MIPSfpga

AHB-Lite interface and MIPSfpga peripheral devices connection

Stanislav Zhelnio
github.com/zhelnio
2018

Thanks to

- David Harris and Sarah Harris, the authors of the great book “Digital Design and Computer Architecture”
- The team of the “Digital Design and Computer Architecture” book translators
- MIPS – for MIPS Academic Community and MIPSfpga source code and documents
- ARM – for AHB-Lite specification, good documents, presentations and youtube channel
- Yuri Panchul - Senior Hardware Design Engineer at Imagination Technologies and MIPS

What is ARM AMBA Specification

- **A**dvanced **M**icrocontroller **B**us **A**rchitecture
- Open standard (no License)
- Widely used in SoC
- Specifications:
  AMBA 5, AMBA 4, AMBA 3, AMBA 2
- Each of them describe set of Interfaces:
  AHB-Lite, APB, AXI ...

Acronyms

- AXI - **A**dvanced **E**xtensible **I**nterface
- ACE - **A**XI **C**oherency **E**xtensions
- AHB - **A**dvanced **H**igh-performance **B**us
- CHI - **C**oherent **H**ub **I**nterface
- APB - **A**dvanced **P**eripheral **B**us
- ATB - **A**dvanced **T**race **B**us
- ASB - **A**dvanced **S**ystem **B**us

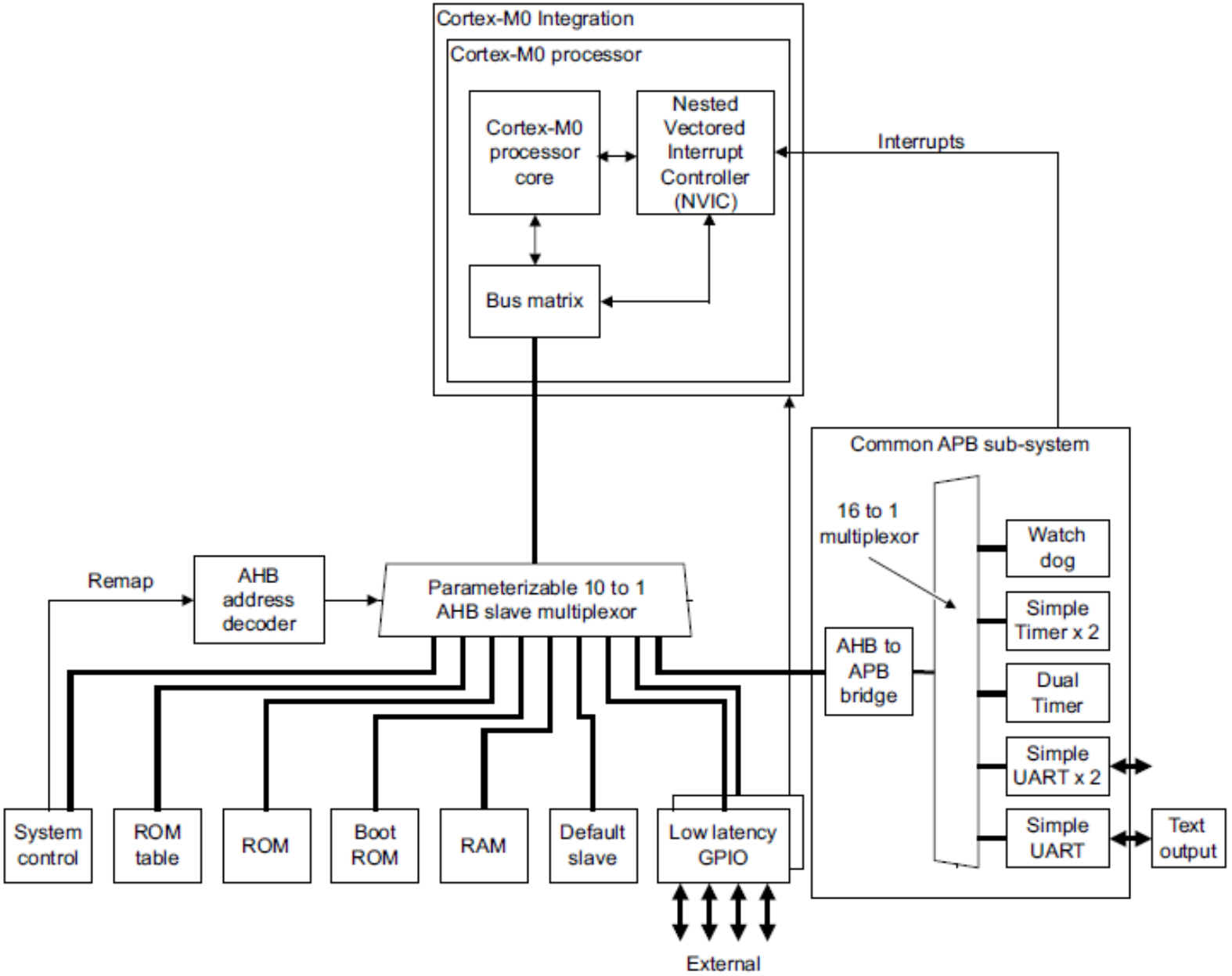
- version numbers: AXI3, AXI4 ...
- simplified versions: AHB-Lite, AXI4-Lite …

Evolution of AHB и AHB-Lite

- 1999, AMBA 2 –AHB Interface
- 2006, AMBA 3 –AHB-Lite Interface :
  - Subset of AHB
  - Reduced interconnect logic
  - Master – Slave architecture
  - Single Master in most of the systems
  - Multiple Master is possible (multi layer interconnect)

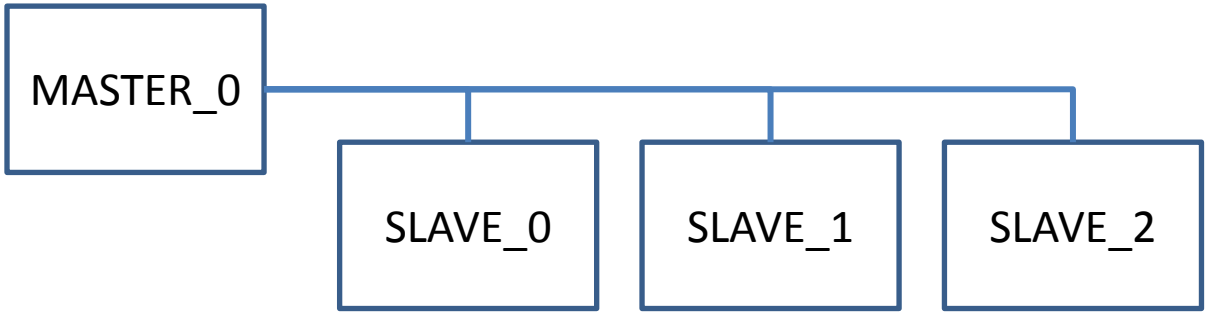
- 2015, AMBA 5 – Interfaces AHB5 и AHB-Lite

AHB-Lite in SoC Architecture



AMBA 3 AHB-Lite

- Typical connection:



- Single Master
- Multiple Simple Slaves
- No arbitration

Transactions

- Master (CPU) request:
  - Single Read (Read, Register Read)
  - Single Write (Write, Register Write)
  - Burst Read
  - Burst Write
- Slave(peripheral) reply:
  - Can make master wait
  - Can give error response

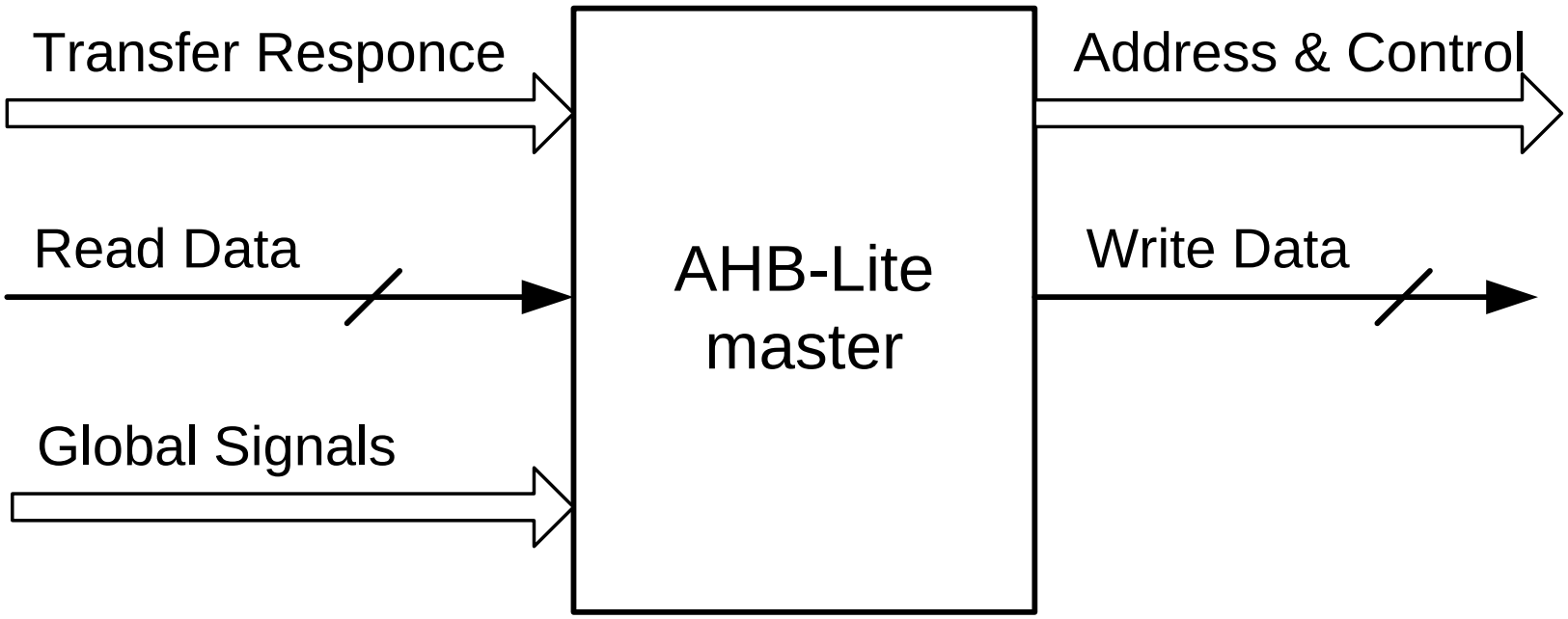
AHB-Lite Features

- Single Clock Edge operation
- Uni-directional busses
  no tri-state
- Pipelined Operation

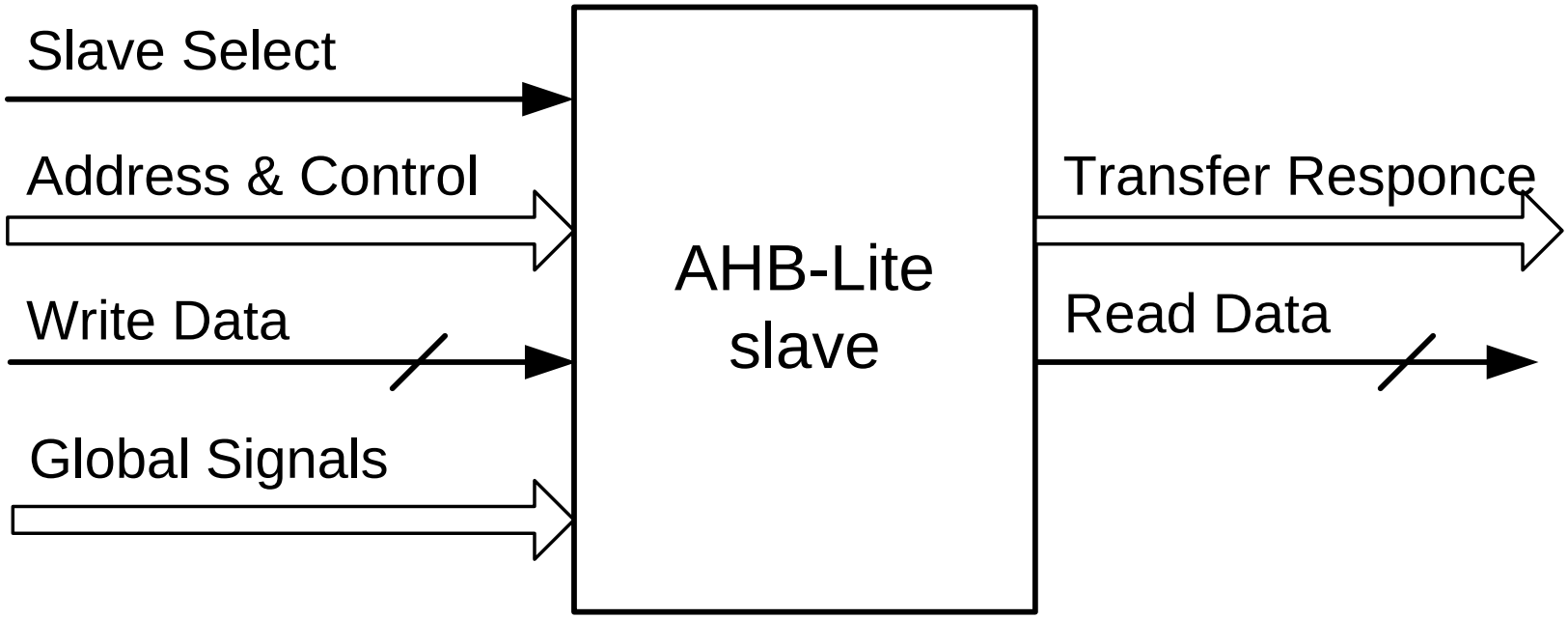
AHB-Lite Components

- Master device (master, CPU)
- Slave devices (slave, peripherals)
- Bus interconnect (bus logic):
  - Address Decoder
  - Multiplexor

