17/12/2015

Onchip Protocols	Peripheral Protocols
1. AMBA AHB	1. UART
2. AMBA APB	2. I2C
3. AMBA AXI	3. SPI

What is AMBA

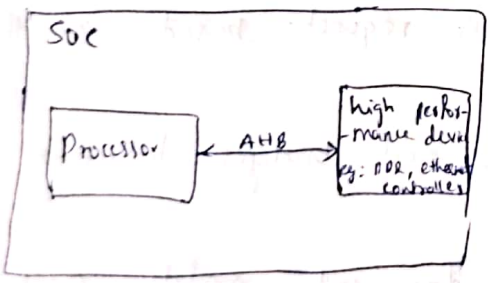
AMBA : "Advanced Microcontroller Bus Architecture" is a set of specifications developed by 'ARM'

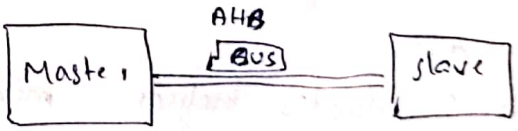
→ to standardize the communication between different functional blocks of an SOC

→ AMBA consists of multiple protocols including

i) AHB (Advanced High-Performance Bus) : high speed, high-bandwidth
ii) APB (Advanced Peripheral Bus) : low power, low-bandwidth
iii) AXI (Advanced extensible Interface) : high performance, low-latency, high bandwidth

* AXI is advanced protocol than AHB, APB





AHB Protocol

The AHB protocol defines the interaction between three main components

i) Manager (Master) ii) Slave (Subordinates) iii) Interconnects

* No component (Master or slave) permanently all as Master or slave they varies eg slave can become Master & Master can become slave

i) Managers (Masters):

Initiate transaction by issuing read and write commands

ii) Subordinates (Slaves): Responds to Manager requests and provide or store data.

iii) Interconnects: Handle data routing and arbitration between Managers and subordinates.

* AHB protocol includes several essential features

↳ i) Single clock-edge operation: protocol operates synchronously on clock triggering ensuring predictable timing

↳ ii) Burst transfers: AHB support single & burst transfer

AHB System components

1) AHB Manager (Master)

→ initiates all transactions on the bus

→ Provides the address, control signals, and data during read & write operations

→ only one Manager can control the bus at a time to avoid conflicts

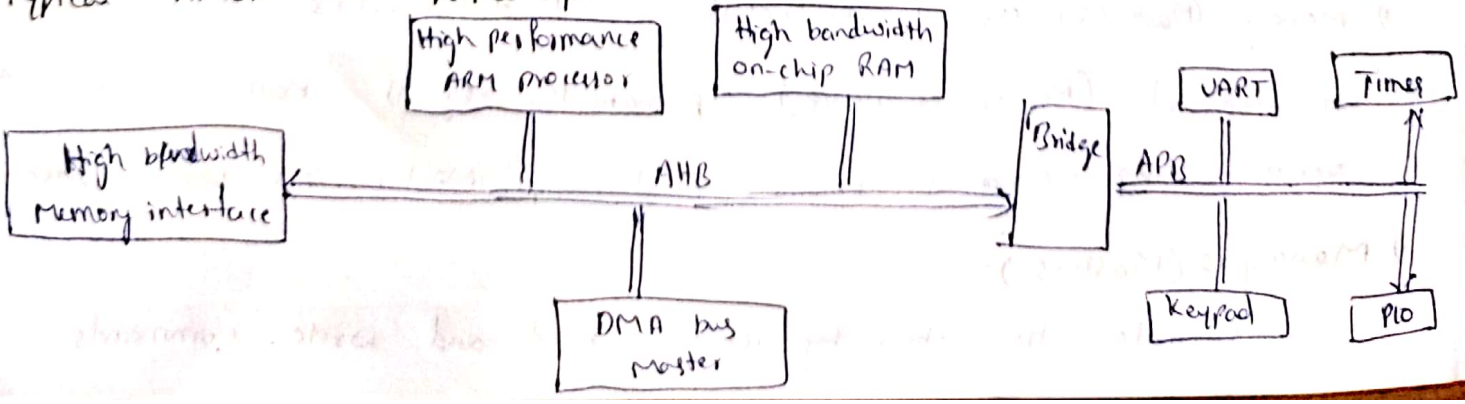
2) AHB Subordinate (Slave)

→ the Subordinate responds to the Manager's requests based on the provided address.

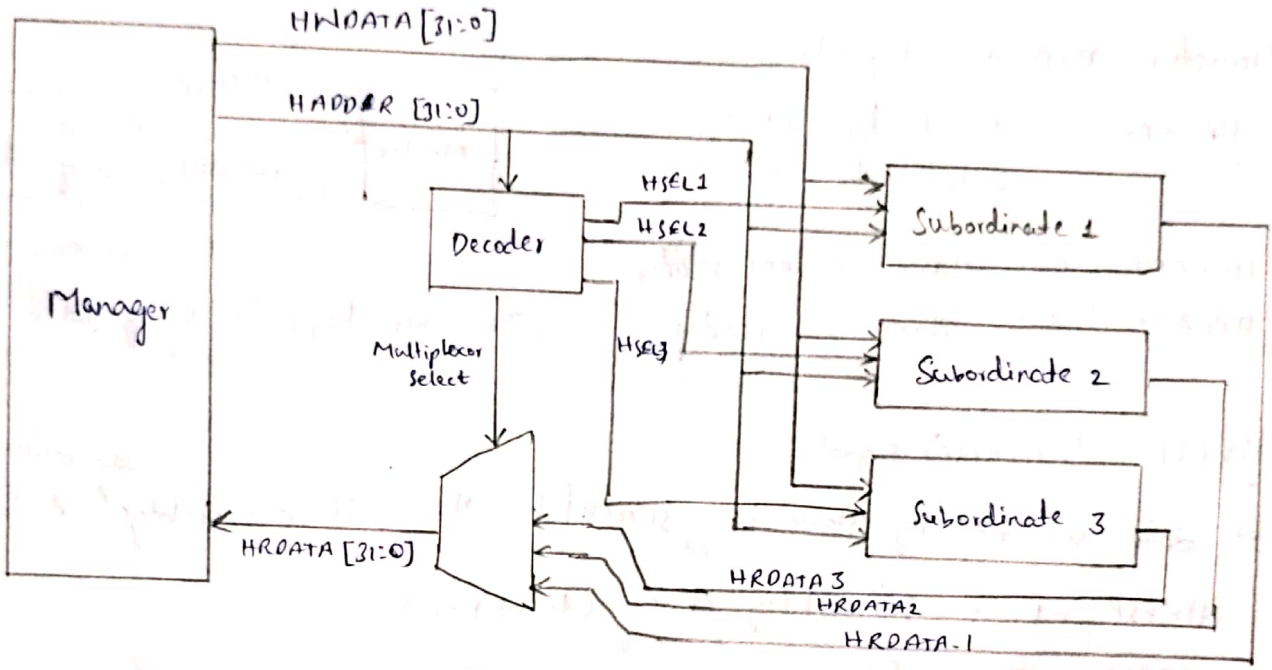
→ Subordinates include memories, peripherals, and bridges to other buses like APB.

