DESIGN OF A DIRECT MEMORY ACCESS MODULE FOR 32-BIT RISC32 PROCESSOR

BY

TAN E-CHIAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER ENGINEERING (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor

Academic Session: JAN 2022

I TAN E-CHIAN

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
16, Jalan Mutiara 1,
Taman Mutiara Gombak 2,
53100 Kuala Lumpur

Ts Dr Chang Jing Jing
Supervisor's name

Date: 21/4/2022

Date: 22 April 2022

FACULTY/INSTITUTE* OF **INFORMATION TECHNOLOGY AND COMMUNICATION TECHNOLOGY**

UNIVERSITI TUNKU ABDUL RAHMAN

Date: **21/4/2022**

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that **Tan E-Chian** (ID No: **1705490**) has completed this final year project/ ~~dissertation/ thesis~~* entitled "Design of a Direct Memory Access Module for 32-Bit RISC32 Processor" under the supervision of Ts Dr. Chang Jing Jing (Supervisor) from the Department of Computer and Communication Technology, Faculty/Institute* of Information and Communication Technology, and Ts Dr. Ooi Chek Yee (Co-Supervisor)* from the Department of Computer and Communication Technology, Faculty/Institute* of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / ~~dissertation/ thesis~~* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

(*Student Name*)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**DESIGN OF A DIRECT MEMORY ACCESS MODULE FOR 32-BIT RISC32 PROCESSOR**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : Tan E-Chian

Date : 21/4/2022

ACKNOWLEDGEMENTS

Firstly, I would really like to thank my project supervisor who is also my academic advisor, Mr. Mok Kai Ming and Dr Chang Jing Jing for his constant supervision, suggestions for the project, as well as all the encouragement and life lessons he had taught me throughout the years of study. I am thankful because he had given me the opportunity to do this project which allowed me to learn many things and would be helpful for the path I am going to pursue in the future years of study or working environment.

I would also like to thank my beloved family who provided me the support and financial help I need during my years of study. Thank you for supporting me in all the good and bad times I express throughout my life and thank you for always being there whenever I need to turn to someone.

Finally, I must say thank you to my fellow course mates who throughout the years of study, been though hardships and many different obstacles and finally overcame it as the years of study is finally coming to an end. Thank you for all the generous helps and kindness.

ABSTRACT

The project is about the design and implementation of a direct memory access (DMA) specifically for Reduced Instruction Set Computer (RISC-32) processor. In this proposal, a short introduction about the knowledge needed to get the project going is stated at the beginning. Followed by the literature review which talks about previous research done by other researchers.

A short explanation about DMA is introduced and how everything is to be implemented into the RISC32 microprocessor. A block diagram will show what is inside the DMA controller. A partitioning level diagram is drawn to make users understand what is inside the DMA Controller. The pin description of the DMA is drawn out.

Lastly, challenges encountered in this project is written and come the conclusion to state what has been done in this overall project together with the progress timeline made during the weeks.

TABLE OF CONTENTS

TITLE PAGE i

DECLARATION OF ORIGINALITY ..... Error! Bookmark not defined.

ACKNOWLEDGEMENTS ..... **v**

ABSTRACT vi

TABLE OF CONTENTS ..... **vii**

LIST OF TABLES ..... **xii**

List of figures xiii

LIST OF ABBREVIATION ..... **xv**

Chapter 1: Introduction **17**

1.1 Background Information 17

1.1.1 Direct Memory Access (DMA) 17

1.1.2 MIPS 18

1.1.3 Bus 19

1.2 Motivation 20

1.3 Problem Statement 21

1.4 Project Scope 21

1.5 Project Objectives 22

1.6 Impact, Significance, and Contribution 22

Chapter 2: Literature Review **24**

2.1: Overview of Direct memory access (designed by Ahmed, 2019) 24

2.1.1: Advanced Microcontroller Bus Architecture (AMBA) 24

2.1.2: Pin description of DMA (designed by Ahmed, 2019) 25

2.1.3: Block diagrams of Direct memory access (designed by Ahmed, 2019) 26

2.1.4: Timing Diagram of DMAC 27

Chapter 2.2: Overview of Direct memory access (designed by Jagtap, 2017) 28

2.2.1: Multichannel DMA controller 28

2.2.2: Block diagrams of the DMAC 28

2.2.3: Datapath of proposed DMA 30

Chapter 3: Proposed Method/ Approach 32

3.1 Methodologies and General Work Procedures 32

3.1.1 RTL Design Flow 32

3.1.2 Micro-architecture Specification 33

3.1.3 RTL Modelling and Verification 33

3.1.4 Logic synthesis for FPGA 34

3.2 Design Tools 34

3.2.1 ModelSim Student edition-x64 10.5 36

3.2.2 Xilinx Vivado Design Suite 36

3.2.3 PCSpim 36

3.4 Implementation Issues and Challenges 36

3.5 Timeline 37

3.5.1 Gantt Chart for Project 1 37

3.5.2 Gantt Chart for Project 2 37

Chapter 4 System Specification 38

4.1 System Overview of Risc32 Pipeline Processor 38

4.1.1 RISC32 Pipeline Processor Architecture 38

4.1.2 Functionality of RISC32 Pipeline Processor 40

4.1.3 Memory Map of the RISC32 Pipeline Processor 41

4.2 Chip Interface of the RISC32 Pipeline Processor 43

4.2.1 Input Description of the Chip Interface of the RISC32 Pipeline Processor ... 43

4.2.2 Output Description of the Chip Interface of the RISC32 Pipeline Processor . 43

4.2.3 Input Output Description of the Chip Interface of the RISC32 Pipeline Processor 44

Chapter 5: Micro-Architecture Specification 45

5.1 DMA Controller Unit 45

5.1.1 Functionality/ Feature of the DMA Controller Unit 45

5.1.2 Operating Procedure (External Operation) 45

5.1.3 Unit Interface of the DMA controller 46

5.1.4 Input Pin Description of the DMA Controller Unit 47

5.1.5 Output Pin Description of the DMA Controller Unit 47

5.1.6 Input Output Pin Description of the DMA Controller Unit 48

5.1.7 Finite State Machine of DMA controller operations 49

5.1.8 Design Partitioning of the DMA Controller unit 50

5.1.9 Micro-Architecture of the DMA Controller Unit (Block Level) 52

5.2 DMAC Datapath Unit 53

5.2.1 Functionality/Feature of the Datapath Unit 53

5.2.2 Block Interface of the DMAC Datapath Unit 54

5.2.3 Input Pin Description of the DMAC Datapath Unit 54

5.2.4 Output Pin Description of the DMA Controller Unit 55

5.2.5 Input Output Pin Description of the DMA Controller Unit 55

5.3 DMAC Timing and Control Unit 55

5.3.1 Functionality/Feature of the Timing and Control Unit 55

5.3.3 Block Interface of the DMAC Timing and Control Unit 57

5.3.4 Input Pin Description of the Timing and Control Unit 58

5.3.5 Output Pin Description of the Timing and Control Unit 58

5.3.6 Input Output Pin Description of the Timing and Control Unit 59

5.4 DMAC Priority Unit 59

5.4.1 Functionality/Feature of the Priority Unit 59

5.4.3 Block Interface of the DMAC Priority Unit 61

5.3.4 Input Pin Description of the DMAC Priority Unit 61

5.3.5 Output Pin Description of the DMAC Priority Unit 62

CHAPTER 6: VERIFICATION SPECIFICATION AND STMULATION RESULT 63

6.1 Test Plan for DMA Controller 63

CHAPTER 7: SYNTHESIS AND IMPLEMENTATION 64

7.1 DMA Interface With RISC32 Processor 64

7.2 Timing Waveforms of the DMA Processes 65

Chapter 8: Conclusion and Future Work 66

8.1 Conclusion 66

8.2 Future Work 66

Bibliography 67

Appendix A: A

A.1 RTL design module of interfaces. ..... A

A.2 DMA Timing and Control module interface ..... D

A.3 Interface for DMA Registers ..... F

APPENDIX B: Bi-weekly Reports ..... a

B.1: Bi-weekly Week 2 ..... a

B.1: Bi-weekly Week 4 ..... b

B.1: Bi-weekly Week 6 c

B.1: Bi-weekly Week 8 ..... d

B.1: Bi-weekly Week 10 ..... e

B.1: Bi-weekly Week 12 ..... f

B.1: Bi-weekly Week 14 ..... g

Poster H

H

Plagiarism Check Result I

LIST OF TABLES

Table	Page
Table 2.1.2.1: AHB side signals	9-10
Table 2.1.2.1: APB side signals	10
Table 3.2.1.: Comparison among Verilog Simulators	19
Table 3.5.1.1: Gantt chart for Project 1	21
Table 3.3.1.2: Gantt chart for Project 2	21
Table 4.1.1.1: Specification of the RISC32 pipeline	24
Table 4.1.3.1: Memory map description of the RISC32 pipeline processor	26
Table 5.1.8.1: Functionality of DMA controller unit's components	35
Table 5.1.8.2: Internal registers of the DMA controller	36
Table 5.2.1.1: Internal registers of DMAC Datapath	38
Table 5.3.1.1: Internal registers of DMAC Timing and Control Unit	39
Table 5.3.1.2: Command and Register Codes	40
Table 5.4.1.1: Internal registers of DMAC Priority Unit	43

List of figures

Figure	Page
Figure 1.1.1.1: Simple representation of Direct Memory Access	1
Figure 1.1.2.1: Conventional pipeline execution	2
Figure 1.1.3.1: Illustration of three types of buses in a computer	3
Figure 2.1.1.1: Typical AMBA-based system	8
Figure 2.1.3.1 Multiple masters connected to slave	11
Figure 2.1.3.2 DMA in computer architecture	11
Figure 2.1.4.1: Timing diagram of read operation	12
Figure 2.2.1.1: Block Diagram of DMA	13
Figure 2.2.3.1: Datapath of DMAC	14
Figure 3.1.1.1: RTL design flow used to design DMAC	16
Figure 4.1.1.1: An overview on the architecture of the RISC32 pipeline processor	23
Figure 4.1.2.1: functionality of MIPS architecture	25
Figure 4.1.3.1: Memory map of the RISC32 pipeline processor.	26
Figure 4.2.1: Chip interface of the RISC32 pipeline processor needed for DMAC	27
Figure 5.1.3.1: DMA controller unit interface.	30
Figure 5.1.7.1 Finite state machine of DMA controller unit	33
Figure 5.1.8.1: Block-level partitioning of the DMA controller unit	35

Table 5.1.9.1: Simplified Micro-architecture of the DMA controller unit 36

Figure 5.2.2.1: Block interface of the DMAC Datapath Unit. 38

Figure 5.3.1.1: Addresses and Description of each Pin Number for Command Register 40

Figure 5.3.1.2: Addresses and Description of each Pin Number for Mode Register 41

Figure 5.3.3.1: Block Interface of the DMA Timing and Control Unit. 41

Figure 5.4.1.1: Addresses and Description of each Pin Number for Status Register 44

Figure 5.4.1.2: Addresses and Description of each Pin Number for Mask Register 44

Figure 5.4.1.3: Addresses and Description of each Pin Number for Request Register 45

Figure 5.4.3.1: Block Interface of the DMAC Priority Unit 45

Figure 7.1.1: DMA Interface With RISC32 Processor 48

Figure 7.2.1: Timing Waveforms of the DMA Processes 49

LIST OF ABBREVIATION

AMBA	Advanced Microcontroller Bus architecture
DMA	Direct Memory Access
CPU	Central Processing Unit
RAM	Random Access Memory
MIPS	Microprocessor without Interlocked Pipelined Stages
I/O	Input/ Output
VHDL	Verilog Hardware Description Language
RISC	Reduced-Instruction-Instruction-set Computing
IP	integrated property
IC	Integrated circuit
ISA	Instruction set architecture
USB	Universal Serial Bus
ISR	Interrupt Service Routine
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
ASB	Advanced System Bus
DDR	Double Data Rate
AXI	Advanced eXtensible Interface

ROM	Read-only Memory
FSM	Finite-state machine
FPGA	Field Programmable Gate Array
RTL	Register Transfer Level
FIFO	First in First out
IF	Instruction Fetch
ID	Instruction Decode
WB	Write Back
EX	Execute
MEM	Memory

Chapter 1: Introduction

1.1 Background Information

An overview of a must-knows that help identify and understand some facts or knowledge about this current project.

1.1.1 Direct Memory Access (DMA)

DMA is a method used to transferring data from the computer's RAM to another part of the computer without passing into the CPU. By doing this it can help save processing time and is a great and efficient way to move data from the computer's memory to other parts of the computer.

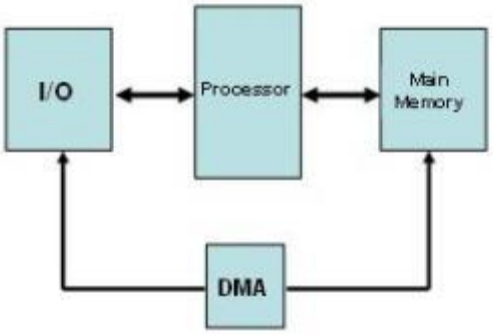

Figure 1.1.1.1: Simple representation of Direct Memory Access

1.1.2 MIPS

MIPS in non-abbreviated form is Microprocessor without Interlocked Pipeline Stages. MIPS is a reduced instruction set computer (RISC) instruction set architecture (ISA) developed by the MIPS Technologies in the USA (Charles, 1995 - https://en.wikipedia.org/wiki/MIPS_architecture#cite_note-Price1995-2). RISC is designed to do a simpler instruction sets with only a few addressing modes as opposed to the Complex Instruction Set Computer (CISC) which has a much more complex sets of instruction. This results in hardware being able to provide much more efficient work as it is less complex, easier and faster for users to build and test. All this of course comes with a price, sacrificing complexity results in an increase of instructions used per program per clock cycle. To avoid this, the instruction is run overlappingly in a pipeline manner as shown in Figure 1.1.1.1. Over the years, the technologies have been improved ever since it is first developed. It can now support 64- bit addressing and operation with high precision and succeeded in making its way into video game consoles, routers etc.

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Figure 1.1.2.1: Conventional pipeline execution

1.1.3 Bus

In a computer subsystem, bus allows the transfer of data between one component to another within the same system mother board or even between two different sets of computers. Transferring data to and from memory or the Central Processing Unit (CPU) and in this case, direct memory access (DMA) is one of the many ways a bus in a computer system can be used. In most case, there are three types of buses that can carry information around the system. These include:

- The address bus which carries memory addresses from the processor to input/ output devices (I/O). this bus is considered unidirectional
- The data bus which carries data between the processor and the I/O. This bus is considered bidirectional.
- The control bus that carries control signals to and from I/O. they can also carry clock signal. This bus is considered bidirectional.

