Implementation of C-Class Microcontroller SoC

A Project Report

submitted by

Venkatesh Bajjuri

in partial fulfilment of the requirements

for the award of the degree of

Master of Technology

Under the guidance of

Dr. V. Kamakoti

Department of Electrical Engineering

Indian Institute of Technology Madras

May 2015

Thesis Certificate

This is to certify that the thesis titled **Implementation of C-Class Micro-controller SoC**, submitted by **Venkatesh Bajjuri**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V. Kamakoti
Project Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place : Chennai
Date :

Acknowledgement

I would like to express my sincere gratitude to my guide, ***Dr. V. Kamakoti*** for his valuable guidance, encouragement and advice. His immense motivation helped me in making firm commitment towards my project work.

My special thanks to ***Mr. G.S. Madhusudan*** for his encouragement and motivation through out the project. His valuable suggestions and constructive feedback were very helpful in moving ahead with my project work.

I would like to thank my co-guide ***Dr.Nitin Chandrachoodan*** and faculty advisor ***Dr. Nagendra Krishnapura*** who have patiently listened, evaluated, and guided us through out the program.

My special thanks to my project team members Neel Gala,Arjun Menon,Rahul,Arnab,Anand Sai Varma,Sukrat,Laxmeesha,Suresh Sandeep,Sunnithith,Rutvik for their help and support.

Abstract

SoC forms the basis for the present day embedded systems.Silicon densities, both for ASICs and FPGAs, can now support true systems-on-a-chip (SoCs). This level of design requires busing systems to connect various components, including one or more microprocessors, memory, peripherals, and special logic.The challenges we face consist of frequency,power profile and silicon area.As the number of peripherals and cores increase,the logic gets more complex and hardware increases.

The AHB supports 32-,64-,128-bit data bus implementations and also support bursting and pipelining.In a typical configuration, the SoC processor(s), memory controllers, on-chip memory, and DMA controllers hang off of the AHB. It handles the high-speed bus interconnections on the chip. The slower peripherals are hung off of the slower, simpler APB. These also include special logic functions and connections to the SoC basic logic.The SoC's system and peripheral buses can run at different clock rates. They link via a bridge that buffers data and operations between the two buses.The master-generated address is decoded by a central address decoder that provides a select signal to the addressed bus slave unit. The bus master can "lock" the bus, reserving it with the central arbiter for a series of locked transfers.APB on the other hand doesn't support bursting.It supports low-speed peripherals such as UARTs, keypads, and Primary I/O.

The entire coding is done in Bluespec System Verilog.

Keywords: Advanced Micro-controller Bus Architecture[AMBA], AMBA High-performance Bus[AHB], AMBA Peripheral Bus[APB], AHB-APB Bridge, System on Chip[SoC]

Contents

List of Figures

List of Tables

Abbreviations

AMBA	Advanced Microcontroller Bus Architecture
AHB	AMBA High-performance Bus
APB	AMBA Peripheral Bus
ISA	Instruction Set Architecture
FSM	Finite State Machine
CPU	Central Processing Unit
HDL	Hardware Description Language
BSV	Bluespec System Verilog
BDW	Bluespec Development Workstation
RISC	Reduced Instruction Set Computer
FIFO	First In First Out

Chapter 1

Introduction

1.1 Overall Microcontroller Architecture

The processor design team of Reconfigurable and Intelligent Systems Engineering[RISE] lab in the computer science department of IIT-Madras has been actively involved in building few processors for academic purposes and other applications. The processor strictly follows the RISC-V Instruction Set Architecture[ISA].Entire design of the processor is done using a Hardware Description Language[HDL] named Bluespec System Verilog[BSV].

The C-Class processor is a 32-bit in-order variant aimed at 50-250 MHz microcontroller variants which have an optional memory protection and the design consumes very low power. The integration forms the basis for synchronizing the core to different peripherals in the microcontroller, which have various operating frequencies. My project work involves the integration of different peripherals onto the C-Class core using the AHB, APB, Bridge protocols. Various dummy slave memories have been implemented to check the read and write operations of the core. The core fetches an instruction from a slave, decodes it, fetches the operands from slaves and stores the result back into one of the dummy slave memory.

H prefix indicates that the signal belongs to the AHB bus and P prefix indicates that the signal belongs to APB bus.

1.2 Objective

- Integrate various peripherals onto the C-Class core to form a microcontroller
- Implement the integration using bus protocols
- Synchronize the fast and slow bus to avoid data loss

1.3 Organization of Thesis

Chapter 2 describes about the HDL named BSV, its key features and BSV constructs. It also describes a little about the in-order pipelined micro processor.

Chapter 3 describes the AHB protocol, the logic involved in it and the timing diagrams such as the read/write operation under different conditions and the validity of the operation. It explains about various signals which can be further exploited to maintain the integrity of a transfer.

Chapter 4 describes the APB protocol, the finite state machine involved in it and the timing diagrams such that read/write operations with and without error. It explains about the various stages through which data needs to flow from master while communicating with various slaves and how a single master identifies each slave. It also explains about the synchronization between the AHB and APB buses through the bridge and also the corresponding FSM.

Chapter 5 describes about the APB Bridge in brief detail and how the signals of the AHB-Lite are mapped on to the APB.

Chapter 6 describes the integration of the whole package between the core and the peripheral and how the read and write operations are performed. The verification of the whole system with various test cases.

Chapter 7 concludes with a short description about future work.

Chapter 2

Background

This chapter deals with the basics of In-order pipelined processor.It also deals with the key features of BSV,why Bluespec is used and how to build a design in BSV.It describes about the different stages in the regular 5-stage in-order execution variant pipelined processor.

2.1 Basics of C-Class Processor

This has been done externally.My project work considers the core as a black box and has access only to the input and output signals coming from this black box.C-Class processor is a in-order variant with a 3-8 stage pipeline with a functional frequency aimed at the 50-250 MHz range microcontrollers. This processor strictly follows the RISC-V ISA.It has been designed for 45 instructions.It receives addresses from the cache controller,which is also designed externally.The regular 5 stage pipeline basics are presented here:

2.1.1 Five stages of regular in-order pipeline

- Instruction Fetch
- Instruction Decode and Operand Fetch
- Execution
- Memory Access

- Write Back

The 5-stage in-order variant processor is a 32 bit processor.This processor is designed only for 28 instructions.It fetches one instruction,at a time,from the instruction memory.The fetched instruction is sent to the instruction decode and operand fetch stage for decoding.The instruction is split into chunks based on the last 7 digits as mentioned in the RISC-V ISA. The operands are then fetched from the respective addresses in the decoded instruction. The operands are executed in the execution stage and then final result is sent to memory execution stage.Depending on the instruction type,the final result is committed back into the memory unit or sent to the write back stage to be written back into the register file.

Each stage is separated by a FIFO. When the FIFO is full,the rule enqueuing data into it won't fire,which is an implicit condition of the rule.This helps in stalling the pipeline when the further stages of a given FIFO are busy.In this processor design,no branch instructions were used.Hence no branch prediction unit is required.

The following Fig 2.1 shows a C-Class core.

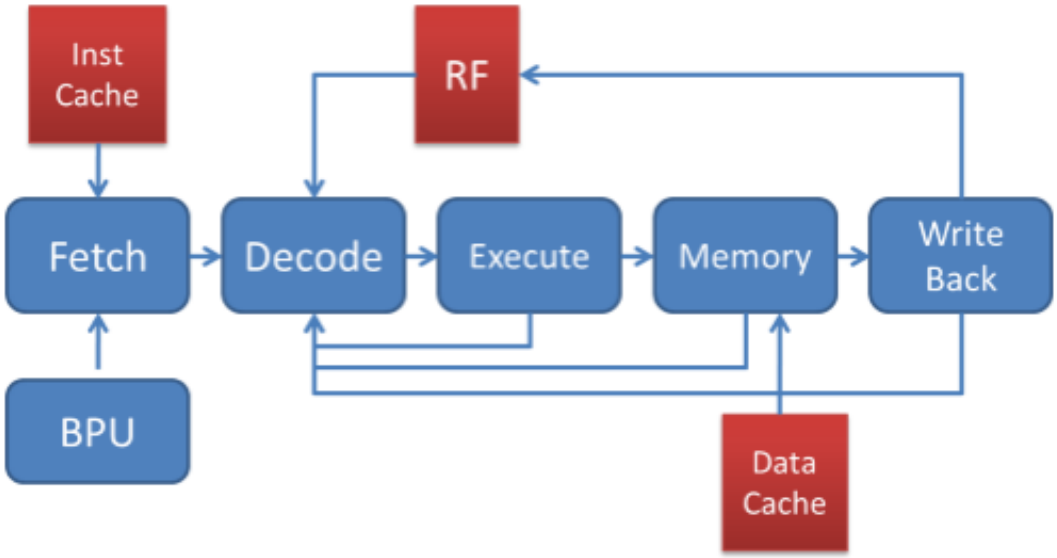

Figure 2.1: C-Class Architecture

Chapter 3

AMBA High-Performance Bus

This chapter deals with the definitions of the AMBA and AHB-Lite protocol.It also describes about the various signals present in this protocol,how they're connected to the bus,the type of transfers and the responses for various transfers from the slave.

