AHB to APB Bridge Verification

→ Bridge is an Interface b/w IP's of two different protocols.

→ Bridge Converts one standard interface Signals to another standard Interface Signals.

→ Here, AHB to APB Bridge Converts the AHB Bus transfers to Equivalent APB Bus transfers.

→ we have to understood the AHB and APB protocols, before the verification of AHB to APB Bridge.

outlines :-

1. AHB protocol
2. APB protocol
3. AHB to APB Bridge
4. AHB to APB Bridge verification.

* what is a Bus?

→ BUS is a standard Interface to Connect IP's

* What is a Interconnect?

→ Interconnect is a Junction b/w multiple Masters and Slaves within a chip, which enables fair data access b/w IP's.

* What is a bridges?

→ Bridges Converts one Standard bus Interface Signals to another Standard bus interface Signals.

Interconnect

→ Connect components within a single system

→ Manage high-speed data transfer within the systems.

→ Interconnects are designed for High speed internal communication within systems.

→ Interconnects typically do not perform protocol conversion b/w different standard interfaces

→ The is primarily for high speed data transfers within a single system (or) subsystem, usually within a common protocol (or) signalling standard.

Bridges.

→ Connect different networks or segments.

→ Manages data flow and protocol translations b/w segments.

→ Bridges can link different network segments, often handling protocol translation and selective data forwarding.

→ Bridges are specially designed to convert the data b/w (or) signals the different protocols (or) standards.

Ex: AHB to APB Bridge

→ AHB transactions/signals are converted to equivalent APB signals.

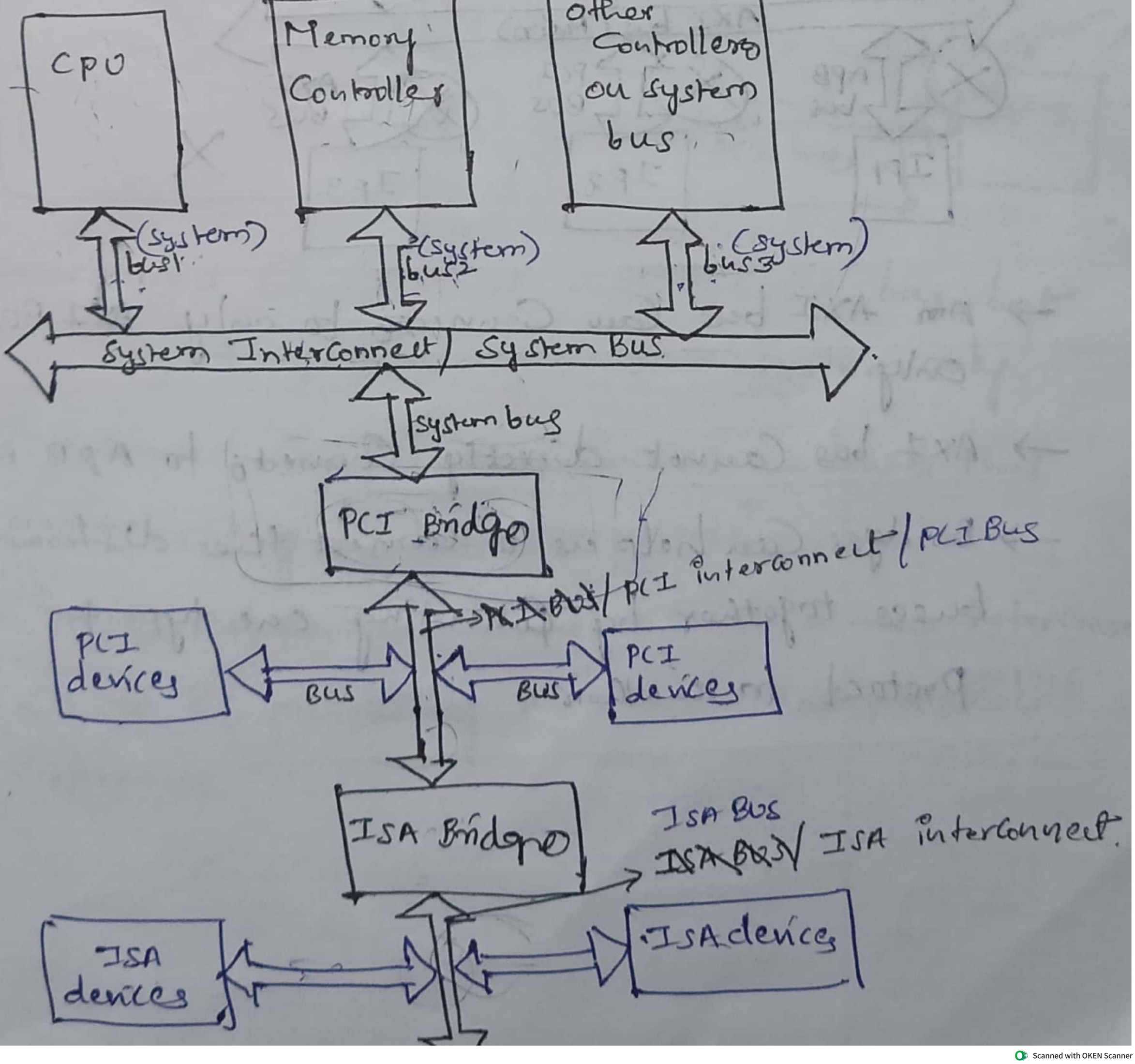
Note:

→ Interconnects lack the translation logic found in bridges,

→ Interconnects can connect the components/IP's of same standard/ protocol only.

→ protocol Converters (or) Bridges are used where the translation b/w different protocols is required.

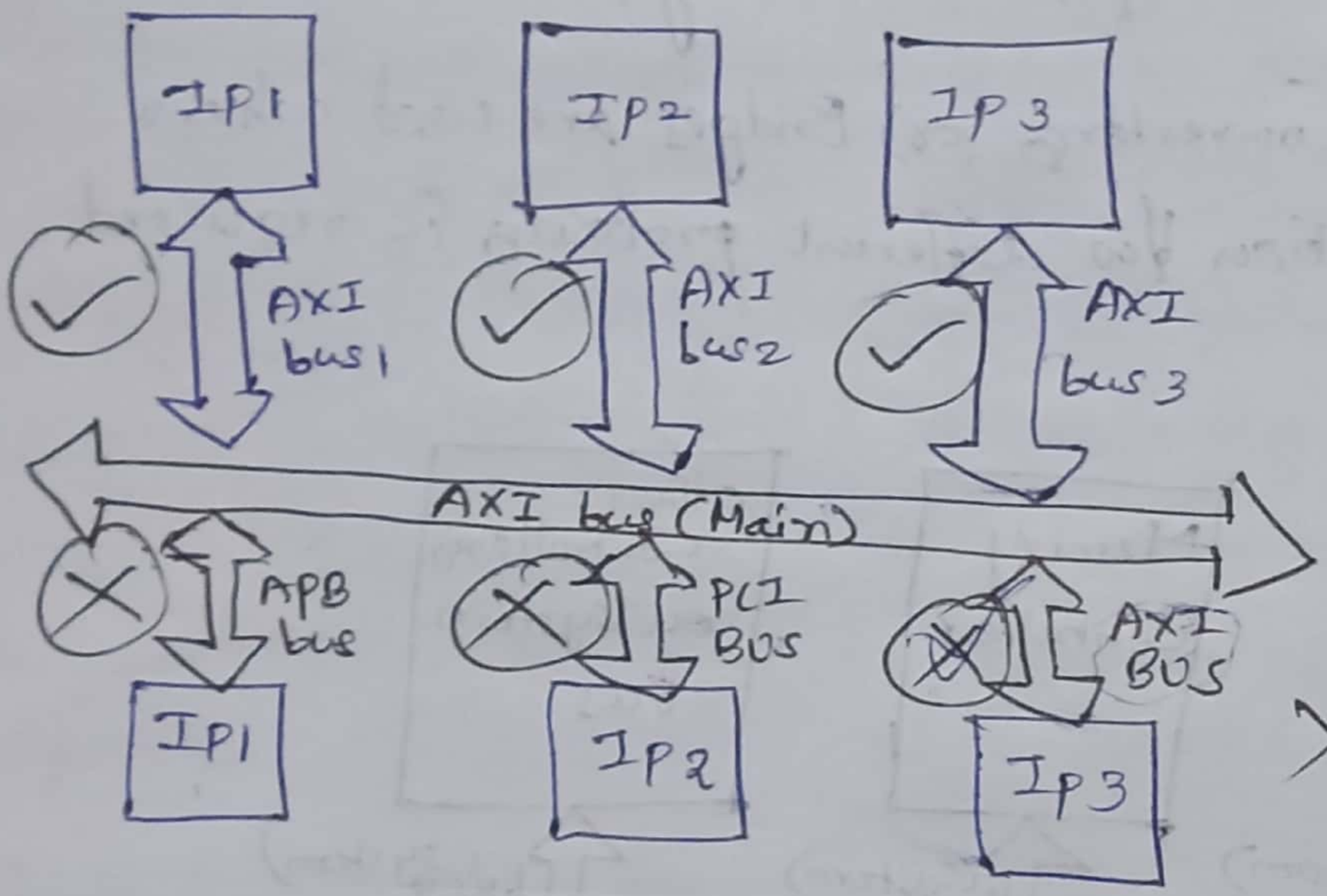
Ex:



Note :-1

→ Bus can connects two or more number of buses of same type

→ Bus cannot connects two different types of buses



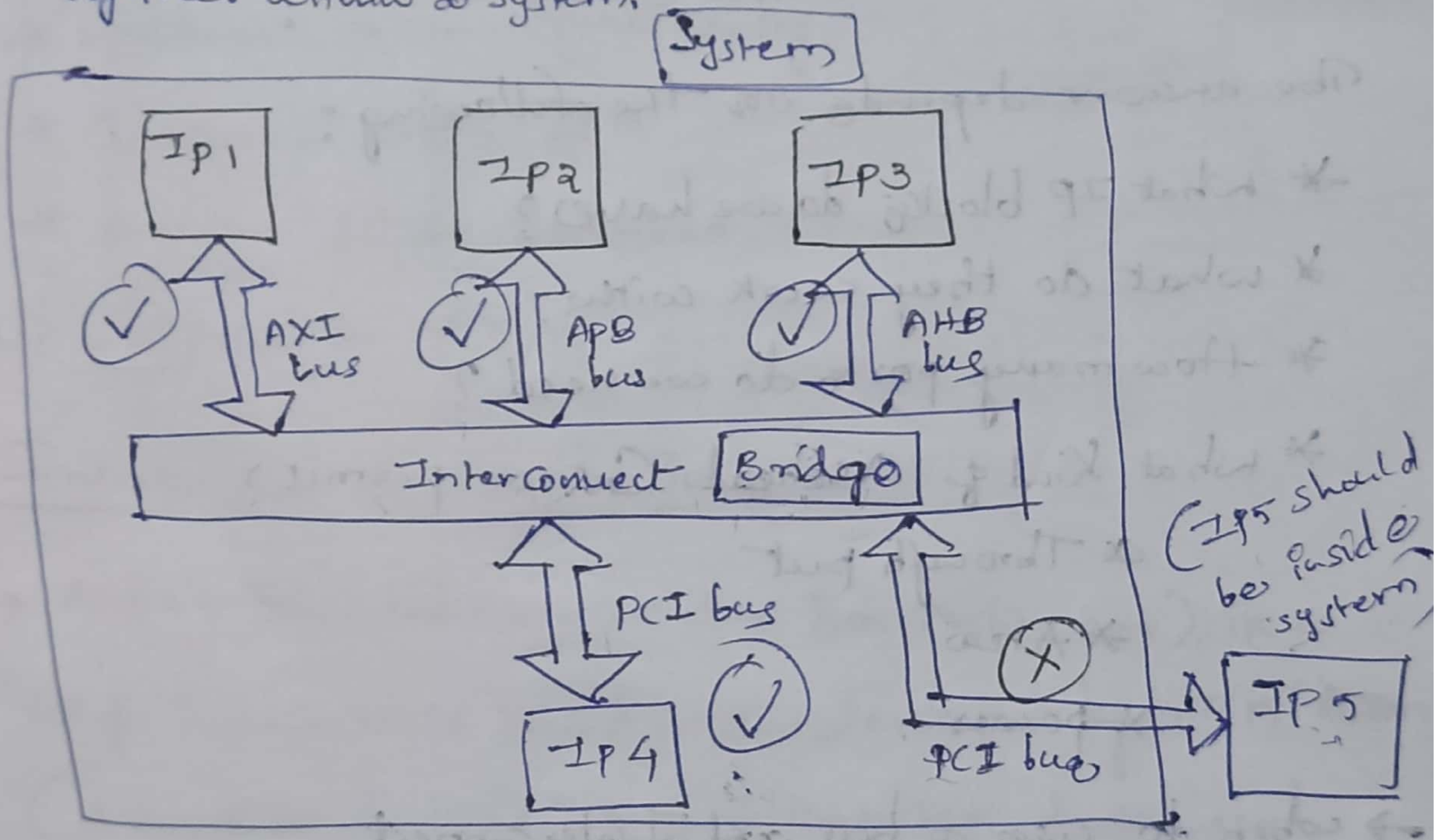
→ An AXI bus can connects to only AXI BUS only

→ AXI bus cannot directly connects to APB Bus

→ Bridges can help us to connect few different buses together by converting one type of protocol to another.

Note 2

→ InterConnects Can Connect the different buses types together within a system.



→ Interconnect Can Connects any types of bus, but within the system/segment/subsystem only.

→ InterConnects Can indirectly Connect tee buses that are operating on different protocols, then a bridge (or) protocol Converter within the InterConnect set up is necessary to handle the protocol differences.

Bus or Interconnect Selection:-

Q: which bus or Interconnect should we use?

The answer depends on the following:

* What IP blocks do we have?
* what do they cook with?
* How many ports do we need?
* What kind of overhead can we permit?
    * Through put
    * Area
    * power.

→ when to use a bus and Interconnect.

Criteria	Use a Bus	Use an Interconnect
System Complexity	Simple, limited no. of. Components	Complex systems with multiple cores / devices
Data Bandwidth	Low to moderate	High BW & heavy data operations
Communication pattern	Sequential & single shared path	Multi-directional, parallel Communication
Scalability	Limited Scalability	High scalability with multiple Components
Component Compatibility	Same protocol, uniform Components	Supports various buses, protocol Conversion ready

Buses and Interconnects :-

* AMBA from ARM (Acorn RISC Machine)
* Wishbone from opencores.org
* CoreConnect from IBM
* Sonics "Silicon Backplane"

We will focus on AMBA :- - - -

Introduction to AMBA Buses :-

→ AMBA (Advanced MicroController Bus Architecture) is a set of interconnects specifications developed by ARM (Acorn RISC Machine) to facilitate the design and Integration of components in SOC (System-on-chip) architectures.

→ AMBA is a standard for Connecting various digital components within a chip, enabling efficient Communication b/w the processor, memory, and peripherals.

→ AMBA, created by ARM as an interface for their microprocessors.

→ Very Common in Commercial SoC's (e.g., Qualcomm Multimedia Cellphone SoC)

→ Actually AMBA Supports 3 standards :-

1. APB (Advanced peripheral BUS)
2. AHB (Advanced High performance Bus)
3. AXI (Advanced eXtensible Interface)

→ AMBA Versions

1. AMBA 2.0 :- (released in 1999)

Introduced 3 primary bus types :-

1. APB
2. AHB
3. AXI

2. AMBA 3.0 :- (released in 2005)

* Enhanced the AXI protocol with support for QoS (Quality of Service) features, allowing system to prioritize certain data transfers over others.
* Added support for a flexible interconnect architecture, enabling easier integration of components and improved system performance.

3. AMBA 4 :- (released in 2010)

* Introduced the AXI family, which includes
  * AXI4
  * AXI4 - Lite (Simplified version of AXI4)

* AXI4-Stream (for Streaming data applications)
(Video processing, high speed data aquisition systems)

5. AMBA 5 :- (released in 2018)

* Introduced CHI (Coherent Hub Interface)

→ designed for multi-core systems, CHI enables Coherent memory access and shared data b/w multiple processors.

Version	Release year	Key features
AMBA 1.0	Early 1990s	Foundation for bus architecture
AMBA 2.0	1999	Introduced APB, AHB, and AXI
AMBA 3	2005	Enhanced AXI with QoS support
AMBA 4	2010	Introduced AXI4, AXI4-Lite, and AXI4-Stream
AMBA 5	2018	Introduced CHI for Coherent multi-core systems.

→ The AMBA specifications defines an onchip Communications standard for designing high-performance embedded microcontrollers.

→ 3 distinct buses are defined within the AMBA Specifications

* AHB
* APB
* AXI

AHB Bus :-

AHB - Advanced High performance Bus

* The AMBA AHB bus is used for high performance and high clock frequency system modules.
* All the high-performance IP's that are working at high clock frequencies will be connected to AHB bus.
* The AHB bus acts as a backbone bus of the high performance system.
* AHB supports
  1. efficient Connection of processors.
  2. Onchip memories & off-chip memory external memory interfaces with low-power peripheral macrocell functions.
* AHB is also easily used in an efficient design flow using synthesis and automated test techniques.

APB Bus :-

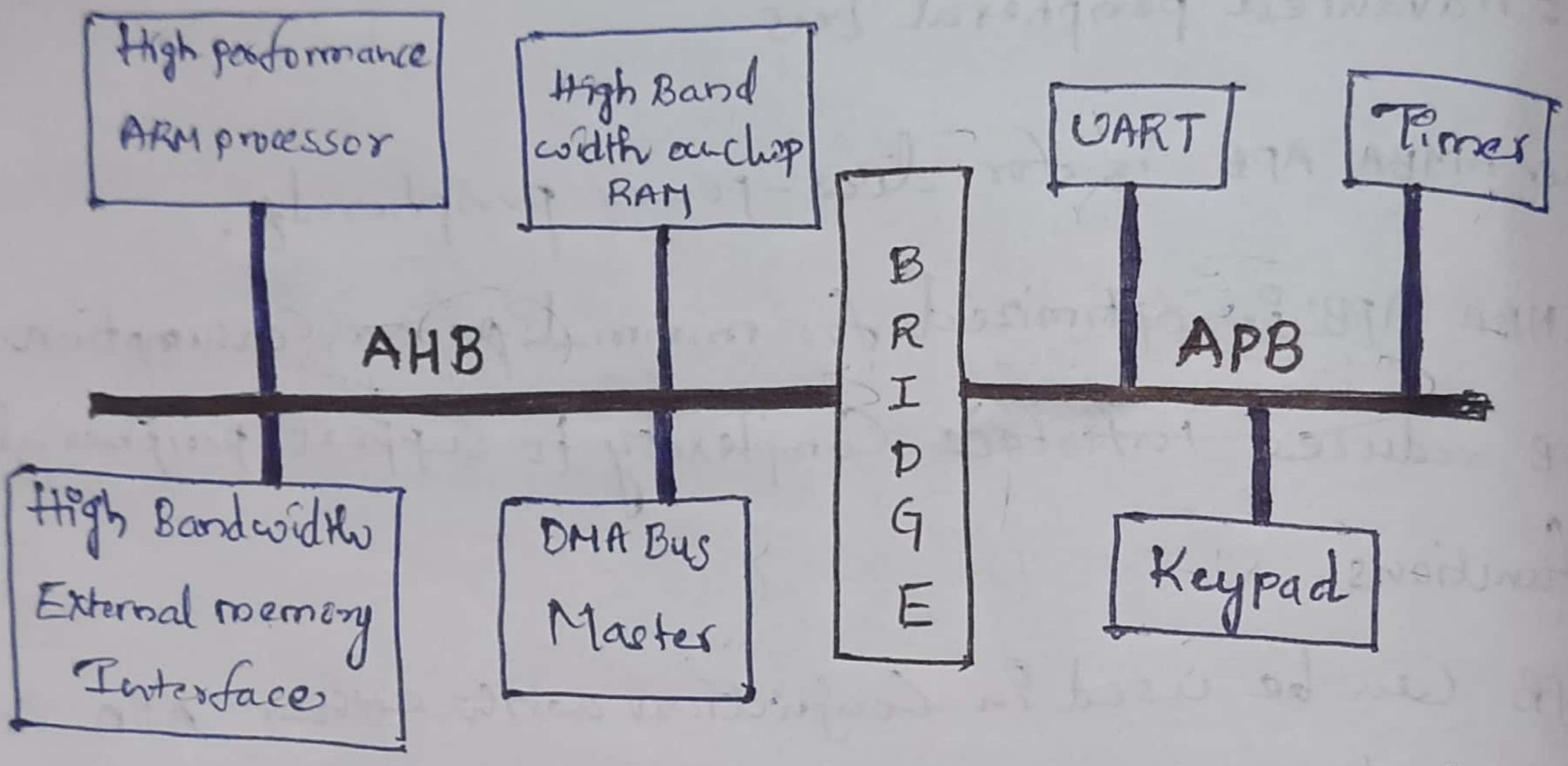
APB = Advanced peripheral Bus

* The AMBA APB is for low-power peripherals.
* AMBA APB is optimized for minimal power consumption
* APB reduced interface Complexity to support peripheral functions.
* APB can be used in conjuction with either AHB or AXI bus system

overview of AMBA Buses :-

AHB	APB
High performance	Low power
Pipelined operation	Latched address and Control
Multiple Bus Masters	Simple Interface
Burst Transfers	-
Split transactions	-
-	Suitable for many low power peripherals.

A typical AMBA-based MicroController



→ All the high-performance IP's/peripherals are Connected to AHB Bus.

