AHB-Lite Master

Sarah Hamed Mahmoud Al-sayed

July 26, 2025

Contents

Introduction

The Advanced High-performance Bus (AHB) protocol is part of ARM's AMBA (Advanced Microcontroller Bus Architecture) specifications designed for on chip communication. AHB employes a pipelined architecutre as well as support to multiple master and slave communication increasing the throughput and bandwidth of transactions. In this documentation we will go through the implementation and testing of AHB-Lite master.

Design Features and Simplifications

The features supported in the implemented AHB-Lite master are:

- Pipelining
- Wait states via **HREADY**
- Transfer types of :
  - Idle(**IDLE**)
  - Non Sequential (**NONSEQ**)
  - Sequential (**SEQ**)
  - Busy (**BUSY**)
- Burst transfers of type:
  - **SINGLE**
  - **INC**: undefined burst length
  - **INC4**: burst of fixed length 4
  - **INC8**: burst of fixed length 8
  - **INC16**: burst of fixed length 16
- Transfer size from 1 byte up to 32 words
- Busy Transfer insertions mid bursts
- Error handling via **HRESP**

The features not supported are:

- Wrap bursts
- Locked Transfers (**HMASTLOCK**)
- Burst Termination
- Protection Control (**HRPORT**)
- Multi-layer AHB

For error handling while it wasn't a strict requirement in the protocol to end a burst transfer when error is raised **HRESP** = 1, this implementation terminates burst transfer in case of error and returns to **IDLE** state.
Additionally, in this implementation inputs signals to the master are driven synchronously with the rising edge of **HCLK** . This is done to ease the testing of the AHB master and ensure it can transfer input transactions correctly to subordinates.

Input Signals and Diagram

The AHB-Lite master is abstracted in the diagram shown in Figure 1 below.

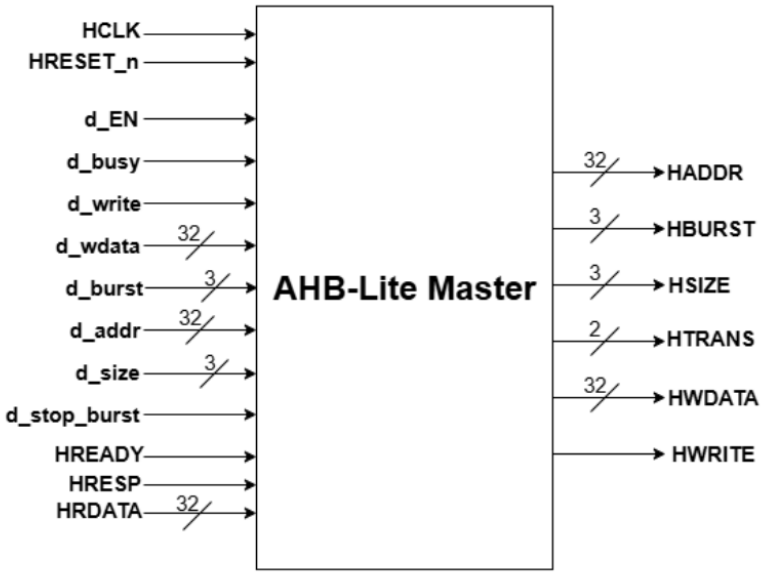

Figure 1: AHB Master Abstracted Diagram

The master receives the following input signals from a driver which is usually a processor:

- d_EN : if 1 indicates a read/write transfer otherwise idle transfer
- d_busy: indicates if driver is busy
- d_write: indicates if request is write or read
- d_wdata: data to be written to slave
- d_burst: indicates if requested transfer is a burst and of what type
- d_addr: address of the requested transfer
- d_size: size of the transfer
- d_burst_stop: stop burst for burst of undefined length

FSM and Design Methodology

According to the specifications of the AHB protocol transactions from the master are issued in two phases: as shown in figure 2 below.

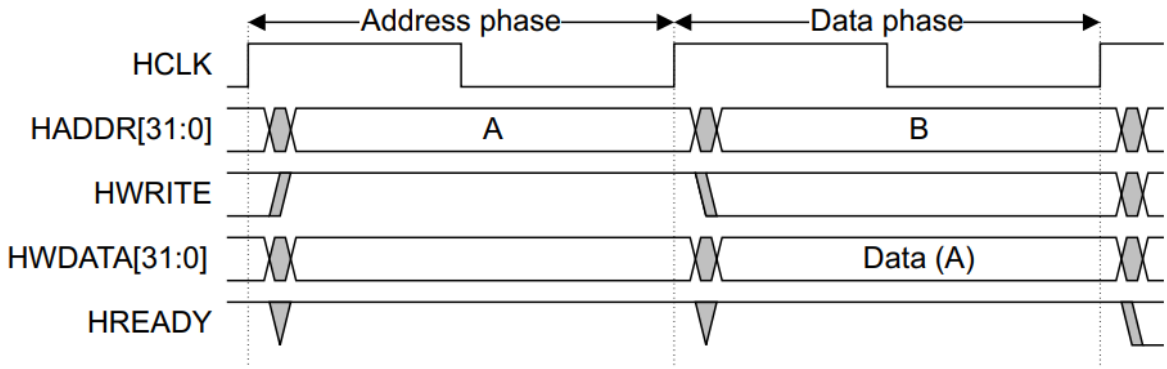

Figure 2: address and data phases of a transaction

- **Address phase:** the master drives the transaction's address and control signals onto the bus, effectively queuing the operation.
- **Data phase:** the actual transfer takes place. where:
  - For write transfers the data is put on **HWDATA**.
  - For reads, the master issues the read request.

  During the data phase the master waits for the slave to assert **HREADY** = 1 to complete the handshake.

To derive the state machine of the AHB-Lite master, the address phase of each transaction is analyzed to examine how it affect the transitioning of the master to other states as well as the master's control signals. In the address phase there are 4 possible transaction types which are:

- **IDLE**: no transfer is active.
- **BURST**: the start of a fixed-length or undefined-length burst sequence.
- **SINGLE**: a standalone (single) transfer.
- **BUSY**: The master is inserting IDLE cycles due to not being able to resume a burst transaction.

Each transaction impacts the subsequent state and control signals of the master; in the next sections, we'll explore each state and its transitions in detail.

IDLE state

In the IDLE state an idle transaction has been issued and is pending in the address phase. The transition from the IDLE state depends on the input signals from the processor as well as the response from the slave where the possible responses could be:

- **HREADY** = 1: The master can drive the upcoming transaction issued by the processor.
- **HREADY** = 0 && **HRESP** = 0: this indicates no error, however, there is a wait cycle required by the slave to complete the transation. During wait cycles as according to seciton 3.7 in the AHB protocol specifications the master is permitted to change the transfer type from **IDLE** to **NONSEQ** type and change the control signals accordingly.
- **HREADY** = 0 && **HRESP** = 1: this indicates the beginning of an error response, so in the next cycle the transfer type must remain **IDLE**. All control signals could be changed, however, **HTRANS** signal must remain **IDLE**.
- **HREADY** = 1 && **HRESP** = 1: this indicates the end of an error response, so after this the master can transition from **IDLE** state to any state in accordance to the next transaction issued by the processor.

So in **IDLE** state the master can transition to other states and take the next transaction issued by the processor. For all possible responses by the slave the master can change the control siganls for the next transaction, however, the **HTRANS** signal must not change from **IDLE** during the first cycle of an error response.

SINGLE state

In the SINGLE state an single burst transaction has been issued and is pending in the address phase. The transition from the SINGLE state depends on the input signals from the processor/driver as well as the response from the slave where the possible responses could be:

- **HREADY** = 1 && **HRESP** = 0: The single transfer can move from address to data phase where it drives its write data on **HWDATA** in case of write or issue a read request in case of read. Then the master can take the next transaction requested by the processor and transition to other states accordingly.
- **HREADY** = 0 && **HRESP** = 0: this indicates no error, however, there is a wait cycle required by the slave to complete the transation. During wait cycles all control signals must be held constant until **HREADY** is asserted HIGH. So, in this case the master remains in the same state with the same control signals.
- **HREADY** = 0 && **HRESP** = 1: this indicates the beginning of an error response, so in the next cycle the transfer type must change to **IDLE**. All control signals could be changed.

So in **SINLGE** state the master can transition to other states and change its control signals only if **HREADY** = 1. At the first cycle of an error response the master must transition to the **IDLE** state and set **HTRANS** to **IDLE**.

BURST state

In the BURST state either the beginning of middle of a burst transaction has been issued and is pending in the address phase. The transition from the BURST state depends on the number of burst transactions that took place, required number of bursts, signals that determine end of a burst for bursts of undefined burst lengths, input signals from the processor/driver that determine the next transaction as well as the response from the slave. The possible responses from the slave could be:

- **HREADY** = 1 && **HRESP** = 0: The single transfer can move from address to data phase where it drives its write data on **HWDATA** in case of write or issue a read request in case of read. If the number of bursts equal that of the required amount as determined by **HBURST** or if the processor requests termination of burst of undefined length then the master can take the next transaction issued by the processor and change its state/ signals accordingly.

  Otherwise, if the processor asserts d_busy signal to high the master transitions to a BUSY state keeping the control signals constant and only changing **HTRANS** to **BUSY** .

  Else if no busy or burst stop requests are applied nor did length of burst meet the required amount, the address **HADDR** is incremented according to the value of **HSIZE** and all control signals remain constant expect **HWDATA** for a write transfer.

- **HREADY** = 0 && **HRESP** = 0: this indicates no error, however, there is a wait cycle required by the slave to complete the transation. During wait cycles all control signals must be held constant until **HREADY** is asserted HIGH. So, in this case the master remains in the same state with the same control signals.

- **HREADY** = 0 && **HRESP** = 1: this indicates the beginning of an error response, so in the next cycle the transfer type must change to **IDLE**. All control signals could be changed also the burst count is set to 0 as the burst transaction will be aborted.

So in **BUSY** state the master can transition to other states and change its control signals only if **HREADY** = 1 , a processor asserts end of burst for bursts of undefined length or the required number of bursts is met. Also, if processor asserts busy signal to high the master transition to BUSY state and keeps all control signals constant expect **HTRANS** which changes to **BUSY**. At the first cycle of an error response the master must transition to the IDLE state and set **HTRANS** to **IDLE**.