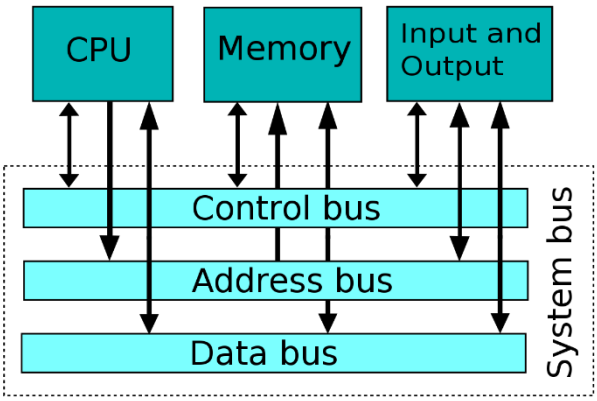

Figure 1.1.3.1: Illustration of three types of buses in a computer

1.2 Motivation

A 32-bit RISC pipeline microprocessor has been developed in the Faculty of Information and Communication Technology (FICT) of Universiti Tunku Abdul Rahman (UTAR) by using Verilog Hardware Description Language (HDL). This project is based on the Reduced Instruction Set Computing (RISC) architecture. The motivations for initiating this project are as follows:

- Microchip design companies have come up with microprocessor designs as Intellectual Property (IP) and is only used for commercial purposes. Microprocessors IP involves the information on a complete design for both front-end (modeling and verification) and the back-end (physical design) of an integrated circuit (IC). These IPs are kept confidential because it usually costs huge sums of money to get the designs.
- There are quite a few microprocessor cores designs that can be found on the internet at OpenCores (http://www.opencores.org/). However, those designs do not utilize the entire MIPS Instruction Set Architecture (ISA), therefore cannot be reused and customized
- In addition to the reason above, these free designs are most of the time complete and underdeveloped as compared to the paid ones. Due to this reason, there isn't any comprehensive verification specification making the verification of a RISC microprocessor slower. This is bad because it will slow the whole design process down significantly.
- When the front-end design is done badly, the physical design would be affected as well and need to be repeatedly changed. This is all due to the poorly design verification specifications for these microprocessor cores.

UTAR has already came up with solutions for the problems mentioned above by creating a 32-bit RISC core-based environment for research as well as specific hardware modelling purpose. The RISC32 project that was initiated in UTAR has completed the CPU designs that supports basic instructions similar to MIPS instructions. The system control coprocessor that is the Coprocessor 0 (CP0) is also available to interface with I/O devices and handle interrupts.

Because of this reason, a DMAC can be developed making data transfers more efficient. Clearly, there isn't many free designs about DMAC available in the internet and even if there is one available, it is either not very detailed due to the advancement of the technologies for a DMA and not comprehensive enough or the data isn't completed for us to build a design.

1.3 Problem Statement

As mentioned above, Direct Memory Access (DMA) is method used for sending and fetching data directly from the input/ outputs to and from the main memory, bypassing the Central Processing Unit (CPU). Without this method, the speed of memory processing in CPU is greatly reduced. In the current generation, it is almost impossible for a CPU to run without the DMA.

CPU will be fully occupied during the process of using a programmed input/ output for the entire read or write operation, making it unavailable to perform other work. Simple task such as transferring files from a USB drive to the computer would be such a hassle. This problem is not only important to tech companies like Huawei, Microsoft, etc, it is also as equally important to anyone that is doing their task on a computer or a laptop. Users would not be able to experience the convenience and speed of the modern world technology as the option to multitask is taken out of the picture. It is crucial for computer engineers as well. Computer engineers spend huge amount of time in simulation which requires a huge workload from CPUs. When DMA is out of the picture, it can be hard for the engineers to complete their work on time which can be very costly in terms failing to fulfil the contract in a required time.

1.4 Project Scope

The project scope will mainly focus on designing and integrating the DMA controller into the existing RISC32 pipeline processor. The specification of the DMAC unit and its internal block will be functionally verified by developing testbenches.

In addition, an Interrupt Service Routine (ISR) for handling all the interrupt requests generated by the DMA controller will be developed and then integrated into the existing exception handler of the RISC32 pipeline processor. Some MIPS test programs will also be written to test the DMA controller's functions after integration as well as to verify the correctness of the ISR execution.

Lastly, a detailed documentation on this project will be developed and maintained. This will be a project that includes a software, simulation result and concept design are expected to be delivered at the end of the project.

1.5 Project Objectives

The objectives of this project are:

- To develop a DMA controller. This involves the micro-architecture modelling and verification of the DMA controller using Verilog language.
- To integrate the DMA controller into the RISC32. This involves the development of the interface between the DMA controller and the RISC32 based on I/O memory mapping technique. An Interrupt Service Routine specifically for the DMA controller will be developed in MIPS assembly language and integrated into the exception handler

1.6 Impact, Significance, and Contribution

After the completion of this project, it can provide a complete RISC microprocessor core-based development environment. The development of this will result in the following:

- A well-developed design and documentation of the chip specification, the top-down architecture specification and the micro-architecture specification
- A fully functional well-developed DMAC that controls the in and out data flow of the system written in Verilog HDL

- A well-developed verification specification of the DMA controller unit. This verification specification contains suitable verification methodology, verification techniques, test plan, testbench architecture etc.

This project would help to come up with an environment that is mentioned above by providing support to hardware modelling research work. Once the project is implemented into the RISC32 MIPS environment, it can significantly increase research speeds of future research work due to the functionality of the DMA controller. Future users can also easily identify its function since a detailed paperwork is being shown here in this project so there won't be any confusions.

Chapter 2: Literature Review

2.1: Overview of Direct memory access (designed by Ahmed, 2019)

2.1.1: Advanced Microcontroller Bus Architecture (AMBA)

Despite this design of DMA controller working on Advanced Microcontroller Bus architecture (AMBA) specifications instead of RISC32 MIPS environment, it can be used as an example of DMA controller designs as this is one of the few designs available on the internet for free. The AMBA specification mainly focuses these two buses: Advanced High-performance Bus and the Advanced peripheral Bus. The DMA controller design works as a bridge between these two buses and allow them to work hand in hand. It provides the options of choosing between buffer or non-buffer data transfer mode according to the peripheral speed. The operation is done synchronized by an asynchronous FIFO.

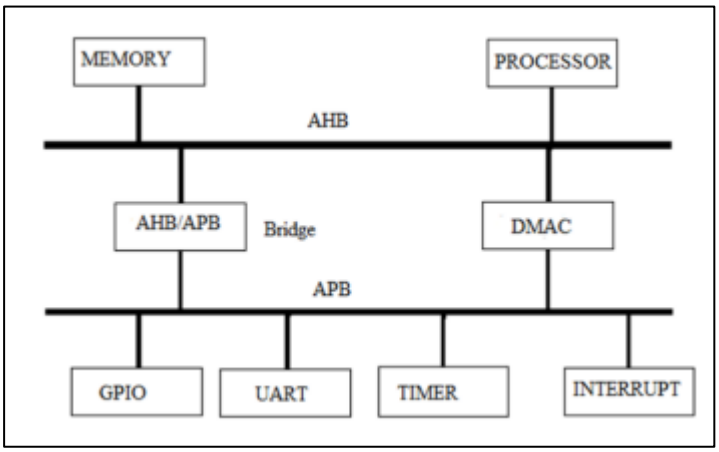

Figure 2.1.1.1: Typical AMBA-based system

- Advanced High-performance Bus (AHB)

  This bus is able to maintain a maximum of 128-bits transfer. It is also able to support several bus masters. The AHB normally contain the AHB slave, AHB master, the control unit (arbitrary), and decoder. In a conventional AMBA design, there will normally be many masters and slaves. The Control unit would give out signals to select one master at a time. This bus I usually at the processor's side.

- Advanced System Bus (ASB)

This is a high performing bus. It will be synchronous once being connected to the control unit from multiple masters allowing them to be accessed in one control unit. This bus implements the pipeline so data and addresses can be transferred concurrently. This is also usually at the processor's side.

- Advanced Peripheral Bus (APB)

  This is a low performing bus which is designed just to connect the peripherals on the System on chip (SoCs). This bus s the AHB master to address one of the slaves of APB system when the APB and AHB is connected. This bus only connects the master and slave together so it is placed on the peripheral side.

2.1.2: Pin description of DMA (designed by Ahmed, 2019)

Name	Bit size	Description
CLK	1	Clock signal
RESET	1	Active low reset
BUSREQ	1	Used to request for bus permission
GRANT	1	Feedback from master that access has been granted
ADDR	32	32-bit address bus
TRANS	2	Transfer type: idle, busy, sequential or non-sequential
WRITE	1	High to write/ low to read
SIZE	3	Size of transfer: byte, half word or word
BURST	3	Transfer form indication
PROT	4	Indicates transfer type, opcode fetch or data access

WDATA	32	Write pins, data transfer from master to slave
RDATA	32	Read pins, data transfer from slave to master
READY	1	High to indicate transfer completion
RESP	1	Feedback status: error, okay, retry or split
SEL	1	Slave selected

Table 2.1.2.1: AHB side signals

Name	Bit size	Description
CLK	1	Clock signal
RESET	1	Active low reset
SEL	16	Able to tell which slave is selected and what type of data transfer
ENABLE	1	Enable Signal
WRITE	1	High to write/ low to read
WDATA	32	Write pins, data transfer from master to slave
RDATA	32	Read pins, data transfer from slave to master
READY	1	High to indicate transfer completion

Table 2.1.2.1: APB side signals

2.1.3: Block diagrams of Direct memory access (designed by Ahmed, 2019)

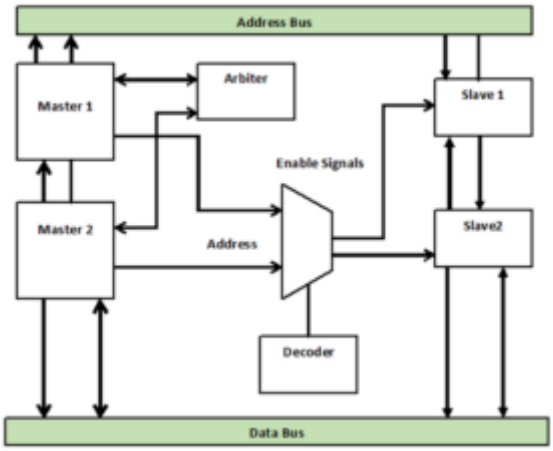

Figure 2.1.3.1 Multiple masters connected to slave

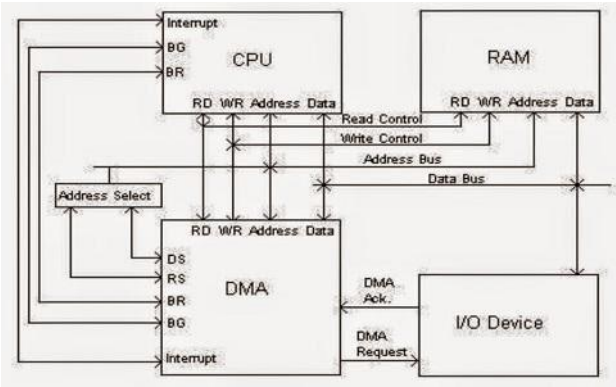

Figure 2.1.3.2 DMA in computer architecture

2.1.4: Timing Diagram of DMAC

To build this AMBA design and implement it so that it works the way it is intended to, the Verilog hardware descriptive language (HDL) is used. The read operations will be activated once the bus has granted access.

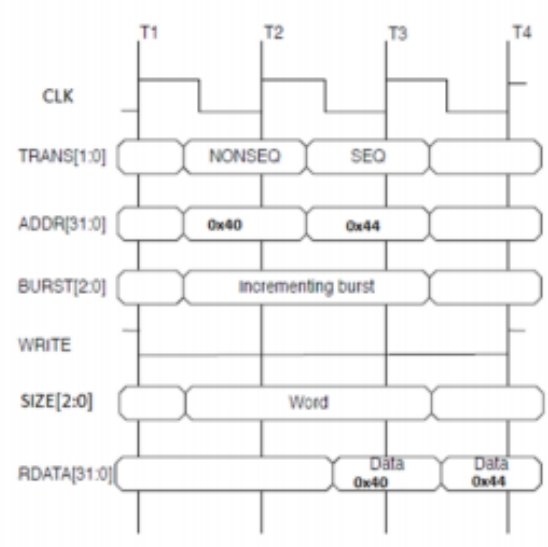

Figure 2.1.4.1: Timing diagram of read operation

Chapter 2.2: Overview of Direct memory access (designed by Jagtap, 2017)

2.2.1: Multichannel DMA controller

Unlike the previous design, this design is implemented into ISA expansion bus. The traditional 4-bit DMA channel is increased to a 16-bit DMAC.

For each channel, two lines are needed in order to function: One of the line request access of buses from the processor and the second one is to prevent data transfer from being disrupted from the processor. For this very purpose, the bus must be used by the DMAC when the processor does not require it or by cycle stealing.

2.2.2: Block diagrams of the DMAC

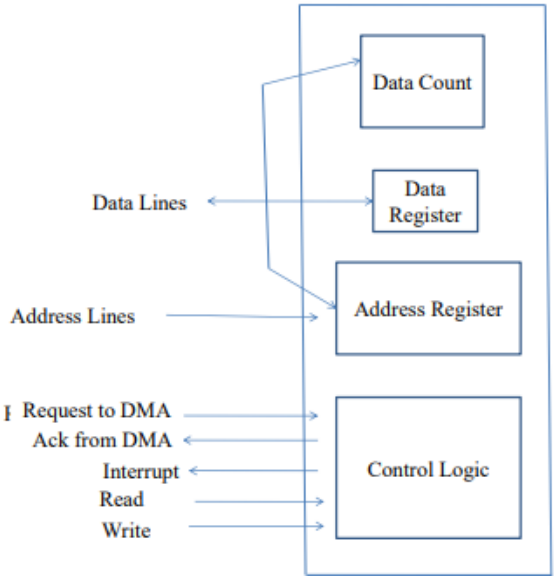

Figure 2.2.1.1: Block Diagram of DMA

The overall functionality is quite the same compared to the previous design. A series of command will be sent from the processor when a huge amount of data is needed to be transferred. The buses will be taken over by the DMA once the procedure has started and can only be stopped once an acknowledge signal is sent from the DMA to the processor that signifies the completion of the process. This is done when the interrupt signal is sent form the DMA.

In this current design, efficient communication is the main focus between the memory and processor with different commands from the control unit which provides more flexibility and versatility to the design. An additional FIFO is added into this design to make up for the

complexity and the less versatile design of the older versions of DMAC. More channels are added to increase the configurability.

With this the DMAC is implemented in the computer architecture and is shown in the figure below. As mentioned above, an interrupt will occur once it senses any data that is going to be transferred from the processor or the memory or vice versa. It stops its current task at hand so that it is able to grant permission to the DMAC to control all the buses once all the requirements are met: handshake, acknowledgment, opcode data, etc. after all this is done, the DMA will check the descriptor for any information related to different DMA channels. Control unit will be able to perform read/ write operation for different DMA channels.