→ All the low-power peripherals are Connected to APB Bus.

→ BRIDGE act as a interface b/w high-performance peripherals and low power peripherals.

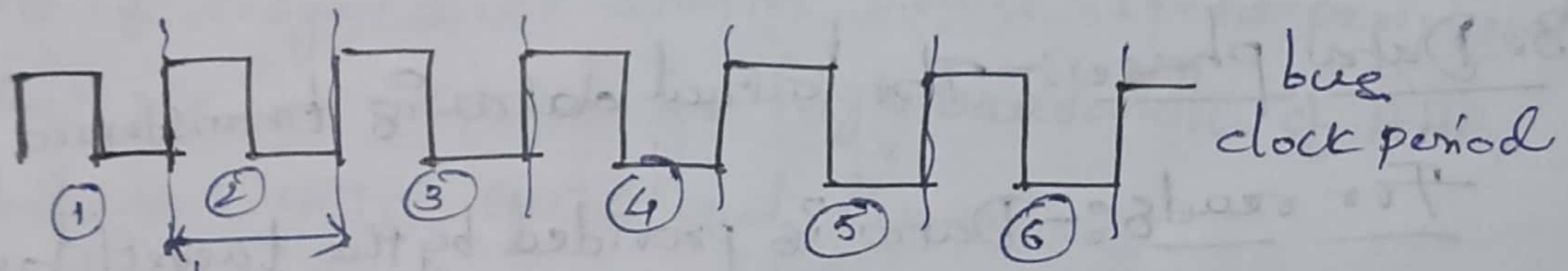
→ BRIDGE Converts the AHB transfers to equivalent APB transfers

→ BRIDGE acts as
  * AHB slave
  * APB Master

Terminology :-

1. Bus cycle

A bus cycle is a basic unit of one bus clock period and it is defined from rising-edge to rising-edge transitions.



→ here, bus clock period = 6 cycles

2. Bus transfer :-

(i) AMBA AHB Bus transfer :-

(ii) AMBA APB BUS transfer

→ Bus transfer (or) Bus transaction refers to the process of transferring data over a shared communication bus b/w different devices(or) components, such as a processor, memory and other peripheral devices.

→ A bus transfer typically includes

1. Address phase :- The bus Master (the device initiating the transfer) places the address of the target device (or) memory location of the slave on to the bus, indicating where data should be read from or written to.

2. Command phase :- The bus Master sends a Command or Control Signal to indicate the type of transfer (read or write).

→ Some protocols specify burst lengths or timing Controls here.

3. Data phase :- The actual data is transferred.

For reads :- Data is provided by the target/slave and sent to the master.

For writes :- Data Comes from the Master and is sent to the target.

1. AMBA AHB bus transfer :-

→ An AMBA AHB bus transfer is a read or write operation of a data object

→ It may take one or more bus cycles.

→ Bus transfer is terminated by a Completion response from the addressed slave.

→ Transfer size supported by the AMBA AHB included

* bytes (8-bit)
* halfword (16-bit)
* words (32-bit)

→ It also supports wider data transfers, including

* 64-bit transfers
* 128-bit transfers

→ Transaction/Transfer phases
1. Address phase
2. Command phase
3. Data phase

2. AMBA APB Bus transfer :-

→ APB is designed for simpler, lower-speed peripherals that do not require the high bandwidth of AHB, such as UARTs, GPIO, and timers.

→ Two-phase transfer
* setup phase
* Access phase

→ An AMBA APB bus transfer is also a read or write operation of a data object, which always requires 2 clock cycles.

→ Transfer sizes supported by AMBA APB include
* 8-bit transfer
* 16-bit transfer
* 32-bit transfer

Feature	AHB	APB
purpose	High-performance components	Low speed peripherals
operations	Read (or) write	Read (or) write
clock cycles	one (or) more clock cycles	always 2 clock cycles
transfer size	8-bit, 16-bit 32-bit / 64-bit, 128-bit	8-bit, 16-bit 32-bit
Pipelining	Yes (supports pipelined transfers)	No (non-pipelined)
Burst transfers	Supported	Not supported
Power Consumption	Higher due to Complexity	Lower due to simplicity
Transfer Speed	Higher speed, suitable for memory	Low-speed, ideal for peripheral

Burst operation :-

→ Burst operation means transferring a sequence of data (multiple words or bytes) in a single transfer/ single transaction.

→ Burst transfer allocas multiple data words to be transferred in a single transaction.

Characteristics :-

1. Single Address phase :- A burst transfer in AHB begin with the master providing an initial address. This address is typically for the first data word in a sequence, and subsequent addresses are generated automatically.

2. Automatic address incrementing :- After each data word is transferred, the address for the next word is incremented (or wrapped), without requiring a new address phase. This minimizes delays.

3. Multiple burst lengths supported :-

AHB defines different burst lengths:

* Single (Hburst = 000) : only one data word is transferred

* 4-beat Burst (Hburst = 011) : Transfers 4 data coords
* 8-beat Burst (Hburst = 101) : Transfers 8 data coords.
* 16-beat Burst (Hburst = 111) : Transfers 16 data coords

beat → Transfer

4 beats → 4 transfers

4. Data phase pipelining :-

→ Address and control phases for subsequent transfers can overlap with the dataphase of current transfers.

Steps

1. Initiation :- The Master initiates the burst transfer by setting the starting address, control signals, and the HBURST signal to indicate the burst length.

2. Data Transfer :- The data is transferred on consecutive cycles. For each clock cycle the data will be read or written to the next sequential address based on the type of control signals (HCONTROL) → HREAD (or) HWRITE

3. Completion:- The burst Completes after the specified number of data words (ex:- 4, 8, or 16 words) Afterward, the bus may return to idle, or a new transaction can begin.

∴ Burst transfers are for fast data movement, such as DMA operations, memory accesses, and transfers to high-speed peripherals for embedded system.

∴ A burst operation is defined as one or more data transactions, initiated by a bus master, which have a consistent width of transaction to an incremental region of address space. The increment step per transaction is determined by the width of transfer (byte, half-word, word

⇒ No burst operations is supported by the APB.

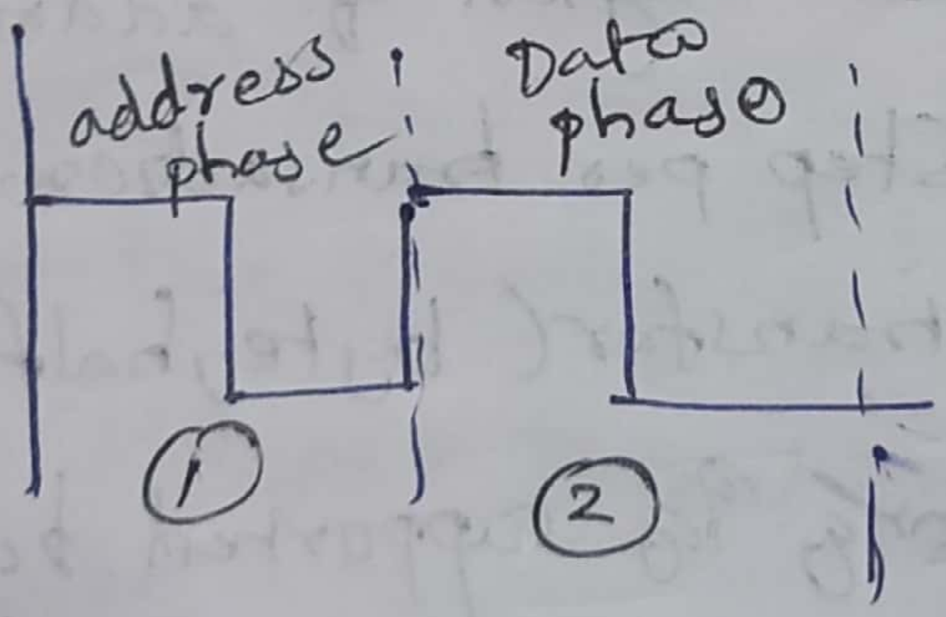
AHB Bus/protocol:-

Features of AHB Bus:-

→ AHB is a new generation of AMBA Bus, which is intended to address the requirements of high-performance synthesizable designs.

→ AMBA AHB is a new level of a bus which sits above the APB and implements the features required for high-performance, high clock frequency systems including:-

* burst transfers
* split transactions.



1st cycle:-

→ address, Control Signals (either read (or) write)

→ Size of each transaction (1, 2, 4, 8, 16 ...)
2^0 2^1 2^2 2^3 2^4

→ type of transfer

2nd cycle:-

→ actual data will do write (or) read operation to the address of the slave.

* Single cycle bus master handover

→ The switching b/w different masters will happen for a single bus cycle.

* Single clock edge operation

→ Read (or) write operation will takes place at the single clock edge only.

* non-tristate implementation

* Wider data bus configuration (64/128 bits).

AHB Complex Bus :-

→ Larger overhead - ~27 control signals

→ Up to 15 masters allowed

Totally 16 Masters → 15 Masters (active)
→ 1 Master (dummy)

→ Split transaction phases : Address, Data (Pipelined)

→ HREADY signal allows insertion of wait states

→ AHB supports the limitations of APB.

* Multi-Master
* multiple outstanding transactions
* back-to-back transactions

→ So, like this interconnect will take care of fair data access b/w the Master and slave.

AHB Signal descriptions :-

Signal Name	Source	Description
HCLK Bus clock	clock Source	This clock will trigger the all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn Reset	Reset Controller	The bus reset signal is active Low and is used to reset the system and the bus. This is the only active Low signal.
HADDR[31:0] Address Bus	Master	The 32-bit system address
HTRANS[1:0] Transfer type	Master	Indicates the type of the Current transfer, which Can be 10 - NONSEQUENTIAL 11 - SEQUENTIAL 00 - IDLE 01 - BUSY

Signal name	Source	Description
HWRITE Transfer Direction	Master	when this signal high, it indicates a write transfer and when low a read transfer. write → Master is sending the data on HWDATA bus- Read → Slave is sending data on HRDATA bus.
HSIZE[2:0] Transfer size	Master	Indicates the size of the transfer, which is typically byte (8-bit), half word (16-bit), word (32-bit). → The protocol allows for larger transfer sizes upto a maximum of 1024 bits.
HBURST[2:0] Burst type Indicates 1. Length of burst 2. type of burst)	Master	Indicates if the transfer forms part of a burst → 4 beat burst → 8 beat burst → 16 beat bursts are supported. and burst may be either incrementing (or) wrapping.

Signal Name	Source	Description
HWDATA [31:0] Write Data Bus	Master	→ The write data bus is used to transfer data from the master to the bus slaves during write operations. → A minimum data bus width of 32-bits is recommended. → However, this may easily be extended to allow for higher bandwidth operations.
HRDATA [31:0] Read data Bus	Slave	→ The read data bus is used to transfer data from the bus slaves to the bus Master during read operations. → A minimum data bus width of 32-bits is recommended. → However, this may easily be extended to allow for higher bandwidth operations.
HREADY Transfer done	Slave	→ When HIGH the HREADY signal indicates that a transfer has finished on the bus.

