. AMBA specifications offer an interface standard that permits IP reuse. Therefore, thousands of SoCs, and IP products, are using AMBA interfaces.

Flexibility: AMBA gives the ability to work with various SoCs. IP reuse requires a common standard at the same time as supporting an extensive type of SoCs with distinctive power, performance, and area requirements. Arm offers a range of interface specifications that are optimized for these different requirements.

Compatibility: A standard interface specification like AMBA, permits compatibility among IP components from different design groups or vendors.

Support: AMBA is well supported. It is widely implemented and supported throughout the semiconductor industry, including support from third-party IP products and tools. Bus interface requirements like AMBA, are differentiated via the overall performance that they enable. The main characteristics of bus interface performance are:

Bandwidth: The rate at which data may be driven across the interface. In a synchronous system, the maximum bandwidth is confined with the aid of using the product of the clock speed and the width of the data bus.

Latency: The delay among the initiation and completion of a transaction. In a burst-based system, the latency figure often refers to the completion of the first transfer rather than the entire burst. The performance of your interface relies upon the volume to which it achieves the maximum bandwidth with zero latency.

1.1 DISTINCTIVE VERSIONS OF AMBA

AMBA1: Arm introduced AMBA in the late 1990s. The first AMBA buses were the Advanced Peripheral Bus (APB) and the Advanced System Bus (ASB). ASB has been superseded by more recent protocols, while APB is still widely used today.

APB is designed for low-bandwidth manage accesses, for example, register interfaces on device peripherals. This bus has an simple address and data phase and a low complexity signal list.

AMBA2: The AMBA High-performance Bus (AHB), a single clock-edge protocol, was introduced in 1999 with AMBA 2. At the AHB, a simple transaction consists of an address phase followed by a data phase. A MUX controls access to the target tool, allowing access to only one master at a time. Even while APB isn't necessarily pipelined for design simplicity, AHB is pipelined for overall performance.

AMBA3: In 2003, Arm introduced the third generation, AMBA 3, which has ATB and AHB-Lite. Advanced Trace Bus could be a a part of the Core Sight on-chip right and trace answer.

AHB-Lite could be a set of AHB. This set simplifies the planning for a bus with one master. Advanced extensible Interface (AXI), the third generation of AMBA interface outlined within the AMBA 3 specification, is targeted at high performance, high clock frequency system styles. AXI includes options that create it appropriate for high-speed sub-micrometer interconnect.

AMBA4: In 2010, the AMBA 4 specifications were introduced, beginning with AMBA AXI4 then AMBA 4 AXI Coherency Extensions (ACE) in 2011.

ACE extends AXI with additional signal introducing system-wide coherency. This system-wide coherency permits multiple processors to share memory and allows technology like massive little processing. At constant time, the ACE-Lite protocol allows unidirectional coherency. Unidirectional coherency allows a network interface to read from the caches of a totally coherent ACE processor.

The AXI4-Stream protocol is intended for one-way knowledge transfers from master to slave with reduced signal routing, that is good for implementation in FPGAs.

AMBA5: AMBA 5 CHI (Coherent Hub Interface) was given by ARM within the year 2013 to empower a superior and versatile framework on-chip innovation. It bolsters non-blocking sound information exchanges between processors utilizing stores. This is often used by Cortex-A57, Cortex-A53 processors, Core Link DMC-520 Dynamic Memory Controller, and Core Link CCN504 Cache Coherent Network.

Figure 1.2 shows the AMBA diagram, which includes a high-performance ARM processor, high-bandwidth on-chip RAM, UART, and an external memory interface, among other features.

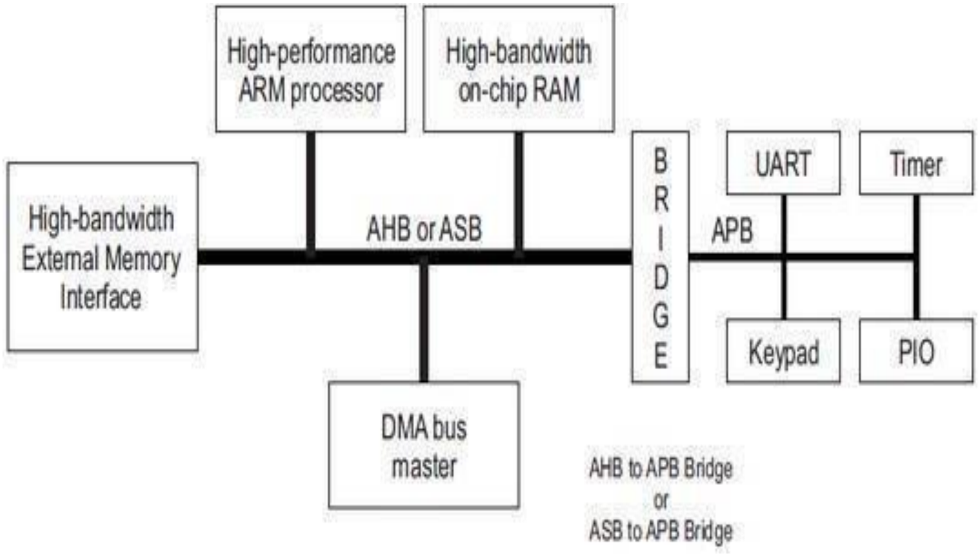

High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
BRIDGE
APB
UART
Timer
Keypad
PIO
DMA bus master
AHB to APB Bridge or ASB to APB Bridge

Fig 1.2 AMBA Bus Architecture diagram [1]

Advanced Peripheral Bus (APB)

Advanced Peripheral Bus is used as a subset of AMBA-based designs to reduce the complexity connected with interfacing and power utilisation. APB is used to connect peripherals with low transfer speeds. Framework execution will be improved thanks to APB. The two main segments of this transportation are the APB Bridge and the APB Slave. APB Bridge is a specialist in the field of transportation. The master is only present for single transport in AMBA-based outlines.

Address hooking, a strobe flag PENABLE, and a choice flag SSELx are the main components of APB Bridge. They set information on APB for composing exchange and make APB information accessible for examined interchange. APB Bridge is aware of a variety of generating parameters. Information and yield parameters are the names of these parameters. The APB slave's interfacing is said to be extremely adaptable. The APB slave task is managed using the same APB timing parameters.

Sclk, SRESETn, SADDR [31:0], SSELx, SENABLE, SWRITE, SRDATA, and SWDATA are the fundamental flags that govern APB's task. APB3 v1.0 is an APB set. SREADY and SLVERR, in addition to alternate signs of APB, are used in these two supplementary signals. The most recent improved adaptation of APB is APB4v2.0. These signals are used to write and read data.

Advanced System Bus (ASB)

ASB will be used as an area of the various inserted microcontrollers as an associate elite pipelined transport. It helps to connect different processors, external memory interfaces, and on-chip memories. ASB will provide the first benefits, such as burst exchange, various transport master assistance, and better-pipelined activity.

ASB master, ASB slave, ASB judge, and ASB decoder are all guideline portions that cannot be avoided. Through address and control information, the ASB master begins to generate the browsing workouts. Various transportation professionals are available in ASB; nevertheless, only one of them will be granted access. ASB Choose can assist professionals in gaining access to ASB. ASB slaves have a standard limit of responding to scrutiny and creating assignments.

The three types of exchanges that can happen through ASB are non-sequential, sequential, and address-only. DSELx, BWRITE, BWAIT, BTRAN [1:0], BPROT [1:0], BSIZE [1:0], BnRES, BLOK, BLAST, BERROR, BD[31:0], BCLK, BA[31:0], AREQx, and AGNTx are the characteristic signals used in this transport.

Advanced High-Performance Bus (AHB)

Massive and elite transit use AHB, which may result in increased data transmission activity. A top level perspective will achieve highlights such as split exchanges, higher information transport setup, burst exchange, single clock edge activity, and so on, thanks to AHB. AHB is typically utilised on ARM7, ARM Cortex-M, and ARM9-based designs. AHB master, AHB slave, AHB decoder, and AHB authority are all part of the AHB framework setup.

AHB uses address and control to read and compose tasks. Only one master will profitably use the transport at any given time. The slave of the AHB reacts to the master of the AHB. The slave responds to the read or compose operation using the address. The slave to the master recognises the status of the knowledge exchange. The status indicates whether the information exchange was a success, a failure, or a pause. AHB arbiter and decoder have the same capacities as ASB.

RWDATA [31:0], RSELx, RRDATA [31:0], RREADY, RRESP [1:0], RSPLITx [15:0], RMASTLOCK, RMASTER [3:0], RGRANTx, RLOCKx, and RBUSREQx are RWDATA [31:0], RSELx, RRDATA [31:0], RREADY, RRESP [1:0], RSPLITx [15:0]. AHB-Lite v1.0 features a high-speed exchange motion. Aside from the basic AHB signals, this adjustment employs a variety of banners to increase activity.

Advanced Extensible Interface (AXI)

ARM introduced AXI v1.0, a burst-based convention, during the third phase of AMBA. It delivers a higher level of execution, a higher rate of return, and a faster task. There is a separate stage for address and information. It exchanges the data that byte strobes use. Burst exchanges can be used to solve the problem. The distinctive classifications of signs introduced in AXI are compose information carrier signals, compose address carrier signals, compose reaction carrier signals, read address carrier signals, read information carrier flags, and low power interface signals. There are five unique channels available for reading and writing. AXI's other main strengths are the completion of request exchanges and the expansion of enlistment stages. AXI4 – light is a more powerful version of AXI. It modifies the essential AXI indications. This set makes use of a predetermined information transit dimension and strobe backings. The most recent modification of AXI is AXI-Stream v1.0.

AXI Coherency Extensions (ACE)

Expert is an AXI upgrade that includes third-level reserves, on-chip RAM, peripherals, and external memory. The AXI read and compose channels are built for a 64-bit or 128-piece interface in this case. It is the foundation for 1:1 clock proportions in processor clocks. It will also run with a large number of CPU clocks. Professional is an ARM Cortex-A processor that includes the Cortex-A7 and Cortex-A15. ACE masters, ACE lite masters, and ACE Lite/AXI slaves are interconnected portions within the master. Skilled offers coherency at the framework level a structure. The essential flags of ACE are read data channel signals, browse address carrier signals, snoop carrier signals, write address carrier signals, and response signals.

A set of ACE may be Pro-Lite. Ace components that do not have instrumentality intelligible reserves use Pro-Lite. They'll show whether the issued exchanges can be controlled within the instrumentality coherent stores of various bosses or whether they can be used to facilitate obstruction exchanges. On the read address channel, it adds more flags, and on the compose address channel, it adds more flags. Snoop channels, snoop indicators, and reaction signals will be blocked by Pro-Lite.

Advanced Trace Bus (ATB)

To debug the framework, the Advanced Trace Bus (ATB) supports an interchange of information around the Core Sight. It supports bite-sized bundles and, as a result, the control signals used to display the number of bytes significant in each cycle. This mode of transportation uses the signals ATCLK, ATCLKEN, ATRESETn, ATREADY, ATVALID, ATID [6:0], ATBYTES [m:0], knowledge [n:0], AFVALID, and AFREADY.

1.2 GOALS OF THE AMBA DETERMINATION

The goals of the AMBA particular are:

- It will upgrade the inserted small-scale controller things with a focal handling unit on SoC.
- It will upgrade the reusability of fringe and IPs and make the design simple.
- It will configure fewer potential interfaces.

1.3 THESIS ORGANISATION

The thesis report is organized as follows :

Chapter 1 gives a brief presentation about the Advanced Microcontroller Bus Architecture (AMBA), advanced peripheral bus (APB), advanced high-performance bus (AHB), and advanced extensible interface (AXI).

Chapter 2 is a brief study of AMBA setups that have existed up to this date. It also highlights the current arrangements' flaws.

Chapter 3 gives us a deep insight into the FPGA board, which is utilized for synthesis purpose.

Chapter4 AMBA protocols, which include advanced peripheral bus (APB), advanced high-performance bus (AHB), and advanced extensible interface (AXI), explain the framework outline and execution.

Chapter 5 presents the simulation results and analysis.

Chapter 6 consists of the work's conclusion and future scope.

CHAPTER 2

LITERATURE SURVEY

ARM Confined demonstrated AMBA, an open-source transport tradition, in 1999 [2]. The author presented an AMBA 2.0 [1] that distinguishes three modes of transportation: ASB, APB, AHB. It investigates all modes of transportation's testing systems. The interface module is a piece of software that allows you to In terms of its restricted state machine, AMBA AHB may plainly be utilized as a conventional joining module because it can read and write data. As it communicates with the peripheral contraptions, the connection module AMBA APB is melodic, displaying a low repeat [3].

Kiran Rawat in [4] proposed an intricate interface between AMBA ASB and APB is the goal of the combination. Verilog dialect with restricted state machine models designed in Model Sim Variant 10.3 and Xilinx-ISE outline suite were used to design the utilization rundown and power reports. The usage of APB Extension necessitates the employment of a referee and a decoder. In the AMBA ASB APB module, the master makes contact with APB transport. The judge begins conversing with the transport after determining on the master's status. The decoder selects a transport slave using the specific address lines, and the slave answers to the transport master with an affirmation. As the outline complexity of the structures rises, the power consumption of SoC structures becomes increasingly essential. The power reports separate the various power components that contribute to power use [5].

Kiran Rawat et al. introduced [6] AMBA APB, a variant of AMBA that provides the lowest power consumption and transmission capacity. For this, a Verilog dialect APB Extension with Reset Controller configuration was used. BnRES and Power-on Reset (PO Reset) conditions are presented by the reset controller so that Meta stable characteristics may be propagated and glitches can be kept at a strategic distance. Power report demonstrates that the different power segments contribute in the power utilization by APB connect design. As the consequence,: extension under PO Reset conditions, On-chip add up to control utilization is 9.52%, Chain of command control utilization is 29.12% and dynamic supply control utilization is 28.89% not as much as Scaffold under no PO Reset conditions. As a result, when Scaffold is designed under Power-on Reset circumstances, it may make effective use of intensity. This is done using the Verilog dialect, which is used to create restricted state machine models and test seats. Model Sim Rendition 10.3 is used to show and reproduce the APB Extension and Reset Controller. For combination and power reporting, the Xilinx-ISE outline suite, version 13.4, is used.

Kiran Rawat et al. proposed [7] the fundamental test for building a plan not just to outline but to organized and synthesize RTL code to plan vitality and streamlined in control utilization. The author's goal is to make the AMBA APB extension a reality by properly organizing framework assets. For this, a replication and synthetization of the span interface is proposed, with the goal of achieving the lowest power consumption and data transfer capacity possible between AMBA fast ASB and low speed APB transports. When a distinction is made between the entrance timings of clock flags, clock skew appears. Limit clock skew can be dealt with either a swell counter or a three-piece up or down counter method. The article uses Verilog HDL to implement an APB Scaffold with a clock skew reduction technique.

Jasmine Chhikara et al. proposed [8] the unit which comprises of littler practical squares called subsystems or module. These modules must be in sync with one another and provide resources for the framework to function properly. The problem arises when one subsystem adopts the same norms as others. Each module uses a different piece rate or baud rate for data exchange, which might be non-concurrent or synchronous. The author also shows how to share information by starting with one convention and then moving on to the next. It takes advantage of I2C's adaptable conventions, making it ideal for usage with the APB AMBA convention. The proposed engineering is a bridge between the I2C Master and the APB salve, allowing information to flow from an I2C-enabled module to an APB-enabled module. The data is transferred in a state of harmony with the area clock, from serial (I2C) to parallel (APB) to serial (I2C). This creates a bidirectional interface between I2C-enabled and APB-enabled modules.

Ashutosh Gupta provided [9] an on-chip representation backdrop for better implanted microcontrollers. This figure depicts the physical implementation of the AMBA advanced system bus (ASB) and advanced peripheral bus (APB) connection modules. To maintain the clock skew to a minimum, a three-piece swell counter was utilised. For showing and reenactment, Model Sim Form 10.3 is utilised, and test chairs are developed for this purpose. For the amalgamation and experience, the RTL Compiler is used, while the Xilinx-ISE plan suite is used to eliminate the union and utilisation rundown. An sophisticated execution framework is employed for the physical plan.

Kanishka Lahiri and Anand Raghunathan et al. proposed [10] the complex System on-chips (SoCs), the framework level on-chip correspondence design is increasing as a huge wellspring of intensity utilization. Supervision and reorganization are the vital segments of SoC control which requires the qualities of the capacity utilization. While compelling, they just address a constrained piece of correspondence design control utilization. A cutting-edge correspondence engineering, involves few segments, for example, transport interfaces, mediators, scaffolds, decoders, and multiplexers, notwithstanding the worldwide transport lines In addition to statistically supporting the perspective that on-chip correspondence is a critical focus for framework level power streamlining, their work demonstrates (i) importance of fully considering correspondence engineering, and (ii) the opportunities for control reduction that exist through careful correspondence engineering design[11-13].

Ge Zhiwei et al. discussed [14] a novel picture on on-chip and CMOS sensor, which is applied to the APB transport. The suggested design shows shading picture planning difficulties and highlights the contrasts between the proposed structure and the conventional image, which considers pipeline as a component of cutting-edge still cameras. This work combines two advantageous automobiles white modify methods to adjust the three self-ruling shading channels, taking into account the gear utilisation and power requirements. The suggested technology, which is linked to FPGA, can effectively restore the picture concept of rough data [15-17].

L. Benini, A. Macii et al. proposed [18] the encoding and decoding counts that will interface the way of reasoning that will limit the normal number of modifications on heavily stacked overall transport lines at no cost in correspondence throughput (i.e., single word is transmitted at each cycle). Given knowledge on word-level estimates, the perceiving feature technique grows low-change development codes and gear execution of encoders and decoders without relying on organizer's sense. A correct system that is suitable to low-width transports, and furthermore deduced procedures that scale well with transport width. In addition, display a flexible building that normally changes encoding to lessen advance development on transports whose word-level estimations are not known from the before[19].

J. Y. Chen *et al.* proposed [20] a technique that proficiently diminishes the exchanged capacitance of the transport. The power devoured by the transport can, consequently, diminished. The fundamental of the transport division is to parcel the transport into a few transport portions isolated by pass transistors. Especially transmission gadgets are situated to adjoin transport fragments, in this way, most information Correspondence can be accomplished by exchanging a little segment of the transport sections. Accordingly, control utilization and delay are both decreased. Exploratory outcomes got by reproducing a defer display and a power show exhibit that the proposed divided transport framework decreases transport control by around 60%-70% and enhances basic transport delay by around 10%-30%.

Recent advancements in SoC technology have enabled the integration of many devices on a single chip, paving the path for more compact devices. The various gadgets from various sellers, each performing a particular function, are all combined on a single chip. These components can be linked using bus-based protocols to enable effective communication. The AHB protocol is the most widely utilized communication method. The AHB protocol allows devices to communicate at fast speeds. Because of their bulkiness, processors are gradually growing quicker, but memory are becoming slower. This becomes a problem since the data retrieval speed does not match the processing speed. As a result, a fast memory controller is necessary, one that can match the CPU speed to the memory speed to ensure effective communication .It's difficult to interface SDRAM with AHB since SDRAM's latency isn't limited to one cycle, forcing AHB to sit idle throughout that period. As a result, bus resources are being used inefficiently [21-22].

K. Shaikh and colleagues created an SDRAM-specific memory unit. When a query is made concerning recently consumed data, for example, the controller's internal memory is searched first before fetching it to the memory. Because the AHB bus architecture provides for increased performance, increased clock frequency system modules, this controller was designed to work with the AHB (AMBA) bus architecture. It's the building block for increased performance systems [23].