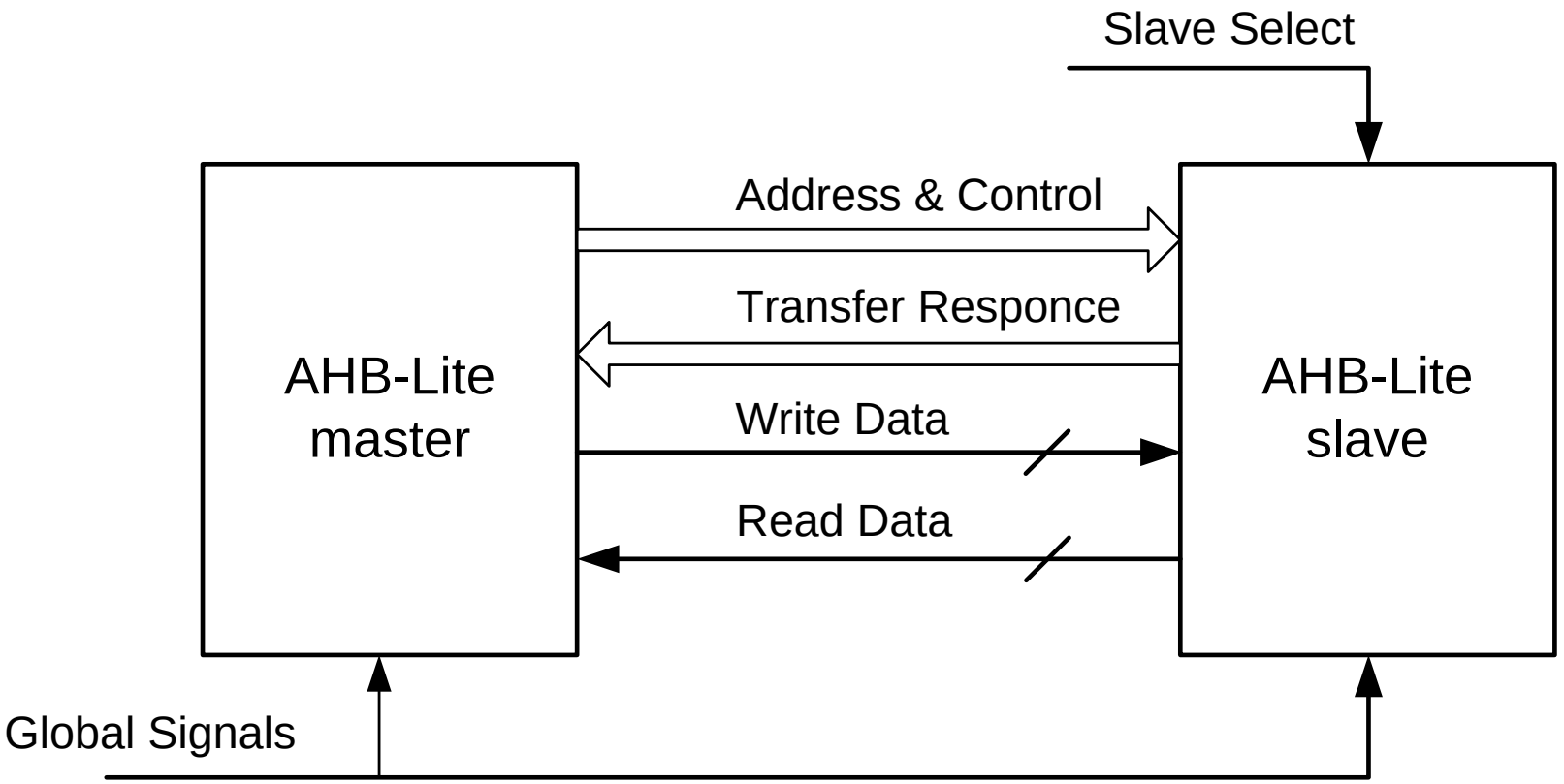
AHB-Lite Master



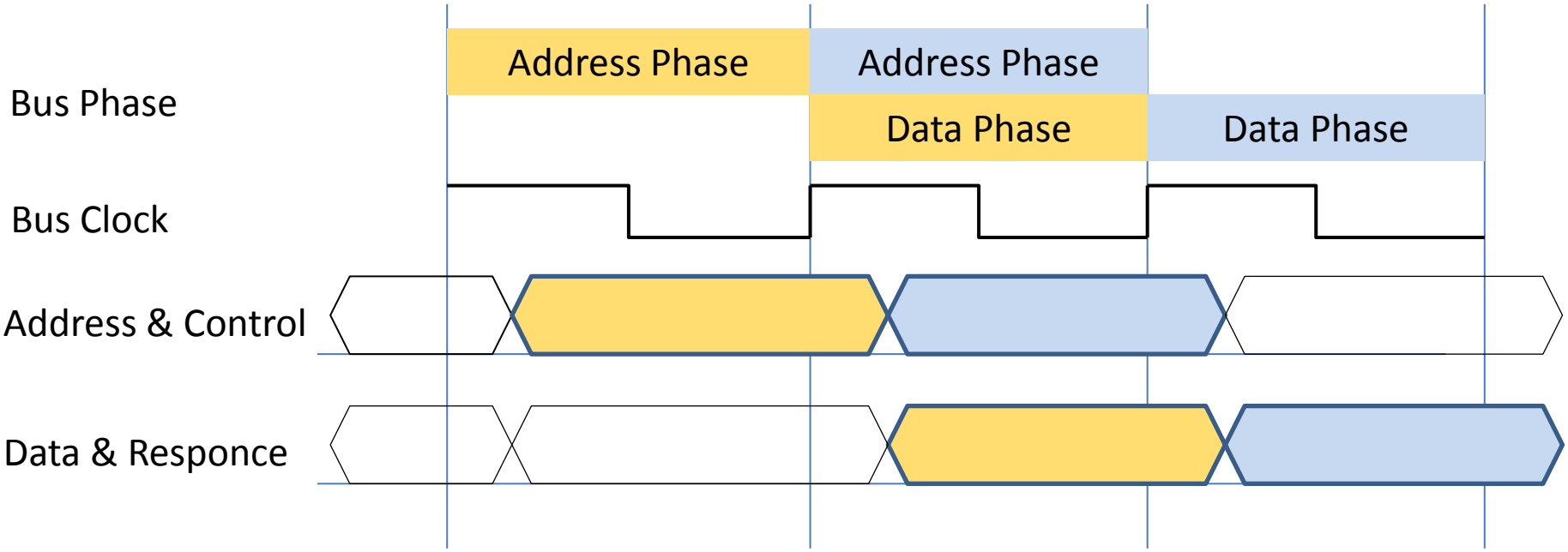
AHB-Lite Slave



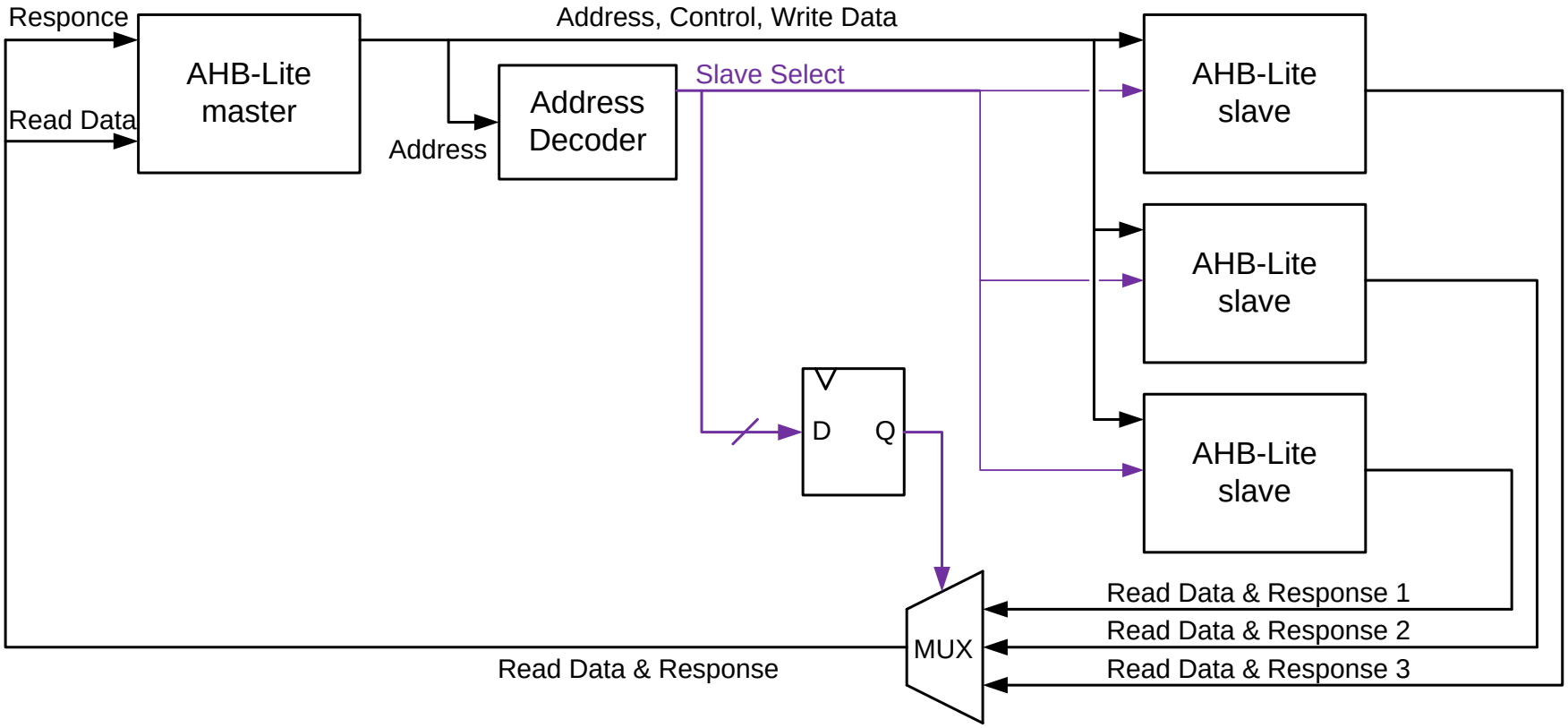
AHB-Lite Master & Slave



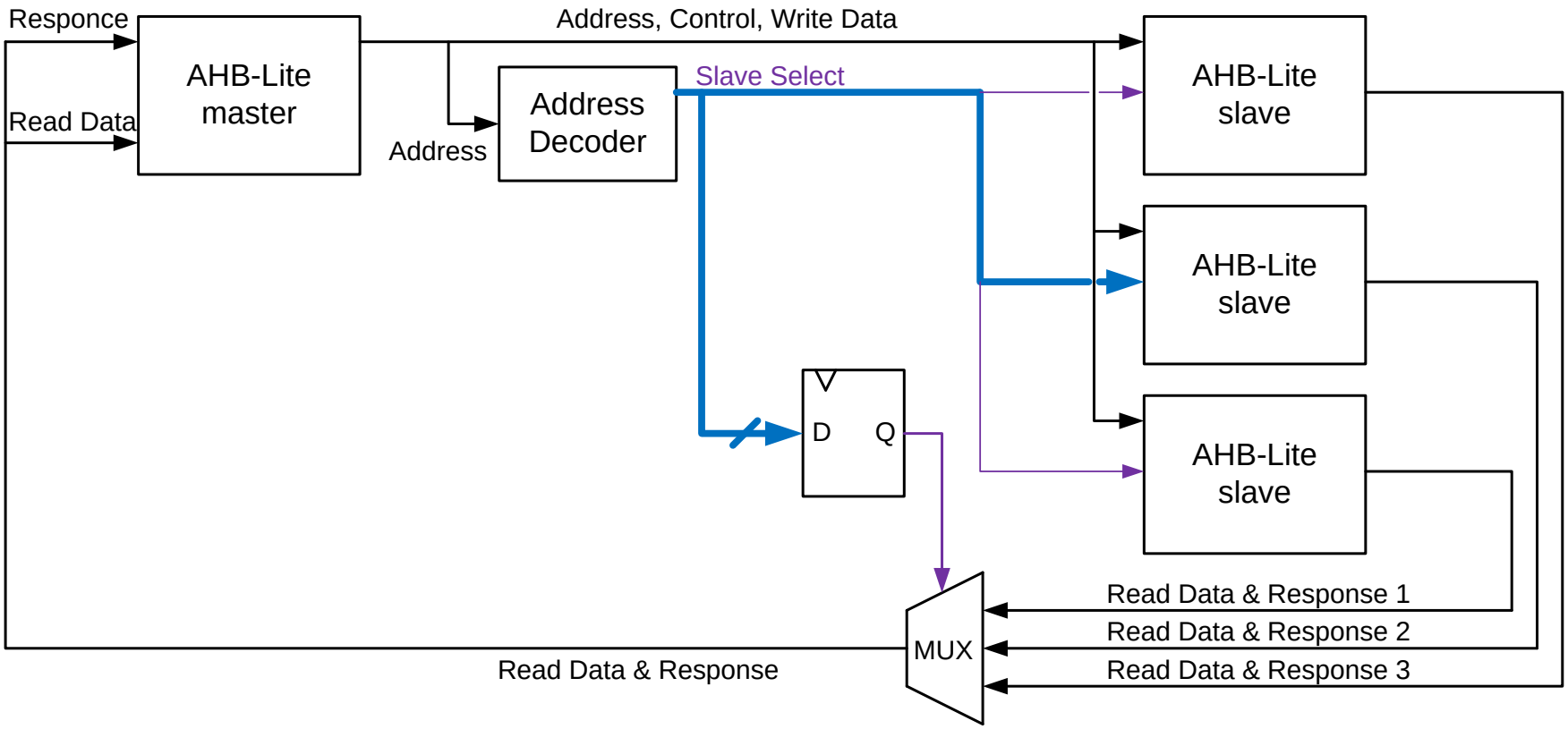
