DMA – Direct Memory Access

- The Direct Memory Access (DMA) is an I/O Technique that provides direct access to the main memory while the CPU is temporarily disabled to speed up the memory operations.
- This process is managed by a chip known as DMA Controller (DMAC).
- I/O devices are connected to System Bus via a special interface circuit called "DMA Controller".
- In DMA both CPU and DMA Controller have access to the main memory via shared system bus having – Address, Data and Control lines.
- DMA allows I/O devices to transfer data directly to or from the main memory without CPU intervention (or simply bypassing the CPU from the path)
- DMA Controller takes over the buses to allow the transfer directly from the main memory to or from I/O devices.

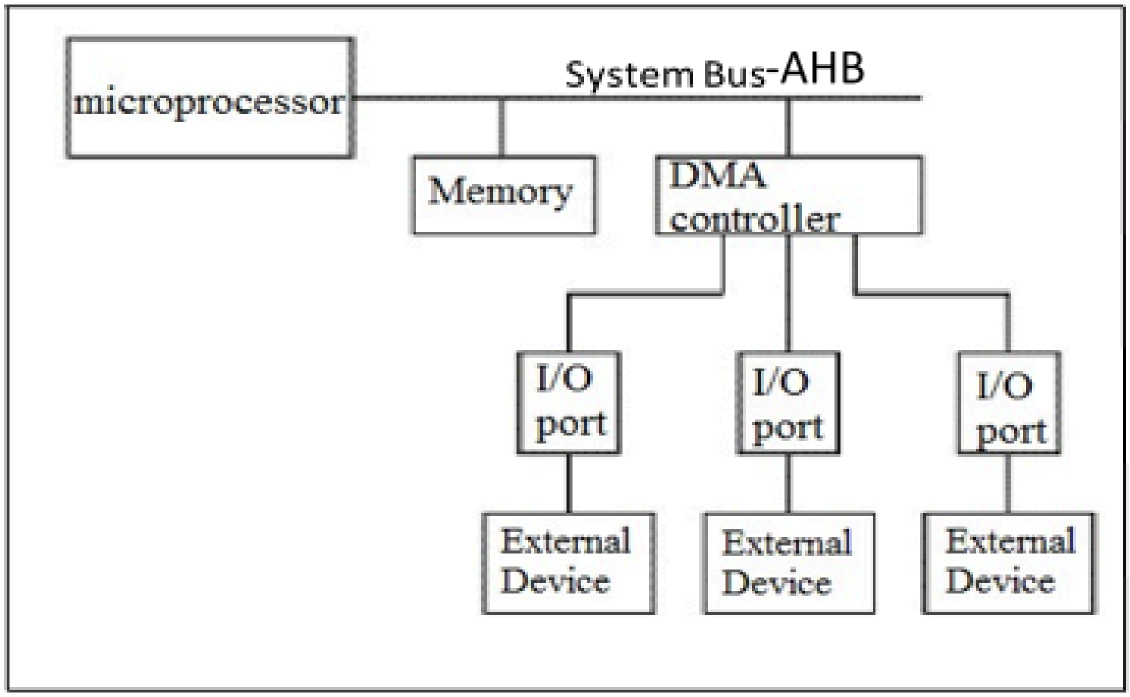


Fig: A Simple Block Diagram of DMA

<u>DMA Working</u>

- When I/O wants to transfer data with main Memory, I/O device sends DMA Request to DMA Controller (DMAC).
- DMAC sends BR (Bus Request) Signal to CPU for requesting the control of Buses.
- DMA waits until CPU sends BG (Bus Grant) Signal to DMA.
- CPU releases the Control of Buses and places Address BUS (ABUS), Data BUS (DBUS), Read (RD) and Write (WR) lines into a High-Impedance state.
- CPU activates the BG (i.e., Bus Grant) signal and becomes in Idle state (disabled).
- DMA Controller takes Control of Buses to conduct direct memory transfer without CPU Intervention.
- When DMA transfer terminates, it disables the BR (i.e., Bus Request) line.
- The CPU disables the BG (Bus Grant).
- After that CPU takes the Control of BUSES and returns to its normal operation.

