Optimization of a Component IP within an Infrastructure Subsystem

Major Project Report

Submitted in fulfillment of the Requirement for the

the Degree

of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

by

Ronita Mitra

(17MECV10)

Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382481

May 2019

Optimization of a Component IP within an Infrastructure Subsystem

Major Project Report

Submitted in fulfillment of the Requirement for the

the Degree

of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

by

Ronita Mitra

(17MECV10)

Under the guidance of

Mr.Prodip Kumar Kundu
Staff Verification Engineer,
Arm Embedded Technologies Pvt. Ltd,
Bangalore,Karnataka.

Dr.Vaishali Dhare
Asst.Professor in EC Department,
Nirma University,
Ahmedabad, Gujarat.

Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382481

May 2019

I dedicate my thesis to my friends and family.

DECLARATION

This is to certify that

1. The thesis comprises of my original work towards the degree of Master of Technology in VLSI Design at Institute of Technology, Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgement has been made in the text to all other material used.

-Ronita Mitra

17MECV10

CERTIFICATE

This is to certify that this project entitled **Optimization Of Component IP within Infrastructure Subsystem** submitted by **Ms. Ronita Mitra (17MECV10)**,towards the fulfillment of the requirements for the degree of Master Of Technology in VLSI Design, Nirma University, Ahmedabad at is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Place: Ahmedabad

Date :

Dr. Vaishali Dhare
Guide and Asst.Professor,
Nirma University,Ahmedabad

Dr. N.M.Devashrayee
Coordinator-VLSI Design,
Nirma University,Ahmedabad

Dr. D.K Kothari
Professor & Head ,
Institute of Technology,
Nirma University,Ahmedabad

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University,Ahmedabad

CERTIFICATE

This is to certify that the Project entitled **Optimization of Component IP within an Infrastructure Subsystem** submitted by **Ms. Ronita Mitra (17MECV10)**, towards the submission of the Project for requirements for the degree of Master of Technology in VLSI Design, Nirma University,Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Place: Banagalore

Date :

Mr.Prodip Kumar Kundu

External Guide,

Staff Verification Engineer,

Arm Embedded Techology Pvt. Ltd.,

Bangalore

ACKNOWLEDGEMENTS

With immense pleasure, I would like to present the report on the work **Optimization of Component IP of an Infrastructure Subsystem**. I am very thankful to all those who helped for the successful completion of the project work and for providing valuable guidance and knowledge throughout. I express my gratitude and sincere thanks to Dr. Vaishali Dhare, my internal guide of M.Tech VLSI Design and Dr.Niranjan M Devashrayee, PG Coordinator of M.Tech VLSI Design for guidelines and mentoring during the review process. I take this opportunity to express my profound gratitude and deep regards to Dr. Vaishali Dhare, guide of my internship project for her exemplary guidance, monitoring and constant encouragement.I would also like to thank Mr. Prodip Kumar Kundu and Mr. Khushal Gelda, external guide of my internship project from Arm Embedded Technologies Pvt. Ltd., for guidance, monitoring and encouragement and pushing forwards for the project.

-Ronita Mitra

17MECV10

ABSTRACT

Arm being a Semiconductor IP company, is a pioneer in market for providing IPs which are used in numerous field of SoCs such as arm based processors in automobiles, IoTs,servers, mobiles and lately in machine learning chips as well. Functional validation for these IPs is one of the challenging task in today's scenario as the heterogeneous complexity of SoC design has increased considerably.But, before delivering these IPs to numerous Arm Partners,verification of these ARM IPs focuses on extensively validating the inter working of multiple ARM IPs in a wide variety of practical kind of systems with wide range of configurability and scalablity which in arm environment is abbreviated **Kits**. With robust verification methodology of IP under the SoC environment helps us to detect bugs at an early stage of design cycle which can detect more bugs comparatively to only functionally verifying IP alone.Main challenge lying here is integrating various IP components within the System, is to achieve a target system performance with minimum resource allocation.So the main objective underneath is to understand the functionality of each module of base element(IP) of infrastructure SoC environment and knock out the additional logic's and modules used as well migration the STM(System Trace Macrocell) from base element to the dedicated Debug IP and thus achieving faster performance,lower gate counts,LUT's and validate it in a designed dedicated test suite environment. Being a part of Kits team as well as Memory modelling team, this dissertation also includes project work from memory modelling team where the primary objective was to write a generic test bench for the memory compiler validation suite to achieve the maximum functional coverage for testing each pins of sram memories. All the simulation and as well as the emulation results are included in this thesis.

Contents

List of Figures

ABBREVIATIONS

ARM	Acorn Risc Machine
SoC	System on Chip
SIF	System Integration Framework
AXI	Advanced Extensible Interface
AHB	Advanced High Performance Bus
APB	Advanced Peripheral Bus
AMBA	Advanced Micro-controller Bus Architecture
NIC	Network Interconnect
QoS	Quality of Service
QVN	Quality Virtual Network
DAP	Debug Access Port
TPIU	Trace Port Interface Unit
JTAG	Joint Test Action Group

Chapter 1

Introduction

1.1 Motivation

For an complete bug free SoC design like mobile,infra or automobiles SoC, verification is considered to be of prime importance. Most amount of time in design cycle is spent in functionally verifying the SoC design after integrating many IPs together as a complete system and these verification cycle consumes of about 60 to 70 percent of SoC design cycle. Even after so many stages of verification of these IPs, as the complexity in SoC integration increases the verification becomes more complex .Even after so much of efforts in validation of these IPs, as SoC technology is becoming more and more intriguing by integrating so many components into one which have elements like memory controllers, display controllers, interconnects, memory management units(MMU),interrupt controllers and other peripherals are embedded as well. The IPs individually are complicated units of design which can be validated personally but no matter how much of rigorous IP-degree verification, it isn't feasible to stumble on all bugs which are particularly those which are sensitized simplest when the IPs engage within a device IP. The objective of the verification team is to offer all arm partners with remarkable IP which have been established to internally perform efficiently and bug free.Thus delivering

the efficient IPs to all the partners and thus reducing the time to market cycle.

So in order to ensure that IP functionality remains the same as per the specification and design in a regular and reproducible way is the important thing intention of verifying these IPs, and also it is to be robustly verified keeping that in mind in the SoC environment. Thus, the focal point of verification is IP but in a any configurable SoC environment so arm tests this IPs under these scenarios of SoC with different system configuration which is called System or SoC integration. A package of SoC can described as a so many IPs included together for a selected target application segment (e.g. cell, IoT, Infrastructure and so on.).ARM generally consists of the complete variety of IPs such as CPUs, GPUs, display controllers,interconnects, memory controller, device controller, interrupt controller, debug and trace, other media components. A complete package of SoC is partitioned down into smaller components known as elements that are considered as basic building blocks for system integration. Thus Arm SoC framework contains as a minimum of one primary IP and white area logic around it. This Subsystems are made which is considered to be representation of ordinary SoCs with specific packages and applications.This integration results gives us the idea and a complete idea of the demanding situations seen by the surroundings of integrating diverse IP components to gain a goal gadget performance. [1]

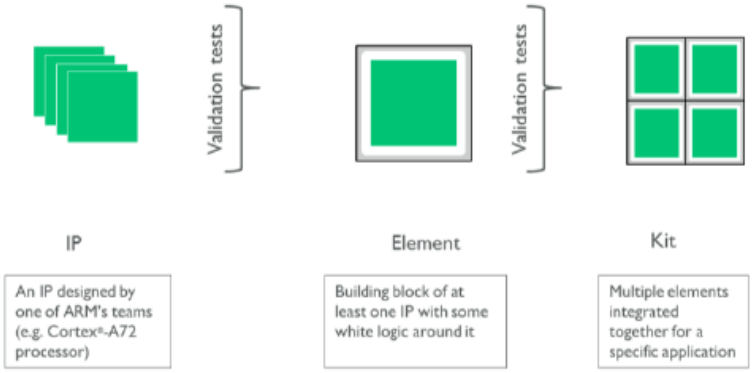

Figure 1.1: Multilayered approach to system validation

Thus as a part of kits team one need to ensure the integration of system aka kits using ARM IPs are integrated under one framework and validating that kits to ensure the testing done on these configurable kits by the system validation is to done successfully.

1.2 Problem Statement

To make sure that the SoC architecture of an infrastructure Subsystem .i.e. for server based system on-chip environment for Arm based IP's have re-usability, scalability and configurability.In order to ensure that arm has a dedicated automation framework known as SIF(System Integration Framework) which integrates the components of SoC with each dedicated IP and white space logic around it and verify it thoroughly on simulation and emulation platforms. Each components or generic common IPs needed around arm IPs have to be configurable and is distributed in terms of hierarchical manner which should acquire minimum resource allocation and completely optimized.Question arises that whether all the components used around ARM IPs are in most optimized form or any additional redundant logic's are present?

1.3 Approach

While gin through with the understanding of all the components around ARM IPs like for example arm processors one of the target components was base IP which was to be optimized.Since the kits build are broken down in an hierarchical manner with top down approach of hierarchical ways all the generic components like timers , watchdogs , interrupts , on chip RAM's and few are modules are integrated under one higher level of abstraction as a complete component or IP which is to be described as base component which is pre-rendered during the system build.

1.4 Scope of Work

Once the optimization of the base component is done and modifications on the RTL to be implemented,the validation is done on this component using dedicated testcases and regression to be done on simulation and emulations tools to ensure that the systems is working fine with the changes done . On an overall basis an achievement would to be reduce the extra resource allocated to this components which can reduce an overall area, power consumption and time as well as increase in speed.

Chapter 2

Literature Survey

2.1 Overview

Arm is a semiconductor company and ARMs IPs are used in a various range of SoCs,with the assurance that the era does exactly what it is designed to do in a constant form and be completely bug free thus ensuring configurability, resuabilty and scalabilty.The focus of verification lies particularly these ARM IPs, however in a totally sensible SoC environment in which these IPs [1] with unique configurations are known as **Kits** for complete SoC flow in order to find more bugs.

The system validation in ARM makes use of variety of stimulus and to take a look at methodology one need to take a look at kits. These test case stimuli is in general software tests which run on CPUs within the device.The tests case written are usually in high-level language i.e. C language or assembly level and the generated test cases make use of the Random Instruction Sequence - RIS equipment so that it will be defined in the imminent sections. In addition to code going for walks on CPUs, a set of Verification IPs (VIPs) are used to to be into the system machine or to behave as observers.