→ This signal may be driven to Low to extend a transfer.

→ Note :- Slaves on the bus require HREADY as both an input and an output signal.

HRESP[1:0]
Transfer response

Slave → slave provides the actual response whether the transfer is Completed or not.

→ The transfer response provides additional information on the status of a transfer.

→ Four different responses are provided.

* OKAY ✓
* ERROR.
* RETRY.
* SPLIT.

HTrans Signal

HTrans[1:0]	Transfer Type	Description
00	Idle	→ Indicates that no data transfer is required. → The IDLE transfer type is used when a bus master is granted the bus, but Master does not wish to perform a data transfer. → Slaves must always provide Zero wait state OKAY response to IDLE transfer. → IDLE transfer should be ignored by the Slave.
01	Busy	→ The BUSY transfer type allows bus Masters to insert IDLE cycles in the middle of the burst transfers. → This transfer type indicates that the bus master is continuing with a burst of transfers, but

the next transfer cannot take place immediately.

→ When a Master uses the BUSY transfer type the address and control signals must reflect in the next transfer of the burst.

→ This transfer should be ignored by the slave.

→ Slaves must always provide zero wait state OKAY response, in the same way that they respond to IDLE transfer type.

10 — Non-sequential → Indicates the first transfer of a burst or a single transfer.

→ The address and control signals are unrelated to the previous transfers.

→ Single transfers on the bus are treated as the bursts of one and therefore the transfer type is NON SEQUENTIAL.

→ very first time transfer.

11 — Sequential → The remaining transfers in a burst are SEQUENTIAL except the very first time transfer. (which is NON SEQ)

→ The address of related to the previous transfer.

→ The Control information of identical to the previous transfer.

→ The address size of equal to the address of the previous transfer size plus the size (in bytes)

→ In the case of a wrapping burst, the address of the transfer wraps at the address boundary equal to the size (in bytes) multiplied by the no. of beats in the transfer (either 4, 8, or

HBurst Signals

→ HBurst signal indicates the type of the burst & length of the burst.

1. length of the burst
   * 1-beat
   * 4-beat
   * 8-beat
   * 16-beat

2. Type of the burst
   * Increment type
   * Wrapping type

Hburst [2:0]	Type	Description
000 (0)	Single	Single transfer (1-beat)
001 (1)	INCR	Length is not specified (undefined/ Variable length)
010 (2)	WRAP4	4-beat wrapping burst
011 (3)	INCR4	4-beat incrementing burst
100 (4)	WRAP8	8-beat wrapping burst
101 (5)	INCR8	8-beat incrementing burst
110 (6)	WRAP16	16-beat wrapping burst
111 (7)	INCR16	16-beat incrementing burst.

→ For WRAP burst, length supported is 4, 8 and 16.

→ For INCR burst, length supported is 4, 8, 16 and with different lengths. (i.e., 15, 20, 29 etc).

→ Burst must not cross a 1KB address boundary.

→ Therefore, it is important that masters do not attempt to start a fixed-length incrementing burst which would cause this boundary to be crossed.

Hsize Signal :- represents the size/width of the transfer data

Hsize [2:0]	Size	Description
000 $0\times2^2+0\times2^1+0\times2^0=0$ 2^0 = 1-byte	1-byte (8-bits)	Byte
001 $0\times2^2+0\times2^1+1\times2^0=1$ 2^1 = 2-byte	2-bytes (16-bits)	Half-word
010 $0\times2^2+1\times2^1+0\times2^0=2$ 2^2 = 4-byte	4-bytes (32-bits)	Full-word
011 $0\times2^2+1\times2^1+1\times2^0=3$ 2^3 = 8-byte	8-bytes (64-bits)	-
100 $1\times2^2+0\times2^1+0\times2^0=4$ 2^4 = 16-byte	16-bytes (128-bits)	4 word line

HSize[2:0]	Size	Description
101 $1\times2^2+0\times2^1+1\times2^0=5$ $2^5=32$-byte	32-bytes (256-bits)	8-word line
110 $1\times2^2+1\times2^1+0\times2^0=6$ $2^6=64$-byte	64-bytes (512-bits)	-
111 $1\times2^2+1\times2^1+1\times2^0=4+2+1$ $2^7=128$-byte	128-bytes (1024-bits)	-

AHB protocol :—

→ In AHB, the entire transfer will be done in two phases

1. Address phase
2. Data phase

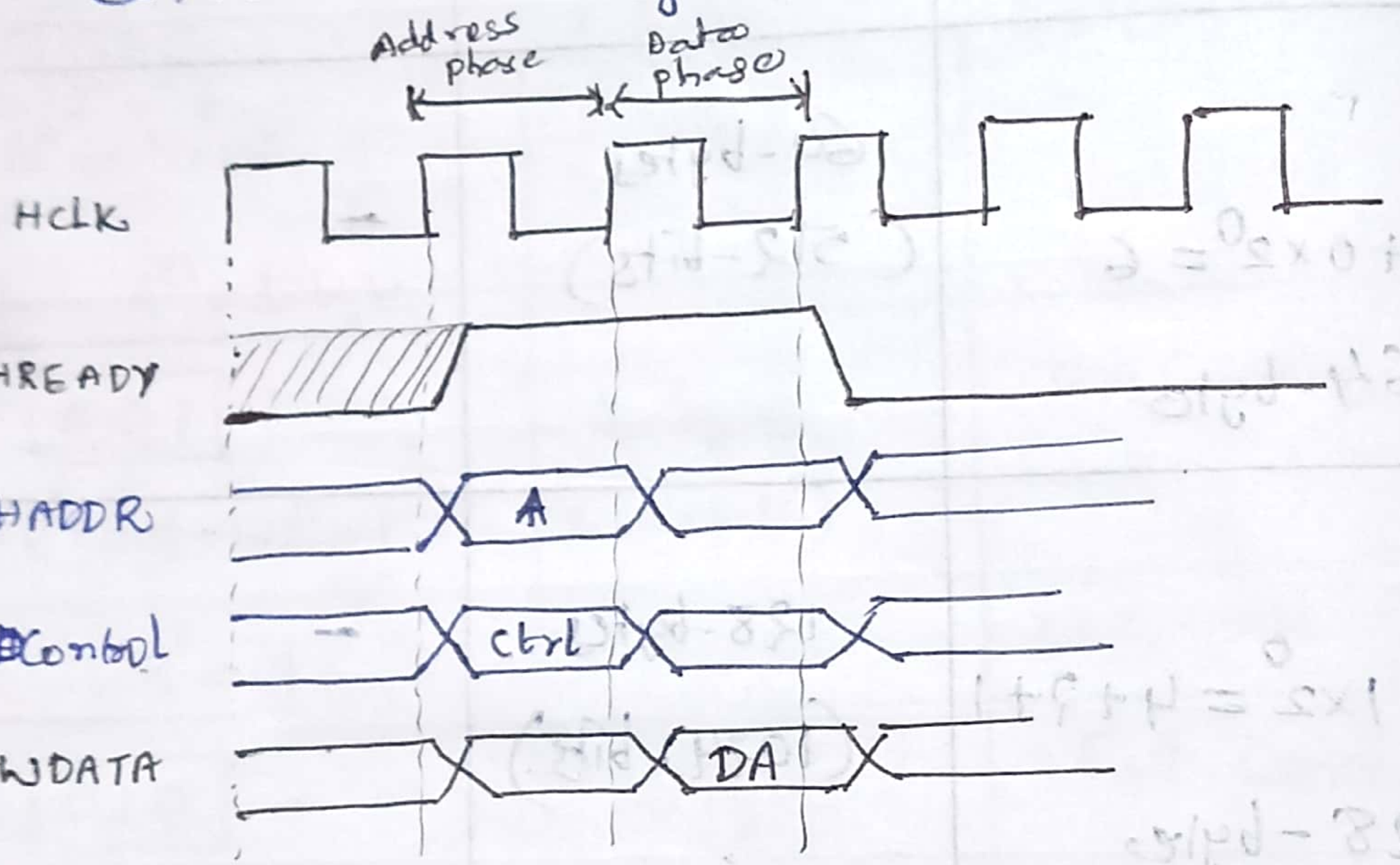
→ In Address phase, the master will send the address and control information.

→ In the Data phase, the data will be available.

→ HREADY signal sent by the slave, indicating whether it can complete the transfer immediately on the next posedge

→ So, to Complete a transfer, it requires minimum of two bus cycles / clock cycles.

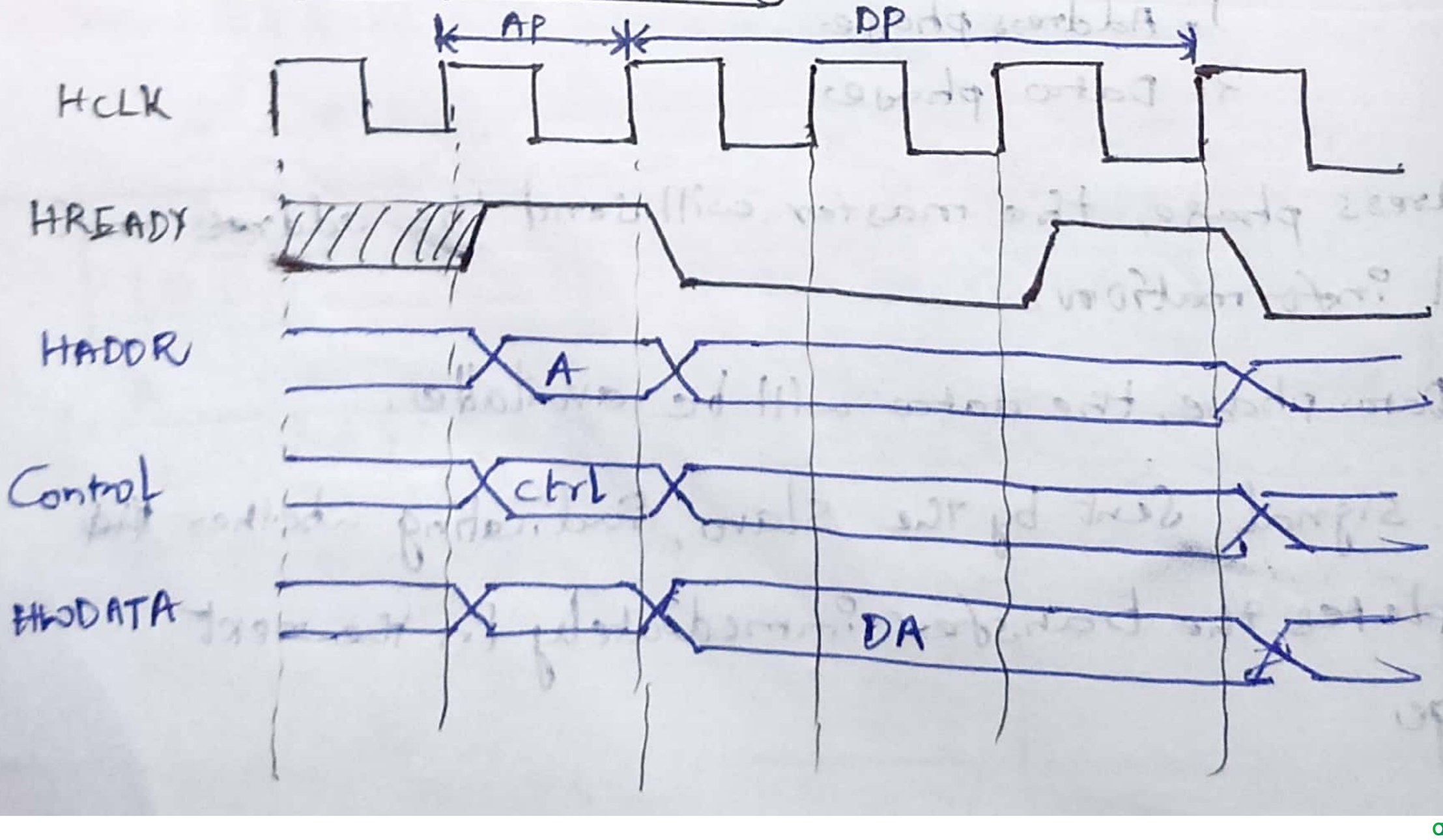
→ Address phase Cannot be extended, but data phase Can be extended by the slaves



Here, Address - A, Control signal - Ctrl Sent by the master in address phase.