2.2.3: Datapath of proposed DMA

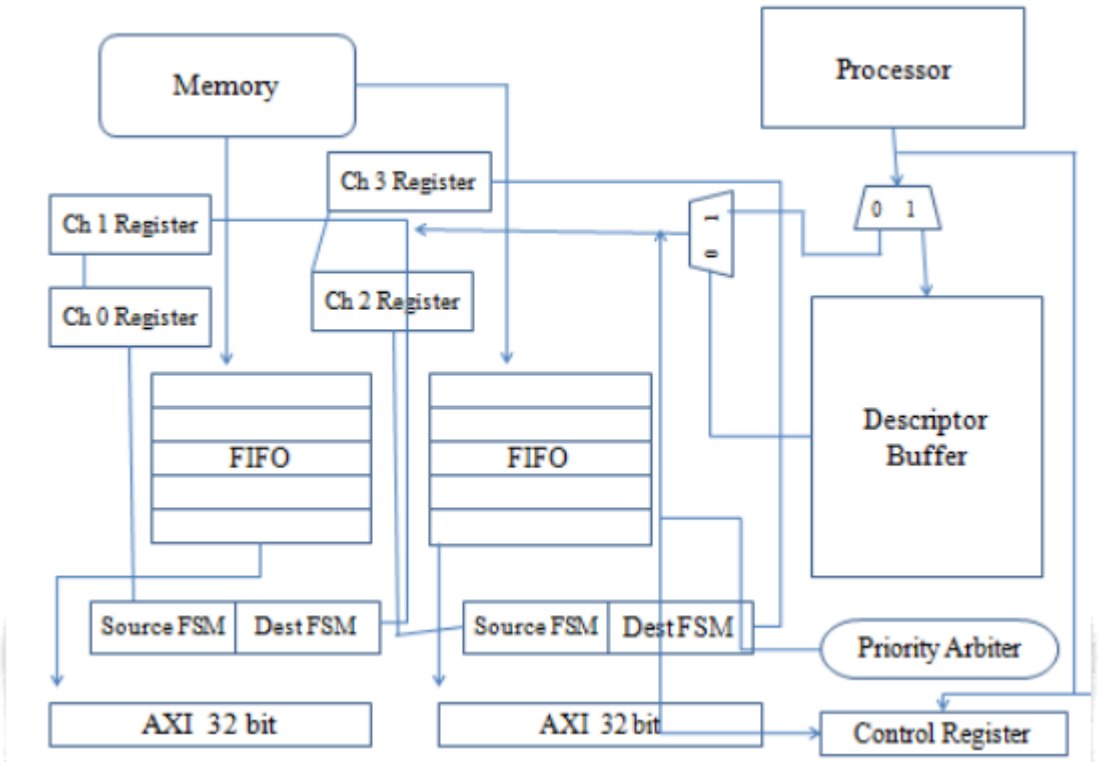

Figure 2.2.3.1: Datapath of DMAC

This design consists of 4 channels, arbiter module, channel registers, 3 AXI masters, one AXI slave, Asynchronous FIFO buffers. Again, the DMAC consists of 4 channel and each performs a specific data transmission type for example channel one is asked to transfer a series of data which might be addresses and channel two for normal data transfer, then they will do what that has been told and not be overlapped causing confusion on the receiving end. Although there are four AXI masters mentioned but there is only one that is associated with the AXI protocol. The two remaining AXI masters can be connected to the external memory or the Double Data Rate (DDR).

There are two modes that this DMA controller can run on: the first one is the data transfer mode and second one being the descriptor transfer mode. The first one is when the DMA controller transfer huge load of data and after that a second priority channel is selected. Whereas in the second mode, tasks are being sent to the channel descriptor by the processor ahead pf schedule. After this had happened, the arbitration scheme would select the highest priority channel which means that the transfer that happen first (data transfer) in the data descriptor will load the work registers and execution of related data transfer would be started. This is how data is being transferred in the DMA Controller without the interference and involvement of the processor.

Chapter 3: Proposed Method/ Approach

3.1 Methodologies and General Work Procedures

For this project's design process' digital system, there will be 3 types of design methodologies being used: firstly, it is the top-down design methodology followed by the bottom-up design methodology and finally the mixed design methodology. The top-down design methodology will be used for designing and developing the DMA controller unit. Next, the top-down design methodology, the top level-representation of a unit is first specified, lastly, lower-level representations based on different important cases such as speed, power consumption, silicon area and functionality.

3.1.1 RTL Design Flow

Register-transfer level (RTL) is a design summary used to represent a synchronous digital circuit in terms of digital signal flow between hardware registers, logical operations, etc, performed on those signals. In RTL design flow, the micro-architectural level design will be used for the main frame of the design because the DMA controller will be designed in unit-level. Using this design flow, a better representation of project progress and accuracy will be achieved.

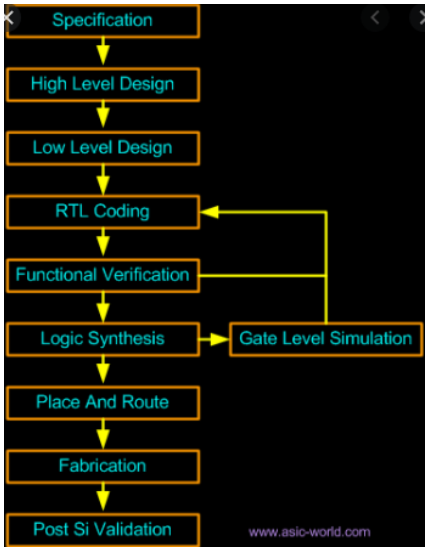

Figure 3.1.1.1: RTL design flow used to design DMAC

3.1.2 Micro-architecture Specification

This part of the project will give a detailed description of the internal design of the DMA controller. The internal design of the DMA controller will give be described in detail and also design-specific technical information will be provided for the design to be as accurate as possible. In this project, the unit level of the DMA controller will include the following information:

- Functionality and the feature description
- Interfaces and I/O pin description
- Functional partitioning into blocks and inter-blocks signalling
    - Additional blocks would be separated once a specific block is too complexed and would be divided into a combination of smaller blocks
- Test plan (functional test)

For the block level design of the DMA controller, this following description would be provided:

- Functionality and the feature description
- Interfaces and I/O pin description
- Internal operation: Finite-state machine (FSM), and etc,
- Schematic and block diagram
- Test plan ((Functional test)

3.1.3 RTL Modelling and Verification

After the micro-architecture specification has been came up with, the RTL coding of the DMA controller can begin. The RTL models will then be verified for functional correctness at each level after the coding has finished. Going deeper into the picture, each block of RTL model is

to be verified before they are integrated into unit level. In the development process of the project, the design does not meet the requirement of the DMA controller, the design flow would need to be repeated until everything is done right. After all the requirements are fulfilled, then this design would be brought over to FPGA technology to be implemented.

3.1.4 Logic synthesis for FPGA

After the DMA controller unit has been fully verified with its functionality, the model would be ready for logic synthesis where RTL code is to be translated into gate level representation. Based on the result, the gate level unit would need to be verified again with its functional correctness. The design is ready for the next phase, which the physical design phase when all the design has been fully verified with the appropriate logic. On the other hand, if it does not meet with the requirement then depending on the severity, would need to be repeated just like the previous step until it is all error free.

3.2 Design Tools

For each of the step mentioned above, from building the design to physical model would require both software and hardware tools. Because of all these procedures, there exists the Electronic Design Automation (EDA) tools for designing at abstraction level. Since the model for this current DMA controller design uses Verilog hardware description language (HDL), thus a Verilog simulator has to be used to emulate Verilog HDL. Here are some examples of Verilog simulators and comparison among themselves:

Simulator	Modelsim	VCS	CVC
Company	Mentor Graphics	SYNOPSYS® Predictable Success	TACHYON DESIGN AUTOMATION

Supported language	• VHDL-2002 • V2001 • SV2005	• VHDL-2002 • V2001 • SV2005	V2001
Platform availability	•Windows XP/Vista/7 • Linux	Linux	• Windows
Free	Yes	No	Yes

Table 3.2.1.: Comparison among Verilog Simulators

From the table above, it is quite apparent that Modelsim is the best choice among the three since in terms of available language supported and price. A free student edition license is available for everyone on the internet. It is also true that VCS offer more range of functionality and versatility but Modelsim is sufficient enough to support all that needs to be done for this project. Modelsim also runs on Windows platform for the majority and even runs on Linux for those who do not use a Windows operating system. We do not use VCS is also because it is too expensive (minimum $20,250 per year subscription) and for semi-professional work such as this project is really not needed.

As for the tools used for synthesis, there are several tools that can be used for logic synthesis. These can include Vivado Design suite by Xilinx, Encounter RTL Compiler by Cadence Design System, Quartus by Altera and many more. For this project, I will be using the Xilinx Vivado Design Suite because its FPGA supported functionality and it is freely available all over the internet.

3.2.1 ModelSim Student edition-x64 10.5

ModelSim from Mentor graphic is the industry-leading simulation and debugging tool for HDL-based design. Furthermore, its license can be obtained online for free. Although it only provides the free license for student version only instead of the full version, it is already sufficient enough because we do not need the full functionality of the software for this current project. We just needed the Verilog and the VHDL languages that is available in the student version of the said simulator. Furthermore, syntax errors are shown once it detects any syntax error in the compiled code. Waveform simulation play an important role in the designing procedure. Testbenches are also available together with the timing diagram for different input and output pins used for model verification.

3.2.2 Xilinx Vivado Design Suite

Vivado Design Suite is a software suite designed by Xilinx. The said software is used to synthesize and analyse the HDL designs. It gives the option to synthesize, perform timing analysis, examine RTL diagrams, verify and test the design. Configuration of the device is also available which makes it easier for users to complete the tasks.

3.2.3 PCSpim

PCSpim is the Window version of spim. This simulation tool loads and executes assembly language program for the MIPS RISC architecture. It also provides a simple assembler as well as a debugger and a simple set of operating services. Hence, this tool would be extremely useful in developing MIPS test program in order to verify the correctness of the interrupt Service routine (ISR).

3.4 Implementation Issues and Challenges

With just the logic analysis, the implementation of the DMA into RISC32 is do-able, with the introduction of clock signals and edge positive sensitive data, a more detailed testing needs to

be conducted. Passing signal and data safely from block with different timing requirement seemed to be quite challenging to achieve. The lack of real life example is also a major challenge encountered during this project therefore, not the most accurate result will be produced at the end.

3.5 Timeline

3.5.1 Gantt Chart for Project 1

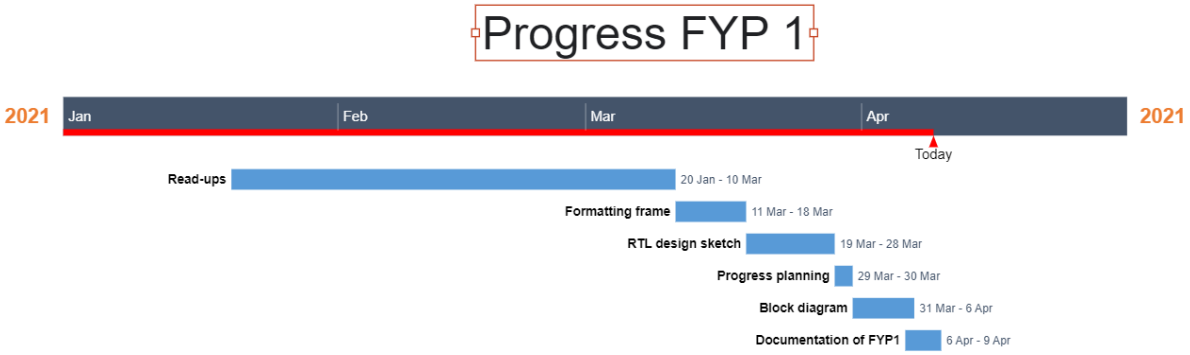

Table 3.5.1.1: Gantt chart for Project 1

3.5.2 Gantt Chart for Project 2

Progress FYP2

Task Name	Start (Date)	End (Date)	Duration (Days)
RTL Modelling Interface	14/1/2022	1/2/2022	18
RTL Modelling Datapath	1/2/2022	1/3/2022	28
RTL Modelling Timing Un	2/3/2022	11/3/2022	9
RTL Modelling Priority Ur	11/3/2022	24/3/2022	13
Intergration all units	24/3/2022	10/4/2022	17
DMAC functionality test	11/4/2022	20/4/2022	9
Documentation	20/4/2022	21/4/2022	1

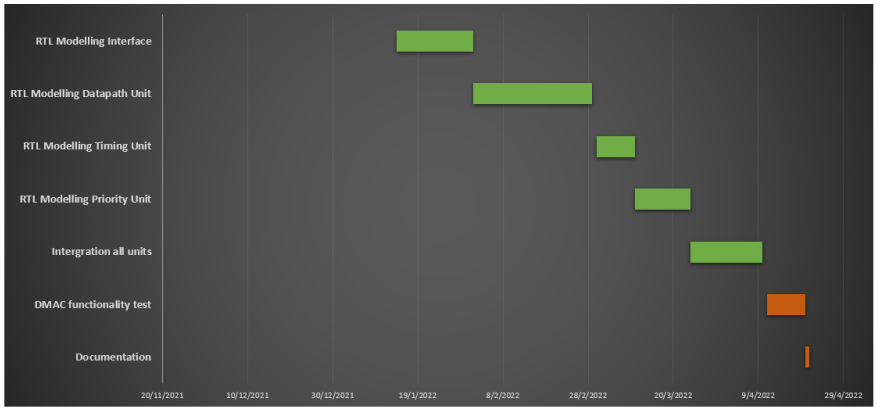

Table 3.5.1.2: Gantt chart for Project 2

Chapter 4 System Specification

4.1 System Overview of Risc32 Pipeline Processor

Since the usage of direct memory access function is not implemented inside the RISC32 pipeline processor, a DMAC would be implemented in this current project to reduce the workload of the processor during the read and write stage of the pipeline processing. The processor used in this project I developed by UTAR FICT because it provides the best software and firmware flexibility advantage for the direct memory access controller front-end design.

4.1.1 RISC32 Pipeline Processor Architecture

The processor is basically made up of three key components that is, the Central Processing Unit(CPU), the memory system(cache unit) and the input and output systems(Kiat, 2018). The developed processing unit was said to be able to run a 5-stage 32-bit MIPS instruction Set Architecture (ISA) and can support a maximum number of 49 instructions. This includes arithmetic, logics, program/ priority control, system instruction and data transfer. The memory system of the processor also consists of different components such as the data cache, instruction cache, stack RAM to store permanent instructions, boot ROM for executing instructions during the startup to load the operating system, and flash memory which stores temporary data. As for the I/O system, it is made up of several parts such as the SPI controller, UART controller, the GPIO controller, and the priority interrupt controller. For the few controllers mentioned above, they are to control various functions like data transfer with sensors, computers, and wireless parts. But for priority interrupt, its function is as it sounds, providing interrupts and priority over an events priority level. The already developed architectural overview on the RISC 32 processor is shown in figure 4.1.1.1 together with the specification provided in table 4.1.1.1

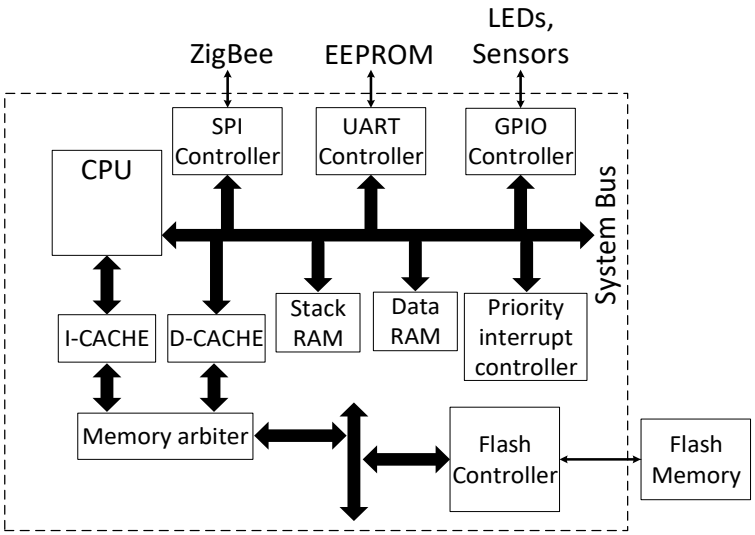

Figure 4.1.1.1: An overview on the architecture of the RISC32 pipeline processor.

Feature	**MIPS Compatible 5-stage pipelined processor**
Data width	32bits
Instruction width	32bits
General Purpose Register	32
Special Purpose Register	hi, lo, pc
Pipeline Stage	5
Clock per Instruction Stage	1
Interlock Handling	No
Data Dependency Forwarding	No

Branch Prediction	No
Multiplication (size of multiplier)	Yes(32bits)
Instruction Supported	28
Floating-point Instruction Supported	No
Memory	512 bytes

Table 4.1.1.1: Specification of the RISC32 pipeline

4.1.2 Functionality of RISC32 Pipeline Processor

Divide execution of instruction into following 5 stages, allow up to 5 instructions to run concurrently:

- IF (Instruction Fetch) Fetch instruction from instruction cache into the datapath.
- ID (Instruction Decode) Decode instruction and fetch $rs & $rt registers.
- EX(Execute) Execute instruction in the ALU.
- MEM(Memory) Access data cache, load or store.
- WB (Write Back) Write back the result to the register file.