Pipelined Transactions



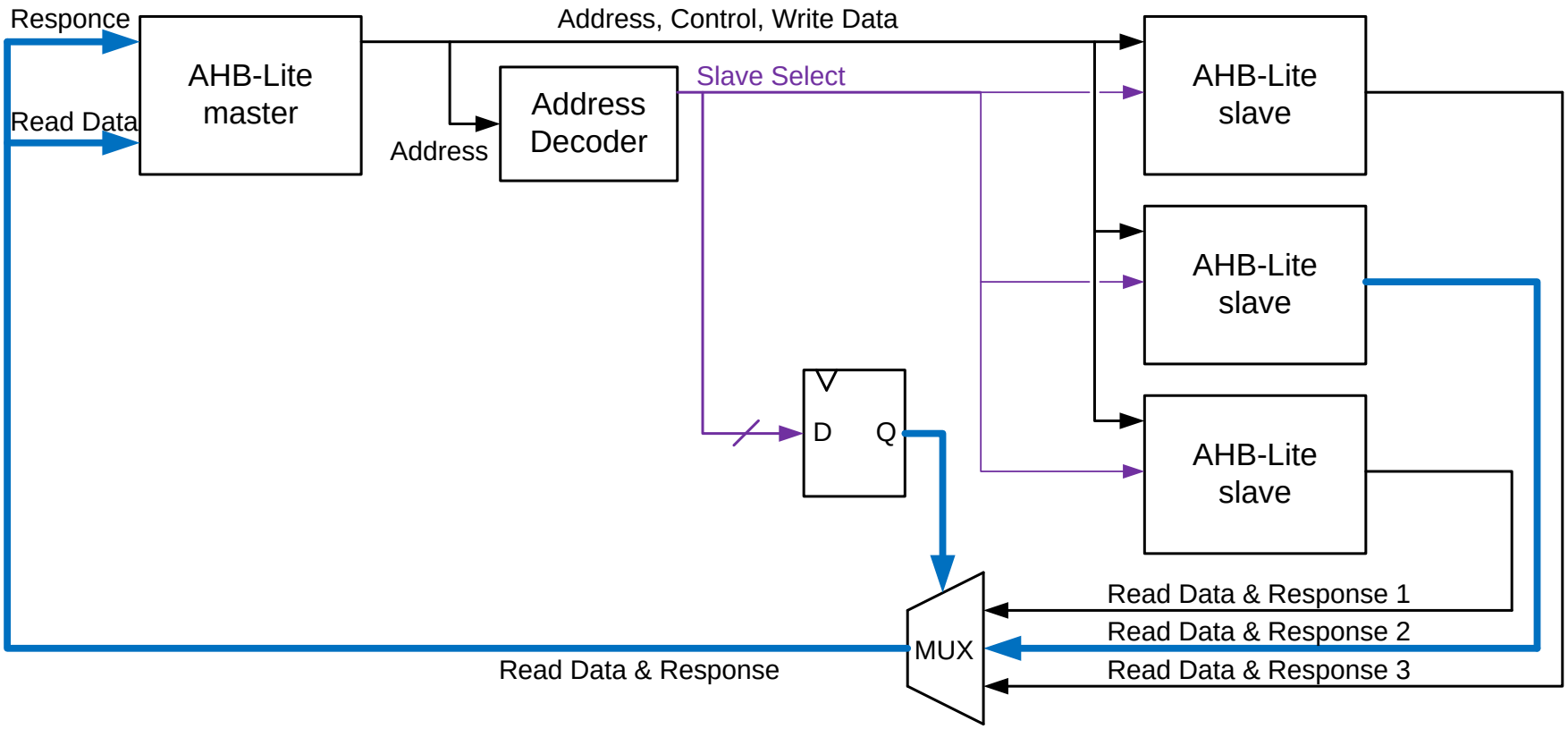
Multiple Slaves (simplified)



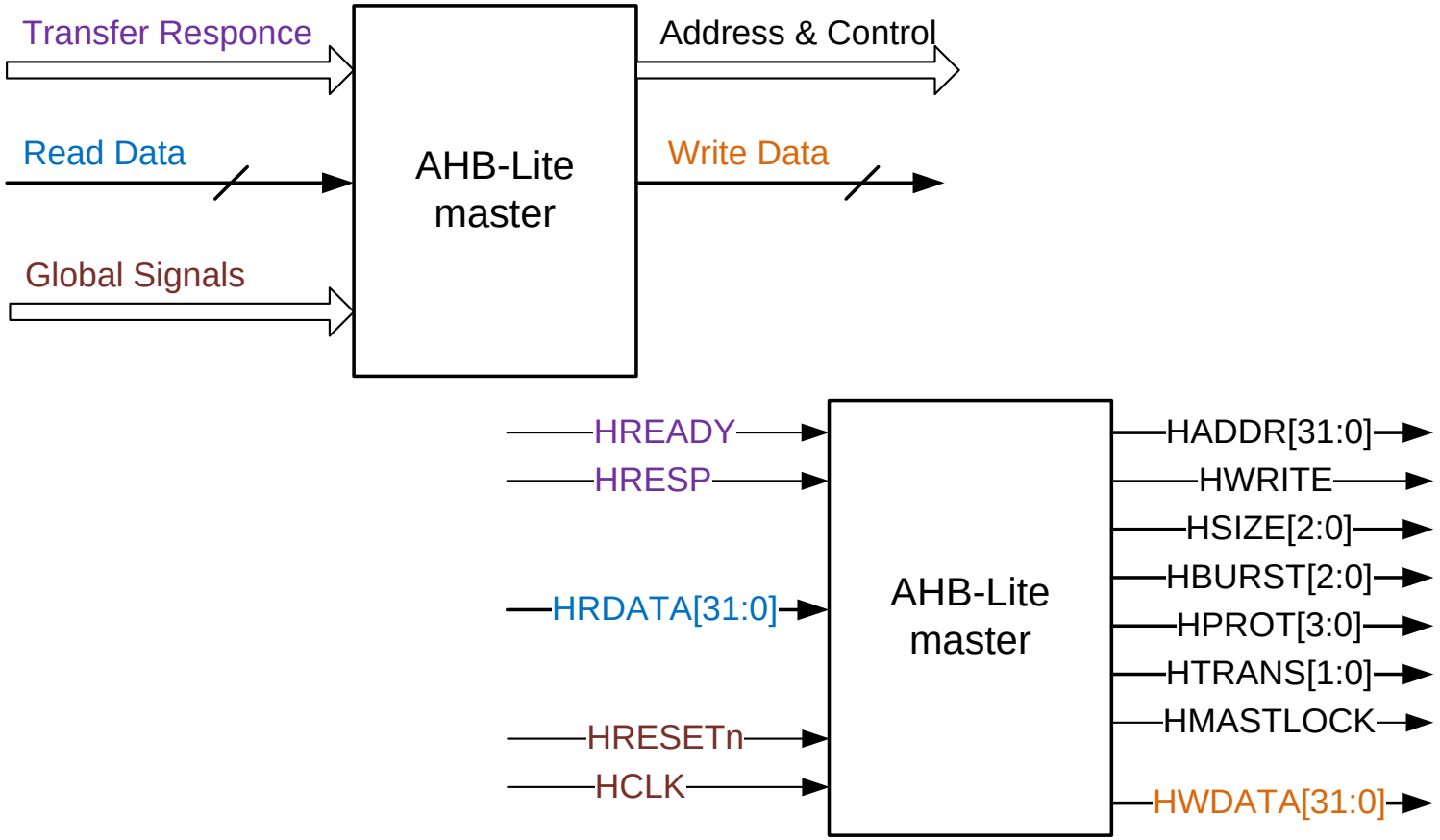
Address Phase (simplified)



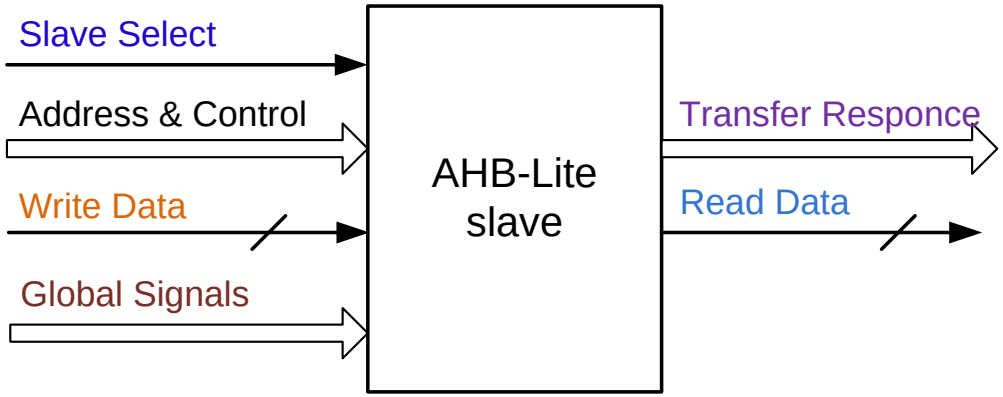
Data Phase (simplified)