3.1 Introduction to AHB-Lite and Specifications

AMBA is the ARM open standard for on-chip buses.It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a System- on-Chip(SoC).It aids in the development of embedded processors with one or more processors and multiple peripherals.

3.1.1 AHB-Lite

It consists of various types of transfers and slave responses which occur in accordance with the transfers.It is a bus interface which supports single bus master and supports data bus configurations from 32-,64- to 1024 bits.The common slaves for AHB-Lite are internal memory,external memory interfaces,APB bridges.

The Fig 3.1 shows the AHB-Lite system with a single master and three slaves.

The other logic consists of an address decoder and a slave to master multiplexor.The decoder monitors the address from the master so that the corresponding slave is selected and multiplexor selects the corresponding slave's output data and sends it to the master.

The Fig 3.2 shows the AHB-Lite Master Interface.

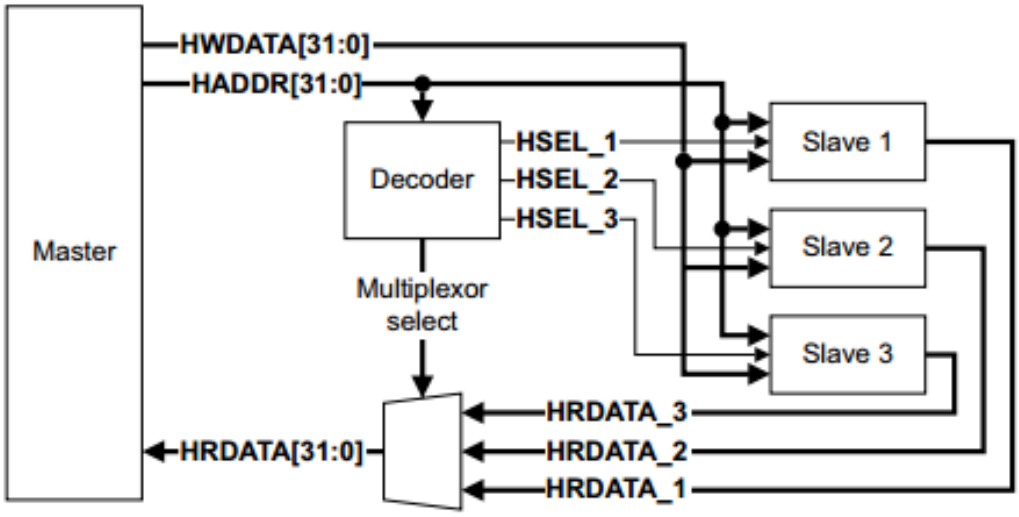

Figure 3.1: AHB-Lite Block Diagram

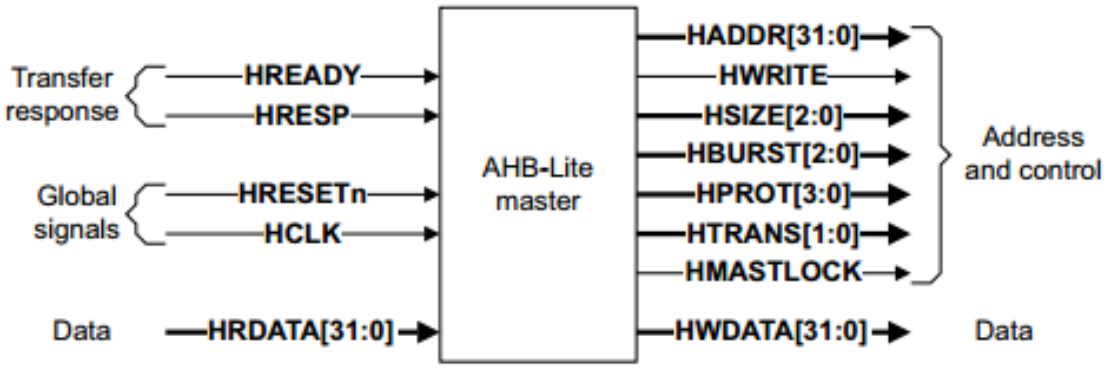

Figure 3.2: Master Interface

The master initiates the read or write transfer by providing the address and control information.These indicate the slave to be selected,the direction of the transfer,width of the transfer etc.

The slave has 3 responses for the transfer viz., Success,Failure,Waiting.The slave identifies its selection from the HSEL signal from the decoder.

The Fig 3.3 shows the AHB-Lite Slave Interface.

The decoder selects the slave from the address provided by the master.It also sends the control signal to the multiplexor while transferring data from the slave to

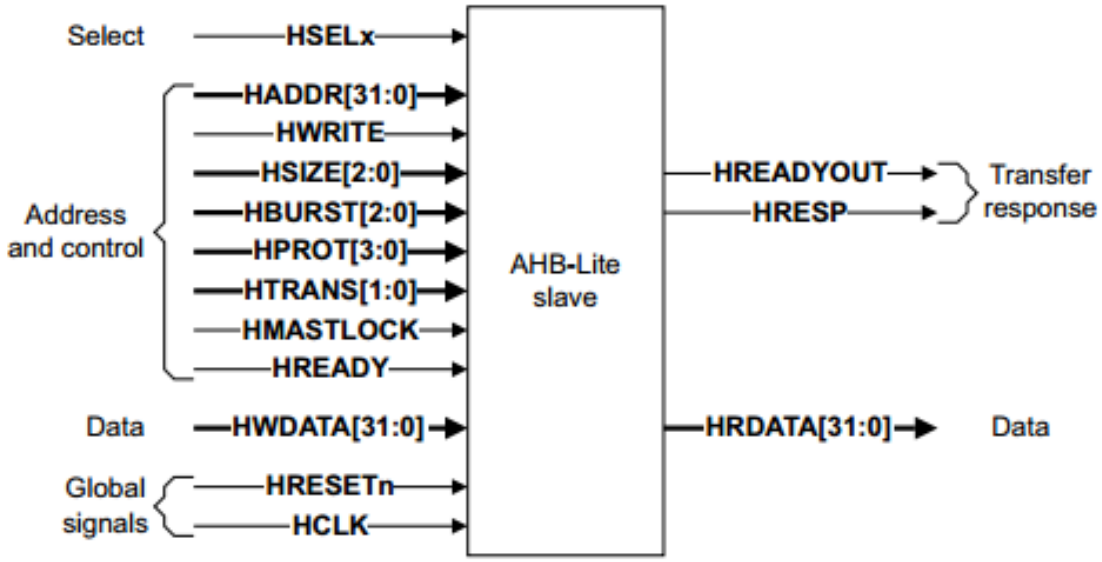

Figure 3.3: Slave Interface

the master.Multiplexor multiplexes the read data and the responses of each transfer from slave to the master.All these form part of AHB-Lite bus.

Every transfer has an address phase and a data phase.The data phase follows the address phase from the next cycle.Depending on whether there is a wait-state or not the data phase is extended.The address phase is never extended.There are 3 types of transfers viz., Single,Incremental Burst,Wrap Around Burst.

3.2 Transfers

3.2.1 Basic Transfer

It consists of two phases:

Address Phase: Lasts for a single cycle unless its extended by previous transfer.

Data Phase: Lasts for one or more cycles depending on the HREADY signal. Refer to Table 3.1 for the list of signals in AHB. HWRITE controls the direction of data transfer.When LOW,its a read transfer. The Fig 3.4 shows a simple read transfer.

A and B are two slaves.When HIGH,its a write transfer.The Fig 3.5 shows a simple

List Of Signals

Name	Description
HCLK	This is the system clock signal.All the signals are sampled at the rising edge of the clock
HRESETn	This is the reset signal.It is an active low signal.It is asynchronous with respect to the system clock
HADDR	Address bus.Can be parametrized
HBURST	This indicates the type of burst such as incremental,wrap around,single length,undefined length etc
HMASTLOCK	This indicates that the transfer between the master and the slave is locked.
HPROT	This signal provides additional information about a bus access and used to implement some level of protection
HSIZE	This signal indicates the size of transfer such as byte,halfword,word
HTRANS	This signal indicates the current type of transfer such as IDLE,BUSY, NONSEQUENTIAL, SEQUENTIAL
HWDATA	Write data bus.Can be parameterized
HWRITE	This signal indicates the transfer direction. When LOW this signal indicates a read transfer.When HIGH this signal indicates a write transfer
HRDATA	Read data bus.Can be parameterized
HREADY	This signal indicates if the slave is ready for the transfer or not. HIGH indicates that the slave is ready. LOW indicates that the slave is busy and wait states are inserted in the transfer
HRESP	Response regarding the transfer, from the slave. LOW indicates that the transfer is a success. HIGH indicates a failure
HSELx	Slave select signal from the decoder based on the address sent by the master. Slave uses this signal to identify if it is selected.