Data of address - A (DA) available in Data-phase.

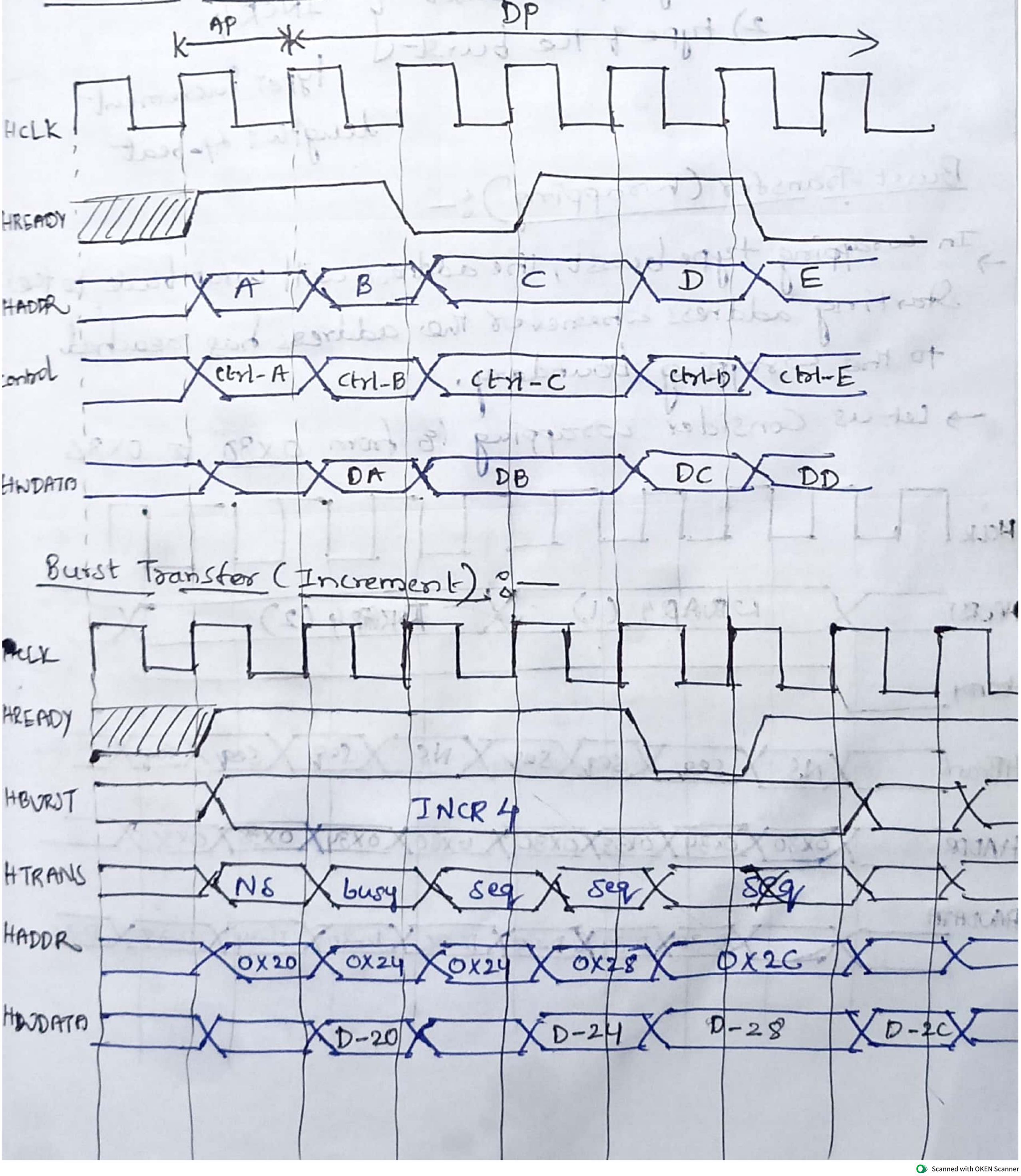
Transfer with wait-states



→ Slave can insert wait states by making HREADY to Zero and extending data-phase.

→ Address phase cannot be extended as per AHB protocol.

Multiple transfers :-



Burst Transfer (Increment) :-

→ In Burst transfer, the very first transfer is Non-seq followed by Sequential (or) other.

→ HBURST should be same throughout the transfer.

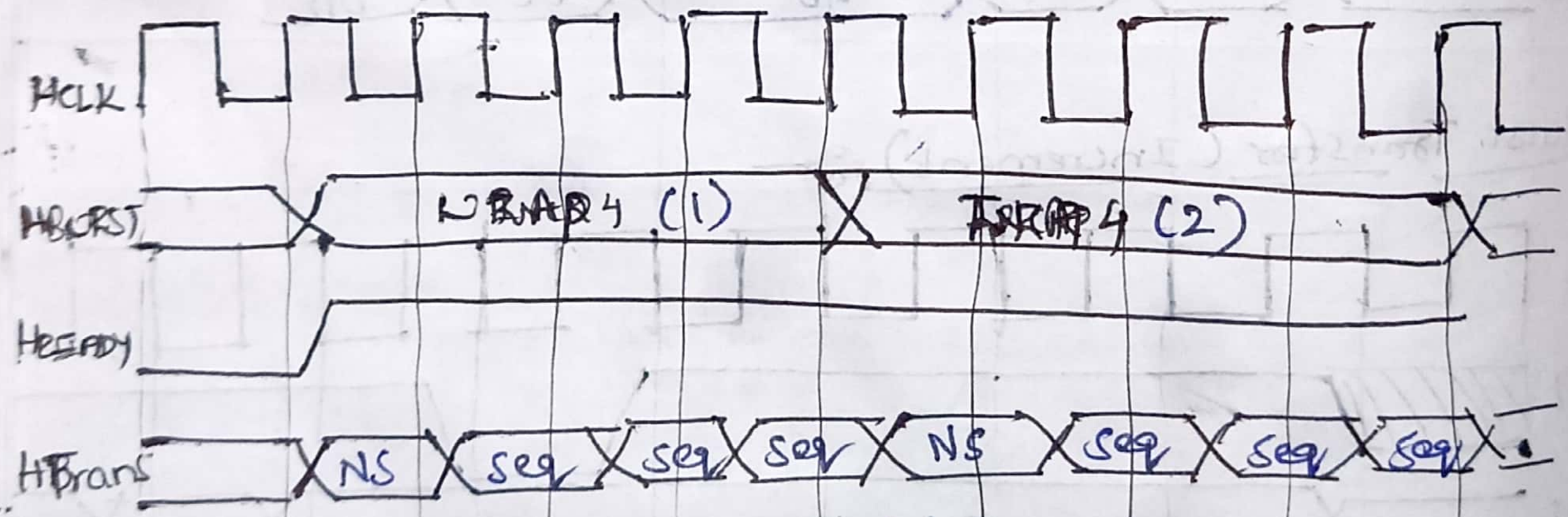
↓

It provides 1) length of the burst } INCR4
2) type of the burst } type: increment-
length: 4-beat

Burst Transfer (Wrapping) :-

→ In wrapping type burst, the address will wrap back to the starting address whenever the address has reached to the wrapping boundary.

→ Let us Consider wrapping 16 from 0x30 to 0x36



→ HBURST → WRAP4 → wrapping type
→ 4-beats (4-transfers)

APB Bus :-

→ APB - Advanced peripheral Bus

→ APB is a simple bus.

→ It has very low overhead with only 4 control signals

→ only one master is allowed.

→ Three states :- IDLE, SETUP and ENABLE

→ APB peripherals (slaves) are non-responsive type, that means the APB slave does not provide response to APB Master.

→ APB slave is non-responsive

→ APB transfer always takes 2 cycles

→ Makes timing easy to design :-

→ Data is always latched b/w the two cycles.

APB Signal Description :-

Signal Name	Description
PCLK Bus clock	This clock will times all bus transfers. Both the LOW phase and HIGH phase of PCLK are used to control transfers.
PRESETn APB Reset	The bus reset signal is active LOW and is used to reset the system.
PENABLE APB strobe	This strobe signal is used to time all accesses on the peripheral Bus. The enable signal is used to indicate the second cycle of an APB transfer. The rising edge of PENABLE occurs in the middle of the APB transfer.
1 PADDR[31:0] 1 APB address Bus	This is the APB address bus, which may be up to 32-bits wide and is driven by the peripheral bus bridge unit.
PWRITE APB transfer direction	When HIGH this signal indicates an APB write access and when LOW a read access.

Signal Name	Description
PRDATA [31:0] APB read data bus	The read data bus is driven by the selected slave during read cycles (when PWRITE is low). The read data bus can be up to 32-bits wide.
PWDATA [31:0] APB write data bus	The write data bus is driven by the peripheral bus bridge unit during write cycles (when PWRITE is High). The write data bus can be upto 32-bits wide.
PSELx APB Select	A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave. This signal indicates that the slave device is selected and a data transfer is required. There is a PSELx signal for each bus slave.

Write transfer :-

→ when PSEL = 0 is called IDLE state.

→ whenever PSEL = 1, then the PWRITE (control signal), PADDR, PWDATA will be available in the same cycle and here PENABLE is low, this is called SETUP State.

→ SETUP State means the state, where the transfer has been started.

→ In the 2nd cycle, the PENABLE goes high, and at the posedge of the clock. PENABLE signal the APB peripherals complete the transfer. This is called as ENABLE State.

