CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Statistics Monitor Design for Data Flow and Performance Analysis of an AMBA-Based SoC System

Master's thesis in Embedded Electronic System Design

Rongpeng Zheng
Vinaykumar Maramahalli Kemparaju

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Statistics Monitor Design for Data Flow and Performance Analysis of an AMBA-Based SoC System

Rongpeng Zheng
Vinaykumar Maramahalli Kemparaju

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Statistics Monitor Design for Data Flow and Performance Analysis of an AMBA-Based SoC System
Rongpeng Zheng, Vinaykumar Maramahalli Kemparaju

Supervisor:
Per Larsson-Edefors, Department of Computer Science and Engineering

Industrial Supervisor:
Faruk Sande, Ericsson

Technical Supporting Team:
Jing Li, Ericsson
Moazzam Fareed Niazi, Ericsson

Examiner:
Lena Peterson, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2022

Statistics Monitor Design for Data Flow and Performance Analysis of an AMBA-Based SoC System

Rongpeng Zheng, Vinaykumar Maramahalli Kemparaju
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Today's advanced system-on-chip (SoC) contains multiple intellectual properties (IPs) and technology with billions and billions of transistors all packed in an ultra-small form factor. All of it needs to perform flawlessly meeting demanding power and performance goals on tight schedules. Hence the complexity of SoC is sharply increasing. However, the performance of the system is not scaling linearly with the number of gate count. Henceforth, understanding the internal, dynamic behavior and having a constructive utilization of resources is critical in SoC design. In this thesis, we present a statistics monitor which is capable of monitoring data flow and performance metrics of AMBA-based SoC systems. The study considers different performance parameters such as system-level throughput, latency, bus efficiency, etc. The statistics monitor outputs such statistics data. The data obtained from the monitor unit provide insights into the SoC design, by assisting in the detection of performance bottleneck of the system.

Keywords: AMBA, SoC, Statistics Monitor, Performance Analysis, Data Flow, AXI, APB, AXI Interconnect.

Acknowledgements

We appreciate our supervisor, Professor Per Larsson-Edefors as well as our examiner, Professor Lena Peterson for their guidance and reviewing of this master thesis. We would like to appreciate our supervisor at Ericsson, Faruk Sande, for proposing this thesis and providing professional technical suggestions throughout the process to guide us through the completion of the project. We also want to thank our line manager at Ericsson, Tomas Larsson for his administrative support. We would also like to thank the technical supporting team, Jing Li and Moazzam Fareed Niazi for providing technical suggestions and introduction of EDA tools.

Rongpeng Zheng, Vinaykumar Maramahalli Kemparaju, Lund, November 2022

Contents

Acronyms

AHB AMBA advanced high-performance bus. 11
AHB-Lite AMBA advanced high-performance bus lite. 11
AMBA advanced microcontroller bus architecture. 1, 2, 10, 16
APB advanced peripheral bus. 2, 10–12, 23, 31, 54, 55
APB M IF APB manager interface. 18
APB Sub IF APB subordinate interface. 4, 11, 13, 15, 16, 18–22, 24, 40
AR channel read address channel. 8, 23, 29, 30, 32, 48, 50, 54
ASIC application-specific integrated circuit. 1, 2, 16
AW channel write address channel. 8, 23, 29–32, 54
AXI advanced eXtensible interface. 2–5, 7, 10, 11, 15, 16, 18, 21–24, 26, 29–32, 35, 37–43, 49, 53–56
AXI Manager IF AXI manager interface. 3, 8, 13, 14, 16, 20, 21, 36, 37
AXI NIC AXI network interconnect. 2–4, 11, 13–15, 18, 20–22, 24, 29, 31, 32, 36–38, 54–56
AXI Sub IF AXI subordinate interface. 8, 18
AXI VIP AXI verification intellectual property. 5
AXI4-Lite advanced extensible interface lite. 11

B channel write response channel. 8, 23, 29–32

DECERR decode error. 30
DFE digital front-end. 1, 2, 16
DP data processing unit. 35, 36, 38, 41, 42
DP/SW data processor or software. 1, 3, 4, 14–16, 18–22, 24–26, 30, 35, 54, 55

EDA electronic design automation. 18
EXOKAY exclusive access okay. 30

FPGA field-programmable gate array. 2

IP intellectual property. 1–3, 14, 20, 56

M IP manager intellectual property. 2, 3, 13, 14, 16–20, 23, 26, 29–31, 43, 44, 54–56

OKAY normal access okay. 30

QoS quality of service. 55, 56

R channel read data channel. 8, 23, 29–32, 40, 42, 48, 50, 51
RTL register-transfer level. 3–5, 16, 24, 32, 49, 53

SLVERR slave error. 30
SM statistics monitor. 17–19, 23, 25, 35, 36
SM_MM main monitor of the statistics monitor. 18, 21, 24–27, 35–42, 54
SM_SUB sub-monitors of the statistics monitor. 25, 26, 35, 38, 41, 54
SM_SUB0 sub-monitor 0 of the statistics monitor. 37, 39, 40
SM_SUB1 sub-monitor 1 of the statistics monitor. 37, 39, 40
SoC system-on-chip. 1–5, 8, 11, 13–20, 25, 31, 53–56
Sub IP subordinate intellectual property. 2, 3, 17, 18, 29, 55, 56

W channel write data channel. 8, 23, 30–32

1

Introduction

The semiconductor industry commends system-on-a-chip (SoC), as it is a way to combine all major functional elements such as a microprocessor, DSP, ASIC, memory, interfaces, peripherals, and analog components needed for a finished product onto a single die [1] [2]. Unifying all these components on a single chip provides the best possible outcome in a die size, unit cost, form factor, power, performance, and reliability [3]. In a race to fit as many functionalities on to a single chip, electronic components have overpopulated the chip area [4].

As the number of transistors in system-on-chip (SoC) increases, their complexity increases [5]. For effective SoC design, it is critical to understand the behavior of the internal SoC interactions. On-chip communication architecture accommodates all inter-component communication in an SoC (for example advanced microcontroller bus architecture (AMBA), Wishbone, Qsys Interconnect, CoreConnect). It allows multiple manager intellectual propertys (IPs) and multiple subordinate IPs to communicate with each other. The communication backbone of the SoC is being strained as application complexity grows. In modern SoCs, performance and power consumption are heavily influenced by the on-chip system bus communication architecture. Communication delay between two prime components on chip will result in a major bottleneck in the chip design [6]. Selection of suitable interconnect structure for a hard real-time system is the most complex task. Debugging system behavior at the board level, such as probing signals with logic analyzers, is impossible.

To debug internal system behavior, performance monitors are introduced inside the SoC. The purpose of the performance monitor is to measure clock accurate information of performance metrics like system-level throughput, latency, bus efficiency, etc. There are different types of performance monitoring systems. Generally, performance monitoring systems can be classified into two categories: software- and hardware-based systems. The software-based monitor provides high-level information (instruction level or above). Normally software monitors are supplemented by underlying hardware support [7]. It is safe to use software monitors in systems with soft deadlines especially if the system is not operating at full capacity else it has a major impact on system behavior [7]. In contrast, hardware-based monitoring fetches low-level information (clock accurate information) without affecting system behavior and timing [8]. The thesis presents hardware-based monitoring system. The monitor watches, gathers, stores, and outputs data to the the data processor or software (DP/SW) of the application-specific integrated circuit (ASIC).

This thesis study is carried out for a Ericsson's modern digital front-end (DFE)

ASIC digital radio. There are important parts in Ericsson's DFE ASICs that occupy a considerable area on the chip. Roughly ranging from two to five percent of the chip area, there are many such blocks in Ericsson's current ASICs. advanced eXtensible interface (AXI) Interconnect is the backbone of connectivity between various IP subsystems. AXI is a dominant protocol in the interactions between CPU subsystem, PCIe, CPRI, manager IP subsystems, and multiple subordinate subsystems. Additionally, there are other AMBA-based protocols used in the design such as the advanced peripheral bus (APB) protocol. The exploration in this thesis is carried out in accordance with design and document specifications of Ericsson's DFE. The hardware implementation of the statistics monitor is devised in SystemVerilog, verified on Cadence Incisive tools and synthesized using Design Compiler.

In this report, we present the statistics monitor unit designed for monitoring the performance and data flow of on-chip communication architecture. The statistics monitor is intended to embed inside *Ericsson's DFE ASICs*. The statistics monitor is capable of monitoring each transaction flowing through the interconnect structure in the SoC and uses the monitored data to analyze and validate the performance of the system. The analyzed results consist of information related to system bus latency, throughput, bandwidth, bus efficiency, and error rate. Studying these results will provide insights in selecting optimal constraints for designing interconnect such as arbiters scheme, clock speed for managers and subordinates, nodes, buffers, size, frequency converters or even moving the position of the block in a floorplan. The most likely outcome of the result is not to reconfigure the entire system, but rather to highlight the areas where tiny modifications have the greatest impact on the whole system's performance.

1.1 Related Works

Xilinx has developed an AXI performance-monitor IP which focuses on monitoring up to eight AXI interfaces for the AMBA AXI system [9]. Kyung et al. designed and implemented a performance analysis unit for AXI-based SoC [10]. These two monitors have different architectures and different definitions of performance metrics. However, these two monitors only consider a simplified AXI network interconnect (AXI NIC) of one level and implement their design on the field-programmable gate array (FPGA).

In actual ASIC SoCs, the top AXI NIC normally consists of multiple levels of AXI NIC components instead of only one level. For example, if one manager intellectual property (M IP) want to access a subordinate intellectual property (Sub IP), its transactions may go through several levels of AXI NIC to reach the destination Sub IP. The one-level AXI NIC enables these two monitors can simply be placed adjacent to the one-level AXI NIC, ignoring the long latency caused by the long wire connection. Moreover, M IPs and Sub IPs are placed at different locations of the floorplan and routing is very restricted in ASIC. For FPGA, it is easy to connect one interface in one specific location with an integrated performance monitor by direct wire connection since FPGA owns an abundant routing resource while ASIC needs to route carefully and do not allow an integrated performance monitor to directly

connect all IPs by wires.

Xilinx's AXI performance monitor and performance analysis unit from Kyung et al. define some normal performance metrics for the AXI manager interface (AXI Manager IF) of M IPs such as transaction count, latency, byte count etc. Some statistics in this thesis are referring to the definition from these two monitors [9] [10]. Kyung et al. propose some metrics about contention events to detect the latency casing by the AXI NIC. For example, transactions generated by several managers access the same subordinate at the same time. However, since the monitor needs to connect with the subordinate port inside the AXI NIC to get the contention metrics and there are multiple levels of AXI NIC and many Sub IP in SoC, it is unrealistic to obtain these contention metrics for our statistics monitor.

1.2 Thesis Goals

The primary goal of this thesis is to design a statistics monitor as a register-transfer level (RTL) solution in SystemVerilog from scratch for the AMBA-based Connect Subsystem. The statistics monitor will be placed at the manager interfaces of all M IPs in SoC to tap their AXI signals since all M IPs initiate the most transactions in the SoC. By tapping the AXI Manager IF of all M IPs, the statistics monitor not only can accumulate and calculate some performance metrics to know the actual performance of the whole connect subsystem, but it also can study the data flow from one specific M IP to all the Sub IPs of the SoC. When the statistics monitor logs all statistics into its memory after receiving a report signal from the DP/SW, the DP/SW will generate AXI read transactions to read all statistics from the statistics monitor to the DP/SW. The software will further process the statistics and generate a dashboard for a more intuitive visual display.

The desired functionality of the statistics monitor is as follows:

- Tapping AXI signals from the manager interfaces of all manager IPs in SoC
- Accumulating and calculating some actual performance statistics in rtl design solution. Those actual performance statistics would be such as round trip time, response latency, data amount, throughput, bus efficiency, error rates, etc. However, Since there is no multiplication and division IPs provided in the rtl design, some actual performance statistics would be accumulated as computational molecules which would be finally be further multiplied or divided by software.
- Storing all performance statistics into its register file.
- Comparing performance statistics from different monitor units in rtl hardware solution to get some comparison results about which tapped M IP has maximum or minimum performance such as max/min round trip time, max/min response latency, max/min throughput, etc.
- Interrupting the DP/SW to inform all statistics in the statistics monitor are ready to read.

- Having APB subordinate interface (APB Sub IF) as a subordinate IP in SoC so that the DP/SW can access it such as configuring its programmable control registers and reading statistics by AXI NIC.

1.3 Challenges

Before developing the statistics monitor, we may encounter some design and verification challenges as shown below:

Design challenges:

- **Microarchitecture**: There are a total of 16 manager IPs with AXI manager interfaces in the SoC, and these IPs are placed in different locations on the floorplan. It is impossible to only design one monitor for monitoring all manager IPs since all manager IPs are placed at different floorplan locations of the SoC and the monitor needs to be placed next to all manager IPs. Thus, one of the challenges is to design the microarchitecture of the statistics monitor with a communication mechanism between the DP/SW and the monitors at all the manager IP interfaces. The communication mechanism includes the data flow of control signals, an interrupt signal and statistics by which the communication protocol interface between the DP/SW and all monitors.
- **Design Requirements**: There are only a few papers, [5] and [8], related to SoC performance monitoring related to AXI, and their parameters are relatively simple, which cannot provide the detailed performance analysis of the entire SoC and analyse the performance bottlenecks of the data transferring. It is necessary to discuss with the Ericsson team about the application of the statistics and make a list of relevant statistics from scratch to confirm the design requirements.
- **RTL Design**: The AXI protocol is a high-performance and complex communication protocol. There are many protocol specifications designed to improve throughput and performance, such as multiple outstanding addresses, out-of-order transaction completion, etc. How to parse the complex AXI protocol and determine the RTL design solution according to the design requirements is a big challenge. In addition, the statistics monitor should not use multipliers and dividers to save area and reduce the design complexity since there are no such two existing IPs in company for the RTL design and these two mathematical operations in hardware consume much area and take more clock cycles to finish.
- **Memory**: Determining which kinds of memory for the statistics monitor and how to intelligently log in the statistics into memory over a period of time are also two design challenges.
- **Area and Power**: The net-list design should try to occupy small area and consume low power. It is necessary to apply some low power design techniques at the RTL level and use some methods to save resource such as reusing counters, reusing registers, using fewer-bit compactors, reducing unnecessary data width, etc.

- **Clock Frequency**: Since manager IPs are running at 491.00 MHz, the statistics monitor also needs to run at the same high-speed clock frequency so the RTL design should meet the high-speed timing constraint, increasing the design complexity.

Verification challenges:

- **Too many possible AXI stimulus**: It is challenging to make a comprehensive verification plan to verify the statistics monitor at the block level and subsystem level verification since it is hard to generate comprehensive stimulus of AXI traffic for one manager IP interface and multiple manager IP interfaces. There are too many AXI stimulus such as outstanding read/write transactions, interleaving read transactions, out-of-order read/write completion transactions for a transaction of single data item or multiple data items, and so on.

1.4 Thesis Outline

This master thesis report is organized as follows:

- **Chapter 2 Technical Background**: It describes the technical background needed for the statistics monitor, including the AXI4 protocol, the AXI interconnect, the APB protocol and the AXI VIP.
- **Chapter 3 Method**: Some system considerations are necessary to be researched to bridge the gap between the thesis goal and the implementation of the statistics monitor. These considerations are about the monitor placement around the SoC, microarchitecture of the statistics monitor, memory and the reusable RTL code of the monitor unit.
- **Chapter 4 Implementation**: It presents the implementation details of the statistics monitor, including the microarchitecture and the data flow of the whole statistics monitor in the SoC, the block diagram description of the main monitor, sub-monitors and a list of the statistics.
- **Chapter 5 Verification**: It presents the verification details of the statistics monitor, which verifies the connectivity and functionality of the AXI controller. Later it presents the verification plan for verifying the monitor unit for different statistics like overall statistics and statistics per unit time.
- **Chapter 6 Results**: This chapter presents the output results obtained from the statistics monitor graphically on the dashboard. It also presents the application of dashboard.
- **Chapter 7 Conclusion**: This chapter summarizes the content of the thesis. It presents the features and limitations of the statistics monitor. It also lists some of the use cases in which statistics monitoring can be used.

2

Technical Background

This chapter provides a technical background on AMBA-based protcols such as AXI4 protocol, AXI interconnect and APB protocol. Each section introduces the basic concept and their key features which are used to carry out this master thesis.

2.1 AXI4 Protocol

AXI is a burst-based communication protocol which is a part of Advanced Microcontroller Bus Architecture (AMBA). It is suitable for high-performance, high-bandwidth and low-latency chip design [11].

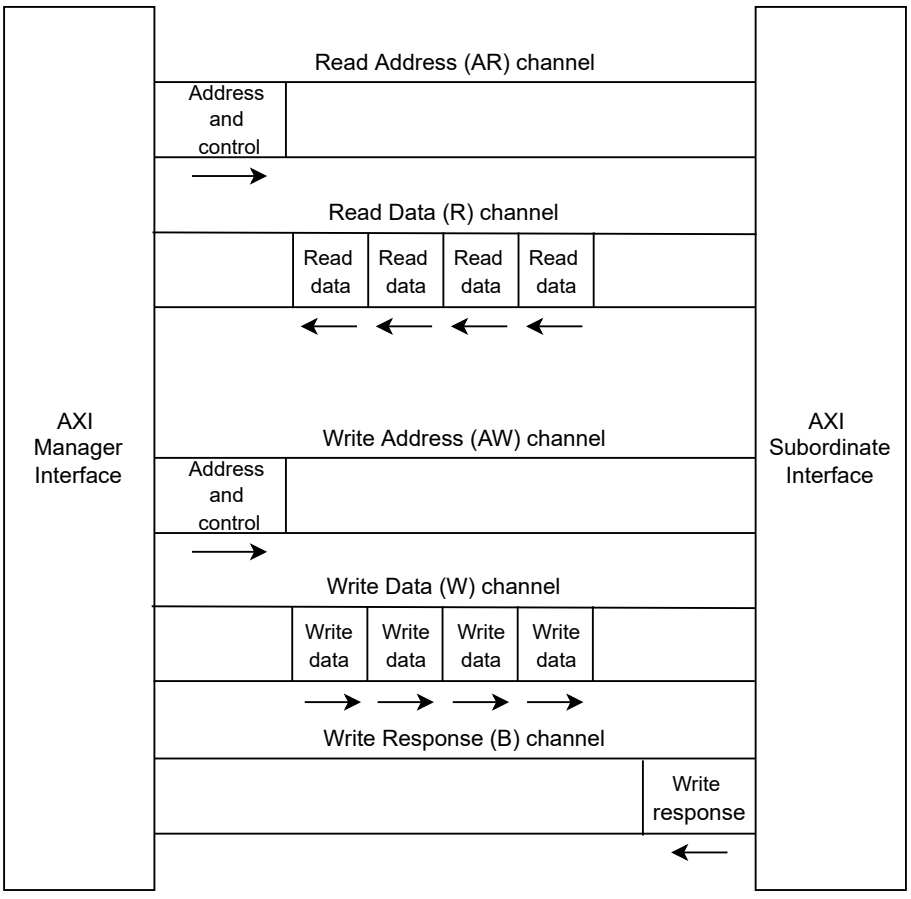

Figure 2.1: AXI channel architecture of reads and writes

There are five individual channels which are read address channel (AR channel), read data channel (R channel), write address channel (AW channel), write data channel (W channel) and write response channel (B channel) in the AXI protocol as shown in Fig. 2.1 [11]. AR, AW and W channels are driven by AXI Manager IF but R and B channels are driven by AXI subordinate interface (AXI Sub IF). All the channels confirm to exchange data by the handshake mechanism whose VALID means the current data is valid from the driver interface and READY means the current data can be received by the driven interface. On the one hand, when AXI Manager IF initiates a read transaction, it sends an address with burst length, burst size, burst type etc. on the AR channel to AXI Sub IF. Then, AXI Sub IF fetches read data items from the requested address to the AXI Manager Interface on the R channel. On the other hand, AXI Manager IF initiates a write transaction, it sends an address with burst length, burst size, burst type etc. on the AW channel and write data items on the W channel to AXI Sub IF. Once AXI Sub IF complete sending all the data items to the target address, it responds with a transaction state such as okay or error on the B channel to AXI Manager IF.