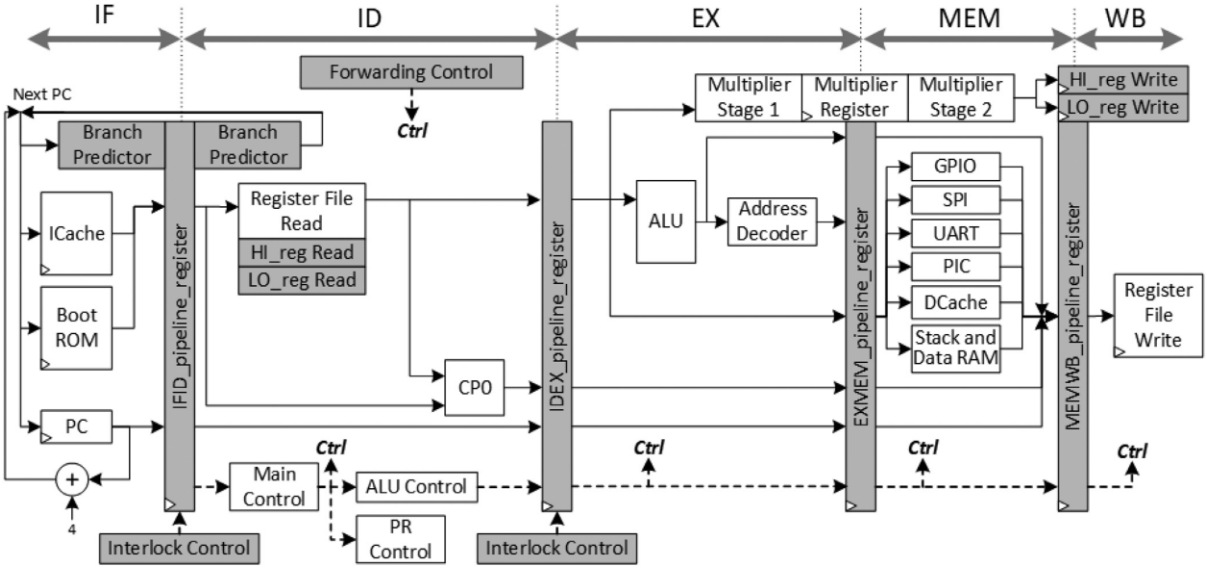

Figure 4.1.2.1: functionality of MIPS architecture

4.1.3 Memory Map of the RISC32 Pipeline Processor

Two different ways are applied in the RISC32 pipeline processor, they are called the physical and virtual addresses each with their own purposes. The physical address is used to allocate physical memory into different things as flash memory, the ROM and the RAM. Whereas the purpose of virtual address is used to access instruction program and data. The figure in 4.1.3.1 below shows the memory map of the RISC32 processor.

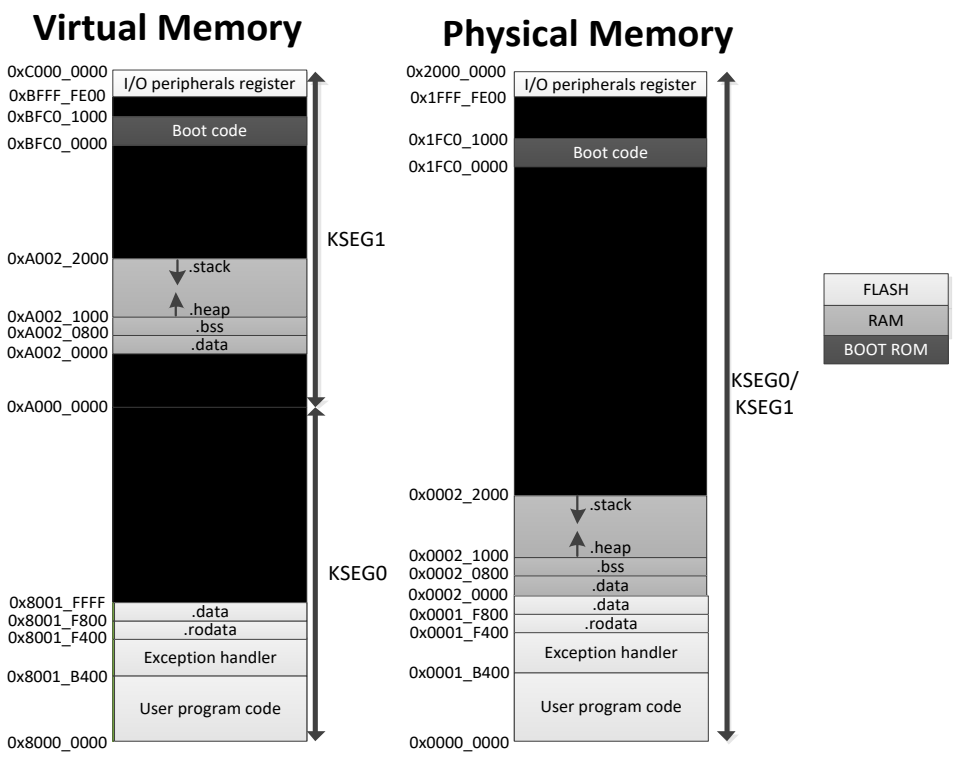

Figure 4.1.3.1: Memory map of the RISC32 pipeline processor.

Memory Usage	Description	Memory Size
I/O peripheral register	Used as the memory-mapped registers for I/O peripheral controllers.	512 bytes
Boot code	Used to store bootloader program code for initial system configuration when powered on.	4k bytes
Stack	Used by procedure during execution to store register values.	8k bytes
Heap	Used to hold variables declared dynamically.	
Exception handler	Used to store the exception handler codes.	16k bytes
User program code	Used to store user program codes	128k bytes

Table 4.1.3.1: Memory map description of the RISC32 pipeline processor.

4.2 Chip Interface of the RISC32 Pipeline Processor

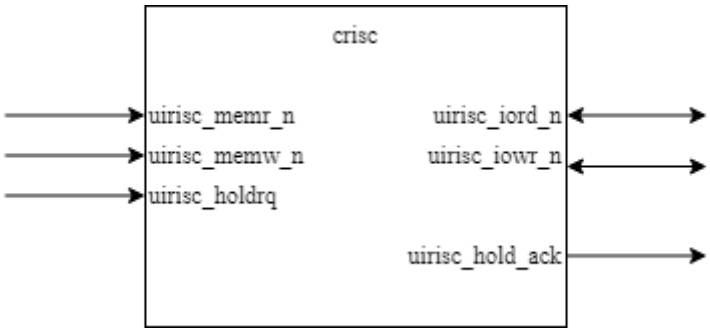

Figure 4.2.1: Chip interface of the RISC32 pipeline processor needed for DMAC.

4.2.1 Input Description of the Chip Interface of the RISC32 Pipeline Processor

Pin name: uirisc_memr_n **Pin class:** Data **Source → Destination:** DMAC → crisc **Pin function:** Memory Read signal, active low when underdoing the DMA Read or a memory-to-memory transfer
Pin name: uirisc_memw_n **Pin class:** Data **Source → Destination:** DMAC → crisc **Pin function:** Memory write signal, active low when underdoing the DMA Write or a memory-to-memory transfer
Pin name: uirisc_holdrq **Pin class:** Data **Source → Destination:** DMAC → crisc **Pin function:** Receive signal from the DMA controller for control of the system bus. Only performs it functions when dreq occurs and corresponding mask bit is clear.

4.2.2 Output Description of the Chip Interface of the RISC32 Pipeline Processor

Pin name: uirisc_hold_ack **Pin class:** Data **Source → Destination:** crisc → DMAC **Pin function:** A signal send to the DMA controller after bus control has been relinquished by the processor and permission for the DMA controller to control the busses has been granted.

4.2.3 Input Output Description of the Chip Interface of the RISC32 Pipeline Processor

Pin name: uirisc_iord_n	**Pin class:** Data
Source → Destination: DMAC ↔ crisc	
Pin function: Input control when used by CPU to read control registers and output signal used to access data from peripheral during DMA Write transfer	
Pin name: uirisc_iowr_n	**Pin class:** Data
Source → Destination: DMAC ↔ crisc	
Pin function: Input control when used by CPU to load information into DMAC and output signal used to load data to peripheral during DMA Read transfer	

Chapter 5: Micro-Architecture Specification

5.1 DMA Controller Unit

5.1.1 Functionality/ Feature of the DMA Controller Unit

The DMA controller is made to improve the rate of data transfer between the I/O device to the memory, or a block of data into an I/O device. It can also perform a memory-to-memory block moves, which fills up a block with data in specific locations. Different modes of operation can make a single byte transfer and discontinuous data streams. The DMA controller is state driven, signal generating device that permits the movement of data from the I/O devices to memory spaces and vice versa without use of temporary registers within the processor. This will iprove the rate of transfer for sequential operations compared to transferring data using the temporary registers in the processor also known as processor move or repeated string instructions. The longest operation undergone by the DMA controller is the memory-to-memory transfer, since it requires the temporary internal storage of data byte between generation of the source and destination addresses. However, this transfer still takes much faster than the typical central processor techniques.

5.1.2 Operating Procedure (External Operation)

The data bus. addresses and control outputs pins of the DMA are connected in parallel to the system busses. This causes a need for the use for an external latch for the upper address byte. The output of the controller will be in a high impedance state while not activated. But after activation by a DMA request, bus control would be taken away by the host, now the DMA would drive the busses and generate control signal to take over the process of data transfer. The different operations that activate the DMA request are programmed and given out through Command, Mode, Address and Word Count registers. Here are the steps of the modes of transfers within the DMA transfer.

Basic transfer of data from RAM to I.O device:

1. Starting address loaded into the DMA controller into the Current and Base Address registers.
2. Length of the block is loaded into the Word Count register.

3. Mode register would then programme the DMA for a memory-to-I/O transfer (read transfer) together with the Command register.
4. The channel ‘s mask bit would be clear to recognize the DMA request.
5. The DMA transfer will then start as the controller outputs data address and at the same time, active low Memory Write and active low I/O Write pulses and select the specific I/O device through DMA acknowledgement.
6. The data passes from RAM to I/O device directly.
7. Address automatically increase or decrease an word count will be decremented.
8. Operation repeats for the next byte.
9. Operation stops when word count reaches zero or an active low End of Process (EOP) is applied.

The states generated by each of the clock cycles must be considered to fully understand the DMA controller’s operation. There are two main clock cycles, the active and the idle cycle. The DMAC will usually be in an idle cycle until DMA request is received on an unmasked channel. The DMA will then request control of the busses and go into an active cycle. There are several states in the active cycle and the event states will be triggered by the required operation.

5.1.3 Unit Interface of the DMA controller

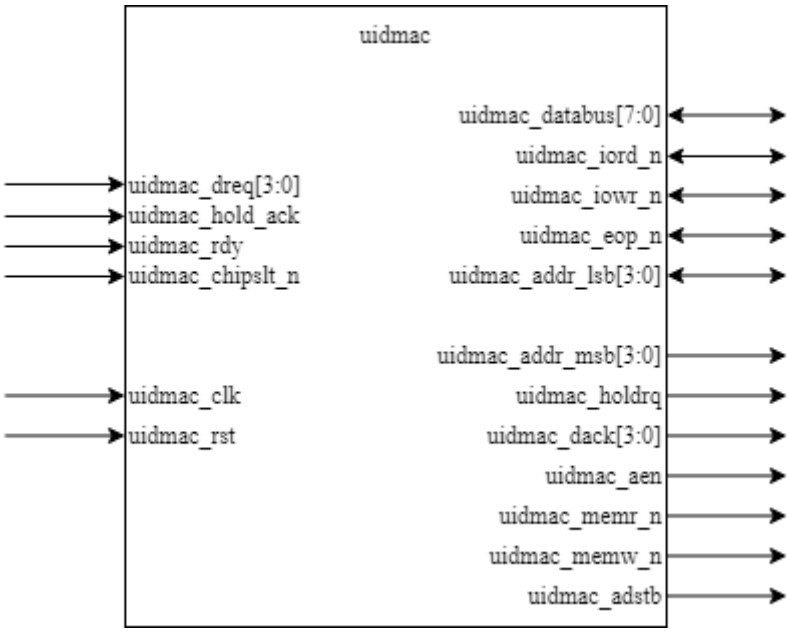

Figure 5.1.3.1: DMA controller unit interface.

5.1.4 Input Pin Description of the DMA Controller Unit

Pin name: uidmac_dreq[3:0] **Pin class:** Data **Source → Destination:** I/O device → DMA controller unit **Pin function:** Activate the DMA if it is in the right state. DREQ0 → DREQ3 – Highest → Lowest Priority Must be maintained until DACK goes active.
Pin name: uidmac_hold_ack **Pin class:** Data **Source → Destination:** CPU → DMA controller unit **Pin function:** Sent from CPU to notify that CPU has already relinquished its control of the system busses. Synchronous means that only activate at the rising edge of the clock cycle.
Pin name: uidmac_rdy **Pin class:** Control **Source → Destination:** CPU → DMA controller unit **Pin function:** Send to extend rea write operation for slow I/O operations.
Pin name: uidmac_chipslt_n **Pin class:** Control **Source → Destination:** CPU → DMA controller unit **Pin function:** Enable controller onto data bus for communications with the CPU.
Pin name: uidmac_clk **Pin class:** Global **Source → Destination:** Global Clock → DMA controller unit **Pin function:** Global Clock
Pin name: uidmac_rst **Pin class:** Global **Source → Destination:** Global Reset → DMA controller unit **Pin function:** Global Reset

Table 5.1.4.1: Input pin description of the DMA controller unit

5.1.5 Output Pin Description of the DMA Controller Unit

Pin name: uidmac_addre_msb[3:0] **Pin class:** Data **Source → Destination:** DMA controller unit → Memory Storage Unit **Pin function:** most significant bit of the 8-bit address line. Three-state outputs and provide four bits of address. Only enabled during a DMA service.
Pin name: uidmac_holdrq **Pin class:** Data **Source → Destination:** DMA controller unit → CPU **Pin function:** Request sent to the CPU the request control of the system bus.
Pin name: uidmac_dack[3:0] **Pin class:** Data **Source → Destination:** DMA controller unit → I/O device

Pin function: Send and acknowledgement to I/O device that it had been granted the DMA cycle whereby data transfer process would start.
Pin name: uidmac_aen **Pin class:** Control **Source → Destination:** DMA controller unit → address bit latch **Pin function:** Enable address bit to be transferred from the DMAC to the latch.
Pin name: uidmac_memr_n **Pin class:** Control **Source → Destination:** DMA controller unit Clock → Memory Storage Unit **Pin function:** Signal given out to access memory location during DMA Read or memory-to-memory transfer.
Pin name: uidmac_memw_n **Pin class:** Control **Source → Destination:** DMA controller unit → Memory Storage Unit **Pin function:** Signal given to write data to the selected memory location during DMA Write or memory-to-memory transfer.
Pin name: uidmac_adstb **Pin class:** Control **Source → Destination:** DMA controller unit → memory latch **Pin function:** strobe input and external latches, speeding up the opearation within the DMA.