In guidance for validation, a test plan is created for each IP within the

package. Test making plans captures various IP configurations, features to be tested, eventualities with a purpose to be protected, stimulus, inter operations consideration with IPs, verification metrics, tracking mechanisms, and diverse flows with a purpose to be part of verification. Testing of kits begins with a simple stimulus this is step by step ramped up to extra complicated stress instances. The system validation crew at ARM has mounted a repeatable and automated package improvement float, which lets in us to build a couple of kits for distinct segments. The mix of IPs, their internal configuration, and the topology of the system are selected to reflect the extensive range of end uses. The kits are tested on two number one targets emulation and FPGA. [1]

- five-6 trillion emulator cycles.
- 2-3 peta FPGA cycles of Kits validation.

A traditional Infrastructure is subdivided into elements, as shown inside the diagram.

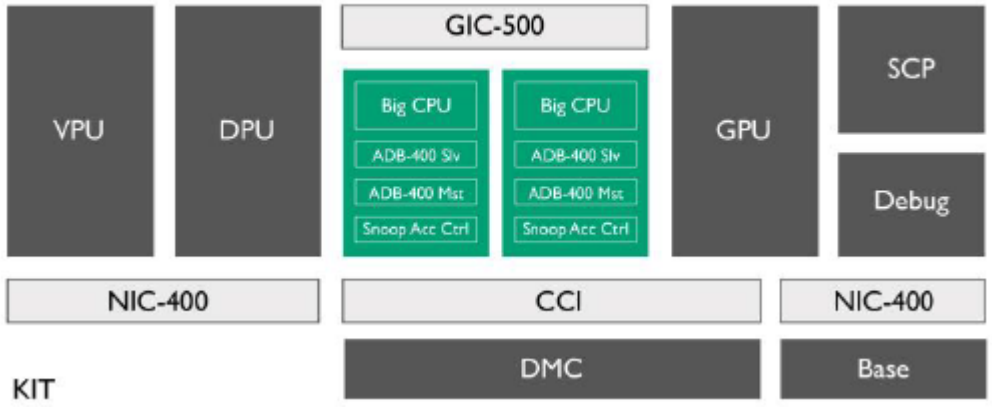

Figure 2.1: Basic Infra Subsystem

Each kit is a set configuration of elements connected together in a described topology.To deal with the automation requirements of the general Kit work-flow; integration is carried out the usage of **System Integration Framework (SIF)**. The system takes in elements with top-stage integration information as

enter to generate the Kit pinnacle-level Verilog and related documents required for compilation. The figure underneath describes the general float.

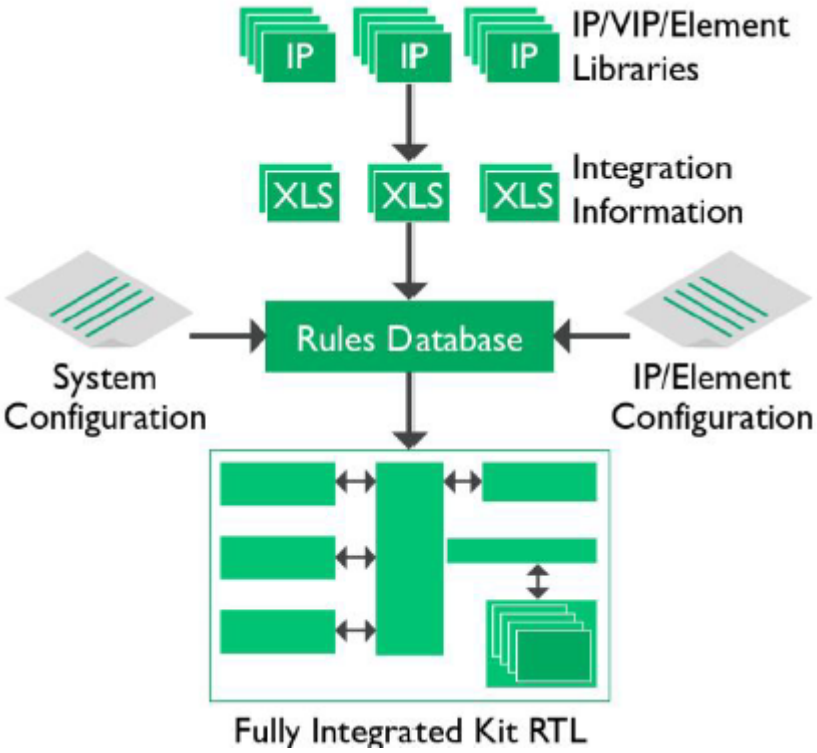

Figure 2.2: System Integration

Newer IPs are absorbed into their respective element class, e.G. CPU detail is greater to guide ultra-modern cores, at the IP Alpha milestone. The new element versions are then integrated into a defined kit.

Kits are categorised into two broad marketplace segments which are cellular and infrastructure. Each segment is further bifurcated as - high-end, extent, wearable (in cellular) and optimized infrastructure, mid-variety infrastructure, and high-end infrastructure(server).

2.2 Server Based SoC Architecture

The ARM structure profiles, Application, Real-time, and Microcontroller(ARM), exist in element to segment that explain the traits of specific target markets.

The variations among merchandise targeted at extraordinary profiles are massive due to the various practical necessities of the marketplace segments. This phase specifies a hardware machine structure, based totally on ARM 64 bit architecture, which server machine software program, such as running systems, hypervisors and firmware can rely on. It addresses system capabilities and key elements of the device architecture. The main intention is to make sure sufficient standard device architecture to allow a definitely-constructed single OS image to run on all hardware compliant. The ARM architecture has basically three profiles which are as follows i.e A for Application,R for Real-time, and M for Microcontrollers, exist in element to section the solutions produced to explain the traits of particular goal markets.

Infrastructure system supports configurable numbers of cores and clusters.Most elements contain clock,reset and power control logic.An ARM based interconnect which connects all the components together to interact with each other.All the processors targeted for specific functionality are bifurcated as Application processors(APP), System control Processors(SCP) and Manageability control Processors(MCP). System includes Memory controllers,generic interrupt controllers, clock generators and on chip memories and Arm CoreSight technologies.The following elements have been defined based on the diagram shown below: [2]

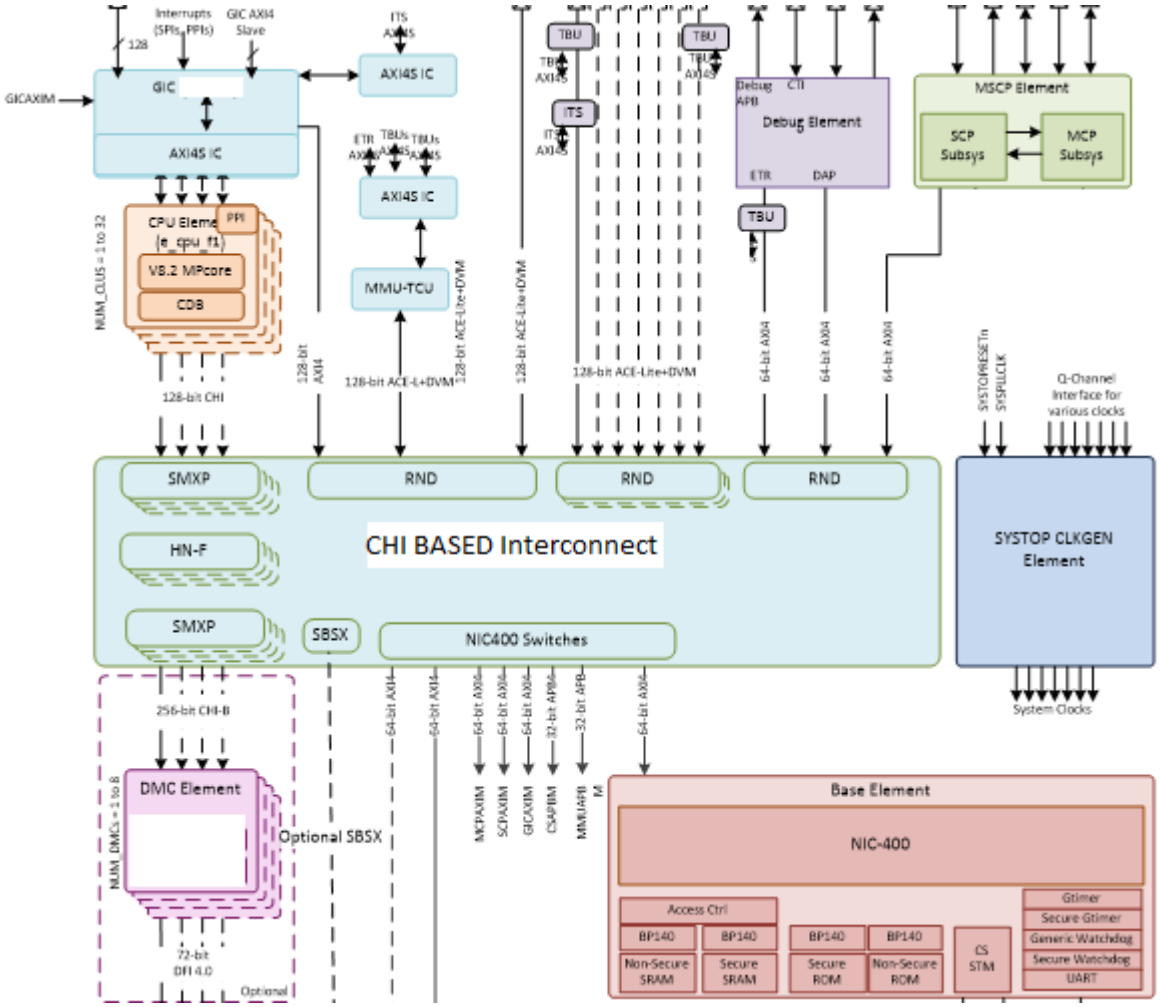

Figure 2.3: Generic Infrastructure Subsystem

- **CPU Element :** Contains High performance with ARM based CPU Clusters.
- **System Control Processors :** This element consists of targeted CPUs which are ARM micro controller based system control and manageability control processors.
- **Base Component :** It includes mainly system periperals which are generic timers, generic watchdogs,scratch ROMs and RAM's.
- **Interconnect Element :** It consists of cache coherent mesh interconnect and switches that provides master and slave extention interfaces.

- **Memory Element :** Consists of dynamic memory controllers which implement CHI(Cache coherent mesh) datapaths to cache coherent interconnect with clocks and reset logics.

- **Clock generators** Clock generation logic are used in the top level of the design.

- **Debug Element :** This implements the coresight based sub-system along with coresight technology.

- **Voltage and Power Domains:**A power domain is a collection of hardware modules within a voltage domain that share common power control. A voltage domain can have one or more power domains. A power gated domain is a power domain whose power can be removed by on-chip power switches.