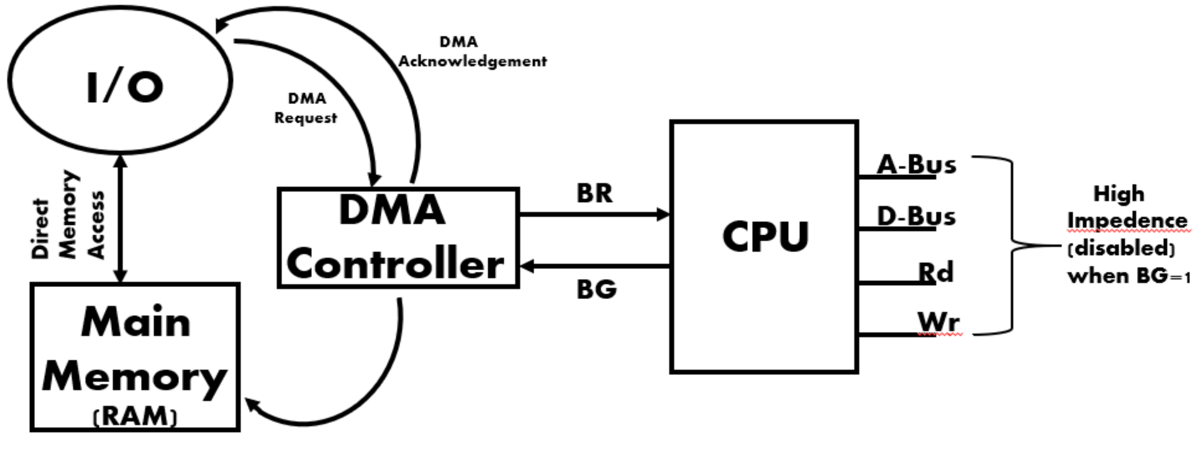
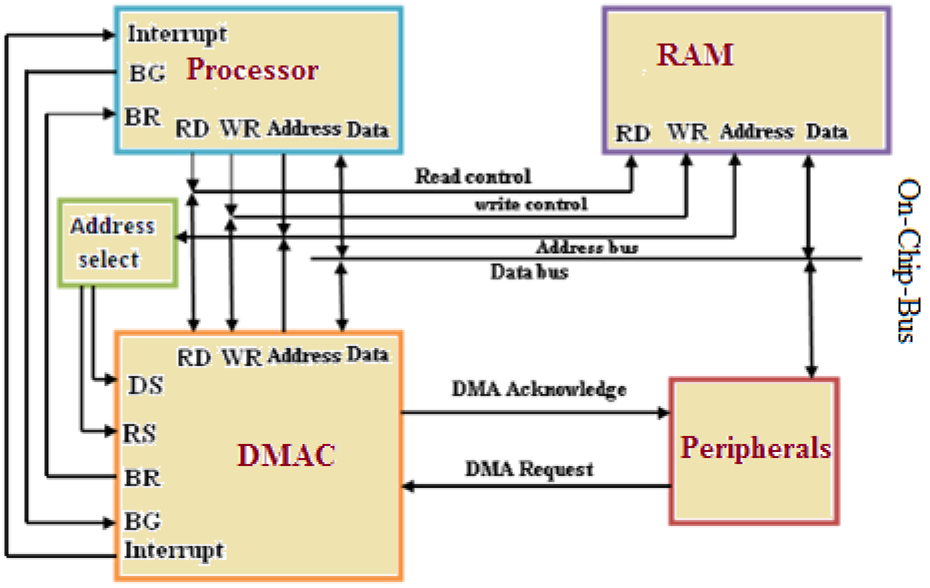


Fig: DMA Transfer Block Diagram

DMA Transfer

- The DMA Controller communicates with the CPU via data bus and control lines.



- The DMA request line is used to request a DMA Transfer.
- The Registers in DMA are selected by the CPU through the Address Bus by enabling DS (DMA Select) and RS (Register Select).
- Read (RD) and Write (WR) lines are Bidirectional.
- When BG (Bus Grant) input is 0 (i.e., BG = 0), the CPU can communicate with DMA Registers, through the Data Bus to read from or write to the DMA Registers.
- When BG = 1, the CPU has to release the Buses and the DMA can communicate directly with the main memory by specifying an address in the Address Bus and activating RD and WR control.
- DMA communicates with the external I/O devices through the request and acknowledgement lines by the Handshaking Procedure.

- The DMA acknowledgement line is set when the system is ready to initiate data transfer.
- The data bus is used to transfer data between I/O devices and memory.
- When the last word of data in the DMA transfer is transferred, the DMA controller informs the termination of transfer to CPU by means of Interrupt lines.

