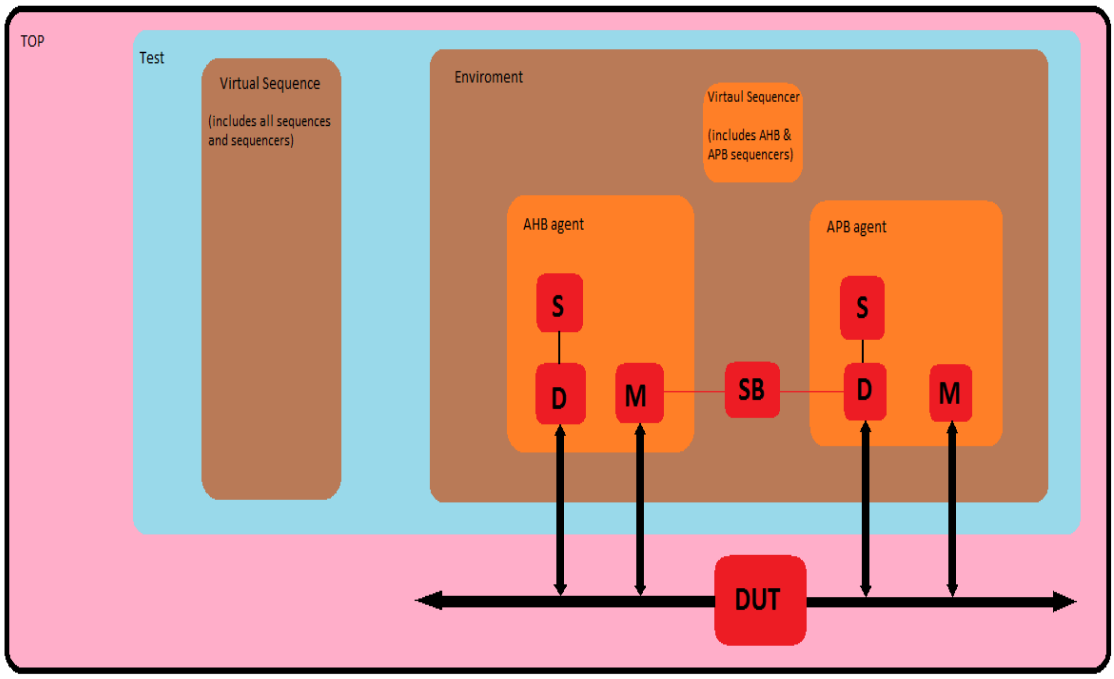
TESTBENCH ARCHITECTURE



V PLAN

Features

AHB SIGNALS

Signal names	Source	Description
Hclk	Clock source	Clock for all transfers
Hresetn	Low reset ctrl	Active low reset control
Haddr[31:0]	Master	32 bit address bus
Htrans[1:0]	Master	Type of transfer(idle, busy, seq, non-seq)
Hwrite	Master	High = write transfer Low = Read transfer
Hsize[2:0]	Master	Size of transfer (byte, halfword, word) max 32 bits
Hburst[2:0]	Master	Type of burst (Wrap or Incr)
Hwdata[31:0]	Master	Write transfer data from master to slave

Hrdata[31:0]	Slave	Read transfer data from slave to master
Hready	Slave	It should be high for transfers
Hresp[1:0]	Slave	Additional info on the status of a transfer (Okay, Error, Retry, Split)

APB SIGNALS

Signal names	**Description**
Pclk	Clock for all transfers
Presetn	Active low reset signal
Penable	Used to time all access on peripheral bus
Paddr[31:0]	32 bit address bus
Pwrite	High = Write transfer Low = Read transfer
Prdata	Transfer data to master when Pwrite is low
Pwdata	Data transfer from master to apb slave when Pwrite is high
Pselx	Selection signal for slaves

Transactors

- Since we do not have real time source and destination so we are using agents
    1. AHB Master agent.
    2. APB Slave agent.
- Required signals are written in transaction class and some of them are randomized based on the requirement.
- Different sequences are written for all burst operation for read and write.
- Sequencer acts as a routing component between sequence and driver.
- Driver is used to drive the signals to DUT, seq_item_port and export is used to connect between driver and sequencer.