The complying AMBA bus hardware IP presented by Acasandrei et al. is a modularized, extremely flexible, reduced power , and technology-independent core written in the HDL language. The Viola-Jones method, which is one of the most frequently used face identification techniques, is accelerated by the IP core. The hardware accelerator IP is used in an embedded face detection system based on the LEON3 Sparc V8 CPU. The authors describe their techniques, challenges, and performance findings for software, hardware, and system level design [24].

Modern SoCs include multi-core clusters and sophisticated peripherals, for which the current AHB protocol, which supports low-complexity shared buses, cannot meet the expectations of today's world high-speed SoCs, due to a number of protocol impediments, including the fact that the AHB has only one outstanding transaction, no full-duplex mode, and only one channel-shared bus that is directed to the employee AXI bus. By identifying how the five-channels in the bus operate independently, as well as the handshaking concept, a more realistic comparison with the advanced AXI bus is drawn [25].

The AMBA AXI4 bus, which is the best in terms of throughput, latency, and high performance/ frequency, is used with single or multiple channels, and the connection block encapsulates the arbitrator, decoder, and multiplexers .The arbiter monitors the priority to access/release the bus to one of multiple masters who begin transactions at the same time using an arbitration technique. The decoder decodes the master's address and control and delivers the transaction to one of 16 slaves for basic and burst read/write operations [26].

The design complexity of SOC is increasing day by day as the result it increased consumer demands. As a result, there is always a productivity gap, and the appropriate protocol is picked for each application. To improve connectivity performance, QoS, and reduce wiring congestion, a migration from AXI4 to AXI4-Lite is required, which allows the processor or masters to access the registers (small and mini peripherals). We choose AXI4-Lite. Using the aforementioned references and keeping an eye on the situation, if the Master agent regularly wants to access information that may be in small registers for which it uses basic registers like cache to interact, an effective and proper bus interface must be chosen to meet the need. This prompted the development of the Verilog HDL work “Design and interface of AXI4-Lite Master core with a modified Memory” [27].

CHAPTER – 3

ZYNQ 7000 FPGA FAMILY

This FPGA series uses a SOC architecture, which combines a dual-core ARM Cortex A9 CPU with 28nm Programmable Logic into a single device. Because the A9 chip is also at the base of the Processing System , this FPGA is an excellent candidate for usage as a SOC. There's also an on-chip boot Read-only Memory (ROM), 16- to 32-bit external memory interfaces, and a few peripheral connection interfaces. It provides a completely programmable alternative to ASIC and system-on-chip customers, as well as a versatile platform for launching new solutions. For computationally intensive and performance-demanding applications, this serves as a highly integrated and optimized alternative. The Z-7010, Z-7020, and Z-7030 devices are members of this FPGA architecture, which is primarily focused on automotive applications [29].

The ZYNQ 7000 is optimized for maximum design flexibility and performance per watt. These SOCs are cost-optimized entry points with ARM single Core processor acquaintance with 28nm Artix 7 based Programmable Logic. The system on chips is an ideal candidate for numerous applications in the motor control field and vision engineering. This family contains up to 10 devices to be chosen from that may be single or dual-core, hence allowing a scalable platform for the consumer. FPGAs are the ideal candidates for implementing sorting algorithms due to their unparalleled features like parallelism, low latency, high bandwidth, faster processing speed, in-built processor core availability among others [24].

The Zynq-7000 SoC devices are able to provide numerous applications including:

- Automotive driver assistance: lane departure, blind-spot detection
- Wireless applications, Reliable Ethernet
- Embedded prototyping
- Software acceleration for DSP functionality
- Time-domain reflectometer

The ZYNQ facilitates the mapping of software and custom logic in the PL and PS, allowing for the development of unique and distinct system functionalities. The architecture of the ZYNQ SoC is shown in Figure 3.1. The following are the key components:

- Processing System (PS)
- Application processor unit (APU)
- Memory interfaces
- I/O peripherals (IOP)
- Interconnect
- Programmable Logic (PL)

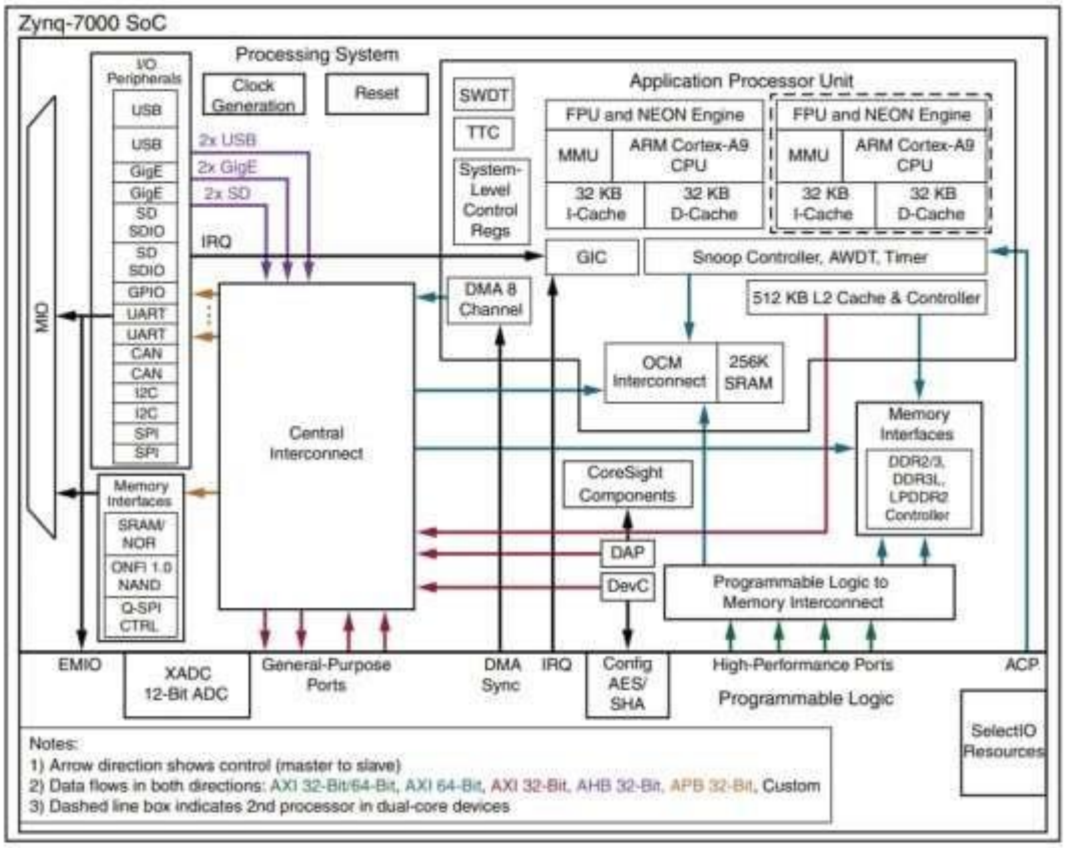


Fig 3.1 Architecture of ZYNQ FPGA chip [29]

3.1 HISTORY

It is an all-programmable system on a chip that is embodied by two hard processors, ADC block Programmable Logic (PL), and a lot more components embedded in one Silicon chip only. Before the innovation of the ZYNQ came into practice, the processes were coupled with an FPGA which made the communication between the Processing System & the Programmable Logic quite intricate and its layout difficult for the engineers to understand. The advanced extensible interface (AXI) standard is used as a means of interfacing across different elements present on the ZYNQ architecture which thereby accounts for the high bandwidth and low latency present in connections. A soft-core processor such as Micro blaze was being used by the users before the ARM processor was implemented inside the ZYNQ device; as its heart. The upper hand provided by the Micro blaze to date is the flexibility of the processor instances within a design. On the other hand, ZYNQ delivers significant performance enhancement with the encompassing of the hard processor in the ZYNQ [30]. Also, the cost to market and the physical size get reduced by simplifying the system to a single chip.

Fig 3.2 shows the significance of AXI interface in ZYNQ.

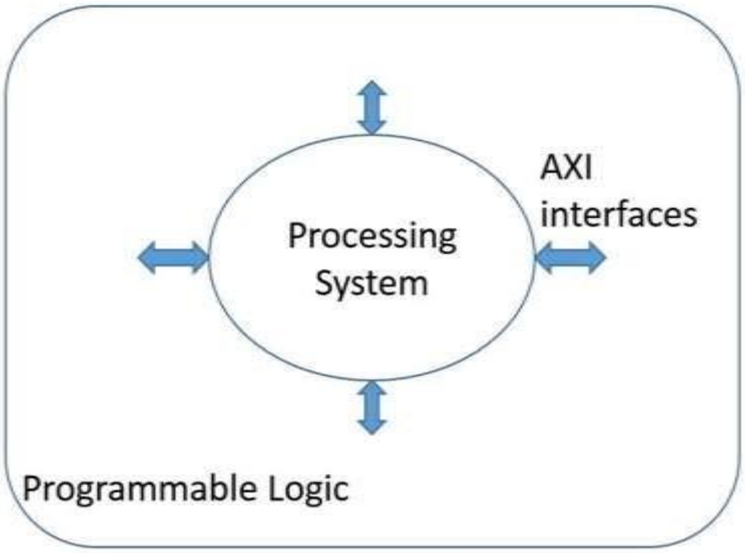


Fig 3.2 ZYNQ overall view [30]

3.2 ZYNQ DESIGN FLOW

This design flow has some steps that are recurrent with a regular FPGA. We start the design cycle by first defining the requirements and specifications of the system, next the different tasks are assigned to implement in either the PL or PS which is called task partitioning. Because the overall performance of the system will depend on the task or function being assigned for implementation, so this stage is most important in the technology node be it hardware or software. The next step is the testing of the hardware and software development. We need to now identify the functional blocks related to the Programmable Logic in order to attain the design characteristics and also to congregate them as IPs and for facilitating the connections between all of these IPs, all the steps are hence governed with respect to the functionality of the Programmable Logic (PL). The software activity includes running on the PS, the code that is developed [30]. To wrap the design, system integration and testing are needed. Figure 4.3 gives the design cycle briefly.

3.3 PROCESSING SYSTEM UNIT

The Application Processing Unit (APU), memory interface, interconnect, and input-output (I/O) peripherals are the four primary blocks.

3.3.1 Application Processing Unit (APU)

Two Cortex A9 processor units are present in it along with the NEON Unit, Memory management unit, floating-point unit, L1 caches. Additionally, L2 cache and Snoop controls are also present in it. The representation of APU block diagram is given in Fig. 3.4.

1. NEON: implementation of the single instruction multiple data in the ARM processor is provided by this unit that acts as a catalyst to the DSP and the media algorithms

2. FPU: the floating-point unit operations are managed by this unit

3. Level 1 Cache: storing the instructions and the data separately we have a data and instruction cache

4. MMU: the virtual memory gets translated to the physical memory address by this unit.

5. Snoop control unit (SCU): its main task is to create interfaces among the processors 4 –way set associative L1 and L2 cache.

6. L2 cache: to check the currently updated value of a variable, the cache is shared between two processors

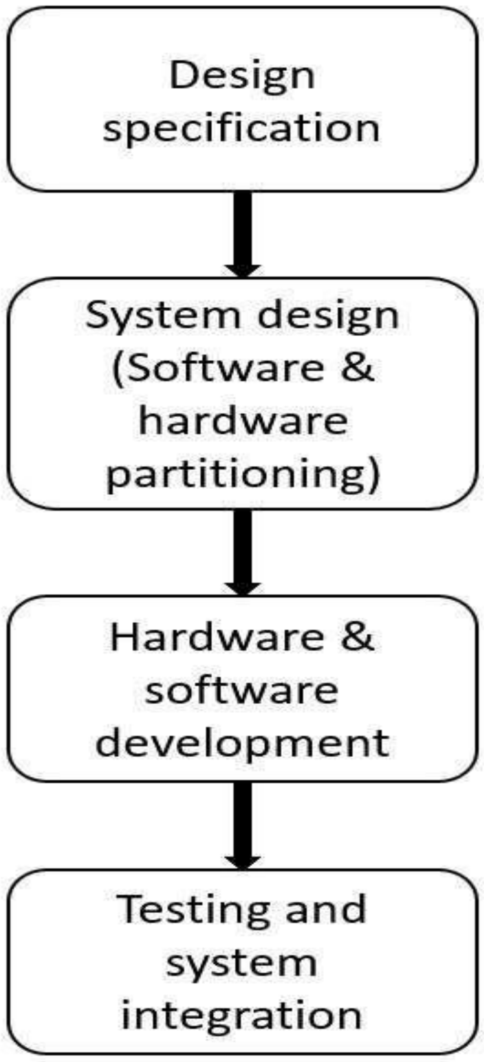

Design specification
System design (Software & hardware partitioning)
Hardware & software development
Testing and system integration

Fig 3.3 ZYNQ Design Flow Steps [30]

3.3.2 General Interrupt Controller (GIC)

Consists of two main components:

- 3 WDT (watch dog timers) (one per CPU and one system WDT)

- 2 TTCs (triple timers/counters).

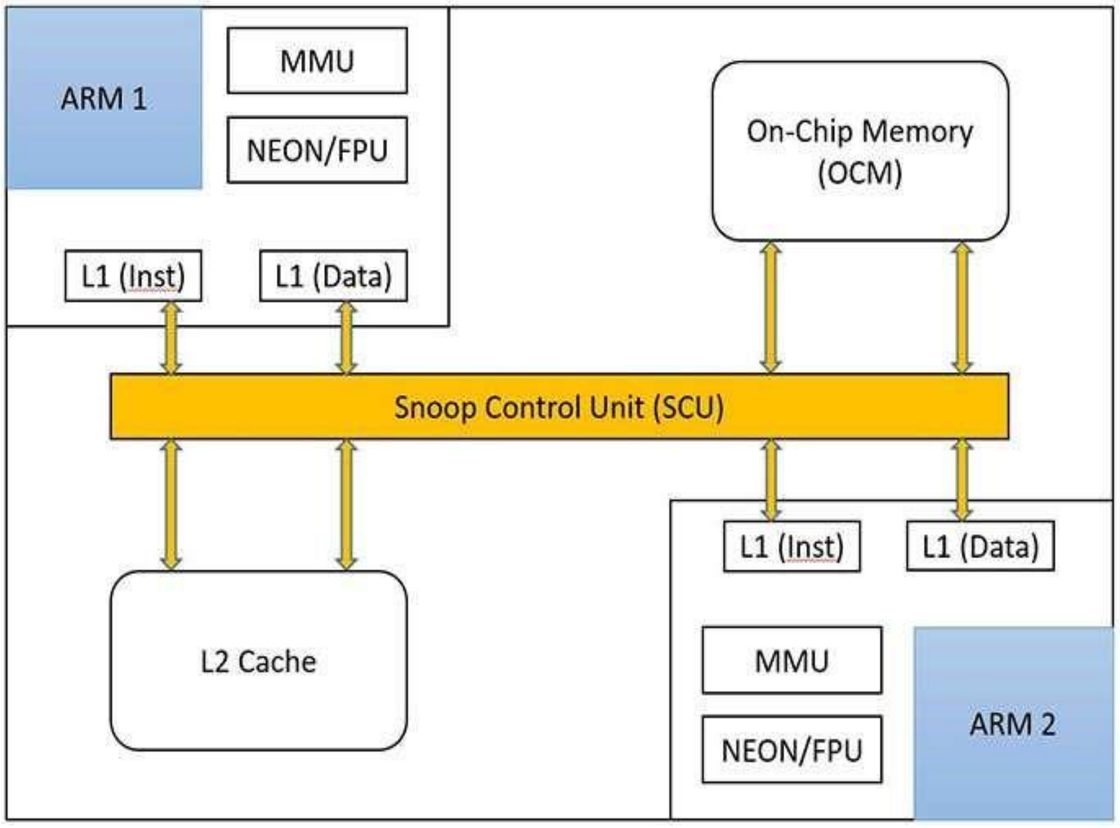


Fig 3.4 Application Processing Unit Structure [29]

3.3.3 Dynamic Memory Interfaces

The multi-protocol memory controller can be configured to provide 16 bit or 32-bit byte accesses using a single rank configuration of 8 bits,16 bits or 32 bits to a 1 GB address space [29]. It incorporates its own set of dedicated IOS and hence speeds of up to 1333 MB per second for the DDR 3 is supported. 4 AXI slave ports are featured for this purpose namely as :

- via the L2 cache controller: a 64-bit port is dedicated for low latency
- for the PL access: two 64- bit ports are designated
- all other AXI Masters share: one 64-bit port via the central interconnect

3.3.4 Static Memory Interfaces

It supports static external memory such as:

- 8-bit data bus up to 64-MB
- 8 bit parallel NOR flash up to 64- MB
- NAND flash support with 1-bit ECC

3.3.5 I/O Peripherals (IOP)

The data communication peripherals are present within the IOP unit. Its features are;

- Ethernet MAC peripherals: Tri-Mode
- Supports an external PHY interface
- High speed and full speed mode in the host, devices by the presence of two USB 2.0 OTG peripherals having 12 endpoints
- Makes use of the 32 bit AHB slave and AHB DMA master interfaces
- Two full CAN (Controller Area Network) bus interface controllers that have automotive applications
- Three peripheral chip select signals accompanied by 2 full-duplex SPI ports
- Two UARTS
- Up to 118 GPIO bits

3.3.6 Interconnect

A multilayered ARM AMBA AXI interconnect is used to connect the APU, memory interface unit, and the IOP to each other and also to the PL. This interconnection supports multiple simultaneous transactions of the master and slave; it's a non - blocking type. And is therefore designed with latency-sensitive masters which have the shortest path to the memory and the bandwidth-critical Masters having the highest throughput with the slaves through which they have to communicate [29]. The traffic generated by the CPU, DMA controller can be regulated by means of a block known as the quality of service present in the interconnect.

3.3.7 PS External Interfaces

They cannot be assigned as PL Pins and are hence designed specifically for the purpose of interfacing. The interfaces are encompassed by:

- Clock, reset, boot mode, and voltage reference
- 32-bit or 16-bit DDR2/DDR3/DDR3L/LPDDR2 memories

3.4 MIO (MULTIPLEXED INPUT – OUTPUT) OVERVIEW

There are up to 54 MIO present for multiplexing access to PS pins that can be used by the static memory interface and the PS interfaces, which can be mapped with the different peripheral pins at all steps of the interfacing framework [29]. The signal routing of the MIO block has been shown below in fig 4.5. If greater than 54 are required then we need to route this through the PL to the input-output associated I/O with the PL and are therefore referred to as the extendable multiplex input-output (EMIO). MIO_PIN [53:0] configuration registers located in the SCLR registers set is controlled by the signal routed through the MIO block [29]. We can program any one of the total pins present on the MIO pins to the reference clock of an external CAN controller.

A typical module block diagram of EMIO to PL has been shown in Fig 3.6.