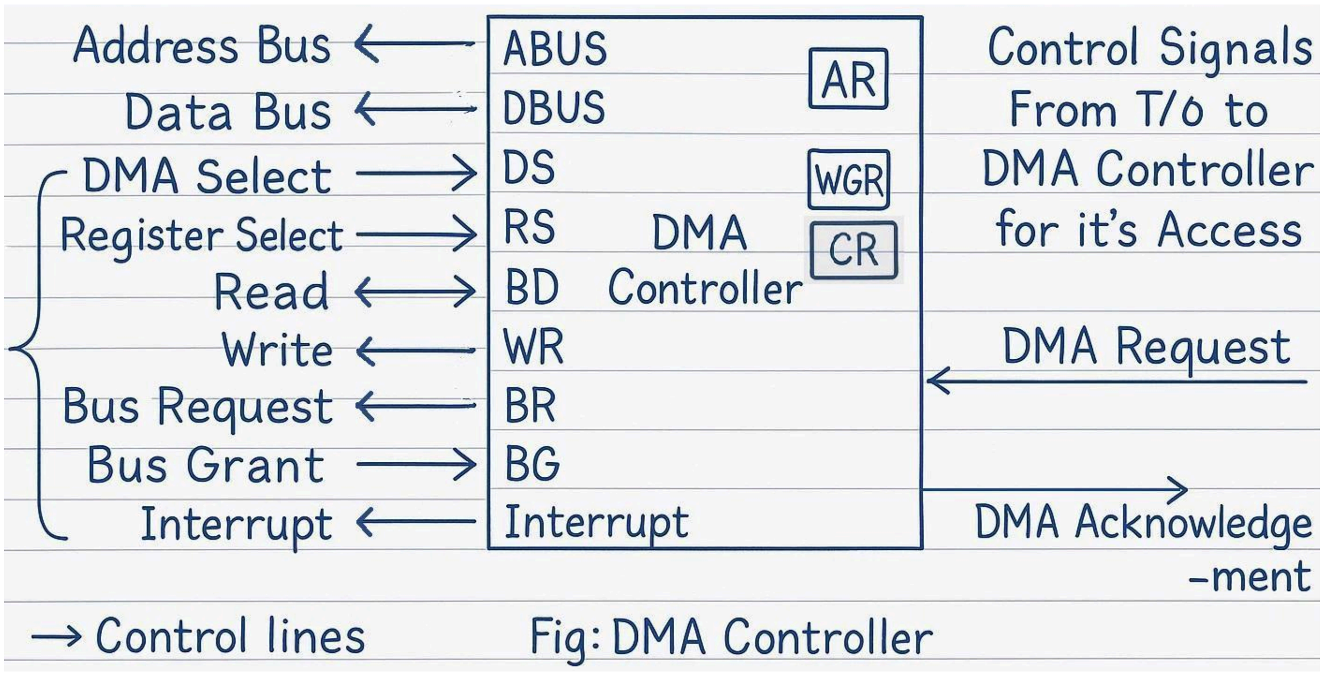


Fig: DMA Controller

Modes of DMA Transfer

Mode-1:
Burst Mode -

- In this mode Burst of data (entire data or burst of block containing data) is transferred before CPU takes control of the buses back from DMAC.
- This is the quickest mode of DMA Transfer since at once a huge amount of data is being transferred.
- Since at once only the huge amount of data is being transferred so time will be saved in huge amount.

Pros:

- Fastest mode of DMA Transfer

Cons:

- Less user friendly because during the DMA transfer CPU will be blocked.

Mode-2:
Cycle Stealing Mode -

- Slow IO device will take some time to prepare data (or word) and within that time CPU keeps the control of the buses.
- Once the data or the word is ready CPU give back control of system buses to DMAC for 1-cycle in which the prepared word is transferred to memory.
- As compared to Burst mode this mode is little bit slowest since it requires little bit of time which is actually consumed by IO device while preparing the data.

Pros:

- Most Efficient way for DMA Transfer.
- CPU won't be blocked entire time.

Cons:

- Rate of DMA Transfer will be less.

Mode-3:
Transparent Mode -

- Whenever CPU does not require the system buses then only control of buses will be given to DMAC.
- In this mode, CPU will not be blocked due to DMA at all.
- This is the slowest mode of DMA Transfer since DMAC has to wait might be for so long time to just even get the access of system buses from the CPU itself.
- Hence due to which less amount of data will be transferred.