2.3 Memory maps

The memory maps for the processors are arranged in the following manner with application processors , system control processors and manageability control processors.The memory map in this and the following sections also shows the overall security attributes associated to each area of memory which are grouped as **Always Secure Access** which is component or region that is only accessible to secure transactions. Any non-secure access targeting these will result in a decoder response.and **Non-Secure Access** which is a component or region that is accessible to both secure and non-secure transactions. Figure **2.4** shows the memory map addressing.

map.png

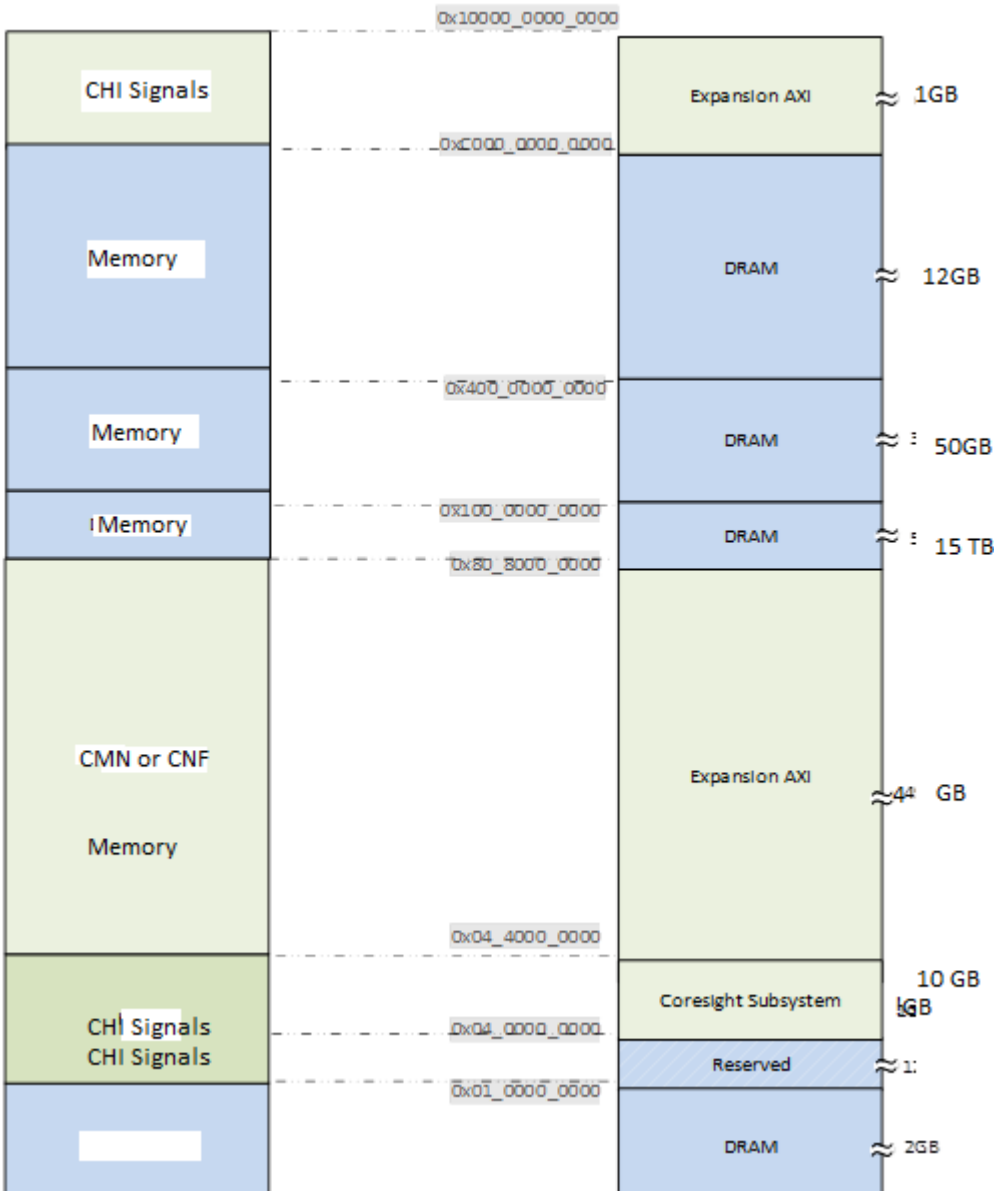

Figure 2.4: Top level Application memory map

All Non-secure on-chip masters in a base server system that are expected to be under the control of the operating system or hypervisor must be capable of addressing all of the Non-secure address space. If the master goes through a System Memory Management Unit(SMMU) then the master must be capable of addressing all of the Non-secure address space when the SMMU is turned off. [2]

2.4 Interrupt Controller

The Generic Interrupt Controller (GIC) structure defines the architectural requirements for coping with all interrupt assets for any processor related to a GIC and a not unusual interrupt controller programming interface applicable to uniprocessor or multiprocessor structures. The GIC is a centralized resource for supporting and managing interrupts in a system that consists of as a minimum one processor. Registers for managing interrupt assets, interrupt the conduct, and the routing of interrupts to one or greater Components in structure.

2.5 I/O Virtualization

Hardware support for I/O Virtualization is optional, but if required shall use a System MMU compliant with the ARM System Memory Management Unit (MMU) specification.Each function, or virtual function, that requires hardware I/O virtualization is associated with a SMMU(System Memory management Unit) context. The programming of this association is IMPLEMENTATION DEFINED and is expected to be described by system firmware data. [2]

Chapter 3

The Base Component

3.1 Architecture definition

As per the discussion in the above sections, the components or IP within the SoC are described in abstract way to ease the understanding of connectivity's in between the IPs.The base element or component is one of the internal IPs in the system can be defined as an abstraction level including system peripherals, watchdog timers, counters, scratch RAM's and ROM's along with the internal debug and trace elements known as System trace macrocell(STM) and UART(Universal Asynchronous Transmitter and Receiver).The figure below shows the hierarchical arrangement of the base component.

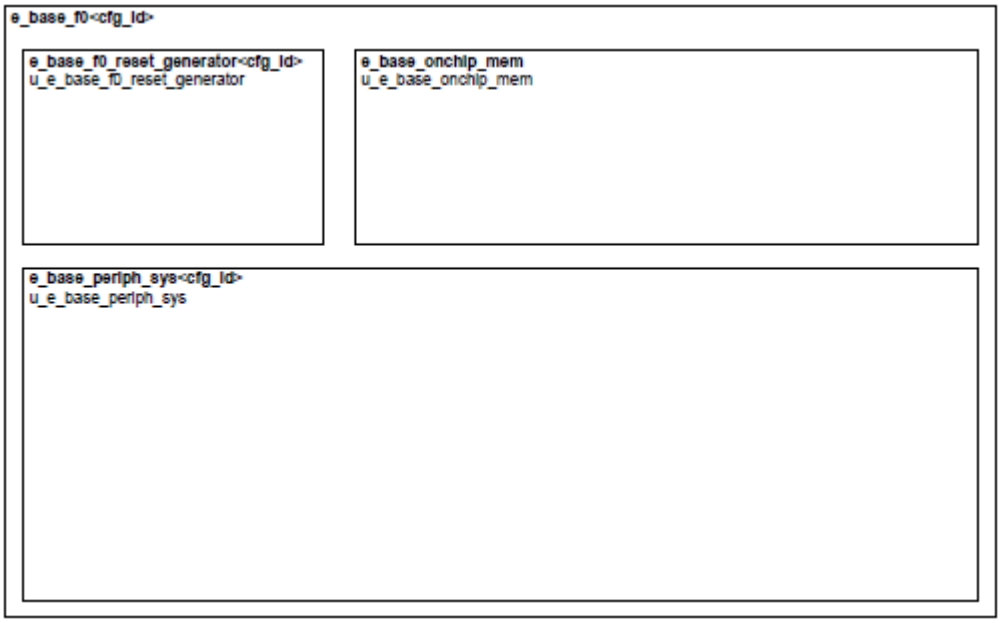

Figure 3.1: Top level Base hierarchy

3.2 Functionality of each module

Since base component comprises of many elements, each modules functionality combined as a whole will describe the overall base component.After understanding each functionality ,description on how the base component is optimized will be covered under this chapter.Below subsection will define the functionality of each modules within to give clear picture of optimization.

3.2.1 Reset Generators

The Base Element implements numerous different reset domain names. The determine beneath suggests the distribution of resets within the element. The cause of the reset domain is whenever the system powers down and want as to boot up once more within a couple of clock area. All connections inside the element no longer belonging to any reset area are assumed to be asynchronous. In a a couple of clock domain design, an asynchronous reset ought to be one by one synchronized for every clock domain as shown in parent

underneath. A design can also have a couple of asynchronous reset sources, which includes external reset (in all likelihood cleaned up from glitches), inner controls (e.g. Voltage area reputation indication) and PLL(Phase locked loop) lock situations. All those together asynchronous indicators may be blended to a unmarried Reset Enable condition, as proven in Figure The top-level system reset SYSTROPRESETn is synchronized with each clock, and then buffered, providing a synchronous reset for each clock domain.

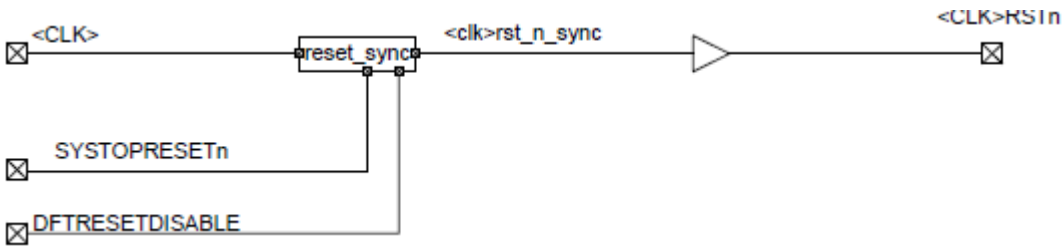

Figure 3.2: Reset generator

Since every reset synchronizer incurs an additional non-deterministic synchronization postpone, expressed in phrases of the targeted clock area cycles, the exclusive clock domain names may additionally go away the reset state at arbitrary exceptional instances. Considering the complexity of a couple of clock domain names, multiple reset situations and a couple of inter-clock reset dependencies, IP-based control of the reset circuitry is favored.

3.2.2 Peripheral Interrupts

Physical Interrupt Interface(PII) are the interface which consists of the interrupts generated by the Generic Timers, and the Generic Watchdogs.

3.2.3 On-Chip Memories

On-Chip Memories that are Scratch ROM(Random On-chip Memories) and Random Access Memory(RAM's) which is used as a boot memories for application processors core with memories with additional Memory BIST support

and different memory interfaces within.

3.2.4 Watchdog Timers

The Generic Watchdog consists of cold reset and a Warm reset.For a cold reset, certain register values are reset to a known state.Watchdog Cold reset should just happen as a component of the watchdog powers up.On a Warm reset, the condition of the watchdog isn't reset, yet other rationale that can be AHB or APB.This is to understand that the system undergoes reset sequence whereas the watchdog retain those states of the events and can be examined during the system in operation.