Table 3.1: List of signals

write transfer. In case of wait states inserted due to HREADY signal.The Fig 3.6 shows a write transfer with a wait state. The Fig 3.7 shows a read transfer with two

wait states.

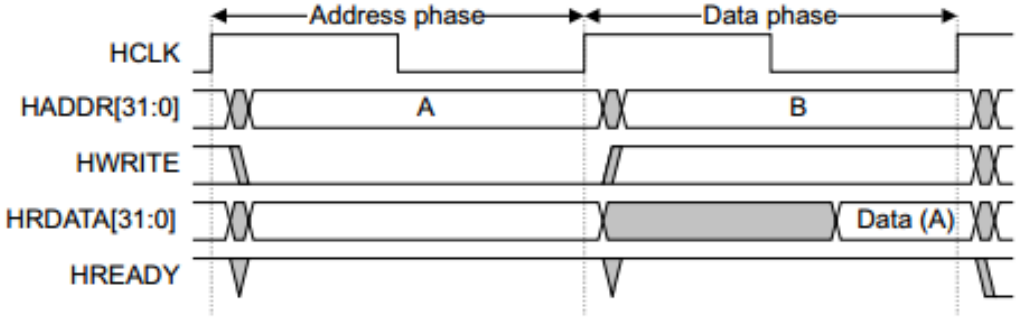

Figure 3.4: Read Transfer

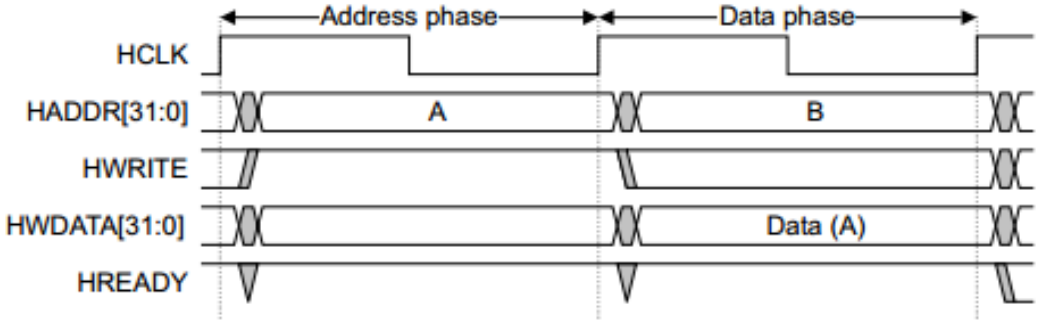

Figure 3.5: Write Transfer

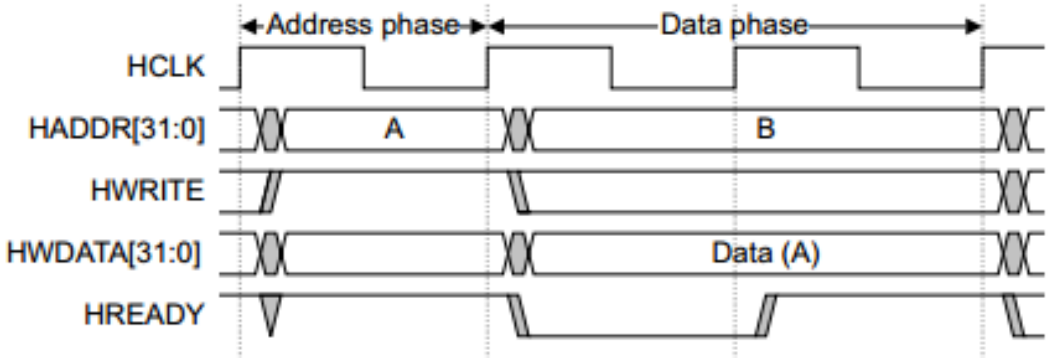

Figure 3.6: Write Transfer with wait states

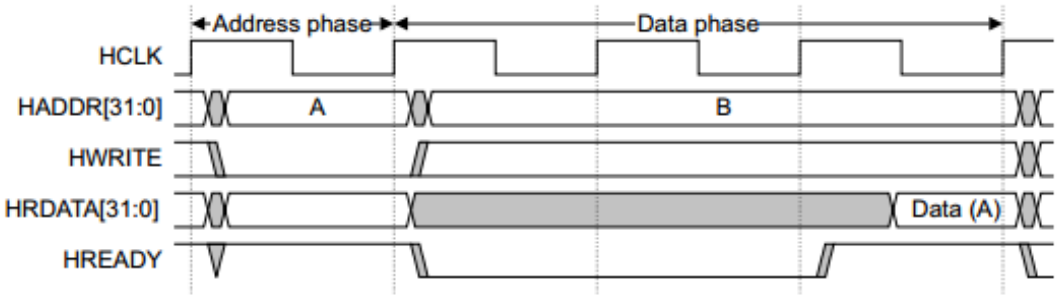

Figure 3.7: Read Transfer with wait states

3.2.2 Transfer Types

There are four transfer types:

IDLE: Indicates that the master doesn't want to perform any data transfer.Any data transferred during this state must be ignored by the slave.

BUSY: Indicates that the master is busy and the next transfer can't be performed immediately after the present one.

NONSEQ: Indicates that this a single length burst transfer.The address and control signals are unrelated to the previous transfer.

SEQ: Indicates that the next address is related to the present address.The control signals remain identical to those of the previous transfer.

Its a 4 beat incremental read transfer.Since the master is unable to perform the second beat a BUSY cycle is inserted and the corresponding address and control signals reflect in the next clock edge.

The Fig 3.8 shows a SEQ,NONSEQ,BUSY transfer types.

In the Fig 3.8,a four-beat read transfer starts with NONSEQ transfer type.In the cycle T1-T2,since the master is BUSY,it can't perform the beat.In this cycle slave returns the data of the previous address.In T2-T3,master performs the second beat.However it ignores any value on the data bus because in the previous cycle there was no valid address.In cycle T4-T5,since the slave is unable to cater the needs of master,it inserts a wait state.

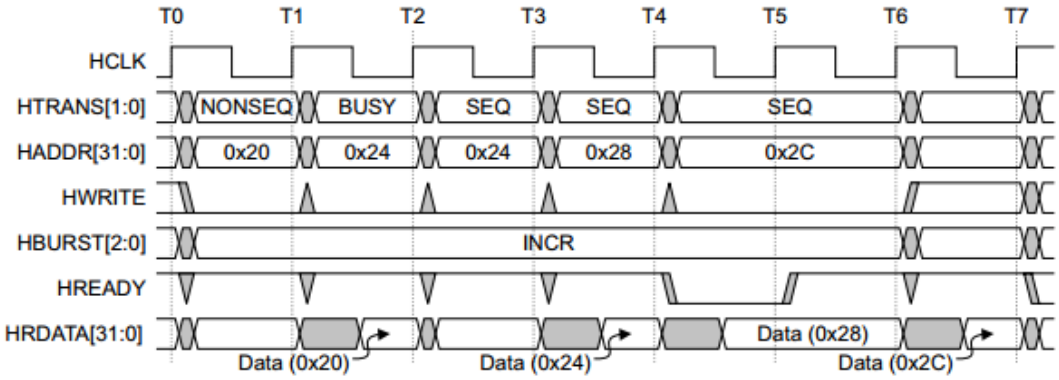

Figure 3.8: Transfer Type Examples

3.2.3 Locked Transfer

If the master asserts the HMASTLOCK signal it means that the present transfer sequence is indivisible to the slave.The slave can't perform any other transfers.This signal is most useful when a slave is controlled by multiple masters.

The Fig 3.9 shows the HMASTLOCK signal.

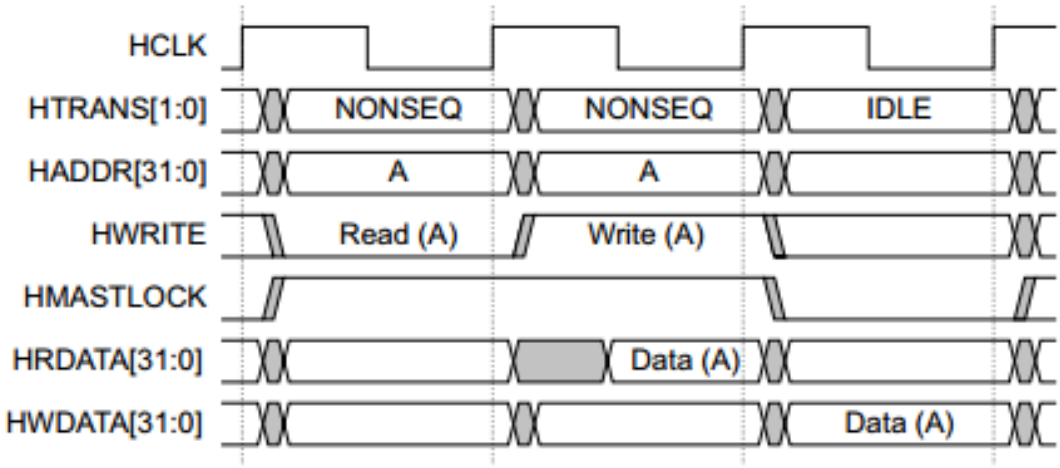

Figure 3.9: Locked Transfer

3.2.4 Transfer Size

Indicates the size of the data transfer.