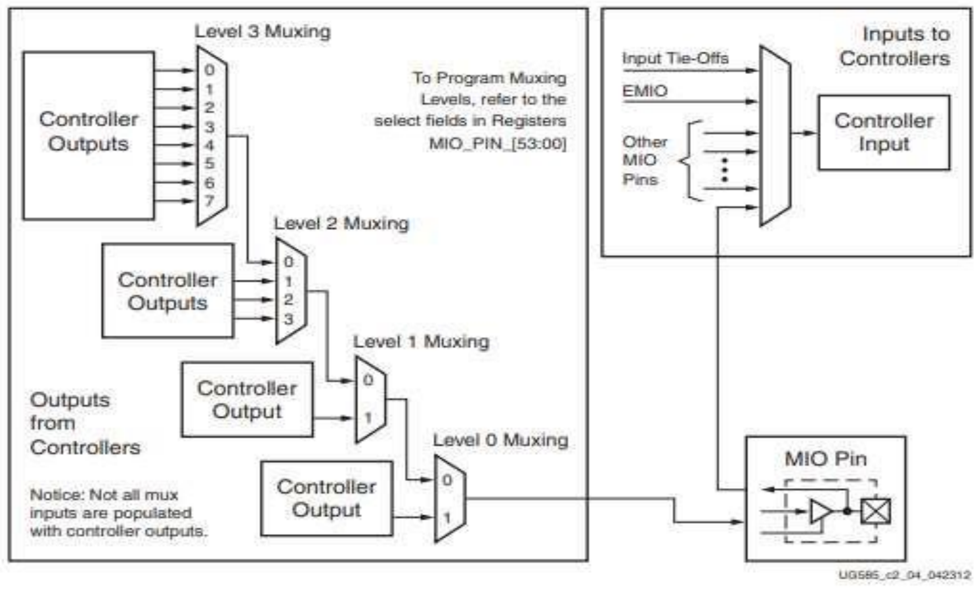


Fig 3.5 MIO Signal Routing [29]

3.5 PROGRAMMABLE LOGIC STRUCTURE

It is comprised of configurable logic blocks (CLBs) which are housed by slices like any other FPGA we are familiar with. Any slice contains a combination of 8 flip flops + 4 LUTs and is accompanied by a switch matrix, also there are DSP slices and Block RAMs as well. Fig 3.7 shows the structure of the Programmable Logic. Its main components are:

1. Slice: It is embodied by resources for implementing the combinatorial and the sequential circuits of the design.

2. Look-up-table (LUT): For implementing a logic function of inputs up to 6 or more RAM and ROM shift registers are used.

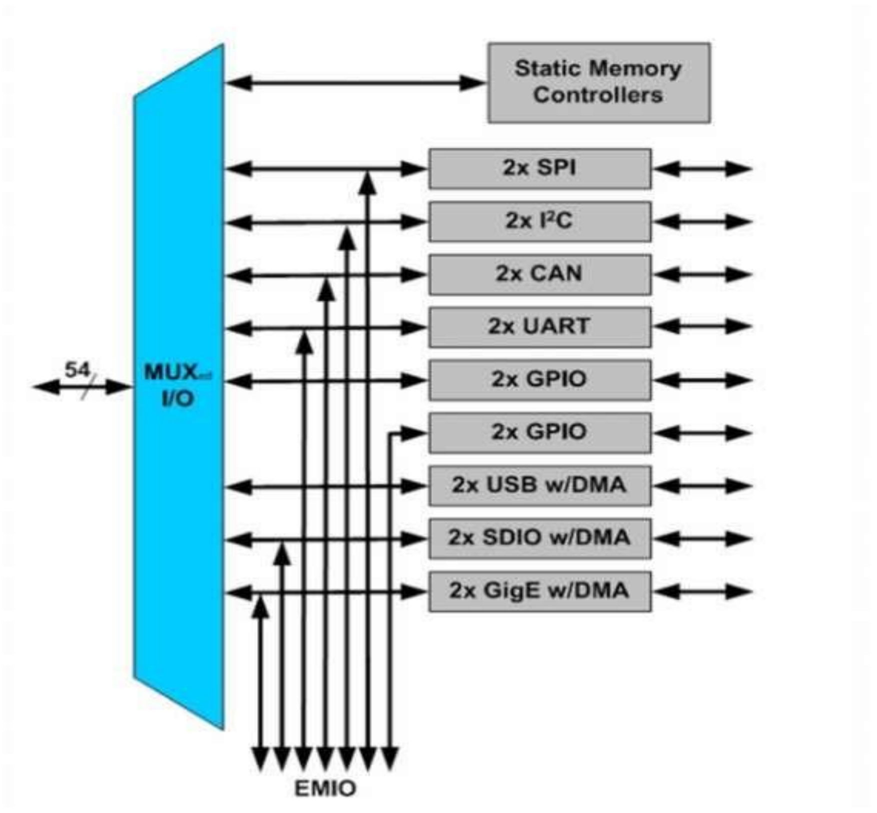


Fig 3.6 Module block diagram of EMIO to PL [30]

1. Flip Flop (FF): Usually employed for the implementation of a 1-bit register with a reset functionality.

2. Switch Matrix: The connections among different parts present within the combinational logic blocks as well as with other CLB and other parts of the programming logic structure.

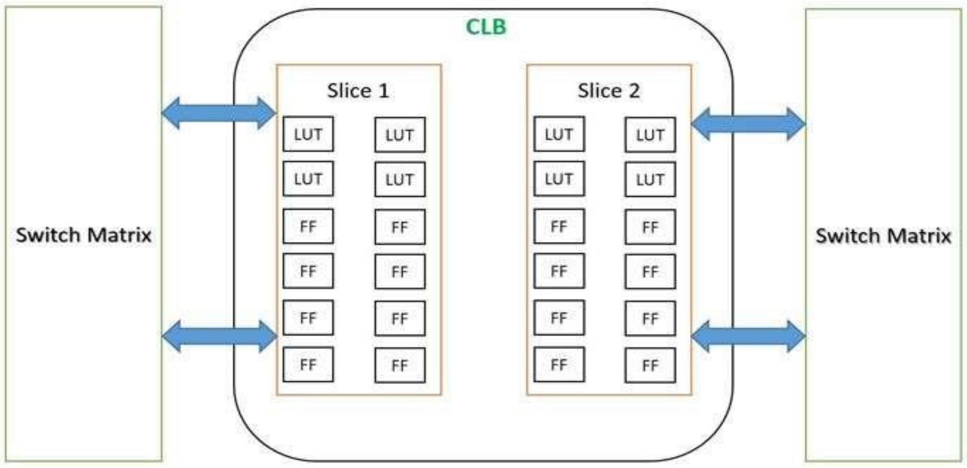


Figure 3.7 Structure of the PL [30]

3.6 DISTINCTIVE FEATURES

The key features of the ZYNQ 7000 are:

• 1 GHz CPU Frequency

• GPIO has four 32-bit banks

• All programmable SOC

• Level 1 cache : 32 KB each

• Level 2 Cache : 512 KB

• Up to 118 GPIO bits

• Over 85k logic cells

• ARM v7 architecture with Trust – Zone security

• 100 Gb/s of I/O bandwidth

• 8 channel DMA ; 4 channel dedicated specifically to PL

• 8 LUTs + 16 FF per CLB

• Two 12 bit ADCs (XADC)

• 36 KB Block RAM

• 25 bit pre-adder, 18 x 25 signed multiply

- ARM Cortex A9 Microprocessor chip
- PCIe supports upto 8 lanes, Gen2 speed
- 4 AXI ports : configurable as 32 or 64- bit interfaces
- 1KB deep FIFO ; 32 word buffer for read acceptance
- 16 Interrupts available
- Upto 220 trans receivers for enhanced capability

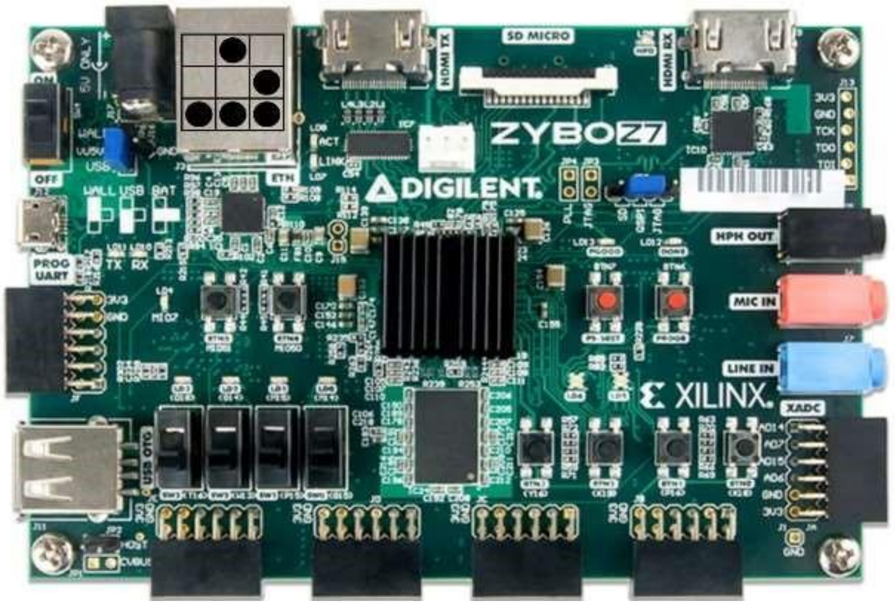


Figure 3.8 ZYNQ 7000 FPGA board [30]

- Scatter-gather DMA capability
- Two USB 2.0 OTG peripherals
- 8-bit PHY external interface
- 1 Mb/s high Speed UART’s

- XADC , JTAG interfaces
- Processor configuration access port (PCAP) for facilitating chip security
- PS boot image authentication
- 8 Clock Management Tiles (CMT)
- CMT has Mixed – Mode Clock Manager (MMCM) & PLL.

CHAPTER 4

FRAMEWORK OUTLINE AND EXECUTION

4.1 Advance Peripheral Bus Protocol (APB)

The IP for the inter-Advanced peripheral bus (APB) protocol is presented in this design. The current VLSI design environment is characterized by high speed, complex functionality, and a short time to market. A reuse-based SoC design method has become important to solve these challenges. The APB Protocol and its slave Verification are the subjects of the project. The goal is to put DUT to the test. This model is used to communicate between the slave and the master. The complete design has been verified and coded in Verilog.

Figure 4.1 depicts the APB protocol's master-slave communication.

APB Block Diagram

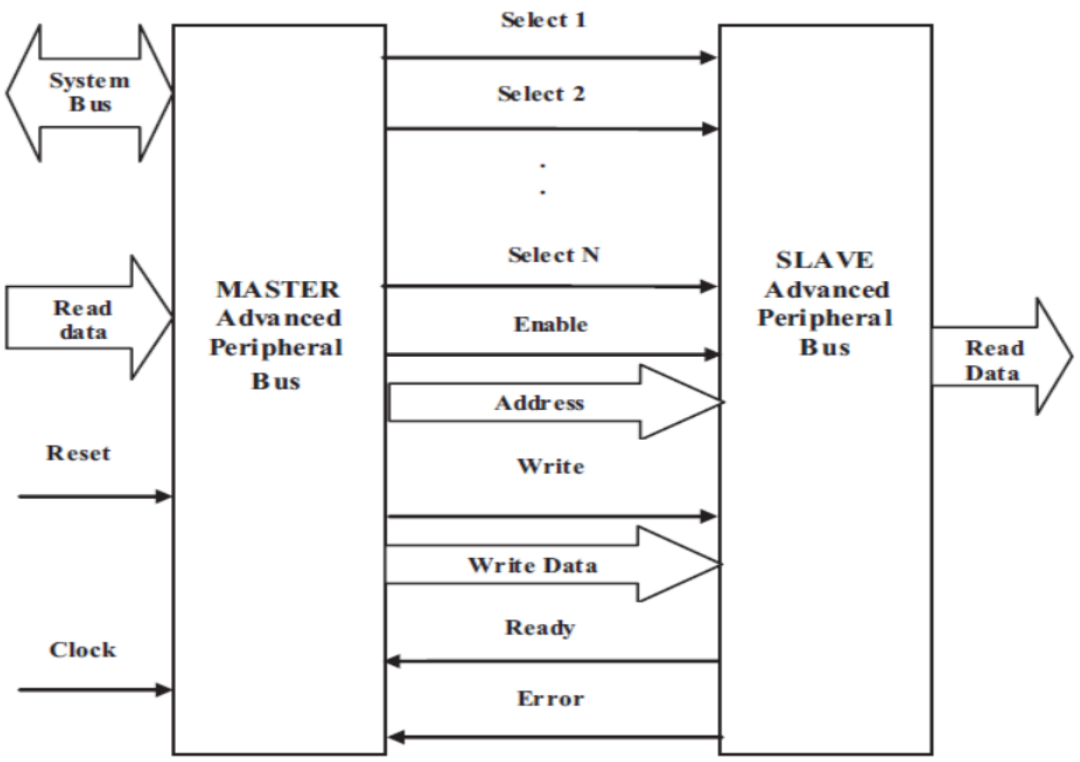


Figure 4.1 Interfacing of APB Master & Slave [5]

Table 4.1 List of APB signals [5]

SIGNALS	**DESCRIPTION**
Clock	For Positive edge clock all sign changes
Reset	Dynamic high reset signal
Address	Address bus can be 4 bits or 8 bit wide
Select signal	Select. Each slave device has a this signal, this demonstrate what slave gadget is chosen for information
Write	Direction. This signal high indicates a write access when it is low indicates read access
Ready	The slave utilizes this sign to expand an exchange
Read Data	The chose slave drive this transport during read cycles when wr is low. This transport is 32 bit wide
Error	This sign shows an exchange disappointment

4.1.1 Operating states of APB

IDLE is the most common condition of the APB. The bus enters the SETUP state and asserts the relevant select signal, PSELx, when a transfer is necessary.

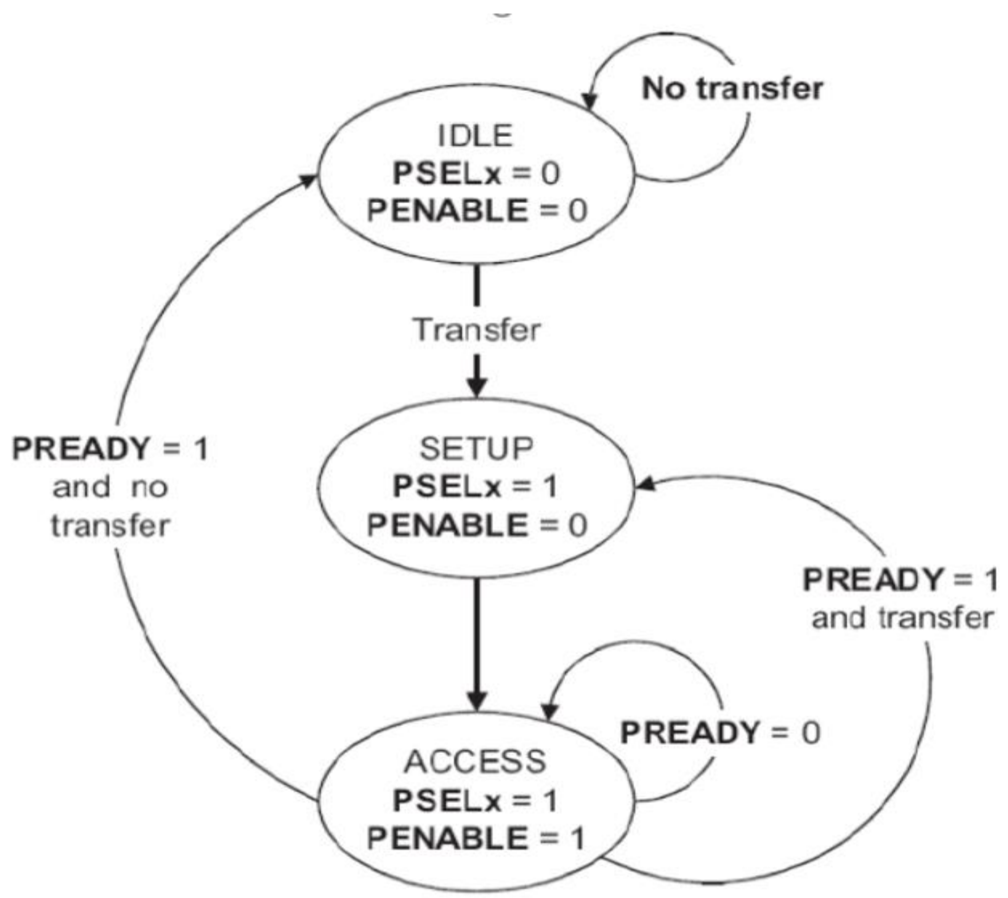


Fig 4.2 State Diagram [2]

The bus only stays in the SETUP state for one clock cycle before moving to the ACCESS state on the clock's rising edge. In the ACCESS state, the ACCESS will enable signal, PENABLE, is asserted. During the transition from the SETUP to the ACCESS state, the write, write data signals, select, and address must all remain stable. When the slave sends the PREADY signal, the ACCESS state determines when to exit. The first condition is that if the slave has PREADY LOW, the peripheral bus remains in the ACCESS state. The second condition is that if PREADY is held HIGH by the slave, the peripheral bus remains in the ACCESS state. The slave drives PREADY HIGH, then exits the ACCESS state and the bus returns to the IDLE state if no additional transfers are necessary. It then repeats the cycle [2][3].

4.1.2 Write cycle

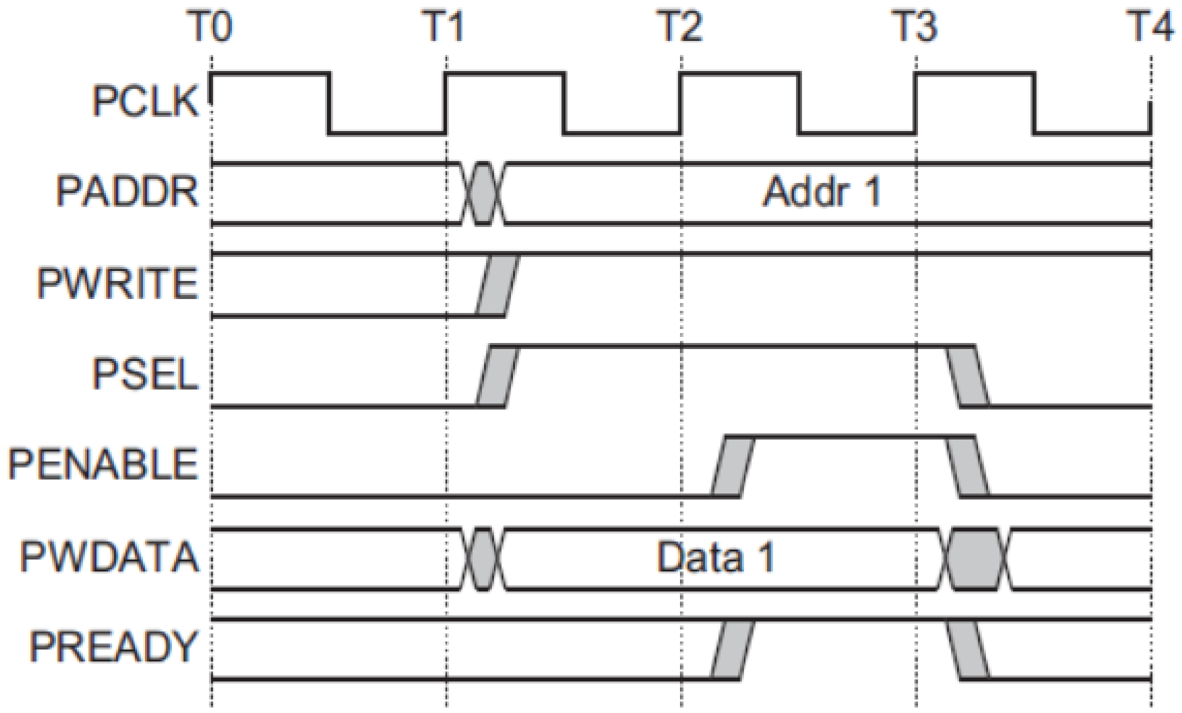


Figure 4.3 Write Cycle [2]

PADDR, PWDATA, PWRITE, and PSEL are registered at the rising edge of PCLK to initiate a write transfer at T1. The SETUP cycle is what it's called. The ACCESS cycle, PENABLE, and PREADY are registered at the next rising edge of the clock T2. When asserted, PENABLE signifies that the transfer's Access phase has begun. PREADY signals that the slave can finish the transfer at the next rising edge of PCLK when asserted. The PADDR, PWDATA, and control signals are all valid until T3, the end of the Access phase, when the transfer is completed. At the end of the transfer, the PENABLE is disabled. Unless the transfer is promptly followed by another transfer to the same peripheral, the choose signal PSEL is also disabled [2-3].