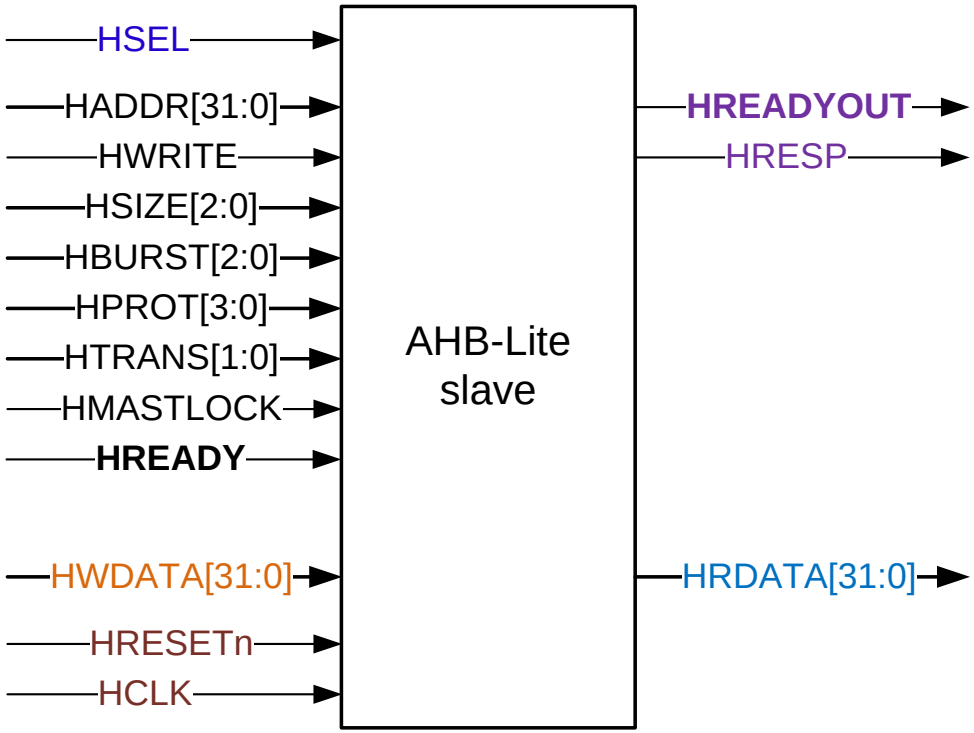


AHB-Lite Master Signals

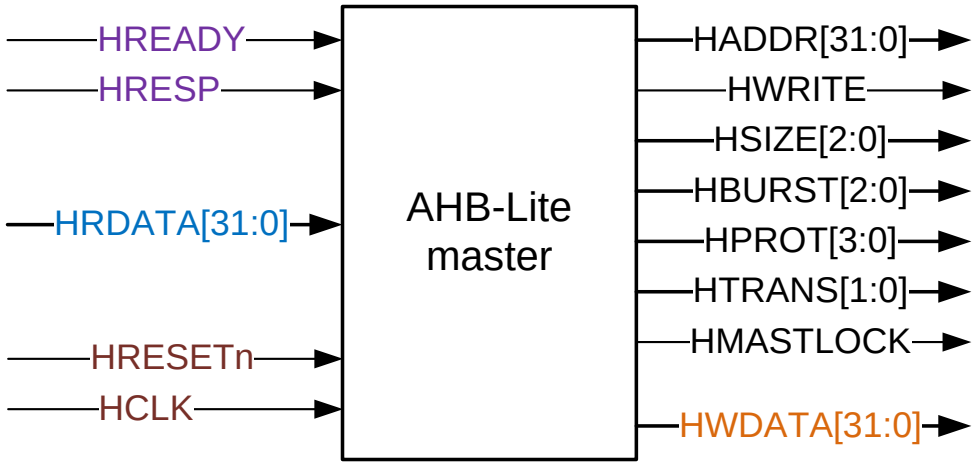


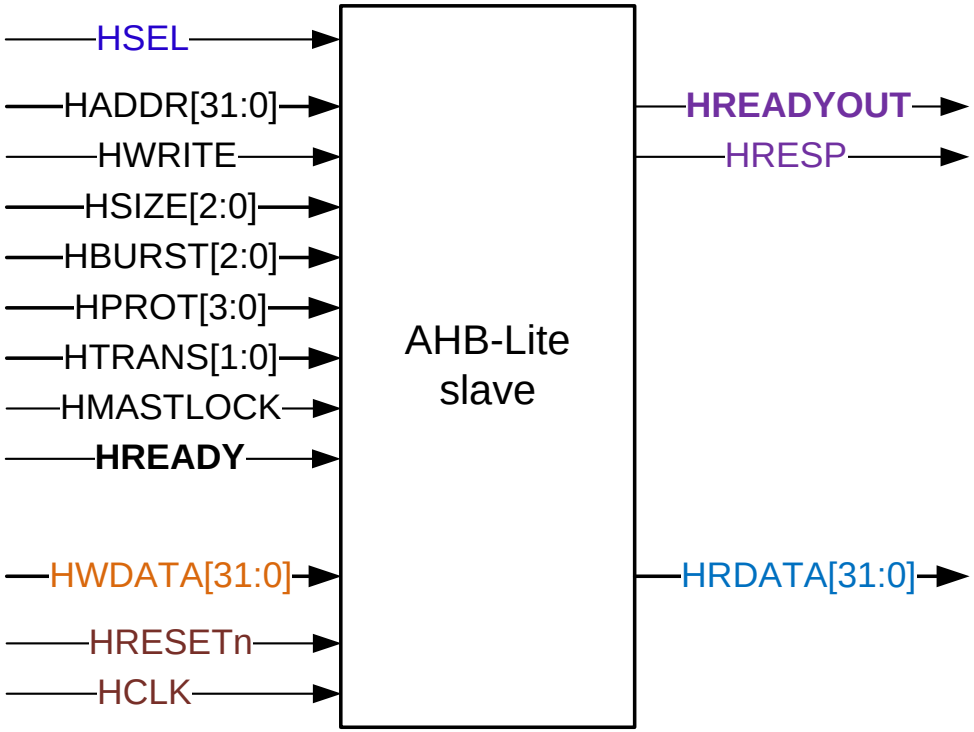
AHB-Lite Slave Signals





AHB-Lite Master & Slave Signals

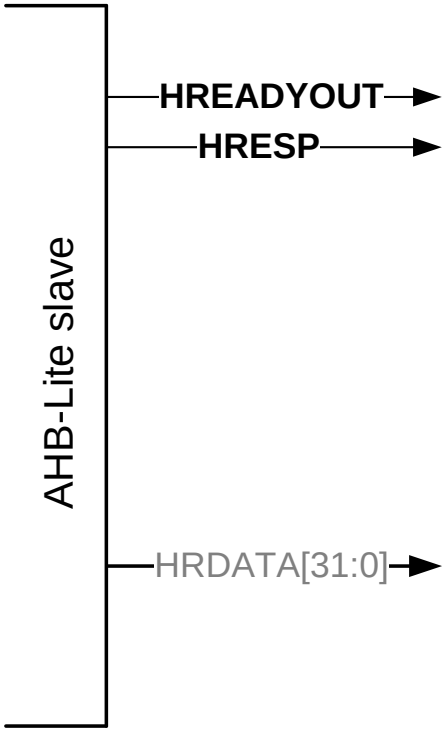




AHB-Lite support in MIPSfpga

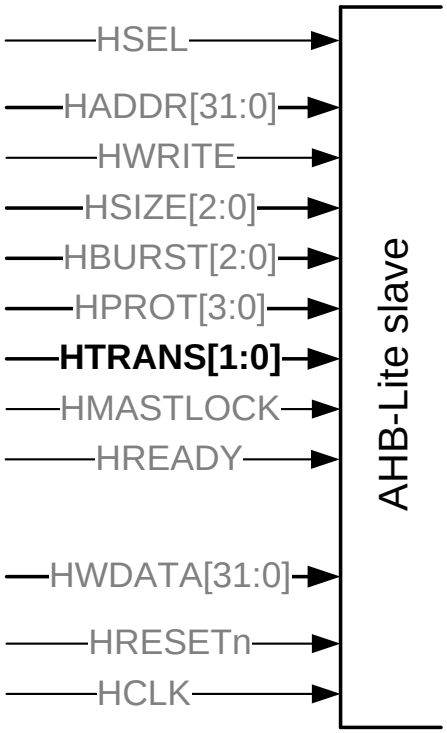
- Not all transaction types (signal combination) supported
- Where to find the information:
  - *MIPS32 microAptiv UP Processor Core AHB-Lite Interface*
  - *MIPS32 microAptiv UP Processor Core Family Integrator's Guide*

Transfer Response



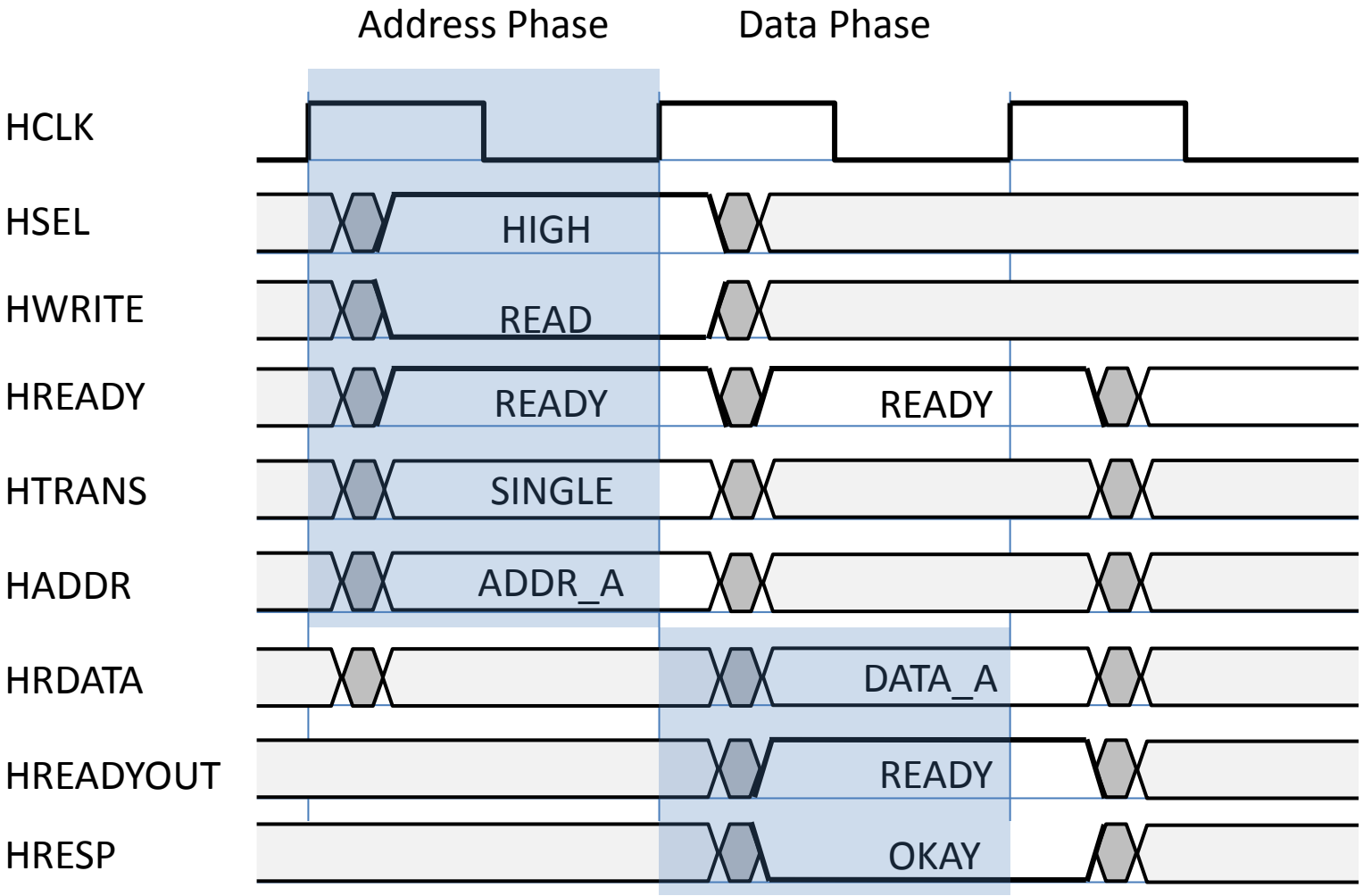
Signal	Description
HREADYOUT	Is transfer finished? 0 – WAIT 1 – READY
HRESP	Was there an error? 0 – OKAY 1 – ERROR => bus error exception

Control Signals. HTRANS



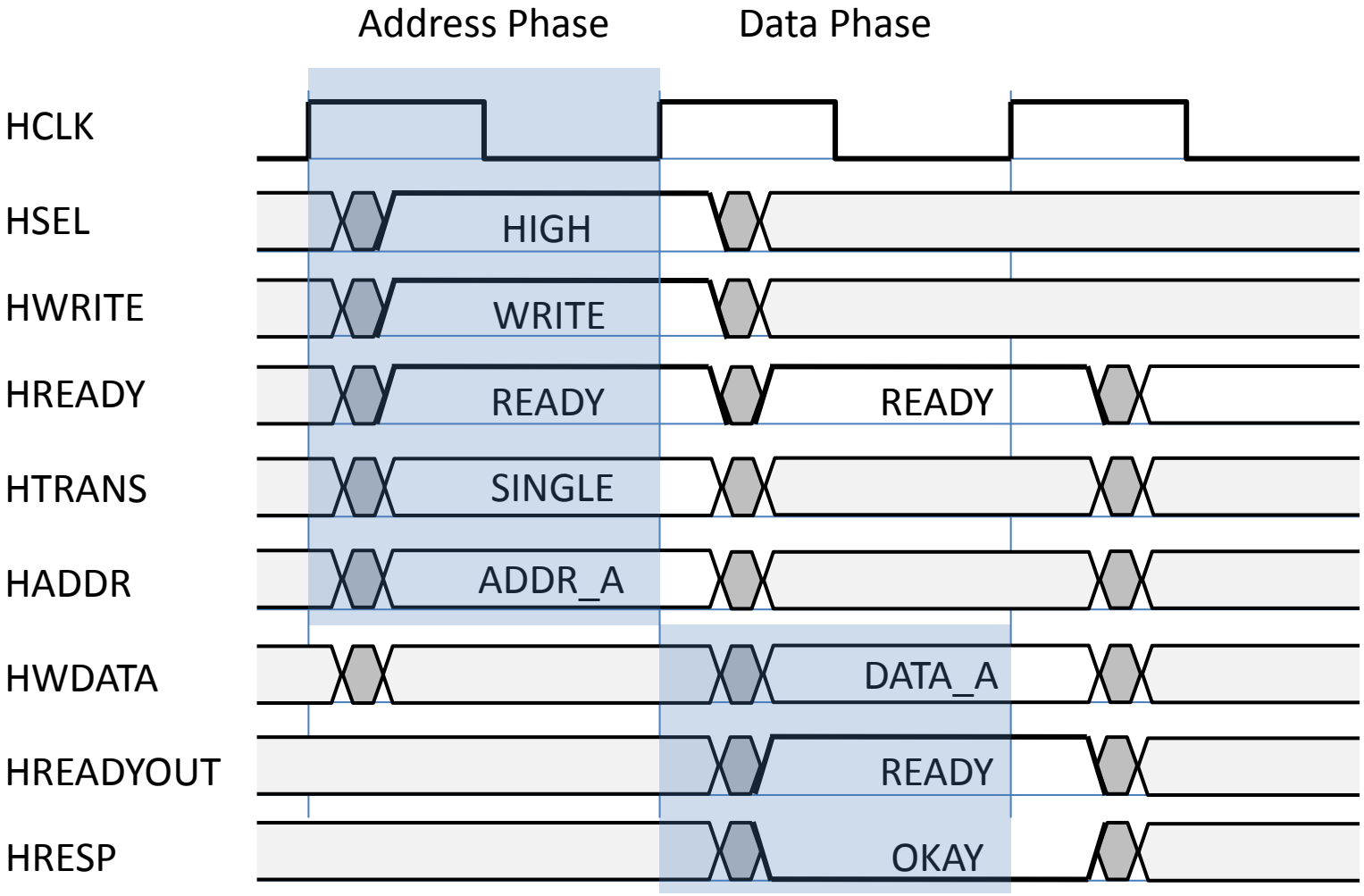
HTRANS[1:0]	Description
2'b 00	IDLE Master does not wish to perform a transfer
2'b 01	BUSY not supported in MIPSfpga
2'b 10	NONSEQ the first transfer of the burst or a single transfer
2'b 11	SEQ the remaining transfers of the burst