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size(bits)	Description
0	0	0	8	Byte
0	0	1	16	Halfword
0	1	0	32	Word
0	1	1	64	Doubleword
1	0	0	128	4-word line
1	0	1	256	8-word line
1	1	0	512	-
1	1	1	1024	-

Table 3.2: Transfer size encoding

This,in conjunction with HBURST, is used in calculating the next address in case of a multi-beat burst transaction.Refer to Table 3.2 for various data sizes supported.

3.2.5 Burst Operation

Indicates the type of sequential transfer which needs to occur.This protocol supports 4,8,16-beats. Refer Table 3.3.

Each beat is an address with the same set of control signals.There are two types of burst mode:

- Incremental.
- Wrap Around.

In case of incremental burst mode the next address is an increment of the present address plus the transfer size.In case of a wrap around burst mode the addresses wrap across an address boundary which is calculated as the number of beats in

HBURST[2:0]	Type	Description
b000	SINGLE	Single burst
b001	INCR	Incremental burst of undefined length
b010	WRAP4	4-beat wrapping burst
b011	INCR4	4-beat incrementing burst
b100	WRAP8	8-beat wrapping burst
b101	INCR8	8-beat incrementing burst
b110	WRAP16	16-beat wrapping burst
b111	INCR16	16-beat incrementing burst

Table 3.3: Burst size encoding

the burst(HBURST) times the size of transfer(HSIZE). A Burst of size 1 is a non-sequential transfer.Undefined length burst is only incremental and not wrapping since the bus exactly doesn't know where to wrap. The burst mode automatically ends in a SEQ transfer unless it is a single length or undefined length burst.In both the cases the next transfer should be IDLE or NONSEQ.In case of undefined length burst only the master can terminate the burst.Also if the slave sends an error response then the master can terminate the burst,not a strict requirement though.

The Fig 3.10 shows a 4-beat incremental burst read transfer of WORD data size with a wait state inserted in between.

The Fig 3.11 shows a 4-beat wrap around burst write transfer of WORD data size with a wait state inserted in between.Since it is a 4-beat wrap around burst of WORD data size the address boundary is multiples of 16 byte.

The Fig 3.12 shows a 8-beat incremental burst write transfer of HALFWORD

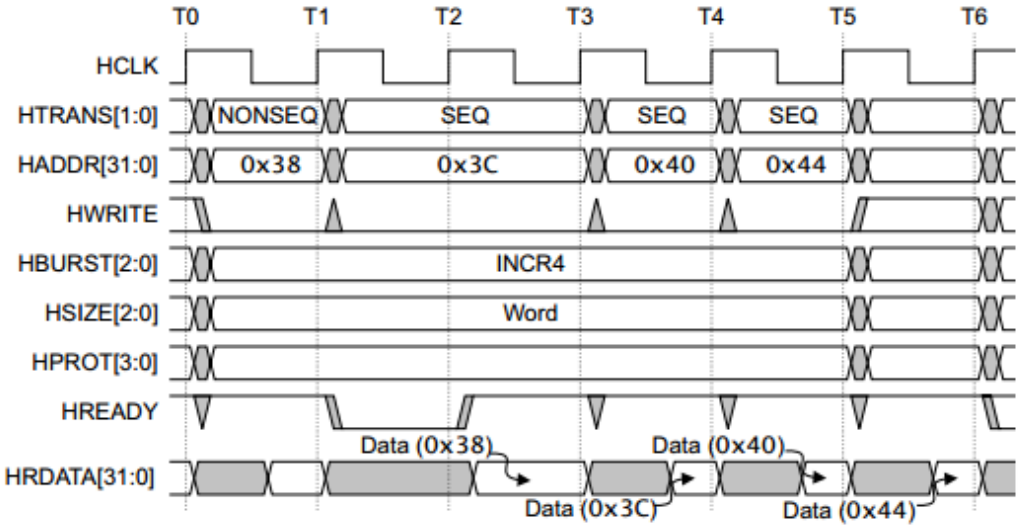

Figure 3.10: Four-beat incrementing burst

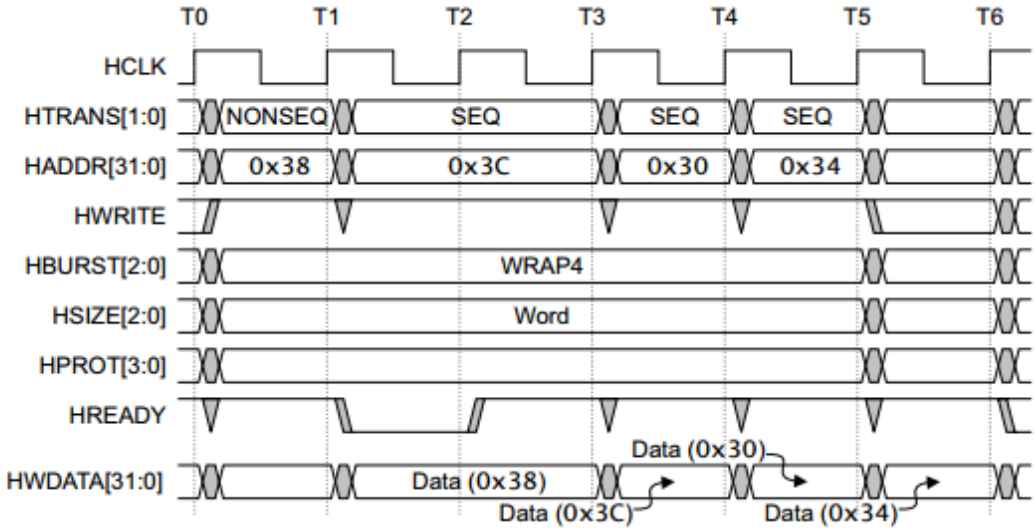

Figure 3.11: Four-beat wrapping burst

data size.

The Fig 3.13 shows a 8-beat wrap around burst read transfer of WORD data size.Since it is a 8-beat wrap around burst of WORD data size the address boundary is multiples of 32 byte. The Fig 3.14 shows a undefined length incremental burst transfer.In this burst the first two transfers are of HALFWORD size and the next