Watchdog timers signals are connected to interrupts where the system can undergo refresh states for proper watch period.And if these refresh occurs successfully then system can undergo normal state whereas if refresh doesn't occurs during the watchdog refresh expiring then a signal generates There are two kinds of refresh state occurs in watchdog timers.

- Timeout refresh occur when the watchdog is enabled and the input time value is greater than the value of the Watchdog Count Value Register (WCVR).
- Explicit refresh occur when one of the following registers is written with Watchdog Offset Register (Control frame), Watchdog Control and Status Register (Control frame) and Watchdog Refresh Register (Refresh frame)

The watchdog generates two interrupt outputs based on sequences of these events.The implementation uses a state machine to track the sequence of events, and the state of the state machine is decoded to generate interrupt outputs.

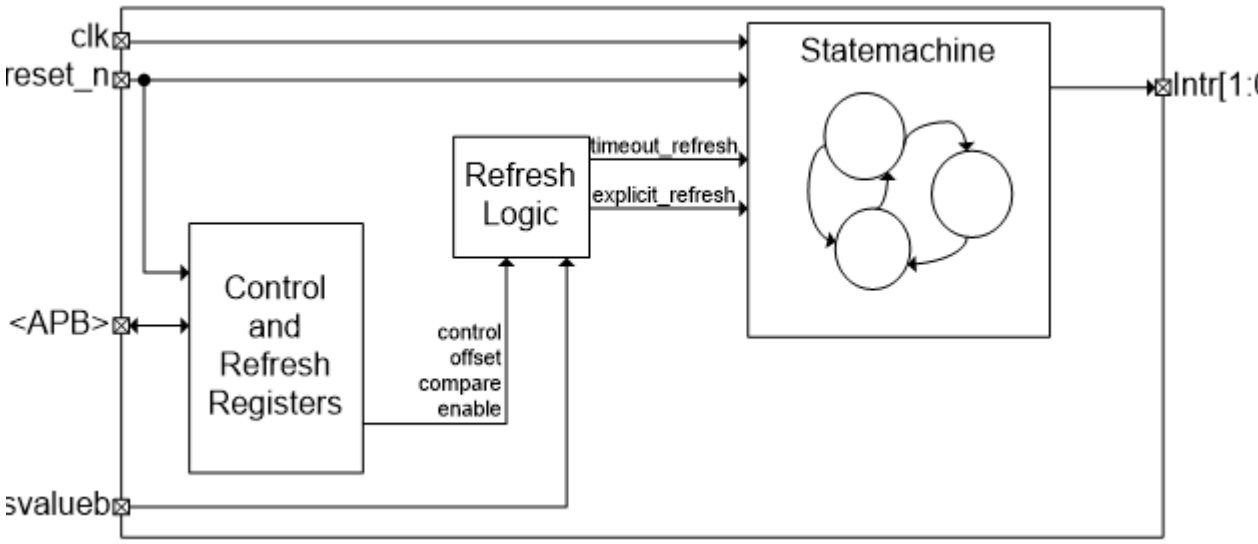

Figure 3.3: Watchdog timer

3.2.5 Generic Timer Control

Timers control in any system is used to execute the scheduled task repeatedly within the system.Also it has many other application such as capturing the timestamp between two events Considering arm cores, there is a system generated clock timer for OS images, they uses periodic timer control to generate interrupts and on each clock pulse and these timer increases and decreases the counters.As soon as an interrupt is generated as per timer starts with the counter .

3.2.6 UART

The UART is an Asynchronous Transmitter and Receiver slave module connected within any System-on-Chip (SoC) peripheral with APB(Advanced Peripheral Bus) which is abstracted in the base component within the infra system .The UART has inclusion of Infrared Data Association (IrDA) Serial Infrared (SIR) protocol Encoder/Decoder.As UART performs serial-to-parallel conversion on data received from a peripheral device and parallel-to-serial con-

version on data transmitted to the peripheral device.

Generic UART is designed to offer a basic facility for software bring up and as such specifies the registers and behavior required for system software to use the UART to receive and transmit data. This specification does not cover registers needed to configure the UART as these are considered hardware specific and will be set up by hardware-specific software.

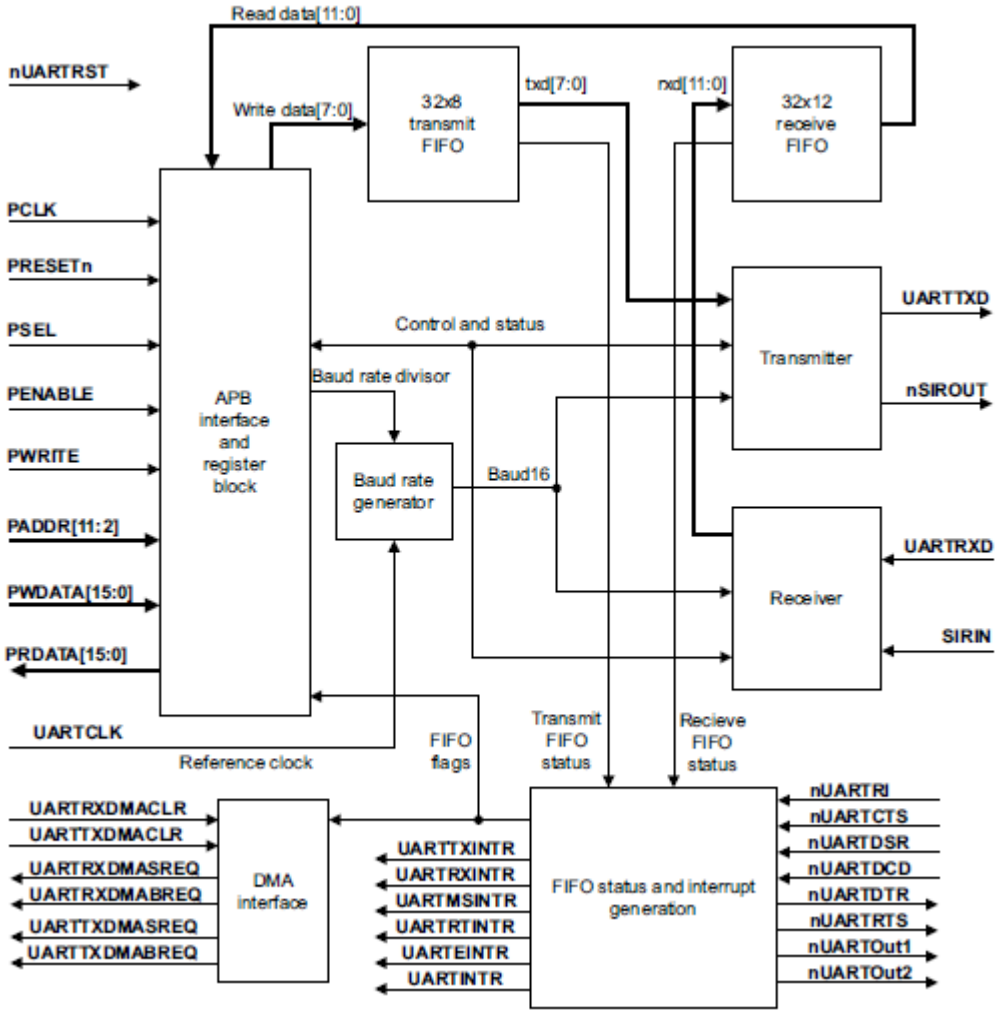

Figure 3.4: UART functional block

The CPU reads and writes statistics and manage/reputation records through the AMBA APB interface. The UART includes a programmable baud charge generator that generates a common transmit and acquire internal clock from

the UART internal reference clock enter UARTCLK. The UART can generate personally-mask-able interrupts from the receive (such as timeout), transmit, modem fame and errors conditions in a single combined interrupt in order that the output is asserted if any of the individual interrupts are asserted and unmasked DMA request alerts for interfacing with a Direct Memory Access (DMA) Controller. [2]

3.2.7 System Trace Macrocell

System trace macrocell is an internal debug and trace module within an arm subsystem which is a part of core sight technology.More details on Coresight technology and functionality of STM is described in chapter 5.

3.2.8 Access Control gates

Access control gates is a mechanism to manipulate the passage of unique channels. It blocks get right of entry to when the get right of entry to manage gate is closed.AXI3 protocol is not supported however if used there's a predicament that the memory needs to now not go back B channel responses before receiving both AW and W channel requests.There are two LPI Q-channels applied for every ACG.

3.2.9 Clock gating and power gating interface

There are two Low power interface applied for each modules

- **Power Q-Channel** :This allows conversation between the ACG and the electricity controller. It uses its PWR QACTIVE sign to suggest the requirement for the gate to be opened or closed. Its handshake protocol lets in the strength controller to carry out manage sequences at the gate.
-]**Clock Q-Channel:**This enables communique between the ACG and a

clock controller. It uses its clock q channel active sign to signify when the clock is needed.

- ACG counts splendid transactions and responses at the ACE or AXI4 channels it sits on. It ensures tremendous transactions are completed before declaring power and clock qchannel active and accepting any Q-Channel quiescent requests

3.3 Optimization Methodology

All the components in an SoC has a support of low power modes i.e with clock and power gating support .Considering only base element with complete understanding of each modules within the base element which has a support for power and clock gating within the modules.Since the base element stays in an always on mode the need for access control gates which acts as a switch between network interconnect(NIC) and on chip rams for low power mode is an redundant module within an system which occupies extra modules and interconnects for AXI4 to AXI3 modules. [3] Thus it is much needed to optimize that part in the base element. Also the clock gating signals which goes into the modules of interconnect and access control gates need to be optimized.
In the entire hierarchical approach of the base element design knocking out these modules from the entire system results in an optimized IP with reduced gate counts, LUT's and registers.This method of knocking off the elements from the entire hierarchy is to be defined as flattening of the component IP. In this methodology flow the NIC interconnect is re-rendered again with set of new AXI3 ports and the Access control gates modules are knocked out from the modules and RTL of base component. This new re-rendered component are stitched together with the other ARM IPs in the System Integration automation Framework which automatically connects all the components together. Since the changes to be done in the base component and then whether its

affecting the other components or not. This the system in form of kits is to be tested on the validation suite automation framework and with the regression of the various tests one can conclude that the functionality of the system is working okay and the test passes every time.With the support of emulation from various vendors within arm environment we can verify the design on different emulators which gives us gate counts , optimization in terms of power ,area and speed.