Pros:

- CPU will not be blocked at all.

Cons:

- Slowest DMA transfer rate.

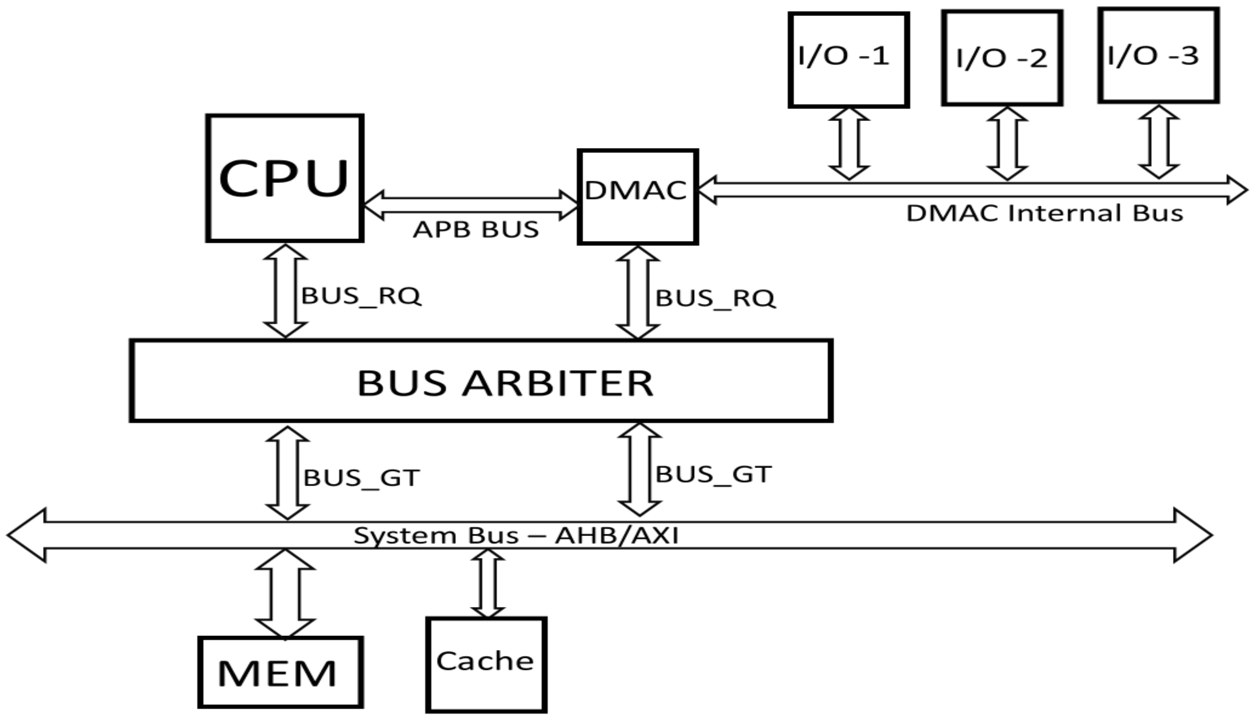
Burst mode, cycle stealing mode, and transparent mode are three common DMA transfer techniques, each with different impacts on CPU access. In burst mode, the DMA controller transfers a large block of data in one go, temporarily blocking the CPU, cycle stealing allows the DMA to transfer one data word per bus cycle, sharing the bus with the CPU, while transparent mode transfers data only when the CPU is idle, causing the least interference.

For this project, burst mode is chosen due to its speed and efficiency in handling high throughput memory transfers, making it ideal for a 3 channel DMA controller in a single core RISC-V SoC.