Table 5.1.5.1: Output pin description of the DMA controller unit

5.1.6 Input Output Pin Description of the DMA Controller Unit

Pin name: uidmac_databus[7:0] **Pin class:** Data **Source → Destination:** DMA controller unit ↔ Memory, I/O devices **Pin function:** Three-state signals connected to the system data bus. Outputs are enabled during I/O Read to output contents of registers to the CPU. Memory enter data bus during read-from-memory transfer, but during the write-to-memory transfer, the data bus outputs the data into a new memory location.
Pin name: uidmac_iord_n **Pin class:** Control **Source → Destination:** DMA controller unit ↔ I/O devices **Pin function:** Idle cycle: input control signal used by CPU to read control register Active cycle: access data during DMA Write transfer
Pin name: uidmac_iorw_n **Pin class:** Control **Source → Destination:** DMA controller unit ↔I/O devices **Pin function:** Idle cycle: input control signal used by CPU to load information into DMAC Active cycle: output control signal used by DMAC to load data during a DMA Read transfer.
Pin name: uidmac_eop_n **Pin class:** Control **Source → Destination:** DMA controller unit ↔ on-chip transistor **Pin function:** Terminates an active DMA service. Will be active also when any register reached its terminal count.
Pin name: uidmac_addr_lsb[3:0] **Pin class:** Data

Source → Destination: DMA controller unit Clock ↔Memory Storage Unit **Pin function:** Idle cycle: the address would be to notify which register is to be loaded or read. Active cycle: output the lower 4-bit address of the output address.

Table 5.1.6.1: Input Output pin description of the DMA controller unit

5.1.7 Finite State Machine of DMA controller operations

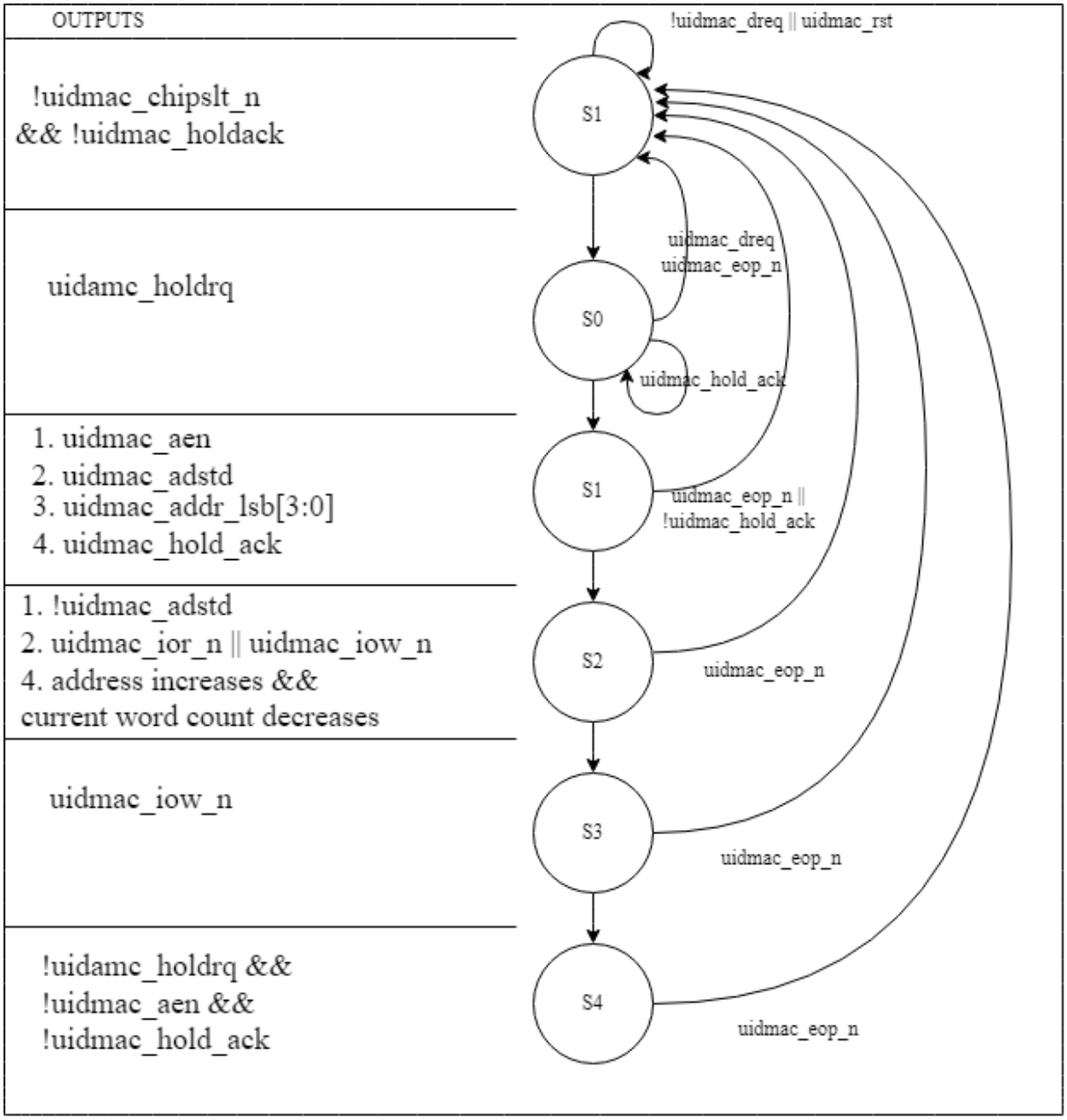

Figure 5.1.7.1 Finite state machine of DMA controller unit

5.1.8 Design Partitioning of the DMA Controller unit

The DMA controller unit in this project consists of three different internal units that contain other registers that work together so that the data transfer operation between the memory spaces and I/O device work correctly when DMA request is received by the DMA itself. The DMAC unit consists of one, Datapath unit, one priority unit and one timing and control unit as shown in figure 5.1.8.1. Table 5.1.8.2 shows the internal register used the unit mentioned above. The function of each register will be discussed further in section 5.2 regarding the Datapath unit. The functionality of each unit is also shown in Table 5.1.8.1.

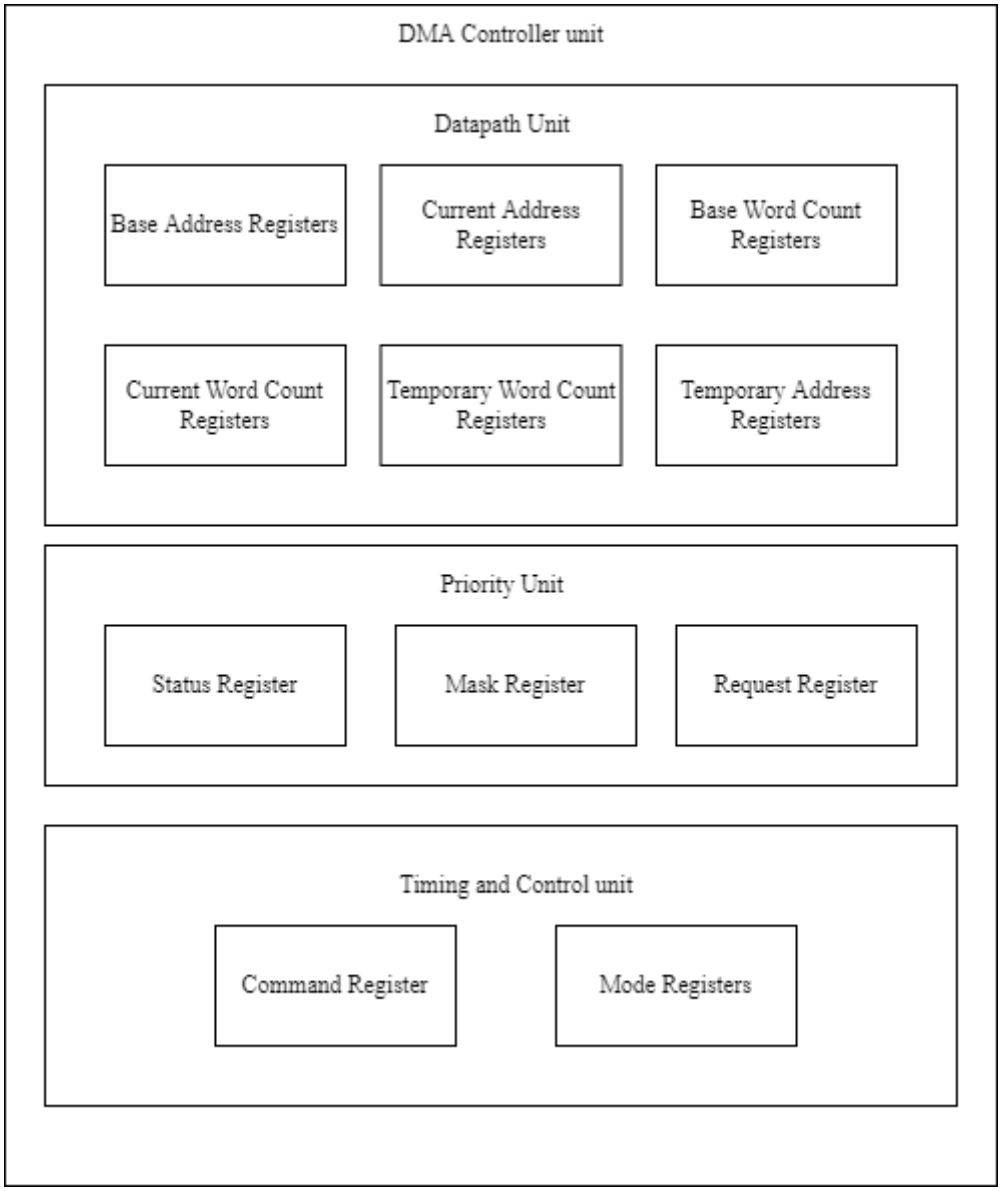

Figure 5.1.8.1: Block-level partitioning of the DMA controller unit

Internal Unit	**Function**
Datapath Unit	Handles all the operation include ALU, registers and busses.
Timing and Control Unit	Handles timing and edge sensitive signals to make sure everything is synchronous when required.
Priority Unit	Handles interrupts and priorities of operations within the DMA controller

Table 5.1.8.1: Functionality of DMA controller unit's components

Name	Size	Number
Base Address Register	16-Bits	4
Base Word Count Registers	16-Bits	4
Current Address Register	16-Bits	4
Current Word Count Registers	16-Bits	4
Temporary Address Register	16-Bits	1
Temporary Word Count register	16-Bits	1
Status Register	8-Bits	1
Command register	8-Bits	1
Temporary Register	8-Bits	1
Mode Registers	6-Bits	4
Mask Register	4-Bits	1
Request Register	4-Bits	1

Table 5.1.8.2: Internal registers of the DMA controller

5.1.9 Micro-Architecture of the DMA Controller Unit (Block Level)

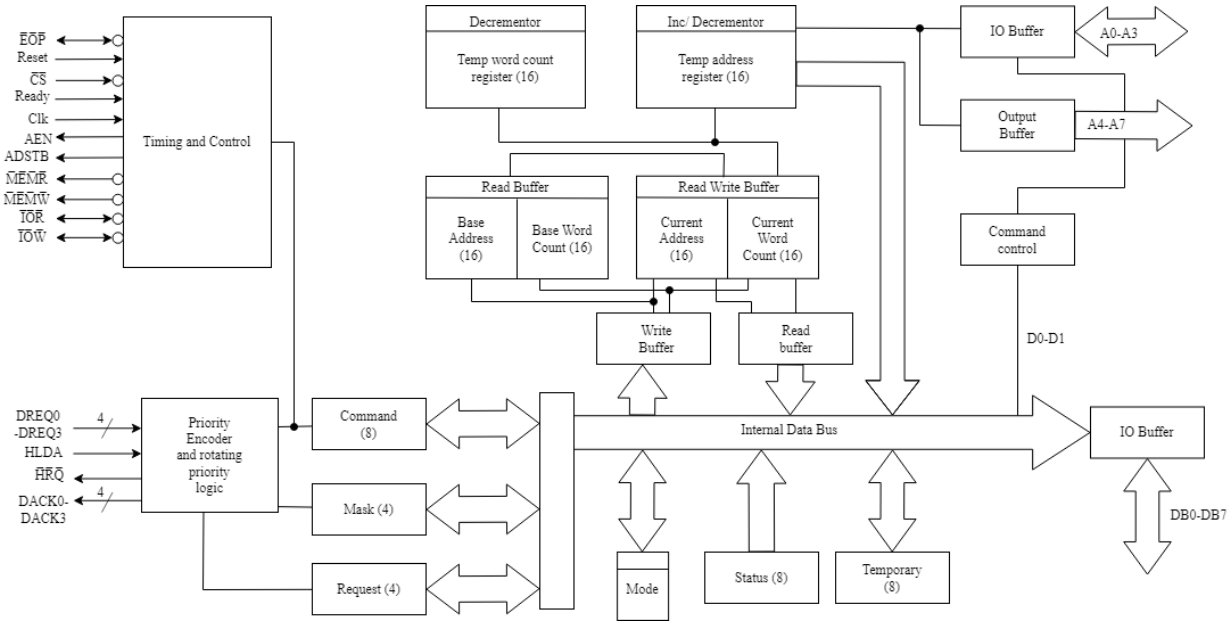

Table 5.1.9.1: Simplified Micro-architecture of the DMA controller unit

5.2 DMAC Datapath Unit

5.2.1 Functionality/Feature of the Datapath Unit

The Datapath Unit is used to handle all arithmetic operation including the increment/ decrement of word count, the increment / decrement of address, the movements in the internal data busses. It consists of several internal registers as shown in Table 5.2.1.1.

Internal Registers	Bit size	Description
Base Address Register	16	Store the original value of the address. The value will then be restored during the Autoinitialize operation after any increment or decrement made to the addresses. The register is written simultaneously with the Base Word Count register.
Current Address Registers	16	Holds the value of the address needed during the DMA transfers. The address would be automatically incremented or decremented after each transfer and stored back to the register.
Base Word Count Registers	16	Store the original value of the Word Count. The value will then be restored during the Autoinitialize operation after any increment or decrement made to the addresses. The register is written simultaneously with the Base Address register.
Current Word Count Registers	16	This register tells the number of transfers left needed to be performed. But due to logical reasons, the number of transfers needed to be one extra because for 100 transfers, the word count has to be 101 so that it actually transfer a hundred times instead of 99 times.
Temporary Word Count Registers	16	Temporarily hold word count of a memory-to-memory transfer. So, it will be usually left with the value of the last transfer unless clearde by a master reset.

Temporary Address Registers	16	Temporarily hold address of a memory-to-memory transfer. So, it will be usually left with the value of the last transfer unless cleared by a master reset.

Table 5.2.1.1: Internal registers of DMAC Datapath

5.2.2 Block Interface of the DMAC Datapath Unit

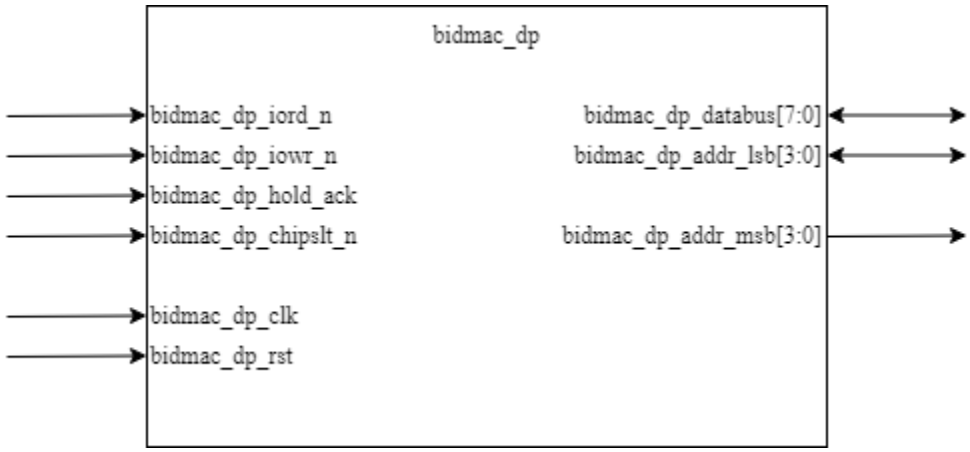

Figure 5.2.2.1: Block interface of the DMAC Datapath Unit.