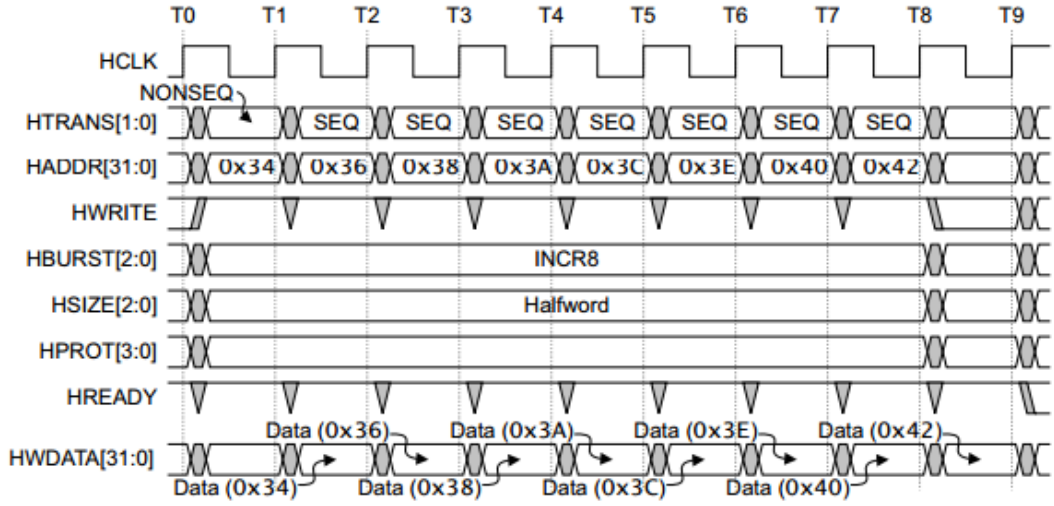

Figure 3.12: Eight-beat incrementing burst

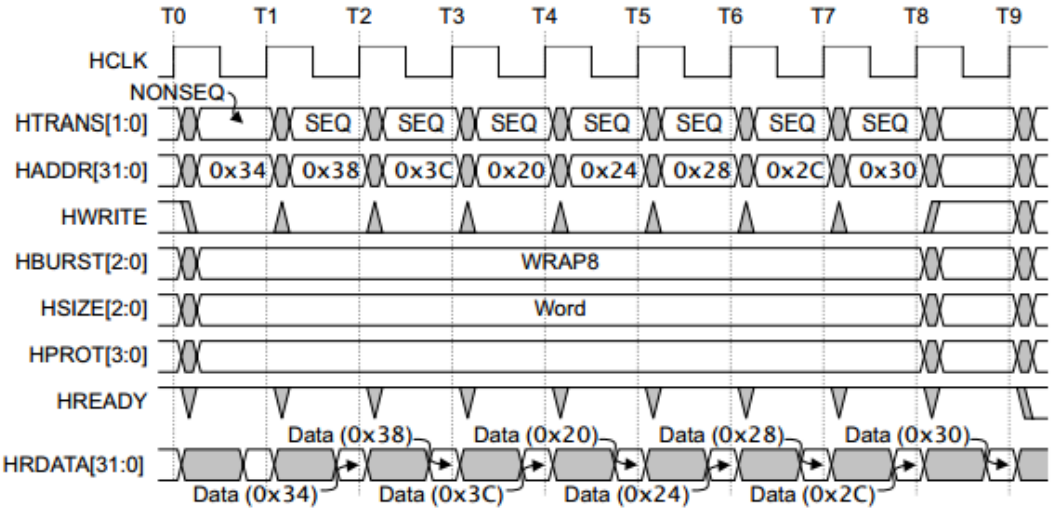

Figure 3.13: Eight-beat wrapping burst

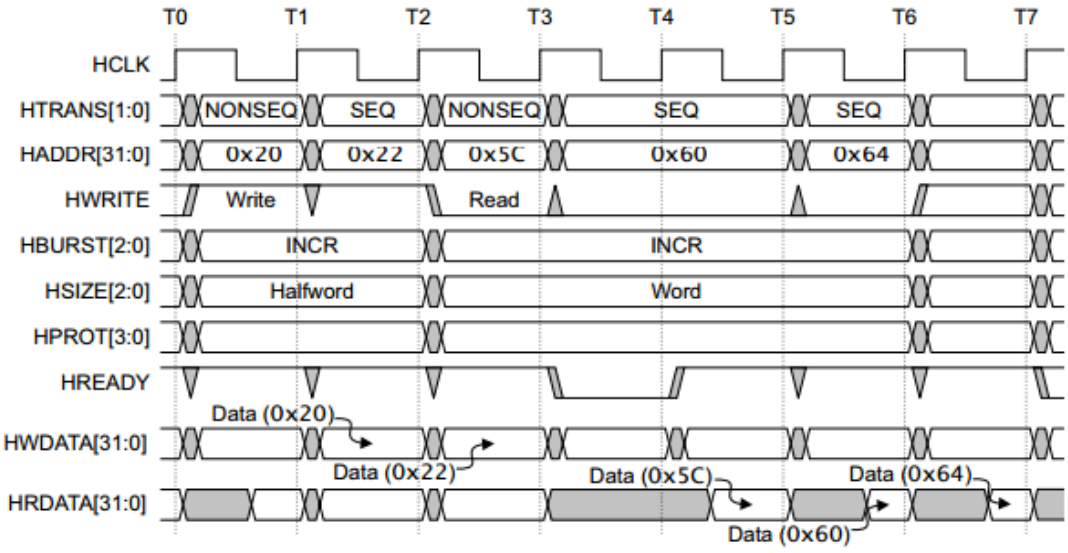

Figure 3.14: Undefined length burst

three transfers are of WORD size.

3.2.6 Protection Control

HPROT[3]	HPROT[2]	HPROT[1]	HPROT[0]	Description
-	-	-	0	Opcode fetch
-	-	-	1	Data access
-	-	0	-	User access
-	-	1	-	Privileged access
-	0	-	-	Non-Bufferable
-	1	-	-	Bufferable
0	-	-	-	Non-Cacheble
1	-	-	-	Cacheble

Table 3.4: Protection signal encoding

This signal indicates if the transfer is opcode fetch,data access,user mode access or privileged access.This is very useful for processors sharing same memory.Refer Table 3.4.

3.3 Bus Interconnection

Address decoding for slave select signals is done by the decoder.The minimum address boundary for each slave is 1KB.It can be increased.Hence for every 1KB change in address,the slave id changes.

The Fig 3.15 shows the slave select signals generated by the decoder.

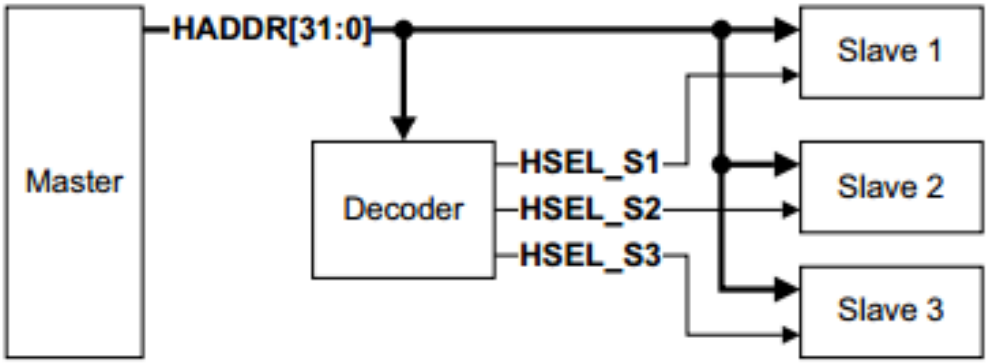

Figure 3.15: Slave select signals

A default slave is required in case non-existent address locations are accessed ie., outside the address locations of the present slaves combined.The response from this default slave is always an ERROR for any transfer.

3.4 Slave Response Signalling

Slave Transfer Response(HRESP) has 2 states.OKAY or ERROR.Once a master starts the transfer it loses control over the transfer.The slave controls the transfer.In case of OKAY the transfer is either successfully completed or requires additional cycles due to wait states inserted by the slave.In case of ERROR the transfer is a failure.ERROR response requires two cycles.Since the data comes a cycle after the address,the HRESP signal is delayed by one cycle for the address sent.So a new address

has already been put onto the bus.So the slave should lower the HREADY signal so that the new address transfer doesn't take place and drive the HRESP HIGH indicating ERROR.In the next cycle the HREADY can be driven high indicating that the slave is ready for another transfer.The HRESP is held HIGH indicating the ERROR.The processor doesn't insert a new address in case of the ERRORs' first cycle .Depending on its state,it can carry on with the address put on the bus or change to a new address in the next cycle. The Fig 3.16 shows a transfer with error response.

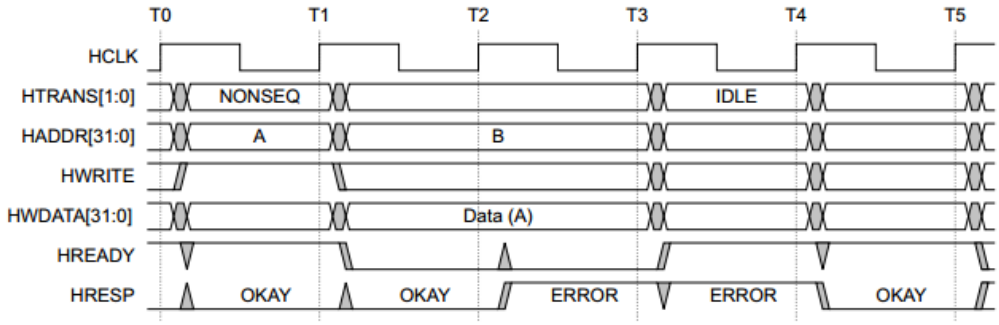

Figure 3.16: ERROR Response

3.5 Clock and Reset

Each AHB-Lite component uses the system clock signal(HCLK).All signals are sampled at the rising edge of this clock.Also each component is reset by the (HRESETn) signal,which is an active LOW signal.The reset can be asserted asynchronously,which is how it has been done in this project,but is de-asserted synchronously after the rising edge of HCLK.

Chapter 4

AMBA Peripheral Bus

This chapter deals with the definitions of APB protocol.It also describes about the various signals present in this protocol,how they're connected to the bus,the type of transfers and the responses for various transfers from the slave.

4.1 Introduction to APB and specifications

It is a simpler protocol than the AHB.It is used with general-purpose peripherals such as timers,interrupt controllers,UARTs and I/O ports.Their connection to the main system bus is through the system-to-peripheral bus bridge that helps in synchronizing the clock difference and also reduces the power consumption.It is non-pipelined protocol.All signals are sampled at the rising edge of the clock.

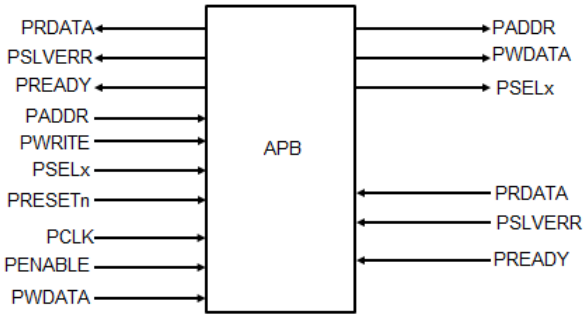

Figure 4.1: Slave Interface

The Fig 4.1 shows the AHB-Lite Slave Interface.