- Monitor is used to sample the signals from DUT and send it to Scoreboard for comparison.
- SB compares from both monitors and provides compared output.

Testcases

- There are 2 types of transfers Wrap and Incr total 16 Test cases (8 for read and 8 for write)

Hburst [2:0]	**Type**
000	Single transfer
001	Undefined Increment(32)
010	Wrap4
011	Increment4
100	Wrap8
101	Increment8
110	Wrap16
111	Increment16

Coverage model

- Coverpoints are created.
- Bin creation can be done implicitly or explicitly.
- Cross products between Bins to Bins or Bins to coverpoints.
- Regression testing can give coverage up to 100%.

AIM & INTRODUCTION

AIM:

This project deals with the class based verification of bridge between high speed AMBA AHB(Advanced High Performance bus) and low-power AMBA APB (Advanced Peripheral Bus) in UVM.

INTRODUCTION:

- **Advanced High Performance bus (AHB):**

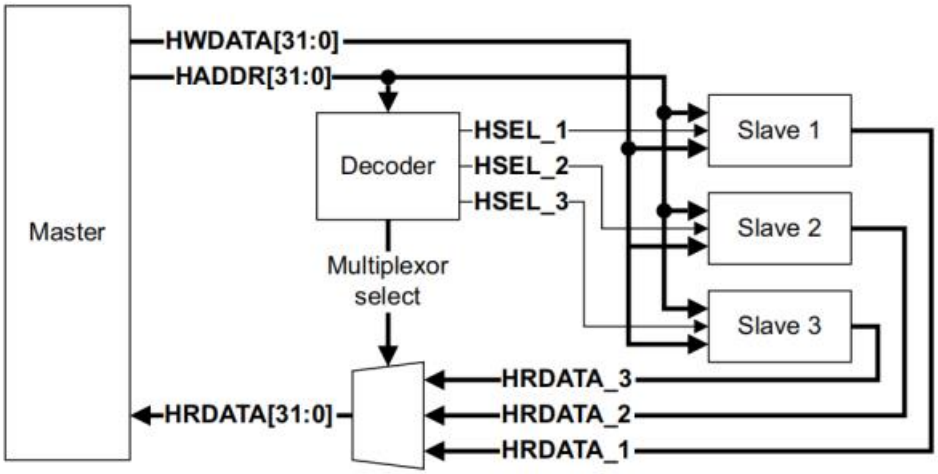


Figure : AHB Block Diagram

The AMBA AHB is for high-performance, high clock frequency system modules.

The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques. It supports multiple masters and multiple slaves, It supports Pipelined operation and has 2 phases data and address, It supports max up to 1024 bits but in our project we set it to max 32 bits. Burst operation is also possible, It supports up to 15 masters and extra 1 dummy master is used which stores the data sent by the slaves when all the masters are full. And one Default master is used which initiate the master communication when none of the masters want to communicate with the slaves. Hready signal allows insertion of max 16 wait states. Because of slave accepts only max 1kb data so it is limited to send maximum of 1Kb of data.

- **Advanced Peripheral Bus(APB):**

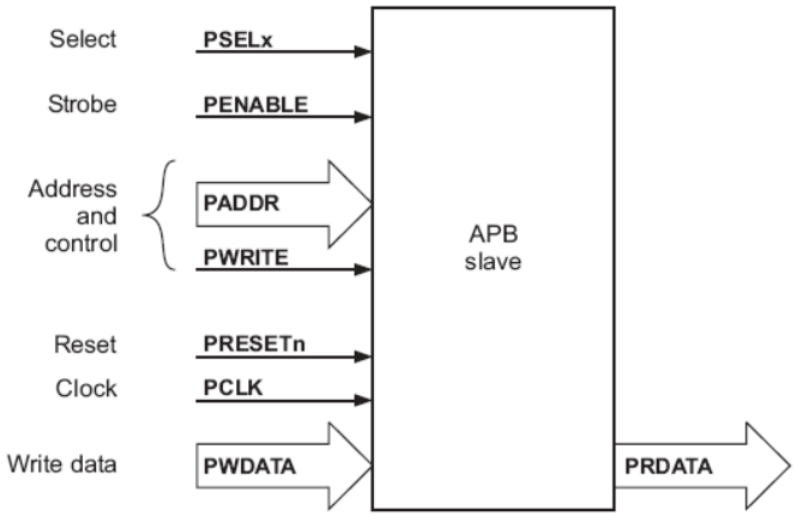


Figure : APB Block Diagram

The AMBA APB is for low-power peripherals.

AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus. It is low overhead only 4 control signals, supports only single master and multiple masters, It has 3 states idle, setup, enable. Slave is non-responsive it always takes two clock cycles for transfer in APB slave data is always latched between the two clock cycles.

- **Functions of Bridge:**

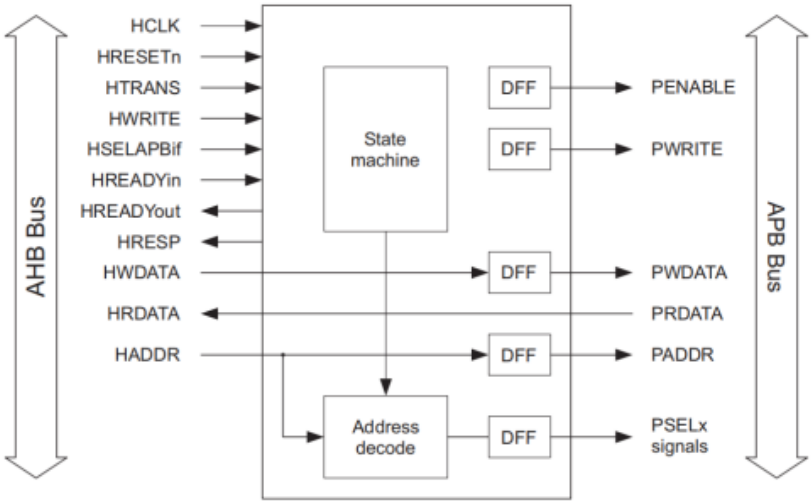


Figure : AHB to APB Bridge Block diagram

1. Latches the address and holds it valid throughout the transfer.
2. Decodes the address and generates a peripheral select, PSELx. Only one select signal can be active during a transfer.
3. Drives the data onto the APB for a write transfer.
4. Drives the APB data onto the system bus for a read transfer.
5. Generates a timing strobe, PENABLE, for the transfer.
6. Here AHB is Master and APB is slave, So bridge acts as a AHB slave and APB master.

OBJECT CLASS

Transaction

Transaction class acts like a blue print for the Testbench creation. We created AHB xtn class for ahb transactions which is extended from uvm_sequence_item and APB xtn class for apb transactions which is also extended from uvm_sequence_item.

- AHB xtn logic : Randomizing the driving signals and other signals are not randomized, and writing constraints for required signals which should obey the protocol and printing it.
- APB xtn logic : Randomizing the only one driving signal and printing it.

Sequence

In sequence class we are starting multiple sequences, we are writing only AHB sequence class so in this seq class from base sequence class we are extending 16 sequences i.e 8 burst operations for write sequence and 8 burst operations for read sequence, For single sequence only Non sequential transaction is sent for write it is kept Hwrite as 1 for read Hwrite as 0 with Hburst as 000 and Htrans as 2.

Htrans [1:0]	Type
00	Idle
01	Busy
10	Non-Seq
11	Seq

For Increment 1st transfer of burst will be Non seq and remaining will be sequential after starting Non-seq sequence we will store it in local registers and start the burst operation for remaining seq transfer it will be done based on the size, all the signals are given to stored local registers but

address calculation is there if Hsize is 0 ,1 byte is sent and Hsize is 1 , 2 byte is sent , if Hsize is 2 , 4 byte is sent in this calculation carry is not neglected.

For Wrap calculation will be the same as Increment but address calculation differs there is no overflow to the next slave so carry will be neglected in wrap calculation so when is reaches the end of the salve address it goes back to the initial address.

Virtual Sequence

Virtual sequence class is extended from uvm_sequence and is parameterized with uvm_sequence _item. It consists of handles of virtual sequencer , all sequencers and all the sequences. All the local sub-sequencers are made to point to sequencers inside virtual sequencer. This is possible by casting m_sequencer and virtual sequencer. Any sequence can be started using start method and it is started on the sequencer.

TESTBENCH COMPONENTS

Driver

Driver is extended from uvm_driver and is parameterized with respective transaction class. The driver is connected to DUT through virtual interface with the help of config file. The driver class has TLM seq_item_port which is used to connect with the sequencer.