4.1.3 Read cycle

The PENABLE, PSEL, PADDR, and PWRITE signals are asserted at the clock edge T1 during reading operation (SETUP cycle). PENABLE, PREADY, and PRDATA are asserted at the clock edge T2 (ACCESS cycle), and PRDATA is also read during this phase. The data must be provided by the slave before the read transmission is completed[2-3].

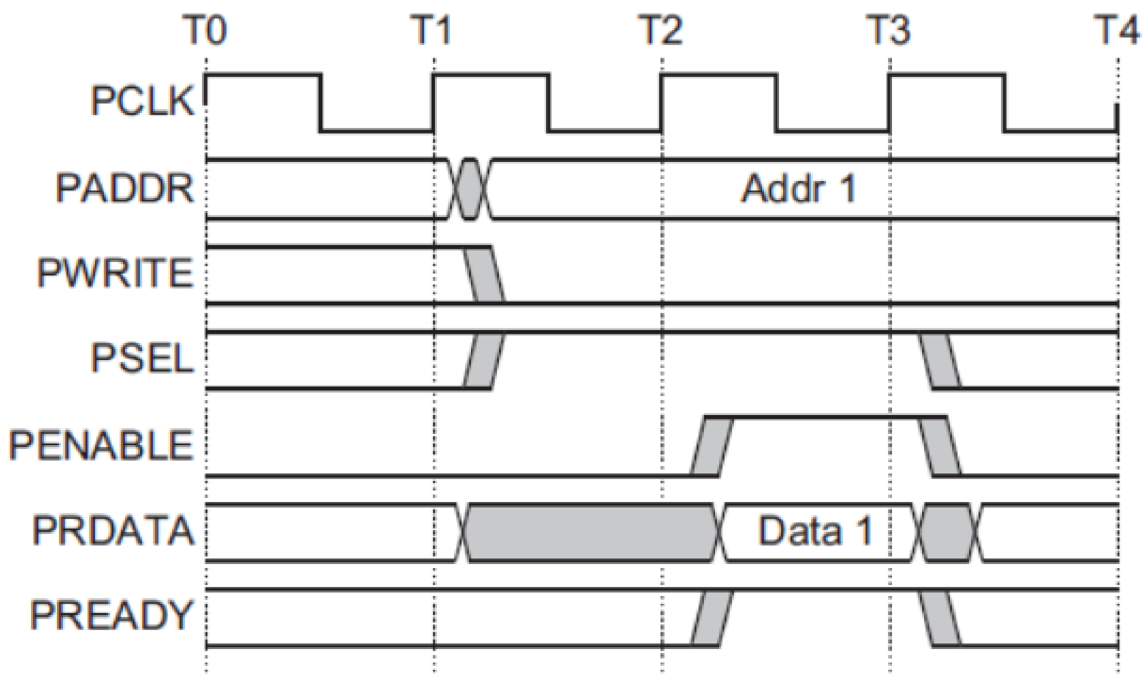


Fig 4.4 Read Cycle [2]

4.1.4 Compose trade

The fundamental create move is showed up in figure 4.5.

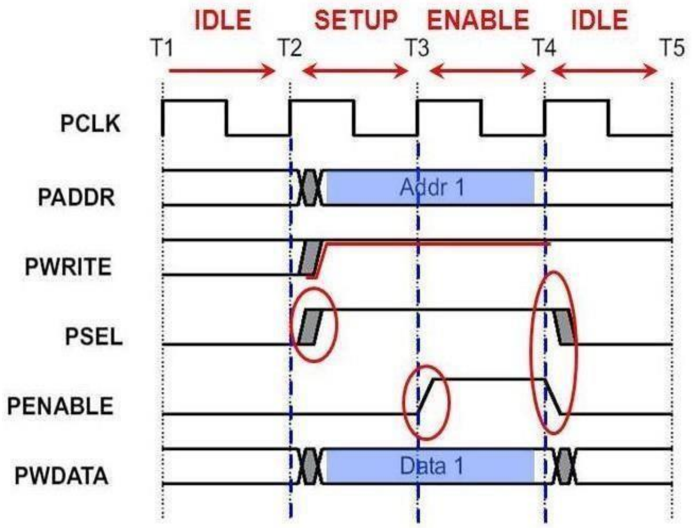


Figure 4.5 Compose trade [2]

The figure starts with the stamp, influence realities, to make wave and pick all flags behind the developing edge of the plot. The very important plot arrangement of the exchange is known as the plan arrangement. Behind running with log arrangement the support flag PENABLE is declared, and this demonstrates the Empower arrangement is going on. The exchange achieved toward the entire of this cycle.

The sum of the deal will be reduced by the PENABLE draw in hail. The select flag will likely go low unless the transaction is promptly supported by another swap to the comparable peripheral. There reduce control, the address flag and the frame hail will not change following an exchange until the minute that the going with get to happens. In hail, the custom just demands a flawless change on the draw. It's possible that the choose and creates signs will be blamed if there are many exchanges.

4.2 AMBA APB SIGNS

The APB motion is depicted with the join as "S" image to detach with different signs appeared in table 4.2.

Table 4.2 APB flag portrayals [2]

Flag	**Portrayal**
SCLK	Log. The developing edges of SCLK all exchanges on the APB.
SRESETn	Cutoff. As far as possible banner is dynamic less. This banner is generally related to the system transport constrain hail.
SADDR	Name. This is a transport with the APB mark. It is handled by the peripheral transport association unit and can be up to 32 bits wide.

SPROT	Protection assembles. This flag indicates if the transaction is an information get to or a direction get to and indicates whether the exchange is normal, exceptional, or secure.
SSELx	Pick. The APB associate part makes the banner to each periphery transport grub. It shows that the grub device has selected a needed data swap.
SENABLE	Empower. This banner shows the following and coming arrangement of an APB trade.
SWRITE	Instruction. When the banner is high, it displays an APB make get to, and when it is low, it shows an APB read get to.
SWDATA	Enlist information. When SWRITE is high, the cars are controlled by the peripheral transport interface unit, which creates an arrangement. The vehicle's width can be up to 32 bits.
SSTRB	Enlist log. The banner depicts which bytes may be upgraded and which can be traded. For every eight bits of create data transfer, there is one form log. As a result, SSTRB[n] is linked to SWDATA [(8n+7) : (8n)].
SREADY	Arranged. Grub uses this banner to extend an APB trade.
SRDATA	Translate data. For this media, they chose grub work and decode cycles while SWRITE is low. The width of this medium can be up to 32 bits.

4.3 APB ASSOCIATE

On the AMBA APB, the APB interface is the fundamental transport master.

4.3.1 Interface layout

On an extraordinarily crucial level, APB depicts the insignificant exertion that is upgraded for less power use and less multifaceted design [1]. This APB custom used to interface the less- information transmission periphery that needn't bother with the AXI tradition.

The APB can communicate with the following systems:

- AMBA Advanced High-performance Bus (AHB)
- AMBA Advanced High-performance Bus Lite (AHB-Lite)
- AMBA Advanced extensible Interface (AXI)
- AMBA Advanced extensible Interface Lite (AXI4-Lite)

Figure 4.6 exhibits the APB hail interfaces of an APB associate.

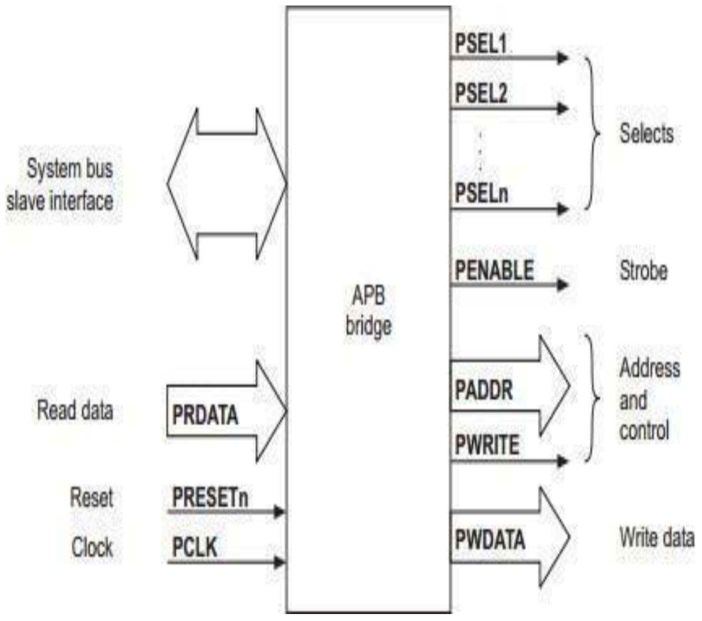


Fig 4.6 APB connect interface [3]

4.3.2 APB associate portrayal

The platform unit basically trades the structure transport into APB and plays with limits:

1. Check it fast and catch it extensively through the exchange.
2. Decrypts the test and sends SSELX, a boundary select. Only a single select standard can be dynamic in a market.
3. Operate the data onto the APB for exchange.
4. Operate the APB data onto the structure transport for a translate exchange.
5. Deliver an orchestrating log, SENABLE, for the exchange.

4.3.3 Schedule graphs

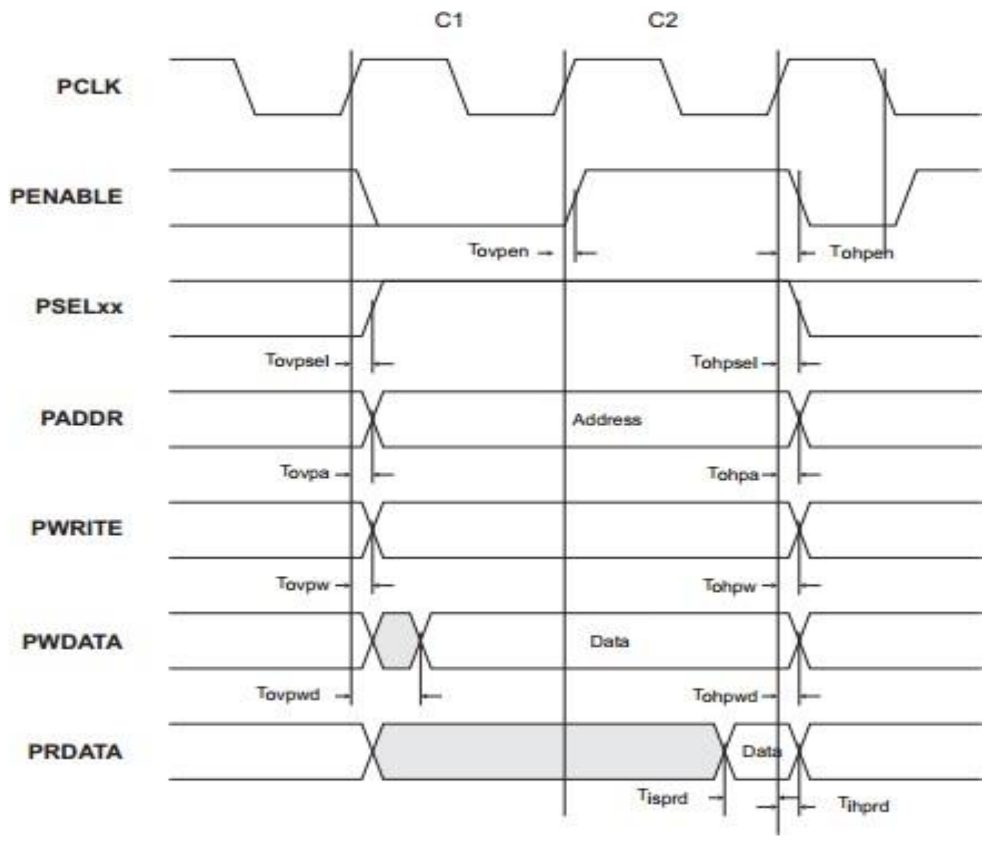


Fig 4.7 APB connect exchange [3]

4.3.4 Planning parameters

Table 4.3 for admission banners and table 4.4 for yield signals lists the arranging criteria for an APB partner.

Table 4.3 APB Expansion admission frameworks [3]

Framework	**Portrayal**
Tclkl	SCLK less minute
Tclkh	SCLK high time
Tisnres	SRESETn de-pronounced format to expanding SCLK
Tihnres	SRESETn de-pronounced catch resulting to developing SCLK
Tihprd	For read exchanges, SRDATA catch in the wake of SCLK

Table 4.4 APB scaffold yield framework [3]

Framework	**Portrayal**
Tovpen	SENABLE substantial in the wake of rising SCLK
Tohpen	SENABLE hold subsequent to rising SCLK
Tovpsel	SSEL substantial in the wake of rising SCLK
Tohpsel	SSEL hold subsequent to rising SCLK
Tovpa	SADDR substantial in the wake of rising SCLK

Tohpa	SADDR hold subsequent to rising SCLK
Tovpw	SWRITE substantial in the wake of rising SCLK
Tohpw	SWRITE hold subsequent to rising SCLK
Tovpwd	For compose exchanges, SWDATA substantial in the wake of rising SCLK
Tohpwd	For compose exchanges, SWDATA hold subsequent to rising SCLK

4.4 APB BOUND

APB bound has a basic, yet adaptable, joining. The correct execution of the joining will be dependent on the plan style utilized and a wide range of choices are possible.

4.4.1 Joining graph

Figure 4.8 demonstrates the flag joining of an APB bound.

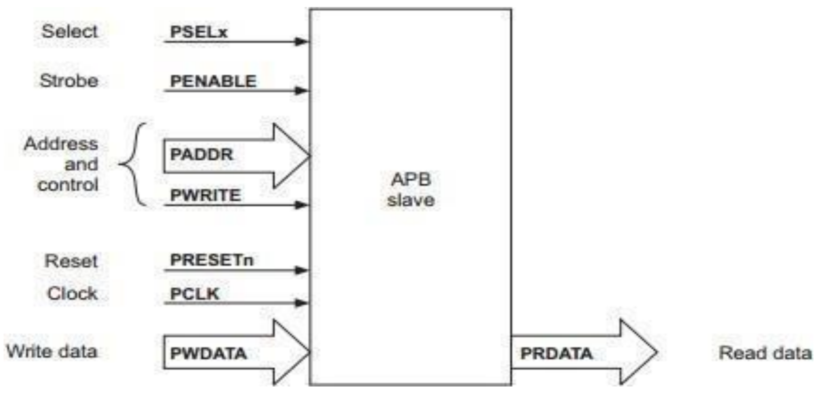


Figure 4.8 APB bound joining depiction[3]

4.4.2 APB bound portrayal

The APB bound joining has a lot of flexibility.

The following focuses can be used to exchange information for a composition:

- When SSEL is high, on each side of SCLK's rising edge
- When SSEL is high, you're on the leading edge of SENABLE.

Combining the choice flag SSEL x, the address flag SADDR, and the compose flag SWRITE yields the compose job. The information can be pushed into the information transport for read exchanges when SWRITE is logic 0 and both SSEL x and SENABLE are logic 1. To decide which enrollment should be read, SADDR is utilized.

4.4.3 Timing outlines

The planning parameters identified with an entrance to an APB transport prisoner are appeared in Fig 4.9.

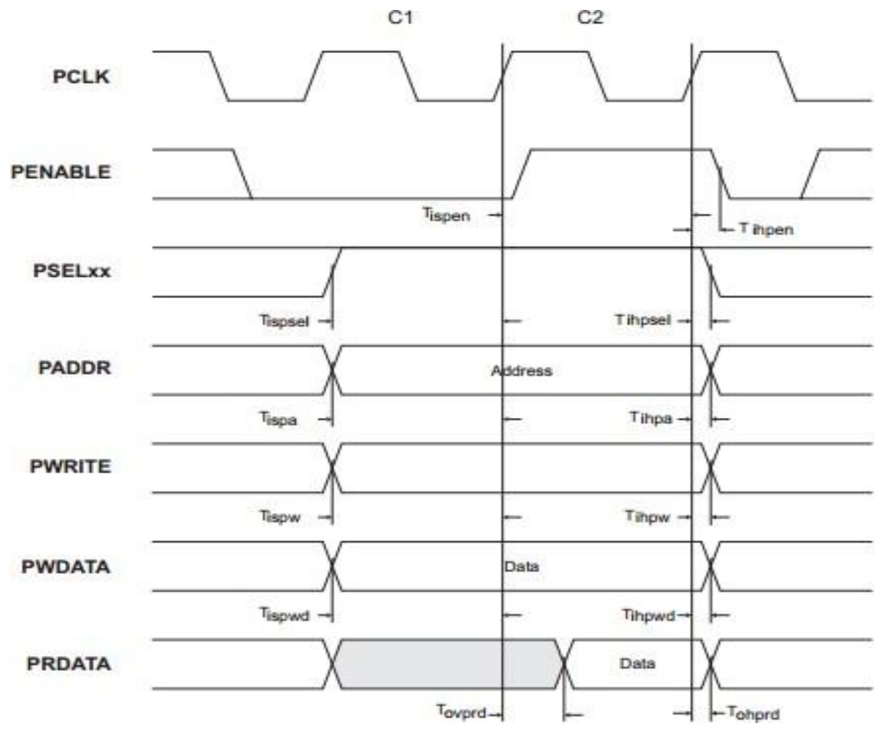


Fig 4.9 APB hostage exchange[3]

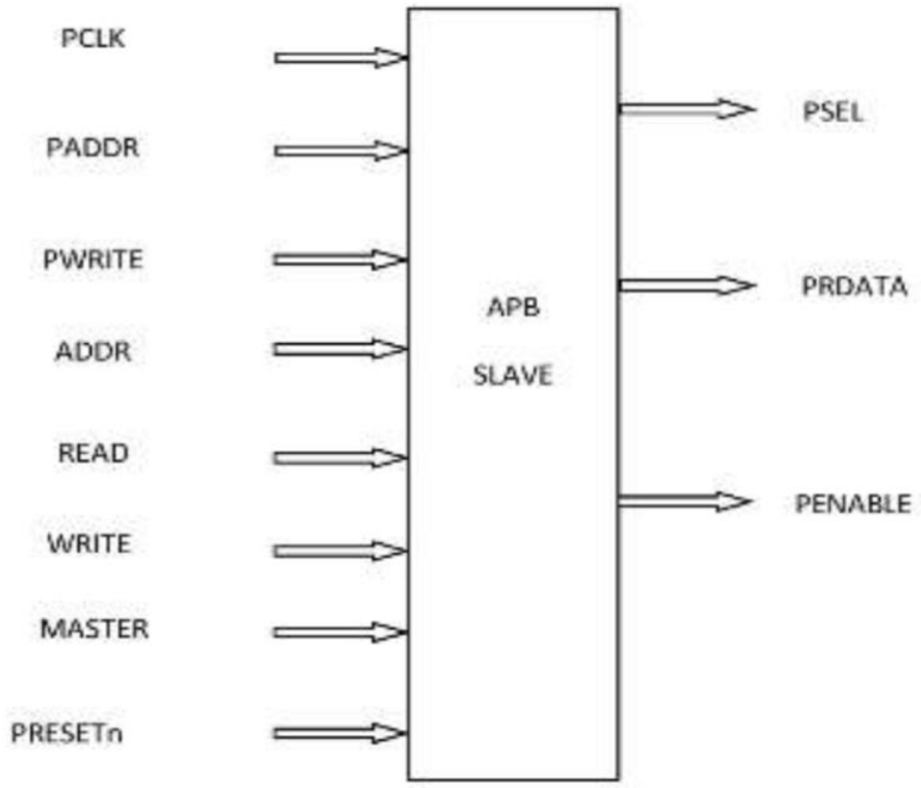


Fig 4.10 Block diagram of APB [3]

The APB Slave input Signals are PADDR, PWRITE, PCLK, PRESETn, PWDATA and PSEL, PRDATA, PENABLE are output signals.