Microarchitectural Planning

Parameter	Choice / Value	Explanation
System Bus Protocol	AHB	Ensures compatibility with the RISC-V SoC using standard burst-based transfers.
DMA Master Interface	AHB Master	Allows the DMA to read/write data directly from/to memory over AHB bus.
DMA Slave Interface	AHB Slave	Enables CPU to configure DMA registers through memory-mapped access.
Number of DMA Channels	3 Channels	Supports concurrent requests from 3 I/O devices for data transfer.
Arbitration Scheme	Round-Robin	Fairly cycles through active channel requests to prevent starvation.
Transfer Mode Supported	Burst Mode Only	Offers high-speed block data transfers with fewer bus handshakes.
Register Set	Control, Status, Address, Count	Used to configure each channel, monitor transfer, and set source/destination.
Data Bus Width	32-bit	Matches the SoC word width for optimal compatibility and throughput.
Addressing Capability	32-bit Address Bus	Supports wide address range for modern memory/peripheral mapping.
Interrupt Support	Optional	DMA can optionally raise an interrupt to CPU upon transfer completion.
Priority Handling	Equal priority (Round-Robin)	No fixed priority; all channels treated equally during arbitration.
Burst Length Support	Configurable	Allows variable-length burst transfers as per control register settings.
Transfer Types	Memory-to-Peripheral, Peripheral-to-Memory	Handles typical DMA use cases for data movement.
Status Monitoring	Done/Error flags per channel	Provides transfer success/failure indication to the CPU.
Clock & Reset	Single system clock, active-low reset	Syncs with SoC for reliable operation and control.

Top Level Architecture



Registers and Configuration Map

1. CTRL_GLOBAL (Offset 0x0000, 32 bits, RW)

This is the DMA controller's global control register. You write here to turn the DMA engine and burst mode on or off, and to clear interrupts.

Bit(s)	Name	Description
0	DMA_EN	When you write a 1 here, the entire DMA engine is enabled—you can start channels. Writing 0 disables all channels.
1	BURST_EN	If set to 1, channels are allowed to perform multi-beat burst transfers (back-to-back data beats). If 0, transfers use single beats (one data word per cycle).
2	INT_GLOBAL_CLR	Write a 1 here to clear all pending channel interrupt flags at once (the INT_STATUS bits).
Bits 31:3	*Reserved*	Must be written as 0 (unused in this design)

Example: To turn on DMA with bursts enabled and clear any old interrupts:

```
WRITE CTRL_GLOBAL = (1 << DMA_EN)
                                    | (1 << BURST_EN)  | (1 << INT_GLOBAL_CLR);
```

2. STATUS_GLOBAL (Offset 0x0004, 32 bits, RO)

This read-only register lets your firmware check overall DMA activity.

Bit(s)	Name	Description
0	DMA_BUSY	1 if any of the three channels is actively transferring data, 0 if the DMA is idle.
3:1	NUM_ACTIVE	A 3-bit count (0–3) of how many channels currently have their CH_EN set and haven't yet finished.
Bits 31:4	*Reserved*	Always read as 0.

Example: Poll STATUS_GLOBAL[DMA_BUSY] until it becomes 0 to know that *all* DMA transfers have completed.

3. INT_ENABLE (Offset 0x0008, 32 bits, RW)

This register controls which channels can generate an interrupt when they finish or hit an error.

Bit	Name	Description
0	CH0_INT_EN	Enable interrupt for channel 0 when it completes or errors.
1	CH1_INT_EN	Enable interrupt for channel 1.
2	CH2_INT_EN	Enable interrupt for channel 2.
3–31	*Reserved*	Must be 0.

Example: To get an interrupt from channel 1 only:

WRITE INT_ENABLE = (1 << CH1_INT_EN);

4. INT_STATUS (Offset 0x000C, 32 bits, RW¹)

This register shows which channels have signaled completion or error. You clear individual bits by writing a 1 to them.

Bit	Name	Description
0	CH0_INT	Read 1 if channel 0 has completed or encountered an error; write 1 here to clear it.
1	CH1_INT	Same for channel 1.
2	CH2_INT	Same for channel 2.
3–31	*Reserved*	Must read as 0, ignored on write.

Example: In your interrupt handler, clear channel 0’s flag by:

WRITE INT_STATUS = (1 << CH0_INT);

5. Per-Channel Registers (Base + 0x0100 + ch×0x10)

For each channel (ch = 0, 1, 2), you have the following registers:

a. CHx_SRC_ADDR (Offset +0x00, RW, 32 bits)

- Holds the starting source address in memory (or peripheral register) from which to read data.

b. CHx_DST_ADDR (Offset +0x04, RW, 32 bits)

- Holds the starting destination address where data will be written.

c. CHx_TRANSFER_COUNT (Offset +0x08, RW, 32 bits)

- Specifies how many 32-bit words (or bytes, if you prefer) to transfer.

d. CHx_CTRL (Offset +0x0C, RW, 32 bits)