During the run_phase of the driver , data is collected from sequencer using get_next_item and acknowledgement is sent back by item_done.

- AHB driver logic: Wait till Hreadyout is high send control signals along with Haddr in the first clock cycle. In the next clock cycle check if Hreadyout is high and then send Hdata to DUT.
- APB driver logic: Wait till Penable is high and then drive Randomised Prdata when Pwrite is low.

Monitor

Monitor is extended from uvm_monitor and is connected to DUT through the virtual interface with the help of config file.

- AHB monitor logic: Wait till Htrans is 2 or 3 and then collect control signals from the interface . In the next clock cycle wait till Hreadyout is high and then collect data signal.Create a copy of collected data and write into the tlm_analysis_fifo using 'write' method.
- APB monitor logic: Wait till Penable is high and check for Pwrite . If Pwrite is high then collect Pwdata and Paddr else collect Prdata, Paddr and Hrdata from the DUT. Send these collected items to second tlm_analysis_fifo in the scoreboard using 'write' method.

Sequencer

The sequencer is extended from uvm_sequnecer and is parameterized with respective transaction class. It is connected with the driver using seq_item_export.

Agent

Agent class is extended from uvm_agent . In the build_phase of agent, based on whether the agent is active or passive, driver, sequencer and monitor is built. In the connect_phase, connection between driver and sequencer is made.

Virtual sequencer

Virtual sequencer class is extended from uvm_sequencer and is parameterized with uvm_sequence_item. It consists handles of AHB and APB sequencers.

Environment

Environment class is extended from uvm_env. In the build_phase of environment AHB agent, APB agent , Virtual sequencer and Scoreboard is build. The connection between Scoreboard's tlm fifo and monitor's analysis port is made in the connect_phase.

Scoreboard

Scoreboard is extended from uvm_scoreboard and consists of 2 tlm fifos for collecting the items from AHB and APB monitor.

Get the data from AHB monitor and if Hwrite is high , push Hwdata and Haddr into locally declared queue. Get the data from APB monitor , simultaneously pop from queue and compare Hwdata with Pwdata and Haddr with Paddr.
If Hwrite is low, compare Hrdata with Prdata and Haddr with Paddr. Covergroup is included for signals which are to be tracked.

TESTCASE DETAILS

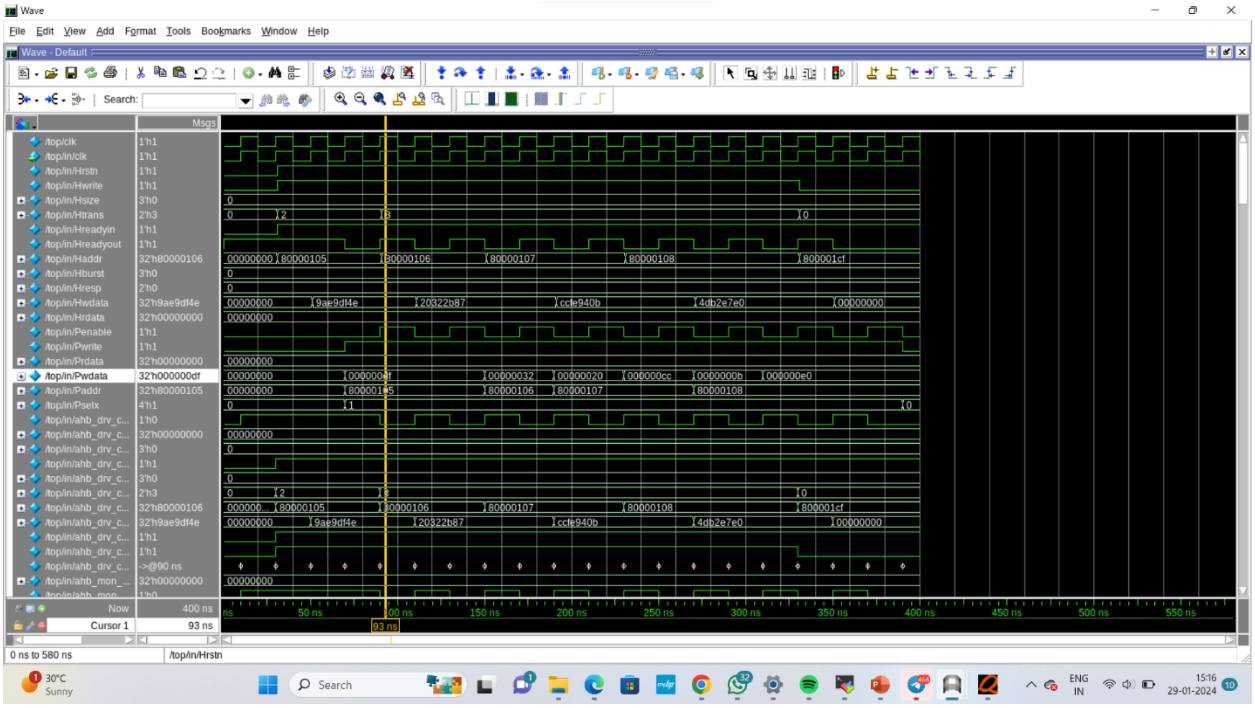
Hwrite, Hsize(constraint added so the value is between 0&2) are randomized and hence with each testcase different scenarios like write/read and different Hsize is covered.
These are the different testcase that were covered, (16 test cases)