In advance microcontroller bus architecture, APB bridge is very important part. APB Bridge performs different type of operations. The operations are data, address and control signal latching for the peripheral. As like AHB. APB has no peripheral protocol. Therefore, APB interfaces the peripheral that have low bandwidth. This will reduce the low power consumption and also interface complexity.

4.5 Advanced High-performance Bus (AHB)

Component interfaces such as masters, interconnects, and slaves.

The AMBA AHB includes the following features for high-performance, high-clock frequency systems:

- Transfers in bursts.
- Only one clock edge is used.
- Implementation in a non-tristate manner.
- 64, 128, 256, 512, and 1024 bit data bus options.

Internal memory devices, external memory interfaces, and high-bandwidth peripherals are the most typical AHB slaves. Although low-bandwidth peripherals can be included as AHB slaves, they are commonly found on the AMBA Advanced Peripheral Bus for system performance reasons (APB). An AHB slave, also known as an APB bridge, is used to connect the higher-performance AHB and APB. With the AHB master and three AHB slaves, Figure 4.11 depicts a single master AHB system configuration. One address decoder and a slave-to-master multiplexor make up the bus connectivity logic. The master's address is monitored by the decoder, which selects the appropriate slave, and the multiplexor transmits the slave output data back to the master.

AHB also enables multi-master designs via an interconnect component that arbitrates and routes signals from several masters to the appropriate slaves.

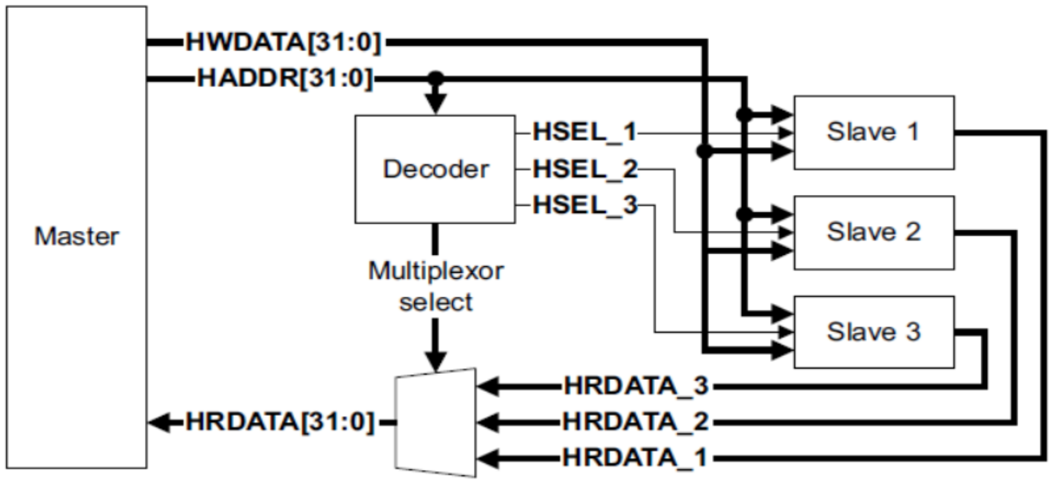


Figure 4.11 AHB block diagram[24]

The following are the primary component kinds of an AHB system :

• Slave
• Master
• Interconnect

4.5.1 Master

A master provides addressing and control the information to start read and write operations.

Figure 4.12 depicts a master interface.

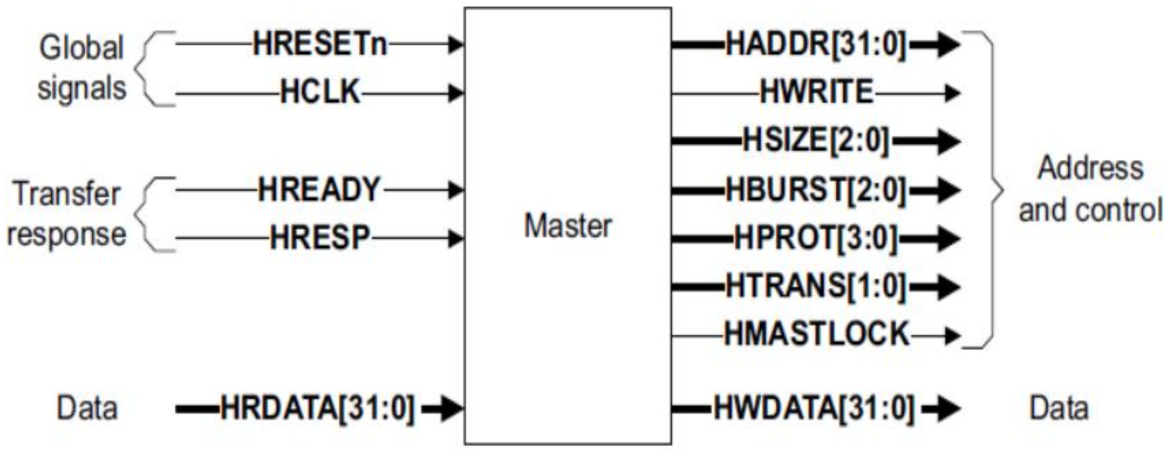


Fig 4.12 Master interface [24]

4.5.2 Slave

In the system, a slave reacts to master-initiated transfers. The slave uses the decoder's HSELx select signal to control when it reacts to a bus transfer.

The slave sends the following message to the master:

• The completion or expansion of the bus transfer.
• The success or failure of the bus transfer.

A slave interface is shown in Figure 4.13.

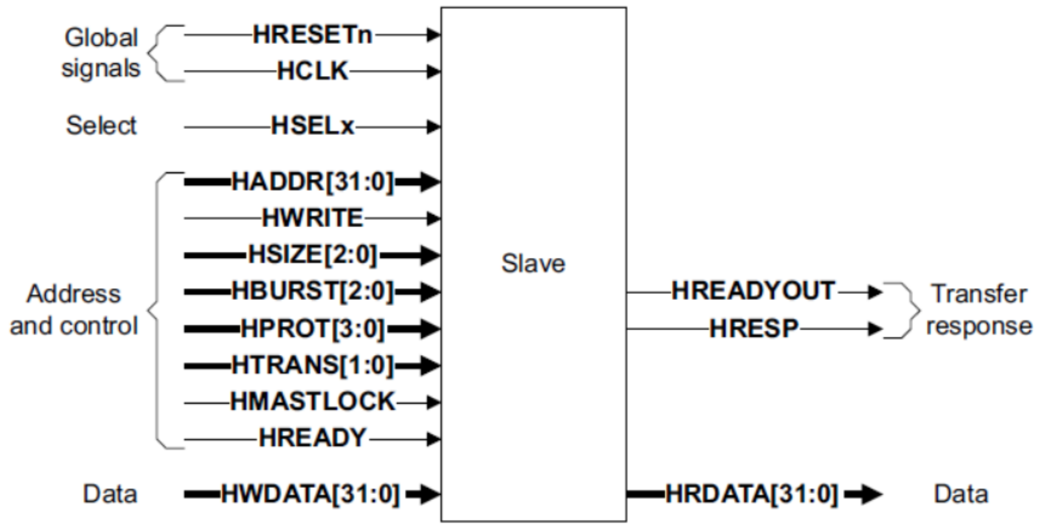


Fig 4.13 Slave interface[24]

4.5.3 Interconnect

The connection is made through an interconnect component. The usage of a Decoder and Multiplexer, as detailed in the following sections, is all that is required for a single master system. In a multi-master system, an interconnect is required to provide signal arbitration and routing from different masters to the appropriate slaves. Address, control, and write data signals all require this route. This specification does not go into detail about the various methodologies used for multi-master systems, such as single layer or multi-layer interconnects.

4.5.3.1 Decoder

This component decodes each transfer's address and sends a choose signal to the slave involved in the transaction. It also gives the multiplexer a control signal. In all configurations with two or more slaves, a single centralized decoder is required.

4.5.3.2 Multiplexer

The read data bus and response signals from the slaves to the master must be multiplexed using a slave-to-master multiplexer. The multiplexer is controlled by the decoder. In all implementations with two or more slaves, a single centralized multiplexer is required.

4.5.4.1 Operation

The address and control signals are driven by the master to begin a transfer. These signals include information about the transfer's location, direction, and width, as well as whether the transfer is part of a burst.

Transfers can take the following forms:

- You are single.
- Bursts that do not wrap at address boundaries are incremented.
- Bursts of wrapping that wrap at certain address limits.

The write data bus transports data from a master to a slave, whereas the read data bus transports data from a slave to a master.

Every transfer is made up of the following components:

- Phase one of the address and control cycle

- Phase of data collection for the data, one or more cycles are required.

Because a slave cannot ask for the address phase to be extended, all slaves must be able to sample the address throughout this time. A slave, on the other hand, can use HREADY to request that the master extend the data phase. When this signal is LOW, wait states are included within the transfer, giving the slave more time to supply or sample data.

HRESP is used by the slave to indicate if a transfer was successful or not.

4.5.5 Signal Descriptions

4.5.5.1 Global signals

NAME	SOURCE	DESCRIPTION
HCLK	Clock source	This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn	Reset controller	The bus reset signal is active LOW and is used to reset

4.5.5.2 Master signals

NAME	Destination	DESCRIPTION
HADDR[31:0]	Slave and decoder	The 32-bit system address bus.
HBURST[2:0]	Slave	Indicates if the transfer forms part of a burst.
HMASTLOCK	Slave	When HIGH, indicates that the current transfer is part of a locked sequence.
HPROT[3:0]	Slave	The protection control signals provide additional information
HPROT[6:4]	Slave	Extended memory types are added to the HPROT signal by a 3-bit extension.
HSIZE[2:0]	Slave	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit).
HTRANS[1:0]	Slave	Indicates the transfer type of the current transfer.
HWDATA[31:0]	Slave	The write data bus is used to transfer data from the bus master to the bus slaves during write operations.
HWRITE	Slave	When HIGH this signal indicates a write transfer and when LOW a read transfer.

4.5.5.3 Slave signals

NAME	Destination	DESCRIPTION
HRDATA[31:0]	Multiplexer	The read data bus is used to transfer data from bus slaves to the bus master during read operations.
HREADYOUT	Multiplexer	When HIGH the HREADY signal indicates that a transfer has finished on the bus.
HRESP	Multiplexer	The transfer response provides additional information on the status of a transfer

4.5.5.4 Decoder signls

NAME	Destination	DESCRIPTION
HSELx	Slave	Each slave has its own slave select signal HSELx, which signals that the present transfer is for the slave now selected.

4.5.5.5 Multiplexer signals

NAME	SOURCE	DESCRIPTION
HRDATA[31:0]	Master	The decoder selects a data bus to read.
HREADY	Master and slave	The HREADY signal conveys to the master and all slaves that the previous transfer is finished when it is HIGH.
HRESP	Master	The decoder chooses the transfer answer.

4.5.6 Transfers

There are two stages to a transfer:

• Unless extended by the preceding bus transfer, the address lasts for a single HCLK cycle.

• Data may necessitate multiple HCLK cycles. Control the number of Clock cycles required to finish the transfer using the HREADY signal.

The direction of data flow to or from the master is controlled by HWRITE. As a result, when :

• When HWRITE is HIGH, it signals that a write transfer is taking place, and the master broadcasts data on the HWDATA [31:0] write data bus.

• When HWRITE is LOW, a read transfer is initiated, and the slave must generate data for the HRDATA [31:0] read databus.

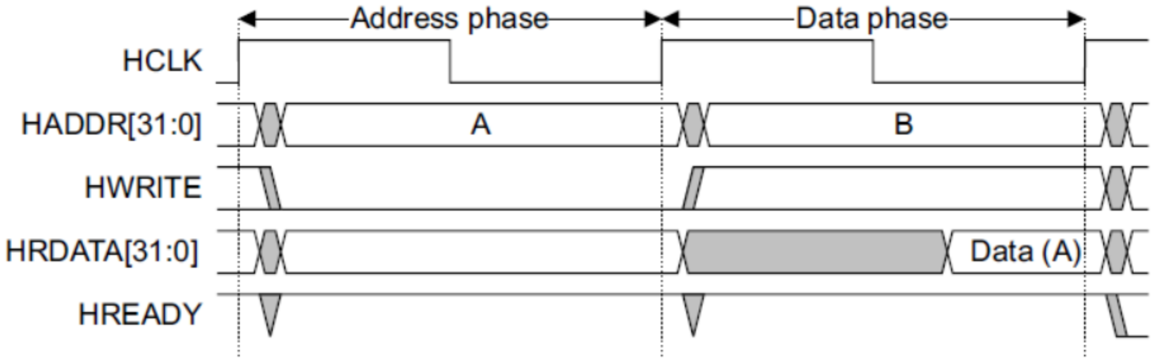


Fig 4.14 Read transfer[24]

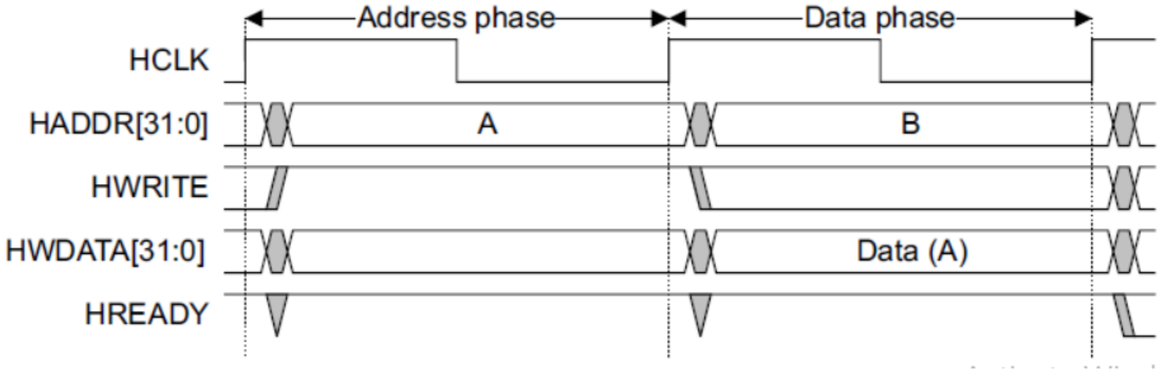


Fig 4.15 Write transfer [24]

In the case of a straightforward transfer with no wait states:

- After the rising edge of HCLK, the master drives the address and control signals onto the bus.
- On the next rising edge of HCLK, the slave samples the address and control information.
- After sampling the address and control, the slave can begin driving the appropriate HREADY response. On the third rising edge of HCLK, the master samples this response.

The address and data phases of the transfer are shown in this basic example throughout different clock cycles. Any transfer's address phase happens during the previous transfer's data phase. The pipelined design of the bus necessitates this overlapping of address and data, which allows for great performance while still allowing sufficient time for a slave to respond to a transfer.

Any transfer can have wait states added to it by a slave in order to give it more time to finish.

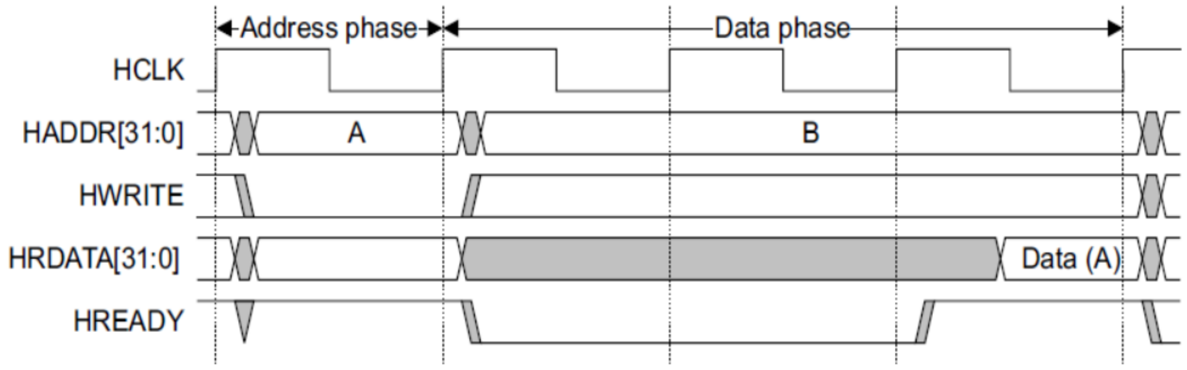


Fig 4.16 Read transfer with wait state [24]

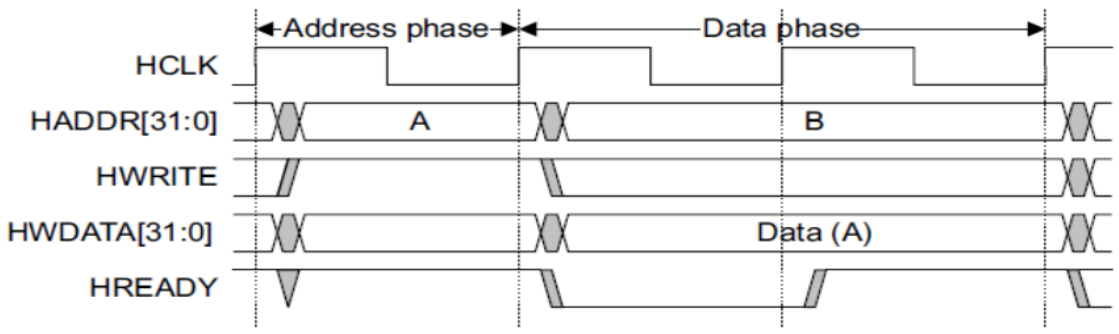


Fig 4.17 Write transfer with wait state [24]

The master maintains data stability over lengthy periods of time for write operations. For read transfers, the slave does not have to provide correct data until the transfer is virtually complete.

There are four different sorts of transfers (HTRANS[1:0]) :

- IDLE (b00)
  - No data transfer is necessary
  - Slave must OKAY without waiting
  - Slave must disregard IDLE
- BUSY (b01)
  - Use a burst of idle cycles
  - After that, the burst will continue
  - In a burst, the address/control reflects the next transfer
  - Slave must accept without waiting
  - Slave must be unconcerned
  - BUSYNONSEQ (b10)
    - A single burst transfer or the initial burst transfer is indicated.
    - Control/address issues unrelated to previous transfers
  - SEQ (b11)
    - Burst's remaining transfers
    - Addr = preceding addr + transfer size

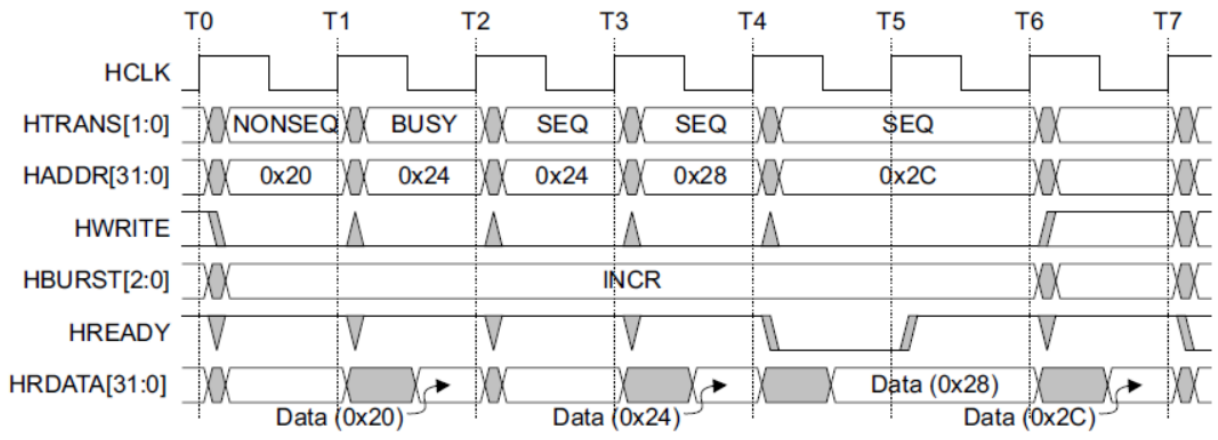


Fig 4.18 Transfer type examples [24]

4.6 Advanced eXtensible Interface (AXI)

For communication between master and slave components, the AMBA AXI protocol offers high-performance, high-frequency system architectures.