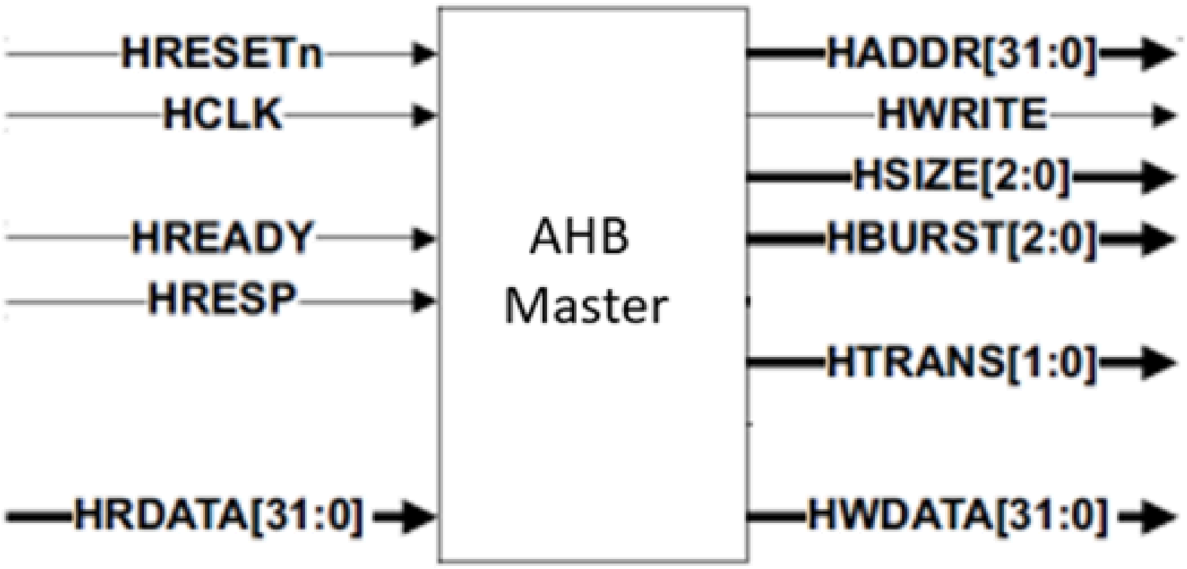
Bit	Name	Description
0	CH_EN	When you write 1, channel ch is *enabled* and will start transferring when granted the bus.
1	DIR	0 = memory→peripheral; 1 = peripheral→memory transfer.
2	BURST	1 = allow multi-beat burst transfers; 0 = single-beat only.
3–31	*Reserved*	Must be 0.

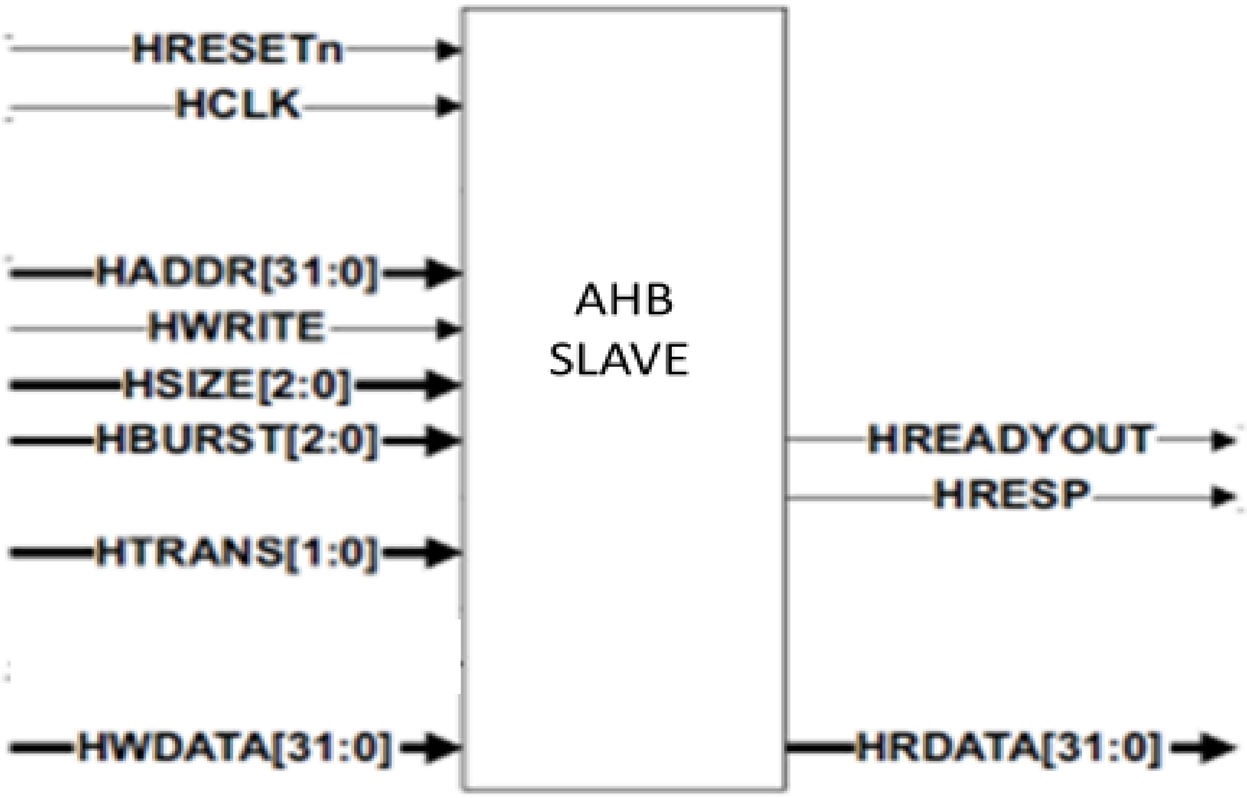
e. CHx_STATUS (Offset +0x10, RO, 32 bits)