→ If the address sent by the Manager does not match any Subordinate, the response is considered invalid.

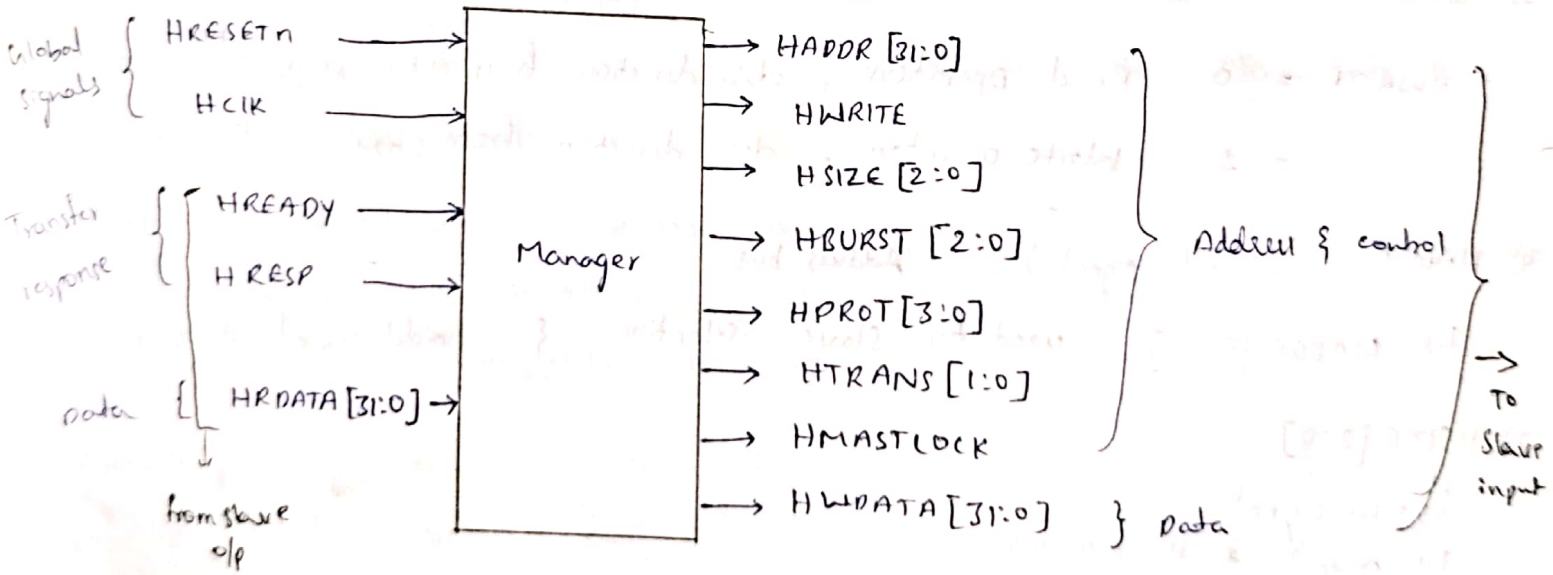
Typical AMBA AHB based system



AHB block diagram



Manager



Note

In AHB protocol every signal is represented by 'H' eg HREADY HWRITE etc

In APB protocol → 'P' eg: PREADY PWRITE etc

This is made to distinguish the signals when data is transfering between two different protocols.

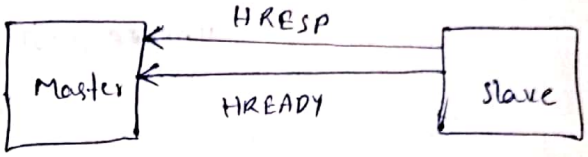
1) Global signals : connected to all components

i) HRESETn : Active low 0 ⇒ reset ; 1 → NO reset

ii) HCLK : clock —

2) Transfer response signals

① HREADY : Input for Master
Output for slave



HREADY = 0 : slave is not ready
HREADY = 1 : slave is ready Transition happens only when slave is ready

② HRESP (Response) signal

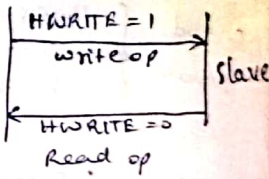
If data received by slave is successful then it acknowledge the master by HRESP

HRESP = 1 ; Successfully done (transition)
HRESP = 0 : ERROR

3) HWRITE decide weather read or write operation

HWRITE = 0 : Read operation , data direction from M to S Master

= 1 : Write operation , data direction from S to M



4) HADDR (32-bit signal) Address bus

↳ HADDR [31:0] used for slave selection & address of data packets

5) HSIZE [2:0]

↳ 3 bit signal
↳ output to the master

HSIZE 000 ⋮ 111 decides the length of the packet that is transferred

6) HBURST

3-bit signal

→ It tells in a transfer how many packets the Master is going to send
→ each packet size is known by HSIZE

HBURST : defines the burst mode of an AHB transaction, specifying how many data transfers occur in a single burst.

→ Determine whether the transfer is single, incrementing or wrapping burst

7) HTRANS 2 bit signal

0 0 → Idle
0 1 → Busy
1 0 → Non sequential
1 1 → sequential

8) HRDATA & HWDATA

Read data — HWRITE = 0

Write data — HWRITE = 1

→ each 32 bit size data bus

L-2 AHB WRITE & READ TRANSFER

A transfer consists of two phases

i) Address phase :- Lasts for a single HCLK cycle unless it is extended by previous bus transfer.

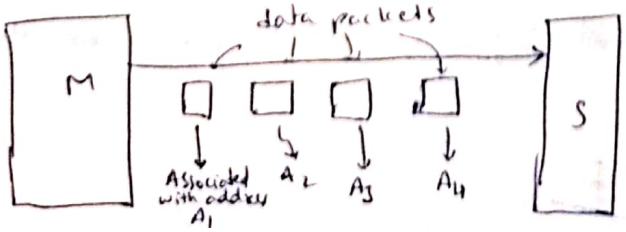
ii) Data phase :- Might require several HCLK cycles

→ use the HREADY signal to control the number of clock cycles required to complete the transfer.