There are some key features improving bandwidth and latency of data transfers [12]:

- **Separate write and read channels**: This feature can simultaneously initiate write and read transactions so that write and read transactions do not block each other.
- **Multiple outstanding addresses**: The AXI manager can initiate multiple transactions on AW channel or AR channel in sequence and does not need to wait for the completion of earlier transactions from B channel or R channel. This feature can issue the write or read requests as soon as possible to make full use of the parallel processing of the system.
- **No strict timing relationship between address and data operations**: For example, when the AXI manager issues a write address on the AW channel, it can provide corresponding write data items on the W channel before or after issuing the write address without the timing requirement. Further, when the AXI manager issues a read address on the AW channel, it can receive corresponding read data items on the R channel only after issuing the read address without the timing requirement. This feature enables early issuing of the write or read request from the AXI manager considering the data transactions with a high priority and processing latency in the SoC.
- **Out-of-order transaction completion**: There are transactions identifiers in AXI so that the AXI transaction with different identifiers are possible to be completed out of order. For example, a fast-response transaction can complete before a slow-response transaction even if the fast-response transaction initiated by the AXI manager is slower than the slow-response transaction. This feature improves the throughput of an AXI manager because the completion of a fast-response transaction would not be blocked by a slow-response transaction.
- **Burst transactions based on start address**: When an AXI manager issues a burst-based transaction for writing or reading multiple data items, it only

needs to issue one address for the first data item rather than issue multiple addresses for all the data items. This feature helps an AXI manager save the number of issuing addresses and process the data items in batches.

2.2 AXI Interconnect

Since the AXI protocol is a point-to-point communication protocol which defines the signals with timing between managers and subordinates, an AXI interconnect in Fig. 2.2 is necessary to enable integration of all IPs for an SoC, involving multiple managers and subordinates [12].

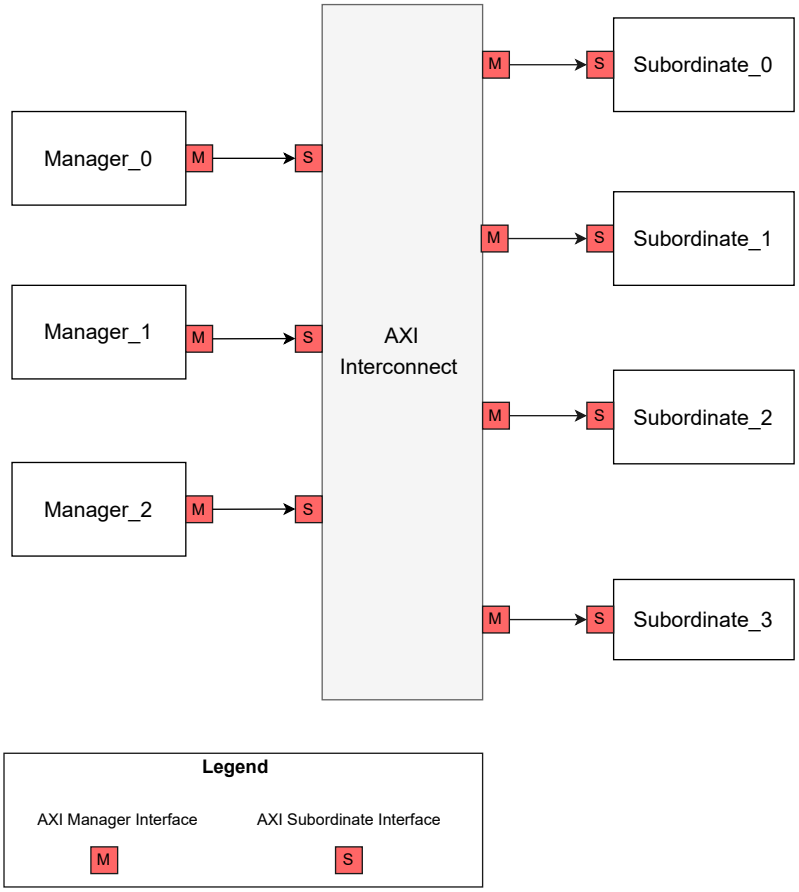

Figure 2.2: AXI Interconnect

The AXI interconnect is a full crossbar structure between managers and subordinates in Fig. 2.2 so that multiple managers can communicate with one subordinate or multiple subordinates at the same time [5]. An AXI interconnect has three main responsibilities as follows [13]:

- The AXI interconnect receives all transactions from managers or subordinates since it has AXI subordinate interfaces to connect with the managers and has AXI manager interfaces to connect with the subordinates.

- After receiving transactions, the AXI interconnect routes those transactions to their destination subordinates according to the issuing addresses on AW or AR channels.
- Once a manager or subordinate receives multiple transactions, there is an arbiter inside the AXI interconnect to organize the sequence of those transactions for resolving access contention.

Moreover, there are some other features supported in the AXI interconnect [12]:

- **Protocol Converters**: The AXI interconnect receives all transactions from managers or subordinates since it has AXI subordinate interfaces to connect with the managers and has AXI manager interfaces to connect with the subordinates.
- **Clock Converters**: Since different IPs inside the SoC may run on different clock frequencies, it is necessary to have clock converters to meet the requirements of clock domain crossings.
- **Bus-width Converters**: Since different IPs inside the SoC may have different bus widths, bus-width converters play a significant role in adjusting different bus widths from other managers or subordinates to a specific manager or subordinate for ensuring correct data transmission.

2.3 APB Protocol

The APB protocol is a communication protocol that is part of the AMBA protocol family. The objective of this protocol is to establish a standardized communication interface, which enables different entities to communicate with each other without having to deal with other entities present in a system, which supports modularity in system design. Unlike AXI, APB is a non-pipelined protocol used for communication with low-bandwidth peripheral which do not need high performance [14]. The APB protocol allows to access programmable control register of peripheral devices [14]. APB protocol is a low-cost interface hence it uses low area and power in the design. In the implementation of the statistics monitor, APB protocol is used to access register files.

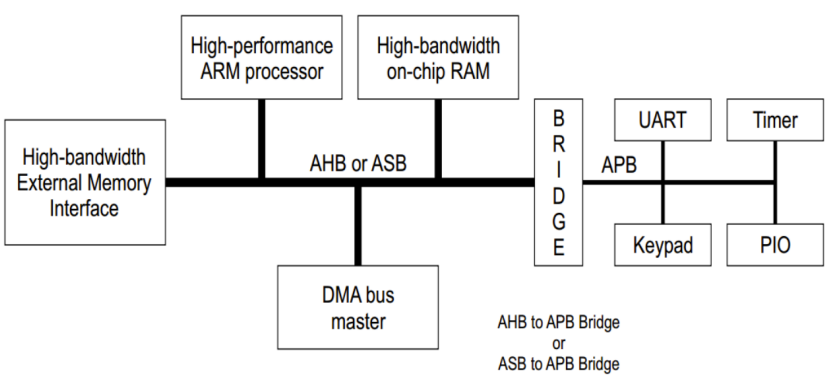

Figure 2.3: AMBA Bus including AHB/ASB and APB with APB connect to UART, Timer, Keypad and PIO slaves [15].

A typical APB-based interface in SoC can be seen in Fig. 2.3. APB protocol is compatible with AMBA advanced high-performance bus (AHB), AMBA advanced high-performance bus lite (AHB-Lite), AXI and advanced extensible interface lite (AXI4-Lite) [14]. AXI NIC has an APB bridge inside it as the APB manager controller to actively initiate transfer while APB Sub IF can only passively receive transfer. This bridge acts as communication interface between high performance bus(like AHB, ASB) and other peripheral devices.

2.3.1 AMBA APB Components and Interconnection

A typical AMBA APB design contains the following components:

APB Master

APB facilitates Write, Read, and Idle exchanges by providing an address and control signals. Due to the nature of APB protocol, only one master can access the bus at any time. Hence there is no need for an arbiter-like AXI protocol. The interface of an APB master is presented in Fig. 2.4. APB performs combinational decode to select the slave via PSELx pin. It is also responsible for driving PENABLE and PDATA onto the system bus.

APB Slave

The APB slave shown in Fig. 2.4 is very basic, yet flexible interface. APB slave responds to requests made from an active master interface. PWRITE decides whether the APB slave is in reading or write mode. When the PWRITE signal is set high, the APB slave is in write mode else in the read mode provided that PSELECT and PENABLE are set high. It is also capable of informing the master if the transaction was successful or not via PSLVERR pin.

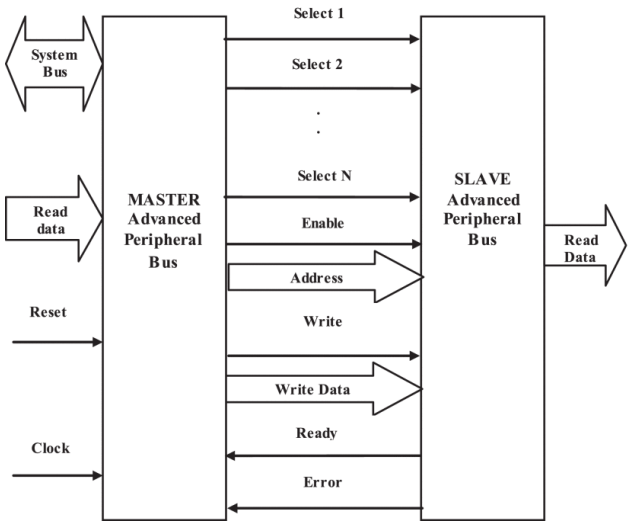

Figure 2.4: Interfacing of APB Master and Slave [16].

2.3.2 APB Protocol Operating Modes

Fig. 2.5 shows the different operating states of an APB protocol. It has three defined states Idle, Setup, and Access. The APB protocol takes at least two cycles to complete a transaction although it takes three cycles to complete the first transaction.

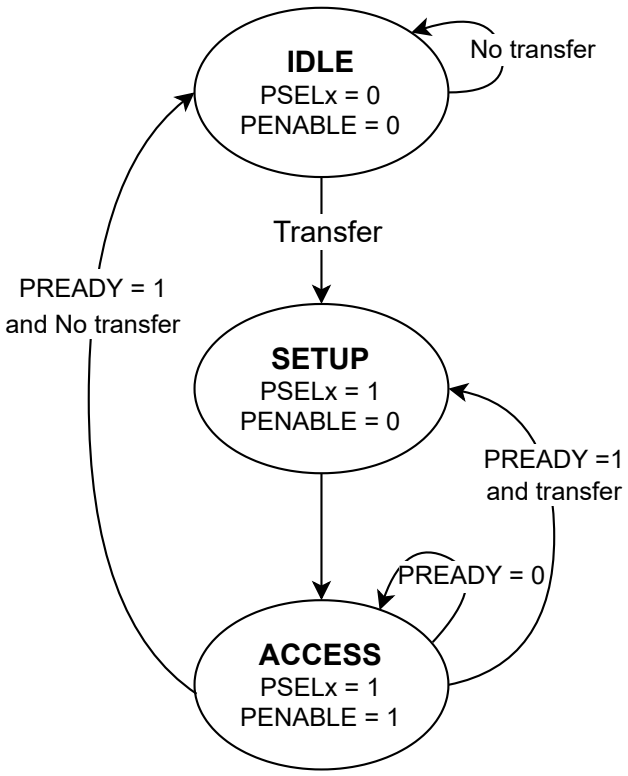

Figure 2.5: Operating states of APB protocol

- **Idle state**: This is a default state in which there will be no data transfer.
- **Setup state**: When the transaction starts PSELx is asserted. The bus moves into the setup state. In the next clock cycle, the bus unconditionally moves to the access state. If the bus remains in the setup phase in the next clock cycle it will run into protocol violation.
- **Access**: The enable signal, PENABLE is asserted in the ACCESS state. During the transition from setup to access the address, write and data signals should remain stable. Else, it will run into protocol violations. The bus remains in an access state until PREADY is zero. When the PREADY is asserted, the bus moves into either an idle state or the setup state. If there is no subsequent transaction the bus remains in the idle state. If there is another transaction the bus moves into a setup state.

3

System considerations

In this chapter, some system considerations are necessary to be researched to bridge the gap between the thesis goal and the implementation of the statistics monitor. These considerations are about the monitor placement around the SoC, microarchitecture of the statistics monitor, memory and the reusable RTL code of the monitor unit.

3.1 Monitor Placement

The main monitor and 15 sub-monitors are individually and strategically placed at all the manager IP interfaces, since manager IP is responsible for generating AXI traffic in the AXI interconnect system. Thus the placement of the monitor is coherent. The advantage of monitoring the Manager IP interface is that all the manager interfaces are using AXI protocol. Even if the subordinate is APB protocol, APB bridge inside the interconnect structure takes care of the protocol conversion. Therefore, all the monitors are AXI-based protocols. The SoC with the monitor placement is shown in Fig. 3.1.

3.2 Microarchitecture

The microarchitecture of the statistics monitor in Fig. 3.1 includes one main monitor and 15 sub-monitors with AXI Manager IF and APB Sub IFs, connected with the AXI NIC. It is impossible to design only one monitor using a direct wire link for monitoring all M IPs in the SoC. The reason is that 16 M IPs in the SoC are placed in different locations on the floorplan, and routing the connection between the only one monitor and 16 M IPs would cross the whole floorplan which is an extremely expensive operation in the SoC. Based on these two reasons, the statistics monitor is divided into 16 monitors, placed at AXI Manager IFs of all 16 M IPs in the SoC for close and immediate acquisition of AXI signals.

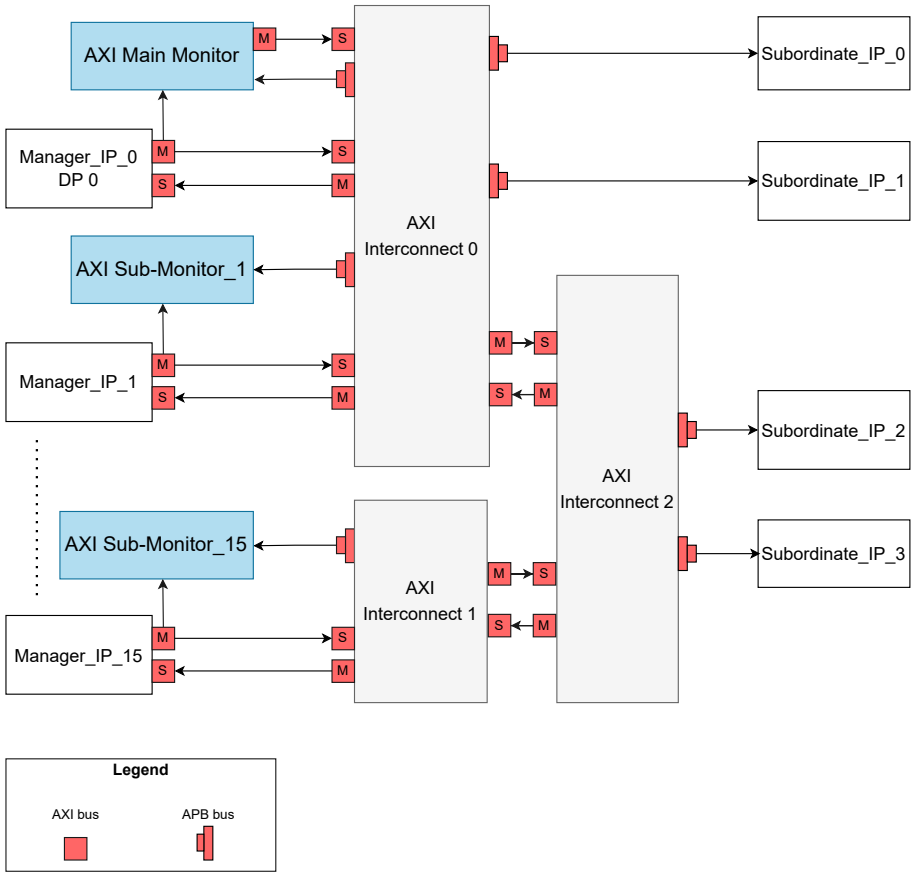

Figure 3.1: The microarchitecture of the statistics monitor for the SoC of 16 manager IPs

One of the 16 monitors is a main monitor and the remaining 15 are sub-monitors. The reason why the statistics monitor needs a main monitor is that the main monitor is a bridge of communication between a DP/SW and 15 sub-monitors. The main monitor is responsible for obtaining statistics from the neighbour M IP, broadcasting control signals from the DP/SW to sub-monitors, collecting statistics from sub-monitors, comparing results and reporting all statistics to the DP/SW. However, 15 sub-monitors does not have such many function as the main monitor but they are capable to obtain statistics from their neighbour M IPs and report their all statistics to the main monitor.

Since the distributed statistics monitor is divided into one main monitor and 15 sub-monitors, the communication mechanism is necessary for the communication between the DP/SW, the main monitor and 15 sub-monitors. It is based on the AXI NIC and protocol interfaces of the main monitor and 15 sub-monitors. Firstly, the AXI NIC is responsible for the communication between all IPs inside the SoC and routing the transactions to destination IP. Since the distributed statistics monitor is divided into 16 separate monitors which needs to communicate with the DP/SW or the main monitor by the AXI NIC. Secondly, the main monitor has two protocol interfaces. The first one is the AXI Manager IF which is used to actively broadcast control signals to all sub-monitors and read statistics from all sub-monitors. The

second protocol interface of the main monitor is APB Sub IF which communicates with the DP/SW for passively receiving control signals and sending all statistics. Thirdly, each sub-monitor have only one communication interface, which is the APB Sub IF. This interface serves two purposes. The first purpose is to receive the control signal from the main monitor and the other purpose is to passively send its statistics to the main monitor or the DP/SW. More detailed explanation about the microarchitecture and the communication mechanism will be given in section 4.1.

3.3 Memory

The memory of the statistics is the register file. There are some reasons as shown below:

- **Memory in the SoC**: Since the statistics monitor is designed for the current tape-out SoC which has allocated specific memory to its current IPs, there is no extra free memory in the SoC that can be allocated to a new IP of the statistics monitor.
- **Disadvantage of using outside memory**: If the statistics monitor wants to use memory other than itself, it needs to actively initiate transactions from time to time, which requires the main monitor and 15 sub-monitors to have AXI manager controllers, increasing the design complexity, area and power consumption. In addition, the fact that the data monitor actively initiates transactions from time to time can reduce the performance of the running AXI NIC. The statistics monitor should avoid affecting the normal data transmission. However, the register file inside the main monitor and sub-monitors can temporally store their statistics which avoid actively initiating transactions and affecting the AXI NIC from time to time.
- **Memory capacity requirement**: From the statistics parameters of the design, since the statistics are reported after accumulating for a period of time, the required storage capacity is not large, and the register file can provide suitable storage space without complex storage control.
- **APB Sub IF**: The register file only needs an APB Sub IF which can passively receive the read or write transactions from the DP/SW or the main monitor. Thus, it does not need a complex AXI manager controller to communicate with AXI NIC.
- **Various register types**: There are many register types in the register file. For example, read-only registers can store statistics and be read by the APB Sub IF; write-only registers allows the APB Sub IF to assert their values for control signals; interrupt register can generate an interrupt signal informing the DP/SW that the statistics monitor has prepared all statistics. Those various register types satisfy the design requirement of the statistic monitor.