Read Transfer



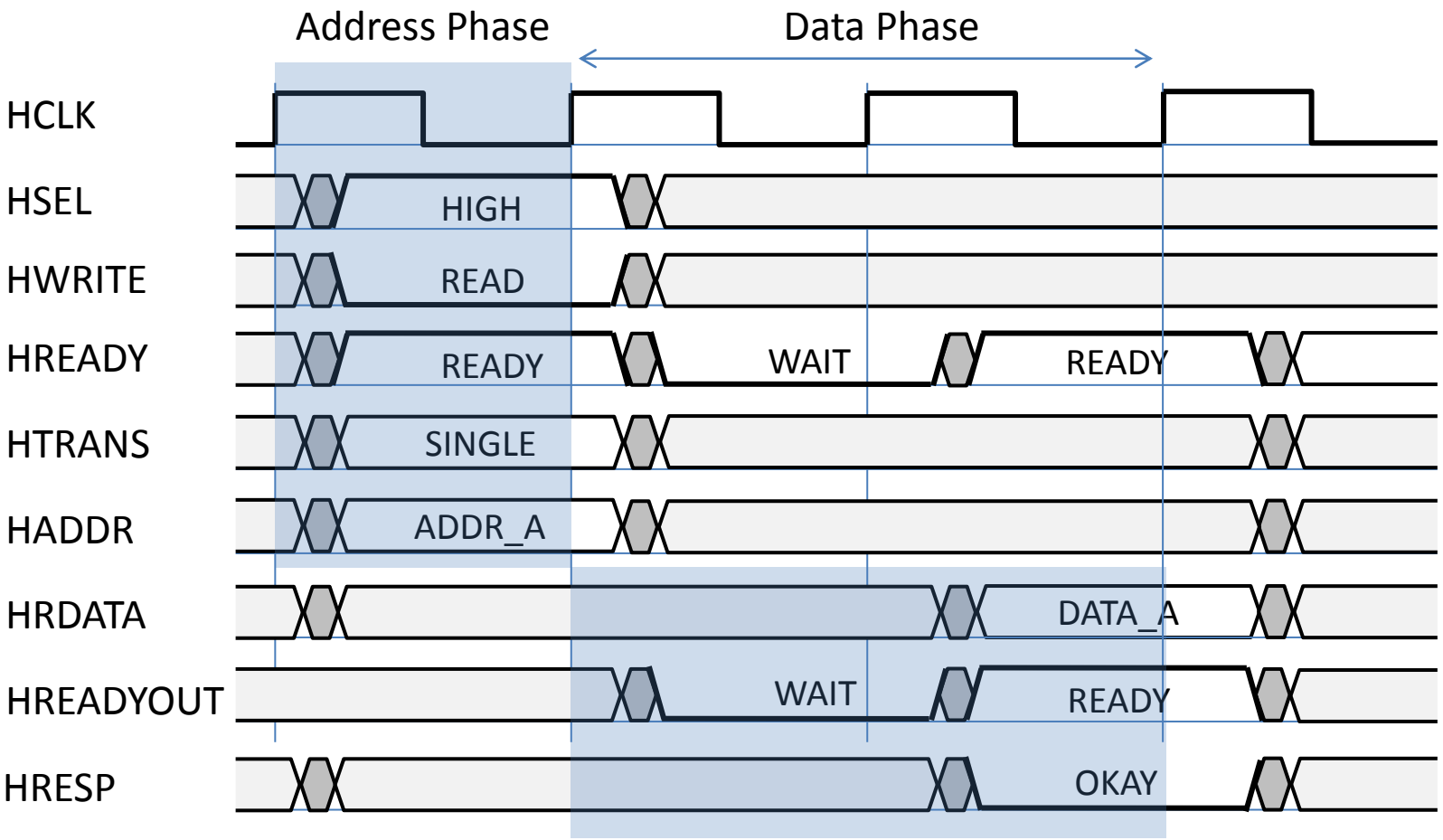
The minimum required set of signals

Write Transfer

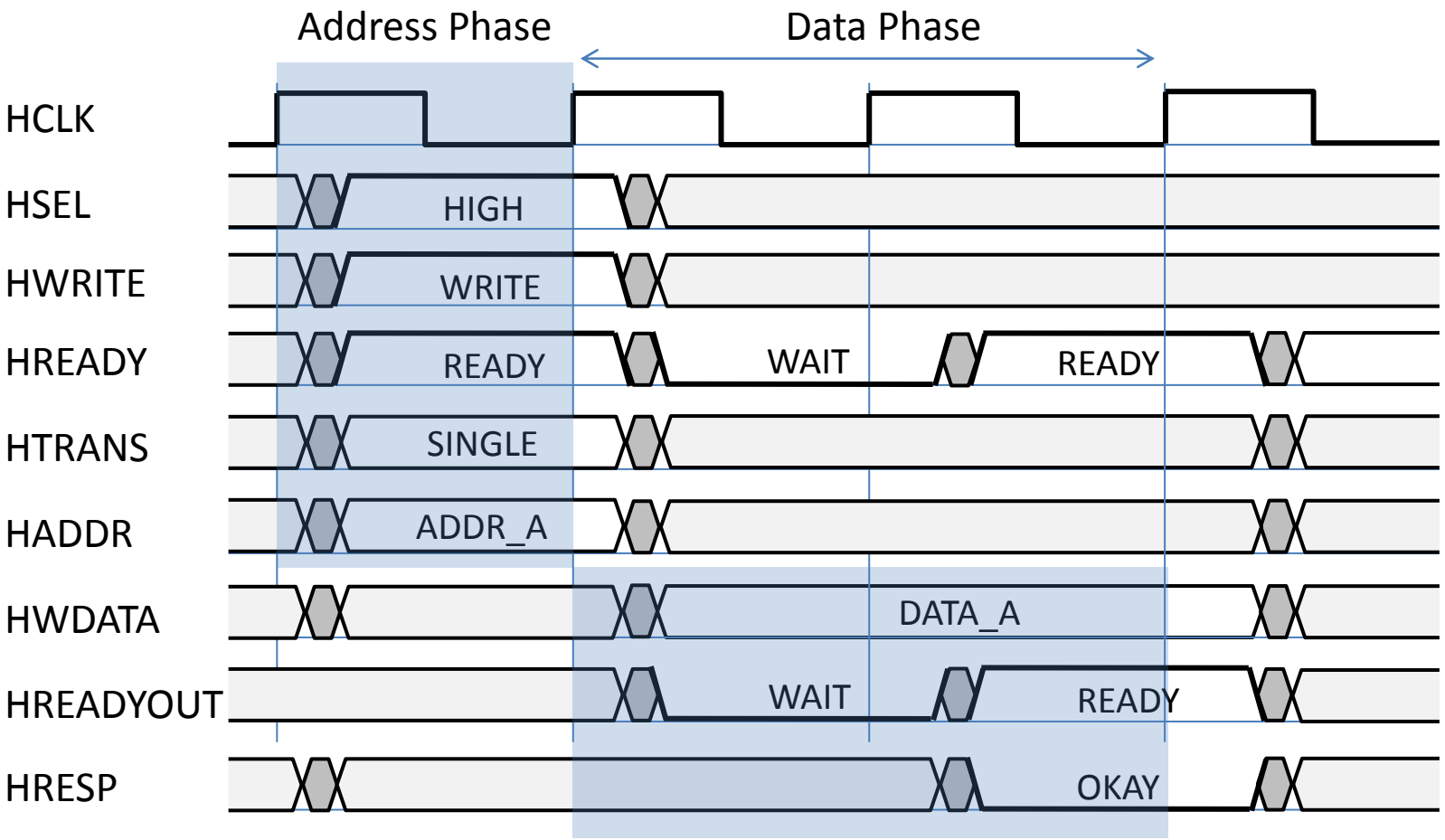


The minimum required set of signals

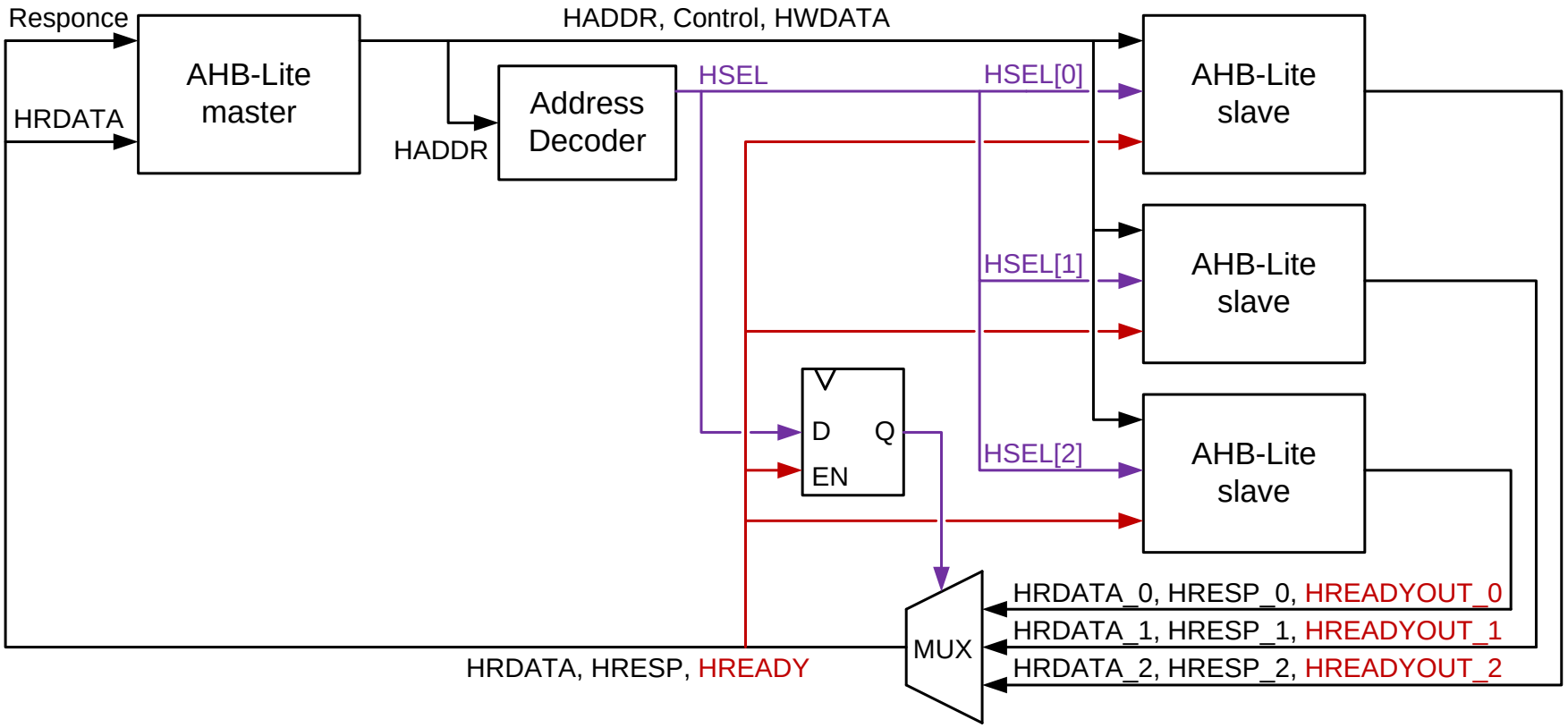
Read Transfer with one Wait state



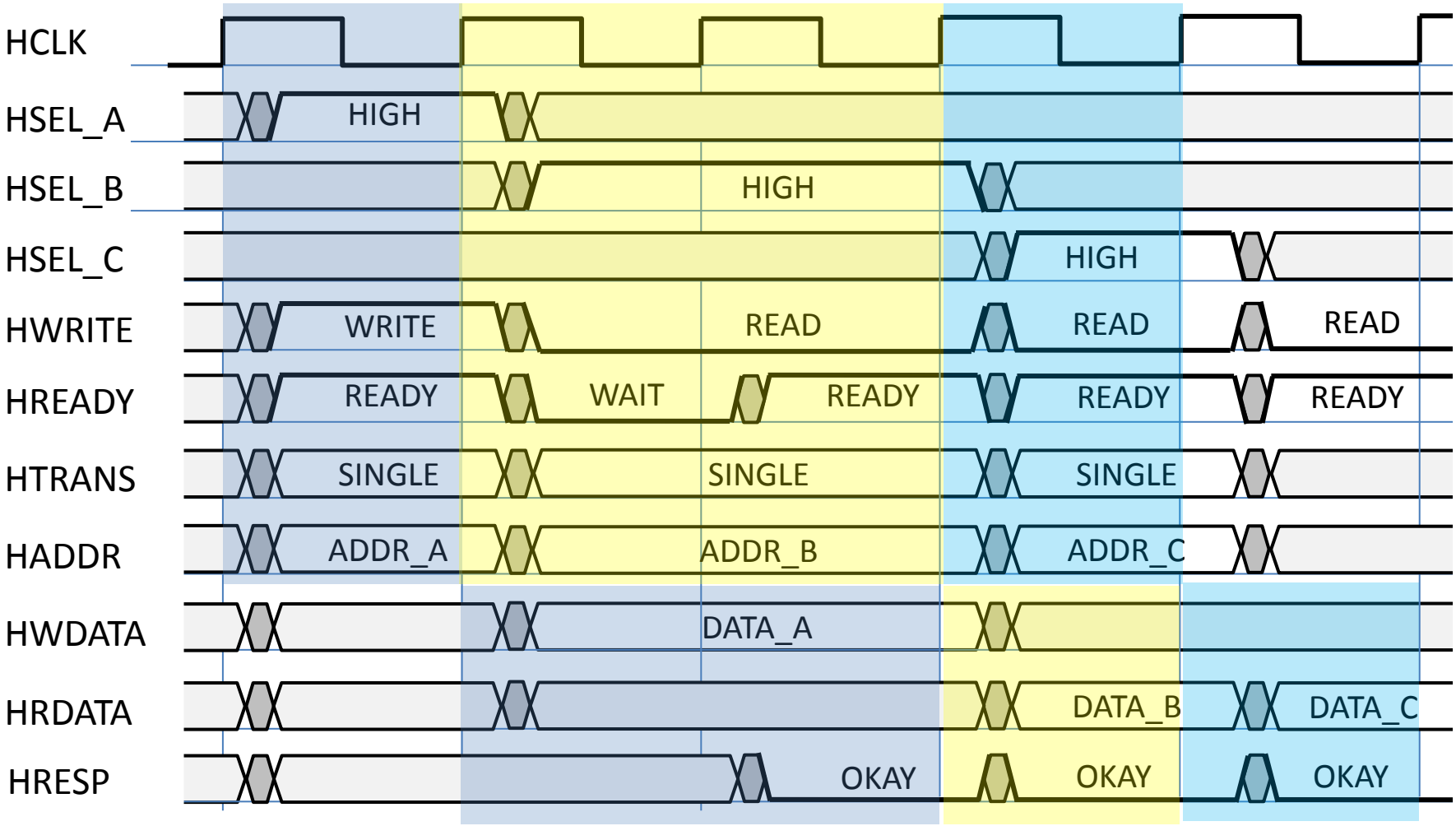
Write Transfer with one Wait state



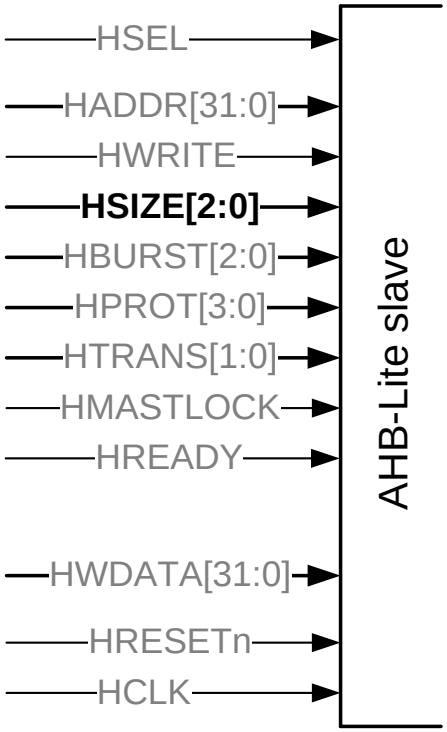
Bus Interconnect and Wait state



Multiple Transfers

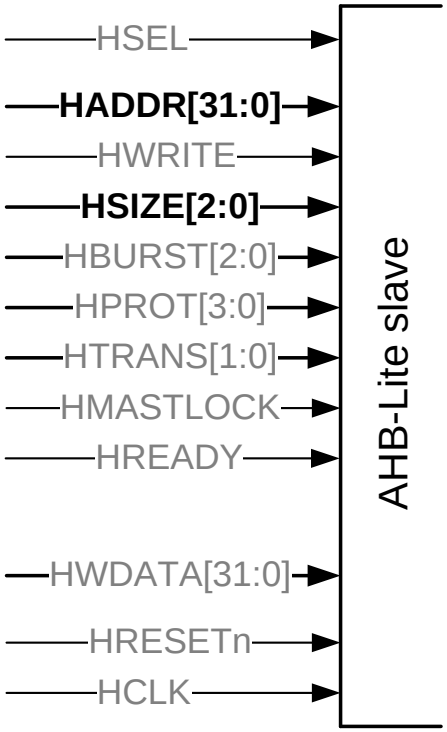


Control Signals. HSIZE



HSIZE[2:0]	Назначение
3’b 000	BYTE (8 bit)
3’b 001	HALFWORD (16 bit)
3’b 010	WORD (32 bit)
3’b 011 - 3’b 111	DOUBLEWORD (64 bit) – … not supported in MIPSfpga

Control Signals. HSIZE & HADDR

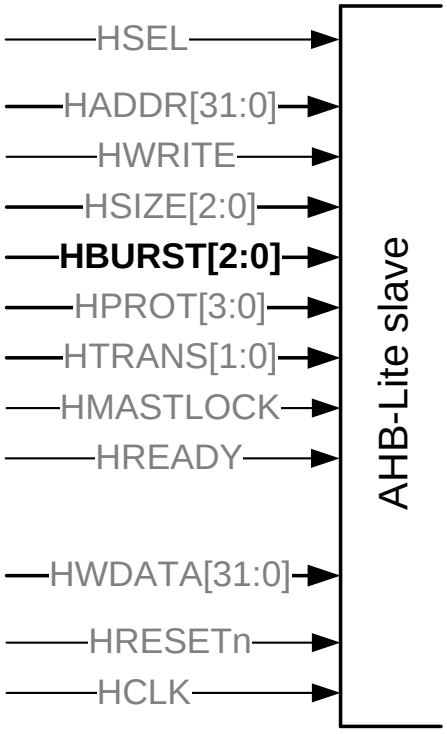


Address phase: Data Phase:

HSIZE [2:0]	Size	HADDR [1:0]	HRDATA, HWDATA			
			[31:24]	[23:16]	[15:8]	[7:0]
3'b 000	BYTE	2'b00				Active
3'b 000	BYTE	2'b01			Active	
3'b 000	BYTE	2'b10		Active		
3'b 000	BYTE	2'b11	Active			
3'b 001	HALFWORD	2'b00			Active	Active
3'b 001	HALFWORD	2'b10	Active	Active		
3'b 010	WORD	2'b00	Active	Active	Active	Active

The Little Endian is shown, as it is used in MIPSfpga

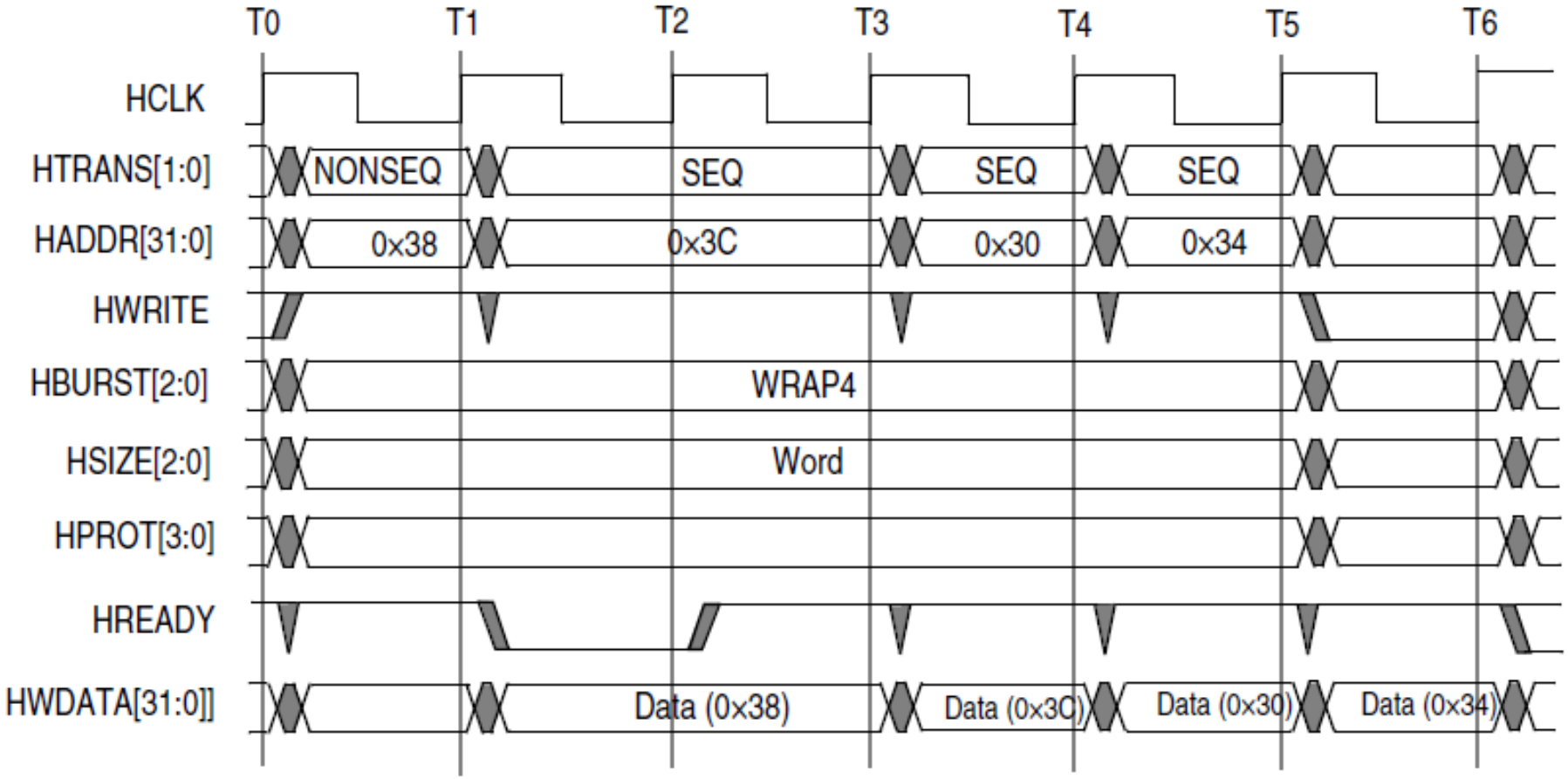
Control Signals. HBURST



HBURST [2:0]	Режим	Описание
3'b 000	SINGLE	Single transfer
3'b 001	INCR	Incrementing burst of undefined length not supported in MIPSfpga
3'b 010	WRAP4	4-beat wrapping burst
3'b 011 - 3'b 111	INCR4, WRAP8 … INCR16	not supported in MIPSfpga