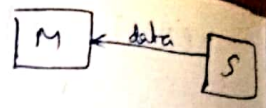
Address phase : All the control signals should be valid & valid address of 1st packet is provided

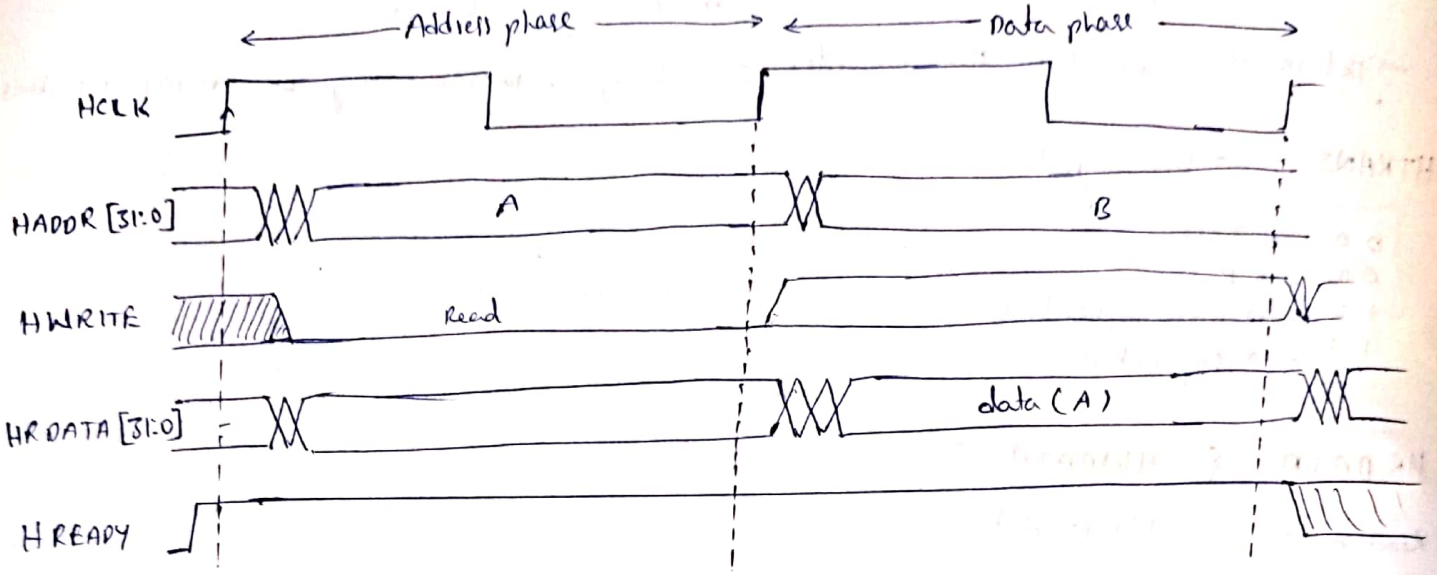
Data phase : Master read / write data

* Every data (packet) which we are going to write / read is associated with an "Address"

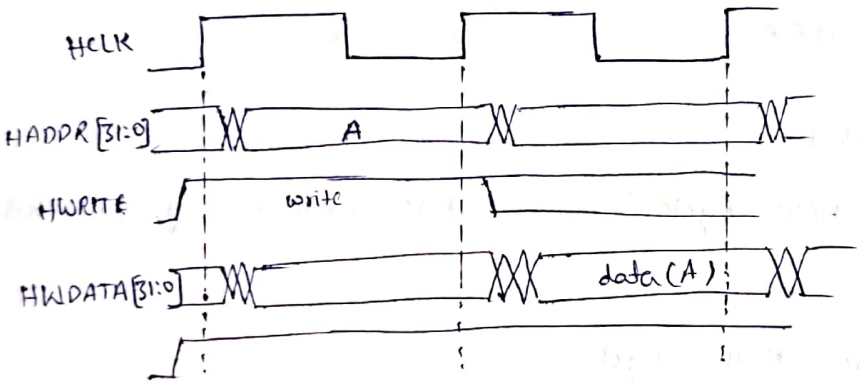


Read Transfer Operation (No wait states)





Write transfer (No wait state)

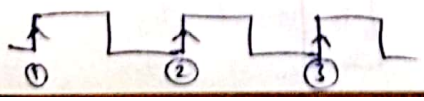


In a simple transfer with no wait state

① The Manager drives the address and control signals onto the bus after the rising edge of HCLK.

② the subordinate then samples the address and control information on the next rising edge of HCLK

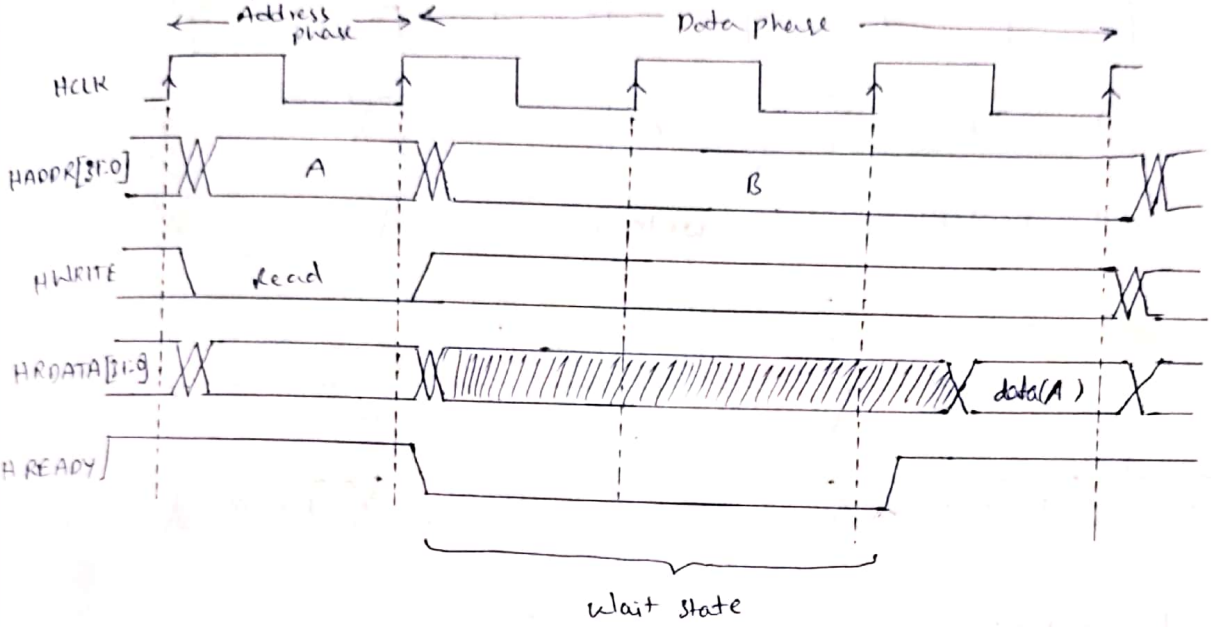
③ After the subordinate has sampled the address and control it can start to drive the appropriate HREADYOUT response. This response is sampled by the Manager on the third rising edge of HCLK