3.4 AXI Monitor Unit

The AXI monitor unit is a configurable, reusable and flexible RTL design for tapping AXI signals and obtaining performance statistics. Firstly, the RTL of the monitor unit can configure its parameters such as burst length width, address width, id width to monitor different AXI Manager IFs. The configuration of the AXI monitor unit enables its RTL design to adjust any AXI Manager IFs. Secondly, the AXI monitor unit is also an reusable IP because it can be duplicated or instantiated as the number of M IPs in the SoC as the sub-monitors and perform the same monitor function. Thirdly, each sub-monitor, instantiated from the AXI monitor unit could be regarded as an individual subordinate IP and be individually accessed through its APB Sub IF by the DP/SW or the main monitor. The DP/SW is possible to only activate one sub-monitor and observe the performance of one M IP instead of activating all the monitors of the statistics monitor, so the AXI monitor unit is also flexible to be activated. Hence, the monitor unit will be compatible with future *Ericsson's DFE ASICs* and other SoC designed on an AMBA-based interconnect system with appropriate parameter configuration.

4

Implementation

In this chapter, the implementation of the statistics monitor is introduced. It includes the microatchitecture of the statistics monitor with the data flow explanation, the functional description of the main monitor and sub-monitors with their block diagrams and a list of statistics which would be obtained by the statistics monitor.

4.1 Overall View of a Simple SoC

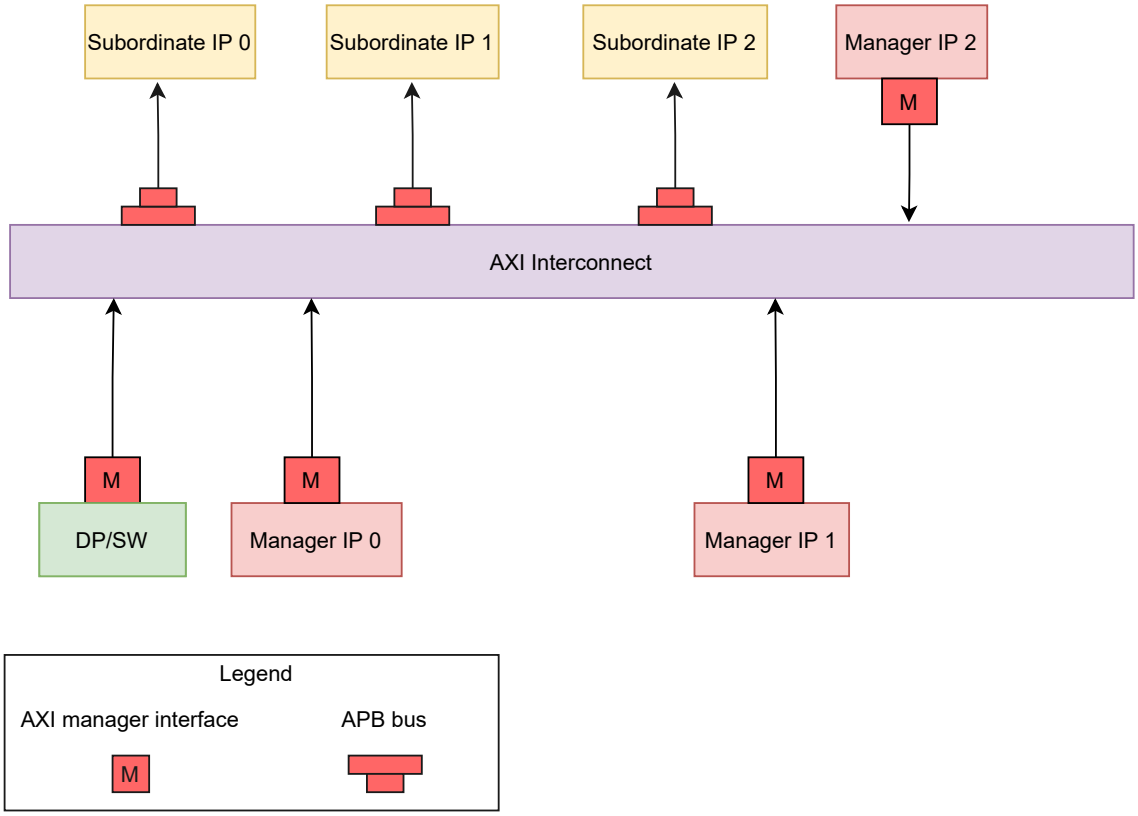

Figure 4.1: An simple SoC has 3 manager IPs and 3 subordinated IPs with AXI interconnect

Since the target SoC has at least 16 M IPs and many more than 16 Sub IPs, it is not necessary for the statistics monitor (SM) to monitor all 16 M IPs. Thus, the

master thesis proposes only monitor a simple SoC which has basic elements such as three M IPs and three Sub IPs with corresponding AXI NIC to demonstrate the data flow and working mechanism of the design. The simple SoC is shown Fig. 4.1

4.1.1 Microarchitecture of the Statistics Monitor

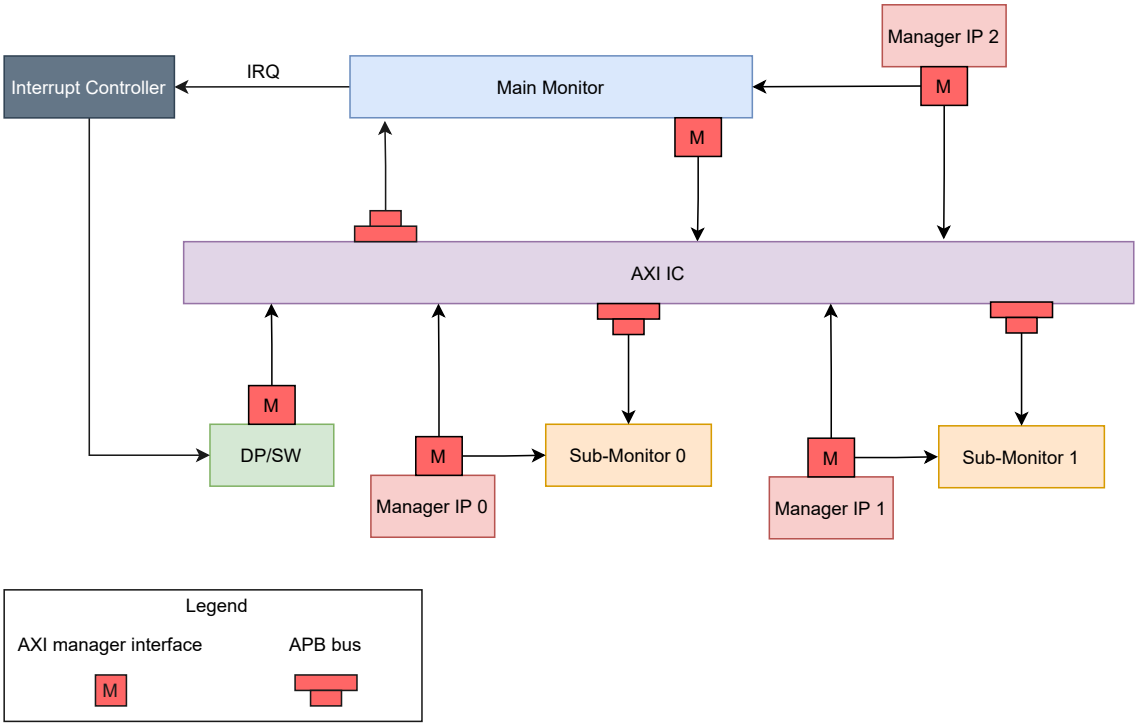

Figure 4.2: The microarchitecture of the statistics monitor inside the simple SoC

An SM is a hardware solution to monitor the AXI signals from three M IPs, connected to the AXI Interconnect in Fig. 4.1. Since M IPs are the originating sources of most transactions flowing through the AXI Interconnect inside SoC, an SM is only placed at the manager interface of three M IPs. An SM placed at the master interface tracks all the information going through the interface to calculate the performance and data flow statistics of the SoC.

The microarchitecture of the statistics monitor in the SoC without Sub IPs is shown in Fig. 4.2. It has one main monitor and two sub-monitors, which are connected with the AXI manager interfaces of three M IPs respectively while it also has its own AXI NIC which includes one AXI Sub IF for the DP/SW, one AXI Sub IF for main monitor of the statistics monitor (SM_MM), one APB manager interface (APB M IF) for SM_MM and one APB M IF for sub-monitor 0 and one APB M IF for sub-monitor 1. The paths in the AXI NIC are from the DP/SW to SM_MM and from SM_MM to all sub-monitors. This AXI NIC is generated by an electronic design automation (EDA) tool, ARM Socrates [17].

The *sub-monitor 0 and 1* in Fig. 4.2 are instantiated by one AXI monitor unit and one register file with APB Sub IF. A monitor unit is a generic RTL design for

tapping AXI signals and calculating performance statistics while a register file with its APB Sub IF could assign the control signal, store statistics and report those statistics to *Main Monitor*. According to tapped AXI signals, the AXI monitor unit will accumulate statistics which are explained in more detail in section 4.5.

Main Monitor in Fig. 4.2 is the highest level module of the statistics monitor which not only instantiates an AXI monitor unit for tapping the AXI signals from *Manager_IP_2* in Fig. 4.2, but also acts as the control centre of the statistics monitor between the DP/SW and two *Sub-Monitors*. The data flow of the control signal and statistics for the SM is shown in section 4.2. The DP/SW could only communicate with *Main Monitor* instead of all *Sub-Monitors* to control the whole SM. In brief, *Main Monitor* is able to monitor the performance of a M IP, broadcast the control signals to all sub-monitors, collect all statistics from all sub-monitors, obtain comparison results between different monitor units, generate an interrupt signal to trigger the DP/SW to read statistics from the *Main Monitor* and report all statistics to the DP/SW.

In the end, the DP/SW will use those statistics from the main monitor and compute averages, using multiplication and division, such as throughput, bus efficiency, average round trip time, etc. Since division and multiplication are resource consuming in hardware, the software helps with those arithmetic calculation operations. The final statistics will be used for generating a dashboard with statistics charts and figures for data visualization so that it would be easier to find the performance bottleneck of the SoC.

4.2 Communication Mechanism

Since the statistics monitor consists of one main-monitor and two sub-monitors, which are located in different floor plan near their tapping M IPs, the distributed statistics monitor need to communicate internally between the main monitor as well as two sub-monitors and communicate externally with the DP/SW. Fig. 4.3 and Fig. 4.4 respectively shows the data flow of control signals and statistics between the DP/SW, the main monitor and two sub-monitors. These two data flows are the communication mechanism of the DP/SW.

4.2.1 The Data Flow of the Control Signal

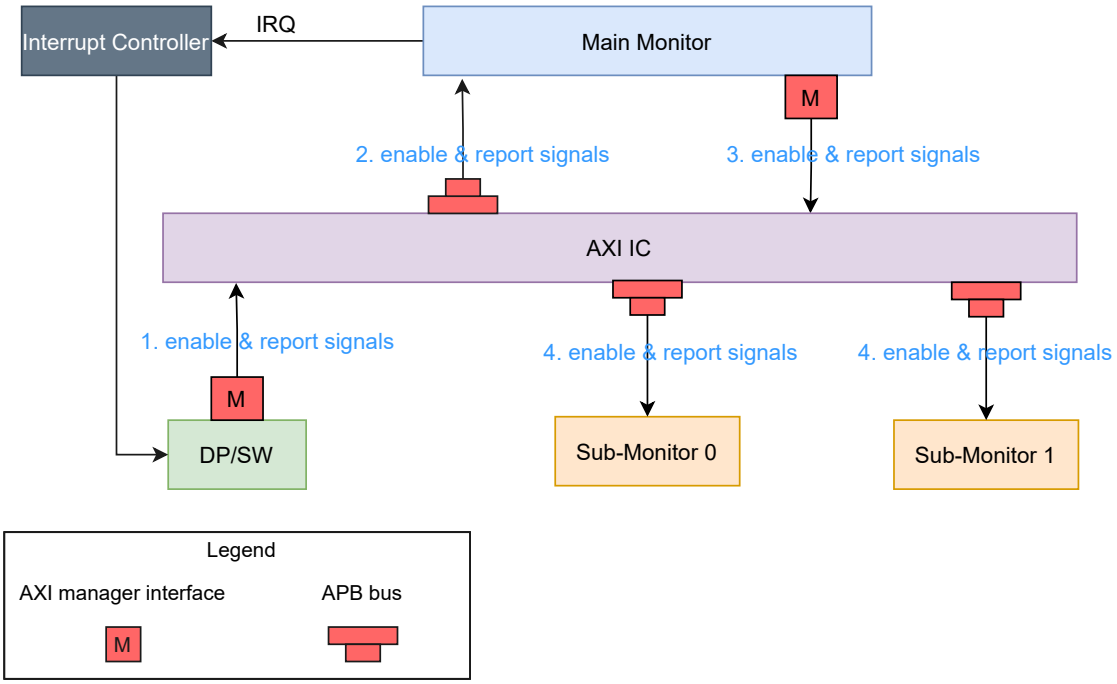

Figure 4.3: The data flow of the control signal in the statistics monitor

Inside the SoC, if an IP wants to communicate with the other IPs, its interfaces need to be connected with AXI NIC rather than directly linking to the destination IP. Thus, the DP/SW, the main monitor and sub-monitors all have their own interfaces for communication. The DP/SW uses its AXI Manager IF to initiate AXI write or read transactions to communicate with *Main Monitor* and *Sub-Monitors*. The main monitor communicates with the DP/SW for receiving the control signals and sending statistics by its APB Sub IF while it also communicates with *Sub-Monitors* for broadcasting control signals and reading statistics by its AXI Manager IF. Furthermore, each *sub-monitor* has an individual APB Sub IF to communicate with *Main Monitor* for receiving control signals and sending statistics.

The control signal is a 32-bit programmable register, including 30 bit for configuring time window scale, 1-bit report signal and 1-bit enable signal. The benefit of merging

these three signals into one 32-bit signal is that one AXI write transaction is enough to transfer a 32-bit data instead of initiating three transactions for assigning these three signals.

Regarding the control signal of enable and report in Fig. 4.3, if the enable signal is high but the report signal is low, *Main Monitor* and *Sub-Monitors* start monitoring and accumulating their statistics; if the enable signal is low but the report signal is high, *Main Monitor* and *Sub-Monitors* stop monitoring, and calculate some statistics requiring further processing and wait for read requests. In addition, the time window scale in the control signal is used to configure the time window scale for each sub window so that the DP/SW could determine what time scale for each sub window enables SM_MM to get a detailed and desirable performance analysis on a smaller time scale.

In Fig. 4.3, there are four steps for the data flow of the control signal:

1. The DP/SW initiates a write transaction from its AXI Manager IF to APB Sub IF of *Main Monitor* to assign the control signal for assigning the time window scale and asserting the enable or report signals high.

2. The control signal from the DP/SW goes through AXI NIC and finally reaches *Main Monitor*.

3. *Main Monitor* receives the control signal, and then broadcast the control signal to two *Sub-Monitors* by initiating two AXI write transactions to AXI NIC.

4. The *sub-monitor 0* and *sub-monitor 1* receive the control signal respectively from AXI NIC.

Moreover, after the DP/SW sends the report signal to *Main Monitor*, the DP/SW needs to initiate another write transaction to enable the interrupt signal in *Main Monitor*. Else the interrupt signal in *Main Monitor* would not be triggered by *Main Monitor* when *Main Monitor* has prepared all statistics for the DP/SW.

4.2.2 The Data Flow of the Statistics

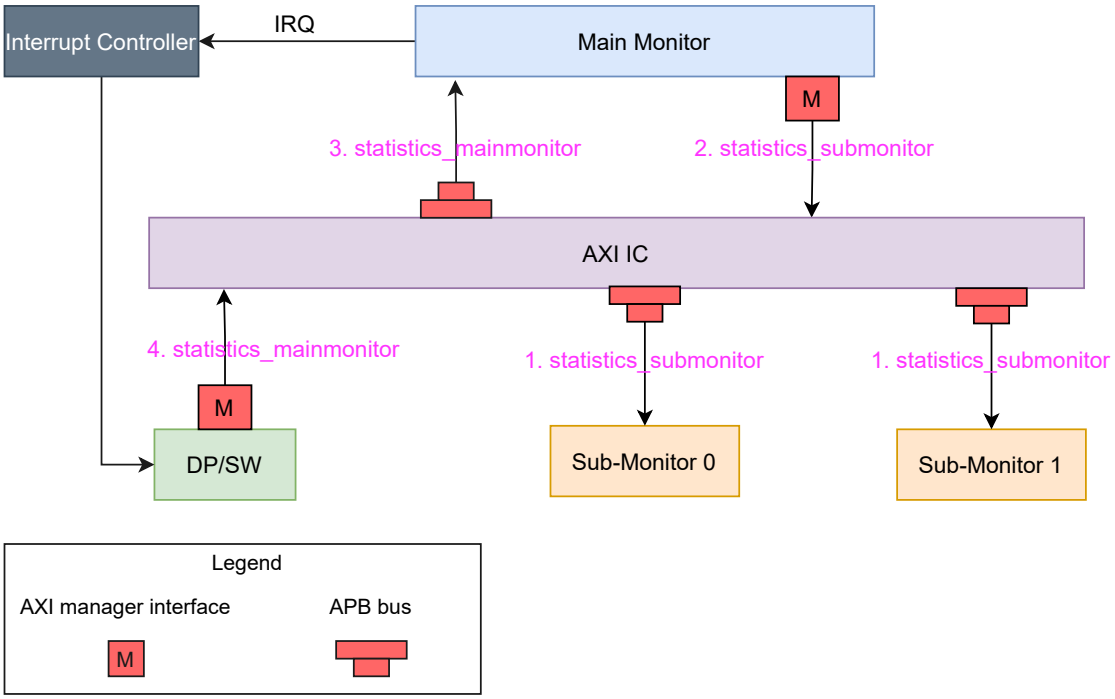

Figure 4.4: The data flow of statistics in the statistics monitor

There are some differences in quantity and kind for statistics inside *Main Monitor* and *Sub-Monitors*. More detailed statistics are given in section 4.6.

Regarding the statistics flow in Fig. 4.4, there are four steps for the data flow of the statistics:

1. Once *Main Monitor* gets a write response signal of sending the report signal to *sub-monitor 0*, it would initiate AXI read transactions to read all statistics from APB Sub IF of *sub-monitor 0*. Then, the statistics would be read out from *sub-monitor 0* to AXI NIC. The flow is also same for *sub-monitor 1*.
2. AXI manager interface of *Main Monitor* receives the statistics one by one from *sub-monitor 0* and *sub-monitor 1* by AXI NIC. Once *Main Monitor* finishes the statistics collection from all *Sub-Monitors*, it compares its statistics with all *Sub-Monitors*' statistics to get some comparison results. Then, it would generate an interrupt signal to the DP/SW for informing all statistics in *Main Monitor* are ready.
3. Once the DP/SW receives the interrupt signal, it would initiate an AXI write transaction to clear the interrupt signal and then initiates multiple AXI read transactions to read out all statistics in *Main Monitor*. Statistics from *Main Monitor* would be read from APB Sub IF of *Main Monitor* to AXI NIC.
4. AXI manager interface of the DP/SW finally receives the statistics one by one from *Main Monitor*. Those statistics could be used to draw some performance bottleneck conclusion and create a dashboard for data visualization.

