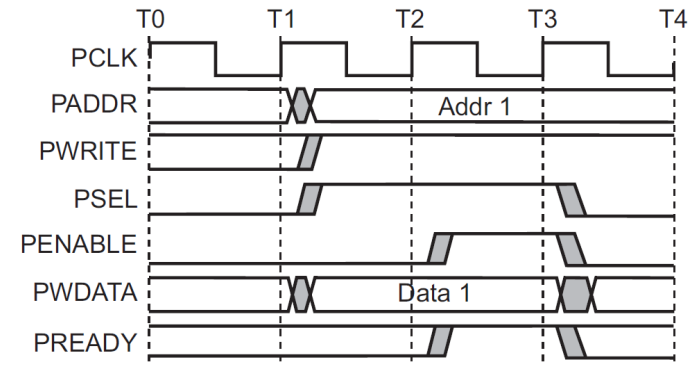




EECS 373

Design of Microprocessor-Based Systems

Prabal Dutta
University of Michigan



Lecture 5: Memory and Peripheral Busses
September 16, 2014

Announcements



- Homework #2
- Where was I last week?
 - VLCS'14
 - MobiCom'14
 - HotWireless'14



Emerging Retail Environment: A Walled Garden



- Often have line-of-sight to lighting
 - Groceries
 - Drugstores
 - Megastores
 - Hardware stores
 - Enterprise settings
- Lots of overhead lighting in retail
- Retailers deploying LED lighting
- Customers using phones in stores
 - Surf, Scan, Share
- Customers installing retailer apps
 - Maps, Barcodes, Deals, Shopping

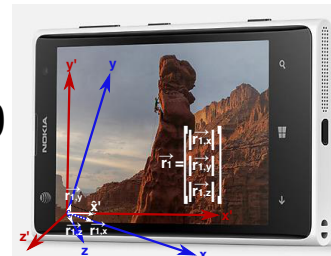


Visible Light Communications and Positioning

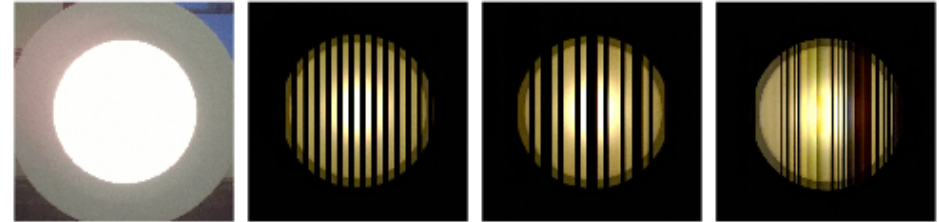


01100101000

LED Luminaire



Smart Phone



Illuminate

Idle

TX <66>

TX packet

Captured using a rolling shutter

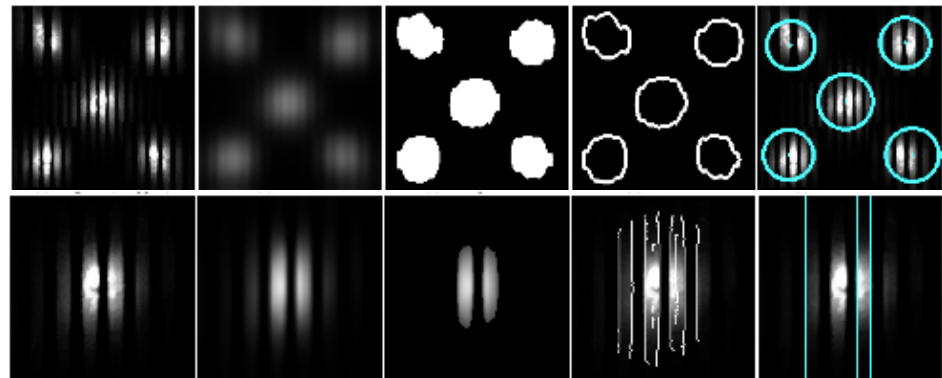
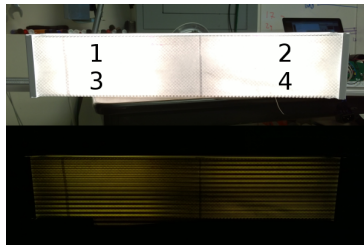
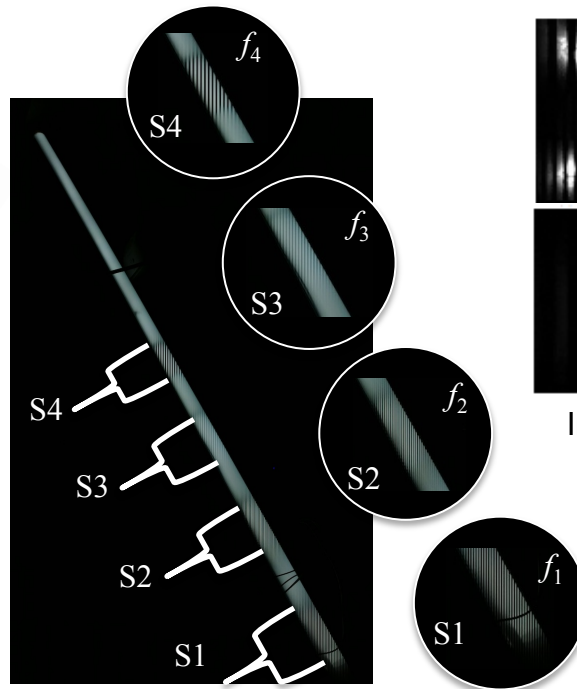