4.2 Transfers

List Of Signals

Name	Description
PCLK	This is the clock signal.All the signals are sampled at the rising edge of the clock
PRESETn	This is the reset signal.It is an active low signal.It is generally connected to system reset signal
PADDR	Address bus.Can be parameterized
PWDATA	Write data bus.Can be parameterized
PWRITE	This signal indicates the transfer direction. When LOW this signal indicates a read transfer.When HIGH this signal indicates a write transfer
PRDATA	Read data bus.Can be parameterized
PREADY	This signal indicates if the slave is ready for the transfer or not. HIGH indicates that the slave is ready. LOW indicates that the slave is busy and wait states are inserted in the transfer
PENABLE	This signal indicates that the master is ready for transfer.Indicates the second and subsequent cycles of the transfer
PSLVERR	Response regarding the transfer, from the slave. LOW indicates that the transfer is a success. HIGH indicates a failure
PSELx	The APB bridge unit generates the select signal to each peripheral bus slave. It indicates that the slave device is selected and that a data transfer is required

Table 4.1: List of signals

Transfers are of two types viz.,With and without wait states.Every transfer takes atleast 2 cycles since they have to pass through a FSM which is shown later in this chapter.There are two phases of each transfer viz.,Setup Phase,Access Phase.In the Setup phase the address and control signals are asserted.In the following clock cycle

the PENABLE signal is asserted which takes the APB to the access phase.The address and control signals all remain in the same state in this phase.Refer Table 4.1

The Fig 4.2 shows a simple APB read transfer. The Fig 4.3 shows a simple APB write transfer.

At the end of Access phase the transfer is completed.The PENABLE signal is

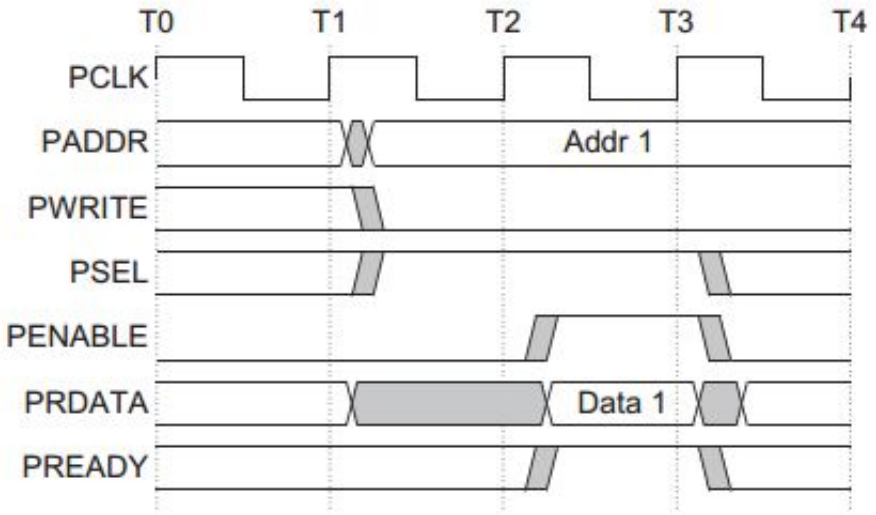

Figure 4.2: Read transfer with no wait states

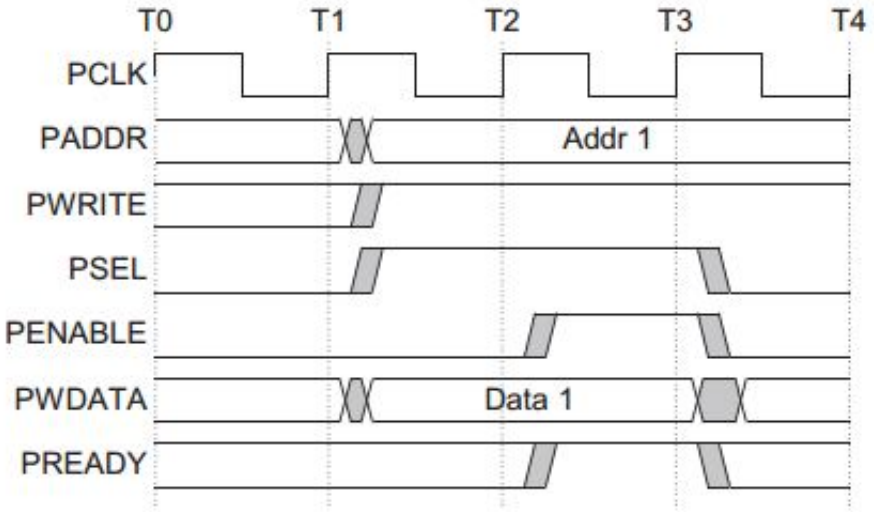

Figure 4.3: Write transfer with no wait states

driven low.The PSELx can be driven low unless there is an immediate transfer from the same slave. Similar to the AHB protocol,slaves can insert wait states here also using the PREADY signal. In the access phase,if the PREADY is LOW then wait state is inserted.The signals remain unchanged during this state.Once the PREADY is driven HIGH,the transfer completes in a cycle.

The Fig 4.4 shows how the PREADY signal from the slave can extend the transfer. The Fig 4.5 shows how the PREADY signal can extend the transfer.

PSLVERR indicates the error response of the transfer.It is considered valid only when the PREADY, PENABLE,PSEL are HIGH i.e., the access stage of the transfer

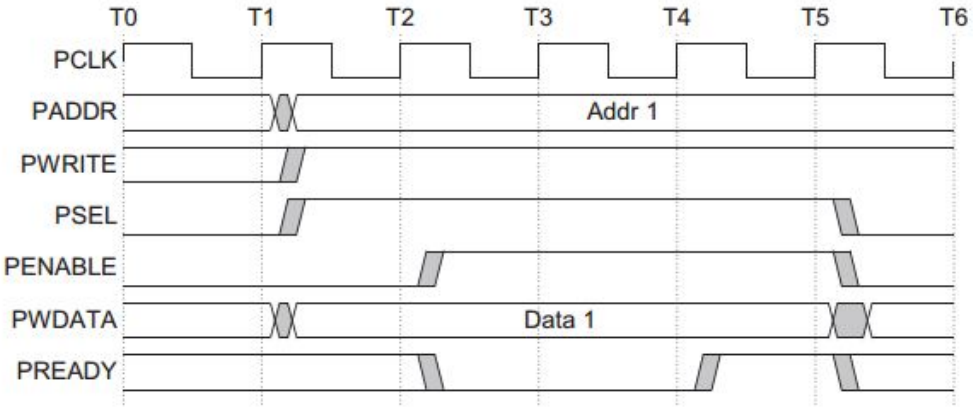

Figure 4.4: Write transfer with wait states

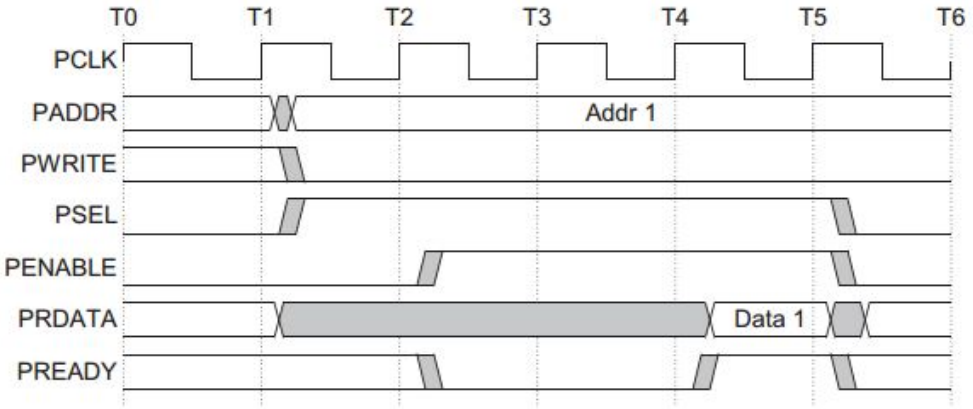

Figure 4.5: Read transfer with wait states

where the data corresponding to the address present on the address bus is transferred.PSLVERR is mapped to the HRESP signal of the AHB.

The Fig 4.6 shows a read transfer completing with an error response. The Fig 4.7 shows an example of failing write transfer that completes with an error.