4.3 Sub-monitors

Fig. 4.5 shows the block diagram of a sub-monitor (SM_SUB). It consists of a *Monitor Unit*, a *Register File* and an APB interface. *Monitor Unit* includes *AXI Watcher* module, *Statistics Calculator* module and *Time Window Trigger* module. When the statistics monitor (SM) want to monitor a new AXI M IP, *Monitor Unit* can be directly instanced and port mapped the AXI signals from the new AXI M IP while *Register File* only needs to modify its block address. Then, a new sub-monitor is easily created. The following is an introduction to the functions of each module in Fig. 4.5:

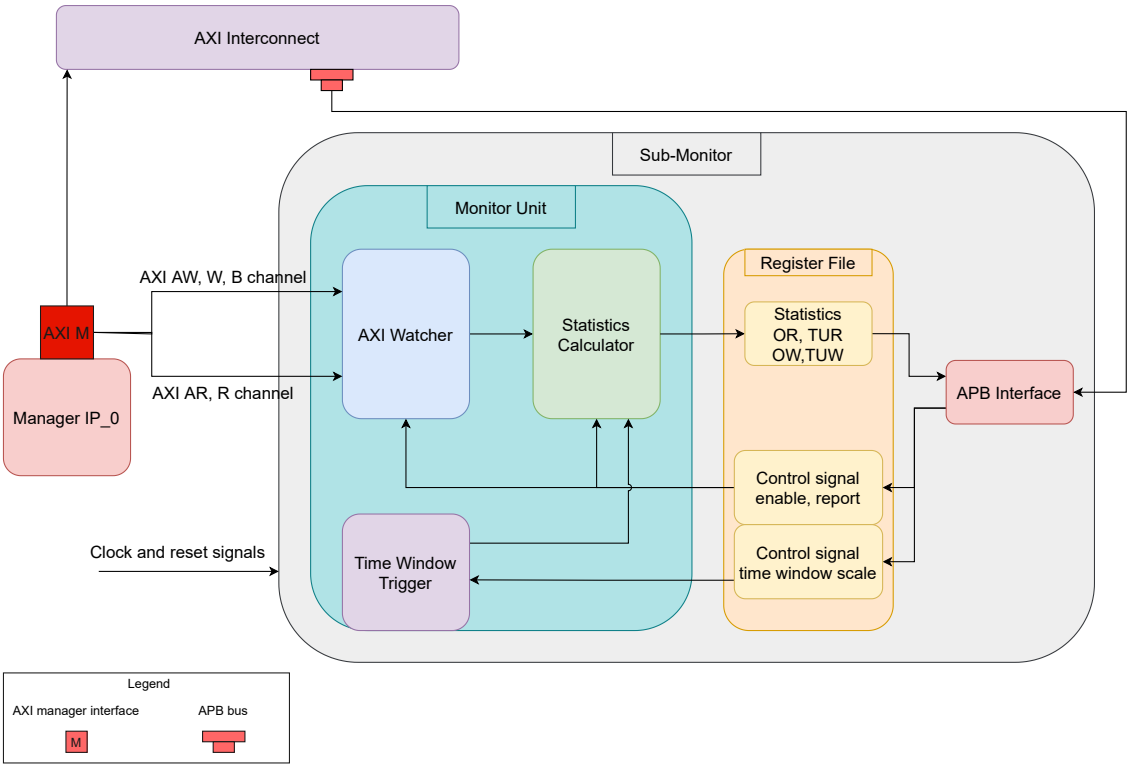

Figure 4.5: The block diagram of a sub-monitor

- *AXI Watcher* module: It taps the necessary AXI signals of each transaction from the manager interface of the nearby Manager IP and stores the AXI information in registers. This module also acts as one-clock-delay registers to ensure the timing closure. It can be triggered when the enable signal is high and the report signal is low in *Register File*. The necessary AXI signals are shown below:
  - AW channel: awid, awlen, awbrust, awvalid and awready.
  - W channel: wstrb, wlast, wvalid and wready.
  - B channel: bid, bresp, bvalid and bready.
  - AR channel: arid, arlen, arsize, arburst, arvalid and arready.
  - R channel: rid, rresp, rlast, rvalid and rready.

- *Statistics Calculator* module: Using the necessary AXI signal from *AXI Watcher* module, it calculates four kinds of the statistics: (1) Overall Read Statistics (OR) (2) Time Unit Read Statistics (TUR) (3) Overall Write Statistics (OW) (4) Time Unit Write Statistics (TUW). Overall statistics mean the statistics over the whole monitor window (e.g maximum 8 second) while time unit statistics mean the statistics over one window scale (e.g maximum 1 second). The RTL design has eight continuous time windows over the whole monitor window so that the statistics monitor (SM) could get more detailed performance data in different time windows. The list of statistics can be seen in section 4.5. The Statistics Calculator module starts calculation when the enable signal is high and the report signal is low and prepares reporting when the enable signal is low and the report signal is high. Preparing reporting means there are some statistics could be calculated by the accumulated statistics at the end of the whole monitor window so that it would save power instead of keep calculation when the accumulated statistics are updated over the whole monitor window. Moreover, *Statistics Calculator* module includes many individual RTL modules to calculate different statistics, so if some statistics are not necessary to be obtained, the *Statistics Calculator* module could just remove the corresponding statistics RTL instances.
- *Time Window Trigger* module: It is a counter based on the system clock to count how many clock cycles have been passed over the whole monitor window. In addition, according to the programmable control signal of time window scale, this module would generate a trigger for *Statistics Calculator* module to update Time Unit Read/Write Statistics to *Register File*.
- *Register File* module: It is a set of 32-bit registers, including read-only-type registers and a write-only-type register. The read-only-type registers store the OR, OW, TUR and TUW statistics while the write-only-type register includes 30-bit time window scale signal, 1-bit enable signal and 1-bit report signal.
- *APB Interface* module: It is the output interface of the sub-monitor which is connected with AXI NIC outside the sub-monitor. It receives the broadcasting control signal from SM_MM and reports its statistics to SM_MM. Furthermore, since the sub-monitor uses APB Sub IF, the standard protocol, any the DP/SW port is possible to individually access the sub-monitor by AXI NIC when the DP/SW port has been set the path to the sub-monitor in AXI NIC.

4.4 Main Monitor

The main monitor is the control centre of the statistics monitor, since it communicates with the statistics monitor (SM) and sub-monitors of the statistics monitors (SM_SUBs). On the one hand, SM_MM receives the control signal as well as the interrupt signal of enabling and clearing from the DP/SW and reports all its statistics to the DP/SW. On the other hand, SM_MM broadcasts the control signal to all SM_SUBs and receives all statistics from all SM_SUBs. In this method, the DP/SW is able to communicate with the statistics monitor (SM) by initiating fewer transactions only through SM_MM instead of initiating many more transactions through all SM_SUBs one by one which achieves reducing the performance impact on the original SoC. The detailed communication mechanism is described in section 4.2.

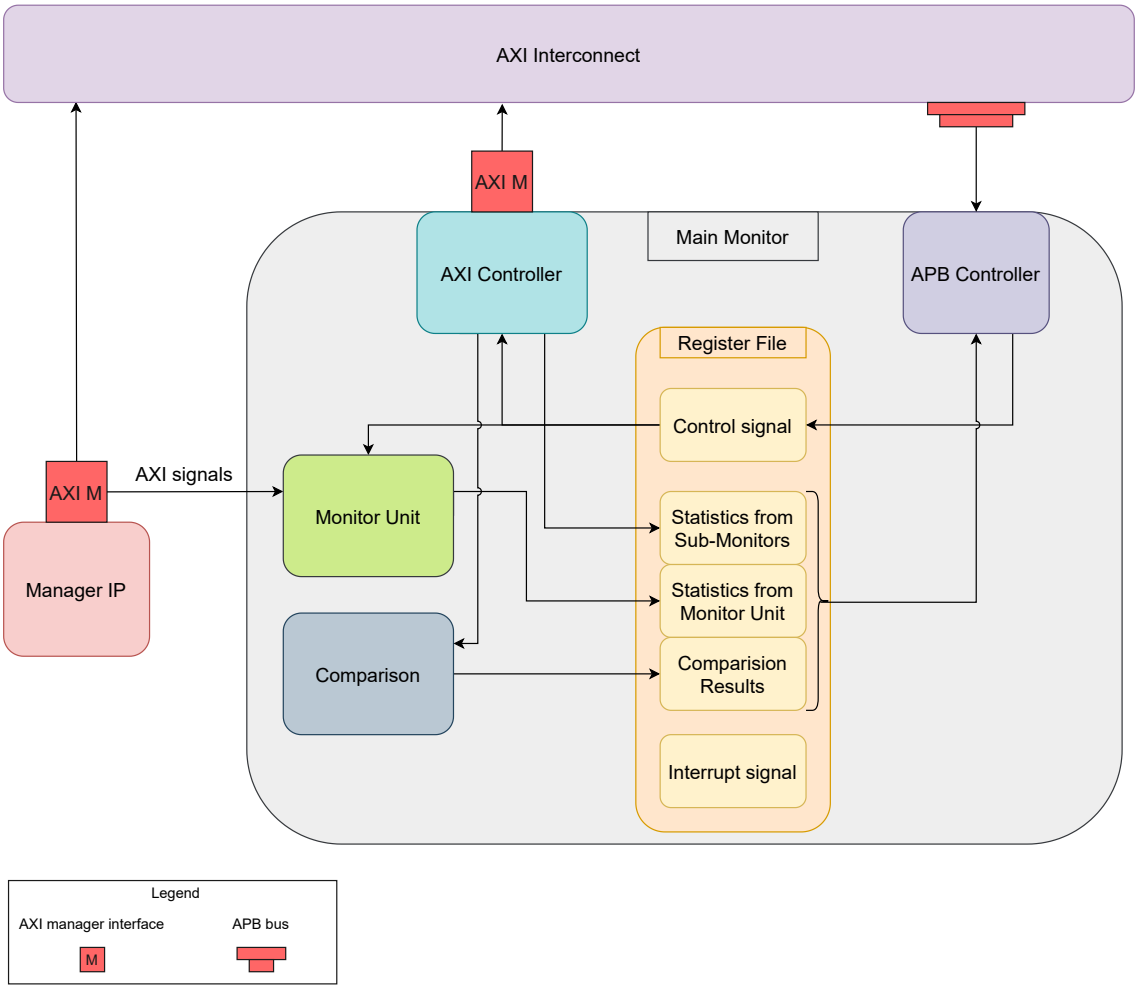

Figure 4.6: The block diagram of the main monitor

Fig. 4.6 show the overall block diagram of the main monitor. It includes a *Monitor Unit* module, a *Register File* module for SM_MM, an *APB Controller* module, an *AXI Controller* module and a *Comparison* module. The following is an introduction to the functions of each module in Fig. 4.6:

- *Monitor Unit* module: It is same as *Monitor Unit* module for SM_SUB in Fig. 4.5. It taps the AXI signals from the nearby M IP and calculates performance statistics. Its output will be stored in *Statistics from Monitor Unit* of *Register File* module for SM_MM.
- *Register File* module for SM_MM: It is a set of 32-bit registers, including read-only-type registers, a write-only-type register and an interrupt-type register. The read-only-type registers stores statistics from sub-monitors, from monitor unit and comparison results. The write-only-type register includes 30-bit time window scale signal, 1-bit enable signal and 1-bit report signal, same as the control signal in SM_SUB. The interrupt-type register firstly needs to be enable by the DP/SW, then triggered by *Comparison* module, finally be cleared by the DP/SW.
- *APB Controller* module:
  - Receiving the control signal from the DP/SW and storing it in *Register File* module.
  - Sending out the statistics from SM_MM to the DP/SW, which are *Statistics from sub-monitors*, *Statistics from Monitor Unit* and *Comparison Results*.
  - Enabling the interrupt signal and clear the interrupt signal from the AXI write transactions, initiated by the DP/SW.
- *AXI Controller* module: Since it needs to broadcast the control signal and read statistics according to the number of monitor units which is a configurable parameter of the *AXI Controller* module. Its functions are as follow:
  - Broadcast the control signal: Once the report signal and enable signal in the control signal updates, *AXI Controller* will initiate two AXI write transactions to two SM_SUBs for broadcasting the latest control signal. The control signal will be synchronous between SM_MM and two SM_SUBs after these two SM_SUBs all successfully received the broadcasting control signal.
  - Statistics collection: When *AXI Controller* module receives the response signal of broadcasting the report control signal for sub-monitor 0, it will initiate AXI read transactions to sub-monitor 0 so that *AXI Controller* collects all the performance statistics from sub-monitor 0 and store them one by one in *Register File* module for SM_MM. Then it will repeat the same process for sub-monitor 1.
  - Collection finished flag: Once the *AXI Controller* module receives all the statistics from all SM_SUBs, it would generate a collection finished flag to trigger *Comparison* module to start comparison.
- *Comparison* module: It compares the statistics from SM_MM and SM_SUBs to get the comparison results such as maximum or minimum latency with the corresponding M IP index; maximum or minimum throughput with the corresponding M IP index; maximum or minimum bus efficiency with the corresponding M IP index, and so on. Since there is only one the comparison

operand hardware for one comparison result, the *Comparison* module is a pipeline comparison whose pipeline stage is the number of monitor units, which is three in the current design. In addition, the number of monitor units is a parameter so that this module is configurable. Finally, when *Comparison* module finishes the comparison, it will generate a comparison finished flag to trigger the interrupt signal in *Register File* module of SM_MM.

4.5 A List of Statistics

This sub-section provides list of performance metrics obtained from the statistics monitor. Figs. 4.7, Figs. 4.8, Figs. 4.9 and Figs. 4.10 provide reference to the statistics mentioned below:

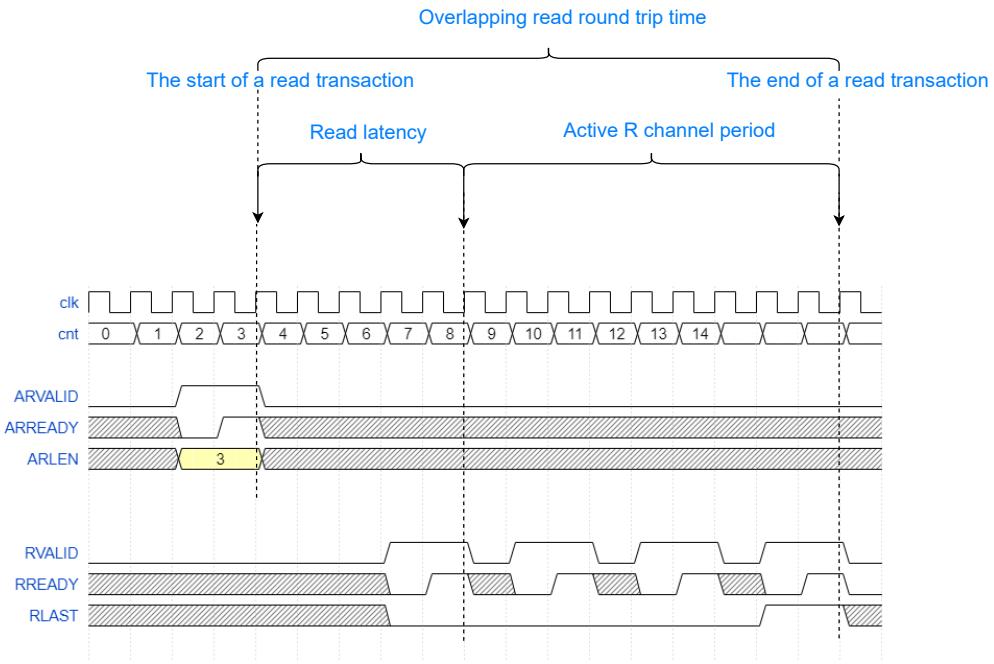

Figure 4.7: Timing diagram of read statistics a single read transaction

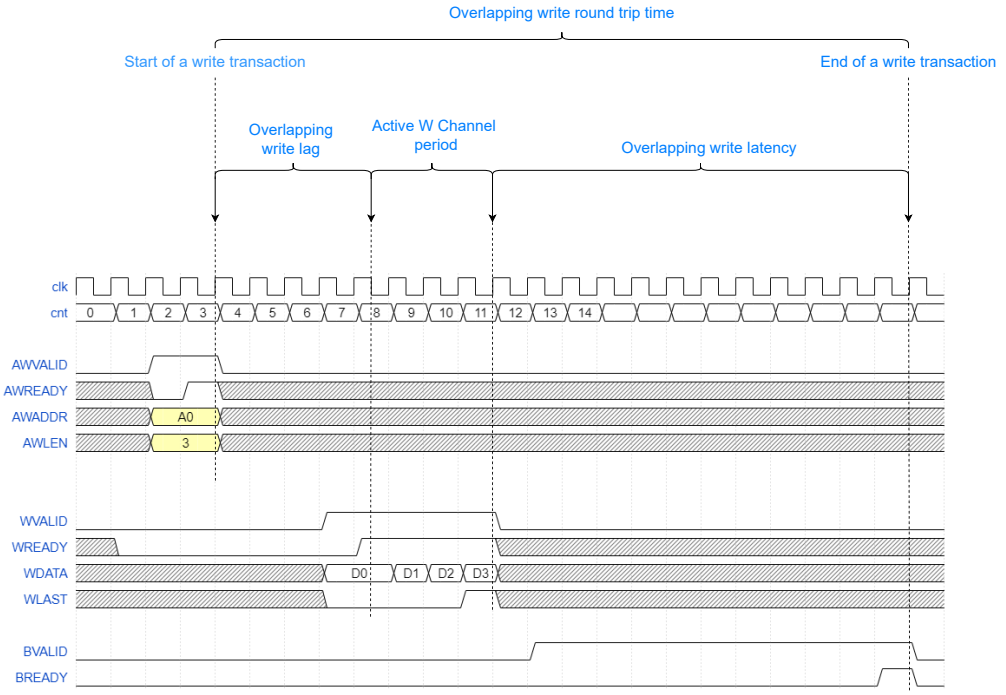

Figure 4.8: Timing diagram of write statistics for a single write transaction

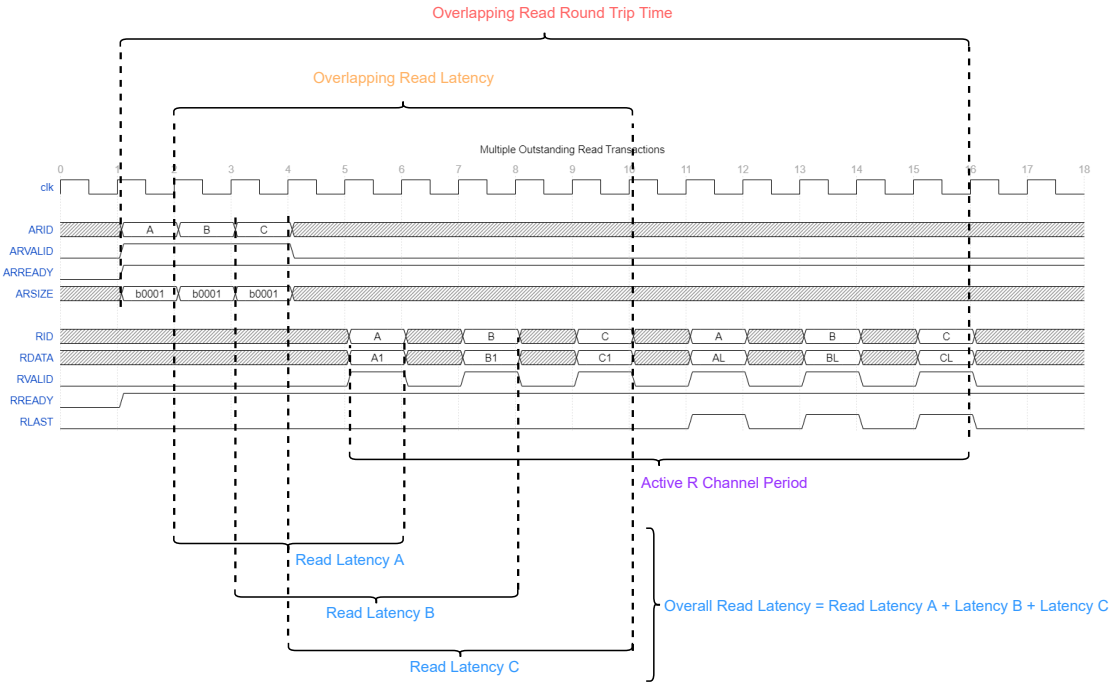

Figure 4.9: The timing diagram of read statistics definition for outstanding read transactions