L-3

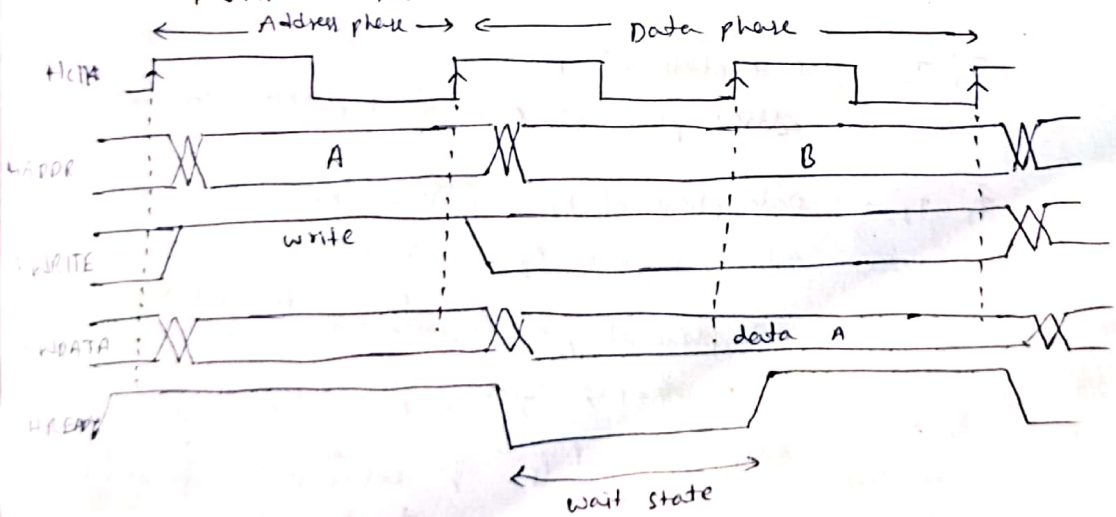
Read Transfer with two wait states

HREADY = 0 ⇒ wait state



If HREADY = 0 then master is not going to change the phase from Data phase to Address phase until and unless HREADY = 1

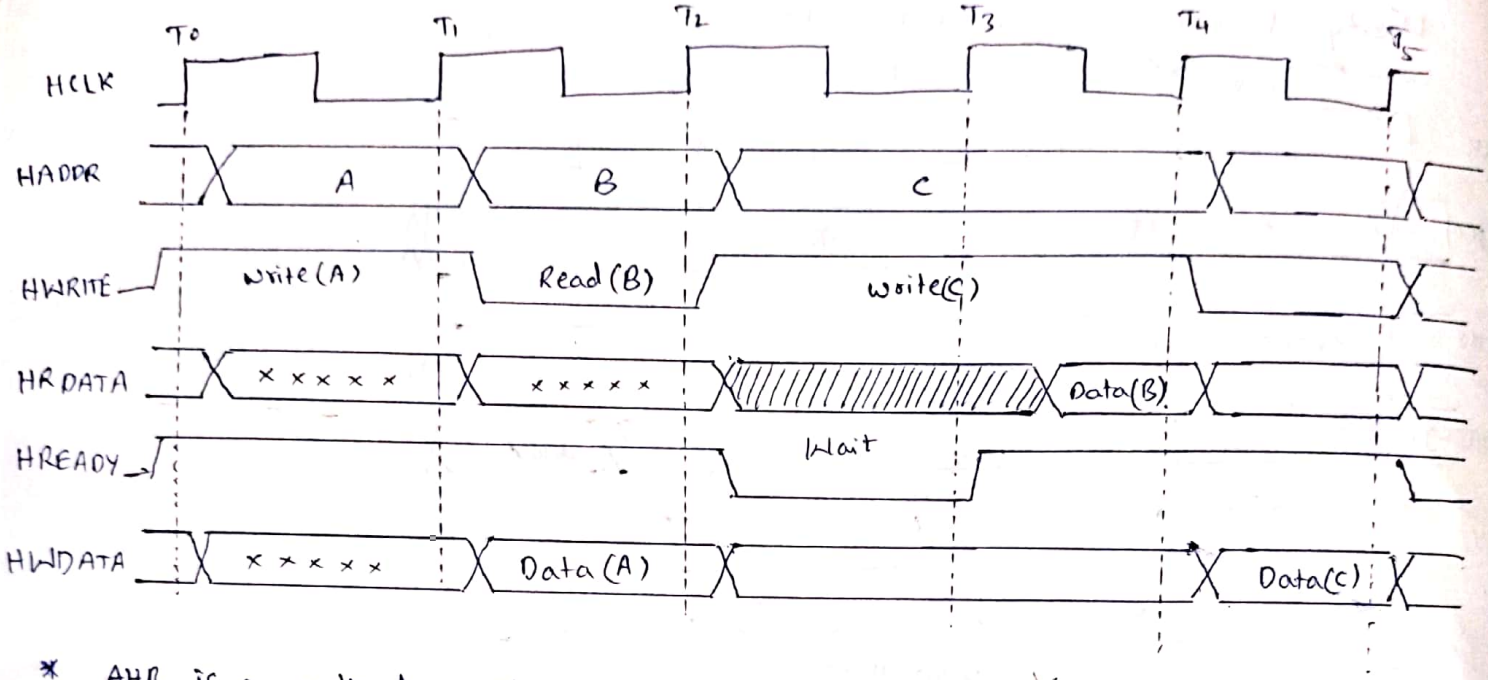
Write Transfer with one wait state



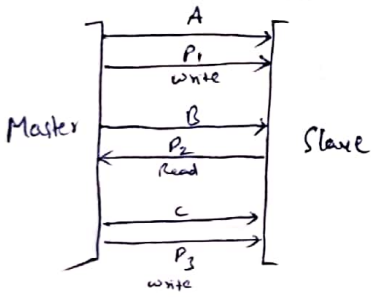
* In write operations Master hold the data stable throughout the extended cycles (wait state)
* For Read transfer Slaves does not have to provide valid data until the transfer is about to complete

L-4

Multiple Transfers



* AHB is a piplined protocol



* Consider 3 packets transfer P_1, P_2, P_3 with addresses A, B, C correspondingly.