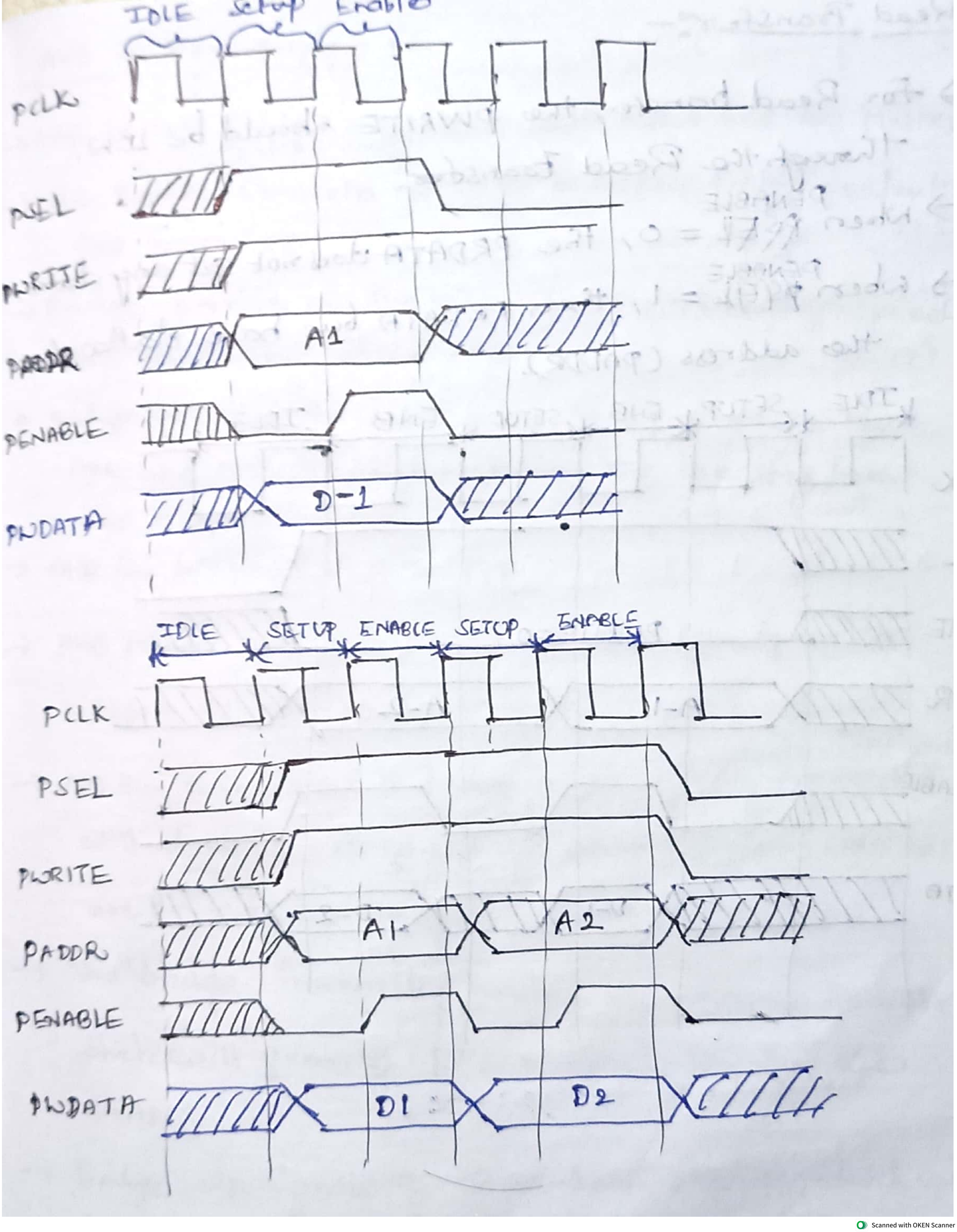
→ In APB for each transfer, it needs 2 clock cycles.

1st cycle → Setup state

2nd cycle → Enable state

→ for write transfer, PWRITE should high throughout the write transfer.

→ after the completion of one transfer, then again PSEL goes low.


IDLE
Setup
Enable
PCLK
PSEL
PWRITE
PADDR
A1
PWDATA
D-1
PENABLE
IDLE
SETUP
ENABLE
SETUP
ENABLE
PCLK
PSEL
PWRITE
PADDR
A1
A2
PENABLE
PWDATA
D1
D2

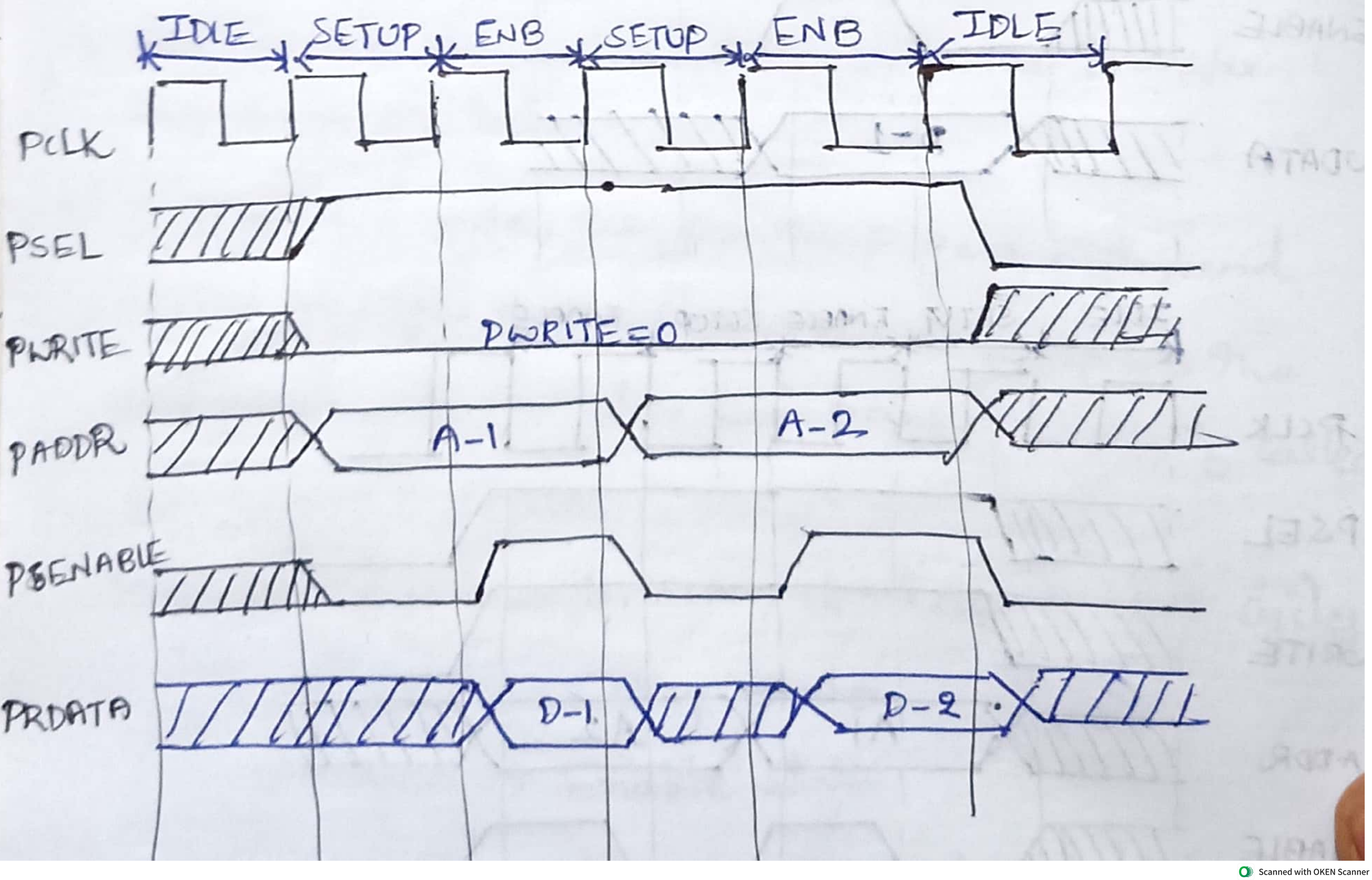
Read Transfer:-

→ For Read transfer, the PWRITE should be LOW.

Through the Read Transfer.

→ When PENABLE = 0, the PRDATA does not get any data.

→ When PENABLE = 1, then PRDATA bus have data from the address (PADDR).



AHB to APB Bridge :-

→ AHB to APB Bridge acts as an AHB slave and APB Master.

→ The Bridge Converts the AHB transfers to equivalent APB transfers.

→ Bridge provides an Interface between the high speed AHB and low speed APB.

→ Bridge will add the wait states to synchronize the APB and AHB transfers during the transfers from AHB to APB

→ AHB has to wait for the APB.

→ AHB Master will drive the all the AHB signals except HRDATA, HREADY. to the bridge

→ APB Bus is connected to bridge on other side, APB Master (Peripherals) will drive the PRDATA and all other signals on APB BUS are driven by the Bridge

→ The bridge internally having an Address Decoder which will generate PSELx signals based on the HADDR.

→ Bridge also Consist of some Local flipflops, which will latch the Current Information driven by the AHB Master.

→ Whenever the wait state has inserted, the AHB Master has to

→ Bridge will holds the same address and the Control Information, for that the Local flipflops are used in bridge.