The AXI protocol:

• Can be used in high-bandwidth, low-latency designs.

• Operates at a high frequency without the use of complicated bridges.

• Can be used with memory controllers that have a long initial access delay.

• Allows for greater freedom in the design of connection topologies.

• AHB and APB interfaces are backwards compatible.

The key features of the AXI protocol are:

• Address/control and data phases are separated.

• Byte strobes are used to provide unaligned data transfers.

• Transactions are burst-based, with only the start address issued.

• Read and write data channels are separated, allowing for low-cost Direct Memory Access (DMA).

• Allows for numerous outstanding addresses to be issued.

• Out-of-order transaction completion is supported.

• Allows for the quick insertion of register stages for timing closure.

4.6.1 AXI Architecture

The AXI protocol is a burst-based protocol that defines five transaction channels:

• Read the address, which starts with the letter AR.

• Read data with signal names that begin with the letter R.

• Write an address with a signal name that starts with AW.

• Write data with a signal name that starts with W.

• Create a response with a signal name that starts with the letter B.

Control information about the kind of the data to be delivered is carried by an address channel.
The data is exchanged between the master and slave using one of the following methods:

• A write data channel that allows data to be transferred from the master to the slave. The write response channel is used by the slave in a write transaction to signal the master when the transfer is complete.

• Data is transferred from the slave to the master via a read data channel. The AXI protocol entails the following steps:

• Allows for the distribution of address information prior to the actual data transfer.

• Allows for many transactions to be active at the same time.

• Allows transactions to be completed out of order.

Figure 4.19 shows how the write address, write data, and write response channels are used in a write transaction.

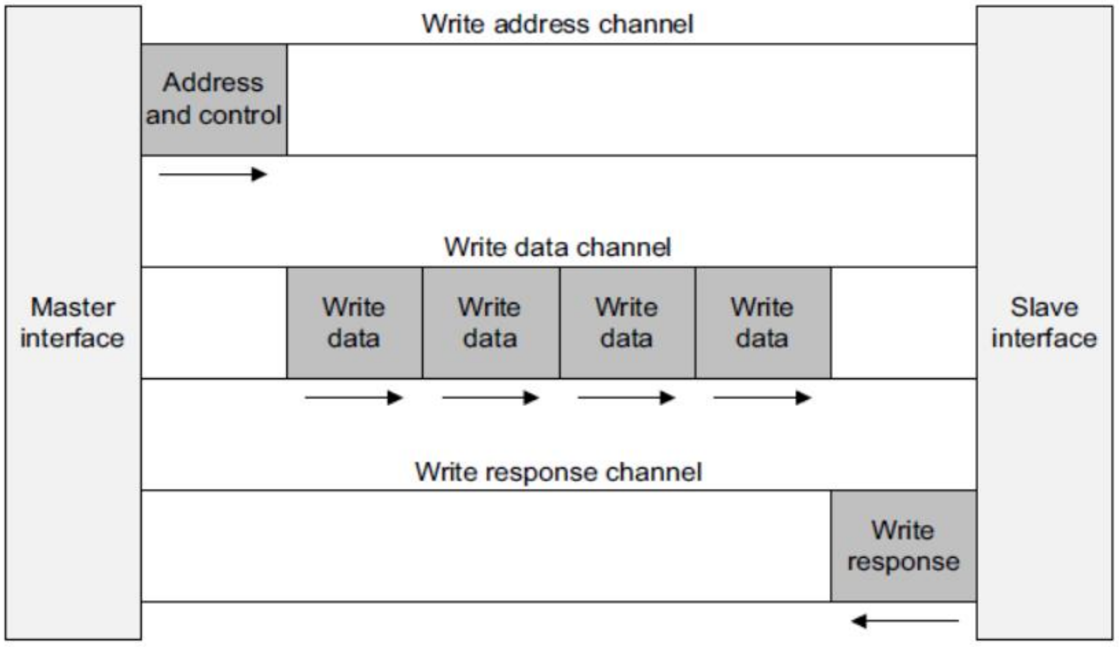


Fig 4.19 Channel architecture of writes[25]

Figure 4.20 shows how the read address and read data channels are used in a read transaction.

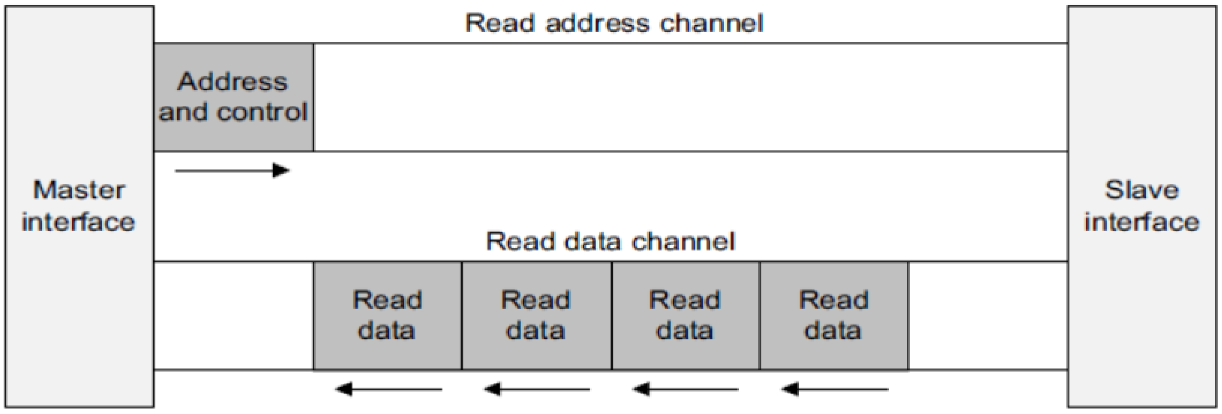


Fig 4.20 Channel architecture of reads [25]

4.6.2 Channel definition

Each of the five separate channels contains a set of information signals as well as the VALID and READY signals that allow for a two-way handshake. The VALID signal is used by the information source to indicate when there is valid address, data, or control information on the channel. The READY signal is used by the destination to indicate when it is ready to take the data. A LAST signal is included in both the read and write data channels to signify the transfer of the last data item in a transaction.

4.6.2.1 Read and write address channels

Each read and write operation has its own address channel. All of the essential address and control information for a transaction is carried on the appropriate address channel.

4.6.2.2 Read data channel

The read data channel transports both read data and read response information from the slave to the master and contains the following information:

• A data bus with a width of 8, 16, 32, 64, 128, 256, 512, or 1024 bits.
• A read response signal showing the read transaction's completion status.

4.6.2.3 Write data channel

The write data channel transports write data from the master to the slave and contains the following information:

- A data bus with a width of 8, 16, 32, 64, 128, 256, 512, or 1024 bits.
- Every eight data bits, a byte lane strobe signal identifying the valid bytes of the data.

The write data channel information is always handled as buffered, so the master can perform write transactions without having to wait for earlier write transactions to be acknowledged by the slave.

4.6.2.4 Write response channel

The write response channel is used by a slave to respond to write transactions. On the write response channel, all write transactions require completion signalling.

4.6.3 Interface and interconnect

A typical system consists of multiple master and slave devices that are interconnected in some way.

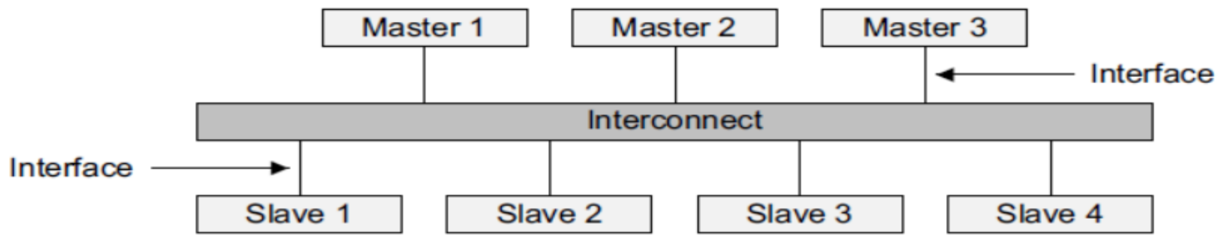


Fig 4.21 Interface and interconnect [25]

The AXI protocol establishes a single interface specification for the following interfaces:

- There's a master and there's an interconnect.
- An interconnect and a slave.
- A slave and a master.

Many distinct connectivity implementations are supported by this interface definition. A device interconnect is the same as another device having symmetrical master and slave ports to which genuine master and slave devices can be connected.

4.6.4 Typical system topologies

One of three interconnect topologies is used by the majority of systems:

- Address and data buses that are shared.
- Multiple data buses and shared address buses.
- Multiple address and data buses on a multilayer system.

The address channel bandwidth need is often much lower than the data channel bandwidth requirement in most systems. By combining a shared address bus with numerous data buses to enable parallel data transfers, such systems can strike a fair compromise between system performance and interconnect complexity.

4.6.5 Register slices

Each AXI channel just transmits data in one way, and the architecture does not require any set channel relationships. Because of these characteristics, a register slice can be introduced at practically any point in any channel, albeit at the penalty of an extra cycle of latency.

These characteristics allow you to do the following:

- A trade-off between delay cycles and maximum operating frequency.

- A direct, quick connection between a CPU and high-performance memory, with simple register slices used to separate a longer path to peripherals with lower performance requirements.

4.6.6 Signal Descriptions

4.6.6.1 Global signals

Signal	**Source**
ACLK	Clock source
ARESETn	Reset source

4.6.6.2 Address channel signals

Write address channel (AW)	Read address channel (AR)	Signal Description
AWID	ARID	Identification tags
AWADDR	ARADDR	Addresses
AWLEN	ARLEN	This information determines the number of data transfers associated with the address.
AWSIZE	ARSIZE	The number of bytes in each data transfer in a write transaction.
AWBURST	ARBURST	Burst type
AWVALID	ARVALID	Valid
AWREADY	ARREADY	Ready

4.6.6.3 Data channel signals

Write Data channel (W)	Read Data channel(R)	Signal Description
WID	RID	Identification tags
WDATA	RDATA	Read/Write Data
WSTRB	---	Write strobes, indicate which byte lanes hold valid data.
WLAST	RLAST	Indicates Last data transfer in the transaction.
---	RRESP	Read response, indicates the status of a read transfer.
WVALID	RVALID	Valid
WREADY	RREADY	Ready

4.6.6.4 Write response channel signals

Write response channel (B)	**Signal Description**
BID	Identification tag
BRESP	The state of a write transaction is indicated by the write response.
BVALID	Valid
BREADY	Ready

CHAPTER 5

SIMULATION RESULTS AND ANALYSIS

In this chapter , AMBA protocol's designs are implemented using Verilog in Xilinx VIVADO 2019.2 by selecting Zynq-7000 xc7z010sclg400-1 FPGA device. The AMBA Protocols: APB, AHB, AXI, AXI4 have been designed.

The Simulation waveforms and synthesized designs are given in the following subsections.

5.1 APB

RTL schematic and synthesized design of APB protocol is shown in Figs 5.1 and 5.2 respectively. In this design 9870 LUTs and 32802 Registers are used. The maximum delay, minimum delay, dynamic power are found to be 4.676 ns, 0.392 ns and 28.746 W respectively.

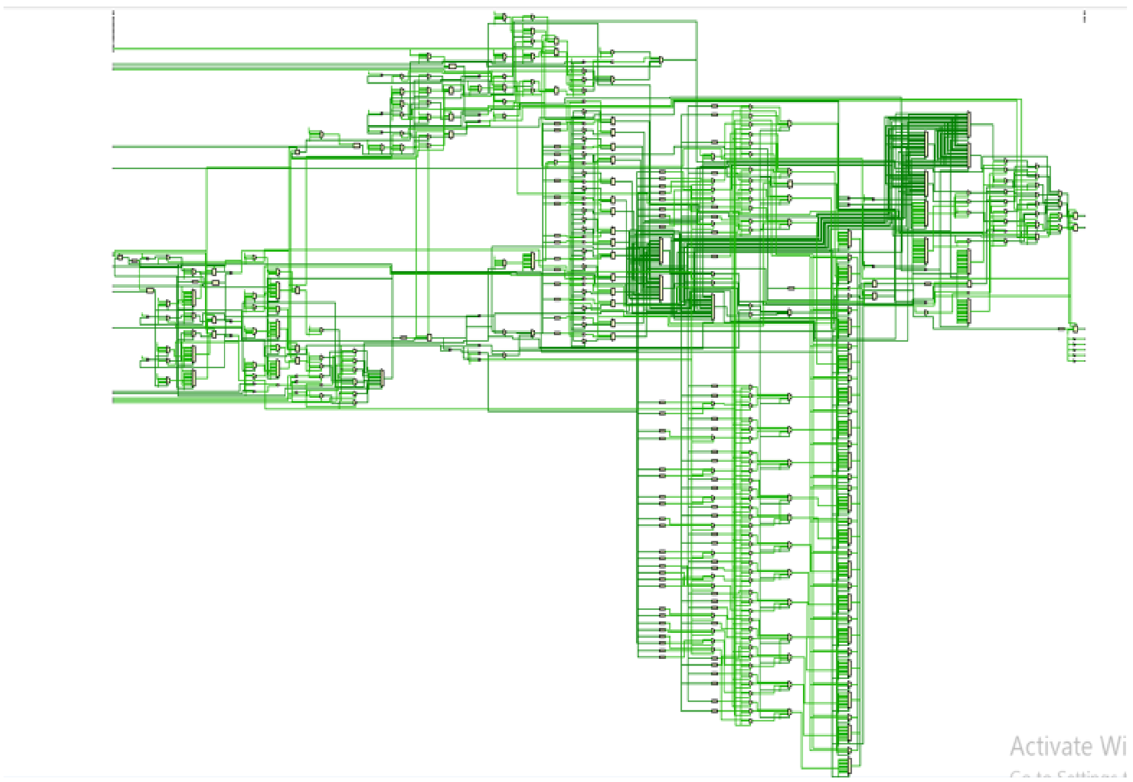


Fig 5.1 RTL Schematic of APB Protocol

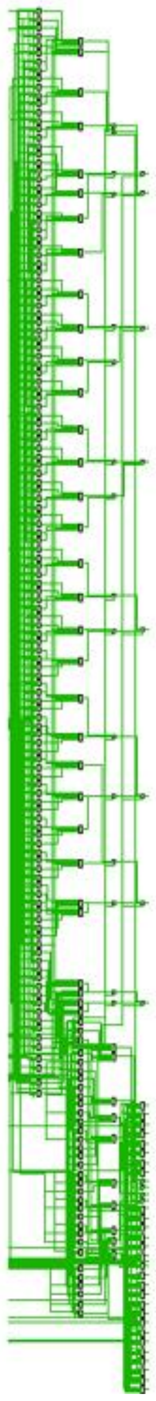
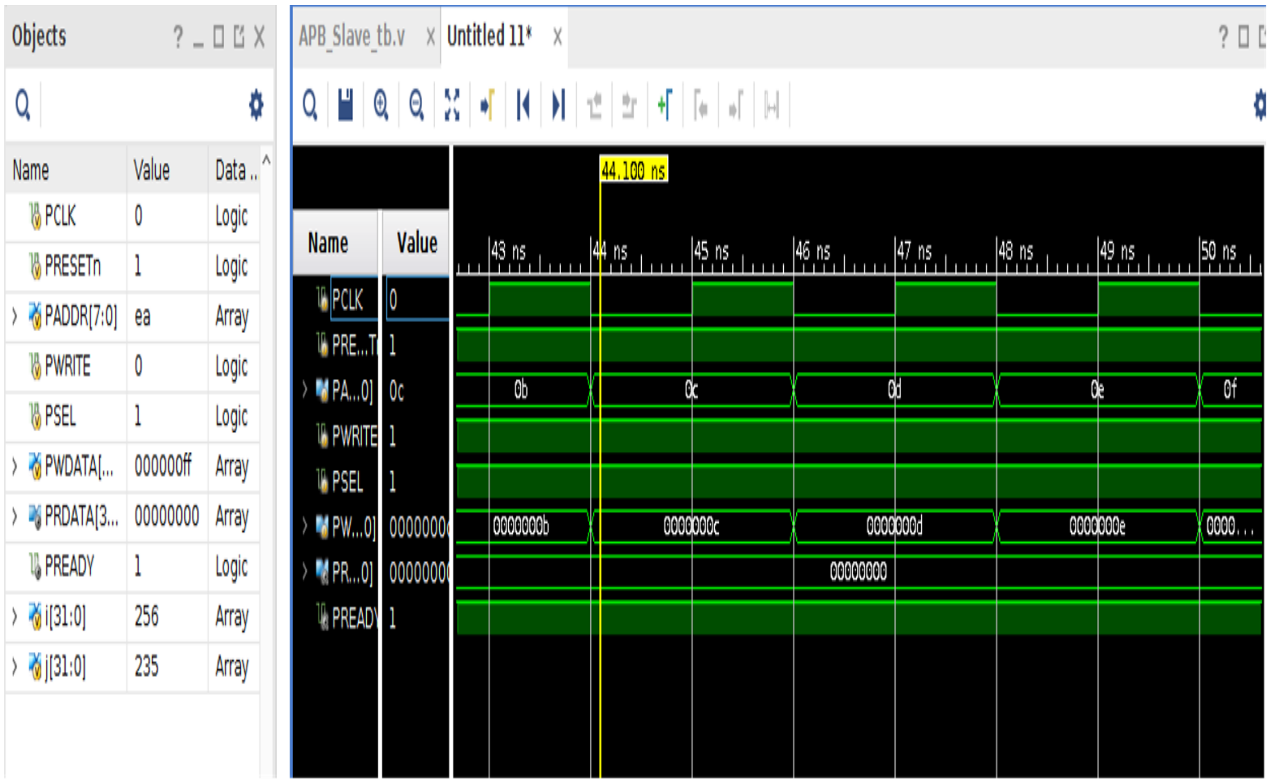


Fig 5.2   Synthesized Design of APB Protocol

Fig 5.3 shows APB Write cycle. The PADDR, PWDATA, PWRITE, and PSEL are registered at the rising edge of PCLK to initiate a write transfer . The SETUP cycle is what it's called. The ACCESS cycle, PENABLE, and PREADY are registered at the next rising edge of the clock . When asserted, PENABLE signifies that the transfer's Access phase has begun. PREADY signals that the slave can finish the transfer at the next rising edge of PCLK when asserted. The PADDR, PWDATA, and control signals are all valid until next cycle , the end of the Access phase, when the transfer is completed. At the end of the transfer, the PENABLE is disabled. Unless the transfer is promptly followed by another transfer to the same peripheral, the choose signal PSEL is also disabled.