The WRAP4 order of address processing is discussed in:
MIPS32 microAptiv UP Processor Core AHB-Lite Interface

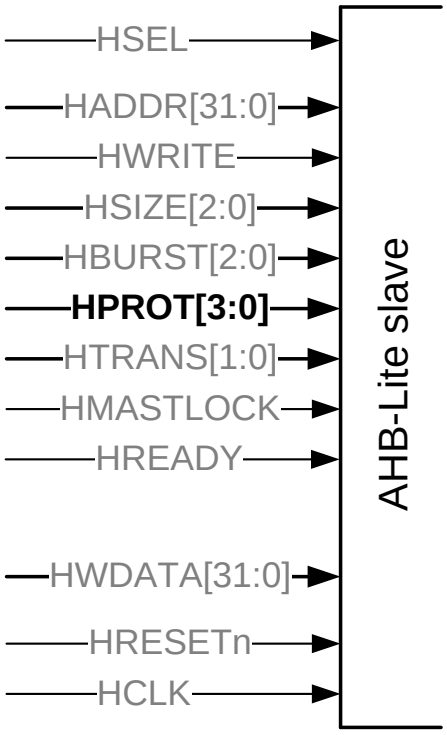
Example of Burst Transfer



Burst Transfer and Peripherals

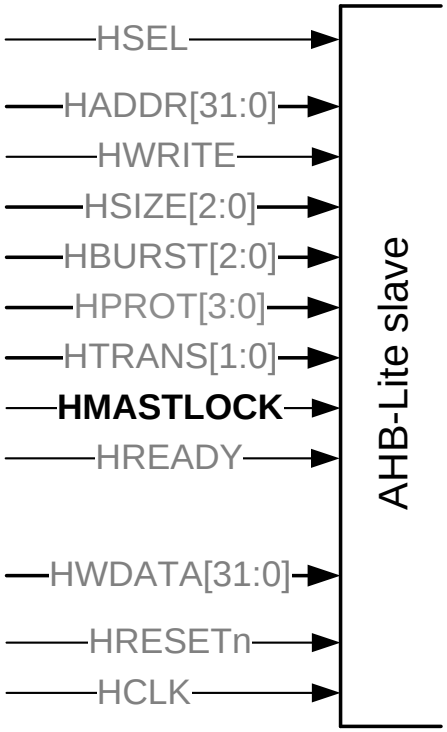
- If the Burst Mode is not supported by the peripheral every transfer will be processed as a SINGLE transaction
- There is no need to implement Burst mode if
  - there is no wait states on read and write transfers (the device is fast)
  - there is a wait state on every read transfer (the read data are not cached by device)

Control Signals. HPROT



HPROT[3:0]	Description
HPROT[0]	0 – Opcode fetch 1 – Data access
HPROT[1]	0 – User access 1 – Privileged access always 1, not supported by MIPSfpga
HPROT[2]	0 – Non-bufferable 1 – Bufferable not supported by MIPSfpga
HPROT[3]	0 – Non-cacheable 1 – Cacheable not supported by MIPSfpga

Control Signals. HMASTLOCK



- is used to lock acces of a **read-modify-write** (RMW) sequence
- Raised by atomic assembler instructions: **LL** and **SC**
- transfer timing and detals: *MIPS32 microAptiv UP Processor Core AHB-Lite Interface*

Control Signals supported by MIPSfpga

HTRANS[1:0]
IDLE
BUSY
NONSEQ
SEQ

HBURST[2:0]
SINGLE
INCR
WRAP [4 | 8 | 16]
INCR [4 | 8 | 16]

HMASTLOCK
UNLOCKED
LOCKED

HSIZE[2:0]
BYTE
HALFWORD
WORD
DOUBLEWORD
…

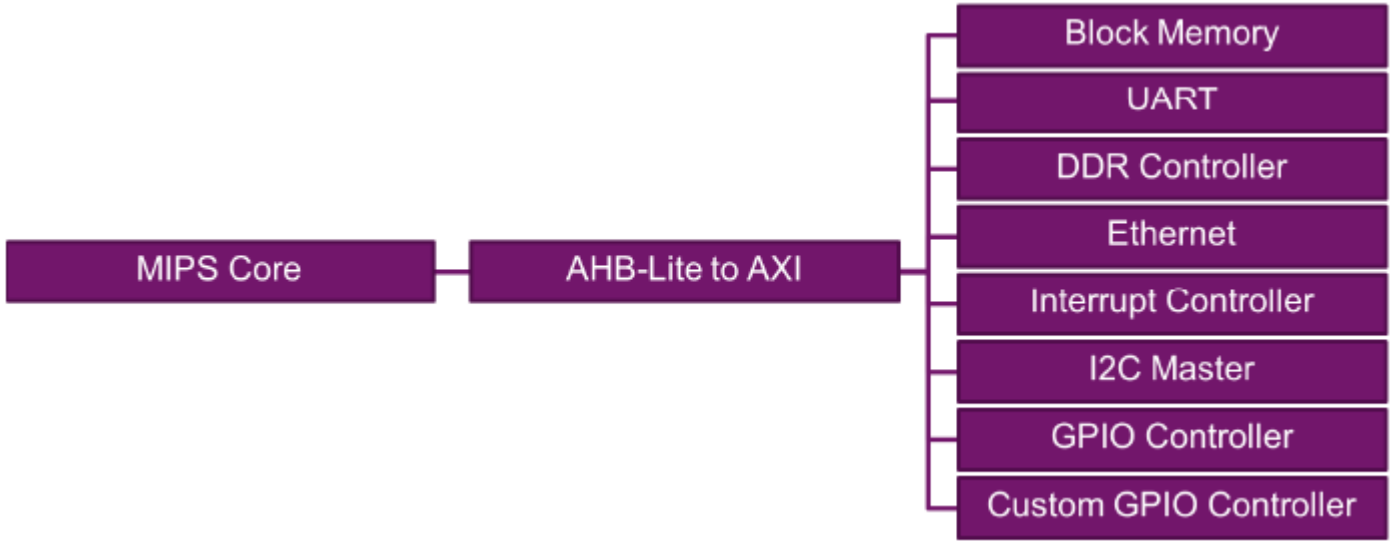
HPROT[3:0]
Data / Opcode
Privileged / User
Bufferable / Non-bufferable
Cacheable / Non-cacheable

Multiple Master?