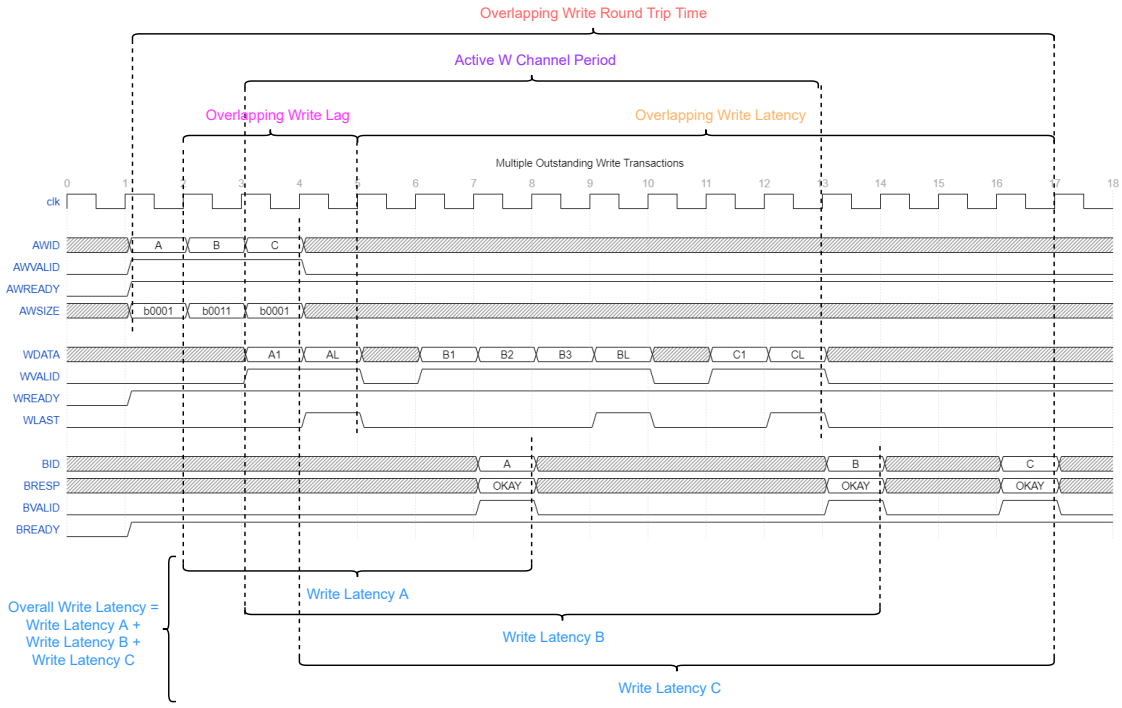

Figure 4.10: The timing diagram of write statistics definition for outstanding write transactions

- **The number of transactions**: This statistic records the number of transactions using the handshake signals from AW channel and AR channel. The monitor records the number of transactions over the entire time window which is throughout the simulation time and records the number of transactions in each sub-window. The period of each sub-window is configurable.
- **Burst length**: This statistic captures the exact number transaction issued by an M IP interface over an entire time window and separate sub-windows. It also captures the maximum and minimum burst length issued by a M IP.
- **Burst size**: The AXI protocol, has the following burst sizes 1, 2, 4, 8, 16, 32, 64 and 128 bytes which specifies the maximum number of data bytes to transfer in each beat. This statistic records the maximum and minimum burst size over an entire monitor window. It also records the number of transactions of different burst sizes for showing the proportion of different burst sizes.
- **Burst type**: The AXI protocol, all the burst types of transactions are fixed-address burst (FIXED), incrementing-address burst (INCR) and wrap-address burst (WRAP). This statistic records the number of transactions of different burst types for showing the proportion of different burst types.
- **Response latency**: The AXI protocol, there are two response channels, R channel and B channel. The response latency at M IP includes the round trip latency in AXI NIC and the processing latency at the destination Sub IP. For read transactions, we define the read response latency counts the clock cycles when a read transaction is initiated on AR channel until the read transaction receives the first read data item with a handshake and the read response

on R channel. For write transactions, we define the write response latency counts when the last handshake on AW channel or W channel happens in a write transaction until the write transaction receives a write response signal with a handshake on B channel. For outstanding transactions in Fig. 4.9 and Fig. 4.10, overlapping latency is the response latency for three outstanding transactions so average overlapping latency is defined as $\frac{overlapping\ latency}{number\ of\ transactions}$. However, since overlapping latency hides the actual latency for each transaction because of the AXI outstanding capacity, the monitor unit introduces average overall latency, $\frac{overall\ latency}{number\ of\ transactions}$, presenting the actual latency for each transaction. The latency for each transaction would be individually recorded and finally accumulated into overall latency.

- **Throughput**: Throughput is defined as $\frac{transfer\ data\ amount}{a\ monitor\ period}$ where transfer data amount records the transfer amount of actual data during the corresponding monitor period. For example, there are 64-byte data going through a AXI port over 10 clock cycles, the throughput is $\frac{64\ bytes}{10\ clock\ cycles}$ is defined as 6.4 byte/cycle. There are three different kinds of throughput in the design:
  - **Overall Throughput**: Over the entire monitor time window, the monitor unit records the overall data amount, so *overall throughput* is defined as $\frac{overall\ data\ amount}{the\ entire\ monitor\ time\ window}$.
  - **Time Unit Throughput**: To get a more accurate throughput estimate to know the trend of throughput over time, the entire monitor window is divided into 8 fixed sub-monitor time windows. the monitor unit would records 8 separate time unit data amounts for 8 fixed sub-monitor time windows. Thus, the throughput for each sub-monitor time window is defined as $\frac{time\ unit\ data\ amount}{each\ sub{-}monitor\ time\ window}$. In addition, the sub-monitor time window scale could be programmed by the DP/SW.
  - **Active Throughput**: Since there are some M IPs maybe idle for a long time during the entire monitor time window, overall throughput would be meaningless to obtain the useful performance statistics. Thus, active throughput only considers the active AXI channel period without the idle time so the monitor unit could extract valid throughput information. *Active Throughput* is defined as $\frac{overall\ data\ amount}{overlapping\ active\ round\ trip\ time}$.
- **Valid transfer data item and valid bus efficiency**: The AXI protocol, there are four responses such as normal access okay (OKAY), exclusive access okay (EXOKAY), slave error (SLVERR) and decode error (DECERR). OKAY means an AXI write transaction successfully sends all write data items to the destination IP or an AXI read transaction successfully receives a read data item. The slave can also return slave and decoder error messages. Valid data transfer records the total number of transactions that returned an OKAY response which means successful data transfer. Thus, valid bus efficiency is defined as $\frac{valid\ transfer\ data\ length}{the\ entire\ monitor\ window}$.
- **Overlapping round trip time**: For the read transaction, round trip time (RTT) is defined as the duration from when the manager IP initiates a read transaction on the AR channel channel to when it receives an acknowledgment

signal for the last data item on R channel. For the write transaction, round trip time is defined as the duration from when the manager IP initiates a write transaction on the AW channel to when it receives the response signal on the B channel. RTT is influenced by the following factor interconnect traffic, distance, and subordinate response time. Overlapping round trip time is also the active period of the whole AXI port. Fig. 4.9 and Fig. 4.10 show an example of the overlapping round trip time for outstanding read and write transactions, respectively.

- **Bus efficiency**: Similar to throughput, bus efficiency has different definitions for different statistics parameters.
  - **Overall bus efficiency**: It is defined as $\frac{number\ of\ data\ items}{the\ entire\ monitor\ time\ window}$, where the number of data items flows through the monitored AXI port during the entire monitor data time window.
  - **Time unit bus efficiency**: It is defined as $\frac{number\ of\ data\ items\ in\ each\ window}{each\ sub{-}monitor\ time\ window}$, where the number of data items flows through the monitored AXI port during each sub-monitor time window.
  - **Active bus efficiency**: It is defined as $\frac{number\ of\ data\ items}{overlapping\ active\ round\ trip\ time}$. where the number of data items flows through the monitored AXI port during the overlapping round trip time.
- **Write lag**: On the W channel, write data items can appear before or after the address is issued on the AW channel. As can be seen from Fig. 4.8. The time difference between data and address channel is going to add extra clock cycles to the processing latency of AXI NIC. This parameter is going to determine which manager is going to add extra clock cycles delay in the design.
- **Total simulation time**: As the name suggests this statistics is going to record the entire monitor window of the statistics monitor. As can be seen from Fig. 4.7 and Fig. 4.8.

4.6 Address Mapping for All Register Files

In the demo implementation of one main monitor and two sub-monitors, their register files with corresponding APB interface need to preset their address so that M IPs are able to access those register files. In Table 4.1, SM_MU_SUB0 and SM_MU_SUB0 are two similar register files for sub-monitor 0 and sub-monitor 1 respectively. Their necessary decoded address bit is 12 and they both own the memory size of 4.00 kB, which is the minimum size for each block in the current SoC. The only difference between these two register files is the address, since they should own unique addresses in SoC and their start addresses are selected by the unreserved address in the address mapping of its tapping M IP. Additionally, SM_MM means the register file for the main monitor whose necessary decoded address bit is also 12 and size is 12.00 kB. Since the main monitor needs to store all statistics for its monitor unit and two sub-monitors, it needs at least $4.00\,\text{kB} \times 3 = 12.00\,\text{kB}$. According to Table 4.1, any M IP in SoC is able to read or write to those three register files

when it initiates transaction with corresponding address and its path is connected to those three register files through AXI NIC.

Table 4.1: Address mapping for the register files of the main monitor and two sub-monitors

Block	Addr Bits	Size	Start addr (Byte)	End addr (Byte)
SM_MM	12	12 kByte	0x80030000	0x800032FFF
SM_MU_SUB0	12	4 kByte	0x80033000	0x800323FFF
SM_MU_SUB1	12	4 kByte	0x80034000	0x800324FFF

Table 4.2 shows more detailed registers in the register files for the main monitor and two sub-monitors. Here are the details of different register types:

- **Control signal**: The control signal consists of 30 bits for the window scale configuration, one bit for the report signal and one bit for the enable signal. The main monitor and two sub-monitor all have one control signal to configure the time window scale and control the states of the statistics monitor such as enable monitoring and start reporting.
- **Interrupt signal**: It has four 32-bit registers which are IRQ status/clear register, interrupt enable register, test register and type control register. The IRQ status/clear register is used to trigger the interrupt and clear the interrupt. The interrupt enable register is used for enabling the interrupt.
- **Write statistics**: These are the performance statistics, coming from the monitor unit from tapping AXI AW channel, W channel as well as B channel. In the current RTL design, there are 160 32-bit registers storing the write statistics per sub-monitor and 480 32-bit registers storing the write statistics for the main monitor.
- **Read statistics**: These are the performance statistics, coming from the monitor unit from tapping AXI AR channel as well as R channel. In the current RTL design, there are 128 32-bit registers storing the read statistics per sub-monitor and 384 32-bit registers storing the read statistics for the main monitor.
- **Comparison results**: These are obtained by the comparison block in the main monitor which compares the statistics between three monitor units to get which monitor unit has maximum or minimum latency, which monitor unit has highest throughput, and so on.

In Table 4.2, the addresses between different register types are discontinuous. Although the continuous addresses could reduce the number of read transactions whose maximum read item is 256 in AXI, discontinuous addresses for different different register types can be used for future function extension and adding more statistics in the register file. For example, for the register file of SM_MM, if there is some extended write statistics need to be stored, the register file definition table only needs to add those registers with address offset following the original last write statistics and does not necessary to modify all the address offsets for read statistics and comparison

results. Further, since the number of statistics is much larger than 256, continuous addresses would not help a lot in reducing the number of read transactions. Moreover, the register file in the main monitor reserves three 32-bit address spaces for control signals and four 32-bit address spaces for interrupt signals to support scalability.

Table 4.2: Address mapping for the register files of the main monitor and two sub-monitors

Block	Register type	Start addr	no of 32-bit registers
SM_MM	Control signal	0x80030000	1
	Interrupt signal	0x80030020	4
	Write statistics	0x80030040	160*3
	Read statistics	0x80031410	128*3
	Comparison results	0x80032800	44
SM_MU_SUB0	Control signal	0x80033000	1
	Write statistics	0x80033010	160
	Read statistics	0x80033810	128
SM_MU_SUB1	Control signal	0x80034000	1
	Write statistics	0x80034010	160
	Read statistics	0x80034810	128

5

Verification

This chapter introduces the verification for the microarchitecture of SM_MM for verifying its connectivity and the functionality of the AXI controller in SM_MM. Moreover, it briefly describes the verification environment of stimulus signals for the monitor unit in SM_SUB and SM_MM.

5.1 Verification for the microarchitecture

In this section, the communication mechanism of the statistics monitor (SM) and its connectivity are verified by a simple setup, which only transfers a few statistics instead of all the statistics. Since the complete verification for the SM involved more than 500 statistics for each SM_SUB and more than 1500 statistics for the main monitor, it is very hard to display the waveform figures visibly and easily understandable so the simple setup is displayed. Fig. 5.1 shows the microarchitecture of the statistic monitor without manager IPs. The name of those ports are matching the below waveform figures. To simulate the DP/SW excitation, the testbench initiates AXI write transactions at the port *DP AXI IF* to enable SM to start monitoring, start reporting, configure the clock cycles of each time window, enable as well as clear the interrupt signal. In addition, after data processing unit (DP) receives the interrupt signal, the testbench also initiates AXI read transactions at the port *DP AXI IF* to read the read, write and comparision statistics in SM_MM.

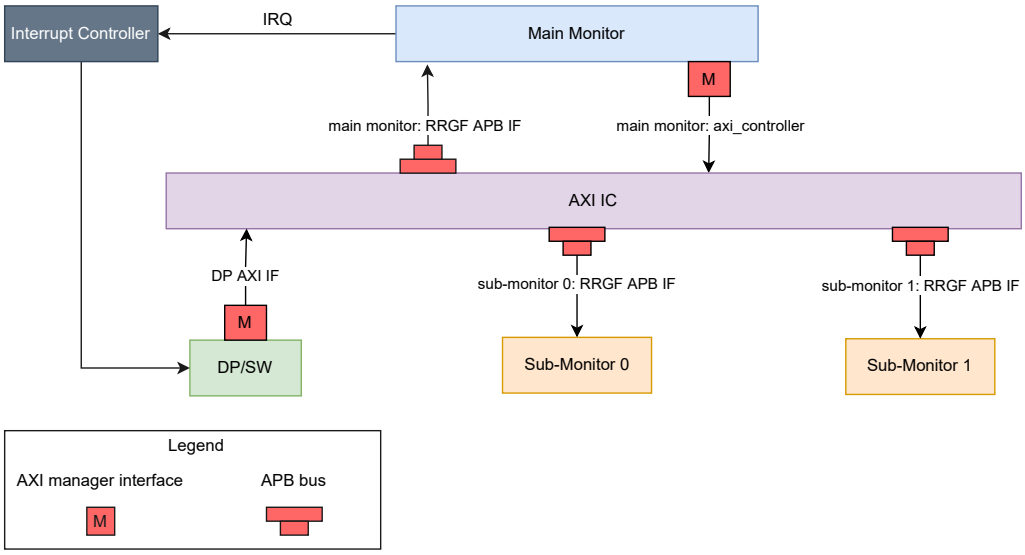

Figure 5.1: The microarchitecture of the statistic monitor without manager IPs

From Fig. 5.2 to Fig. 5.12, we show the waveform figures to illustrate the complete flow of the communication mechanism. The complete flow is displayed in sequence of the numbers with a blue rectangle, which are explained below:

Figure 5.2: The DP sends the enable monitoring signal to SM_MM

1. The DP port initiates an AXI write transaction to send the control signal to SM_MM for starting monitoring of SM. The `awaddr` signal is 0x80030000 which is the global address for the control signals of the main monitor. The `wdata` signal is 0x00000065 which asserts the configurable time window scale, the report signal and the enable signal in the SM_MM as 100 clock cycles, low and high respectively.

2. The write transaction from DP goes through AXI NIC and finally reaches the sub-monitor 0. When apb_psel, apb_penable and apb_pready are all high, the control signal has been successfully transferred into the main monitor. Furthermore, the monitor unit in SM_MM start to monitor its tapping AXI Manager IF.

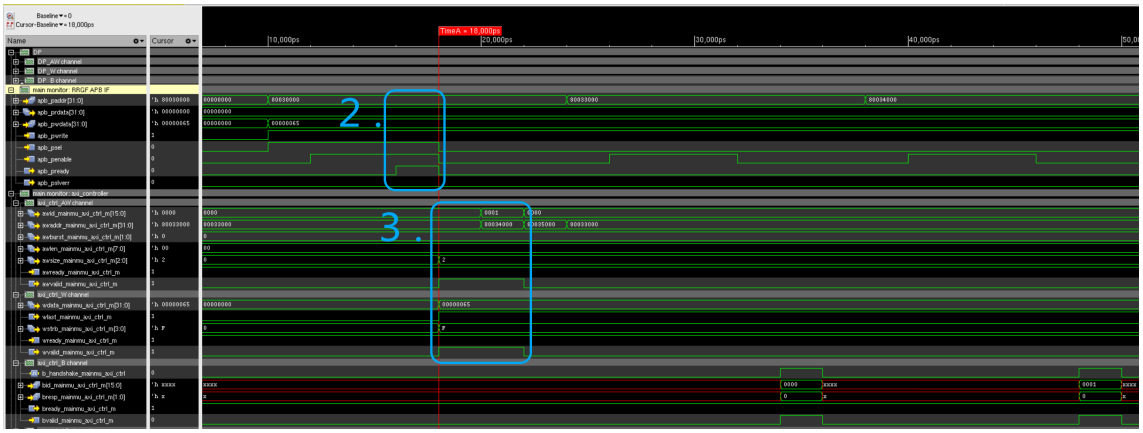

Figure 5.3: The SM_MM broadcasts the enable monitoring signal to two sub-monitors

3. Once SM_MM receives the control signal of starting monitoring, it initiates two AXI write transactions to two sub-monitors respectively for broadcasting the same control signal to the two sub-monitors. The `awaddr` signal passes 0x80033000 and 0x80034000 which are the control signal addresses for sub-monitor 0 and 1 respectively. Moreover, the number of the sub-monitors is a configurable parameter for the main monitor. It could configure the number of broadcast transactions as well as the comparison loop index in the comparison module of the main monitor.