Name	Value	Data ..
PCLK	0	Logic
PRESETn	1	Logic
PADDR[7:0]	ea	Array
PWRITE	0	Logic
PSEL	1	Logic
PWDATA[...	000000ff	Array
PRDATA[3...	00000000	Array
PREADY	1	Logic
i[31:0]	256	Array
j[31:0]	235	Array

Name	Value
PCLK	0
PRE...T	1
PA...0]	0c
PWRITE	1
PSEL	1
PW...0]	0000000
PR...0]	0000000
PREADY	1

Fig 5.3 Simulation results of write cycle for APB Protocol

Fig 5.4 shows APB Read cycle. The PENABLE, PSEL, PADDR, and PWRITE signals are asserted at the clock edge during reading operation (SETUP cycle). PENABLE, PREADY, and PRDATA are asserted at the clock edge (ACCESS cycle), and PRDATA is also read during this phase. The data must be provided by the slave before the read transmission is completed.

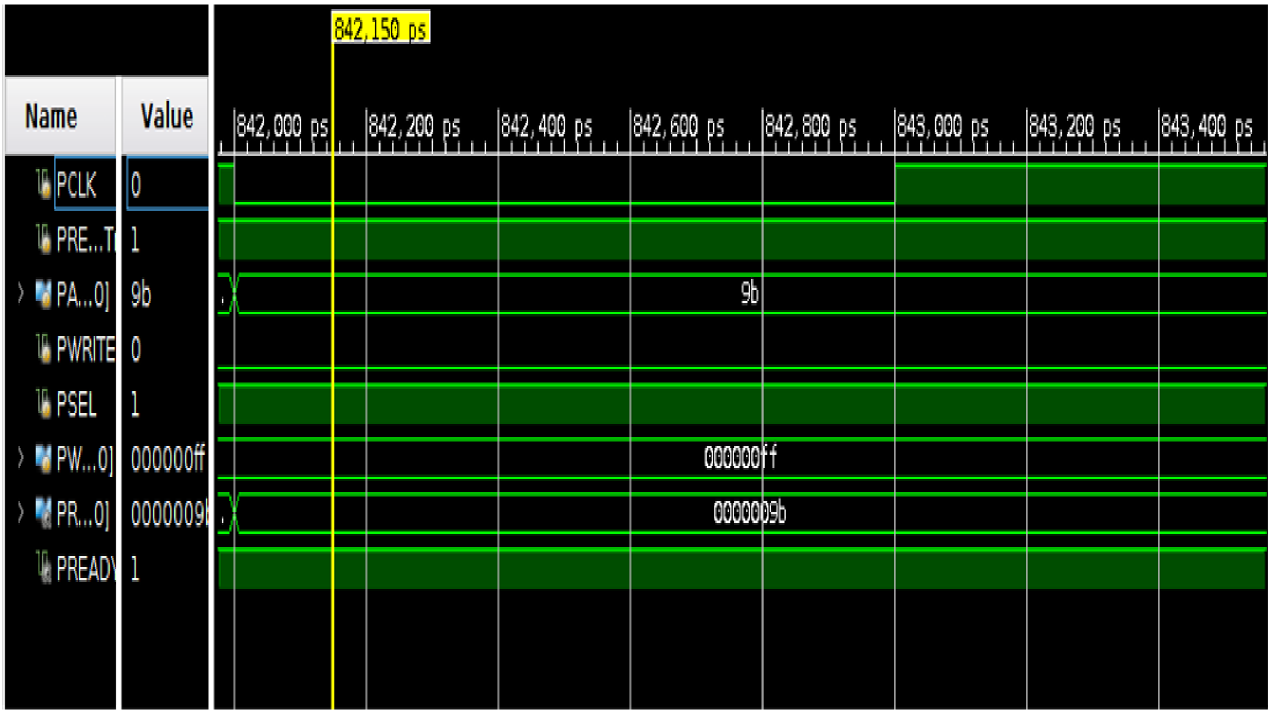


Fig 5.4 Simulation result of Read Cycle for APB Protocol

5.2 AHB

RTL schematic and synthesized design of AHB protocol is shown in Figs 5.5 and 5.6 respectively. In this design 1406 LUTs and 4355 Registers are used. The maximum delay, minimum delay, dynamic power are found to be 4.076 ns, 0.305 ns and 9.319 W respectively.

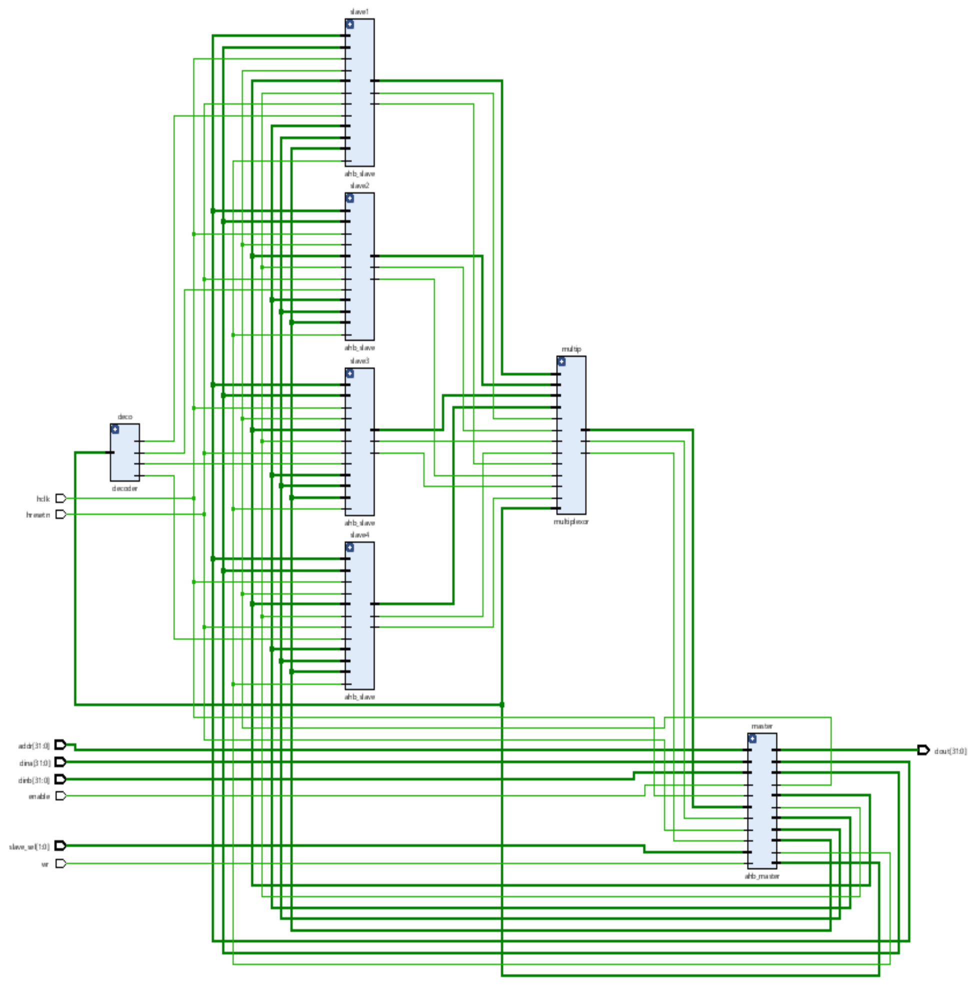


Fig 5.5 RTL Schematic of AHB Protocol

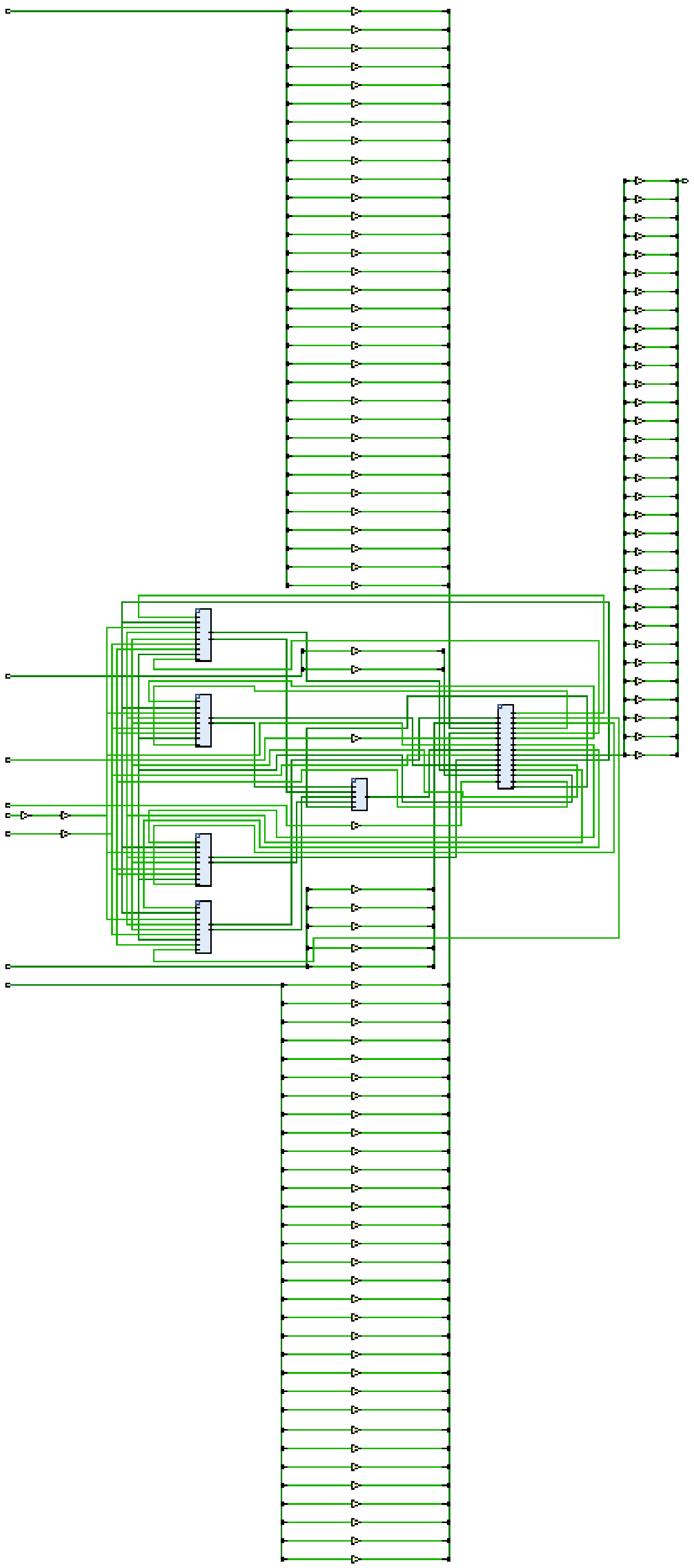


Fig 5.6 Synthesized Design of AHB Protocol

Figure 5.7 depicts a simulation of an AHB system with one AHB master and four AHB slaves. The bus connection logic consists of one address decoder and one slave-to-master multiplexer. The decoder monitors the master's address and picks the suitable slave, while the multiplexor sends the slave output data back to the master. Multi-master designs are also possible with AHB thanks to an interconnect component that arbitrates and directs signals from many masters to the appropriate slaves.

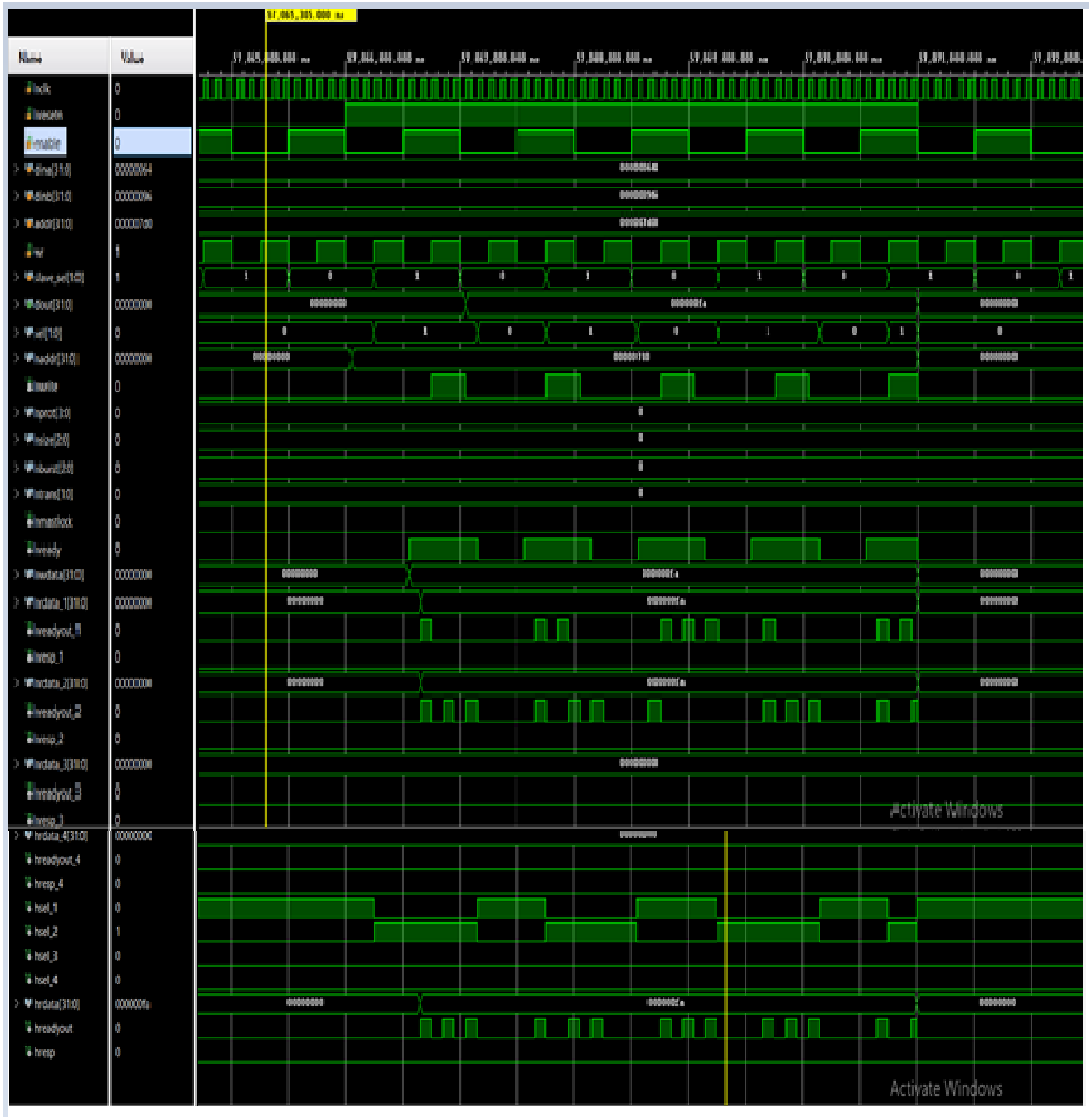


Fig 5.7 Simulation results of AHB Protocol

5.3 AXI Slave

RTL schematic and synthesized design of AXI protocol is shown in Figs 5.8 and 5.9 respectively. In this design 1328 LUTs and 649 Registers are used. The maximum delay, minimum delay, dynamic power are found to be 5.541 ns, 0.395 ns and 12.036 W respectively.

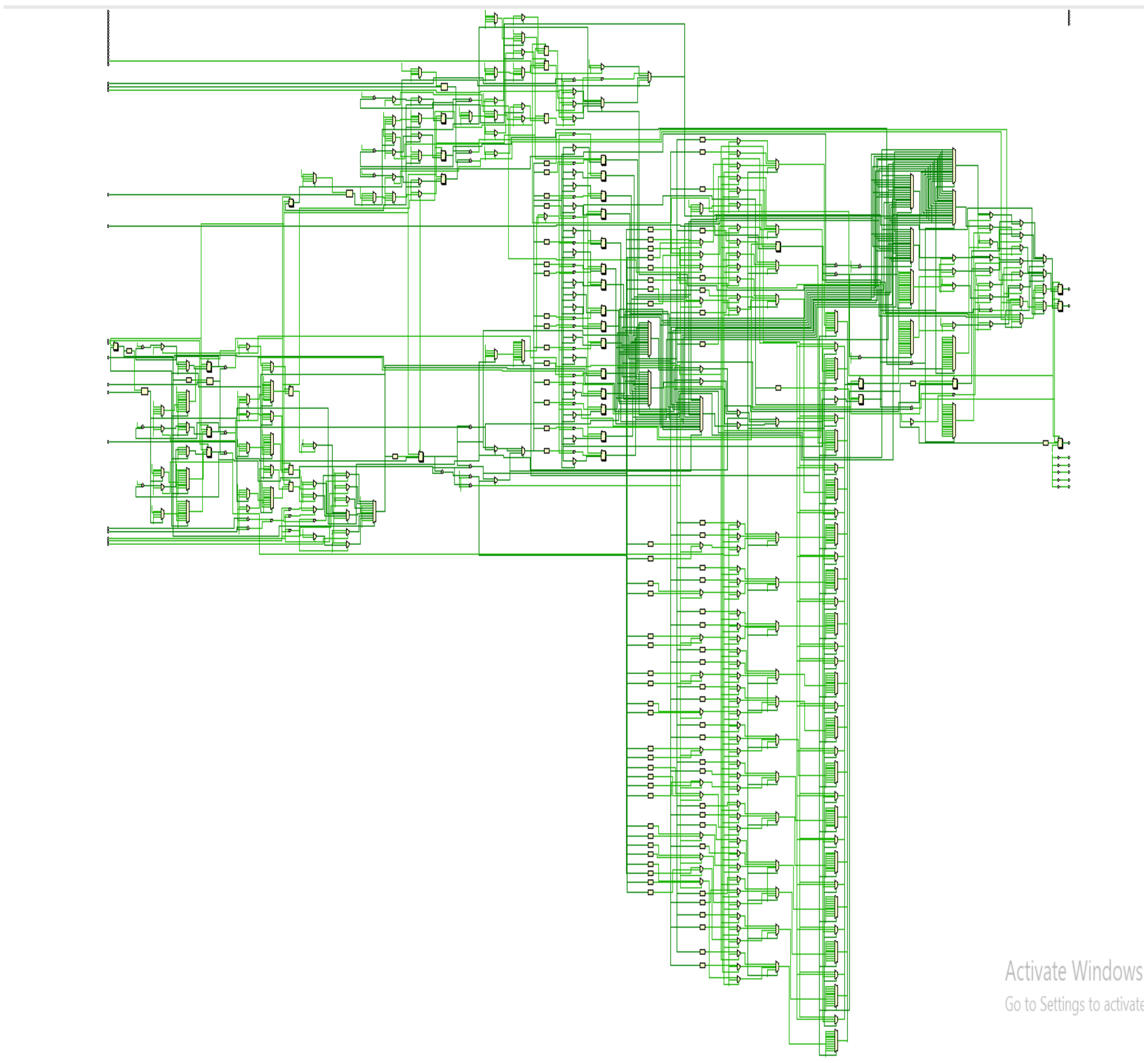


Fig 5.8   RTL Schematic of AXI Slave

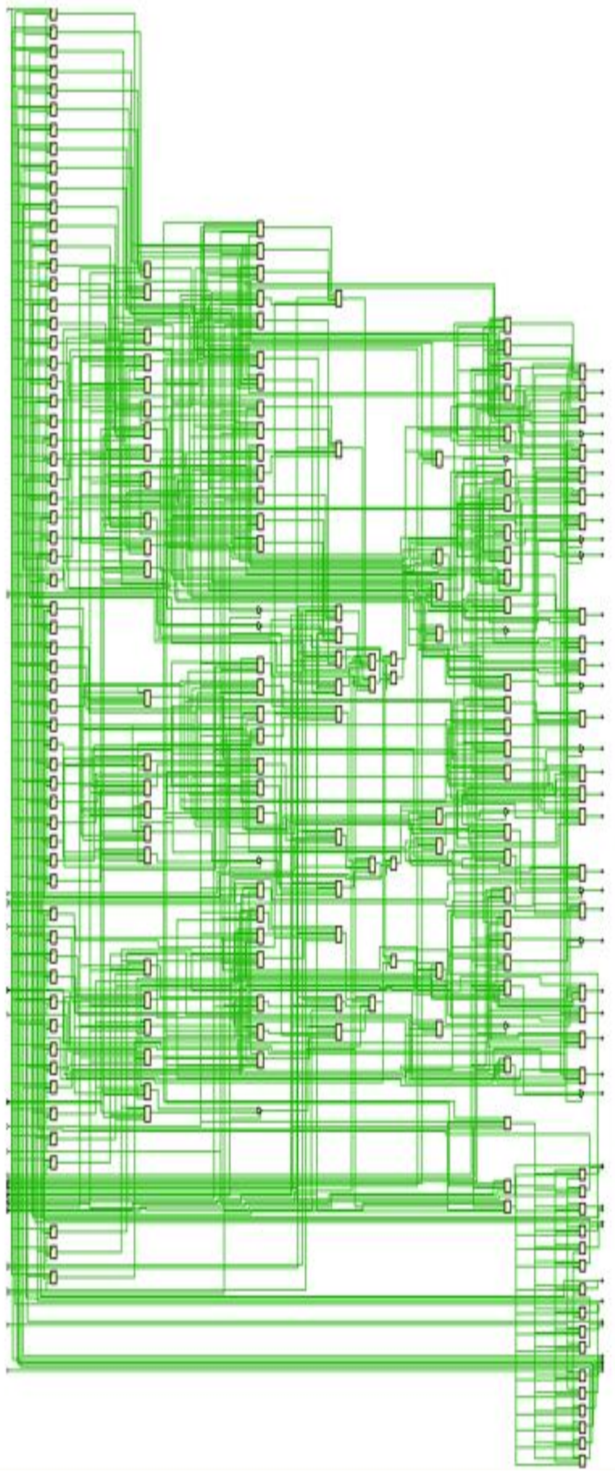


Fig 5.9 Synthesized Design of AXI Slave

5.4 AXI 4 LITE SLAVE

RTL schematic and synthesized design of AXI 4 Lite Slave protocol is shown in Figs 5.10 and 5.11 respectively. In this design 222 LUTs and 341 Registers are used. The maximum delay, minimum delay, dynamic power are found to be 4.076 ns, 0.385 ns and 5.456 W respectively.