1. Single transfer
2. Undefined increment.
3. 4-beat wrapping burst.
4. 4-beat incrementing burst.
5. 8-beat wrapping burst.
6. 8-beat incrementing burst.
7. 16-beat wrapping burst.
8. 16-beat incrementing burst

EXPECTED WAVEFORMS

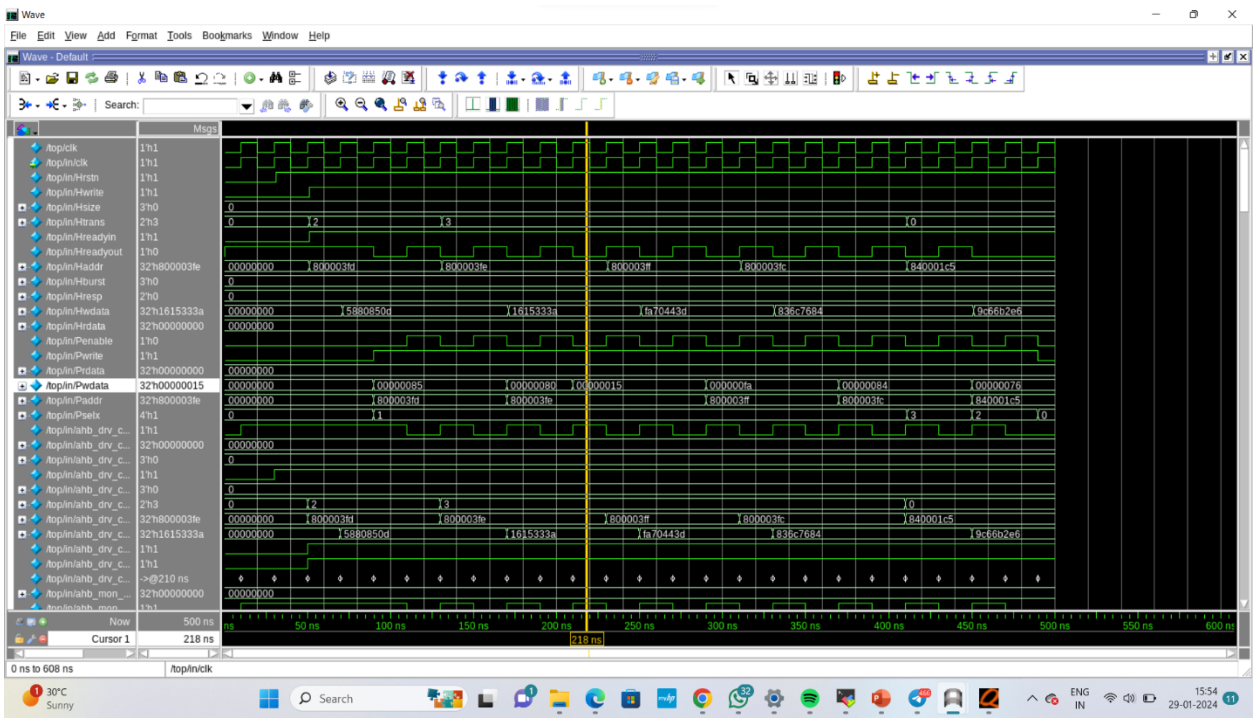
Increment 4 write operation

Here the Burst operation for Increment 4 is 3(011) , In this operation first transfer must be non-sequential , In 1st clock cycle the address and other control signals are sent and in the next clock cycle data signals are sent so it performs pipelined operation, so the next 3 transfers will be sequential and through bridge based on the address APB slave samples the Pwdata as it is non responsive it takes two clock cycles to sample the output. At the end of every burst operation we are sending a Idle transfer to complete the operation.