Possible implementations:

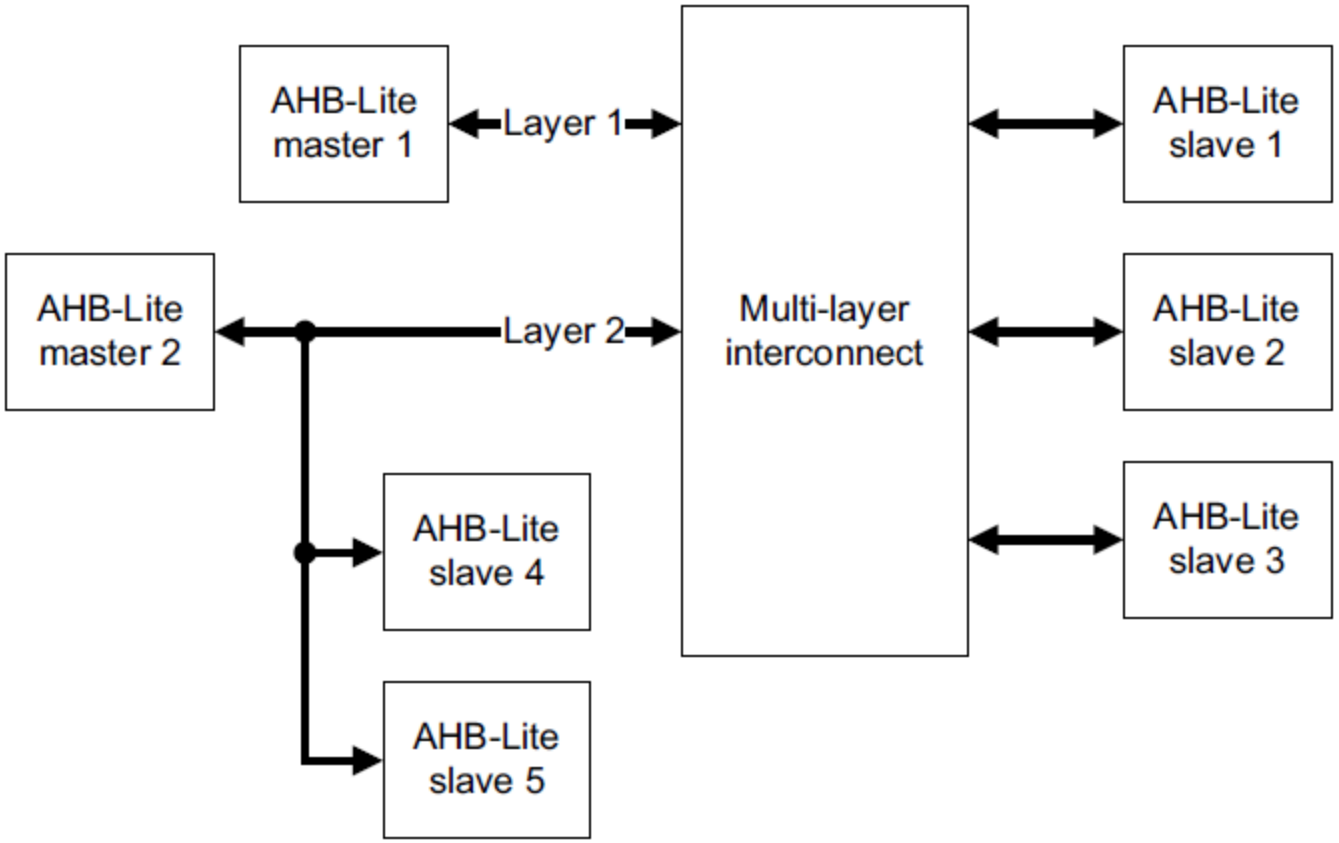
- bridge to another bus (AXI or Wishbone)
- multi layer AHB-Lite

AHB-Lite – AXI bridge

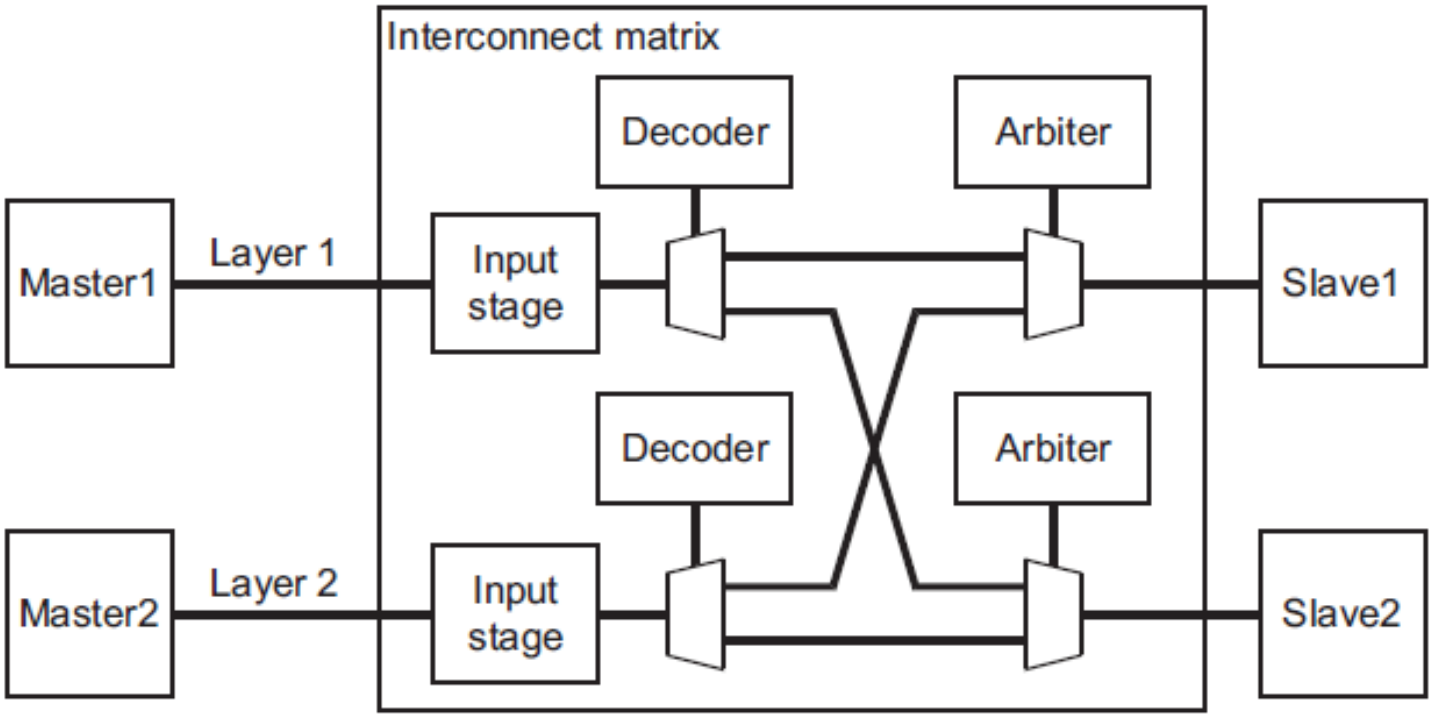


- implemented in MIPSfpga SoC for Xilinx platform
- discussed in
  MIPSfpga SOC Advanced Starter Tutorial

Multi-layer AHB-Lite



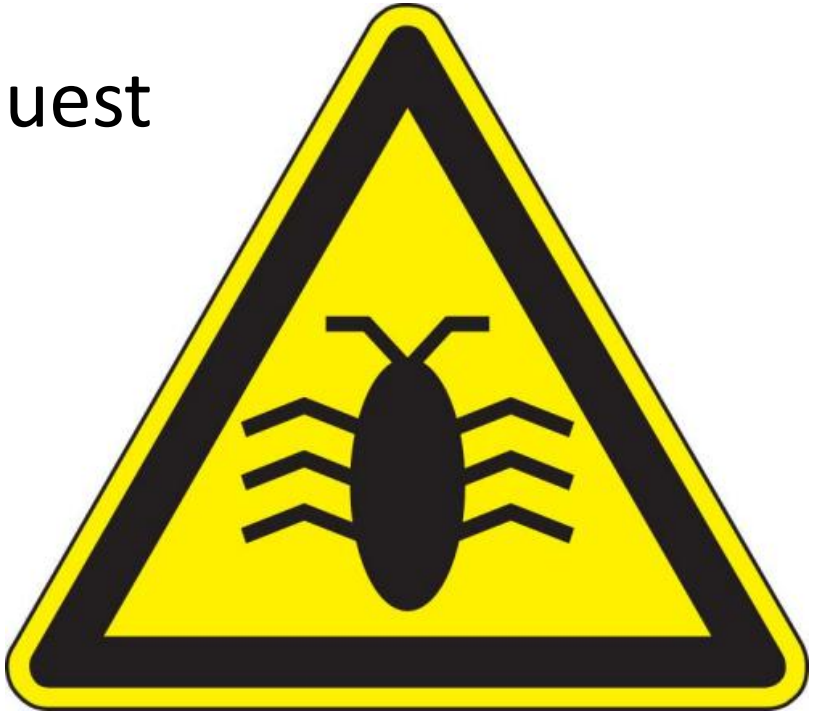
Multi-layer AHB-Lite Interconnect Matrix



Discussed in *ARM DVI 0045B. Multi-layer AHB*

Where bugs may come

- BYTE & HALFWORD transfers
- Read-after-Write
- HREADYOUT before Request

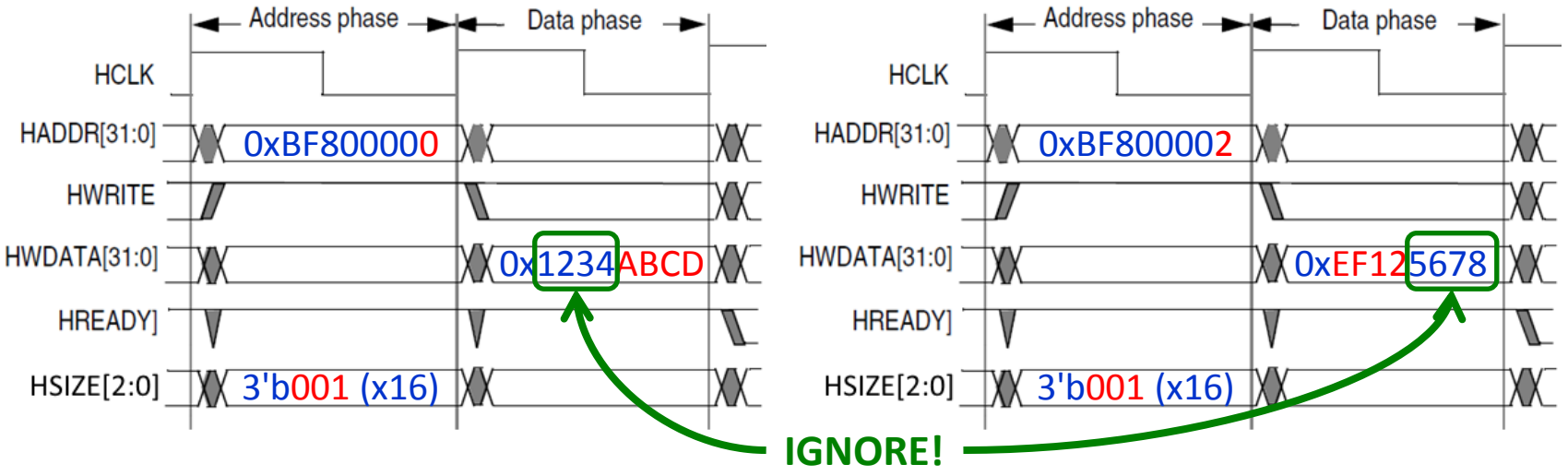


BYTE & HALFWORD Transfers

- x8 and x16 transfers support is optional
- The WORD (x32) transfer support is enough in most of cases
- If x8 and x16 transfers are not supported, but requested:
  - READ – Ok
  - WRITE – data corruption
- Only WORD (x32) transfers are used by MIPSfpga for cached memory access

HALFWORD Write Transfer

```
(* (volatile uint16_t *) 0xBF800000 ) = 0xABCD;
(* (volatile uint16_t *) 0xBF800002 ) = 0xEF12;
```