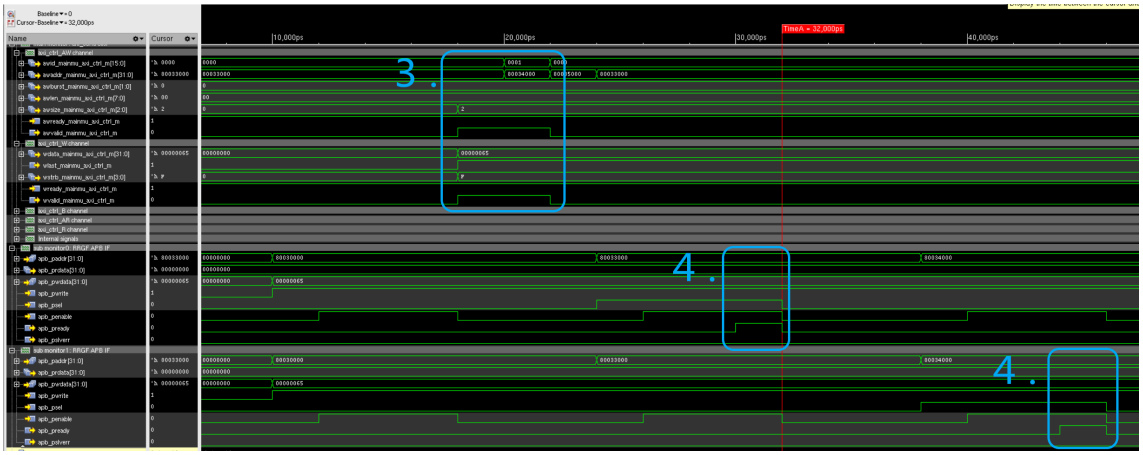

Figure 5.4: The sub-monitor 0 of the statistics monitor (SM_SUB0) and SM_SUB0 successfully receive the enable monitoring signal

4. The enable broadcast transaction from SM_MM goes through AXI NIC and reaches SM_SUB0 and sub-monitor 1 of the statistics monitor (SM_SUB1), so they successfully receive the control signal of start monitoring respectively and then they start monitoring their tapping AXI Manager IF.

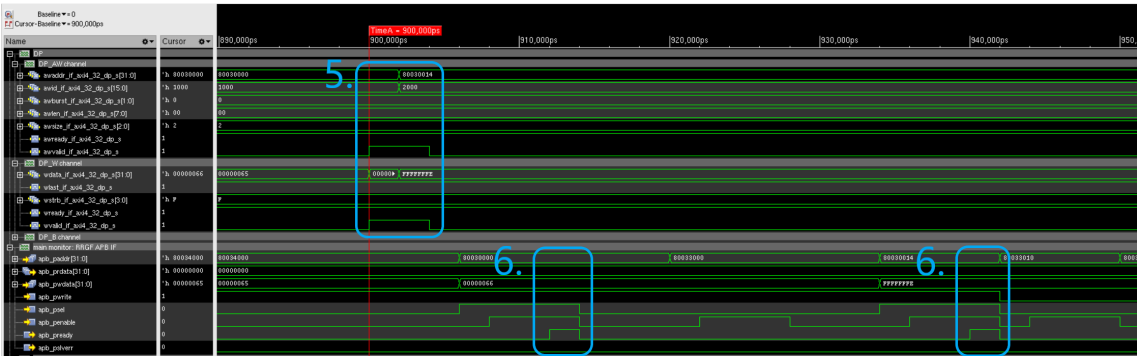

Figure 5.5: The DP sends the report signal to SM_MM and enables the interrupt signal of SM_MM

5. When the DP decides to make SM_MM stop monitoring and start reporting, it initiates an AXI write transaction to assert the control signal as 0x00000065. This data asserts the configurable time window scale, the report signal and the enable signal in the SM_MM as 100 clock cycles, high and low respectively. Then, the DP needs to initiate another AXI write transaction to enable the interrupt signal so SM_MM can generate the interrupt signal to inform the DP the accurate timing to read the statistics from SM_MM. By this interrupt mechanism, the DP does not need to wait and be idle to waste the DP resources since SM_MM needs to spend many clock cycles on collecting all the statistics from all SM_SUB. The total collecting clock cycles depend on the number of collecting statistics from SM_SUB and the timing of the data path in AXI NIC from SM_SUB to SM_MM.
6. SM_MM receives the report signal so the monitor unit in SM_MM stops monitoring and further processes its statistics. Next transaction, SM_MM receives the enable the interrupt signal, so when SM_MM finishes the comparison, the interrupt signal could be asserted. This write transaction transfers the `awaddr` signal of 0x80030024 and the `wdata` signal of 32'hFFFFFFFE to enable the interrupt signal in SM_MM.

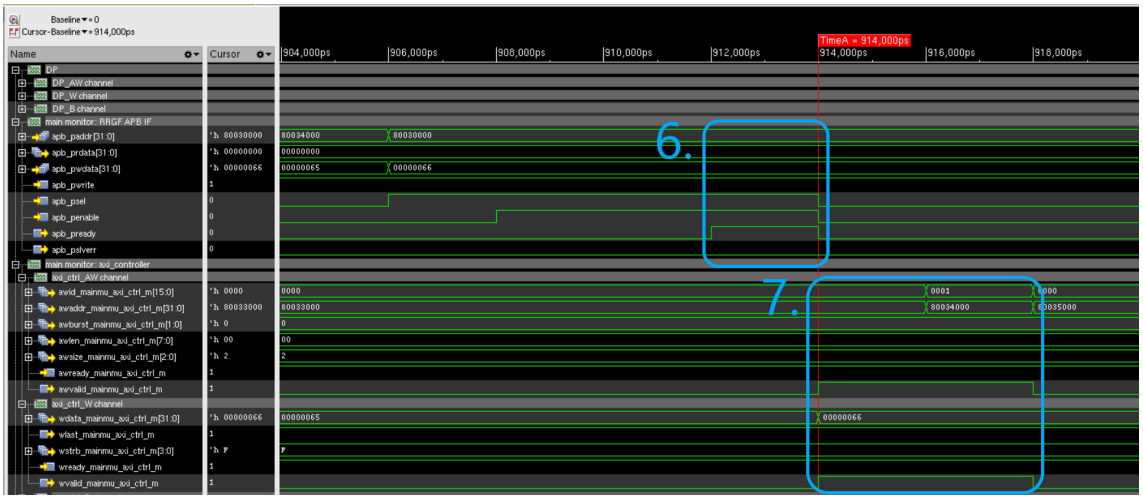

Figure 5.6: The SM_MM broadcasts the report signal to two sub-monitors

7. SM_MM initiate two AXI write transaction to two sub-monitor for broadcasting the report signal.

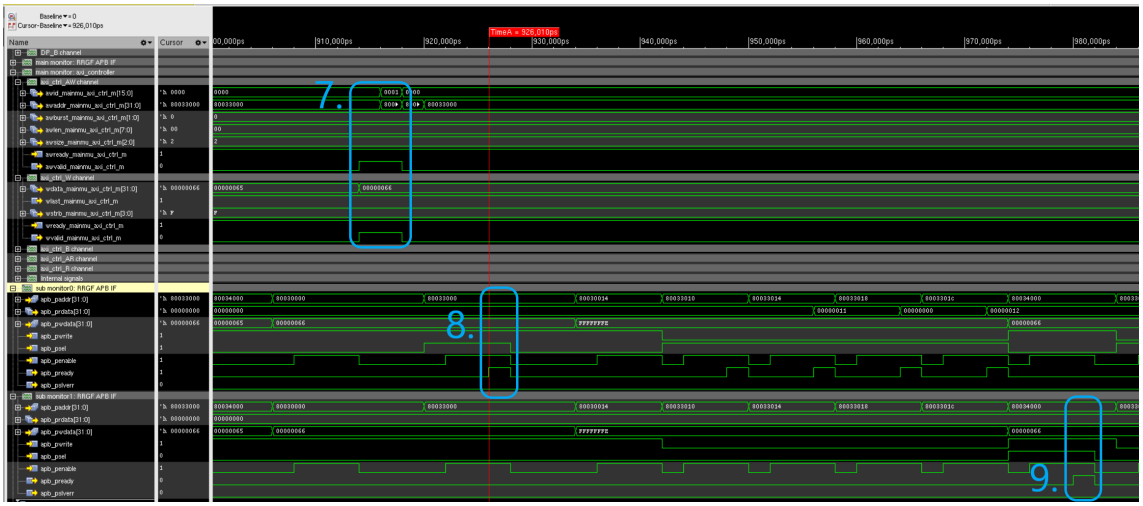

Figure 5.7: The SM_SUB0 and SM_SUB1 successfully receive the report signal

8. SM_SUB0 successfully receives the report signal and then it stops monitoring and further processes its statistics.
9. SM_SUB1 successfully receives the report signal and then it stops monitoring and further processes its statistics.

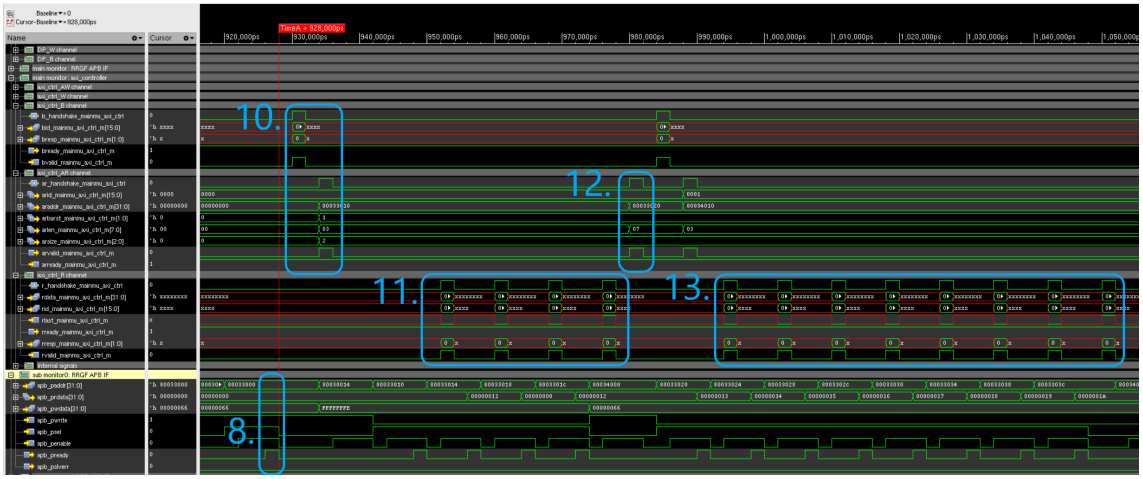

Figure 5.8: SM_MM starts to read statistics from SM_SUB0

10. Once SM_MM gets the write response signal from the write transaction, which broadcasts the report signal from SM_MM to SM_SUB0, SM_MM initiates an AXI read transaction to collect the read statistics from SM_SUB0. The `araddr` signal for this read transaction is the initial address of read statistics for the register file in SM_SUB0 while `arlen` signal should be the number of read statistics in SM_SUB0 minus 1.

11. SM_SUB0 sends out its read statistics by its APB Sub IF, then SM_MM continually collects the read statistics from SM_SUB0 by its AXI R channel.

12. Once SM_MM gets the read last response signal with the same rid as arid, it initiates another read transaction to collect the write statistics from SM_SUB0. The `araddr` signal for this read transaction is the initial address of write statistics for the register file in the SM_SUB0 while `arlen` signal should be the number of write statistics in SM_SUB0 minus 1.

13. SM_SUB0 sends out its write statistics by its APB Sub IF, then SM_MM continually collects the write statistics from SM_SUB0 by its AXI R channel.

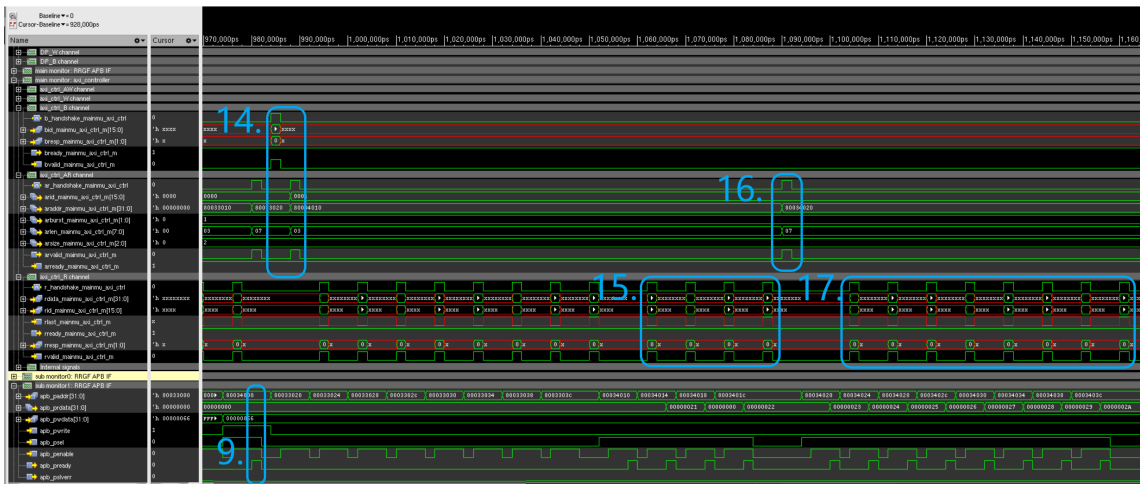

Figure 5.9: SM_MM starts to read statistics from SM_SUB1

14. Once SM_MM gets the write response signal from the write transaction, which broadcasts the report signal from SM_MM to SM_SUB1, SM_MM initiates an AXI read transaction to collect the read statistics from SM_SUB1. The `araddr` signal for this read transaction is the initial address of read statistics for the register file in SM_SUB1 while `arlen` signal should be the number of read statistics in SM_SUB1 minus 1.

15. SM_SUB1 sends out its read statistics by its APB Sub IF, then SM_MM continually collects the read statistics from SM_SUB1 by its AXI R channel.

16. Once SM_MM gets the read last response signal with the same rid as arid, it initiates another read transaction to collect the write statistics from SM_SUB1. The `araddr` signal for this read transaction is the initial address of write statistics for the register file in the SM_SUB1 while `arlen` signal should be the number of write statistics in SM_SUB1 minus 1.

17. SM_SUB1 sends out its write statistics by its APB Sub IF, then SM_MM continually collects the write statistics from SM_SUB1 by its AXI R channel.

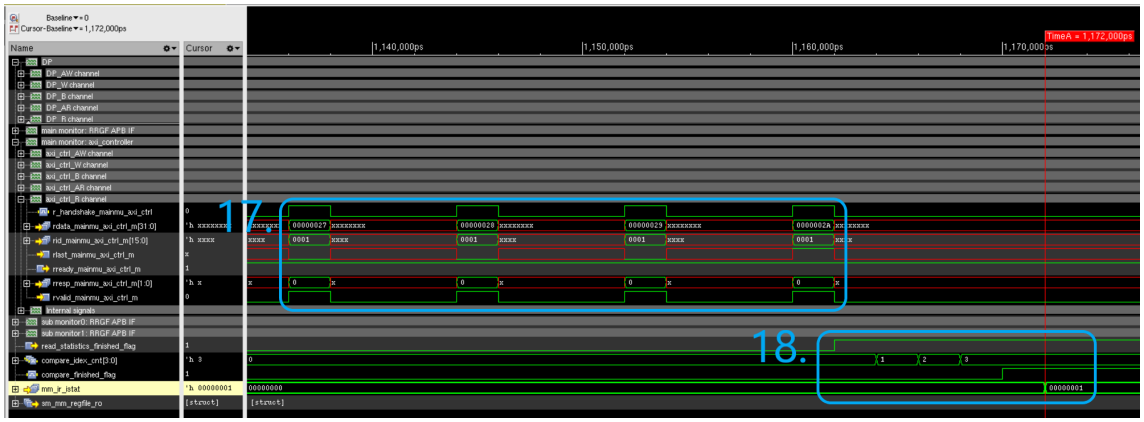

Figure 5.10: The generation of the interrupt signal from SM_MM

18. In Fig. 5.11, when SM_MM finishes collecting all statistics from all SM_SUB, *read_statistics_finished_flag* would be asserted and the comparison module starts to compare the statistics between different sub-monitors to get comparison results. Since the comparison module reuses one comparison RTL instance for each necessary statistics, *compare_index_cnt* is the index to choose the statistic from different sub-monitor as the input of the comparison RTL instance. Once *compare_index_cnt* is equal to the number of sub-monitors plus 1, *compare_finished_flag* would be asserted which is the trigger of the interrupt signal *mm_ir_istat*.

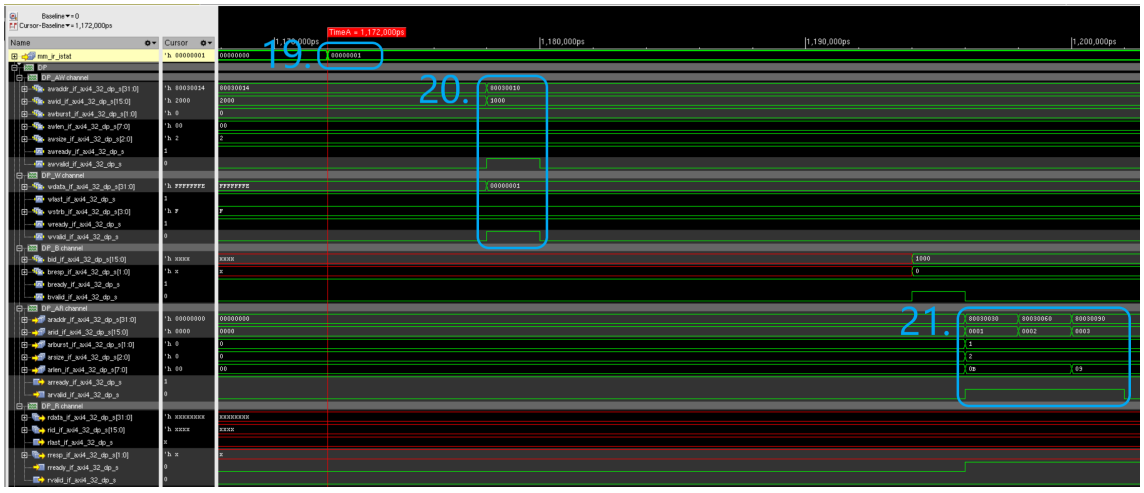

Figure 5.11: The DP clears the interrupt signal and initiates read transactions to read all statistics in SM_MM

19. The interrupt signal is finally asserted to inform the DP that all statistics in SM_MM are ready and that it may initiate read transaction to read those statistics.

20. DP initiates an AXI write transaction to clear the interrupt signal. The `awaddr` signal of 0x80030020 is the clear bit address of the interrupt signal of the

register file in SM_MM. The `wdata` signal should be 1 to clear the interrupt signal.

21. DP initiates at least three read transactions to collect the read statistics, write statistics and comparison results respectively. The `araddr` signal should be the initial addresses of the read statistics, write statistics and comparison results in the register file of SM_MM. The `arlen` signal should be the total number of the read statistics minus 1, total number of the write statistics minus 1 and total number of the comparison results minus 1 in SM_MM. Since the displayed waveform figures are created by a simple demo testbench, the `araddr` and `arlen` signals could be replace by the definition of the register file in SM_MM. Moreover, all statistics in the register file of SM_MM is not contiguous in address. There are two reasons. One of the reason is that it is recommended by company to have some unreserved global addresses for future functional expansion and consistency of data types of read or write. The another reason is that the final number of the read statistics or the read statistics in SM_MM are both more than 256, which is the maximum read items capability of AXI. Thus, it is not necessary to make statistics contiguous in address without the benefit of initiate less read transactions.