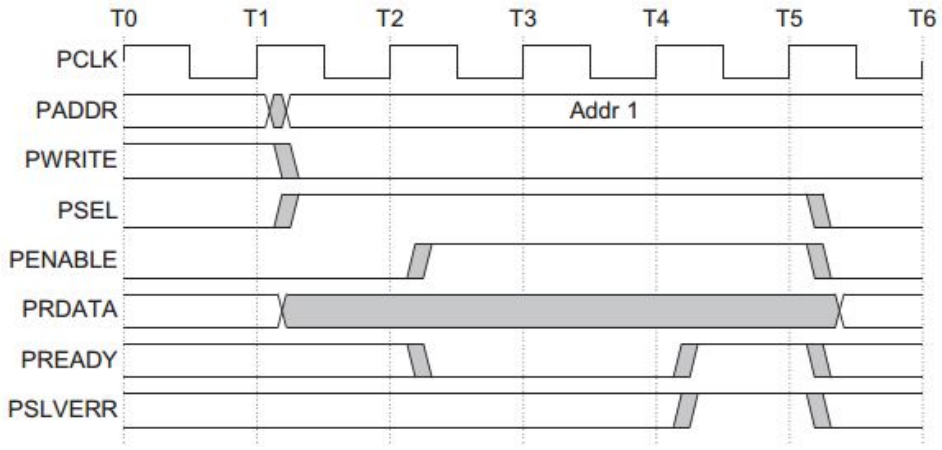

Figure 4.6: Failing read transfer

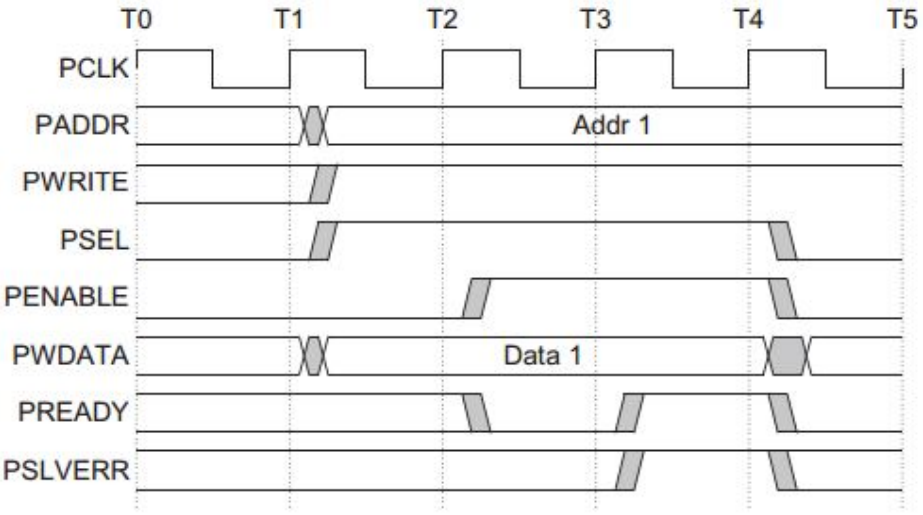

Figure 4.7: Failing write transfer

4.3 Operating States

The Fig 4.8 shows the operational activity of APB.

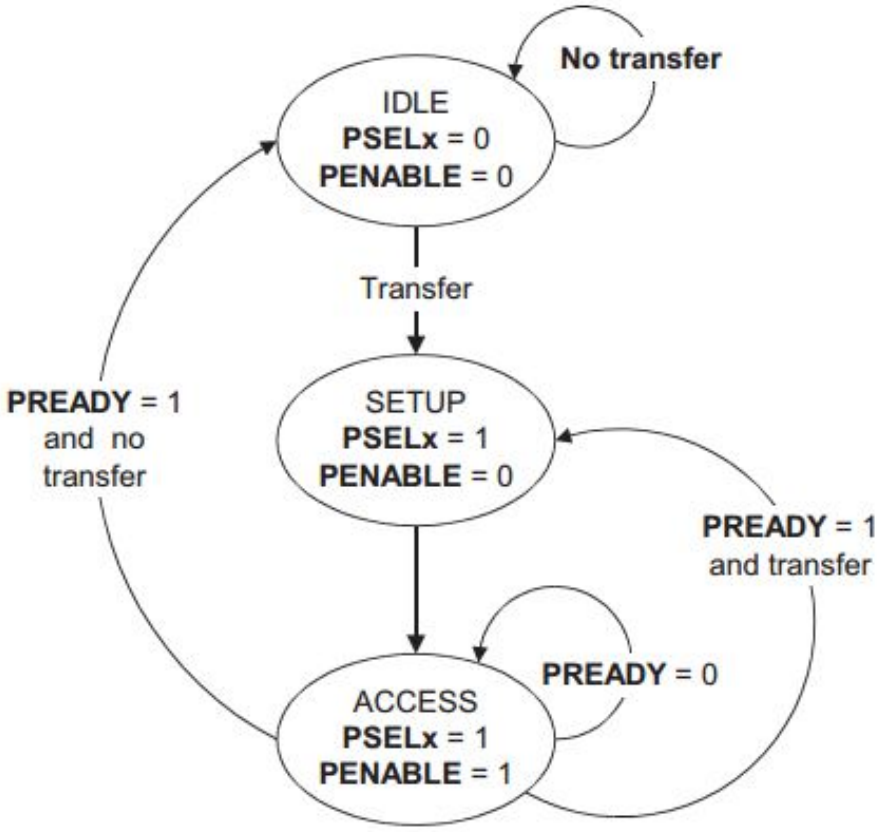

Figure 4.8: State Diagram

IDLE is the default state of the APB.APB shifts to the SETUP state when the PSEL is asserted. The bus remains in this state only for 1 clock cycle.Then it moves onto the ACCESS state when the PENABLE is asserted.All the other signals are maintained at their respective levels during this transition.If the PREADY is low the bus stays in the ACCESS state else it moves to SETUP state if there is an immediate transfer following or else moves to IDLE state.

Chapter 5

APB Bridge

This chapter deals with the definitions of APB Bridge protocol.It also describes about the two sides of the APB bridge,the system bus side and peripheral bus side,how the signals of one side are mapped onto the another. The Fig 5.1 shows the AHB to APB bridge module diagram.

5.1 Introduction to APB Bridge and specifications

The AHB-APB bridge acts as a slave to the AHB and a master to the APB,providing interface between the high-speed AHB and the low-speed APB.Transfers and the respective responses of APB are accordingly mapped onto the AHB through the bridge interface.Since the APB is not pipelined and is slow,the bridge has to insert

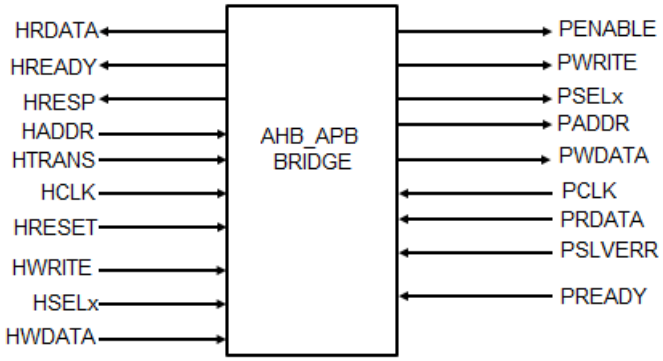

Figure 5.1: AHB to APB bridge module

wait states into the AHB when there are transfers to and from the APB.All the address and data signals from the AHB such as HADDR, HSEL, HWRITE, HWDATA, HRDATA, HRESP are mapped onto PADDR, PSEL, PWRITE, PWDATA, PRDATA, PSLVERR respectively. Here the signals carry their original description mentioned above.

Chapter 6

Integration

This chapter deals with the connections between the C-Class core and various peripherals through the bus protocols mentioned above.It briefly explains about the various packages in this chain and how different signals of one bus are mapped onto another .It is the core part of my project work.It also discusses the various test cases run to check the correctness of the code written and of the timing cycles.

The following Fig 6.1 shows a C-Class core.

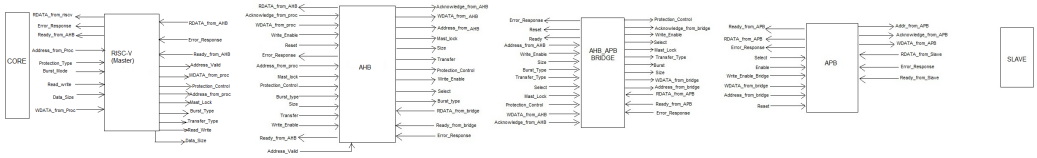

Figure 6.1: Integration of all modules

As the block diagram explains the RISC-V master and the AHB put together make up the AHB Master interface.The Bridge and AHB_slaves make up the AHB slave interface. Also the Bridge is the master for the APB and the APB_Slave make up the APB slave interface.

6.1 RISC-V master

In this package,an instance of the core module is created which sends information about the transfer such as the protection type,address,burst type,direction of the

transfer,size of transfer and data in case of a write transfer.The RISC-V master starts in an IDLE state and takes care of the transfer type.This package sends out the address,direction of transfer,size of transfer,burst type,protection type,transfer type,mastlock and write data to the AHB package.In short the RISC-V module acts as a buffer between the core and the AHB module except for generating the transfer type and mastlock for AHB.The processor is so designed that it sends the next valid address to RISC-V only when the present address transfer has been completed successfully.