Chapter 4

Network Interconnect

4.1 Overview

The network interconnect is a library of interconnection with more than one masters and slave interfaces and additionally exceedingly configurable and multi-electricity area tools.The Network Interconnect is a package of key interconnect IP that allows you to build a scalable and configurable network interconnect.One can integrate the NIC with AMBA Domain bridges or thin Links bridges into a single interconnect. The NIC additionally includes: Low Power Distributor. CoreLink AXI4 to AHB(Advanced High-Performance bus)Bridge. Network Interconnect Advanced Quality of Service. Advanced Quality of Service for Virtual Networks. One can utilize the high level of configurability of NIC for optimization and tuning. The blessings of using the NIC-450 are: Unified low-power interfaces while applicable. Single design environment to configure IP blocks and connect them collectively. Using the NIC with CoreLink internal tool Creator employs algorithms to aid the introduction of valid configurations that are based on one's specific layout necessities.

NIC interconnect consists of multiple switches with support of AMBA master and slave interface blocks.It supports up-to one to 128 master and slave inter-

faces with hierarchical clock gating support within. The NIC has a support for QoS i.e Quality of Service programmable facilities as well as have support for QVN (Quality Virtual Network) extension which provides a way which avoids blocking of overhead path and cross-path blocking between different data flows and one uses this QVN to allocate the register space to virtual channels within the interconnect in the SoC. [4]

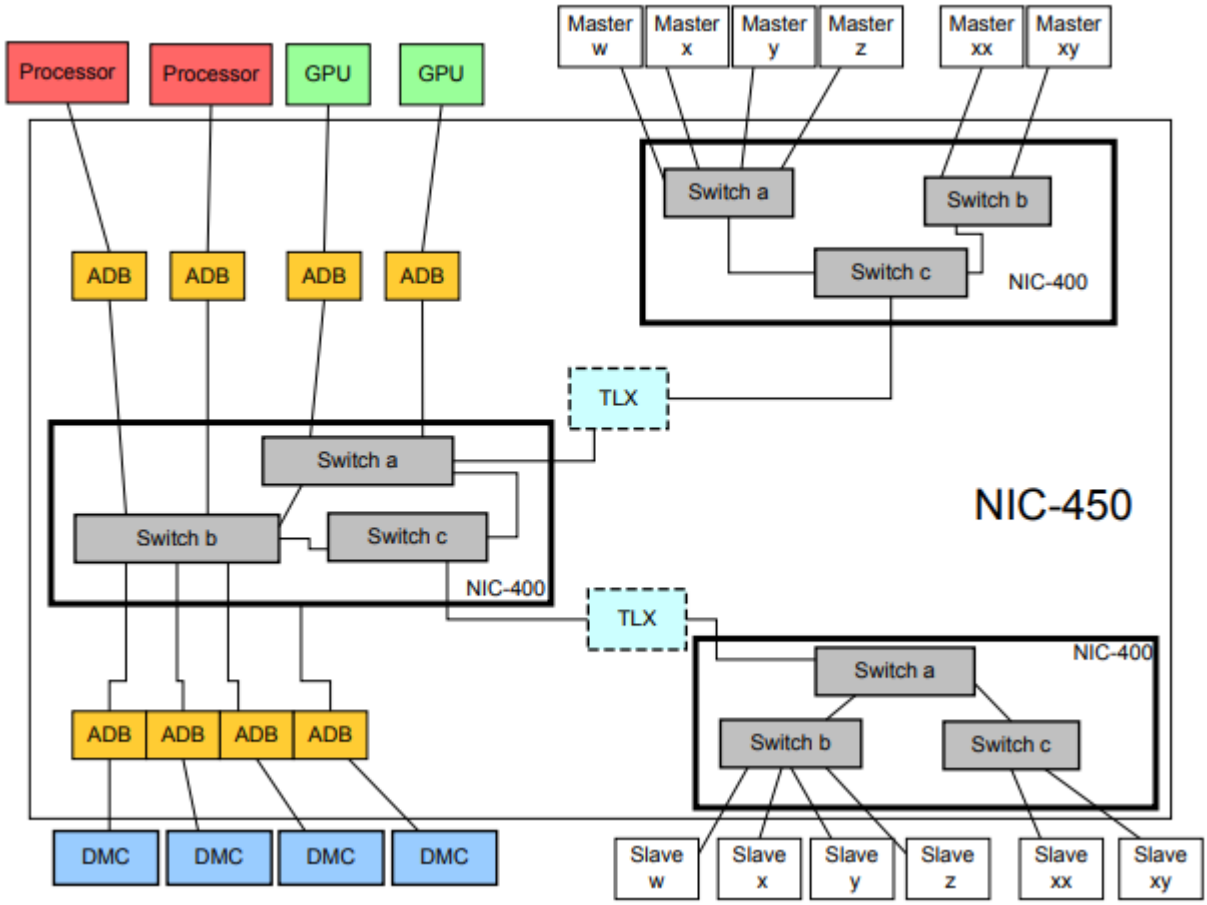

Figure 4.1: NIC functional block

4.1.1 Master interfaces

The NIC can have following types of protocols from the AMBA variants which are AXI ,ACE ,APB, AHB master interfaces.It has many configurable options while designing them in AMBA Designer which are address width, data width, the types of timing closures, think links , user sideband signals and upsizer-

downsizer widths.It also have the options of security with trustzones like secure access where one can only have secure transactions for accessing the components, non-secure access where both secure and non secure can have access for the transactions and boot Security access where the software determines which is to be permissible for accessing the components.
This NIC also has a support for clock and power gating as a low power interfaces which is a C channel in AXI domain with hierarchical clock feature.

4.1.2 Slave interfaces

The NIC supports many kind of ARM variant AMBA Protocols like AXI,AHB,APB,AHB protocols which depends upon the version of the NIC used by the user.It also has configurable ports such as address width, data width, data upsizers, downsizers, read and write acceptability, synchronous and asynchronous frequency options, timing closures options, thin links, security options like Secure and Non Secure per access for accessing the components.

4.1.3 Protocol Conversion

One the main advantage for using NIC interconnect is that different components works on or supports different kinds of AXI or ACE Protocols from AMBA series of protocols by ARM.So for these components with different protocols to communicate with each other, the conversion is necessary and NIC does that . Considering an example of AXI3 to AXI4 or AXI4 to AXI3 protocol conversion the where AXI4 will have more number of transactions, bifurcation of long bursts is done into small AXI bursts and the output transaction can be determined as a ratio of total incoming bytes for the input transaction to the total number of bytes for the output beat.

4.2 Using ARM AMBADesigner

AMBA Designer is a configuration tool using which NIC switches can be designed using a GUI mode that generates a specific implementation of a CoreLink NIC-400 Network Interconnect. AMBA Designer drives the Interconnect generation engine to provide the following for a set of configuration parameters after the design is complete which are:

- Verilog Register Transfer Level (RTL)
- testbench and stimulus
- synthesis scripts.

AMBA Designer supports the following design tasks:

- Configuring CoreLink and CoreSight devices.
- Generating RTL for the configured devices.
- Optimizing the AMBA interconnects.
- Stitching together interconnects and CoreLink or CoreSight components into an AMBA-compliant system with IP-XACT stitching.
- Optimizing the AMBA interconnects.

AMBA Designer generates different kind of configurations that are compatible on all platforms and generate the RTL Verilog files and the associated Out Of Box (OOB) test benches for verifying the RTL. In the AMBA Designer has configurable IP components and one can use them to create systems using a graphical representation that shows the components, their ports, and the connections between ports.One can also use AMBA Designer from the command line in batch mode. For downloading one must have all the required libraries of IP bundles installed. All the specification of which protocols supports are

there for which component of NIC is given from the option of which module to design as per the versions. The workflow for the amba designer is give in below figure .Ambadesigner has following design flow for creating the different sytems keeping mind the re rendering of NIC which is highly configurable using this tool version.All the peripherals and interconnects od SoC used are configrable in larger extent. Figure below describe the complete flow of Arm amba designer using which NIC module was re-rendered in our design optimization with genrated RTL and testbench around the system.

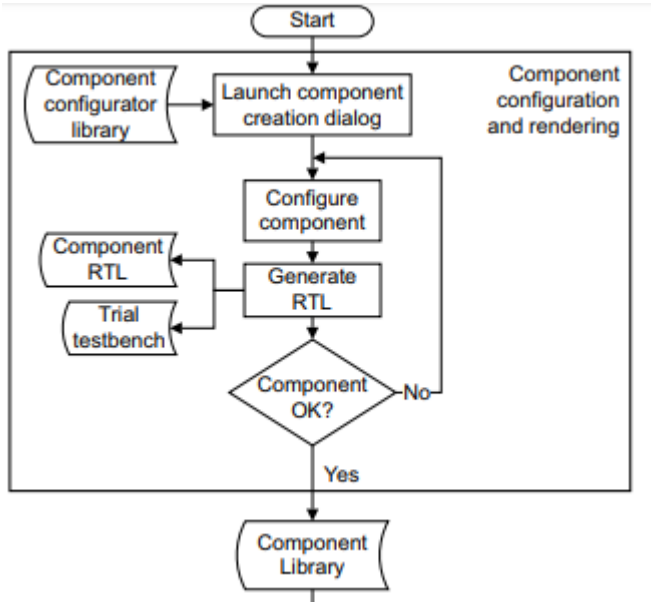

Figure 4.2: Part1:Design flow of Arm ambadesigner

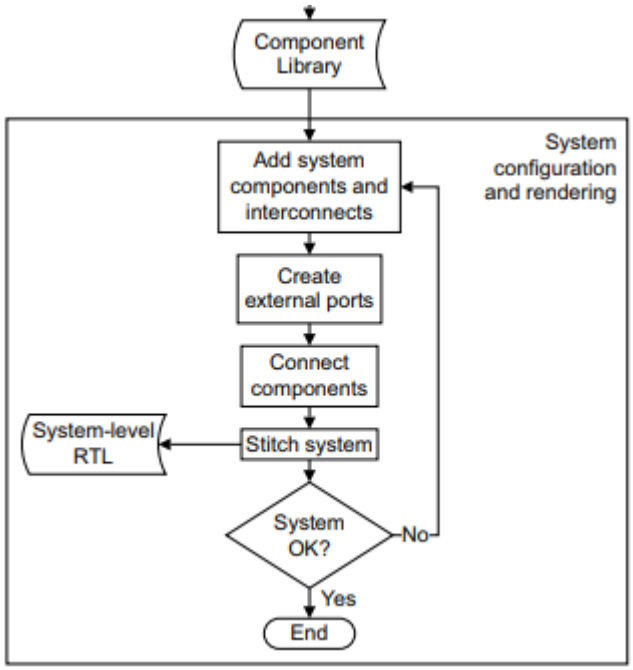

Figure 4.3: Part1:Design flow of Arm ambadesigner

Chapter 5

CoreSight on-chip Trace & Debug

Arm CoreSight technology is primarily an ARM-based IP which is used to debug and trace software that runs on Arm-based SoCs.Debugging functions are used to take a look at or adjust the states of parts of the design whereas trace features permit for non-stop series of device information for later off-line analysis.With Core Sight, each are used together at all stages of design flow.The systems included in this chapter demonstrates the most basic configurations of a CoreSight system. More complex systems might involve clusters of processors, multiple clock-domains etc. [5]

- **Debug functions** mainly used to read the states of the components under operation or during the booting options.It can also change the value stored in registers of the components. Mainly the debug has a feature to collect the values of registers and monitors it as well as can change the values during booting issues.