5.2.3 Input Pin Description of the DMAC Datapath Unit

Pin name: bidmac_dp_iord_n **Pin class:** Control **Source → Destination:** bi_ctrl_time → bidmac_dp **Pin function:** access data during DMA Write transfer during the active cycle
Pin name: bidmac_dp_iorw_n **Pin class:** Control **Source → Destination:** bi_ctrl_time → bidmac_dp **Pin function:** load data during a DMA Read transfer in the active cycle
Pin name: bidmac_dp_hold_ack **Pin class:** Data **Source → Destination:** CPU → bidmac_dp **Pin function:** Sent from CPU to notify that CPU has already relinquished its control of the system busses. Synchronous means that only activate at the rising edge of the clock cycle..
Pin name: bidmac_dp_chipslt_n **Pin class:** Control **Source → Destination:** CPU → bidmac_dp **Pin function:** Enable controller onto data bus for communications with the CPU.
Pin name: bidmac_dp_clk **Pin class:** Global **Source → Destination:** Global reset → bidmac_dp

Pin function: Global clock.
Pin name: bidmac_dp_rst **Pin class:** Global **Source → Destination:** Global reset → bidmac_dp **Pin function:** Global reset.

5.2.4 Output Pin Description of the DMA Controller Unit

Pin name: bidmac_dp_addr_msb[3:0] **Pin class:** Control **Source → Destination:** bidmac_dp → memory **Pin function:** most significant bit of the 8-bit address line. Three-state outputs and provide four bits of address. Only enabled during a DMA service.

5.2.5 Input Output Pin Description of the DMA Controller Unit

Pin name: bidmac_dp_databus [7:0] **Pin class:** Data **Source → Destination:** bidmac_dp ↔ Other registers **Pin function:** Three-state signals connected to the system data bus. Outputs are enabled during I/O Read to output contents of registers to the CPU. Memory enter data bus during read-from-memory transfer, but during the write-to-memory transfer, the data bus outputs the data into a new memory location.
Pin name: bidmac_dp_addr_lsb[3:0] **Pin class:** Data **Source → Destination:** DMA controller unit ↔ I/O devices **Pin function:** Idle cycle: the address would be to notify which register is to be loaded or read. Active cycle: output the lower 4-bit address of the output address.

5.3 DMAC Timing and Control Unit

5.3.1 Functionality/Feature of the Timing and Control Unit

Internal Registers	Bit size	Description
Command register	8	Controls the operation of the DMA controller. Figure 5.3.1.1 will show the details for each pin commands.
Mode Register	6	When the register is being written to, the different bits determine which channel Mode register is to be written. Table 5.3.1.2 will show the operation and register codes for the pins.

Table 5.3.1.1: Internal registers of DMAC Timing and Control Unit

Operation	A3	A2	A1	A0	IOR	IOW
Read status Register	1	0	0	0	0	1
Write Command Register	1	0	0	0	1	0
Read Request Register	1	0	0	1	0	1
Write Request Register	1	0	0	1	1	0
Read Command Register	1	0	1	0	0	1
Write Single Mask Bit	1	0	1	0	1	0
Read Mode Register	1	0	1	1	0	1
Write Mode Register	1	0	1	1	1	0
Set First/Last F/F	1	1	0	0	0	1
Clear First/Last F/F	1	1	0	0	1	0
Read Temporary Register	1	1	0	1	0	1
Master Clear	1	1	0	1	1	0
Clear Mode Reg. Counter	1	1	1	0	0	1
Clear Mask Bits	1	1	1	0	1	0
Read All Mask Bits	1	1	1	1	0	1
Write All Mask Bits	1	1	1	1	1	0

Table 5.3.1.2: Command and Register Codes

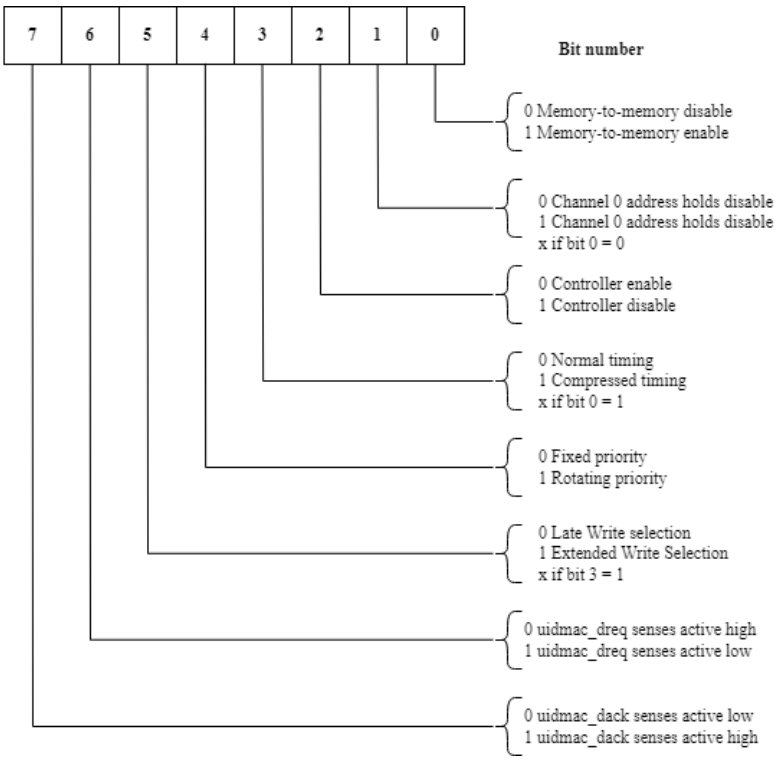

Figure 5.3.1.1: Addresses and Description of each Pin Number for Command Register

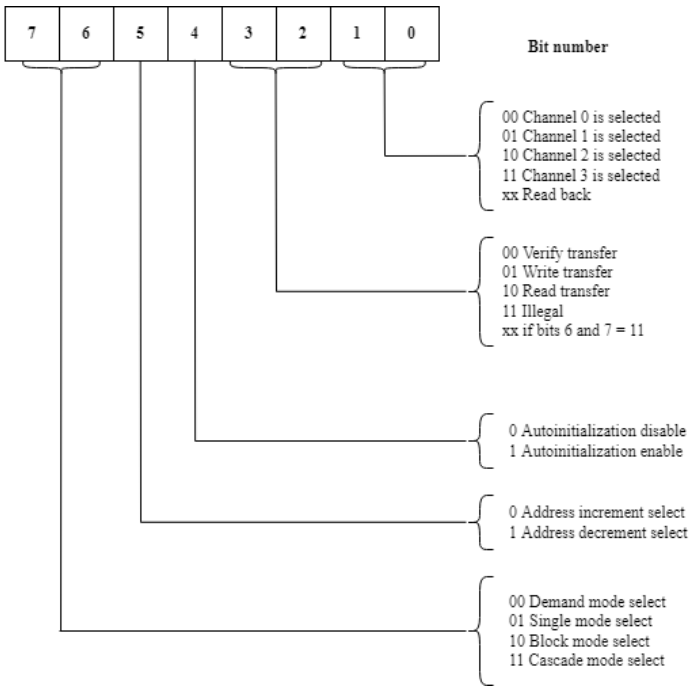

Figure 5.3.1.2: Addresses and Description of each Pin Number for Mode Register

5.3.3 Block Interface of the DMAC Timing and Control Unit

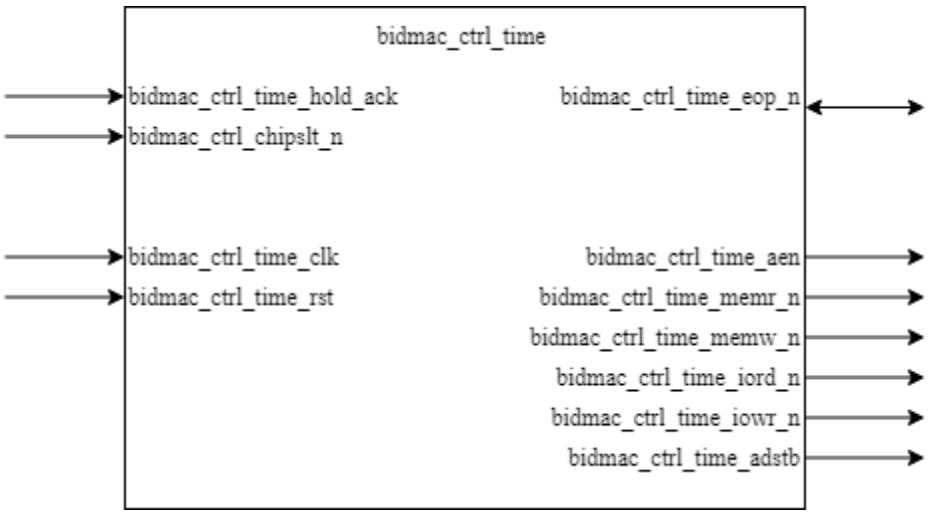

Figure 5.3.3.1: Block Interface of the DMA Timing and Control Unit.

5.3.4 Input Pin Description of the Timing and Control Unit

Pin name: bidmac_ctrl_time_hold_ack **Pin class:** Control **Source → Destination:** CPU → bidmac_ctrl_time **Pin function:** Sent from CPU to notify that CPU has already relinquished its control of the system busses. Synchronous means that only activate at the rising edge of the clock cycle.
Pin name: bidmac_ctrl_time_chipslt_n **Pin class:** Control **Source → Destination:** CPU → bidmac_ctrl_time **Pin function:** Enable controller onto data bus for communications with the CPU.
Pin name: bidmac_ctrl_time _clk **Pin class:** Global **Source → Destination:** Global reset → bidmac_ctrl_time **Pin function:** Global clock.
Pin name: bidmac_ctrl_time _rst **Pin class:** Global **Source → Destination:** Global reset → bidmac_ctrl_time **Pin function:** Global reset.

5.3.5 Output Pin Description of the Timing and Control Unit

Pin name: bidmac_ctrl_time _aen **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → address bit latch **Pin function:** Enable address bit to be transferred from the DMAC to the latch.
Pin name: bidmac_ctrl_time _memr_n **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → Memory Storage Unit **Pin function:** Signal given out to access memory location during DMA Read or memory-to-memory transfer.
Pin name: bidmac_ctrl_time _memw_n **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → Memory Storage Unit **Pin function:** Signal given to write data to the selected memory location during DMA Write or memory-to-memory transfer.
Pin name: bidmac_ctrl_time _iord_n **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → bidmac_dp **Pin function:** access data during DMA Write transfer during the active cycle
Pin name: bidmac_ctrl_time _iorwr_n **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → bidmac_dp **Pin function:** load data during a DMA Read transfer in the active cycle
Pin name: bidmac_ctrl_time _adstb **Pin class:** Control **Source → Destination:** bidmac_ctrl_time → Memory Latch **Pin function:** strobe input and external latches, speeding up the opearation within the DMA.

5.3.6 Input Output Pin Description of the Timing and Control Unit

Pin name: bidmac_ctrl_time _eop_n	**Pin class:** Control
Source → Destination: bidmac_ctrl_time → on-chip transistor	
Pin function: Terminates an active DMA service. Will be active also when any register reached its terminal count.	

5.4 DMAC Priority Unit

5.4.1 Functionality/Feature of the Priority Unit

Internal Registers	Bit size	Description
Status Register	8	Can be read out of the DMAC by processor. Tells the status of the devices at that point of time, specifically if the device has reached its terminal count and which channel still has a pending DMA request. Channel 0-3 is set when EOP is detected. Whereas channel 4-7 are set when corresponding devices requests DMA service. Register will be shown in figure 5.4.1.1
Mask Register	4	Registers are set to disable incoming DMA request. Each associated channel produces EOP signal if the channel is not Autoinitialized. Register will be shown in figure 5.4.1.2
Request Register	4	The DMA controller responds to commands as well as a DMA request. Each channel has a request bit that is related to the 4-bit request register. Register will be shown in figure 5.4.1.3