$T_0 - T_1$: Address phase of P_1 ; address 'A' transfered from Master to slave

$T_1 - T_2$: Data phase of P_1
address phase of P_2
} Piplined concept

$T_2 - T_3$: Data phase of P_2
Address phase of P_3
} Here HREADY = 0 wait state so not getting data (B) here so it is extended to next cycle

$T_3 - T_4$: Data phase of P_2
HREADY = 1 ; getting packet P_2 in HRDATA bus
Address phase of P_3 ; extended from previous cycle due to wait state

- 'xxxx' → random data - not required

L5 HTRANS Signal

HTRANS [1:0] ; 2 bit signal , four combinations

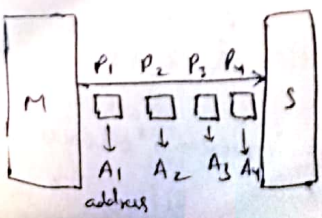
1) HTRANS = 00 : IDLE
   → Indicates no data transfer required
   → A manager uses an IDLE transfer when it does not want to perform data transfer
   → Subordinates must always provide a zero wait state OKAY response to IDLE transfers ~~and the~~
   → The subordinates must ~~be~~ ignore transfer

2) 01 : BUSY
   → Enables master to insert idle cycles in the middle of a burst.
   → This transfer type indicates that the manager is continuing with a burst but the next transfer cannot take place immediately.

3) 10 : Non SEQ
   → Indicates single transfer (or)
   → The first transfer of a burst
   → the address & control signals are unrelated to the previous transfer.
   → signal transfers on the bus are treated as burst of length one and therefore the transfer type is NONSEQUENTIAL.

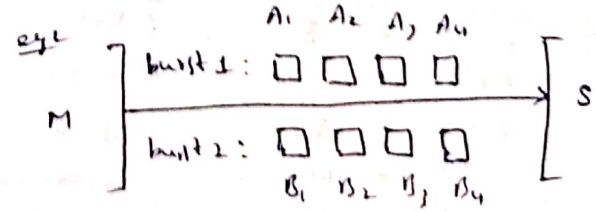
4) 11 : SEQ
   → The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer.

eg1



A1 - independent (Nonseq)
A2, A3, A4 } Address depends on previous packet address (Sequential)

eg2

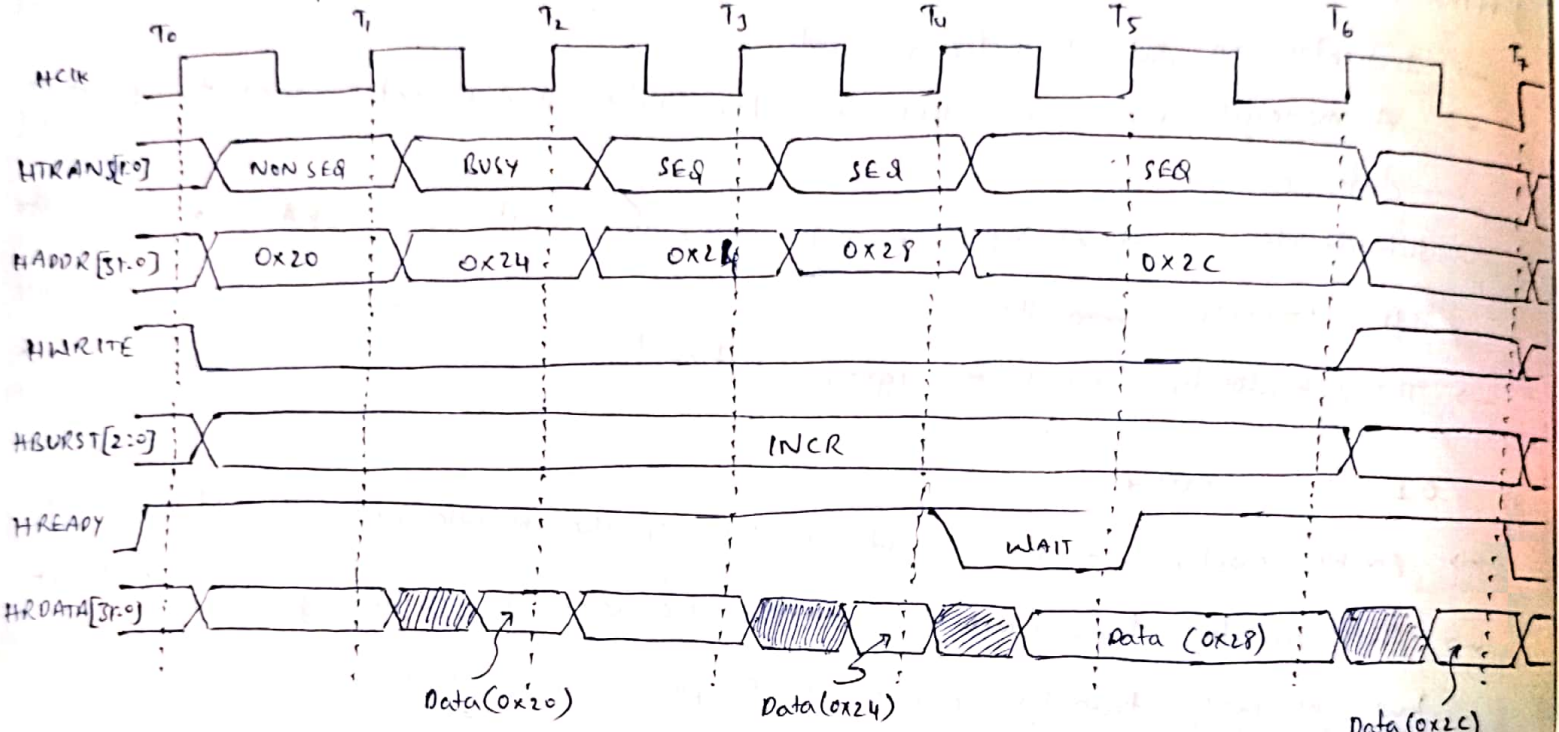


address "Bi" 1st packet of burst 2 is independent (doesn't depend on A4 last packet of burst 1)

HBURST [2:0] = INCR ; then undefined length burst

we don't know how many packets master is going to read/write

Transfer type examples



$T_0 - T_1$: 1st packet address phase, NON SEQ, address '0x20' is independent of other packet addresses in burst

$T_1 - T_2$: Master is asserting 'BUSY' signal (i.e Master is busy with some other transaction)

'////' → we are not getting data, when ever master asserting busy

HSIZE signal

HSIZE [2:0] → 3 bit signal, output to Master

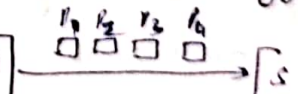
→ defines size of the packet

→ HSIZE must be less than or equal to the width of data bus

eg: 32 bit data bus HSIZE must only use values 0b000, 0b001, 0b010

HSIZE	Packet size	
000	8	Byte
001	16	Half word
010	32	Word
011	64	double word
100	128	4 word line
101	256	8 word line
110	512	
111	1024	

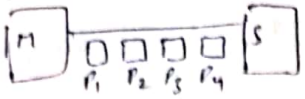
Note: Size = 1 byte



(HBURST) independent → P_1 = 0x20, P_2 = 0x21, P_3 = 0x22, P_4 = 0x23

Address of packet = previous packet address + size

Eg: 4 packets : 1 burst



HSIZE = 001 (ie 2 bytes)

address of P_1 = 0x24 (first setup)

P_2 = 0x24 + 2 bytes = 0x26

P_3 = 0x26 + 2 = 0x28

P_4 = 0x2A

Total data transfered in this burst = 4 packets x size of each packet

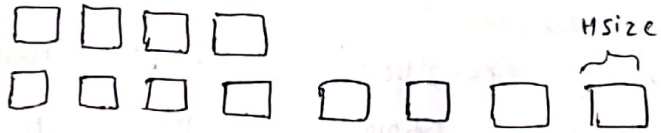
= 8 bytes.