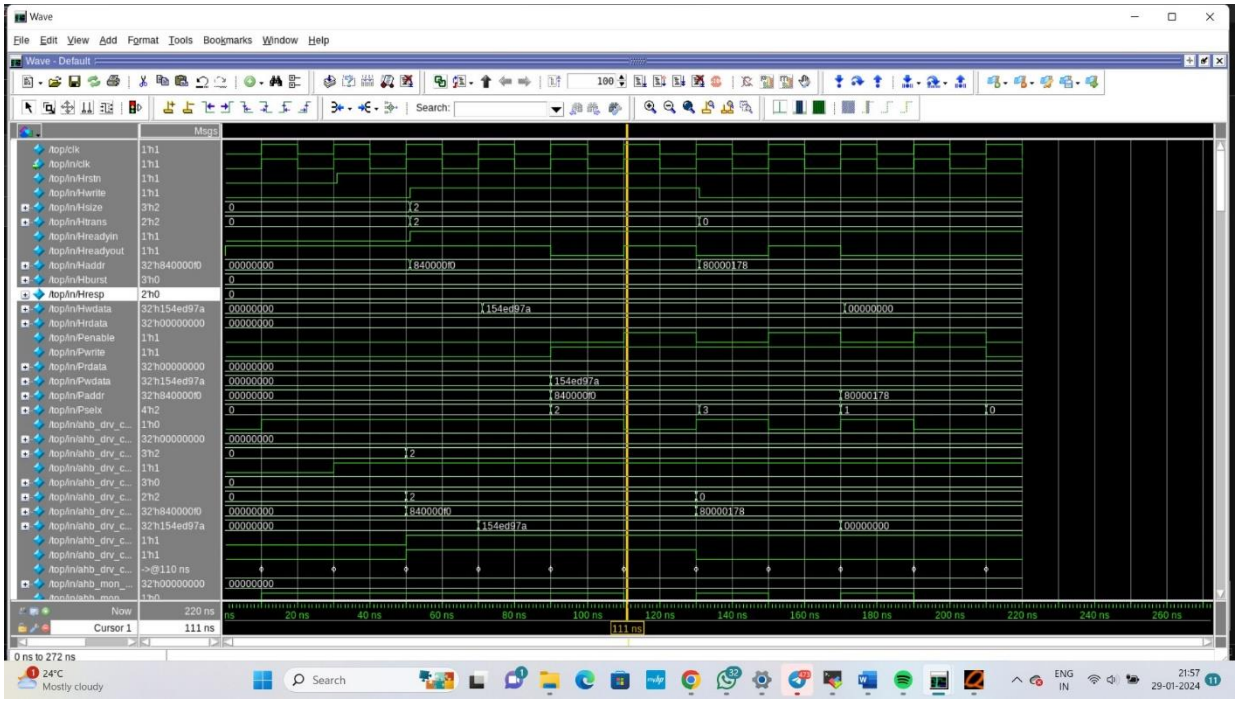
Wrap 4 write Operation

Here the Burst operation for Wrap 4 is 2(010), It follows the same steps as in Increment 4 but the address will not go out of bound instead it will roll back to its initial address.



Single write transfer

Here the Burst operation single transfer is 0(000). A non-sequential transfer will be done from master to slave.



Take away from the project

- Learnt Why AMBA protocols are required and its importance in the on-chip communication.
- Conversion between AHB to APB protocol, Importance of bridge.
- Sequence creation and execution of sequence as it is the important part of the project.
- Analyzing the pipelined waveforms.
- Main take away is its specification as it is the tricky part of the project.

There was no such difficulty in the project but the tricky part is the specifications and analyzing waveforms.