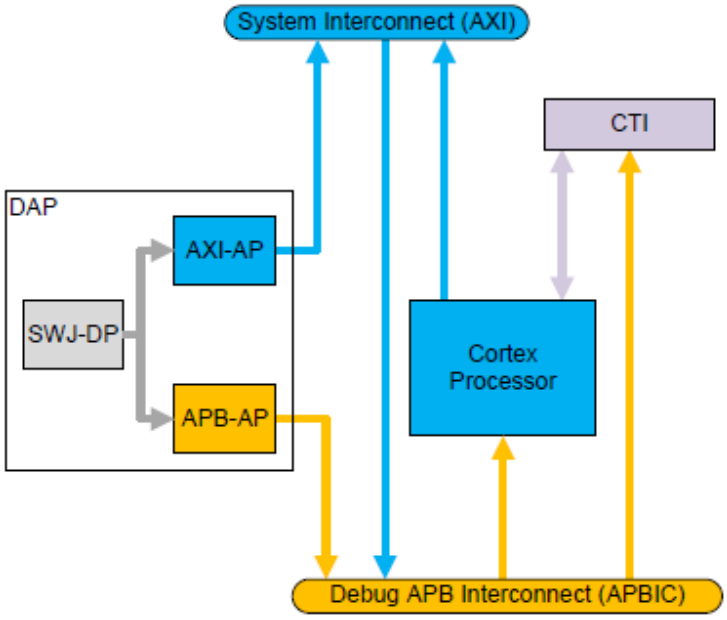

Figure 5.1: Single processor with Debug APB access

- **Trace functions** will collect the values of the states or monitor the states during the system is the non working state analysis.Execution trace generation macrocells exists within systems, with dedicated software and some peripherals for trace generation and monitoring trace streams. [6]

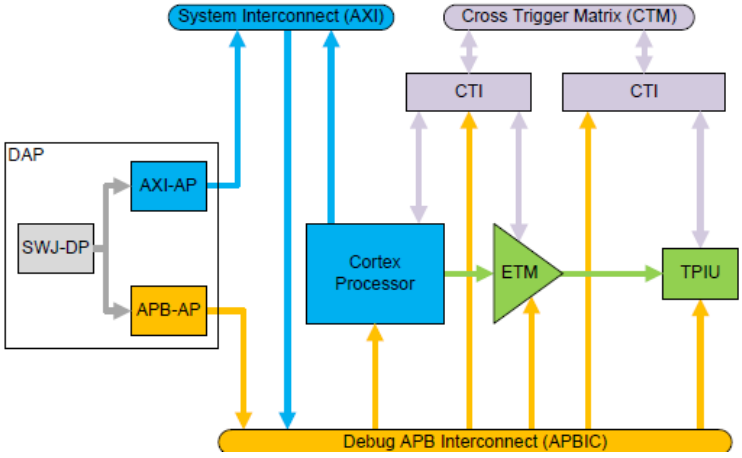

Figure 5.2: Single source trace with the TPIU

5.1 About Coresight Component

Earlier before the coresight technology, the listed methods were considered for debugging: [6]

1) JTAG as an debug element which acquired hard coded pins that is invasive debug with core halted the usage of:

-Breakpoints and watchpoints are two things which used to stop the activity withing processor for some amount of time ony on a particular activity.

-A dedicated pins were used to look at and regulate registers and memory and provide unmarried-step execution.

2) Conventional self-hosted debug reveal that is invasive debug with the processor running the use of a debug monitor that is living in memory.

3) Trace is non-invasive debug with the processor walking at complete velocity using:

- A series of facts on practice execution and information transfers.

-Delivery off-chip in real-time, or capture in on-chip memory.

For committed IPs of CoreSight Technology, a multicore debug and trace answer has the subsequent advantages over the traditional approach:

- debug and trace visibility of whole systems
- pass triggering support between SoC subsystems
- higher facts compression than previous answers
- multi-source hint in a single stream
- trendy Programmers Models for standard device assist
- open interfaces for third-party cores
- low silicon overhead.

CoreSight Technology will increase the debug task in one-of-a-kind trends of SoC design: [5]

- Systems are tracing extra facts per sec and should transfer this out of the SoC so that pin interface frequencies are not rising as fast as on-chip frequencies.
- System logic is sufficiently decoupled from core execution to require direct visibility.
- Clock and energy domain implementations are complicated.

5.2 Coresight Components

This segment describes a number of the fundamental capabilities of CoreSight Technology that enable us to deal with the troubles and demanding situations of debugging complicated SoCs.This section also describes the individual components that make up CoreSight systems and it's overview.
Figure **5.1** shows a system containing CoreSight components.

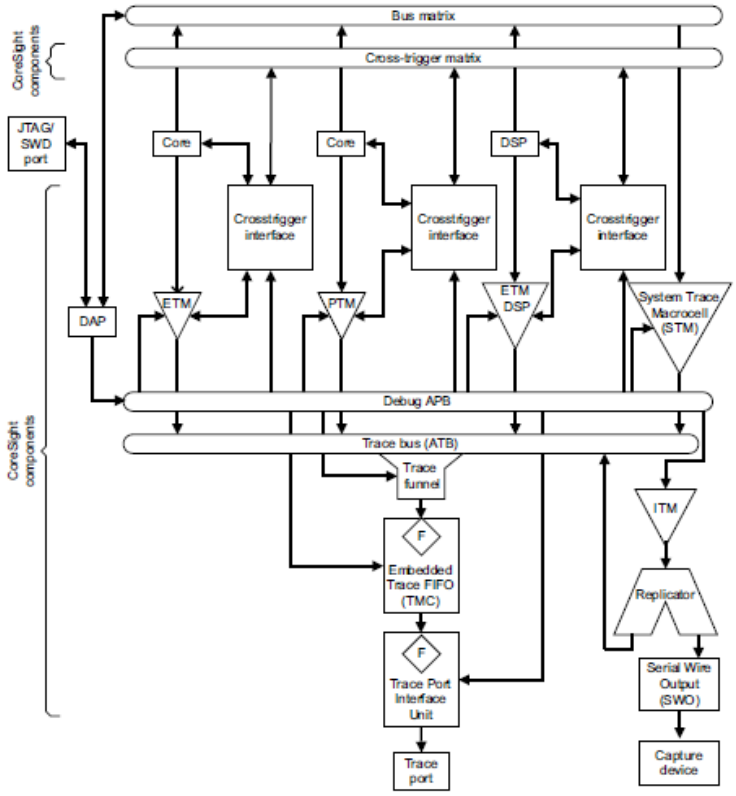

Figure 5.3: Coresight System Components

5.2.1 Debug Access

The debug access in CoreSight systems use of the Debug Access Port (DAP) which offers real-time access of memory without halting the core also debug control to all status registers. This is quicker than the conventional JTAG mechanism that makes use of the processor core to write data to memory when the debugger does not support this approach.The figure below suggests an example device with debug components and a DAP in SoC design.

access port.PNG

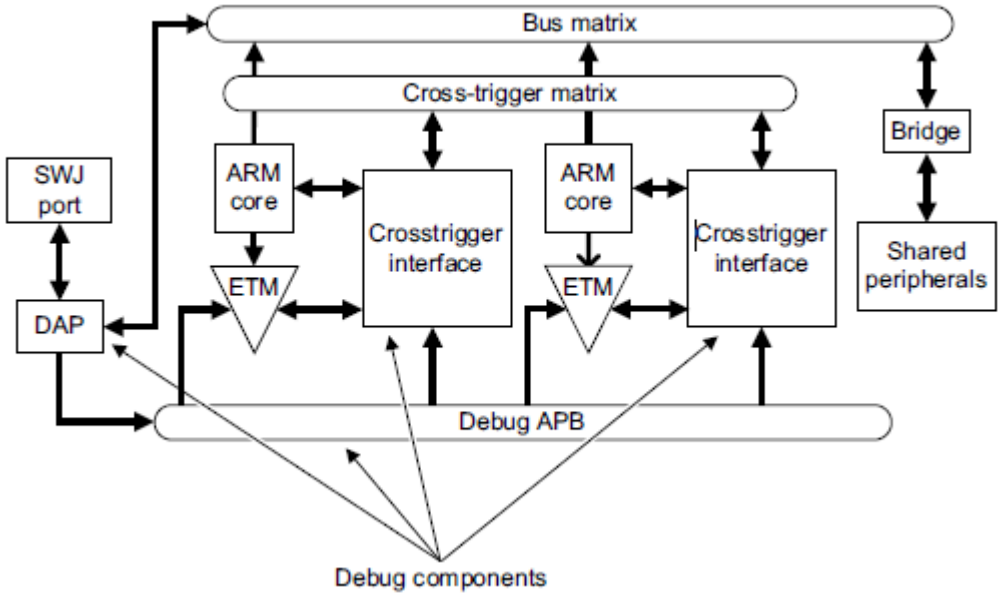

Figure 5.4: DAP Connections inside an SoC

The DAP offers the following advantage for multi-core SoC designs:

- Since the component with no low power support or an always on element has no effect on the access of debug to different components, minimum power authorities affects low on debug.
- The rate of accessing components is not affected by other IPs of SoC and one can have direct access to each individual devices.
- More than one core can control debug capability, in place of not allowing this to the core being debugged.
- A choice of physical debug interfaces is present as JTAG is not the single choice for a device because of other lower-value interfaces are viable.
- The device does no longer ought to support a totally asynchronous clock, TCK, due to the fact the DAP manages the clock. Debug clock synchronization is a problem for synthesized cores due to the fact it's far

simpler to preserve the frequency of clock properly underneath the processor clock, or use a handshaking clock signal and also does not require a go back TCK, RTCK, off-chip due to the fact synchronization is accomplished inside the DAP.

- One makes good sized financial savings in gate vicinity by using not having to implement a TAP controller and associated clock domain synchronization circuitry for each new debug detail inside the SoC.
- One add more manage over software get admission to to the debug register file among the middle and the debug bus.
- With complete backward compatibility, we can serially daisy-chain JTAG test chains with the DAP to offer access to them.

5.2.2 Cross triggering

The Embedded Cross Trigger (ECT), comprising of the Cross Trigger Interface (CTI) and Cross Trigger Matrix(CTM), provides a preferred interconnect mechanism to bypass debug or profiling activities across the SoC.