Bit	Name	Description
0	BUSY	1 while channel ch is actively transferring.
1	DONE	1 when transfer completes; cleared by writing INT_STATUS.
2	ERROR	1 if a bus error or transfer-count underflow/overflow occurs.
3–31	*Reserved*	Always reads 0.

AHB Protocol

1. Block Diagram





2. Signal Description

Signal	Master/Slave	Bits	Description (and values)
HCLK	Global	1	System clock for all AHB logic (rise-edge driven).
HRESET	Global	1	Active-high synchronous reset.
HADDR	Master	[ADDR_WIDTH-1:0]	Address bus for current transfer. Aligned per HSIZE.
HTRANS	Master	2	Transfer type: • 00 = IDLE • 01 = BUSY • 10 = NONSEQ (first beat) • 11 = SEQ (subsequent beats)
HSIZE	Master	3	Size of transfer: • 000=8 bits • 001=16 bits • 010=32 bits • 011=64 bits • 100=128 bits • 101=256 bits
HBURST	Master	3	Burst type: • 000 SINGLE • 001 INCR (unspecified length) • 011 INCR4 • 101 INCR8 • 111 INCR16
HWRITE	Master	1	Direction: 0=read, 1=write.
HWDATA	Master	[DATA_WIDTH-1:0]	Write data bus. Driven in write transfers.
HREADY	Master (input)	1	Indicates slave ready. If 0 → insert wait states.
HRDATA	Slave	[DATA_WIDTH-1:0]	Read data bus returned to master.
HRESP	Slave	1	Transfer response: 0=OKAY, 1=ERROR.

Research paper links

1. Design and Implementation of a Direct Memory Access Controller for Embedded Applications

- **Authors**: Mohammed Altaf Ahmed, Abdullah Aljumah, M. Gulam Ahmad
- **Publication**: International Journal of Technology, 2019

(PDF) Design and Implementation of a Direct Memory Access Controller for Embedded Applications

2. A Low-Area Direct Memory Access Controller Architecture for a RISC-V Based Low-Power Microcontroller

- **Authors**: Hanssel Morales, Ckristian Duran, Elkim Roa
- **Publication**: 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS)

A Low-Area Direct Memory Access Controller Architecture for a RISC-V Based Low-Power Microcontroller

3. Design and Implementation of an Advanced Direct Memory Access Controller on Advanced Microprocessor Bus Architecture

- **Author**: Ravitesh Mishra
- **Publication**: Journal of Emerging Technologies and Innovative Research (JETIR), 2019

JETIR1907T73.pdf

4. A High-performance, Energy-efficient Modular DMA Engine Architecture

- **Authors**: Thomas Benz, Michael Rogenmoser, Paul Scheffler, Samuel Riedel, Alessandro Ottaviano, Andreas Kurth, Torsten Hoefler, Luca Benini

2305.05240