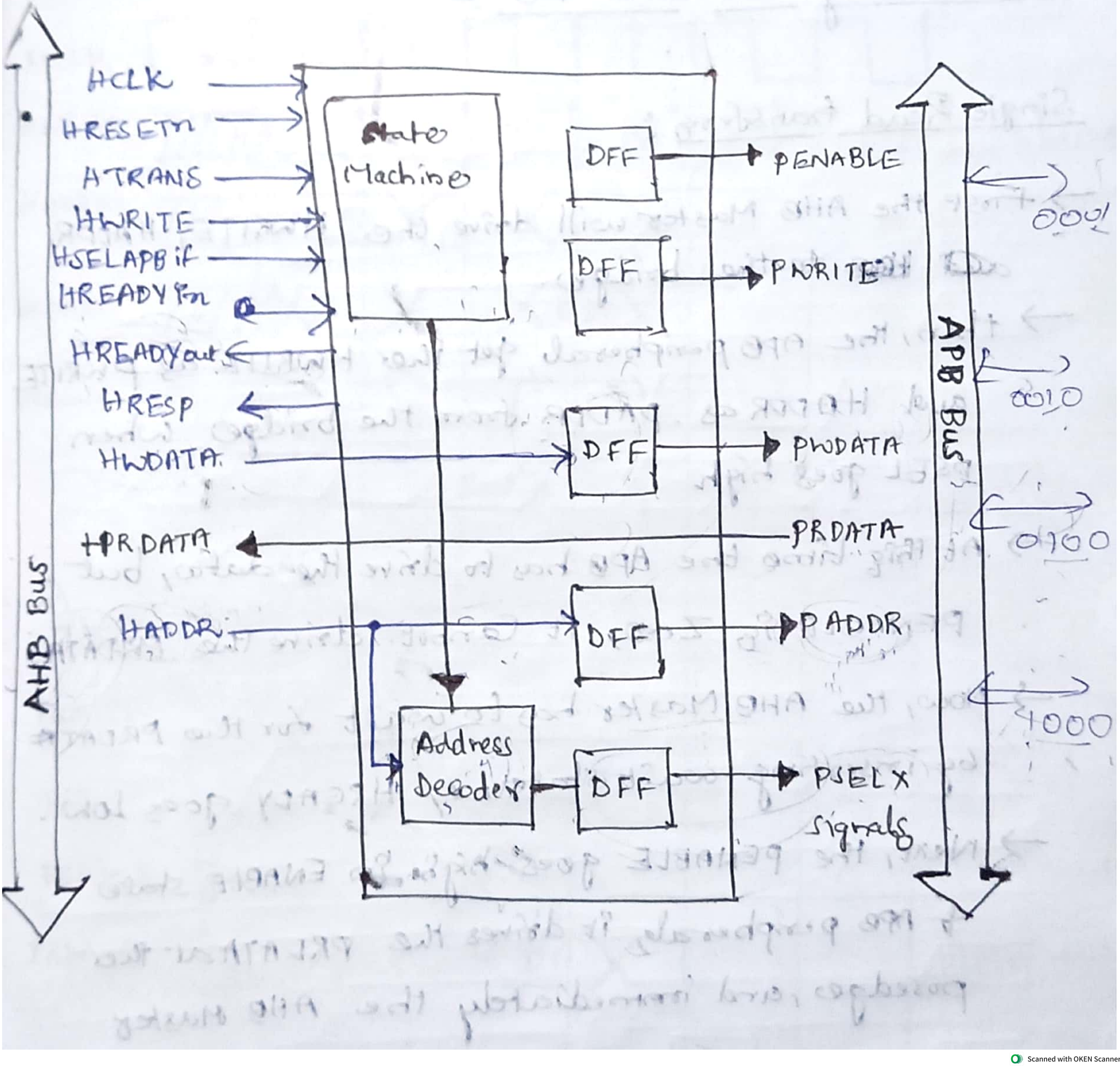
APB Bridge Description :-

The bridge unit Connects system bus transfers into APB transfers and performs the following functions :-

* Latches the address and holds it valid throughout the transfer.
* Bridge decodes the address (HADDR) and generates a peripheral select, PSELx. only one select signal Can be active during a transfer.
* During write transfer bridge will Convert the HWDATA to PWDATA.

  Drives the data onto the APB for a write transfer
* During read transfer bridge will Convert the PRDATA to HRDATA.

  Drives the APB data onto the system bus (AHB) for a read transfer).
* Generates a timing strobe, PENABLE, for the transfer.


HCLK
HRESETn
ATRANS
HWRITE
HSELAPB if
HREADY in
HREADYout
HRESP
HWDATA
HRDATA
HADDR
AHB Bus
State Machine
DFF
DFF
DFF
DFF
DFF
Address Decoder
PENABLE
PWRITE
PWDATA
PRDATA
PADDR
PSELX signals
APB Bus
0001
0010
0100
1000

Interfacing AHB and APB :-

Single Read transfers :-

→ First the AHB Master will drive the HWRITE, HADDR to the bridge.

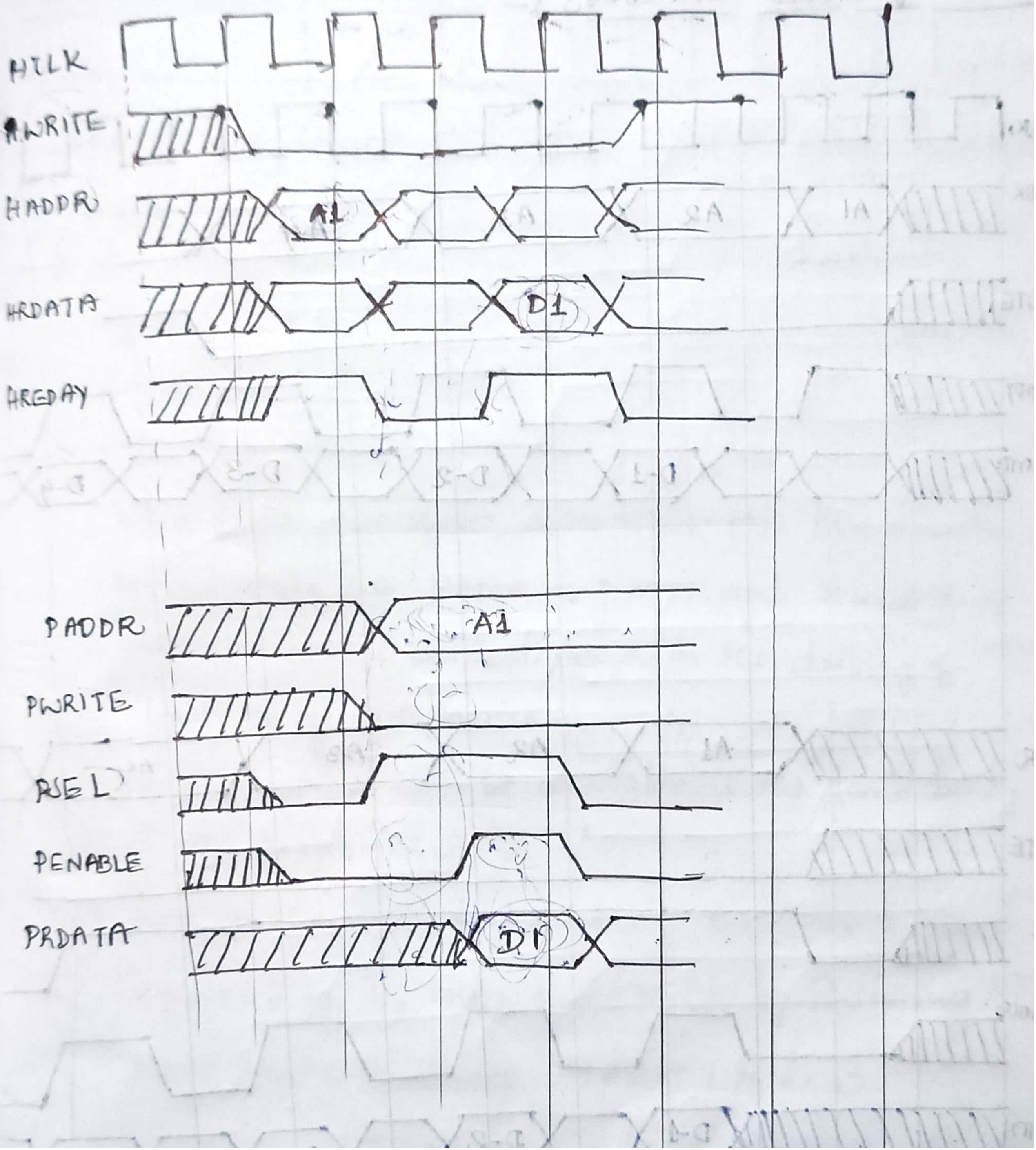
→ then, the APB peripheral get the HWRITE as PWRITE and HADDR as PADDR from the bridge when PSEL goes high

→ At this time the APB has to drive the data, but PENABLE is Zero, it cannot drive the PRDATA.

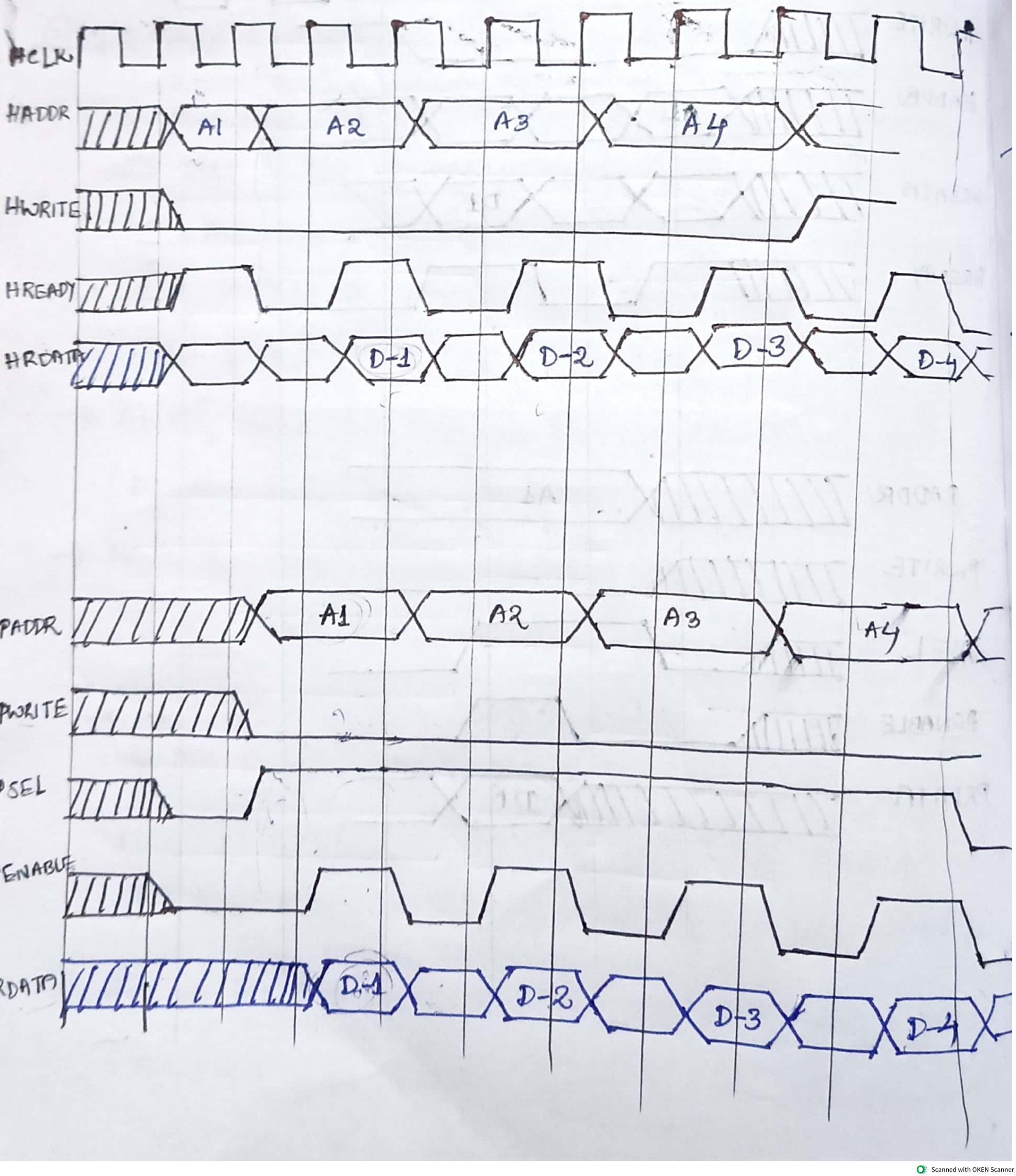
→ Now, the AHB Master has to wait for the PRDATA by inserting wait states i.e, HREADY goes LOW.