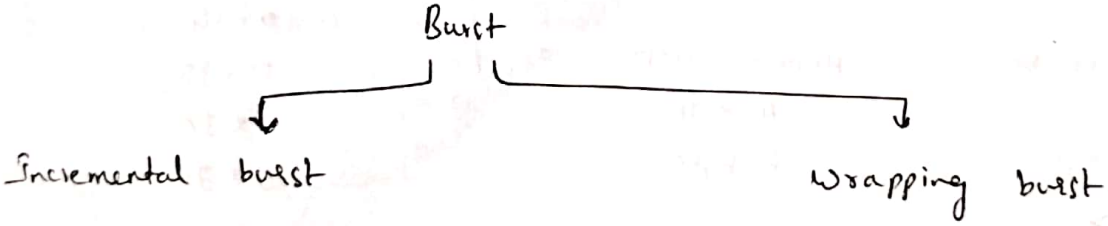
L-6

Burst Operations

* Burst = Collection of data items (eg packets data)
* Beat = no. of packets that we are going to send
* Burst of → 4, 8, 16 beats are defined in this AHB protocol
  - → undefined length bursts
  - → Single transfer
* Also support incrementing & wrapping bursts.

HBURST { 4-beat, 8-beat





Incremental Burst :

* It access sequential locations and the address of each transfer in the burst is an increment of the previous address

Eg:) HSIZE = 3'b000 (1 byte) & INCR = 4 (beats)

0x00 → +1 → 0x01 → +1 → 0x02 → +1 → 0x03

ii) HSIZE = 3'b001 (2 byte) & INCR = 4 beats

0x00 → +2 → 0x02 → +2 → 0x04 → +2 → 0x06

Wrapping Burst :

* we will wrap back to the initial / start address after reaching the address boundary
* **Address Boundary = HSIZE x HBURST**

eg: HSIZE = 3'b000 : 1 byte
HBURST = 3'b010 : WRAP 4
A-Boundary = 4 byte

* Wrapping bursts wrap when they cross an address boundary.
* The address boundary is calculated as the product of the number of beats in a burst and the size of the transfer.
* no. of beats are controlled by "HBURST"
* transfer size controlled by "HSIZE"

eg: four-beat wrapping burst of word access wraps at 16 byte boundaries.

i-e HBURST = 4 beats i.e 3'b010
HSIZE = word = 3'b010
(4 bytes)

Address boundary = HBURST x HSIZE
= 4 x 4
= 16 byte

* if start address = 0x34
then four addresses are
0x34
0x38
0x3C
0x30

BURST signal encoding

HBURST [2:0]	Type	Description
000	SINGLE	single transfer burst
001	INCR	Incrementing burst of undefined length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	
101	INCR8	
110	WRAP16	
111	INCR16	

example :

Slave	Start address	End address
Slave 0	0x0000_0000	0x0000_03FF
Slave 1	0x0000_0400	07FF
Slave 2	0800	BFF
Slave 3	0C00	0FFF

In burst the starting address can be any thing within slave address range"

Make sure INCR must not exceed ending address of particular slave

Four beat wrapping burst, WRAP4

Starting address → odd unaligned address

even Aligned addresses

eg: 0x00, 0x02, 0x04

0x01, 0x03, 0x05

AHB will not support unaligned addresses

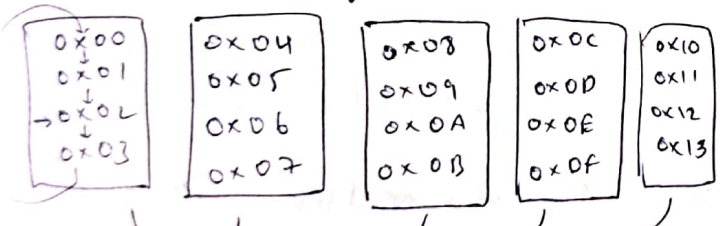
"starting address of burst cannot have odd address"

* when data size = 1 byte then no need to worry about aligned, unaligned addresses
* These come into picture only when data size ≥ 1 byte (HSIZE)

eg1) HBURST = WRAP4 (010)

HSIZE = 1 Byte (000)

Address Boundary => 4 Byte



4 byte different address boundaries

i.e HSIZE = 1byte ; Starting address can be anything (even or odd)

> 1byte ; Starting address must be even.

if starting address = 0x02

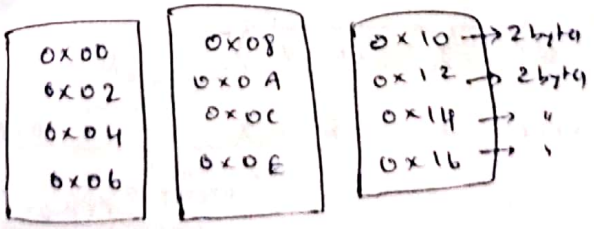
then addresses are 0x02 → 0x03 → 0x00 → 0x01

eg2 HSIZE = 2 bytes

HWrite = 1

H-BURST = WRAP-4

Address Boundary = 2×4 = 8 bytes



8-byte address boundaries

eg i) 0x00 → 0x02 → 0x04 → 0x06

ii) 0x02 → 0x04 → 0x06 → 0x00

L-7

WRAP8 : Eight beat Wrapping burst

1) HSIZE = WORD (4byte)

HBURST = WRAP8

A. Boundary = 4×8 = 32 byte

eg:

0x34 → 0x38 → 0x3C → 0x20

↳ 0x24 → 0x28 → 0x2C → 0x30

0x20 ← 4 byte each
0x24
0x28
0x2C
0x30
0x34
0x38
0x3C

32 byte A. Boundary

INCR8 : Eight ~~burst~~ beat incrementing burst

0x34 → 0x38 → 0x38 → 0x3A → 0x3C → 0x3E → 0x40 → 0x42 (No boundaries)

HSIZE = Half word (2bytes)