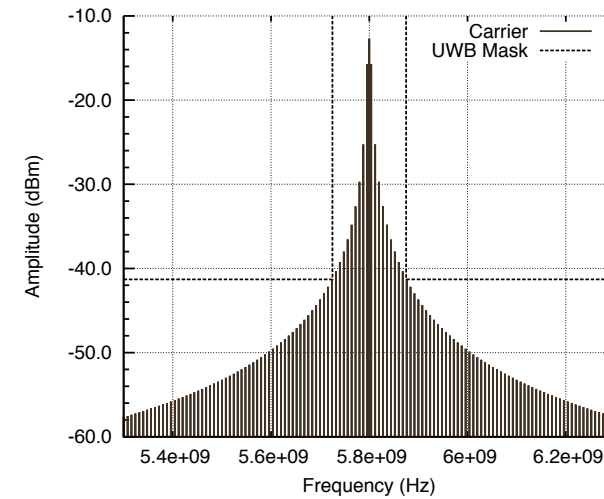
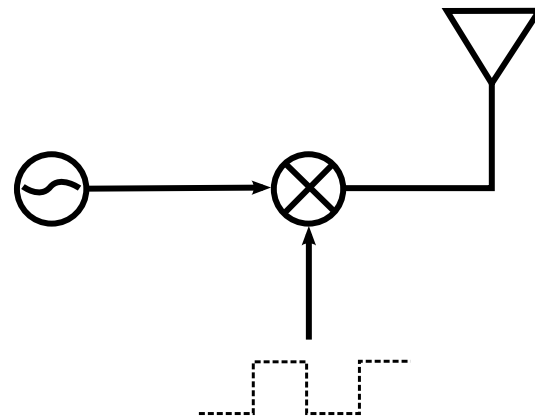
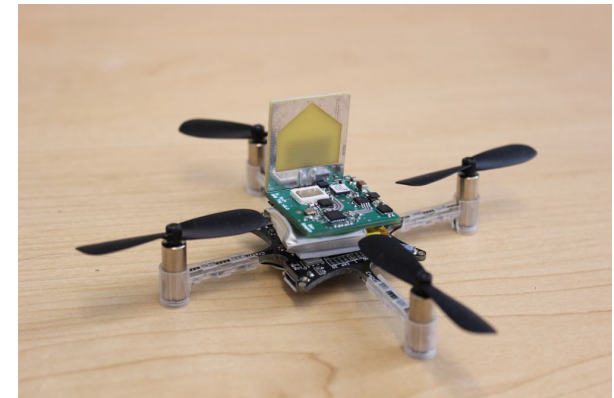
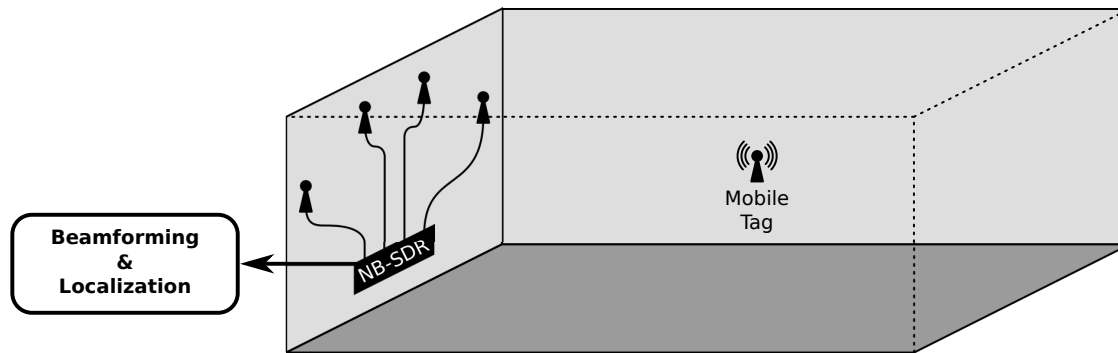
Image processing extracts beacon locations and frequencies

$$\begin{aligned}
 d_{0,1}^2 &= (u_0 - u_1)^2 + (v_0 - v_1)^2 + (w_0 - w_1)^2 \\
 &= (K_0 a_0 - K_1 a_1)^2 + (K_0 b_0 - K_1 b_1)^2 + Z_j^2 (K_0 - K_1)^2 \\
 &= K_0^2 |\vec{Oi_0}|^2 + K_1^2 |\vec{Oi_1}|^2 - 2K_0 K_1 (\vec{Oi_0} \cdot \vec{Oi_1}) \\
 &= (x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2,
 \end{aligned}$$

$$\sum_{m=1}^N \{ (Tx - xm)^2 + (Ty - ym)^2 + (Tz - zm)^2 - Km^2 (am^2 + bm^2 + Zj^2) \}^2$$