→ Next, the PENABLE goes high in ENABLE state of APB peripherals, it drives the PRDATA at the posedge, and immediately the AHB Master get the PRDATA as HRDATA. by HREADY as high.

→ For a Read transfer, the ControlSignals PWRITE and HWRITE should be low throughout the read transfer.


HCLK
HWRITE
HADDR
A1
HRDATA
D1
HREADY
PADDR
A1
PWRITE
PSEL
PENABLE
PRDATA
D1

Burst of Read transfers :-



Simple write transfer :-

→ For write transfer, the respective Control signals PWRITE and HWRITE should go high throughout the write transfer.

→ In write transfer, the Data will be driving by the AHB Master and APB peripherals has to sample that driven data.

→ The AHB Master will drive the HADDR and the HWDATA in next cycle, after that the APB peripheral will sample the HADDR as PADDR and HWDATA as PWDATA at the same time after the driving of HWDATA by AHB Master.

→ PADDR, PWDATA will be available at the same time at APB peripheral sides.

→ AHB Master has to wait for transfer to be completed by the APB peripherals by inserting wait states making HREADY to low.

→ Whenever PENABLE goes high, the APB peripheral samples the HWDATA as actual PWDATA and completes the transfer.

AP DP

HCLK

HADDR A1

HWRITE

HWDATA D1

HREADY

PADDR A1

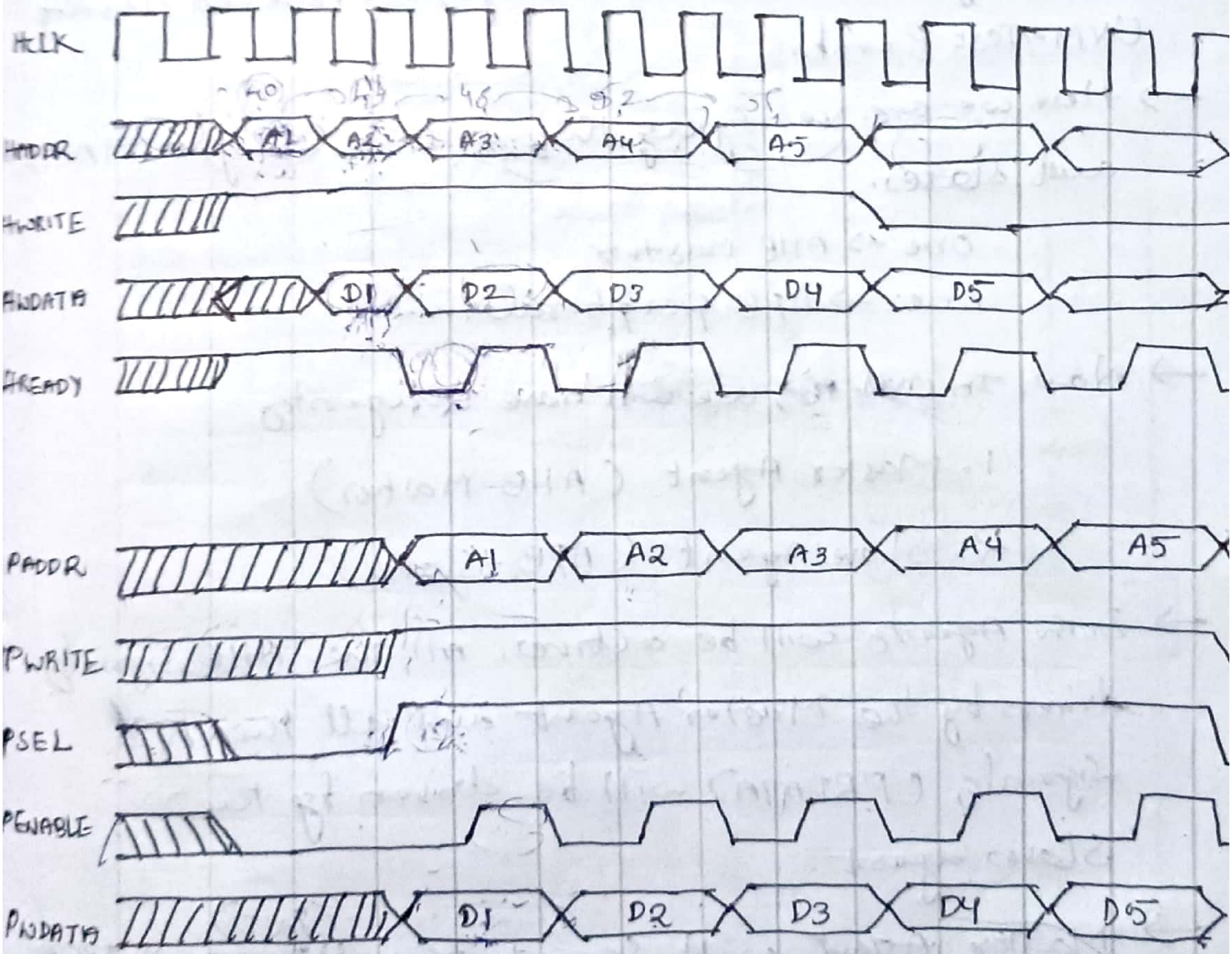
PWRITE

PSEL

PENABLE

PWDATA D-1

Burst & write transfers :-



TB Architecture :-

→ To verify this AHB2APB-Bridge, we have to develop UVM-Test Bench.

→ Here we are verifying this with a Single MASTER and slave.

One → AHB Master

One → APB peripheral

→ Now, In UVM-TB, we will have 2 agents

1. Master Agent (AHB-Master)
2. Slave Agent (APB-slave)

→ Both Agents will be active. All the AHB signals driven by the Master Agent and all the APB signals (PRDATA) will be driven by the Slave agent.

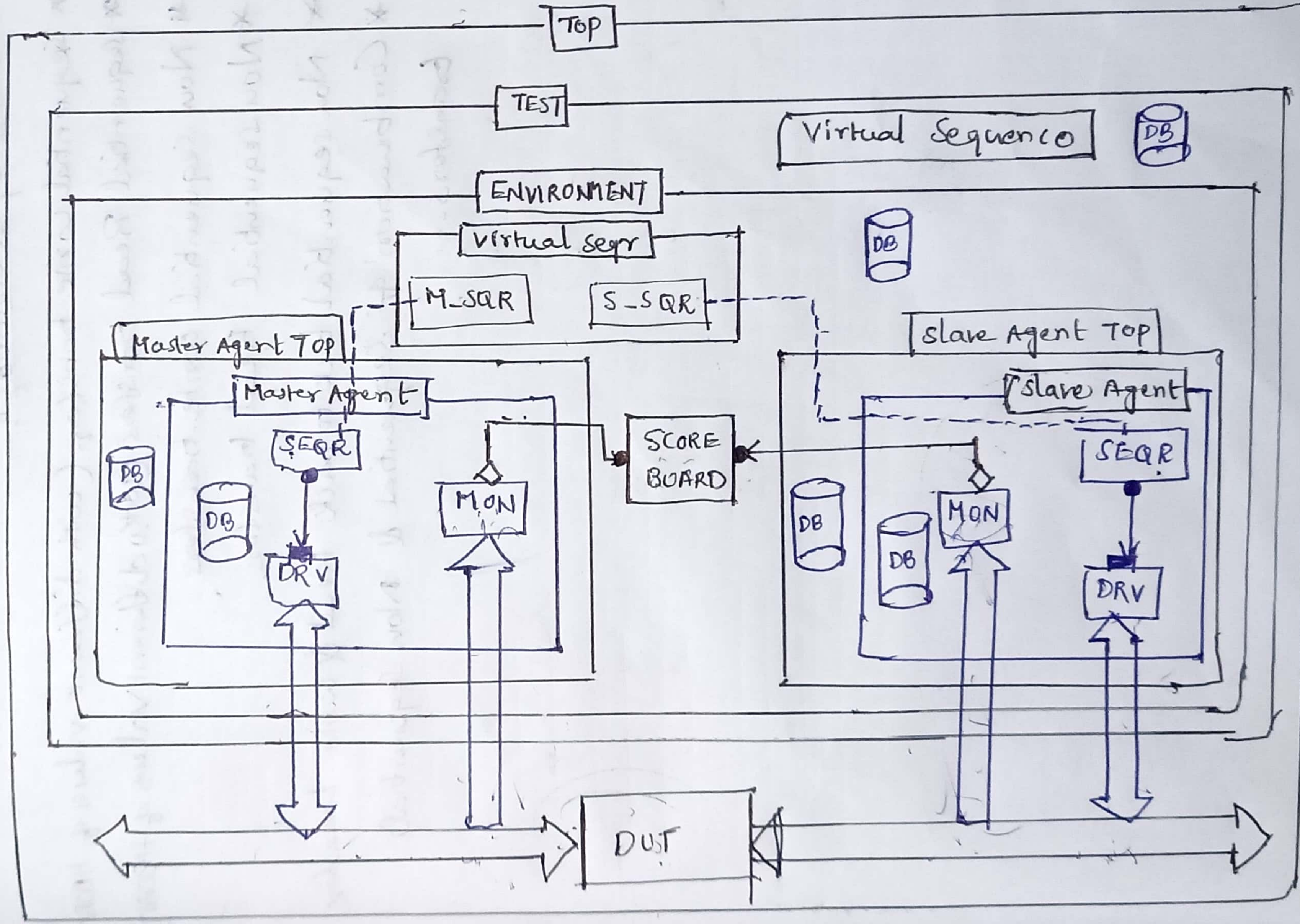
→ Master Agent will sample the HRDATA, HWDATA (DUT)

→ Slave Agent will sample all the APB signals like PRDATA, PWDATA (DUT)

→ Now in Scoreboard, we will Implement the Comparision logic to Compare the

(i) PWDATA with HWDATA

(ii) PRDATA with HRDATA.


TOP
TEST
Virtual Sequenco
DB
ENVIRONMENT
Virtual seqr
M_SQR
S_SQR
DB
Master Agent TOP
Master Agent
SEQR
DB
DB
DRV
MON
SCORE BOARD
Slave Agent TOP
Slave Agent
SEQR
DB
DB
MON
DRV
DUT

Test Cases / Scenarios :-

* Sequential write transfers (with different values of HBURST)
* Sequential Read transfers (with different values of HBURST)
* Non sequential write transfers
* Non sequential Read transfers.
* Non sequential back to back Read & write transfers.
* Combination of sequential & Non sequential transfers.