HBURST = INCR8

INCR : Undefined length bursts

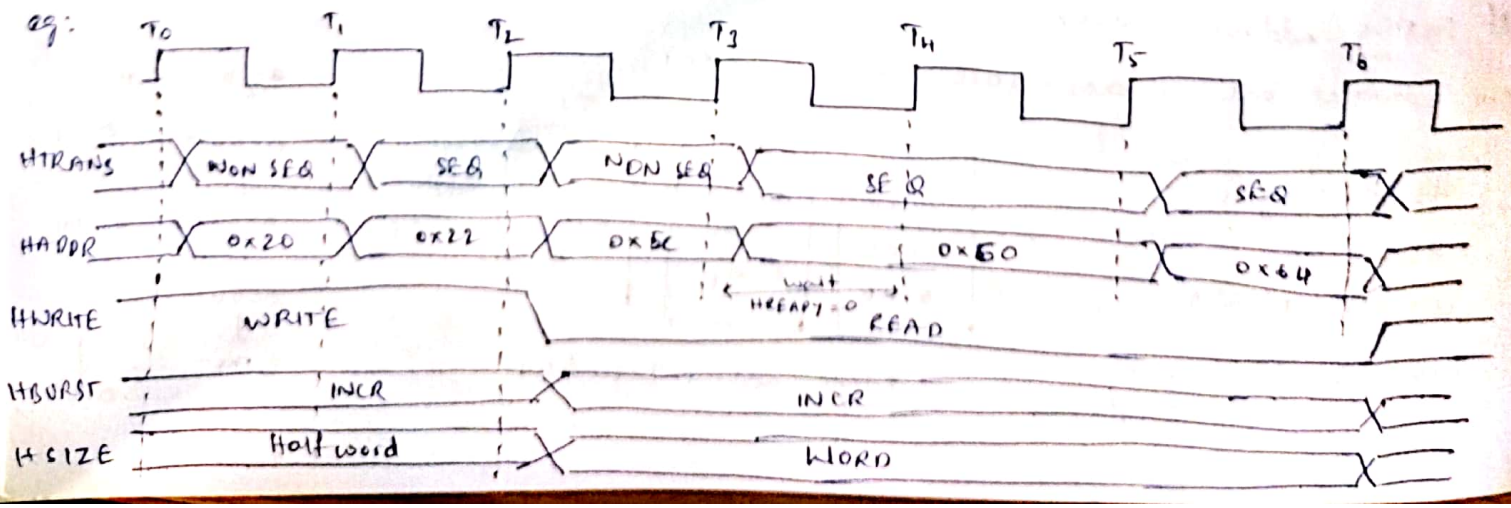
* no. of beats = unknown i.e In this INCR master doesn't know with how many beats of data it is dealing with

* In INCR4, INCR8, WRAP8 etc master knows how many beats of data is been transmitted.

Q) How slave judges whether burst transmission completed or not?

Sol: If transfer type changing from SEQ → NON SEQ then slave considered the previous transfer is completed & Master has started a new transfer

eg:



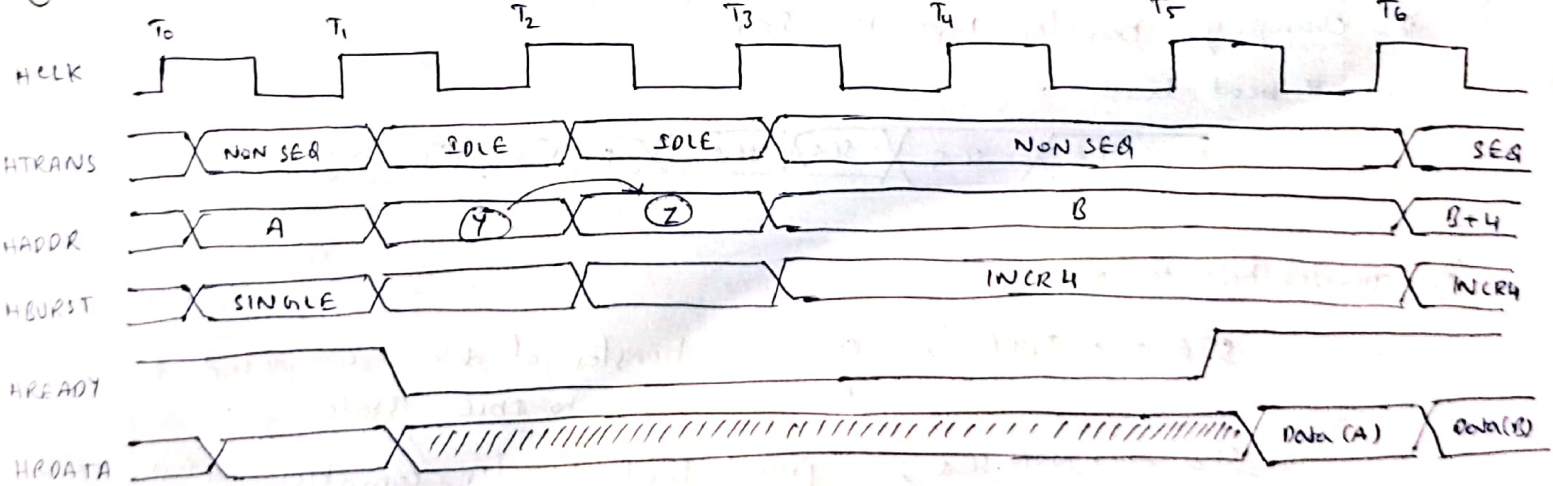
L-8 Subordinate Response Signalling

HRESP Response signal

↳ once the data transfer complete (Read/write) the slave will acknowled-ge the master through HRESP signal

HRESP	Response	Description
0	OKAY	* The transfer has either completed successfully or * additional cycles required for the subordinate to complete the request. The HREADYOUT indicates weather transfer is pending or complete
1	ERROR	* An error has occured during transfer. * The error condition must be signaled to the Manager so that it is aware of the transfer has been unsuccessful. * A two-cycle response is required for an error condition with HREADYOUT being asserted in the second cycle

① Address changes during waited transfer, with an IDLE transfer



When HREADY = 0 ; the Master should not change any of its control, address, data signals.

exceptions

i) HREADY = 0 & Transfer type = IDLE then Master is allowed to change address

ii) HREADY = 0 & HTRANS transfer type changes from IDLE to NON SEQ, then manager must not change any signal till HREADY is HIGH.

Scenario ② :

When HREADY = 0 (WAIT) ; → the Master is allowed to change HTRANS only from SEQ to BUSY or BUSY to SEQ

→ Master should not change any other signals (address etc) except HTRANS

Scenario ⑤ :

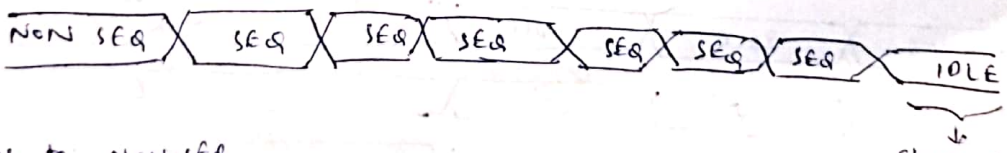
* During a waited transfer for an undefined length burst (INCR), the Manager is permitted to change from BUSY to any other transfer type, when HREADY = 0 is low.
* The BURST continues if SEQ transfer is performed but terminates if an IDLE or NONSEQ transfer is performed.