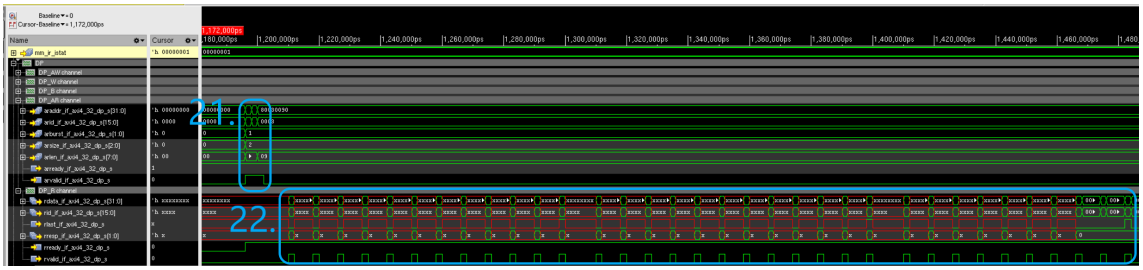

Figure 5.12: DP reads all statistics by its AXI R channel

5.2 Verification of the Logic Functionality of the Monitor Unit

To test the system design a comprehensive test case was written. Each module was verified individually or as a part of a parent entity. The testbench consists of tasks that are used to vary different bus functionalities. Different transaction patterns were created to test and verify the design, starting with the single read and write AXI transaction to a more complex back-to-back transaction. For module which calculates overall and overlapping statistics like latency, round trip time, throughput, and other performance metrics, AXI patterns with variable length, size, burst, responses, and outstanding and interleaving patterns were generated. Module modules that calculate statistics per time unit like latency distribution, throughput, bus efficiency and other performance metrics, AXI patters with varying numbers of the transaction, number of the data item, response, length and size were generated in the different time windows. This allows us to verify predominate AXI patterns in the design.

5.2.1 Block Level Verification

In block level verification, individual modules are integrated together (such as a monitor unit, a register file and a APB interface) and functionality of combined block is verified. To verify the functionality of the monitor unit, different test cases were written to verify the metrics which calculate statistics for different time units and also overall statistics. The following transaction is for different time windows as can be seen in Fig. 5.13.

- window 0 - single transaction - single data item
- window 1 - single transaction - multiple data item
- window 2 - single transaction - out of order transaction
- window 3 - multiple transaction - multiple data item
- window 4 - multiple transaction - outstanding transaction
- window 5 - multiple transaction - interleaving transaction
- window 6 - multiple transaction - varying size
- window 7 - multiple transaction - interleaving transaction

The following section provides the results obtained by simulating the above test cases.

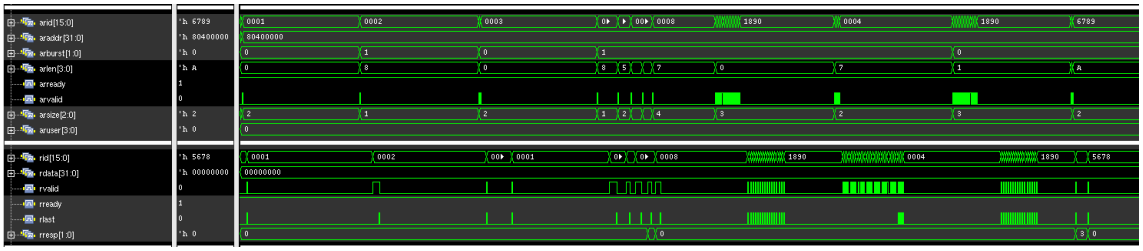

Figure 5.13: AXI transactions generated to verify different statistics per time unit with different sets of transactions in different window

5.2.2 Subsystem Level Verification

In subsystem level verification, all the functional blocks are integrated (such as the main monitor and sub-monitors). A simple AXI interconnect setup with three monitor units, as shown in Fig. 5.1, was used in the verification. This test focuses on testing block integration, connectivity, and the functionality of the whole monitor unit. The output statistics results were verified manually. The main focus of the design is to obtain the statistics results from different monitor units and compare the best or worst performance between different monitor units. In subsystem verification, we exercise that condition by generating three different AXI patterns for three monitor units. The main monitor is connected to a M IP interface which generates multiple single AXI read transactions flowing from a M IP to a subordinate IP. These AXI patterns were captured by the monitor unit to calculate performance metrics. Fig. 5.14 shows the AXI transactions at the main monitor interface. Similarly, sub-monitor 0 was connected to a different M IP interface which generates multiple

outstanding read transactions as can be seen from Fig. 5.15, and sub-monitor 1 was connected to the M IP interface which generates outstanding and interleaving read transactions as can be seen from Fig. 5.16. The number of data items, size of transaction and latency are kept the same for all three monitor units. The total number of data items for all the monitor units was 32 and the latency was twelve clock cycles. Even though the data item and latency are kept the same in all three monitor units the nature of the transaction patterns (outstanding, interleaving, out-of-order patterns) generated makes the difference in performance which can be observed on the dashboard as shown in the chapter 6.

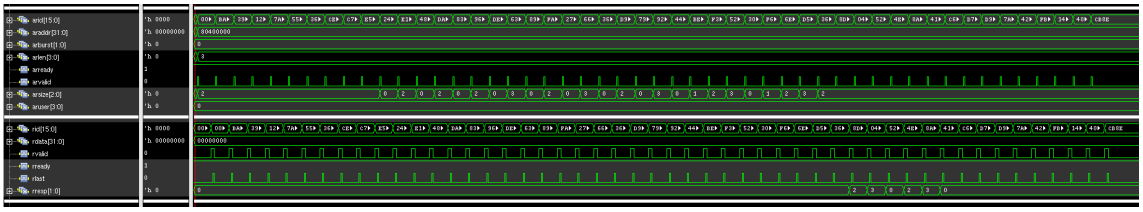

Figure 5.14: Single transactions for overall statistics parameter simulated for main monitor

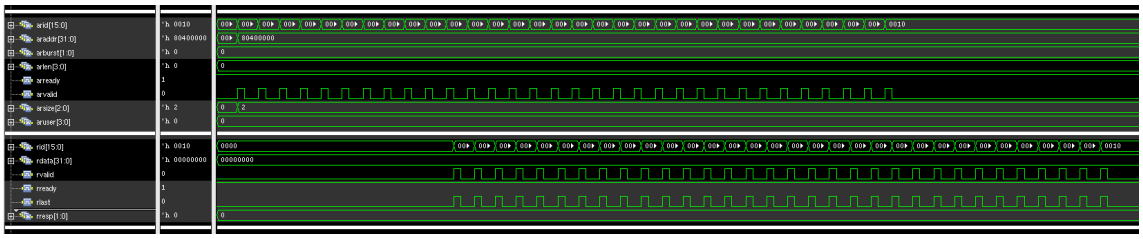

Figure 5.15: Multiple outstanding read transactions for overall statistics parameter simulated for sub-monitor 0

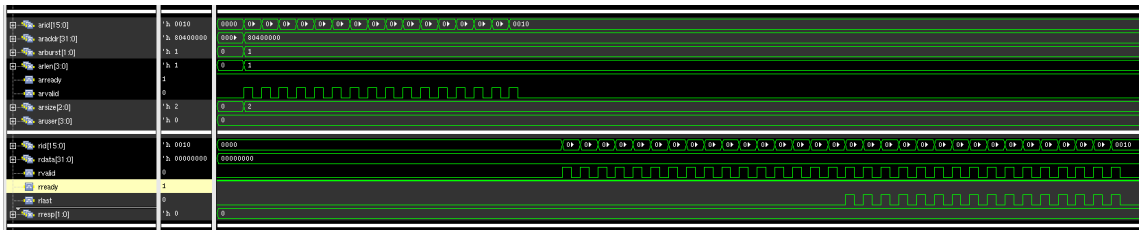

Figure 5.16: Multiple outstanding + interleaving read transactions for overall statistics parameter simulated for sub-monitor 1

6

Results

The simulation output from the statistics monitor is presented in this section. The results are plotted on the dashboard after the output has been carefully verified.

6.1 Dashboard

The statistics findings are shown on the dashboard that is created in Python. Results from the statistics monitor throughout the system are presented on the dashboard in a graphical format. The dashboard includes data for both overall statistics and statistics broken down by time units. The main monitor and sub-monitors all produce the statistics results. The results of the overall statistics from different monitor units are fed into the block called comparison, which outputs the results of the comparison. Statistics per time unit compares the outcomes from several time frames. Details on every statistic that the monitor unit computes are provided in section 4.5. Some of the significant performance indicators gleaned from the statistics monitor are depicted in the figures below in section 6.1.1 and section 6.1.2. A screenshot of a dashboard can be seen in Fig. 6.1. Users can select the statistics they want to plot on the left side. The dashboard displays the chosen graph on the right-hand side.

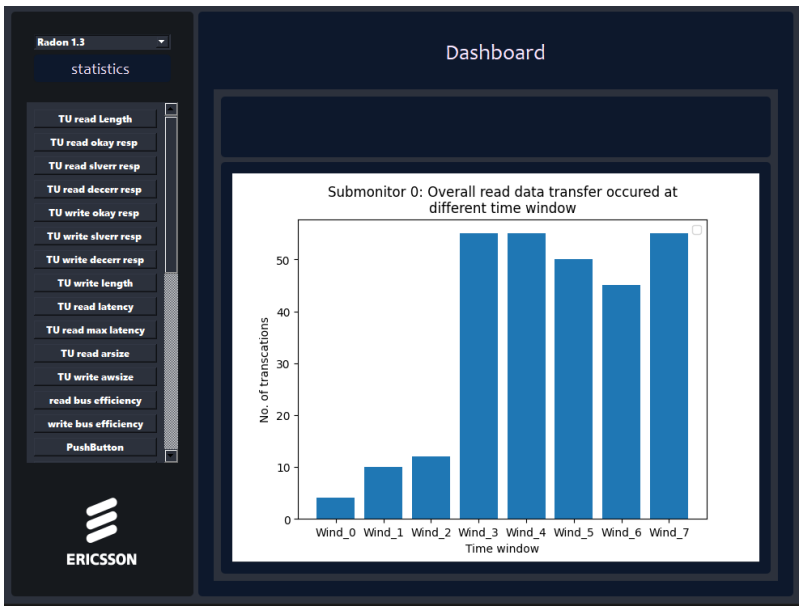

Figure 6.1: The Dashboard

6.1.1 Overall Statistics

The quantitative findings from the subsystem level verification are presented in this section. The test case displayed in section 5.2 was executed to provide the results. The findings from the three monitor units are plotted on the dashboard display.

Fig. 6.2 shows the overall latency per transaction of different monitor units. It was expected that sub-monitor 0 and sub-monitor 1 have a little less latency compared to the main monitor. However, sub-monitor 1 has a very high latency. This result is anticipated because sub-monitor 1 lacks interleaving transactions, and the number of outstanding transactions is very high. As a result, the reaction time is lengthy. The red horizontal line represents the average latency of all the monitor units.

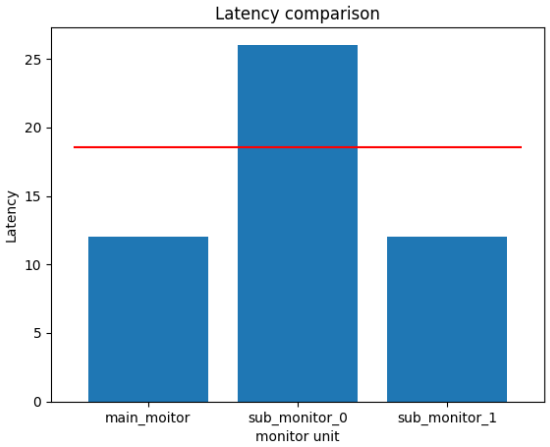

Figure 6.2: Overall read latency comparison

Fig. 6.3 shows the overall throughput of the different monitor units. As expected, the main monitor has a significantly lower throughput, sub-monitor 0 has a slightly higher throughput and sub-monitor 1 has a considerable higher throughput. This is because sub-monitor 0 has outstanding transactions and sub-monitor 1 has both outstanding and interleaving transactions.

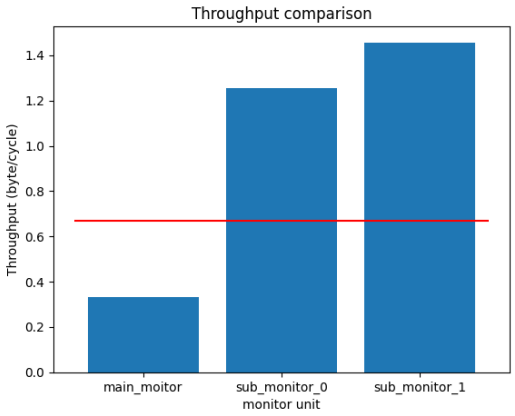

Figure 6.3: Overall read throughput comparison

The overlapping round trip times of the three monitor units are displayed in Fig. 6.4. As previously noted, all transactions retain the same data items. However, sub-monitor 0 and sub-monitor 1 create different numbers of transactions keeping number of data item same. In contrast to sub-monitor 1, which has 16 transactions of length 2, sub-monitor 0 and main monitor has 32 single transactions. Hence the overlapping round trip time of the main monitor is high because of single transactions. Whereas it is considerably low in sub-monitor 1 because of outstanding transactions, the round trip time is significantly higher in sub-monitor 1. It is because sub-monitor 1 has the least number of transactions compared to other monitor units.

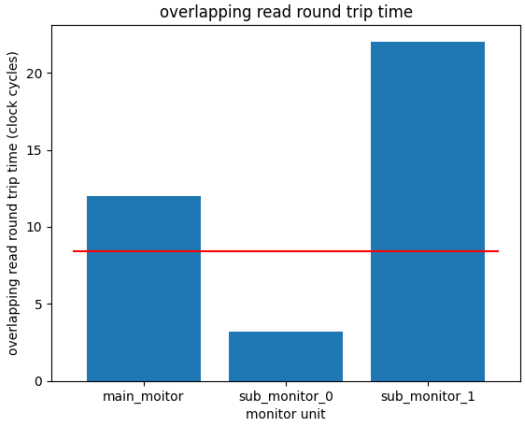

Figure 6.4: Overlapping read round trip time comparison

Fig. 6.5 shows the overall bus efficiency of the different monitor units. As expected the main monitor has the least bus efficiency, sub-monitor 0 has significantly high bus efficiency and sub-monitor 1 has a noticeably higher bus efficiency. It is because sub-monitor 0 supports outstanding transactions and sub-monitor 1 has both outstanding and interleaving transactions.

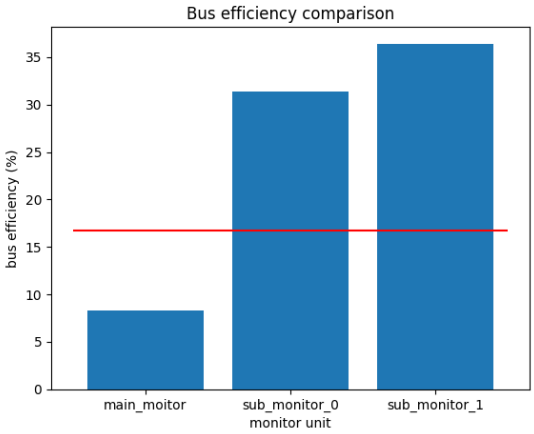

Figure 6.5: Overall read bus efficiency comparison

Fig. 6.6 shows the overlapping read latency of different monitor units. Overlapping

latency is the time difference between the AR channel handshake and the first R channel handshake of the last data item. Overlapping read latency is not affected in the main monitor because single transactions are present at the main monitor interface. Sub-monitor 0 has the least overlapping latency because of outstanding transactions and sub-monitor 1 has slightly higher overlapping latency because of fewer transactions.

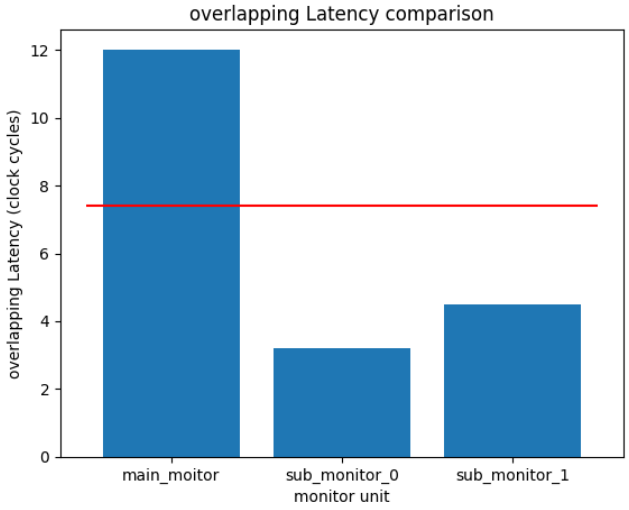

Figure 6.6: Overlapping read latency comparison

Fig. 6.7 shows the overall length captured from different monitor units. As mentioned earlier, the total number of data items is kept the same for all the monitor units, as can be seen in the graph below.

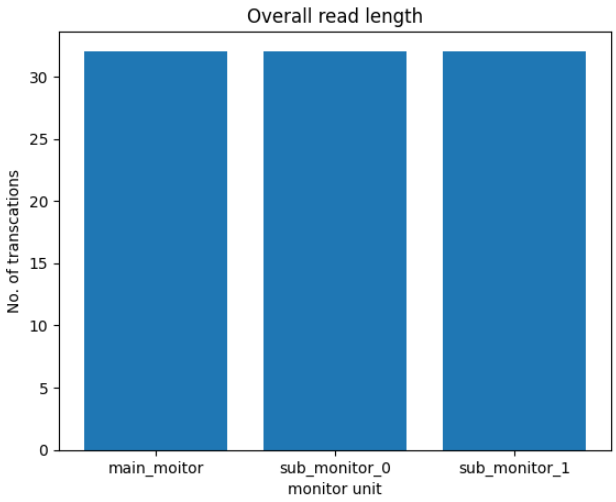

Figure 6.7: Overall read length

6.1.2 Statistics Per Time Unit

The per time unit statistics can show the trend of statistics in different time windows. The design consists of eight time windows. The time scale of each time window is configurable by the parameter in the RTL design. One of the monitor units is selected to verify the functionality RTL design for statistics per time unit. As mentioned in section 5.2, different sets of transactions were simulated in the different time windows. The figures below show some of the dominant performance metrics obtained from the design.

The plot of read throughput over eight-time frames is displayed in Fig. 6.8. Due to the number of single transactions in the first three time windows, the chart demonstrates a relatively low throughput. However, other windows with more transactions and data items have significantly high throughput.

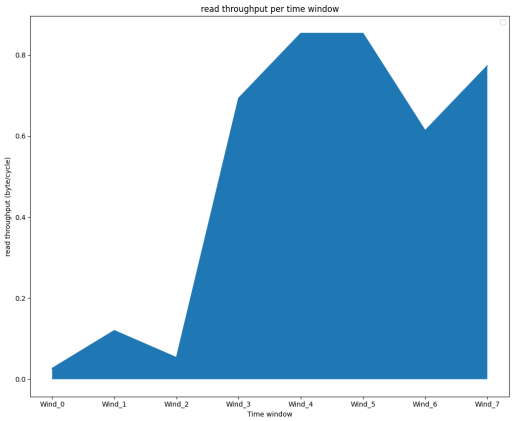

Figure 6.8: Read throughput per time window

Fig. 6.9 shows the read latency distribution. The graph shows details of the total number of transactions distributed over a given latency range, so the latency characteristic for the transactions going through the tapping AXI port can be analysed. For example, the first candle shows six transactions fall in the latency distribution range of one to eight clock cycles and seven transactions fall in the latency distribution range of nine to sixteen clock cycles, and so on.