BUSY state

In the BUSY state the master cannot forward a transaction mid-burst into the data phase. The transition from the BUSY state depends on busy signal from the processor (d_busy), request to terminate the burst transfer for undefined length bursts as well as the response from the slave. The possible responses from the slave could be:

- **HREADY** = 1 && **HRESP** = 0: If d_busy is asserted LOW the master can transition back to the BURST state and continue the burst transfer otherwise it remains in the BUSY state and keep its control signals constant.
- **HREADY** = 0 && **HRESP** = 0: this indicates no error and if d_busy is asserted LOW the master can transition back to the BURST state and continue the burst transfer or if the processor requests termination of the burst for bursts of undefined length then the master can transition to anther state and not resume the burst transfer.
- **HREADY** = 0 && **HRESP** = 1: this indicates the beginning of an error response, so in the next cycle the transfer type must change to **IDLE**. All control signals could be changed also the burst count is set to 0 as the burst transaction will be aborted.

So in **BUSY** state the master return to BURST state only if processor busy signal d_busy is asserted low. Also if **HREADY** = 0 && **HRESP** = 0 the master can terminate the burst transfer for burst of undefined length and drive next transaction as controlled by the processor. When **HREADY** = 0 && **HRESP** = 1 the master must transition to IDLE state and clear counter for number of bursts as burst transfer will be aborted.

The finite state machine diagram of the AHB master is provided in Figure 3 below to illustrate the transition of master to different states as well as the change of its control signals.

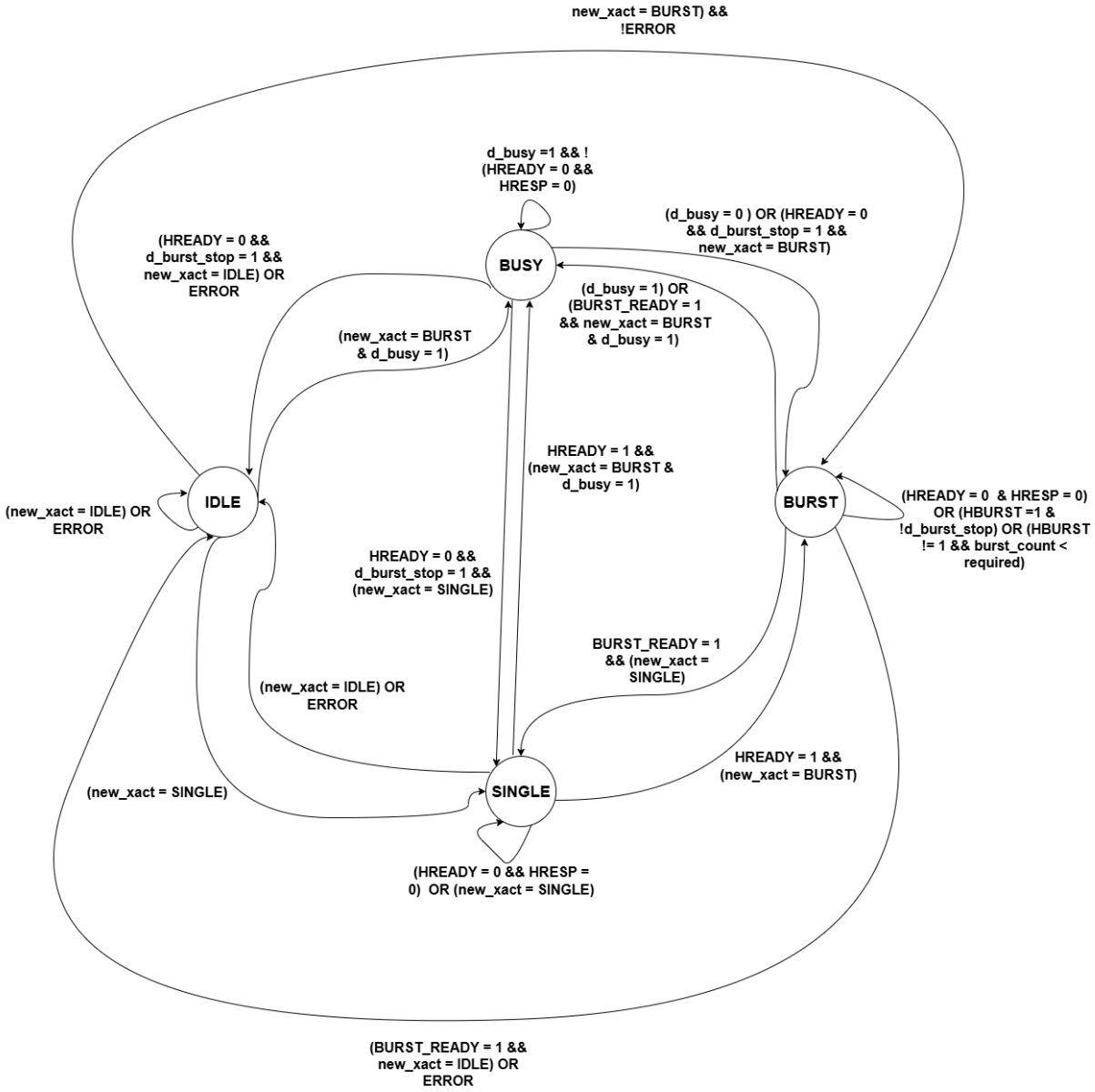

Figure 3: AHB Master FSM

Note in the diagram the signal

- ERROR = (**HREADY** == 0 && **HRESP** == 1)
- BURST_READY = (**HREADY** == 1 && (**HBURST** == 1 && d_burst_stop == 1) && (**HURST** != 1 && burst_count == required_count))
- new_xact = type of new transaction which is determined by the d_burst and d_EN signals.

The substitution was done to avoid making the diagram too cumbersome. Also the control/output signals of the master are omitted from the diagram. When loading or taking a new transaction from the processor the output signals are taken from the inputs provided by the processor.

However, a signal like **HTRANS** is set as follows:

- For IDLE state **HTRANS** = 0
- For SINGLE state or transition to BURST for the first time **HTRANS** = **NONSEQ**
- For the continuation of burst transfers **HTRANS** = **SEQ**

If stalling occurs in SINGLE or BURST state due to **HREADY** begin asserted LOW then the control signals are held constant.
In the continuation of burst transfer all control signals are held constant expect **HADDR** which is incremented by a value dependent on **HSIZE**.

Testing and Output Waveforms

To test the master, it is driven by input transactions from a processor and various response from the slave to ensure its correct functionality under different scenarios. In the test environment the slave is modeled as a D flip-flop that mirrors the address signal **HADDR** and has its enable signal be controlled by **HREADY** and **HRESP**.

Four Consecutive Reads with 0 Waits

A test was performed by sending 4 consecutive reads transaction to ensure pipelining is taking place. The output waveform for this test is shown in Figure 4 below:

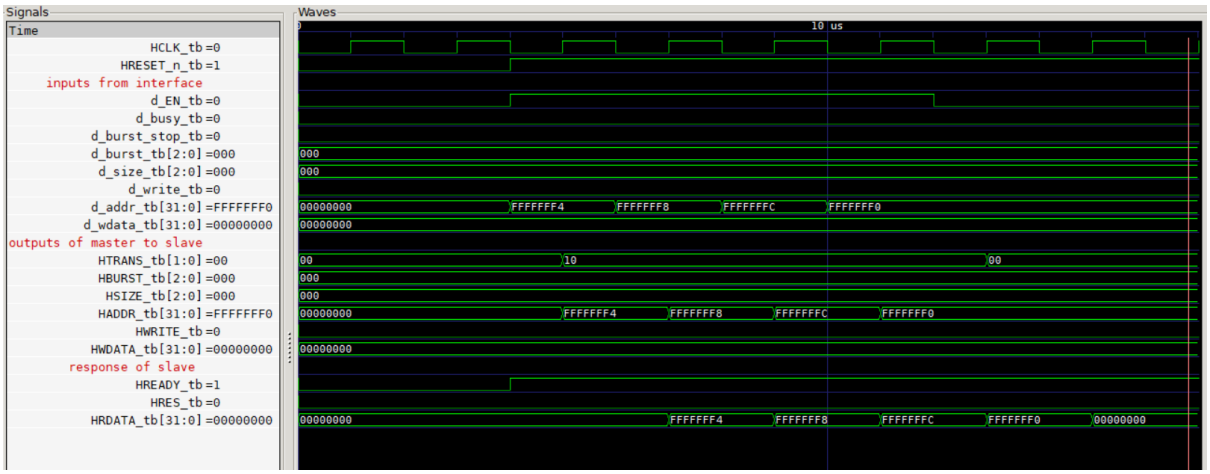

Figure 4: Waveform for 4 consecutive read transactions with no wait cycles

Four Consecutive Writes with 0 Waits

A test was performed by sending 4 consecutive writes transaction to ensure pipelining is taking place. The output waveform for this test is shown in Figure 5 below:

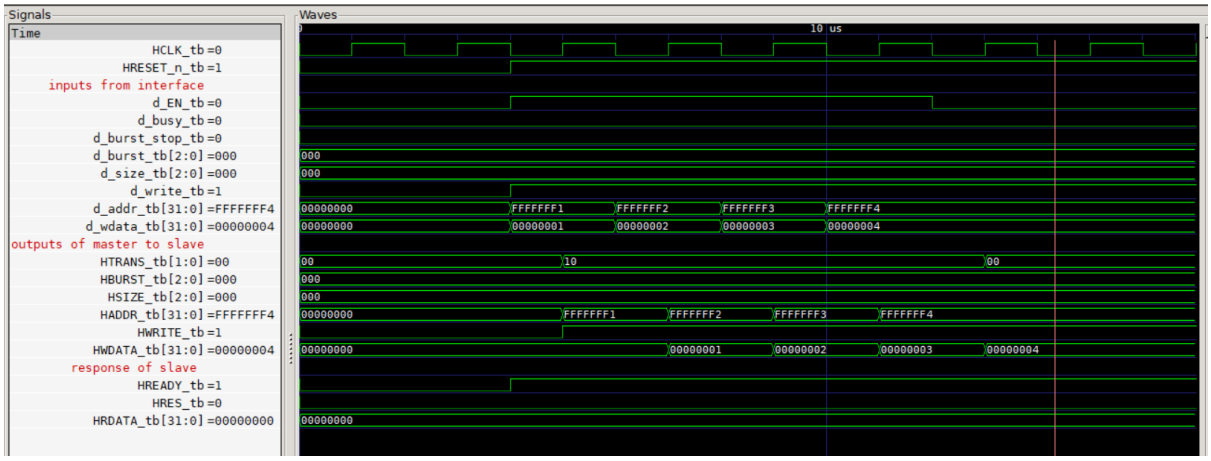

Figure 5: Waveform for 4 consecutive write transactions with no wait cycles

Write then Read with 0 Waits

A test was performed by sending write then read transaction to ensure pipelining works when transactions are of varying types. The output waveform for this test is shown in Figure 6 below:

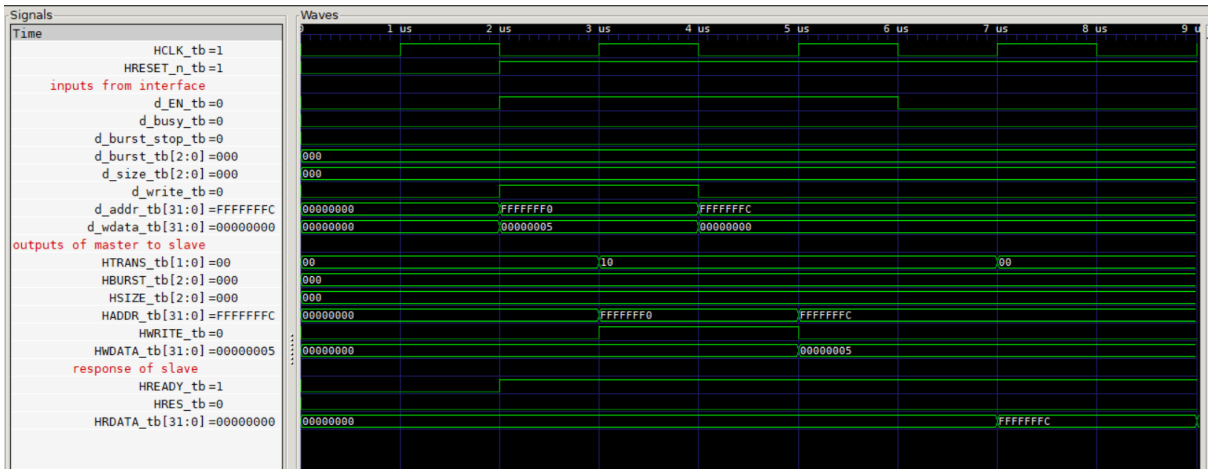

Figure 6: Waveform for write then read with no wait cycles

Write Read with 1 Wait

A test was performed by sending write then read transaction with 1 wait cycle to ensure control signals stall while **HREADY** = 0. The waveform of this test is compared to figure 3.4 in the AHB specifications to ensure our design meets correct timing. The output waveform for this test is shown in Figure 7 below:

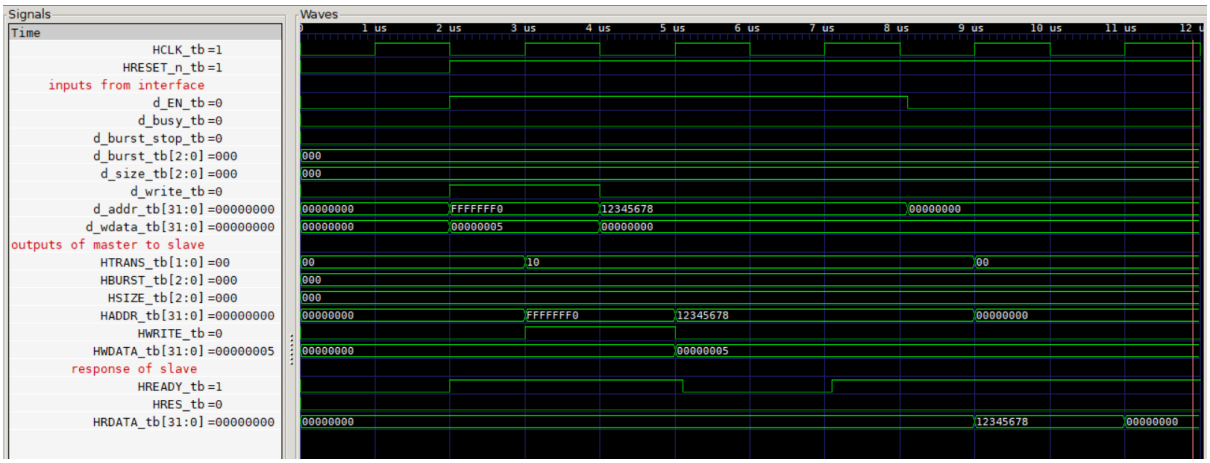

Figure 7: Waveform for Write Read with 1 wait cycle

Figure 3.4 in the AHB specifications is shown below in Figure 8 for comparison. As shown below, both figures match.

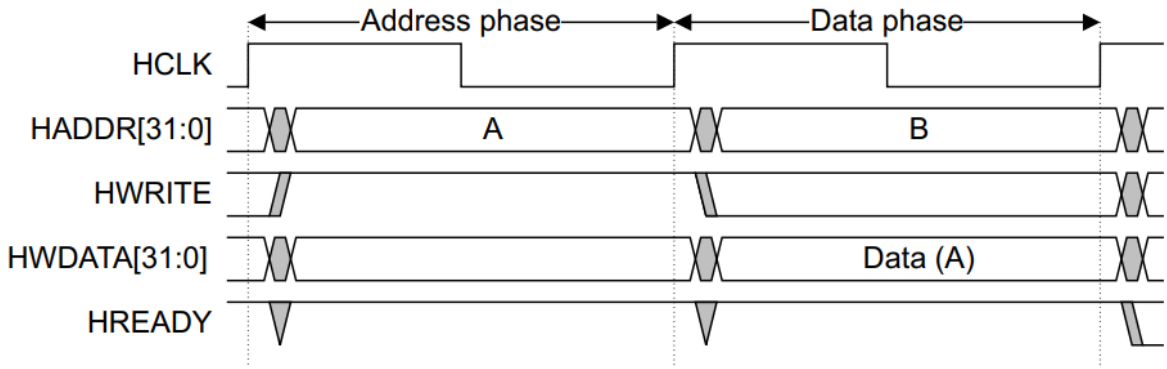

Figure 8: Figure 3.4 in AHB specifications

Read then Write with 0 Waits

A test was performed by sending read then write transaction to ensure pipelining works when transactions are of varying types. The output waveform for this test is shown in Figure 9 below:

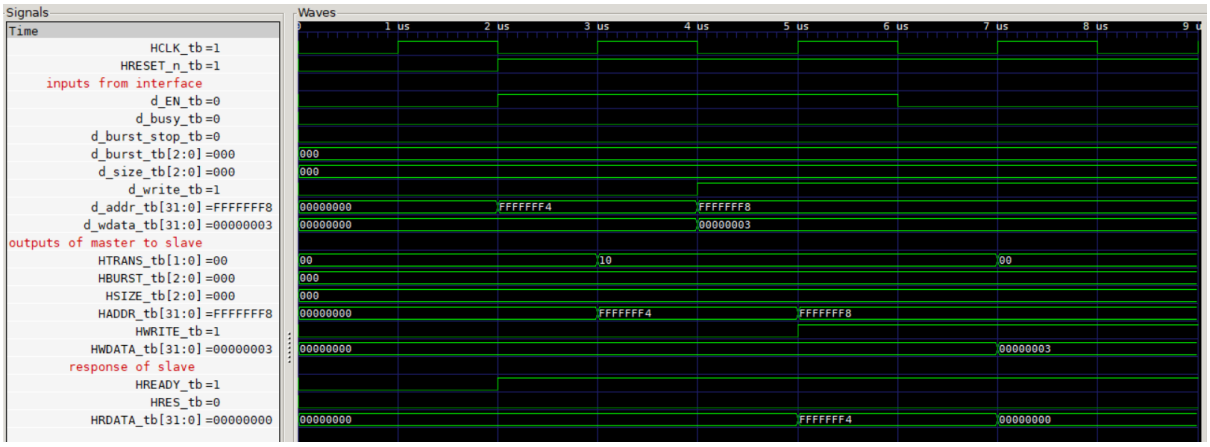

Figure 9: Waveform for read then write with no wait cycles

Write Write with 1 Wait

A test was performed by sending write then write transaction with 1 wait cycle to ensure control signals stall while **HREADY** = 0. The output waveform for this test is shown in Figure 10 below:

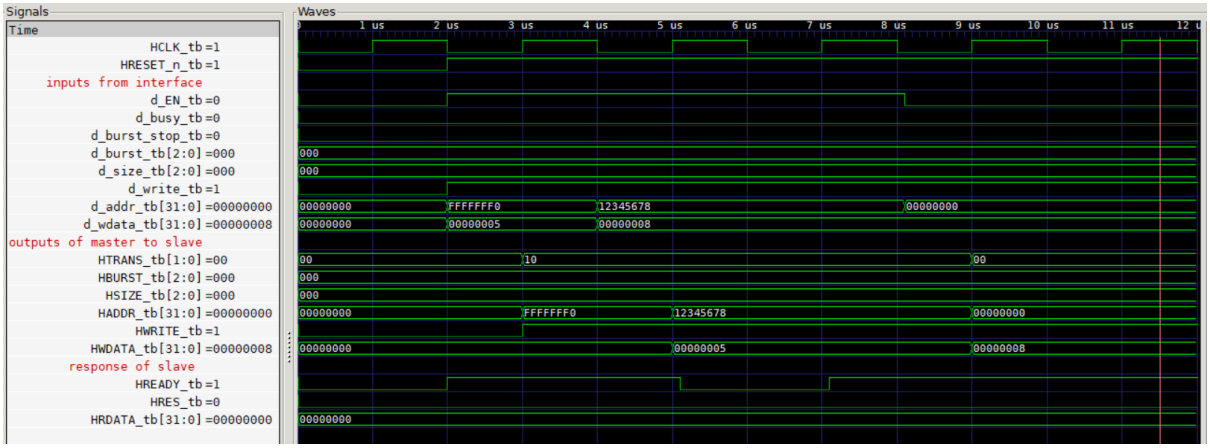

Figure 10: Waveform for Write Write with 1 wait cycle

Read then Write with 2 Waits

A test was performed by sending write then read transaction with 2 wait cycle to ensure control signals stall while **HREADY** = 0. The waveform of this test is compared to figure 3.3 in the AHB specifications to ensure our design meets correct timing. The output waveform for this test is shown in Figure 11 below:

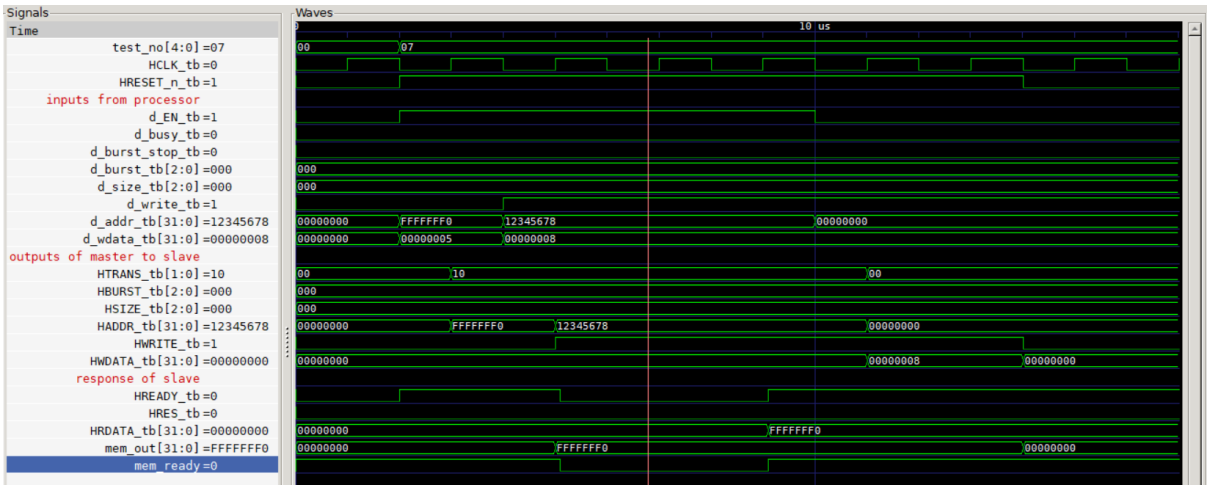

Figure 11: Waveform for read then write with 2 wait cycles

Figure 3.3 in the AHB specifications is shown below in Figure 12 for comparison. As shown below, both figures match.

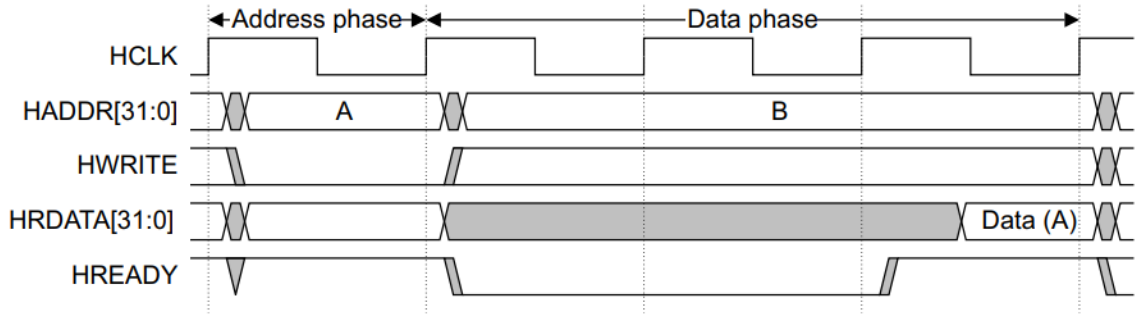

Figure 12: Figure 3.3 in AHB specifications

Multiple Transfers with Waits

A test was performed by sending write read write transaction with wait cycles to ensure control signals stall while **HREADY** = 0. The waveform of this test is compared to figure 3.5 in the AHB specifications to ensure our design meets correct timing. The output waveform for this test is shown in Figure 13 below:

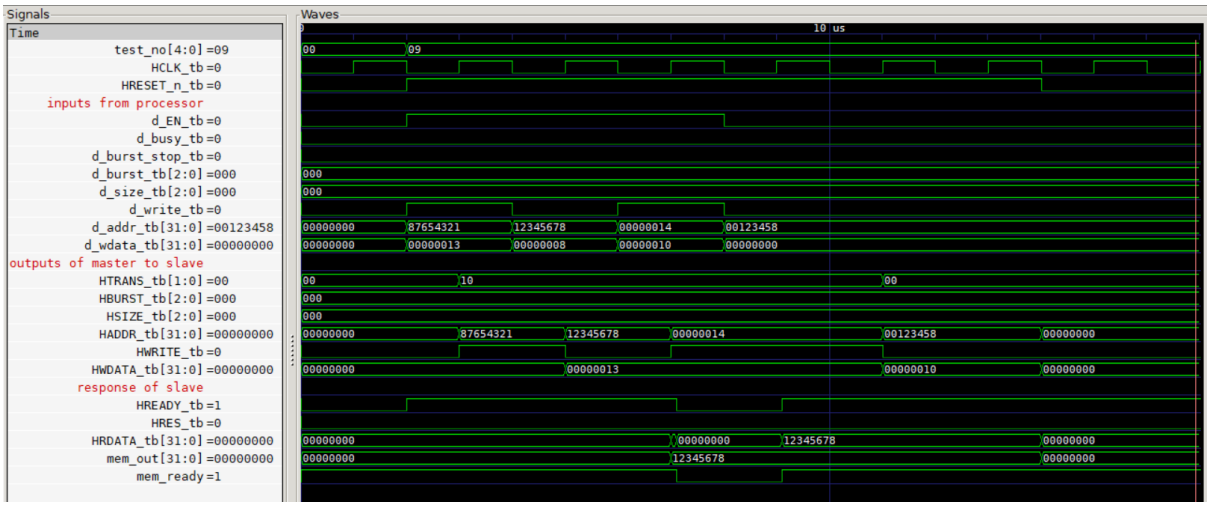

Figure 13: Waveform that mimics figure 3.5 in the specifications

Figure 3.5 in the AHB specifications is shown below in Figure 14 for comparison. As shown below, both figures match.

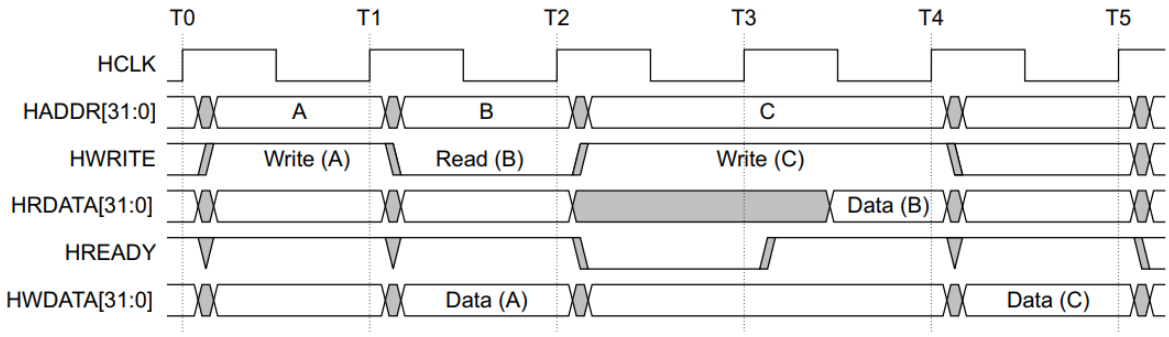

Figure 14: Figure 3.5 in AHB specifications

Busy Transfer with Waits (Fig 3.6)

A test was performed by sending busy transfer with wait cycles to ensure control signals stall while **HREADY** = 0 as well as correct handling of busy transactions. The waveform of this test is compared to figure 3.6 in the AHB specifications to ensure our design meets correct timing. The output waveform for this test is shown in Figure 15 below:

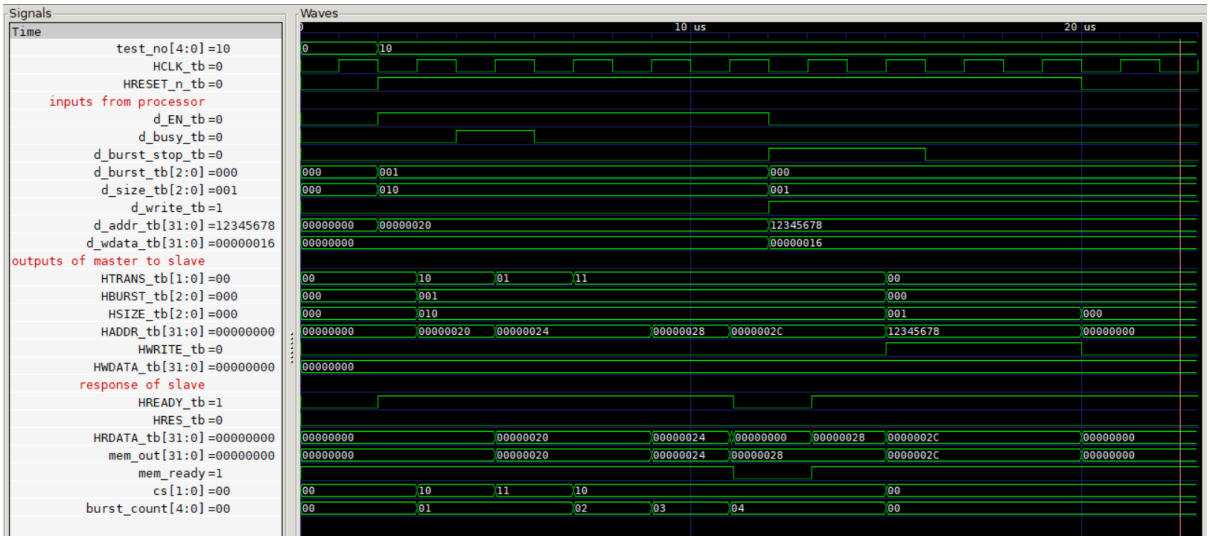

Figure 15: Waveform that mimics figure 3.6 in the specifications

Figure 3.6 in the AHB specifications is shown below in Figure 16 for comparison. As shown below, both figures match.