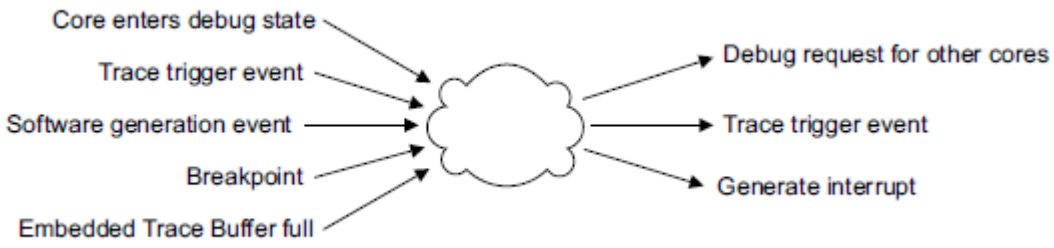

Figure 5.5: Cross triggering

The ECT presenting along with a very known method to attach distinctive signal types. Distinctive layouts of popular triggers for cores and Embedded Trace Macrocells(ETMs) are predefined and one can upload triggers for

third-party cores.The ECT permits tool developers to deliver a preferred manage conversation so that software program programmers can connect trigger activities.

5.2.3 Trace

CoreSight Technology presents additives that support a standard infrastructure for the capture and transmission of hint data, a mixture of multiple records streams by way of funneling together, and then output of facts to a hint port, or storage in an on-chip buffer. Figure 5.4 suggests some CoreSight components.

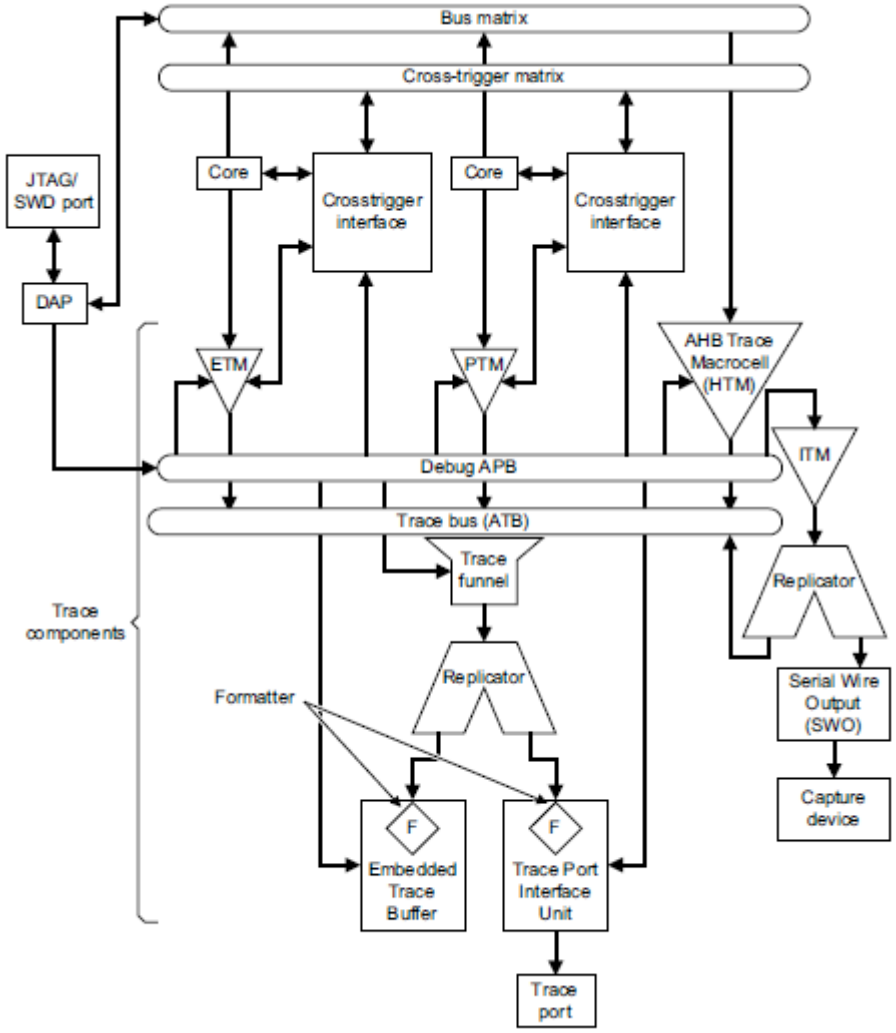

Figure 5.6: Example system with trace components

CoreSight Technology permits:

- simultaneous hint of asynchronous cores, busses, and wise peripherals
- debug and hint of an AMBA2AHB bus.
- tracing of instrumented bus masters.
- the output of trace data to the Trace Port which can run at an unbiased frequency to on-chip busses ,an embedded trace buffer for on-chip garage of hint facts in committed RAM or machine RAM, the DAP (Debug

Access Port) for low-call for trace solutions in pin count constrained objectives.

5.2.4 Buses

The CoreSight structures use the subsequent bus protocols to attach components collectively, and to allow integration in an SoC:

- **AMBA Trace Bus (ATB):**The ATB transfers trace data through the CoreSight infrastructure in an SoC. Trace assets are ATB masters, and sinks are ATB slaves. Link additives offer both master and slave interfaces. The ATB transfers trace information thru the CoreSight infrastructure in a SoC. Trace assets areATB masters and sinks are ATB slaves. Link components offer each master and slave interfaces.

  The ATB protocol helps stalling of hint assets to enable the CoreSight components to funnel and integrate sources into single trace flow, an association of trace data which can generate source using source Trace ID. Also, a CoreSight device can trace as much as 111 special items at any one time. A flushing mechanism is gift to pressure historic trace to drain from any assets, hyperlinks or sinks up to the point that the request became initiated.

- **AMBA3 Advanced Peripheral Bus (AMBA3 APB):**CoreSight supports the AMBA 3 APB protocol to allow transfer extension using wait states. The Debug APB bus makes use of the AMBA 3 APB protocol within a CoreSight machine. The Debug APB is a bus dedicated to the relationship of debugging and hint components in a CoreSight-compliant SoC. All CoreSight additives are configured and accessed over this bus via the APB-Mux in the DAP.

- **Advanced High-performance Bus (AHB):** CoreSight helps get admission to to a gadget bus infrastructure using the AHB Access Port

(AHB-AP)in the DAP. The AHB-AP provides an AHB grasp port for direct get admission to to machine memory. CoreSight additionally supports AHB bus tracing using an AHB Trace Macrocell (HTM) that provides non-invasive debug visibility to any bus transactions on AHB connections.

- **AMBA Advanced eXtensible Interface (AXI):** CoreSight supports the usage of AXI within the system interconnect. Direct get entry to to the AXI system can be supplied thru a Cortex center as an AXI bus grasp, or through the usage of an AHB to AXI bridge at the AHB Access Port within the DAP. CoreSight additionally supports hint technology from bus masters at the AXI via the use of the STM that converts stimulus writes to the device into a trace statistics flow.

5.3 CoreSight System Trace Macrocell

The STM abbreviated as System Trace Macrocell, which is typically a trace source that is incorporated into a CoreSight framework, structured principally for high transmission capacity trace of instrumentation implanted into programming. This instrumentation is comprised of memory-mapped writes into the STM Advanced extensible Interface (AXI) slave, which convey data about the behavior of system.The STM incorporated in the base component design is shifted to the Coresight component which has a support of this macrocell within in the top level architecture hierarchy of Coresight in infrastructure subsystem.The basic STM intergrated into base system is shown below,

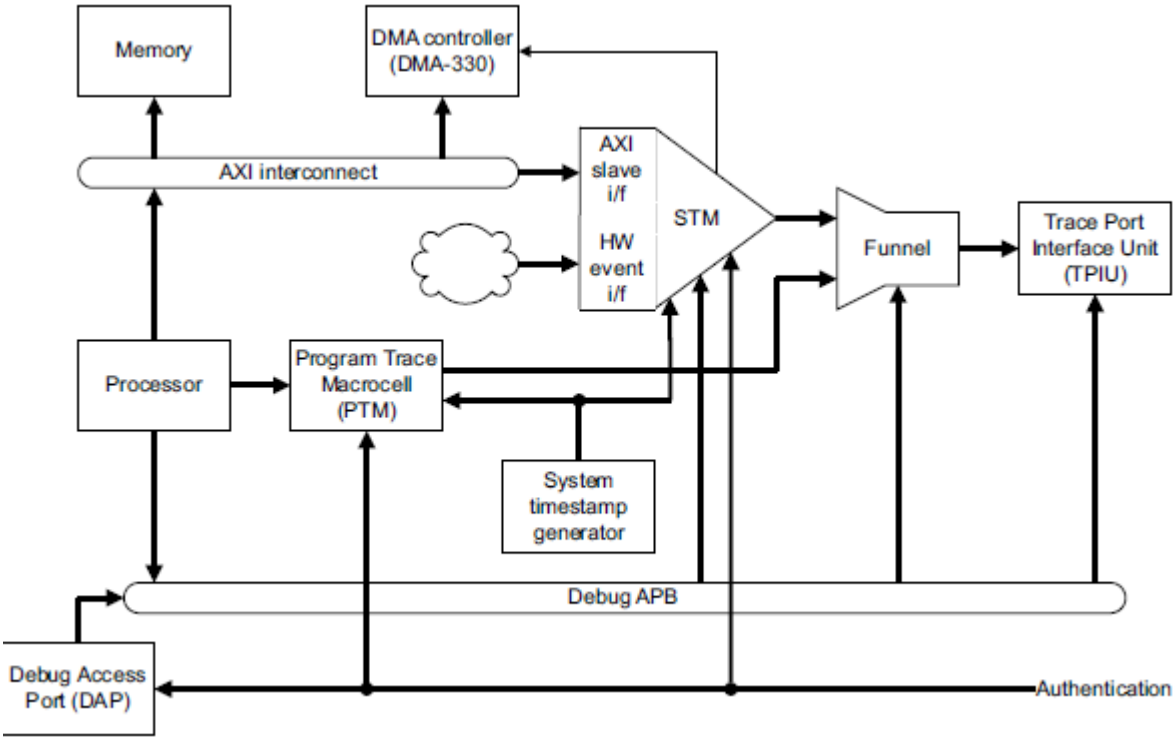

Figure 5.7: STM integrated into a typical system.