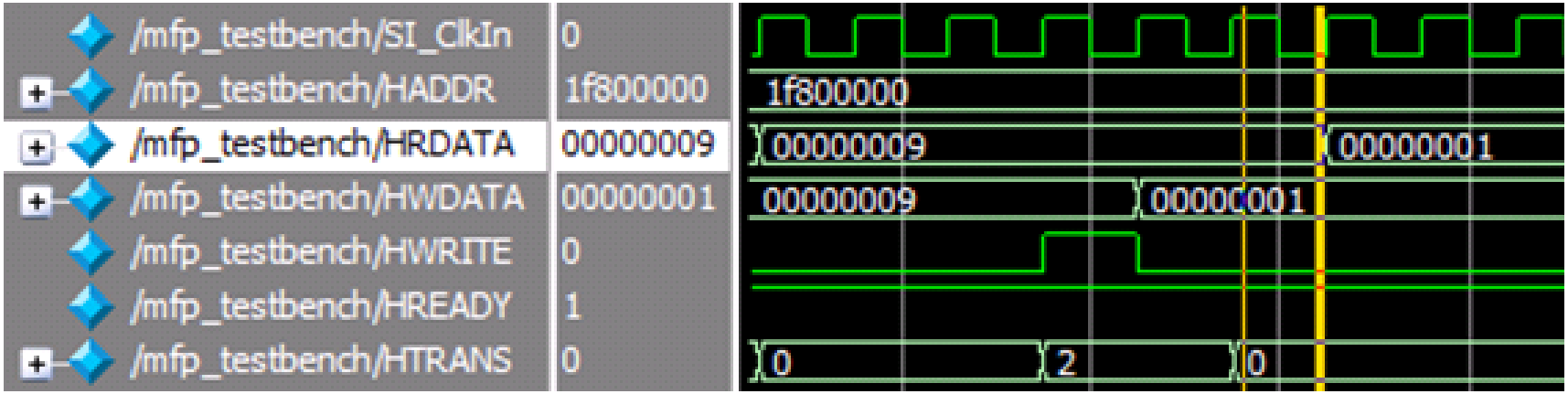
BYTE and HALFWORD Access Test

```
#include <stdint.h>

#define MFP_TEST_BASE 0xBF800000

void __attribute__((optimize("O0"))) accessTest(void)
{
    (* (volatile uint32_t *) MFP_TEST_BASE     ) = 0x12345678;

    (* (volatile uint16_t *) MFP_TEST_BASE     ) = 0xABCD;
    (* (volatile uint16_t *) MFP_TEST_BASE + 2 ) = 0xEF12;

    (* (volatile uint8_t  *) MFP_TEST_BASE     ) = 0x34;
    (* (volatile uint8_t  *) MFP_TEST_BASE + 1 ) = 0x56;
    (* (volatile uint8_t  *) MFP_TEST_BASE + 2 ) = 0x78;
    (* (volatile uint8_t  *) MFP_TEST_BASE + 3 ) = 0xAB;
}
```

Read-After-Write

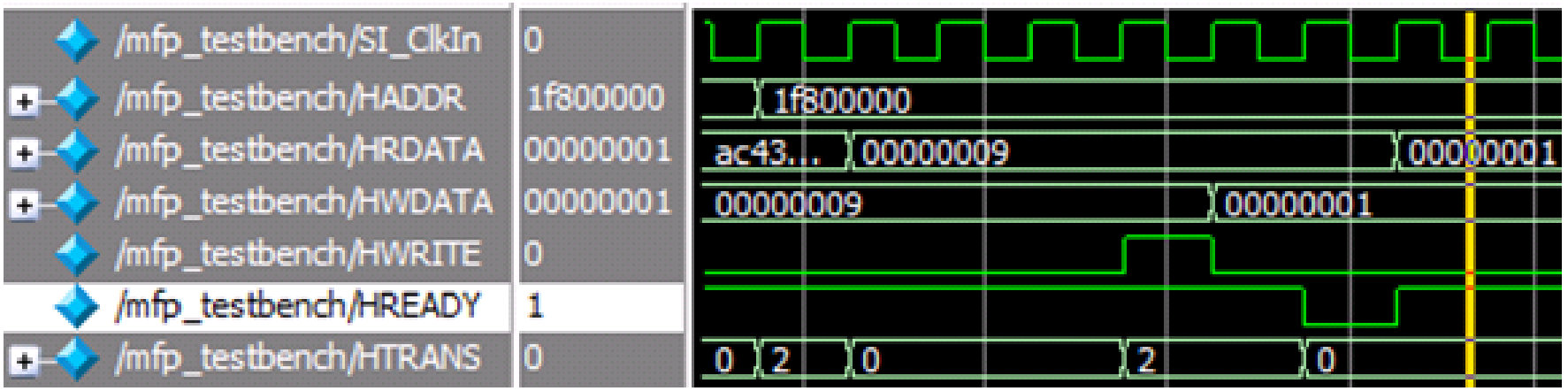
```
#define MFP_RED_LEDS_ADDR 0xBF800000
#define MFP_RED_LEDS (* (volatile unsigned *)
                                MFP_RED_LEDS_ADDR  )
int main ()
{
    long long int n = 0;
    for (;;)
    {
        MFP_RED_LEDS   = n++;
        MFP_GREEN_LEDS = MFP_RED_LEDS;
    }
}
```

Read-After-Write Error



- Read and Write are processed on the same clock edge
- The “old” read value is transferred to the bus
- uncached access only
- heisenbug, depends on the compiler asm output

Read-After-Write Processing



- Hardware - one Wait state added (HREADY = 0)
- Software – add NOP after register write

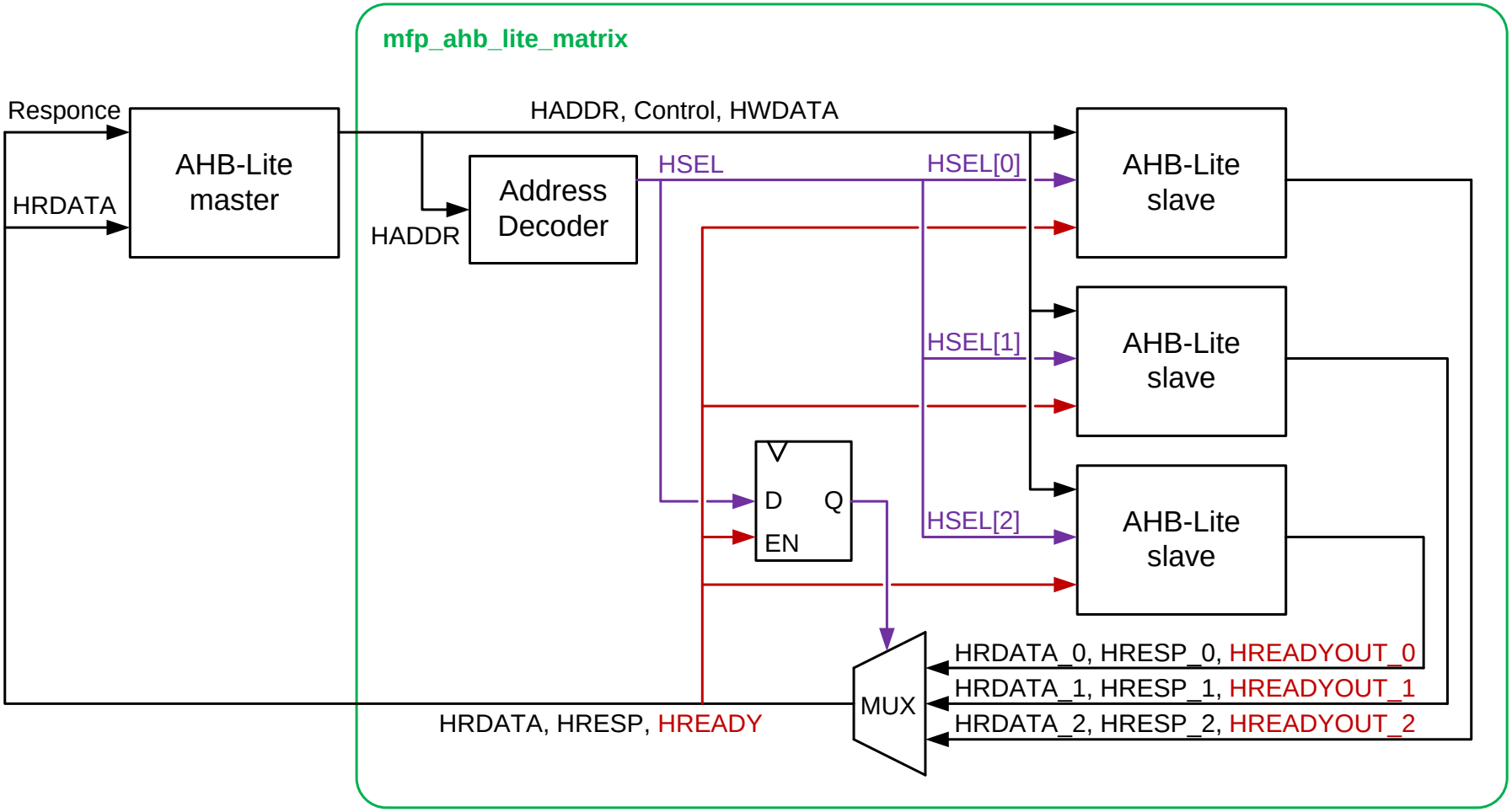
HREADYOUT before request

- If not requested peripheral have set HREADYOUT = 0 it can miss the next incoming request

- Temporary workaround example:

```
reg READY; // HREADYOUT after bus mux

//MFP SDRAM bug workaround
`ifdef MFP_USE_SDRAM_MEMORY
    assign HREADY = READY & HREADYOUT[1];
`else
    assign HREADY = READY;
`endif
```

AHB-Lite bus in MIPSfpga-plus



MIPSfpga-plus Address Decoder

```
module mfp_ahb_lite_decoder
(
   input  [                          31 : 0 ] HADDR,
   output [ `MFP_AHB_DEVICE_COUNT - 1 : 0 ] HSEL
);
   // Decode based on most significant bits of the address

   // RAM 4 MB max at 0xbfc00000 (physical: 0x1fc00000 - 0x1fffffff)
   assign HSEL [0] = ( HADDR [28:22] == `MFP_RESET_RAM_ADDR_MATCH );

   // RAM 64 MB max at 0x80000000 (physical: 0x00000000 - 0x03FFFFF)
   assign HSEL [1] = ( HADDR [28:26] == `MFP_RAM_ADDR_MATCH );

   // GPIO 4 MB max at 0xbf800000 (physical: 0x1f800000 - 0x1fbfffff)
   assign HSEL [2] = ( HADDR [28:22] == `MFP_GPIO_ADDR_MATCH );
   …
endmodule
```

MIPSfpga-plus Address Mux

```
module mfp_ahb_lite_response_mux
(
  input      [ `MFP_AHB_DEVICE_COUNT - 1 : 0 ] HSEL_R,
  input      [                          31 : 0 ] RDATA_0,
  input      [                          31 : 0 ] RDATA_1,
  input      [                          31 : 0 ] RDATA_2,
  ...
  input      [ `MFP_AHB_DEVICE_COUNT - 1 : 0 ] RESP,
  input      [ `MFP_AHB_DEVICE_COUNT - 1 : 0 ] HREADYOUT,

  output reg [                          31 : 0 ] HRDATA,
  output reg                                     HRESP,
  output reg                                     HREADY
);

  always @*
    casez (HSEL_R)
       6'b?????1 : begin HRDATA = RDATA_0; HRESP = RESP[0]; HREADY = HREADYOUT[0]; end
       6'b????10 : begin HRDATA = RDATA_1; HRESP = RESP[1]; HREADY = HREADYOUT[1]; end
       6'b???100 : begin HRDATA = RDATA_2; HRESP = RESP[2]; HREADY = HREADYOUT[2]; end
       ...
    endcase
endmodule
```

Peripheral Types and AHB-Lite features

- Fast onchip memory (Block Memory)
- Slow External Memory (SDRAM/DDR)
- Memory-mapped I/O (Registers)

AHB-Lite Feature	Block Memory*		SDRAM/DDR		MMIO*	
	Cached	Uncached	Cached	Uncached	RW	RO
Burst Transactions	-	-	recomm.	recomm.	-	-
WORD и HALFWORD Transactions	-	yes	-	yes	-	-
Read-After-Write	recomm.	yes	-	-	yes	-

* - single clock edge access

Implementation Examples

- Block Memory
  - MFP: mfp_ahb_lite_slave.v
  - MFP: mfp_ahb_ram_busy.v
  - schoolMIPS: ahb_ram.v
- SDRAM
  - MFP: mfp_ahb_ram_sdram.v
- MMIO Registers
  - MFP: mfp_ahb_gpio_slave.v
  - MFP: mfp_ahb_lite_uart16550.v
  - MFP: mfp_ahb_lite_pmod_als.v
  - schoolMIPS: ahb_gpio.v

Document Sources

- Get Started AMBA 3 AHB–Lite HD (youtube.com) from ARM
- ARM DVI 0045B. Multi-layer AHB
- ARM IHI 0033A. AMBA 3 AHB-Lite Protocol v1.0
- ARM DUI 0926A. ARM Cortex-M0 DesignStart RTL Testbench User Guide
- MD01082. MIPS32 microAptiv UP Processor Core AHB-Lite Interface
- MD00941. MIPS32 microAptiv UP Processor Core Family Integrator's Guide
- MIPSfpga SOC Advanced Starter Tutorial
- Practical experiences based on MIPSfpga

Additional Sources

- MIPS Academic Community (link)
  download MIPSfpga and Digital Design & Computer Architecture translations

- ARM DesignStart University Program (link)
  download *Cortex-M0 DesignStart Eval* and *Cortex-M3 DesignStart Eval.* The CPU cores are obfuscated, dev tools are time limited. The system bus source code is open.

- ARM Technical Docs (link)
  download ARM AMBA specifications

- ARM Youtube channel (link)
  AHB-Lite, ARM cores and other technical and marketing materials

Thank you!
Your Questions?