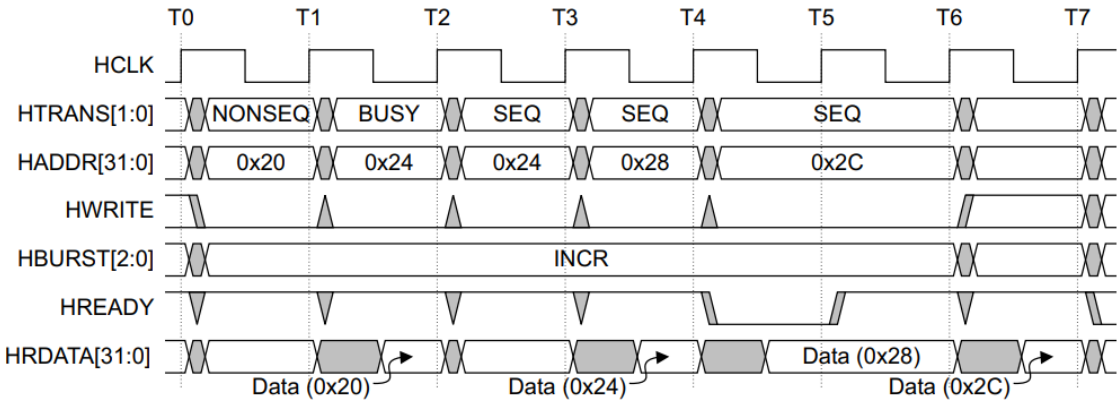

Figure 16: Figure 3.6 in AHB specifications

Bursts of Undefined Length (Fig 3.12)

A test was performed by sending burst transaction of undefined length to ensure burst transfers are sent correctly. The waveform of this test is compared to figure 3.12 in the AHB specifications to ensure our design meets correct timing. The output waveform for this test is shown in Figure 17 below:

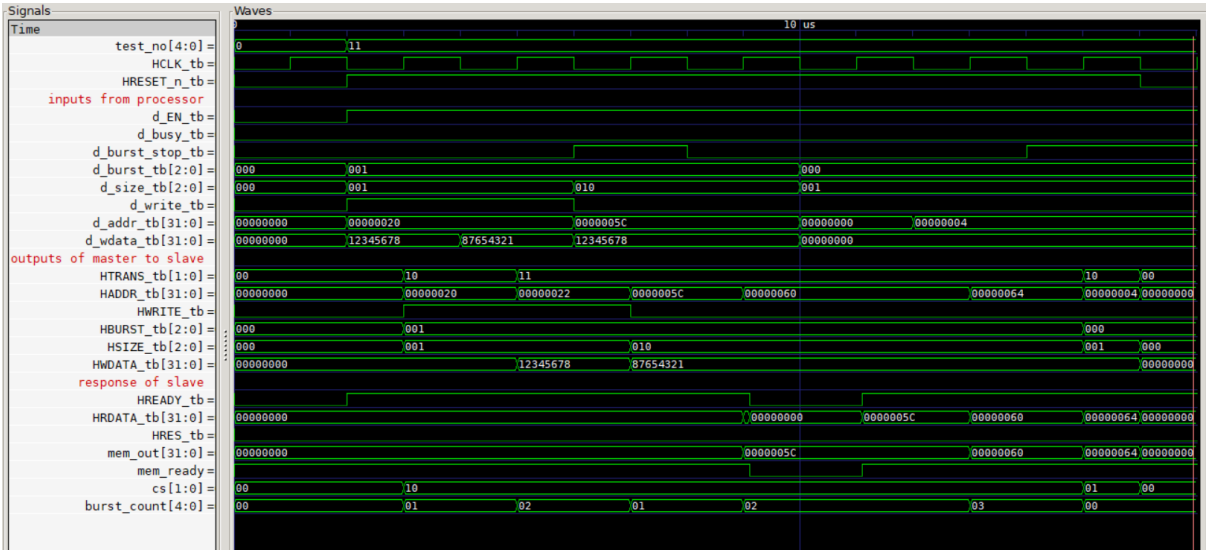

Figure 17: Bursts of undefined length waveform

Figure 3.12 in the AHB specifications is shown below in Figure 18 for comparison. As shown below, both figures match.

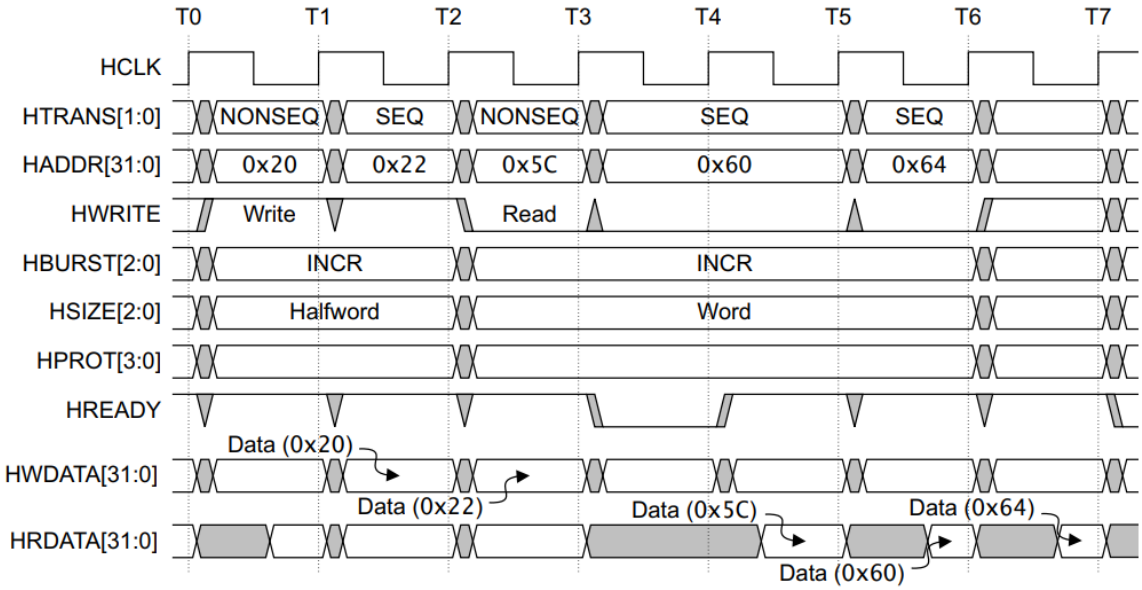

Figure 18: Figure 3.12 in AHB specifications

IDLE to NONSEQ Transfer During a Wait (Fig 3.13)

A test was performed by sending idle followed by NONSEQ transfers to the master while it is in wait state. The waveform of this test is compared to figure 3.13 in the AHB specifications to ensure our design does support transition from IDLE to non sequential transfer. The output waveform for this test is shown in Figure 19 below:

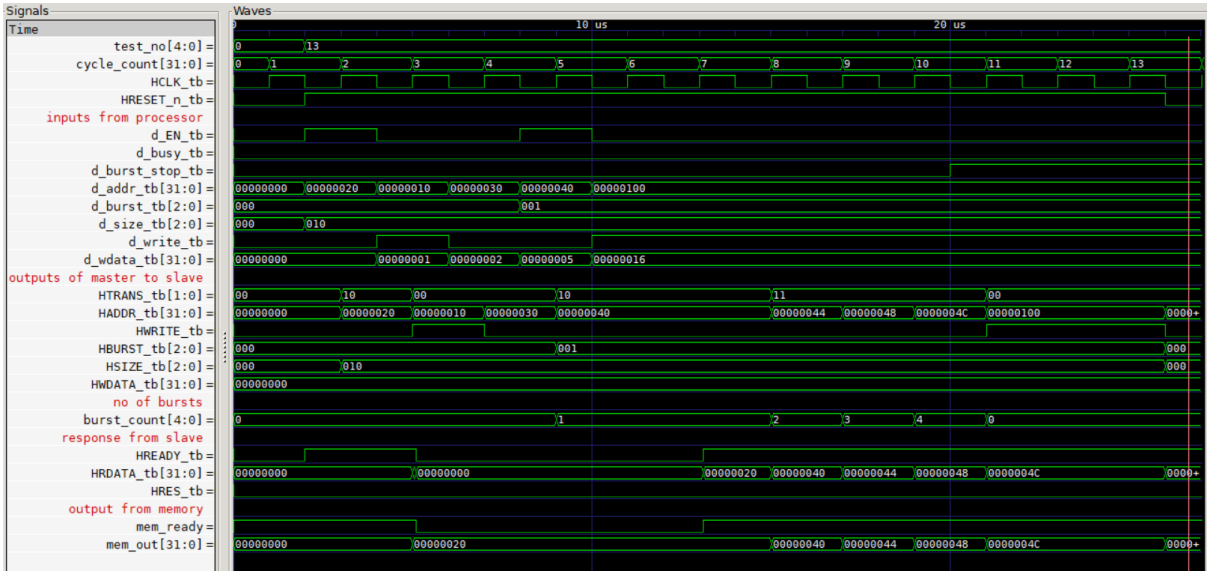

Figure 19: IDLE to NONSEQ transfer during a wait state

Figure 3.13 in the AHB specifications is shown below in Figure 20 for comparison. As shown below, both figures match.

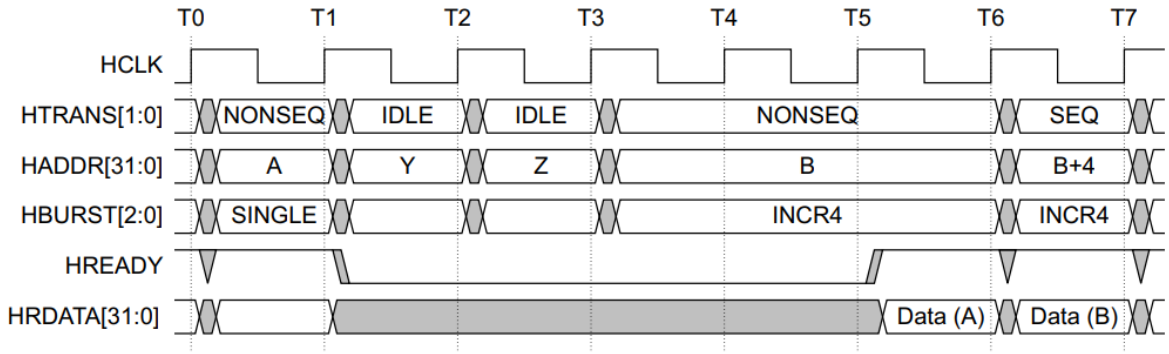

Figure 20: Figure 3.13 in AHB specifications

BUSY to SEQ Transfer During a Wait (Fig 3.14)

A test was performed by sending de asserting busy signal mid wait cycle to test if the transfer will change from busy to sequential as per the specifications. The waveform of this test is compared to figure 3.14 in the AHB specifications to ensure our design does supports resuming burst transfer after a busy during wait. The output waveform for this test is shown in Figure 21 below:

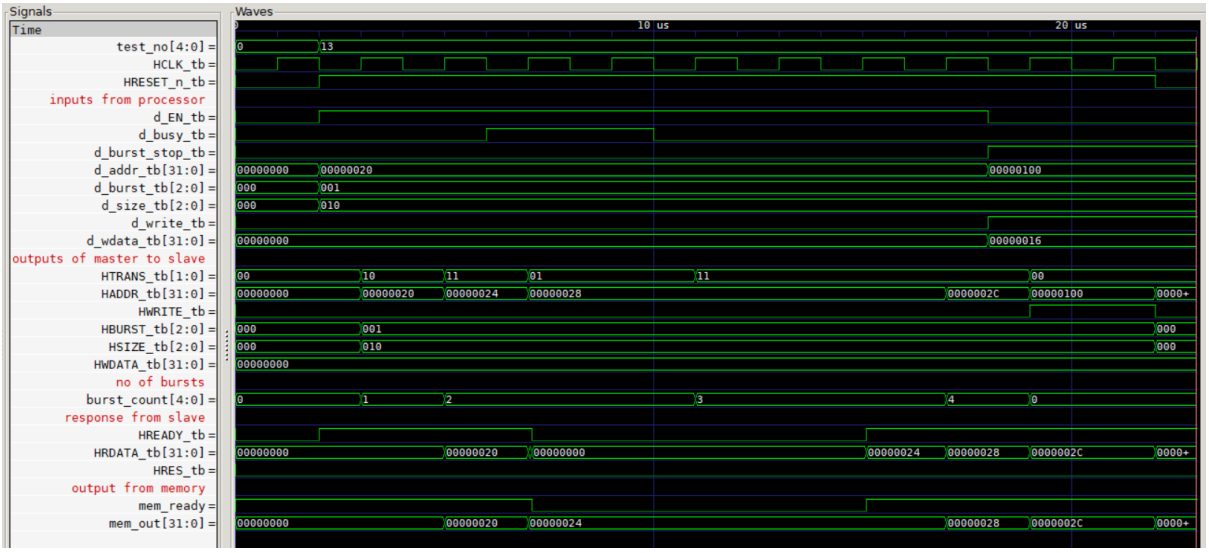

Figure 21: BUSY to SEQ transfer during a wait state

Figure 3.14 in the AHB specifications is shown below in Figure 22 for comparison. As shown below, both figures match.

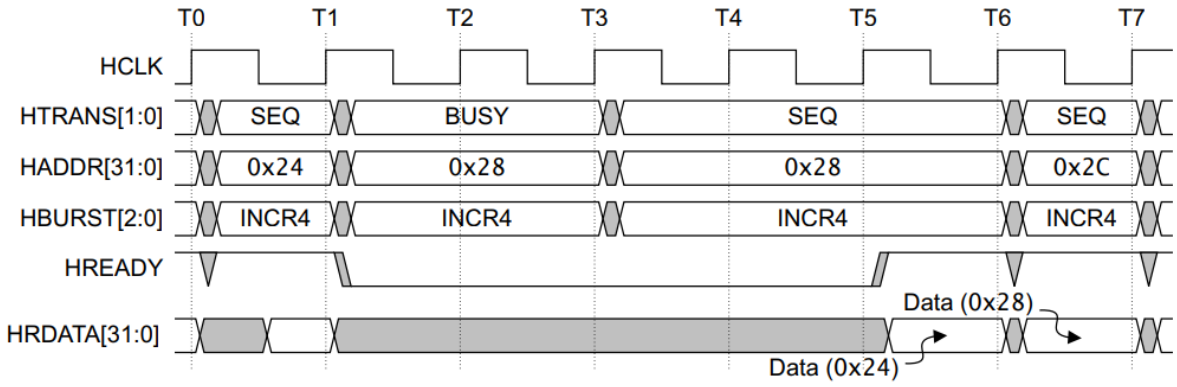

Figure 22: Figure 3.14 in AHB specifications

BUSY to NONSEQ Transfer During a Wait (Fig 3.15)

A test was performed by setting burst stop signal HIGH mid wait cycle to test if the transfer will change from busy to non sequential as per the specifications. The waveform of this test is compared to figure 3.15 in the AHB specifications to ensure our design does supports terminating burst transfer of undefined length after a busy during wait. The output waveform for this test is shown in Figure 23 below:

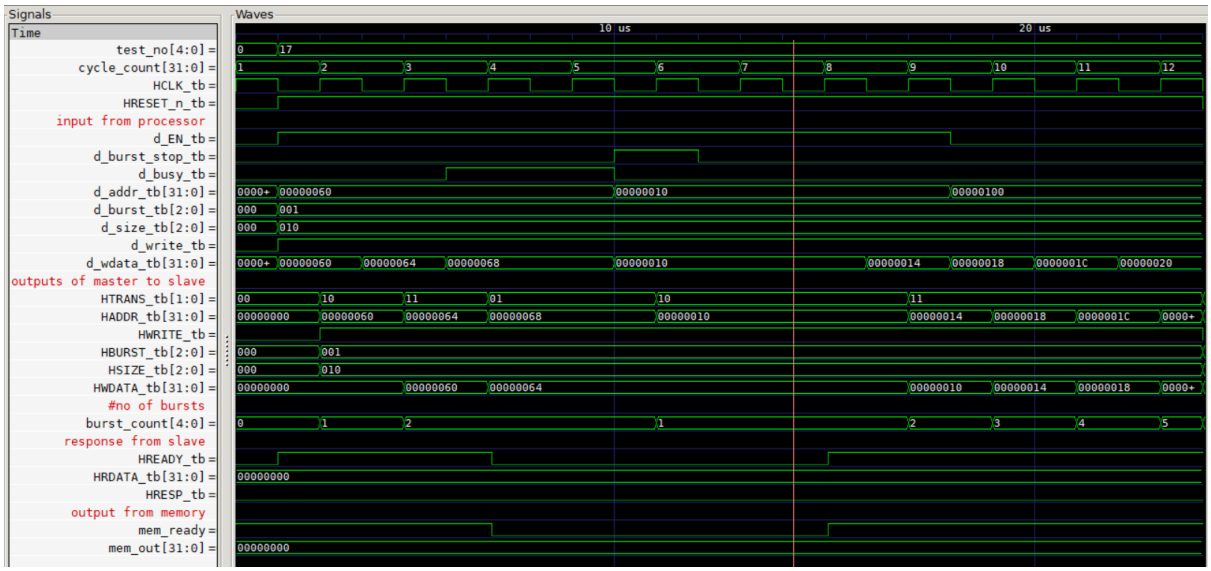

Figure 23: BUSY to NONSEQ transfer during a wait state

Figure 3.15 in the AHB specifications is shown below in Figure 24 for comparison. As shown below, both figures match.

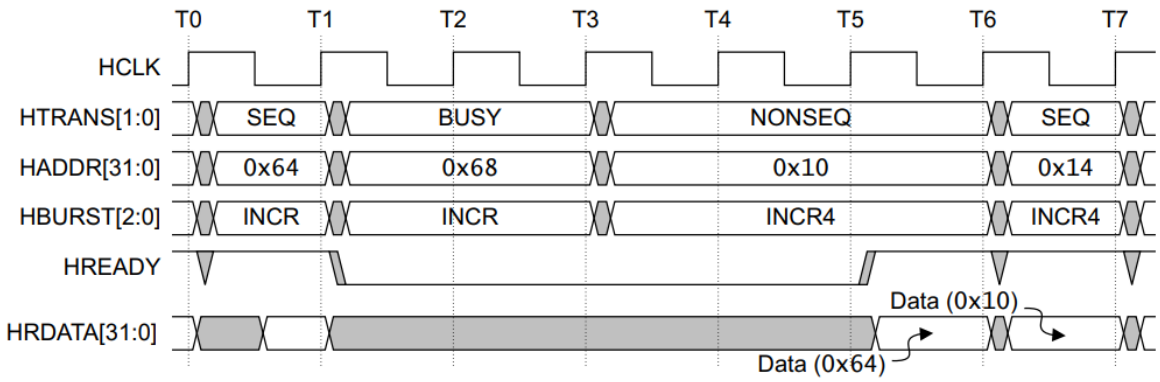

Figure 24: Figure 3.15 in AHB specifications

Error Handling (Fig 3.17 & Fig 5.1)

This test is of a burst transfer with got an error response by the slave then the transfer got aborted. It is compared to that of figure 3.17 in the AHB specifications to ensure our implementation handles error response appropriately. The output waveform for this test is shown in Figure 25 below:

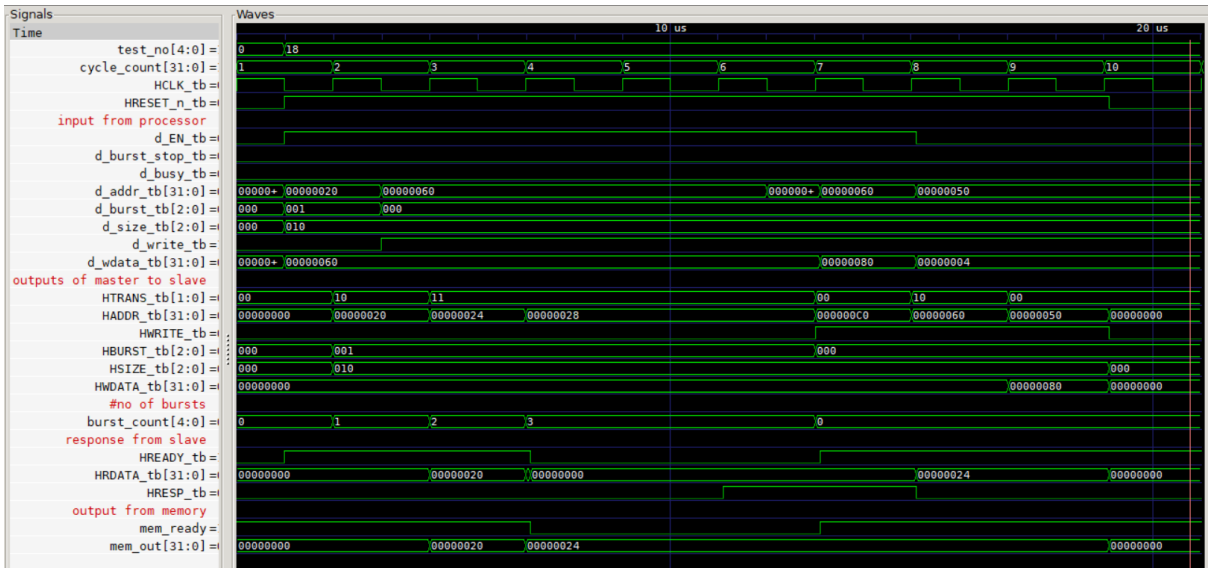

Figure 25: Burst transfer with error response

Figure 3.17 in the AHB specifications is shown below in Figure 26 for comparison. As shown below, both figures match.