The STM AXI slave is connected to a system interconnect which enables all system masters,such as processors and DMA controllers, to generate trace by writing to the STM stimulus ports.For interaction with DMA controllers, the STM provides a DMA request interface compatible with the AMBA DMA Controller. [7]For configuration purposes, the STM is connected to Debug APB so that it can be accessed by off-chip and on-chip debug agents.CoreSight authentication signals are used to control debug permissions.The STM trace stream is output through the ATB interface and integrated with the rest of the CoreSight trace infrastructure. [7]

5.3.1 Interfaces

The interfaces of the STM which are to be connected with the top level of Coresight System hierarchy are listed below:

AXI Slave This interface connects the STM to the system bus.This design

provides a 32-bit AXI slave.

bfHardware event observation interfaceHardware events on this interface are captured and trace is generated based on captured events.This interface consists of 32 input signals, and connects to various signals from the system, like interrupt lines, DMA request lines, and Cross-Trigger Interface (CTI) trigger outputs.

DMA peripheral request interfaceconnects to an AMBA DMA Controller. When the STM is programmed to initiate a DMA transfer, this interface requests the DMA controller to write to the STM AXI.

Debug APB slave interfaceprovides access to the STM configuration and status registers.

ATB master interface This is the interface for trace output. It also provides handshaking signals for making flush requests to the STM.

Cross-trigger interface Three trigger output ports are implemented to connect to a cross-trigger interface in a CoreSight system, to indicate trigger events.

External synchronization interface where the synchronous reset input port enables an external component to control the frequency of periodic synchronisation. This signal is provided for compatibility with future architectures and can be tied low in most designs.

Clocking and resets The STM has a single clock input,which is synchronous to the system bus clock and must use asynchronous bridges when connecting the STM interfaces to differently-clocked buses.the STM also implements architectural clock gating.The STM has two asynchronous active LOW resets:

ARESETn This is used to reset the AXI slave and DMA peripheral request blocks.

STMRESETnThis resets the rest of the STM, including the hardware event observation interface, APB interface register file, and the Trace Generation Unit(TGU).

Hence understanding the complete flow and hierarchy of Arm Coresight IP which is dedicated for debug and trace and all the IPs in the SoC design will utilize the CoreSight IP for debug which will save a lot of resources, gate counts and LUT's.

Chapter 6

Functional Validation of Memory Compilers

6.1 Summary for the Work done

6.1.1 Memory compilers overview

Memory compilers are kind combined architectures of single srams all together and have flexible configurations with larger storage.The configurable sram architecture are implemented in industries in larger scales. Each of the sram design which is in the form of small cells are holding each single bit of memory compilers and these small cells combined together in form of two dimensional arrays.

From the memory compilers one can understand that the each sram memory cell is considered to be bit cell and thing that is noticeable here is the easy scalability of the structure. If larger size memory is needed it is achieved by creating larger array. With new columns periphery is added, for new row - decoder is expanded. While expanding the cell increases SRAM size and storage the components used remain the same. This means that designing new memories of different sizes from existing components can be automated.Below is the figure depicting the basic arrangements of bit cells in sram architecture.

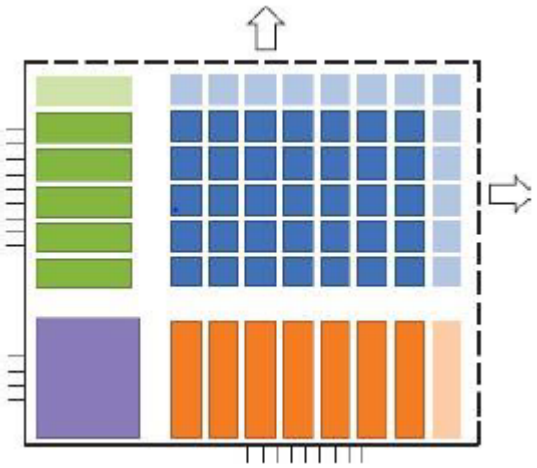

Figure 6.1: Memory array arrangement

After understanding the functionality of sram architecture and memory compilers, there are various methods to arrange this cells so as to reduce space and power as well as performance known as multiplexing which also reduces size of decoder increases. For multiplexing the bit cell array.Another approach is banking, when memory array is divided into separate banks which are connected to the same decoder through buffers amplifying decoder signals. As with the multiplexing, banking helps saving memory space by reusing the same decoder for larger array.

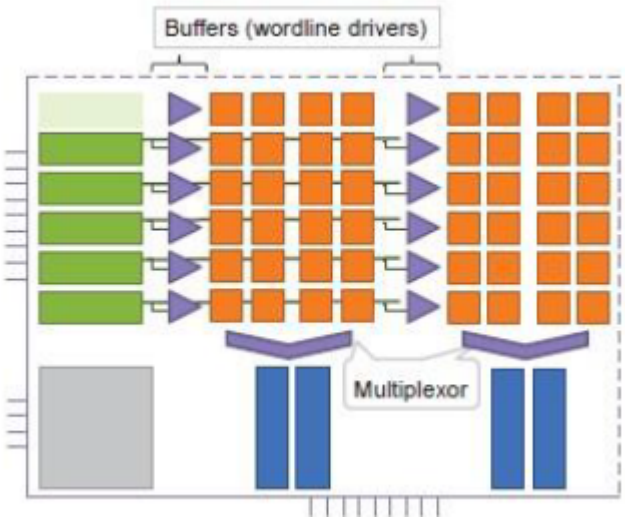

Figure 6.2: Memory banking and multiplexing

As for the structure of the compiler, separate generator (assembler) structures are used which take as input architecture templates together with memory design components and generate views.The main goal is for development was providing software capable of compiling different types of SRAM with different parameters, being free from intellectual Property restrictions.Below the steps for each views.

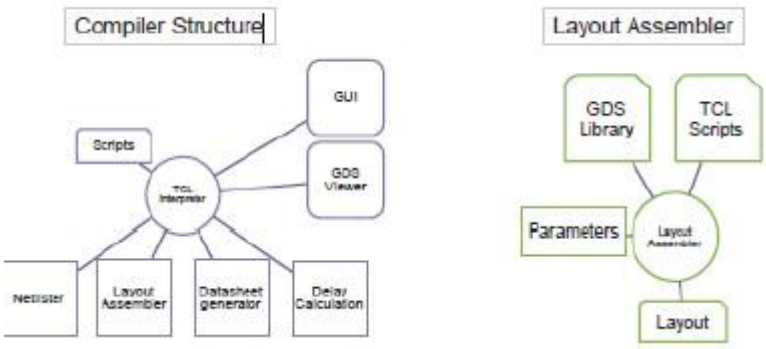

Figure 6.3: Memory banking and multiplexing

6.1.2 Work done

For writing the testbench for the memory compilers, after understanding the functionality of each and every pins of sram architecture, to check those functionality each and every pins is to be tested in such a way for timing violation that for one pin very exhaustively and maximum functional coverage. For that numerous testbench approaches has been used such as linear testbench approach where linearly the input stimulus is given to check for each test vectors.But this is a very basic way of writing and consumes more space.The second approach was writing the testbench in linear random fashion,but the disadvantage was to that it should have readability and covering maximum number of test vectors. But if these vectors are random then they may or may not generate same number of input stimuli. Also,third approach could be writing the self checking testbenches bit it requires more of an extra efforts but it will reduce the debugging time by providing the practical scene for testing the

Design under Test(DUT) which generates the output samples and compares it with the golden one which should be the expected output.Based on that one can compare the results and declare for passing and failing scenarios.Among that that best approach for the readability and compilation becomes fast was the approach of task and function based testbench.Consider for an example:

```
task read(input integer address,output integer data);
begin
@(posedge clock);
read_write = 0;
address = $random;
// Do some operation to get data
end
endtask
```

Thus the final conclusion was writing testbench for every pin event scene, task and function approach was better which gave us more coverage compared to other approaches and functional coverage also increased.

Chapter 7

Conclusion

7.1 Results And Snapshots

Below is data metric table which showcases the overall achievement in reducing the overall area, gate-counts and LUT's which also increase in the performance.Since system is verified in a test environment suite by applying regression methodology for parallelism in verification methodology. One can conclude that improvement in the data-metrics available from the emulator logs after doing the emulation for the system.

Base component	Gate counts	LUT's	Domain used
base element	35845223254	25644845	220000
Optimized base_element	35541254761	24100240	221658

Figure 7.1: Data metric count for the optimization

From the above data metric we have covered with almost 15 to 20 percentage of improvement compared to the older component.This above data have been collected from the emulation logs after the emulation has been running successfully.

7.2 Conclusion

Based on the the above work showcased, working for two different teams for SoC integration and Memory modelling the end experience was learning on system level as well as IP level.Since IP Optimization is much required for saving the resource and chip area in silicon when integrated in system environment. Major contribution to this project was reduction in power, area and speed which is viewed after the emulation logs.On wider perspective ARM has support from different emulator vendors,validation was done on different emulators to get consistency in results for overall system.Thus one major conclusion was understanding the complete automation workflow for system integration and implementing the changes in component level which is not only validated on IP level but on the complete environment of system on chip which gives us the broader picture to detect more bugs at higher level.Also, writing the testbench for memory compiler in the automated environment which gave more readability to the team for debugging test failures easily and achieve more functional coverage for the n numbers of compilers as the code is generic with different configuration of memory compilers.

References

[1] "System validation at arm," April 2016. [Online]. Available: https://community.arm.com

[2] "Arm internal documents."

[3] K. Wong, "Smarter route optimization for mobile ip," *16th international symposium*, 2015. [Online]. Available: http://ieeexplore.ieee.org.elibrary.nirmauni.ac.in/stamp/stamp.jsp?tp=&arnumber=1651704

[4] "Amba axi and ace protocol specification," August 2013. [Online]. Available: http://infocenter.arm.com/help/index.jsp

[5] "Coresight technical introduction," August 2013. [Online]. Available: http://infocenter.arm.com/help/index.jsp

[6] "Coresight technical introduction," August 2013. [Online]. Available: http://infocenter.arm.com/help/index.jsp

[7] "Coresight system trace macrocell," August 2013. [Online]. Available: http://infocenter.arm.com/help/index.jsp

Optimization of Component IP of an Infrastructure Subsystem

ORIGINALITY REPORT

9%	7%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	community.arm.com Internet Source	2%
2	www.boardcon.com Internet Source	2%
3	R. Goldman, K. Bartleson, T. Wood, V. Melikyan, E. Babayan. "Synopsys' Educational Generic Memory Compiler", 10th European Workshop on Microelectronics Education (EWME), 2014 Publication	2%
4	mobile.arm.com Internet Source	1%
5	www.ariat.hk Internet Source	1%
6	testbench.in Internet Source	<1%
7	Brendan Horan. "Chapter 9 Serial Server", Springer Nature, 2013 Publication	<1%

8	M. Siraj Rathore, Markus Hidell, Peter Sjodin. "PC-based Router Virtualization with Hardware Support", 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, 2012 Publication	<1%

Exclude quotes	On	Exclude matches	< 6 words
Exclude bibliography	On		