HREADY = 0 & HBURST = INCR then Manager can BUSY → NON SEQ / SEQ / IDLE

* Two ways to acknowledge SLAVE that burst transfer is completed

i) Changing transfer type to IDLE

eg 4 beat burst



Slave recognises the burst is completed

ii) Transfer type to NON SEQ

SEQ → IDLE ; Complete transfer of data completed & Master moves to IDLE state

SEQ → NON SEQ ; Present burst transfer completes, Master starts new transfer

"In this way Master & Slave will be in Synchronization in undefined length burst"

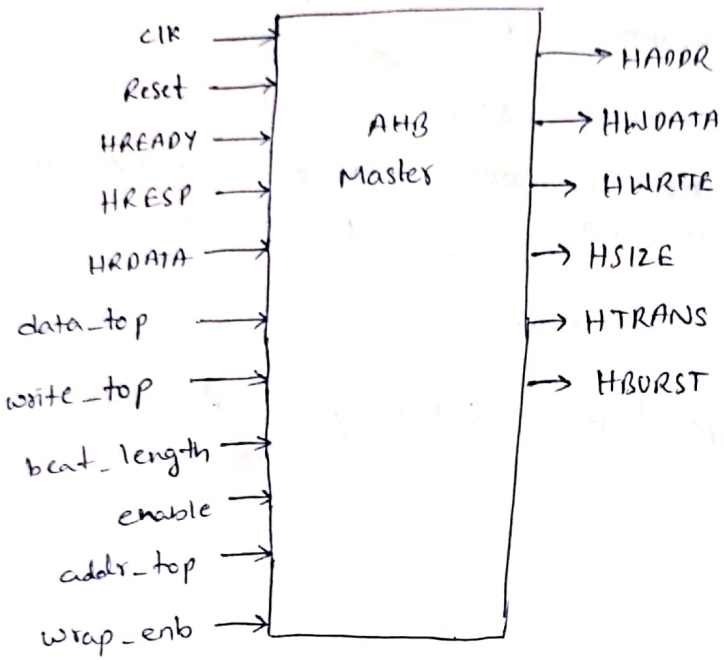
HREADY = 0
HBURST = INCR

BUSY → NON SEQ ; new transfer begins

BUSY → SEQ ; continuing same transfer

BUSY → IDLE ; present transfer completed we don't know

1-9 AHB Master in Verilog



i) data_top
→ user i/p signal
→ The data to be transfered is given here

ii) write_top
→ user i/p
→ write or read operation
write_top = 1 ; Master perform write op
=0 ; Read operation

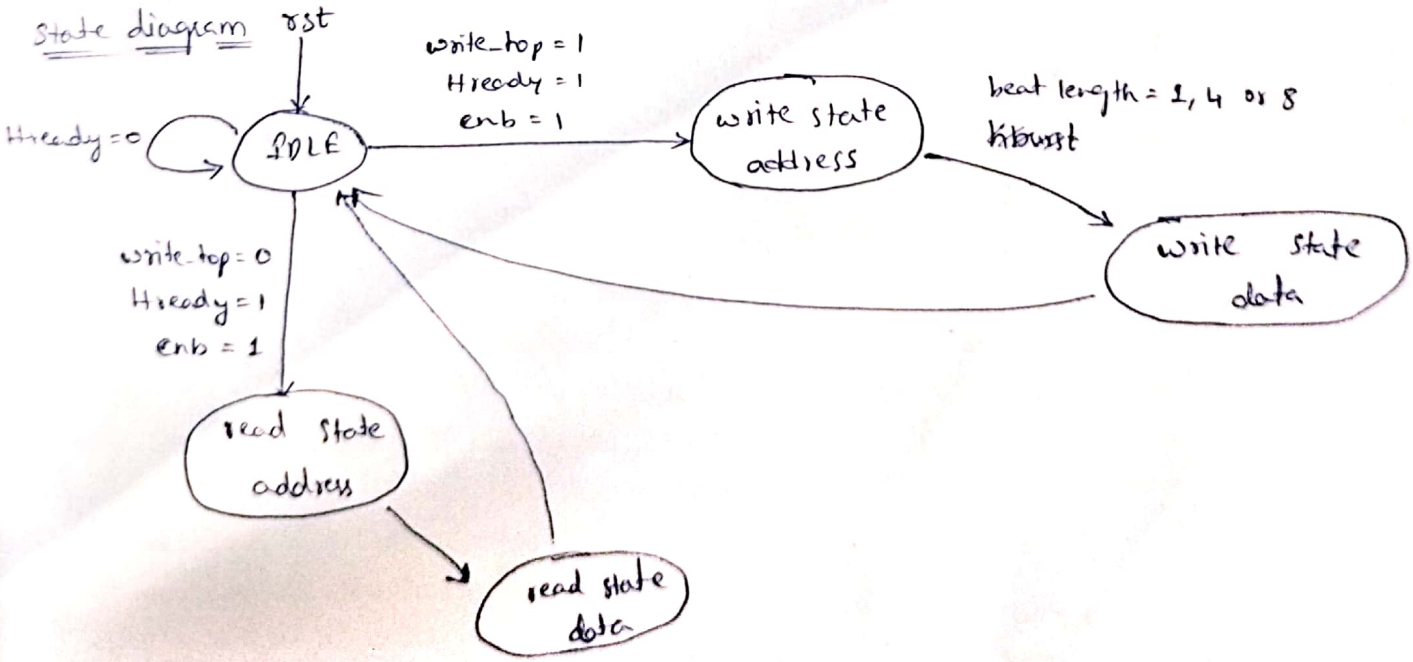
iii) Read_length
→ user i/p
→ If user want to perform burst operation then beat length is given here.

iv) enable
The user will provide all the data at data_top signal then make "enable" active so that master can understand the data is given by user & operation is perfo
enable = 0 ; user still feeding data
= 1 ; completed feeding data & Master proceeds it's operation

v) addr_top
user provide address

vi) wrap_enb
= 0 ; normal burst or INCR operation
= 1 ; wrap operation

State diagram



1) IDLE state :

HBURST = xx (don't care)

HSIZE = xx

HTRANS = 0 (IDLE)

2) Write state address

HSIZE = 3'b010 (4 byte data) eg

HWRITE = 1

3) Read state address .

HBURST = 000 HSIZE = 010 (4 bytes)

HTRANS HWRITE = 0

4) Write state data

beat_length
Hready = 1 } check & perform write operation

5) Read state data

check for beat length, WRAP condition & type of burst & Hready if condition met then perform read oper