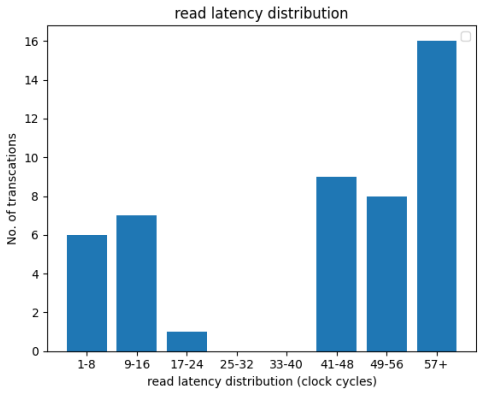

Figure 6.9: Read latency distribution

Fig. 6.10 shows the maximum, minimum, and average latency over different time windows. The black lines indicate the maximum and minimum latency in a particular time window. The upper back line is the maximum latency and the lower black line is the minimum latency. The horizontal red line indicates the average latency over eight time windows.

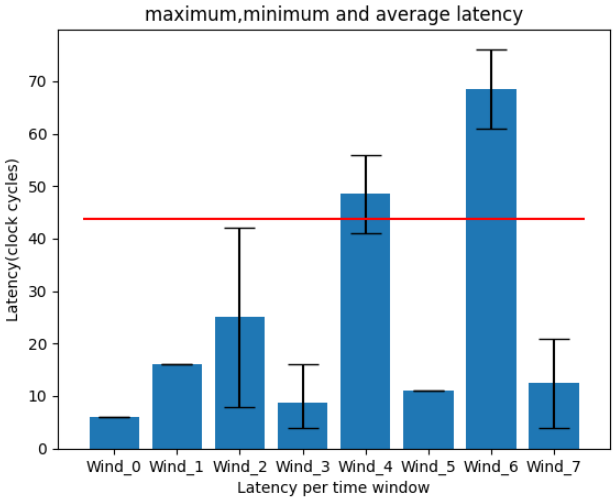

Figure 6.10: Maximum, minimum and average read latency

The plot of read bus efficiency across eight time frames is shown in Fig. 6.11. The first three windows have less than three transactions, indicating significantly low bus efficiency for those time frames. In contrast, other windows with more transactions and data items have significantly high bus efficiency.

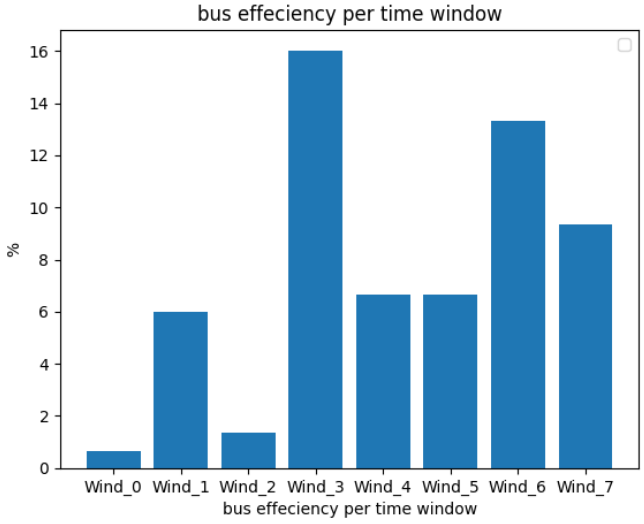

Figure 6.11: Read bus efficiency per time window

Fig. 6.12 shows the plot of overlapping latency over eight time windows. Overlapping latency is the time difference between the AR channel handshake and the first R

channel handshake of the last data item. SinceR channel can accept interleaved read data items from different read transactions, R channel does not need to receive a read item from a new read transaction until the old read transaction is completed. Thus, it improves the throughput and hides latency. Hence to identify the actual latency issue in the design, overlapping statistics are used. As can be seen from the figure, window 2 has significantly high overlapping latency since window 2 has outstanding transactions and a significantly high response rate.

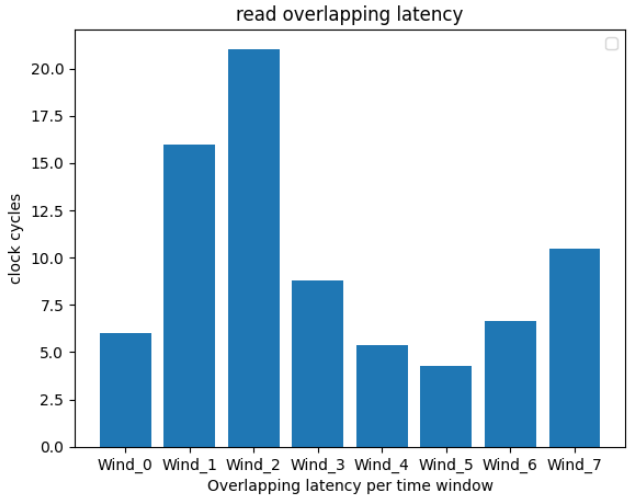

Figure 6.12: Read overlapping latency per time window

Fig. 6.13 shows the plot of read length over eight time windows. The graph indicates the total number of read transaction initiated at different time windows. The first three windows have less number of transactions, and the other windows have multiple transactions.

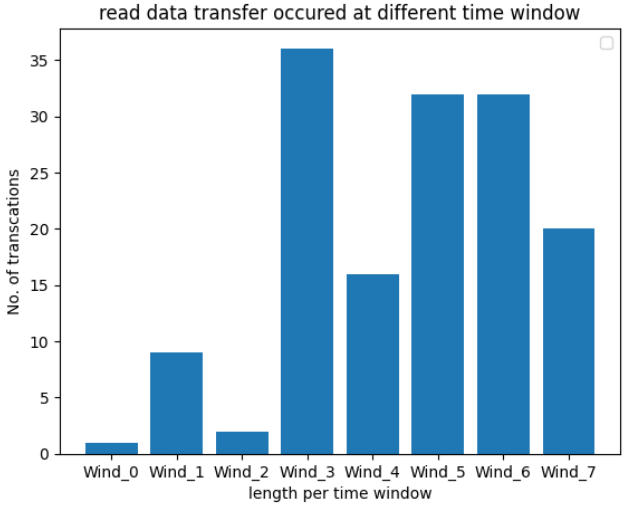

Figure 6.13: Read length per time window

Fig. 6.14 shows the maximum and minimum transaction sizes issued in eight time

windows. The window has a single dash if the maximum and the minimum transaction sizes issued are the same. If the quantity of issued transaction sizes fluctuates, the upper line indicates the maximum transaction sizes issued in the window, and the lower line denotes the least transaction sizes issued in the window.

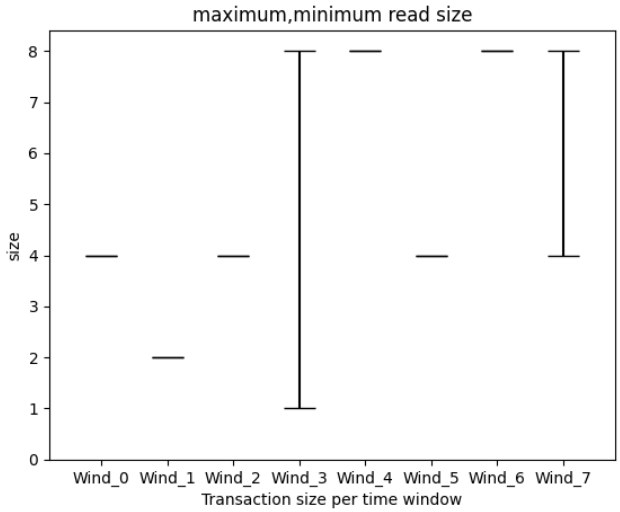

Figure 6.14: Maximum and minimum read size per time window

There are countless applications for the statistics monitor. To comprehend the behaviour of the system, various performance metrics depending on the use case must be examined. Unfortunately, the statistics monitor was not put to the test in a real-world setting during the thesis study. However, the architecture is tested using test cases that simulate real-world situations by adding extra delay, error messages, etc. The dashboard displays and verifies the associated results from the stimulus outcomes.

Applications of Dashboard

The applications of dashboards are as follows:

- Generating plots like latency distribution, length, size, throughput, and other important parameters across the system.
- Comparing two comparable manager IP blocks aids in the work distribution among each manager IP.
- Comparing the results from the different monitor units.
- Identifying the slaves contributing to more mistakes or idle cycles in a transaction by comparing the various manager-subordinate pairings.
- Assigning dashboard results can be used to assign priority to the systems containing multiple bandwidths or latency-sensitive manager IPs.

7

Conclusion

In conclusion, the statistics monitor is a distributed monitor based on an RTL solution and is suitable for a SoC with many manager IPs. It can obtain performance statistics from the SoC, which is running either in a verification environment or a real scenario. Those performance statistics can provide an inside view of data transferring for us to analyse the performance bottleneck of the SoC and try different alternative methods or fix setting errors to maximize the performance of the SoC. This chapter will introduce some summary features, limitations, and applications of the statistics monitor in more detail.

7.1 Summary Features of the Statistics Monitor

These are some features of the RTL design of the statistics monitor:

- **Configurable**:
  - **Hardware configuration (parameters)**: The statistics monitor can configure the AXI width such as the address width, id width, size width and write strobe width to make the monitor unit adjust to most AXI interfaces. According to the number of sub-monitors, the AXI controller module can initiate proper transactions to broadcast the control signal to all sub-monitors and collect statistics from all sub-monitors, and the comparison module can compare the statistics in a proper pipeline based on the the number of sub-monitors plus one. The latency distribution range is also configurable as a parameter in the RTL design to obtain an expected latency range of several clock cycles for the latency distribution dashboard.
  - **Software configuration (a programmable register)**: The statistics monitor can configure the time scale for each time window by assigning the desired value to the programmable register to obtain detailed information from time unit statistics.
- **Supporting all AXI transaction patterns**: The statistics monitor is able to monitor different AXI transaction patterns such as a maximum of 16 outstanding read/write transactions, interleaving read transactions and out-of-order completed transactions.
- **Modular RTL design**: For any AXI port, the monitor unit, including modules of AXI Watcher, Statistics Calculator and Time Window Trigger, can be

instanced without any modification. The register file with an APB interface can be generated by allocating a new global address and modifying its block name as well as area name. Thus, a sub-monitor instance for monitoring an AXI interface is created.

- **Accurate statistics**: Since the tapping sources for the statistics monitor are the AXI manager interfaces of manager IPs inside the SoC at the same clock frequency, the precise granularity is based on the clock cycle level or transaction level.
- **Distributed monitors**: Manager Intellectual Properties (M IPs) are located in different locations in the floor plan in the SoC. Having only one monitor instance monitoring all M IPs in the SoC would consume limited line resources within the SoC, add many register slices to meet the timing constraint and increase the pressure on placement and routing. Therefore, the statistics monitor is a distributed monitor, based on monitor units with a dedicated AXI NIC. It places the main monitor (SM_MM) and sub-monitors(SM_SUBs) near all M IPs to reduce registers slices and wire resource consumption while the statistics monitor has a dedicated AXI NIC to reduce pressure on placement and routing, and to build the communication mechanism.
- **Centralized interaction with the DP/SW**: The DP/SW only needs to communicate with the main monitor to achieve controlling the whole statistics monitor. Since the main monitor can broadcast control signals to all sub-monitors, collect statistics from all sub-monitors and get the comparison results between sub-monitors and the main monitor, it is not necessary for the DP/SW to communicate with sub-monitors individually.
- **A standard APB interface with a global address map for every register file**: Since the statistics, the control signal and the interrupt signal are inside the register file with the global addresses, the APB interface of the register file can help a manager IP access those signals arbitrarily if the manager IP has a path to access the register file in AXI NIC. Therefore, it is possible for the DP/SW to only enable one of the sub-monitors and only monitor the corresponding AXI port of a manager IP.
- **Dashboard**: The dashboard visualizes statistics from the statistics monitor, making easier to identify performance bottlenecks of the SoC.

7.2 Limitations of the Statistics Monitor

There are some limitations of the statistics monitor below:

- **Within the AXI protocol**: First, since the monitor does not tap the signals of `LOCK`, `CACHE`, `PROT` on AW channel and AR channel, it cannot get any statistics from those signals. Second, the current statistics monitor only supports a burst size for one AXI transaction less than or equal to 8 bytes. The reason is that there is no multiplier block inside the monitor and thus the throughput statistic uses a lot of 32-bit registers to store the number of items belonging

to which byte. For example, a maximum size of 8 bytes uses 8 32-bit registers for overall write throughput and 64 32-bit registers for time unit write throughput.

- **Performance impact on the original system**: Although the interrupt mechanism, the centralized interaction with the DP/SW and the dedicated AXI NIC reduce the performance impact on the original system as much as possible, the DP/SW port still needs to take a significant number of clock cycles to read out the statistics inside the main monitor. The total number of consumed clock cycles depends on the number of statistics inside the main monitor and the reading limitation of the APB interface. There is a total of 908 statistics inside the main monitor and the APB interface takes at least 6 clock cycles to read out one statistic. There are two possible improvements to this limitation. The first method is to use an AXI subordinate interface for the main monitor to replace its APB interface. The DP/SW reads statistics via this AXI subordinate interface would save a significant number of clock cycles and read faster. The second method is that the DP/SW can only read some particular or expected statistics instead of reading all statistics by initiating many AXI read transactions with different read addresses and lengths.
- **Reading specific statistics from sub-monitors**: Since the current AXI controller inside the main monitor is a fixed hardware design without programmable registers, it only can read some or all statistics inside all sub-monitors but cannot read any particular statistic according to the commands from the DP/SW. A possible improvement is optimizing the AXI controller with programmable registers to read specific statistics from sub-monitors.
- **Total run time**: Since a 32-bit counter only counts 8-second data at a clock frequency of 500.00 MHz, the statistics monitor can monitor 8 seconds when the SoC is running. If the statistics monitor requires more monitoring time, the clock cycle counter and the statistics registers should be larger than 32 bits.
- **Data flow**: The statistics monitor cannot study the data flow from a specific M IP to all corresponding Sub IPs. However, it is possible to achieve the function of data flow when the statistics monitor adds an address decoder of all corresponding Sub IPs and allocates the register file for those statistics. In this case, the statistics monitor can get the latency, throughput and bus efficiency from the specific M IP to its corresponding Sub IPs.

7.3 Applications of the Statistics Monitor

Since the statistics monitor can gauge the performance and identify the performance bottleneck for the SoC, there are some possible applications or scenarios below:

- **Determining the optimal hardware priority in AXI**: In the AXI4 specification [11], there is a quality of service (QoS) signal, which assigns the priority of transactions to AXI manager ports in the AXI NIC. According to the QoS signal, the AXI NIC would choose the transaction with the higher QoS value to

process first when the AXI NIC process more than one transaction at the same time. Regarding to determine the QoS signal for transactions or AXI manager ports, the statistics monitor can monitor all AXI manager ports and get the performance results of which QoS scheme can maximize the performance when the SoC runs in a real scenario from different QoS schemes.

- **Determining the optimal software strategy**: When different software programs are applied on the SoC in a specific use scenario, the statistics monitor can help the software developer to determine which software program performs better and is more suitable on the SoC.
- **Checking the accurate latency from a specific M IP to its all Sub IPs**: The verification developers for integration need to get the latency from a M IP to its relative Sub IPs to verify the integration connectivity, but they only can get an approximate latency by software or from other teams. The statistics monitor can help them to get the accurate latency at the system clock level.
- **Comparing performance when there is a change in the system**: While rearranging, adding, or removing some IP blocks from the system design, it is important to check the compatibility of the system with the changes. In such a scenario, the statistics monitor can provide insights into the change in the system by comparing the old performance of the design with the new results.

Bibliography

[1] G. Martin and H. Chang, "System-on-chip design," in *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No.01TH8549)*, 2001, pp. 12–17, doi: 10.1109/ICASIC.2001.982487.

[2] E. Davidson, "SoC or SoP? a balanced approach!" in *2001 Proceedings. 51st Electronic Components and Technology Conference (Cat. No.01CH37220)*, 2001, pp. 529–534, doi: 10.1109/ECTC.2001.927778.

[3] M. Mehendale, "SoC - the road ahead," in *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*, 2006, p. 1, doi: 10.1109/VLSID.2006.149.

[4] A. Hekmatpour, K. Goodnow, and H. Shah, "Standards-compliant IP-based ASIC and SoC design," in *Proceedings 2005 IEEE International SOC Conference*, 2005, pp. 322–323, doi: 10.1109/SOCC.2005.1554521.

[5] H. M. Kyung, G. H. Park, J. W. Kwak, T. J. Kim, and S. B. Park, "Performance monitor unit design for an AXI-Based multi-core SoC platform," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, ser. SAC '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1565–1572, doi: 10.1145/1244002.1244336. [Online]. Available: https://doi.org/10.1145/1244002.1244336

[6] S. Sarkar, S. Chanclar G, and S. Shinde, "Effective IP reuse for high quality SOC design," in *Proceedings 2005 IEEE International SOC Conference*, 2005, pp. 217–224, doi: 10.1109/SOCC.2005.1554498.

[7] A. Hopkins and K. McDonald-Maier, "Debug support for complex systems on-chip: A review," *Computers and Digital Techniques, IEE Proceedings*, vol. 153, pp. 197 – 207, August 2006, doi: 10.1049/ip-cdt:20050194.

[8] R. S, K. Ezra, and K. Mallikarjun, "Design of a bus monitor for performance analysis of AXI protocol based SoC systems," pp. 6313 – 6324, 2014.

[9] Xilinx, "AXI Performance Monitor v5.0 LogiCORE IP Product Guide (PG037)," vol. v5.0, pp. 5 – 22, October 2017, accessed on 2022-05-20. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg037_axi_perf_mon

[10] H. M. Kyung, G. H. Park, J. W. Kwak, T. J. Kim, and S. B. Park, "Design and implementation of performance analysis unit (PAU) for AXI-based multi-core System on Chip (SoC)," *Microprocessors and Microsystems*, vol. 34, no. 2, pp. 102–116, 2010, doi: 10.1016/j.micpro.2010.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933110000050

[11] ARM, "AMBA® AXI™ protocol lec-pre-00490-v4.0 ARM AMBA specification licence," vol. Revision C, pp. 1–1 – 1–6, April 2010. [Online]. Available: https://archive.alvb.in/bsc/TCC/correlatos/amba_axi4.pdf

[12] ——, "ARM Introduction to AMBA AXI," vol. Issue 03, pp. 11–16, March 2021. [Online]. Available: https://developer.arm.com/documentation/102202/0200/Atomic-accesses

[13] Z. Jiang, N. Audsley, D. Shill, K. Yang, N. Fisher, and Z. Dong, "Brief industry paper: Axi-interconnectrt: Towards a real-time axi-interconnect for system-on-chips," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021, pp. 437–440, doi: 10.1109/RTAS52030.2021.00046.

[14] ARM, "AMBA® APB™ protocol lec-pre-00490-v4.0 ARM AMBA specification licence," vol. Revision C, pp. 1–1 – 1–6, April 2010, accessed on 2022-02-01. [Online]. Available: https://developer.arm.com/documentation/ihi0024/c/Introduction/About-the-APB-protocol

[15] K. Rawat, K. Sahni, and S. Pandey, "RTL implementation for AMBA ASB APB protocol at system on chip level," in *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, 2015, pp. 927–930, doi: 10.1109/SPIN.2015.7095347.

[16] P. Jain and S. Rao, "Design and verification of advanced microcontroller bus architecture-advanced peripheral bus (AMBA-APB) protocol," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 462–467, 2021.

[17] ARM, "arm Developer: ARM Socrates," accessed on 2022-07-01. [Online]. Available: https://developer.arm.com/Tools%20and%20Software/Socrates#Technical-Specifications