Also as mentioned in AHB,each transfer that starts IDLE states goes to the NONSEQ state and only then to the SEQ state even if is multi-beat burst transfer.It also sends an address valid signal to the AHB.The use of this signal can been seen in the next module (AHB) discussed below.Since my project involves only a single master there is no need of HMASTLOCK signal.Hence it is left as default HIGH value. In case of a READ transfer the read data,response from the AHB module is sent to the core.AHB can insert wait states using the ready signal as mentioned previously.

6.2 AMBA High-Performance Bus

In this package,we will see the whole logic explained earlier in Chapter 3.To differentiate between each signal,each signal is updated in different rules.The address valid signal the RISC-V sends to the AHB is used to indicate whether or not to enqueue the FIFOs in AHB.In the absence of this signal the FIFOs get enqueued with garbage values when the RISC-V doesn't send any address to the AHB.In case of single beat burst(NONSEQ) transfer,the top elements of the FIFO are directly latched to the outputs of AHB.

In case of multiple beat burst transfer,as we've seen already,the first beat needs to be NONSEQ in nature and the remaining are SEQ.This part is taken care in the RISC-V.In the AHB module a fire signal is introduced which is HIGH by default.In the case of multi-beat burst transfers,after the initial NONSEQ transfer the fire signal is made LOW which activates all the rules which calculate the values of various signals for the remaining beats in the burst and also deactivates all the rules which fire in case of NONSEQ transfer.Hence even when there are SEQ and NONSEQ transfers alternating each other,the rules don't conflict,the FIFOs are not dequeued until the SEQ transfer is completed.A separate counter is implemented to keep track of the

number of beats left in the burst.As we've seen in Chapter 3,all the signals except address coming from the master are maintained at the respective levels till all the beats are completed in a multiple beat burst transfer.

The slave selection is based on the address.As explained in the AHB protocol,the minimum address boundary for each slave is 1KB,which is how the selection has been done in this project.In case of a read transfer the top elements in HRDATA and HRESP FIFOs are latched onto the output of the AHB which is sent to RISC-V.In case the AHB is busy it can extend the transfer by lowering the HREADY signal sent to the RISC-V module.

6.3 APB Bridge

In this package we see how the signals from faster bus are mapped onto a slower bus. Bluespec supports SyncFIFOs which help us with this interface.They're capable of being clocked by two different clocks,one on the input side and another on the output side.The reset signal of each clock helps reset the FIFO's corresponding side.Address,Control and Data signals from the AHB are enqueued into the SyncFIFOs and the APB dequeues them

In case of a read transfer from the core to the general-purpose peripheral,a tag bit has been implemented,from the bridge module,towards the peripherals side.The reason for using this tag bit is that in the AHB module the data is transferred immediately after the address is sent in case no wait states are inserted for any direction of transfer.However ,in the present scenario, for a read transfer atleast six cycles difference exists between the address sent from AHB to the data received by AHB.So the data received and the next addresses get messed up.Hence we stall the bridge,in case of READ transfer from the AHB,till we receive back the tag bit we sent.

6.4 AMBA Peripheral Bus

In this package,we see the whole logic explained earlier in Chapter 4.Since the APB needs atleast two cycles to complete a transfer FIFOs are implemented to speed up the transfers from the APB Bridge.Three different rules have been implemented to identify each state of the APB.As seen before the APB accepts new address and

control signals only in the IDLE state or the ACCESS state.Hence any address and control signals sent by the APB Bridge when the APB is in the SETUP state are lost.The FIFOs accumulate this signals preventing the loss of transfers.They implicitly prevent the APB Bridge from sending new transfers in case they are full,since the corresponding rules don't fire.Their is a trade off with the increase in the hardware though.

In the read operation,the tag bit is received from the APB Bridge along with the address. It is forwarded to the Peripheral Slaves along with the address.The APB receives the data along with the Tag bit which is then forwarded to the APB Bridge to indicate the completion of the read transfer.

6.5 AHB Slaves

In these packages,we see the dummy memory implementation.Bluespec supports Register File which is a register bank.In this case the memories are connected to the AHB bus directly. As we've seen in Chapter 3,the address phase is completed and immediately data phase follows unless wait states are inserted by the slave itself.So,in case of a write operation,the address is delayed by one cycle so that the slave receives the data related to that particular address in the next cycle.In case of a read operation,the data is put into a register ,as soon as the address is received,which itself causes a cycle delay.So the AHB effectively sees the address phase and the data phase.Tag bit is not required in these slaves since the transactions follow the timing diagrams presented in Chapter 3.

6.6 APB Slaves

As explained above,we see the dummy memory implementation in these packages.In this case the memories are connected to APB bus.As we've seen in Chapter 4,the address maintains its value till the data transfer occurs in the ACCESS phase.So in case of a write operation the memory is updated accordingly with no cycle delays .In case of a read operation the APB stays in the ACCESS state till the corresponding data appears and tag bit is received ,along with address, from APB and sent back to APB, along with the data.

6.7 Verification

Here we shall see how the integration built is verified in BSV and the various test cases used for this verification.The verification process mainly focuses on the functional behaviour of the whole integration package.The main challenge involved in the verification was the integration.The cycle delay that occurs in each module affects the functioning of next module.So aligning all signals accordingly was the bottle neck.The implementation of FIFOs eased the cycles problem.In this project the address FIFO is taken as reference ,for the set of address and control signals,to check the FIFO full and empty conditions.It is a valid assumption since all the address and control signals for a given transfer come in the same cycle.The signals which differ in the cycles are DATA signals and ready signal.Ready signal comes from the slave.So the conditions for these signals have been done accordingly.The test bench module connects all the interfaces as shown in the integration package and sends inputs the RISC-V module.

These are the test cases used to verify the design:

- AHB,APB modules tested individually.
- Peripheral Slave modules tested individually.
- Only NONSEQ transfers.
- Only SEQ transfers.
- SEQ and NONSEQ transfers continuous.
- SEQ and NONSEQ transfers with IDLE transfers in between

Also the whole design including the core is tested using ISS-AAPG environment.AAPG generates approximately 20000 odd instructions in each test case.It also gives the final output in the memory and registers after test case is run.When the corresponding instruction and data memory is loaded in the register files in the system slaves and the test bench is run,the output memory and register files are generated which have matched with files earlier generated by AAPG.

6.8 Synthesis

Bluespec Compiler converts the design in BSV to synthesizable Verilog code.The design has been synthesized using Xilinx ISE for Virtex 5 xc5vlx110t-1-ff1136.All default settings were used.The design strategy was set to reach the expected range of frequency of operation.The slice utilization and timing summary for various cases are provided below.

Core to AHB:

```
Timing Summary:
---------------
Minimum period: 2.305ns
Maximum Frequency: 433.839MHz
Minimum input arrival time before clock: 4.589ns
Maximum output required time after clock: 4.118ns
```

Integrated Package:

```
Timing Summary:
---------------
Minimum period: 6.809ns
Maximum Frequency: 146.864MHz
Minimum input arrival time before clock: 5.183ns
Maximum output required time after clock: 3.398ns
```

APB to Slaves:

```
Timing Summary:
---------------
Minimum period: 6.719ns
Maximum Frequency: 148.832MHz
Minimum input arrival time before clock: 5.673ns
Maximum output required time after clock: 4.659ns
```

Chapter 7

Conclusion and Future work

7.1 Conclusion

The AHB,APB Bridge,APB protocols have been successfully implemented ,in BSV, connecting the C-Class core to various peripherals.The constraints and logic explained in the protocols have been taken into consideration.Slaves at multiple levels(System and Peripheral) have been implemented to verify the working of different kinds of integration possible viz., Core to AHB slave,APB Bridge to APB slave,Core to APB slave.

7.2 Future Work

In the existing design,the following things can be implemented further to enhance the working of bus protocols:

- multi-layer AHB-Lite to implement multiple masters using the AHB-Lite protocol.
- locked transfers in case of multiple masters and multiple slaves to avoid errors due to shared memories.
- protection control if multiple masters are implemented.This enhances the level of protection to each transfer.In this project,instruction memory is implemented as read-only.However provisions can be made to add instructions into those.The corresponding code is already written and commented.

Bibliography

[1] Bluespec,Inc.Bluespec System Verilog Reference Guide, Revision 30 July 2014.

[2] ARM Limited. AMBA 3 AHB-Lite Protocol v1.0, Revision March 2010.

[3] ARM Limited. AMBA 3 APB Protocol v1.0, Revision 17 August 2004.

[4] ARM Limited. AMBA Design Kit v.r3p0, Revision August 2007.

[5] ARM Limited. AMBA University Kit v.r0p0, Revision September 2001.

[6] RISE Lab, Shakti Series Processors.ppt