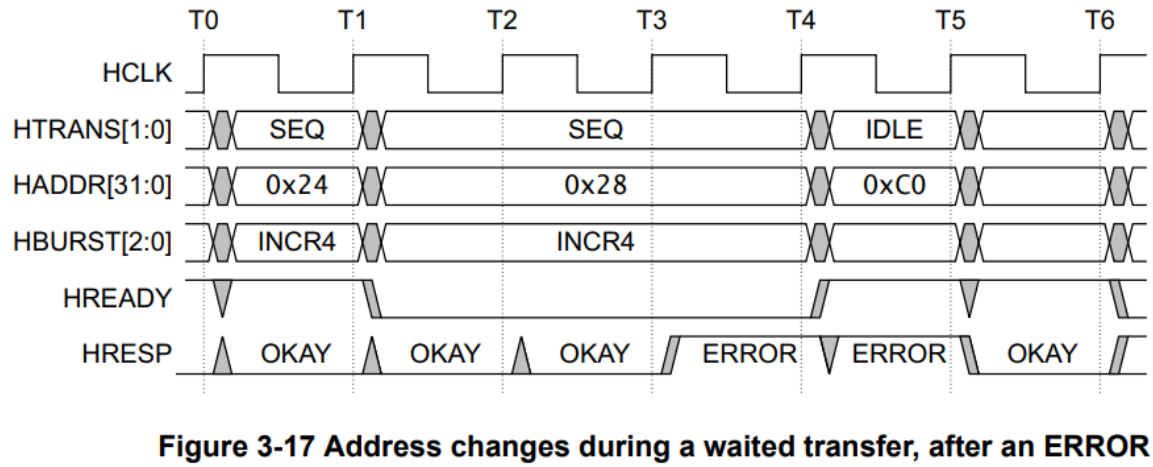

Figure 26: Figure 3.17 in AHB specifications

Anther test of error signal was done with non sequential transfer which had the same sequence like figure 5.1 in AHB specifications. The output waveform for this test is shown in Figure 27 below:

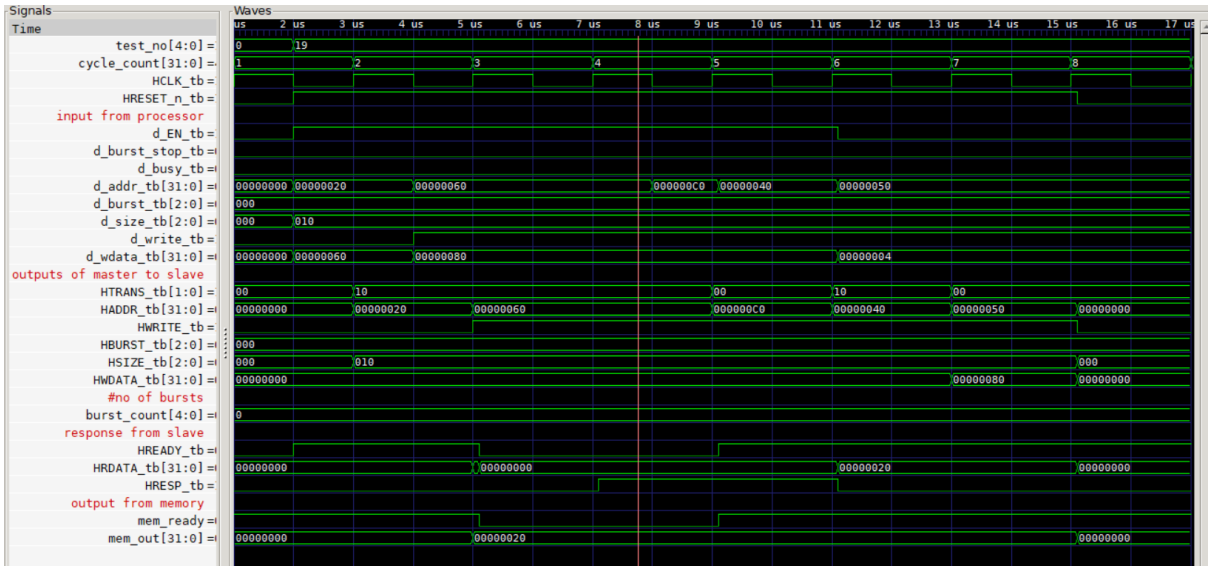

Figure 27: Non sequential transfer that lead to ERROR

Figure 5.1 in the AHB specifications is shown below in Figure 28 for comparison. As shown below, both figures match.

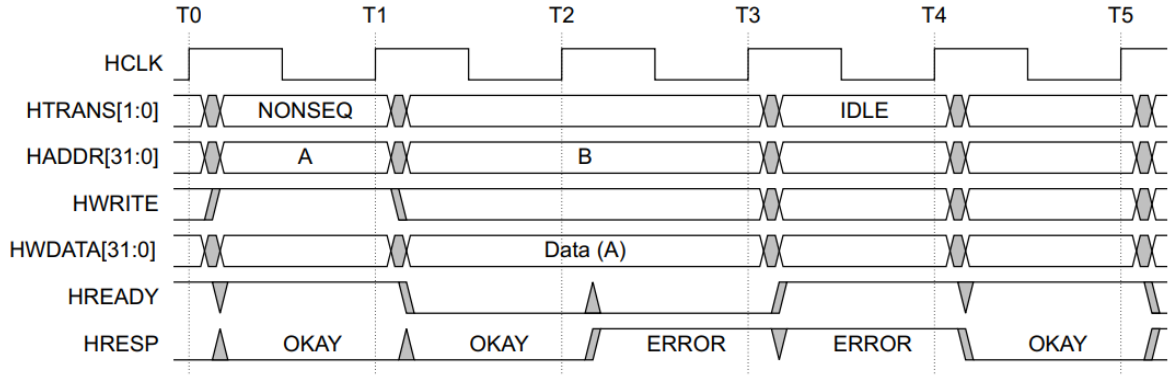

Figure 28: Figure 5.1 in AHB specifications

INC4 Transfer (Fig 3.9)

This test is of a INC4 burst transfer with one wait cycle. It is compared to that of figure 3.9 in the AHB specifications to ensure our implementation carries out burst of fixed length correctly. The output waveform for this test is shown in Figure 29 below:

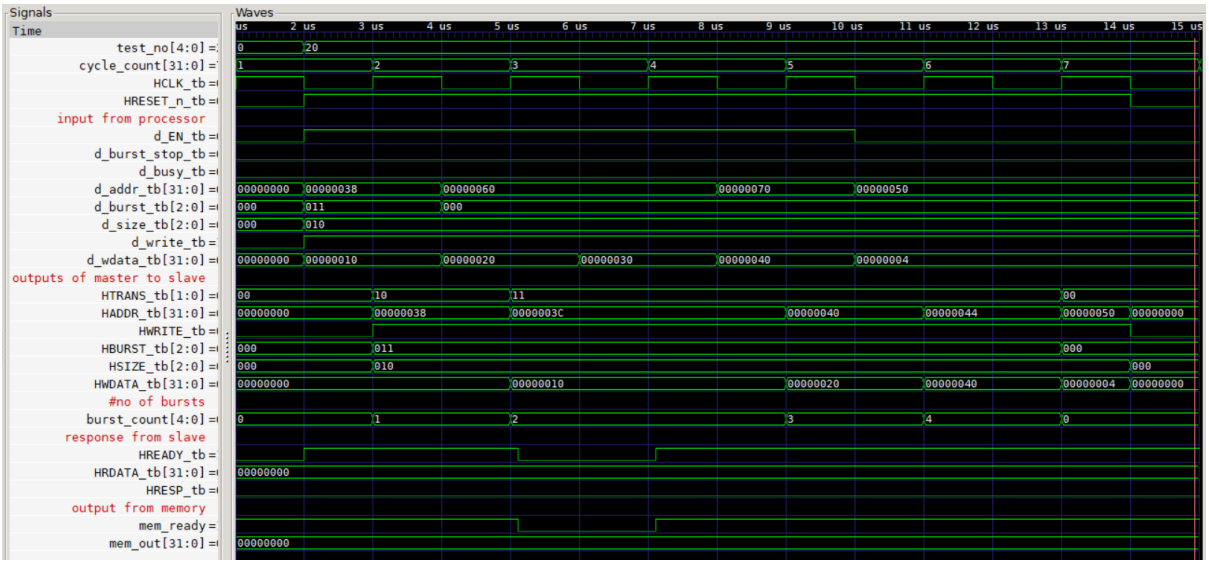

Figure 29: INC4 Transfer

Figure 3.9 in the AHB specifications is shown below in Figure 30 for comparison. As shown below, both figures match.