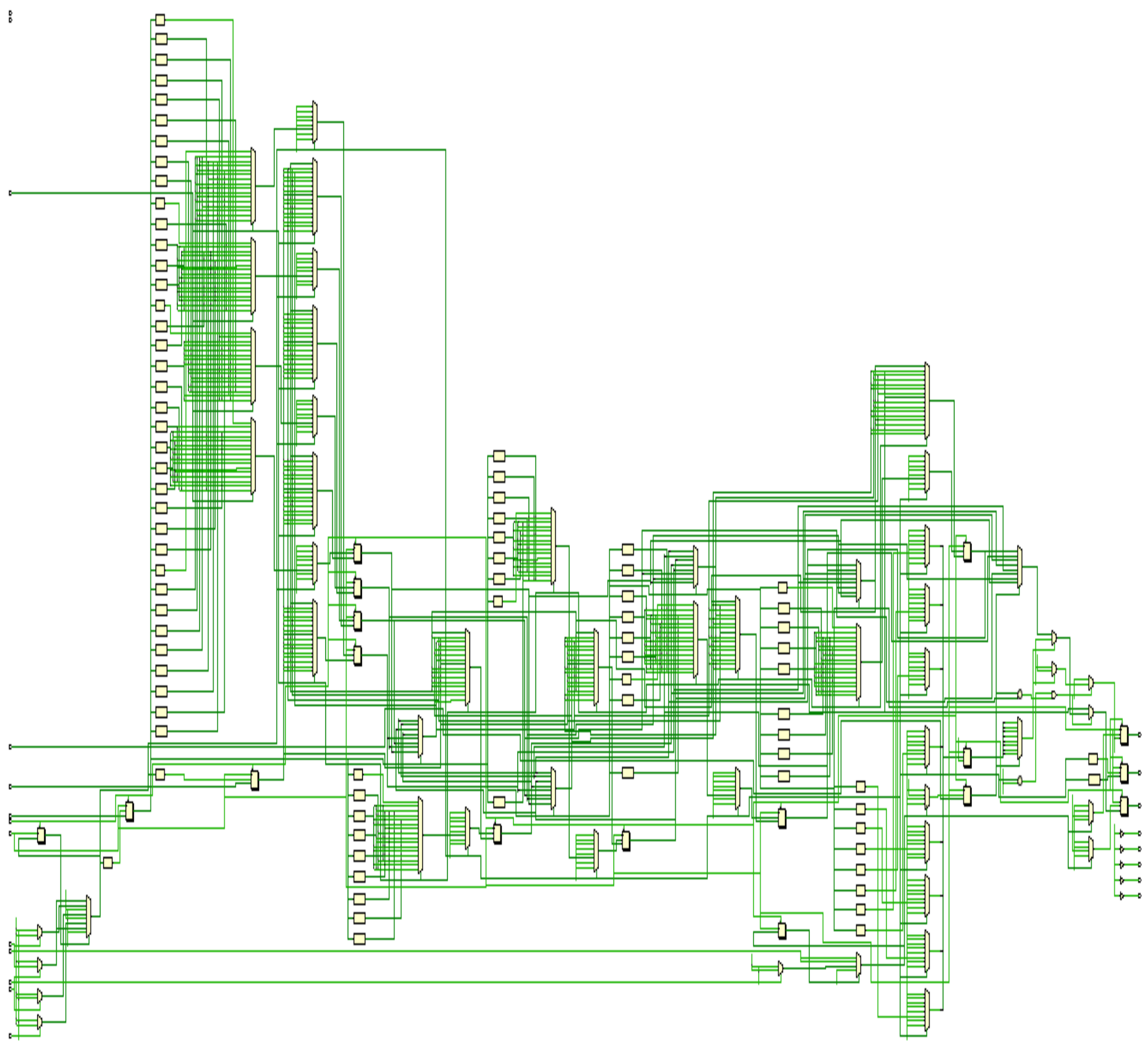


Fig 5.10   RTL Schematic of AXI 4 Lite Slave

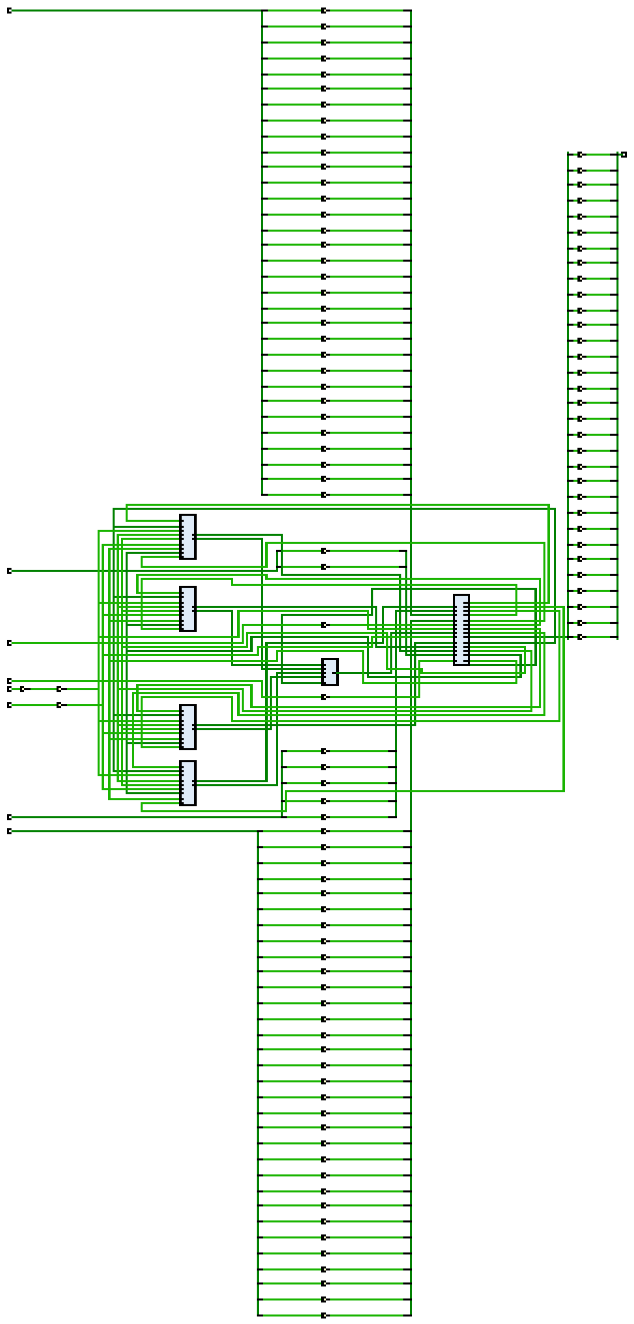


Fig 5.11 Synthesized Design of AXI 4 Lite Slave

The AXI4-Lite Slave is a basic Memory with 16 register locations of 32-bit data content, as shown in Fig. 5.12. Reading operations are performed from customized Memory to Master, which reads the data in memory.

The following are the key tasks or functions that the slave and Master perform:

- During the reading process, the Master communicates a specific address from which it wishes to read data from the memory file/peripheral through the ARADDR signal on the read address channel, as well as validating the address.

- If there is a valid address, ARVALID will be equal to logic one .Until the RREADY signal is logic one, the Master's address remains steady.

- Memory acknowledges Master by declaring ARREADY and placing the data on a data bus, signaling that it accepts the address.

When valid data is present on the data bus, the RVALID Signal becomes high, signaling that the master accepts the data and reads the data.

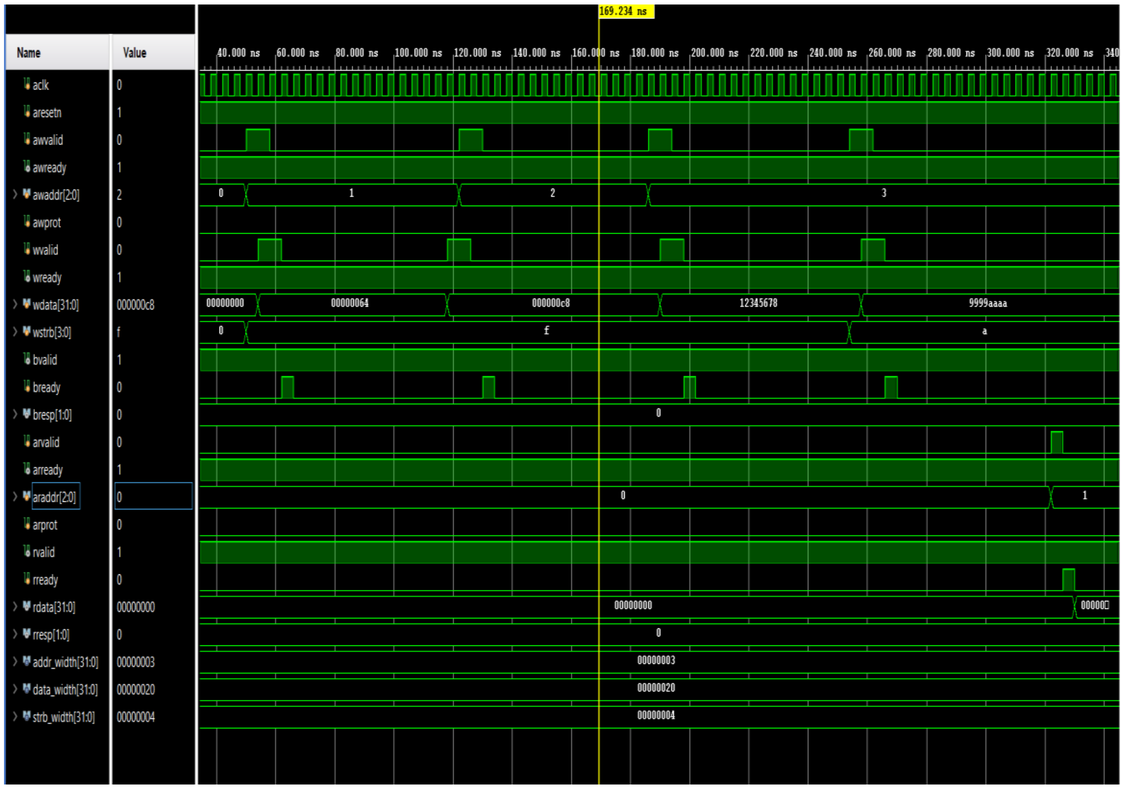


Fig 5.12 Simulation Result of AXI 4 Lite Slave

5.5 Comparison

All designs are functionally verified using simulation on Vivado 2019.2 and selecting Zynq-7000 xc7z010sclg400-1 FPGA. The designs are compared in terms of area (Look Up Tables i.e LUTs), maximum delay, minimum delay and dynamic power. The findings are summarized in Table 5.1.

Table 5.1 Summary of AMBA Protocols Implementations

AMBA Protocols	**Slices**		**Power (in Watt)**		**Delay (in ns)**	
	LUTs	**Registers**	**Dynamic Power**	**Static power**	**Maximum delay (Setup)**	**Minimum delay (Hold)**
APB	9870	32802	28.746	0.747	4.676	0.392
AHB	1406	4355	9.319	0.747	4.076	0.305
AXI	1328	649	12.036	0.747	5.541	0.395
AXI4	222	341	5.456	0.291	4.076	0.385

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

The AMBA Bus protocols i.e APB, AHB, AXI, AXI4 have been implemented in VIVADO 2019.2 by selecting Zynq-7000 xc7z010sclg400-1 FPGA, written in Verilog HDL language. They have further been analysed in terms of hardware utilization (number of slices, which is comprised of Look Up Tables or LUT and Registers), timing (delay in ns) and power consumption (in watt). We now see from the above timing diagrams that our RTL design based on the state diagram is indeed correct as verified by the timing diagrams of read and write cycle.

The integration trend continues in the current era of heterogeneous computing for HPC and data centre sectors, with an increasing number of processor cores and many heterogeneous computing parts such as GPU, DSP, FPGAs, memory controllers, and IO sub systems. AMBA4 AXI4: Extending System-Wide Coherency with AMBA 4 AXI Coherency Extensions is an introduction to AMBA4 AXI4 (ACE). As a re-design of the AXI/ACE protocol, the CHI (Coherent Hub Interconnect) protocol was created. The AXI/ACE signal-based protocol was replaced with the new packet-based CHI layered protocol, which can scale very well in the near future.

REFERENCES

[1] “AMBA Specification (Rev 2.0)”, available at http://www.arm.com.

[2] ARM. “AMBA Open Specifications” http://www.arm.com/products/system- ip/amba/ambaopen specifications.php.

[3] http://en.wikepedia.org/wiki/ Advanced_MicrocontrollerBus_Architecture.

[4] Kiran Rawat *et al*. (2015). “RTL Implementation for AMBA ASB APB Protocol at System on Chip Level” *2nd International Conference on Signal Processing and Integrated Networks (SPIN,)* pp. 927-930.

[5] Yasemin M. Akay *et al*. (2009). “Hippocampal gamma Oscillations in Rats” *IEEE Engineering in Medicine and Biology Magazine*, vol. 28, pp. 92-95.

[6] Kiran Rawat *et al*. (2014). “Design of AMBA APB Bridge with Reset Controller for Efficient Power Consumption” *9th International Conference on Industrial and Information Systems (ICIIS),* pp.1-5.

[7] Kiran Rawat *et al.* (2015). *“*Implementation of AMBA APB Bridge with Efficient Deployment of System Resources” *International Conference on Computer, Communication and Control (IC4),* pp.14.

[8] Jasmine Chhikara et al. (2015). “Implementing Communication Bridge between 12C and APB” *IEEE International Conference on Computational Intelligence & Communication Technology*, pp. 235-238.

[9] Ashutosh Gupta *et al.* (2016). “Physical Design Implementation of 32-bit AMBA ASB APB module with improved performance” *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT),* pp. 3121-3124.

[10] Kanishka Lahiri and Anand Raghunanthan (2004). “Power Analysis of System-Level On-Chip Communication Architectures” *International Conference on Hardware/Software Codesign and System Synthesis*, pp. 236-241.

[11] Raed M. Salih and Laszek T. Lilien (2015). “Protecting Users’ Privacy in Healthcare Cloud Computing with APB-TTP” *IEEE International Conference on Pervasive Computing and Communication Workshops (Per Com Workshops)*, pp. 236-238.

[12] Roopa M. *et al.* (2013) “UART Controller as AMBA APB Slave” *National Conference on Challenges in Research & Technology in the Coming Decades*, pp. 1-6.

[13] Chenghai Ma *et al.* (2011) “Design and Implementation of APB Bridge based on AMBA 4.0” *International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 193-196.

[14] Ge Zhiwei *et al.* (2009). “Design of On-chip Image Processing Based on APB Bus with CMOS Image Sensor” *IEEE 8th International Conference on ASIC*, pp. 963-966.

[15] Miss. Dhage Naiyna Kashinath and Prof. S.I. Nipanikar (2015). “AMBA Bus with Multiple Masters Using VLSI” *IJEDR*, vol. 3, pp. 97-102.

[16] 17 P. P. Sotiriadis and A. P. Chandrakasan (2002). “A Bus Energy Model for Deep Submicron Technology,” *IEEE Trans. VLSI Systems*, vol. 10, pp. 341–350.

[17] M. R. Stan and W. P. Burleson (1995). “Bus Invert Coding for Low Power I/O,” *IEEE Trans. VLSI Systems*, vol. 3, pp. 49–58.

[18] L. Benini, A. Macii *et al.* (2000). “Architectures and Synthesis Algorithms for Power-efficient Bus Interfaces,” *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 969–980.

[19] C.-T. Hsieh and M. Pedram (2002). “Architectural Power Optimization by Bus Splitting,” *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 408–414.

[20] J. Y. Chen *et al.* (1999). “Segmented Bus Design for Low Power,” *IEEE Trans. VLSI Systems*, vol. 7, pp. 25–29.

[21] T. Lv *et al.* (2003). “A Dictionary-based En/decoding Scheme for Low-power Data buses,” *IEEE Trans. VLSI Systems*, vol. 11, pp. 943–951.

[22] Sharma, Archana C., and C. Z. Ali. "Construct High-Speed SDRAM Memory Controller Using Multiple FIFO’s for AHB Memory Slave Interface." International Journal of Emerging Technology and Advanced Engineering3, no. 3 (2013): 907-916.

[23] Kareemullah Shaik, Mohammad Mohiddin, Md. Zabirullah, “A Reduced Latency Architecture for Obtaining High System Performance”, IJRTE, 2012.

[24] Acasandrei, Laurentiu, and Angel Barriga. "AMBA bus hardware accelerator IP for Viola-Jones face detection."IET Computers & Digital Techniques 7, no. 5 (2013): 200-209.

[25] Gandhani, P., & Patel, C. (2011). Moving fromAMBA AHB to AXI Bus in SoC Designs: A Comparative Study. *International Journal of Computer Science & Emerging Technologies (IJCSET)*, *2*(4), 476-479.

[26] Kandiya, M. M. N., Harniya, M. M. K., & Govani, K. K. (2014). Implementation of Read/Write operation for AMBA AXI4 Bus using VHDL. *IJFTMR*. *I*, IV, 1-3.
[27] Samir Palnitkar. (2003). *Verilog HDL:* A Guide to Digital Design and Synthesis, Second Edition Publisher:Prentice Hall PTR.
[28] Xilinx Datasheet: www.xilinx.com/zynq7000-Pkg-Pinout.
[29] www.aldec.com/en/company/blog/144--introduction-to-ZYNQ-architecture.
[30] www.store.digilentic.com.