Table 5.4.1.1: Internal registers of DMAC Priority Unit

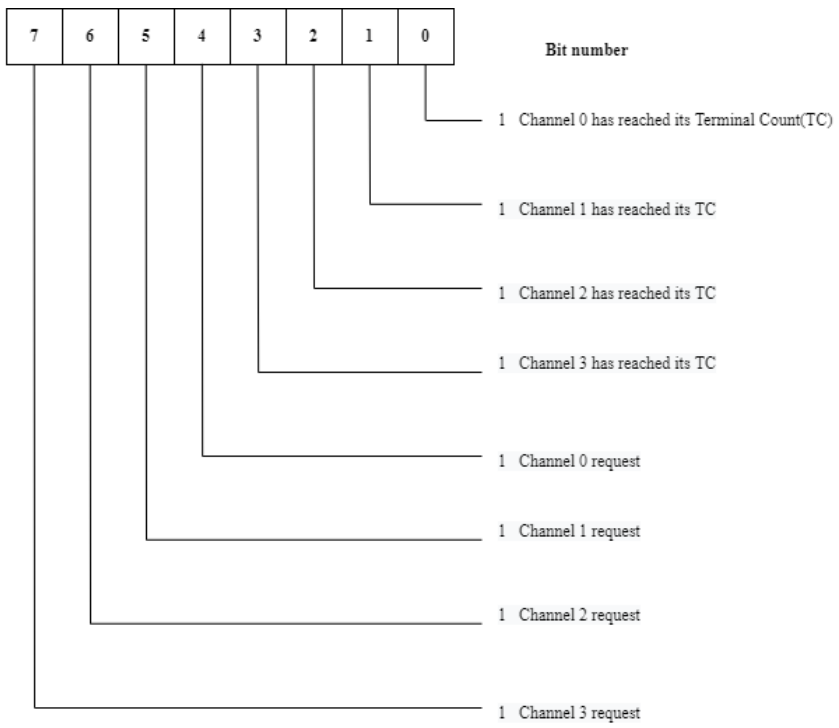

Figure 5.4.1.1: Addresses and Description of each Pin Number for Status Register

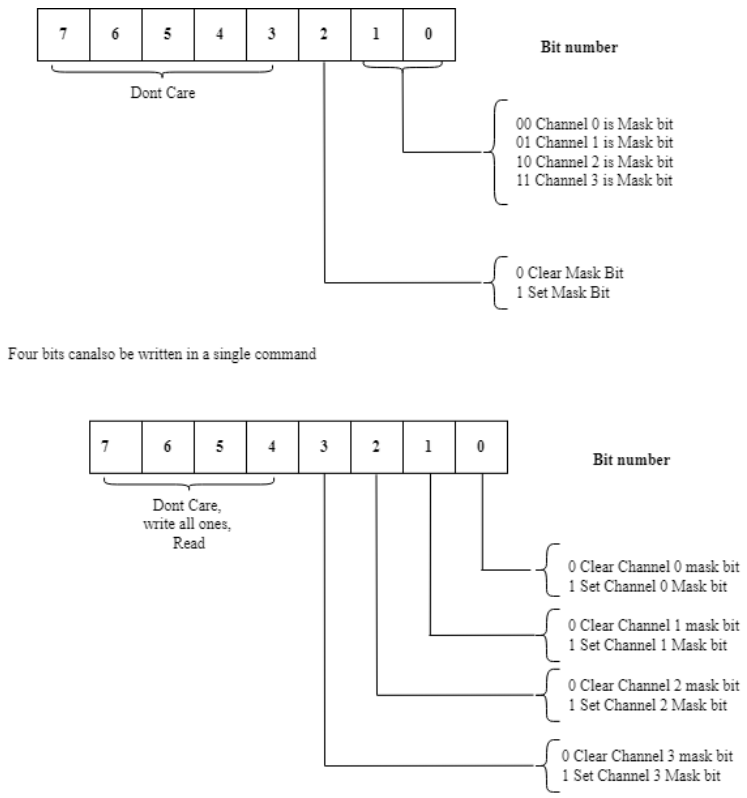

Figure 5.4.1.2: Addresses and Description of each Pin Number for Mask Register

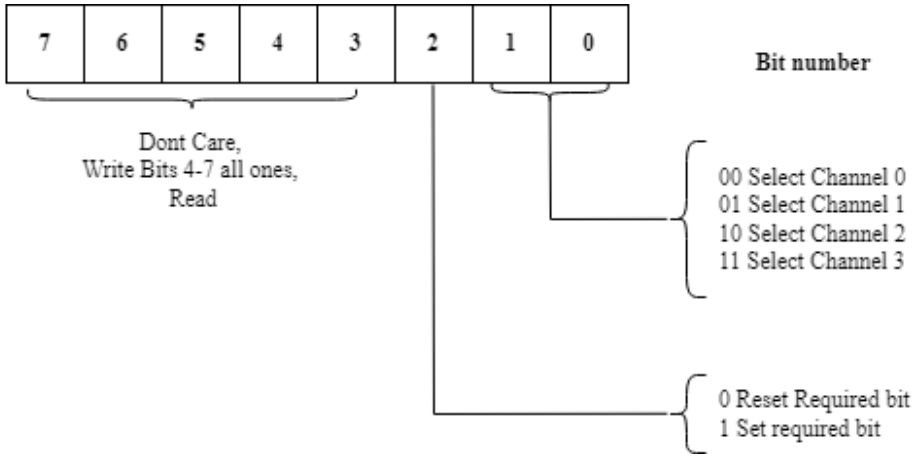

Figure 5.4.1.3: Addresses and Description of each Pin Number for Request Register

5.4.3 Block Interface of the DMAC Priority Unit

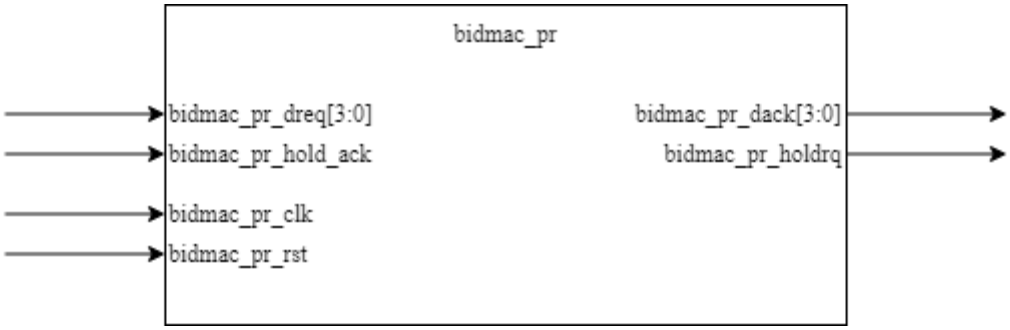

Figure 5.4.3.1: Block Interface of the DMAC Priority Unit

5.3.4 Input Pin Description of the DMAC Priority Unit

Pin name: bidmac_pr_hold_ack **Pin class:** Control **Source → Destination:** CPU → bidmac_pr **Pin function:** Sent from CPU to notify that CPU has already relinquished its control of the system busses. Synchronous means that only activate at the rising edge of the clock cycle.
Pin name: bidmac_ctrl_time_chipslt_n **Pin class:** Control **Source → Destination:** CPU → bidmac_pr **Pin function:** Enable controller onto data bus for communications with the CPU.
Pin name: bidmac_ pr _clk **Pin class:** Global **Source → Destination:** Global reset → bidmac_pr **Pin function:** Global clock.
Pin name: bidmac_ pr _rst **Pin class:** Global **Source → Destination:** Global reset → bidmac_pr

Pin function: Global reset.

5.3.5 Output Pin Description of the DMAC Priority Unit

Pin name: bidmac_ pr _dack[3:0] **Pin class:** Data **Source → Destination:** bidmac_ctrl_time → address bit latch **Pin function:** Enable address bit to be transferred from the DMAC to the latch.
Pin name: bidmac_ pr _holdrq **Pin class:** Data **Source → Destination:** bidmac_ctrl_time → Memory Storage Unit **Pin function:** Signal given out to access memory location during DMA Read or memory-to-memory transfer.

CHAPTER 6: VERIFICATION SPECIFICATION AND STMULATION RESULT

6.1 Test Plan for DMA Controller

Test	Expected Output	Status
Test Case #1: System Reset **<u>Function to be tested</u>** • Able to reset the whole DMA controller unit **<u>Procedure</u>** 1. Reset both devices.	• bidmac_dp_iord_n=1'hx • bidmac_dp_iorw_n=1'hx • uidmac_holdrq=1'h0 • uidmac_dack[3:0]= 4'hf • uidmac_dreq[3:0]=4'hf • commandReg=8'h00 • requestReg=8'h00 • maskReg=8'h00 • statusReg=8'h00 • tempReg=8'h00 • tempAddrReg=8'h00 • tempWordReg=8'h00	Pass

CHAPTER 7: SYNTHESIS AND IMPLEMENTATION

7.1 DMA Interface With RISC32 Processor

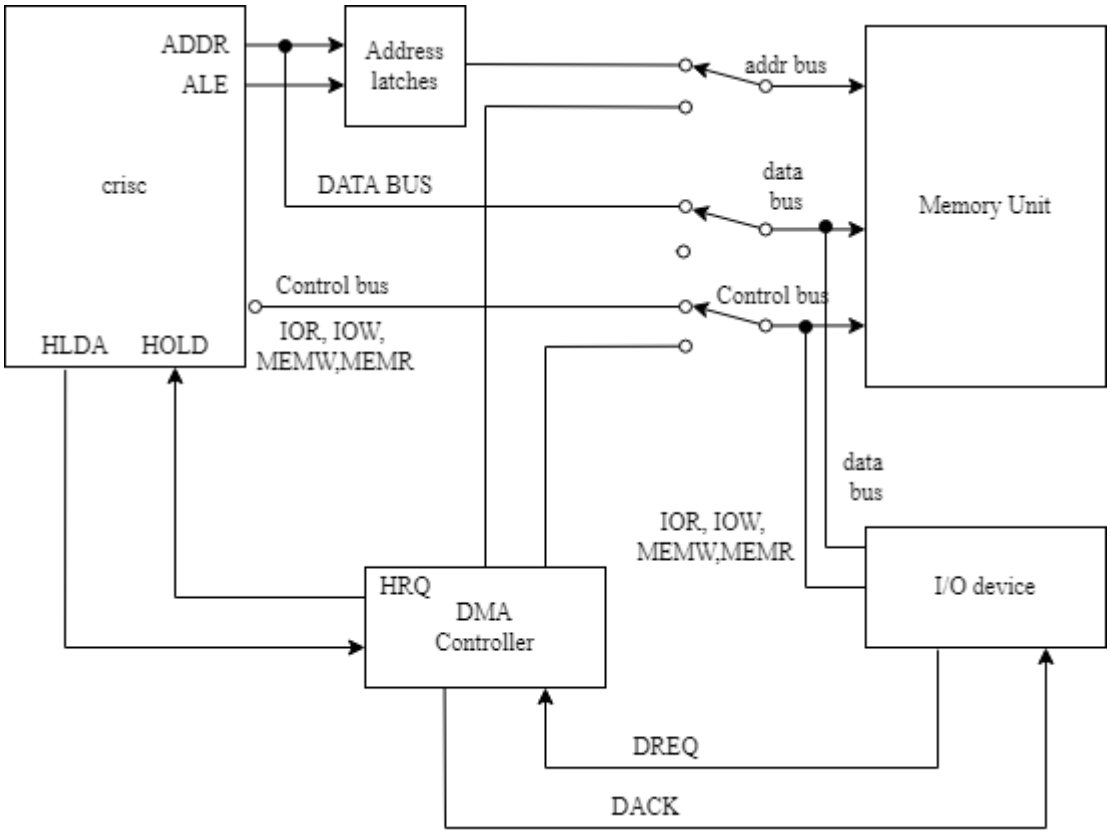

Figure 7.1.1: DMA Interface With RISC32 Processor

As shown above, the DMA request(DREQ) will be generated by the I/O device. After receiving the signal, the DMA Controller will give out the Hold request (HREQ) to the processor. The access to system bus by the DMA will not be granted until a Hold Acknowledge signal returns from the processor. After the signal is received, addresses and control signals are created by the DMA controller to start the DMA transfer operation. The data will eb delivered directly from I/O device to the memory or the other way rounf with IOR and MEMW being active.

7.2 Timing Waveforms of the DMA Processes

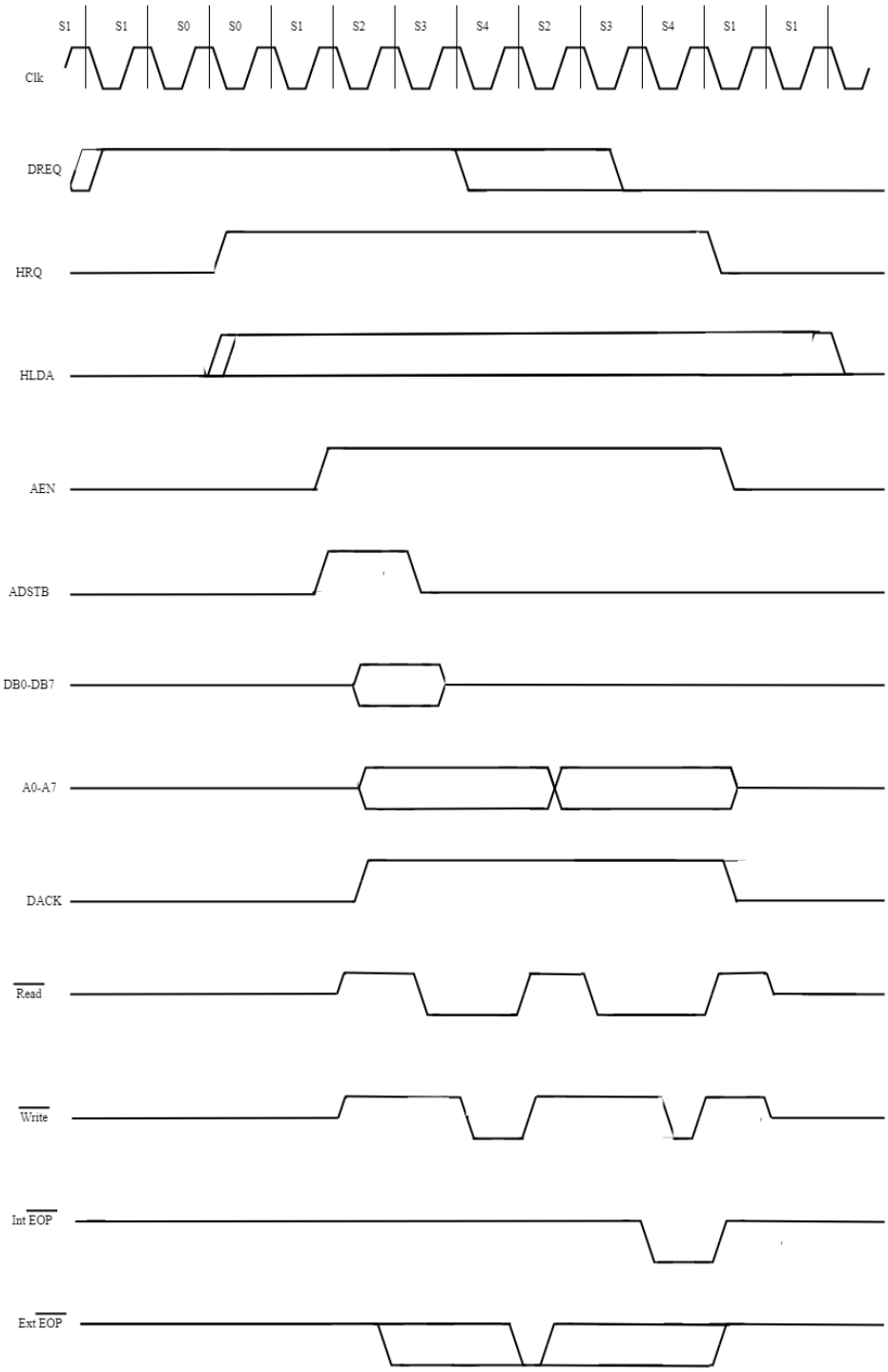

Figure 7.2.1: Timing Waveforms of the DMA Processes

Chapter 8: Conclusion and Future Work

8.1 Conclusion

The first objective and some parts of the objective two has been achieved. The exception handler is not able to come up with due to the lack of information found in books and papers. On the other hand, the previously developed DMA controller has been revised and enhanced further. With few added functions such as different modes of transfer instead of the default memory-to-memory transfer. The Microarchitecture of the DMA has also been developed and can be found in chapter 5. With the information, a complete RTL design can be completed with ease.

Although the implementation to the processor is quite vague, the basic functionality and its timing diagram is shown in chapter 7. The physical design can't be implemented as well due to the unavailability of the facilities.

8.2 Future Work

In the future, a further implementation of the DMA can be tested on board together with any physical designs planned earlier for the project. A further detail can also be included into the project with more testings such as the electronic design testing which will specifically show how much the power consumption of the device is and device can be simulated in real time instead of doing it in softwares like modelsim and vivado design suite.

Bibliography

[1] N. Singh, S. V. Sai Santosh and S. J. Darak, "Toward Intelligent Reconfigurable Wireless Physical Layer (PHY)," IEEE, 2021.

[2] D. Priya and D. Deepa, "Solar tracking system with high precision implementation on FPGA with GUI on LabVIEW," IEEE, 2014.

[3] S. Roy, H. Nagle and M. McNamer, "QRS/BIST: a reliable heart rate monitor ASIC," IEEE, 1990.

[4] R. Dorsch, R. Rivera, H. Wunderlich and M. Fischer, "Adapting an SoC to ATE concurrent test capabilities," in *International Test Conference*, Baltimore, 2002.

[5] M. Caldari, M. Conti, M. Coppola, S. Curaba, L. Pieralisi and C. Turchetti, "Transaction-level models for AMBA bus architecture using SystemC 2.0," in *2003 Design, Automation and Test in Europe Conference and Exhibition*, Munich, 2003.

[6] Xiao Li, L. Ji, B. Shen and W. Li, "VLSI implementation of a high-performance 32-bit RISC microprocessor," in *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference*, 2002.