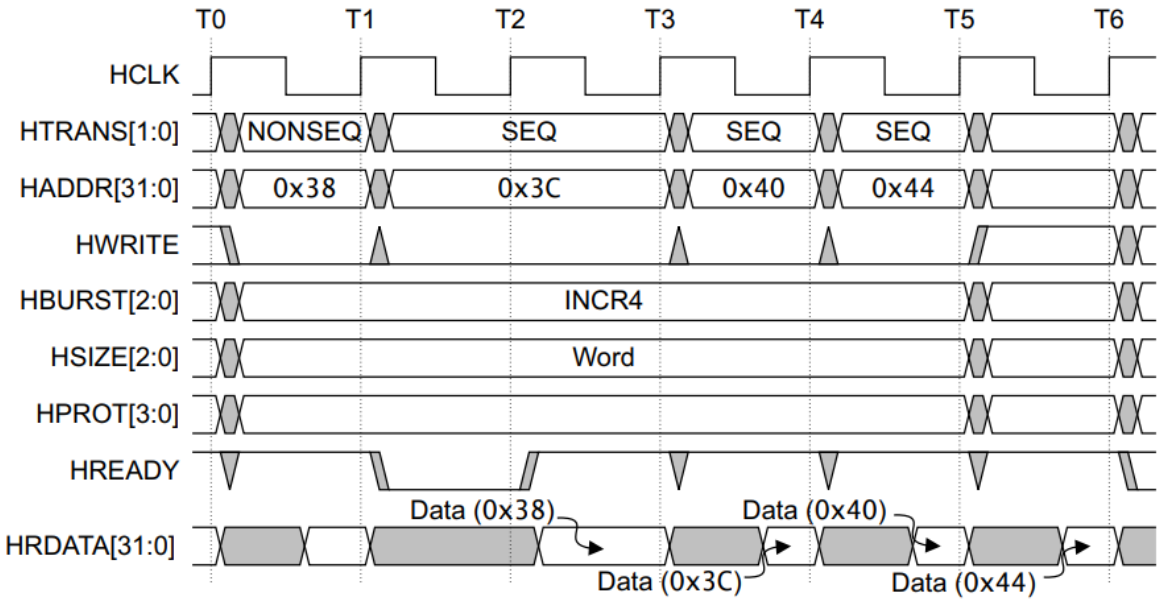

Figure 30: Figure 3.9 in AHB specifications

INC8 Transfer (Fig 3.11)

This test is of a INC8 burst transfer with one wait cycle. It is compared to that of figure 3.8 in the AHB specifications to ensure our implementation carries out burst of fixed length correctly. The output waveform for this test is shown in Figure 31 below:

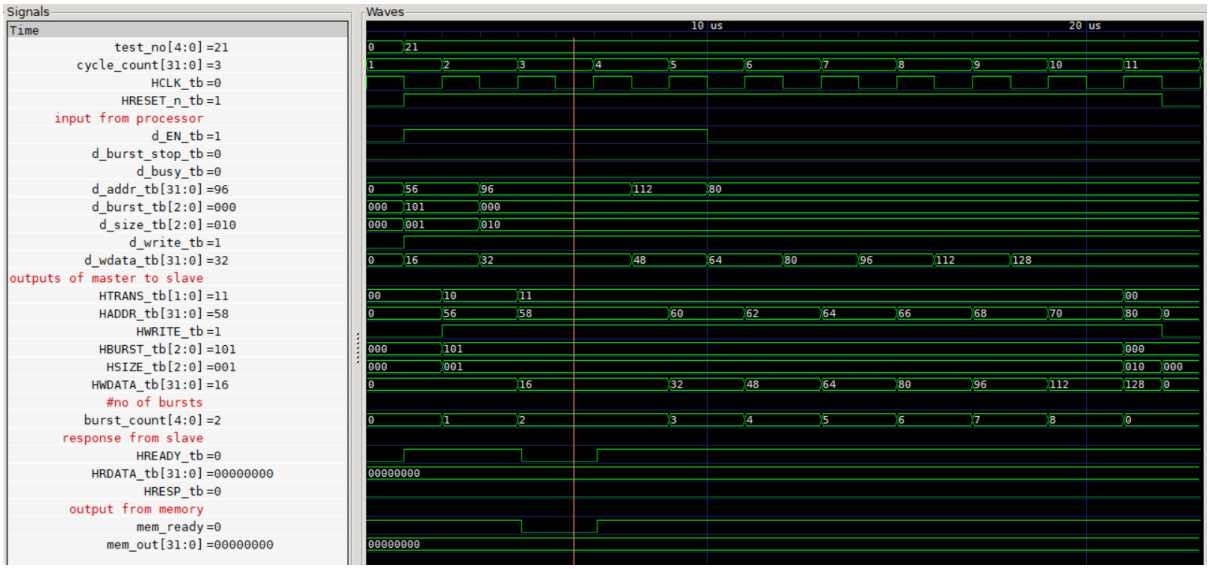

Figure 31: INC8 Transfer

Figure 3.11 in the AHB specifications is shown below in Figure 32 for comparison. As shown below, both figures match.

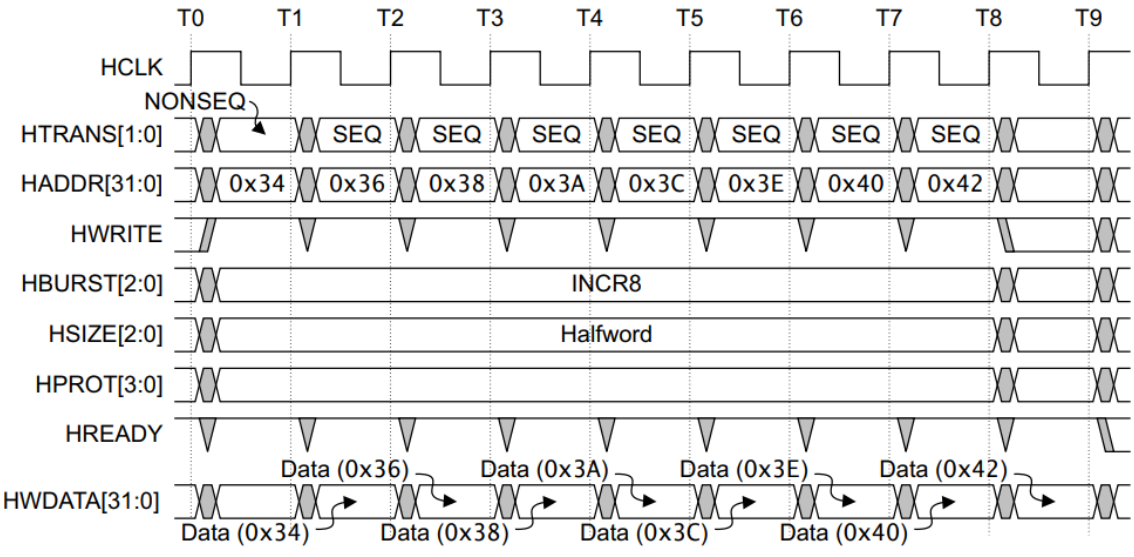

Figure 32: Figure 3.11 in AHB specifications

INC16 Transfer

This test is of a INC16 burst transfer with one wait cycle. The output waveform for this test is shown in Figure 33 below:

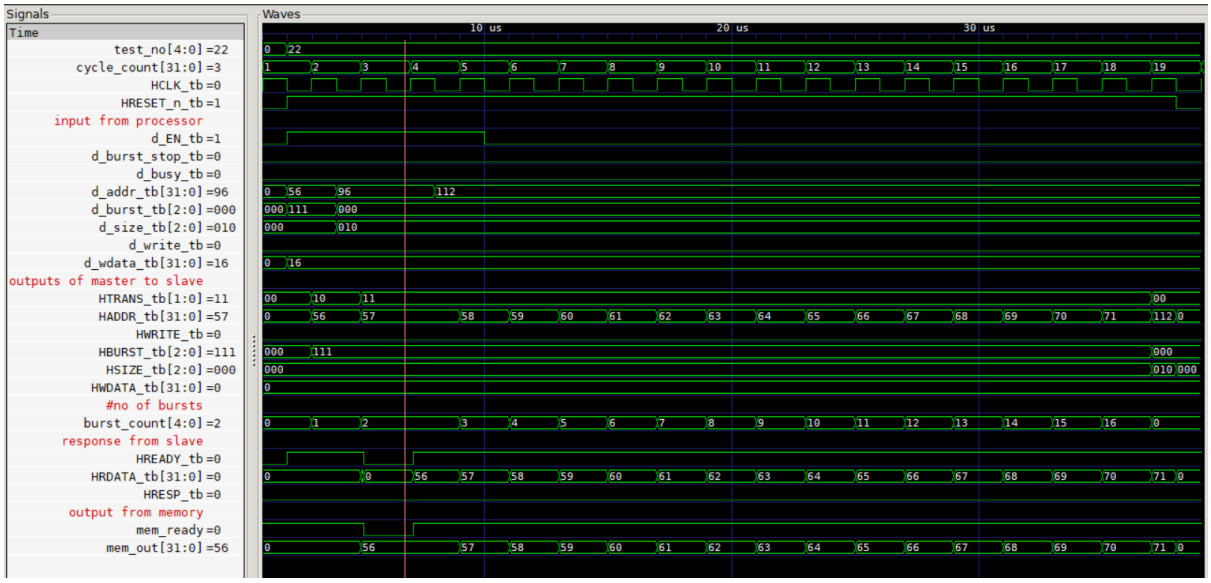

Figure 33: INC16 Transfer

Read then Read with 2 Waits

A test was performed by sending 2 Read transaction with 2 wait cycle to ensure control signals stall while **HREADY** = 0. The output waveform for this test is shown in Figure 34 below:

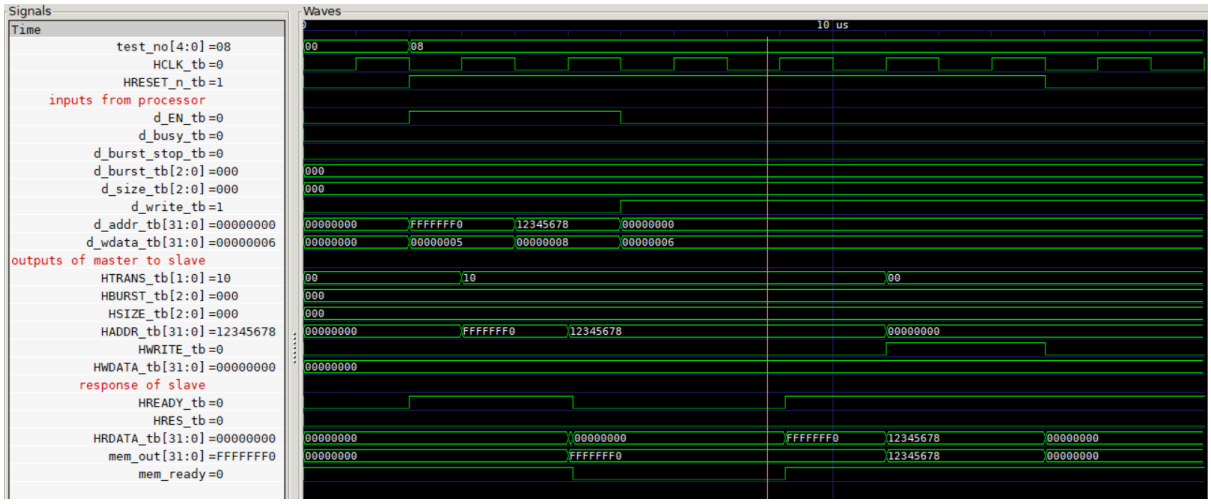

Figure 34: Waveform for read then read with 2 wait cycles

References

[1] AMBA®3 AHB-Lite Protocol Specification, https://www.eecs.umich.edu/courses/eecs373/readings/ARM_IHI0033A_AMBA_AHB-Lite_SPEC.pdf