Appendix A:

A.1 RTL design module of interfaces.

```
interface dma_if(input logic CLK, input logic RESET);

	/* interface to RISC32 processor */
	logic	MEMR_N;		// memory read
	logic	MEMW_N;		// memory write
	wire	IOR_N;		// IO read
	wire	IOW_N;		// IO write
	logic	HLDA;		// Hold acknowledge from CPU to indicate it has relinquished bus control
	logic	HRQ;		// Hold request from DMA to CPU for bus control

	/* address and data bus interface */
	logic [3:0] ADDR_U;	// upper address which connects to address A7-A4 of 8086 CPU
	wire  [3:0] ADDR_L;	// lower address which connects to address A3-A0 of 8086 CPU
	wire  [7:0] DB;		// data
	logic	CS_N;		// Chip select
	logic	AEN;		// address enable

	/* Request and Acknowledge interface */
	logic  [3:0] DREQ;	// asynchronous DMA channel request lines
	logic  [3:0] DACK;	// DMA acknowledge lines to indicate access granted to peripheral who has raised a request

	/* interface signal to 8-bit Latch */
	logic	ADSTB;		// Address strobe

	/* EOP signal */
```

```
wire        EOP_N;              // bi-directional signal to end DMA active transfers

// modport for design top
modport DUT(
                input  CLK,
                input  RESET,
                inout  IOR_N,
                inout  IOW_N,
                inout  DB,
                inout  ADDR_L,

                inout  EOP_N,

                input  DREQ,
                input  HLDA,
                input  CS_N,

                output ADDR_U,
                output DACK,
                output HRQ,
                output AEN,
                output ADSTB
        );

// modport for Datapath
modport DP(
                input  CLK,
                input  RESET,
                input  IOR_N,
                input  IOW_N,
                input  HLDA,
```

```
                    input  CS_N,
                    inout  DB,
                    inout  ADDR_L,
                    output ADDR_U
);

// modport for Priority logic
modport PR(
                    input  CLK,
                    input  RESET,
                    output DACK,
                    output HRQ,
                    input  DREQ,
                    input  HLDA
);

// modport for Timing Control logic
modport TC(
                    input   CLK,
                    input   RESET,
                    input   HLDA,
                    output  IOR_N,
                    output  IOW_N,
                    output  MEMR_N,
                    output  MEMW_N,
                    input   CS_N,
                    inout   EOP_N,
                    output  AEN,
                    output  ADSTB
);
```

```
        /* Modport for Test Bench */
        modport TB(clocking cb);

        /* Clocking Block to drive stimulus at cycle level */
        clocking cb @(posedge CLK);

                default input #0 output #0;

                inout   IOR_N;
                inout   IOW_N;
                inout   DB;
                inout   ADDR_L;
                inout   EOP_N;
                output  DREQ;
                output  HLDA;
                output  CS_N;
                input   ADDR_U;
                input   MEMR_N;
                input   MEMW_N;
                input   DACK;
                input   HRQ;
                input   AEN;
                input   ADSTB;

        endclocking

endinterface
```

A.2 DMA Timing and Control module interface

```
interface DmaControlIf(input logic CLK, RESET);

logic hrq;
```

```
logic ldCurrAddrTemp;
logic ldCurrWordTemp;
logic enCurrAddr;
logic ldTempCurrAddr;
logic ldTempCurrWord;
logic Program;
logic validDACK;



logic VALID_DREQ0;
logic VALID_DREQ1;
logic VALID_DREQ2;
logic VALID_DREQ3;

modport DP(
        input CLK,
        input RESET,
        input VALID_DREQ0,
        input VALID_DREQ1,
        input VALID_DREQ2,
        input VALID_DREQ3,
        input ldCurrAddrTemp,
        input ldCurrWordTemp,
        input enCurrAddr,
        input ldTempCurrAddr,
        input ldTempCurrWord,
        input Program
);

modport TC(
        input  CLK,
```

```
            input  RESET,
            input  VALID_DREQ0,
            input  VALID_DREQ1,
            input  VALID_DREQ2,
            input  VALID_DREQ3,
            output hrq,
            output ldCurrAddrTemp,
            output ldCurrWordTemp,
            output enCurrAddr,
            output ldTempCurrAddr,
            output ldTempCurrWord,
            output Program,
            output validDACK
);
modport PR(
            input  CLK,
            input  RESET,
            input  hrq,
            input  validDACK,
            output VALID_DREQ0,
            output VALID_DREQ1,
            output VALID_DREQ2,
            output VALID_DREQ3
);

endinterface
```

A.3 Interface for DMA Registers

```
// Interface for DMA Registers

interface DmaRegIf(input logic CLK, RESET);
```

```
// DMA Registers
 logic [5:0]  modeReg[4];
 logic [7:0]  commandReg;
 logic [7:0]  requestReg;
 logic [7:0]  maskReg;
 logic [7:0]  statusReg;



modport DP(
 input  CLK,
 input  RESET,
 output modeReg,
 output commandReg,
 output requestReg,
 output maskReg,
 output statusReg

);

modport TC(
 input  CLK,
 input  RESET,
 input modeReg,
 input commandReg,
 input statusReg
);

modport PR(
 input  CLK,
 input  RESET,
 input commandReg,
```

```
 input requestReg,
 input maskReg
);

endinterface
```

APPENDIX B: Bi-weekly Reports

B.1: Bi-weekly Week 2

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 2**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

Finished studying DMA interface outsourcing., started design RTL model for DMA Interface using modelsim SE.

2. WORK TO BE DONE

Continue RTL Modelling

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

No Problem so far

Supervisor's signature

Student's signature

B.1: Bi-weekly Week 4

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 4**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

RTL Modelling for DMA interface

2. WORK TO BE DONE

Datapath modelling using modelsim

3. PROBLEMS ENCOUNTERED

Was put on hold so progress is abit slow due to family reasons

4. SELF EVALUATION OF THE PROGRESS

Abit too slow for my comfort

Supervisor's signature

Student's signature

B.1: Bi-weekly Week 6

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 6**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

Some parts of Datapath design

2. WORK TO BE DONE

Datapath modelling using modelsim

3. PROBLEMS ENCOUNTERED

Was put on hold so progress is abit slow due to family reasons

4. SELF EVALUATION OF THE PROGRESS

Going slightly faster since problem has been solved

Supervisor's signature

Student's signature

B.1: Bi-weekly Week 8

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 8**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

RTL Modelling for Datapath Unit, Blok interface, pin description and functionality, some register commands diagram and FSM

2. WORK TO BE DONE

Control and timing unit for DMAC modelling using modelsim

3. PROBLEMS ENCOUNTERED

Might take slightly longer time since it's quite a big part

4. SELF EVALUATION OF THE PROGRESS

Can be achieved, need to put in effort

Supervisor's signature

Student's signature

d

B.1: Bi-weekly Week 10

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 10**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

RTL Modelling for Timing and Control unit pin description and functionality, some register commands diagram and FSM

2. WORK TO BE DONE

Priority unit modelling using modelsim

3. PROBLEMS ENCOUNTERED

So far so good

4. SELF EVALUATION OF THE PROGRESS

Going good because of some external references

Supervisor's signature

Student's signature

e

B.1: Bi-weekly Week 12

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 12**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

RTL Modelling for Priority unit pin description and functionality, some register commands diagram and FSM

2. WORK TO BE DONE

Started integrating the units together so that it is ready for simulation

3. PROBLEMS ENCOUNTERED

Easier individually but does not seem to work when put together

4. SELF EVALUATION OF THE PROGRESS

Getting help from online sources.

Supervisor's signature

Student's signature

f

B.1: Bi-weekly Week 14

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: JAN 2022	**Study week no.: 14**
Student Name & ID: Tan E-Chian 1705490	
Supervisor: Ts. Dr. Chang Jing Jing	
Project Title: Design of a Direct Memory Access Module for 32-Bit RISC32 Processor	

1. WORK DONE

Integration on of DMAC on processor, but slightly vague since its not fully achievable. Documentation and editing formats of report.

2. WORK TO BE DONE

Waiting to hand in report

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

Supervisor's signature

Student's signature

g

Poster


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of information and communication technology
DMA
Direct memory access also known as the DMA is a must in every computer system because without it, data transfer wuold seem also impossible to achieve.
This project will go over the functionality of the DMA and how to build it and implement it in the RISC32 pipeline processor.
Introduction
• Overview of the DMA
• Project Background
Literature Review
• Previous Research Done
• Detailed functionality of more advanced DMA
Methodologies
• Verilog HDL
• QTSpim
System Design
• Overall Block diagram
• Functionality & description of I/O pins
Conclusion
• Future planning
• Gantt Chart progress report

Plagiarism Check Result

Match Overview

9%

1	Lubna Aziz, Md. Sah bi... Publication	5%
2	Peng Zhang. "Program... Publication	1%
3	"Fundamentals of Com... Publication	<1%
4	S PASRICHA. "On-Chip ... Publication	<1%
5	Shuangbao Paul Wang.... Publication	<1%
6	J PARK. "The PC for re... Publication	<1%
7	Neelam Singh, S. V. Sai ... Publication	<1%
8	"Advanced Parallel Pro... Publication	<1%
9	Abdullah Aljumah, Moh... Publication	<1%
10	D. Priya, P. Deepa. "Sola... Publication	<1%
11	Manami Ogawa, Susu... Publication	<1%
12	Theo Fowinkel. "Huma... Publication	<1%
13	Mohamed Ben-Romdh... Publication	<1%
14	Ruchita Kawle, Shubha... Publication	<1%
15	W.T. Krakow. "QRS/BIS... Publication	<1%
16	R. Dorsch, R.H. Rivera, ... Publication	<1%
17	Vaibbhav Taraate. "Cha... Publication	<1%
18	Siew Keng Chan, Yong ... Publication	<1%
19	Yao, Ying Biao, Xian Bin... Publication	<1%
20	Ken Yang, Chih-Kong. "I... Publication	<1%

21	M. Caldari, M. Conti, M.... Publication	<1%	>
22	Xu Ke, Zhu Kejia, Li Qia... Publication	<1%	>
23	Zainalabedin Navabi. "... Publication	<1%	>
24	"Computer Hardware f... Publication	<1%	>
25	Lecture Notes in Electri... Publication	<1%	>
26	M.B. Blake. "An agent-b... Publication	<1%	>
27	Reuben James L. Austr... Publication	<1%	>
28	K JAMES. "Data transfe... Publication	<1%	>

5% match (publications)
Lubna Aziz, Md. Sah bin Haji Salam FC, Usman Ullah Sheikh, Surat Khan, Huma Ayub, Sara Ayub. "Multi-level Refinement Feature Pyramid Network for scale imbalance object detection", IEEE Access, 2021

1% match (publications)
Peng Zhang. "Programmable-logic and application-specific integrated circuits (PLASIC)", Advanced Industrial Control Technology, 2010

<1% match (publications)
"Fundamentals of Computer Organization and Design", Springer Science and Business Media LLC, 2003

<1% match (publications)
S PASRICHA. "On-Chip Communication Architecture Standards", On-Chip Communication Architectures, 2008

<1% match (publications)
Shuangbao Paul Wang. "Computer Architecture and Organization", Springer Science and Business Media LLC, 2021

<1% match (publications)
J PARK. "The PC for real time work", Practical Data Acquisition for Instrumentation and Control Systems, 2003

<1% match (publications)
Neelam Singh, S. V. Sai Santosh, Sumit J. Darak. "Toward Intelligent Reconfigurable Wireless Physical Layer (PHY)", IEEE Open Journal of Circuits and Systems, 2021

<1% match (publications)
"Advanced Parallel Processing Technologies", Springer Science and Business Media LLC, 2015

<1% match (publications)
Abdullah Aljumah, Mohammed Altaf. "Amba Based Advanced DMA Controller for SoC", International Journal of Advanced Computer Science and Applications, 2016

<1% match (publications)
D. Priya, P. Deepa. "Solar tracking system with high precision implementation on FPGA with GUI on LabVIEW", 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), 2013

<1% match (publications)
Manami Ogawa, Susumu Yamamoto, Yuka Kousa, Fumitaka Nakamura et al. "Development of soft x-ray time-resolved photoemission spectroscopy system with a two-dimensional angle-resolved time-of-flight analyzer at SPring-8 BL07LSU", Review of Scientific Instruments, 2012

<1% match (publications)
Theo Fowinkel. "Human Resource Management Systems in New Business Creation", Springer Science and Business Media LLC, 2014

<1% match (publications)
Mohamed Ben-Romdhane. "Extending the transaction level modeling approach for fast communication architecture exploration", Proceedings of the 41st annual conference on Design automation - DAC 04 DAC 04, 2004

<1% match (publications)
Ruchita Kawle, Shubhada Thakare. "Designing, Analysis and Synthesis of 32-Bit Configurable Hack CPU", 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2021

<1% match (publications)
W.T. Krakow. "QRS/BIST: a reliable heart rate monitor ASIC", Third Annual IEEE Proceedings on ASIC Seminar and Exhibit, 1990

<1% match (publications)
R. Dorsch, R.H. Rivera, H.J. Wunderlich, M. Fischer. "Adapting an SoC to ATE concurrent test capabilities", Proceedings. International Test Conference, 2002

<1% match (publications)
Vaibbhav Taraate. "Chapter 5 Processor Cores and Architecture Design", Springer Science and Business Media LLC, 2019

<1% match (publications)
Siew Keng Chan, Yong Haur Tay, Christian Viard-Gaudin. "Online text independent writer identification using character prototypes distribution", 2007 6th International Conference on Information, Communications & Signal Processing, 2007

<1% match (publications)
Yao, Ying Biao, Xian Bin Zeng, and Guang Pei Zhao. "Design and Implementation of MIPS Simulator Oriented Memory Hierarchy Research", Applied Mechanics and Materials, 2012.

<1% match (publications)
Ken Yang, Chih-Kong. "Input/Output Devices", Electrical Engineering Handbook, 2004.

<1% match (publications)
M. Caldari, M. Conti, M. Coppola, S. Curaba, L. Pieralisi, C. Turchetti. "Transaction-level models for AMBA bus architecture using SystemC 2.0", 2003 Design, Automation and Test in Europe Conference and Exhibition, 2003

<1% match (publications)
Xu Ke, Zhu Kejia, Li Qiang, Min Hao. "The architecture comparison and the VLSI implementation of the 32 bit embedded RISC", 2003 5th International Conference on ASIC Proceedings (IEEE Cat No 03TH8690) ICASIC-03, 2003

<1% match (publications)
Zainalabedin Navabi. "Digital System Test and Testable Design", Springer Science and Business Media LLC, 2011

<1% match (publications)
"Computer Hardware for Industrial Control", Industrial Control Technology, 2008

<1% match (publications)
Lecture Notes in Electrical Engineering, 2009.

<1% match (publications)
M.B. Blake. "An agent-based cross-organizational workflow architecture in support of Web services", Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002

<1% match (publications)
Reuben James L. Austria, Annaliza L. Sambile, Kahr-Lile M. Villegas, Jay Nickson T. Tabing. "Design of an 8-bit five stage pipelined RISC microprocessor for sensor platform application", TENCON 2017 - 2017 IEEE Region 10 Conference, 2017

<1% match (publications)
K JAMES. "Data transfer", PC Interfacing and Data Acquisition, 2000

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Programme / Course	BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMPUTER ENGINEERING
Title of Final Year Project	Design of a Direct Memory Access Module for 32-Bit RISC32 Processor

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 9 **%** **Similarity by source** Internet Sources: NA % Publications: 9 % Student Papers: NA %	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and
(ii) Matching of individual sources listed must be less than 3% each, and
(iii) Matching texts in continuous block must not exceed 8 words

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Chang

Signature of Supervisor

Name: Ts Dr. Chang Jing Jing

Date: 22 April 2022

Signature of Co-Supervisor

Name: _____

Date: _____

M

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	1705490
Student Name	Tan E-Chian
Supervisor Name	Ts Dr. Chang Jing Jing

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Plastic Cover (for hardcopy)
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
✓	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)
Date: 22/4/2022