Compute
Minimize

Harmonia Tag



Outline



- Announcements
- Review
- ARM AHB-Lite



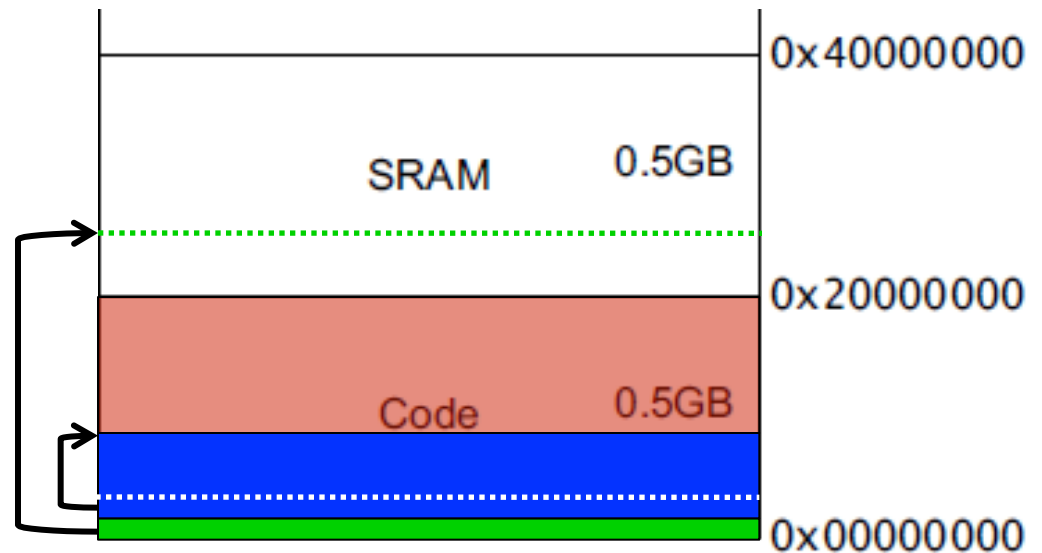
What happens after a power-on-reset (POR)?

- On the ARM Cortex-M3
- SP and PC are loaded from the code (.text) segment
- **Initial stack pointer**
 - LOC: 0x00000000
 - POR: SP \leftarrow mem(0x00000000)
- **Interrupt vector table**
 - *Initial* base: 0x00000004
 - Vector table is relocatable
 - Entries: 32-bit values
 - Each entry is an address
 - Entry #1: reset vector
 - LOC: 0x00000004
 - POR: PC \leftarrow mem(0x00000004)
- **Execution begins**

```
.equ    STACK_TOP, 0x20000800
.text
.syntax unified
.thumb
.global _start
.type   start, %function

_start:
.word   STACK_TOP, start

start:
movs r0, #10
...
```



System Memory Map



	Memory Map of Cortex-M3	Memory Map of FPGA Fabric Master, Ethernet MAC, Peripheral DMA	
	System Registers		0xE0043000 – 0xFFFF2FFF
			0xE0042000 – 0xE0042FFF
			0x78000000 – 0xE0041FFF
	External Memory Type 1	External Memory Type 1	0x74000000 – 0x77FFFFFF
	External Memory Type 0	External Memory Type 0	0x70000000 – 0x73FFFFFF
			0x601D0000 – 0x6FFFFFFF
			0x60180000 – 0x601CFFFF
			0x60100100 – 0x6017FFFF
	eNVM Controller	eNVM Controller	0x60100000 – 0x601000FF
			0x60088200 – 0x600FFFFFFF
	eNVM Aux Block (spare pages)	eNVM Aux Block (spare pages)	0x60088000 – 0x600881FF
	eNVM Aux Block (array)	eNVM Aux Block (array)	0x60084000 – 0x60087FFF
	eNVM Spare Pages	eNVM Spare Pages	0x60080000 – 0x60083FFF
	eNVM Array	eNVM Array	0x60000000 – 0x6007FFFF
Peripheral Bit-Band Alias Region of Cortex-M3	Peripherals (BB view)		0x44000000 – 0x5FFFFFFF
			0x42000000 – 0x43FFFFFF
	FPGA Fabric	FPGA Fabric	0x40100000 – 0x41FFFFFF
	FPGA Fabric eSRAM Backdoor	FPGA Fabric eSRAM Backdoor	0x40050000 – 0x400FFFFF
			0x40040000 – 0x4004FFFF
		APB Extension Register	0x40030004 – 0x4003FFFF
	Analog Compute Engine	Analog Compute Engine	0x40030000 – 0x40030003
			0x40020000 – 0x4002FFFF
			0x40017000 – 0x4001FFFF
			0x40016000 – 0x40016FFF
	IAP Controller	IAP Controller	0x40015000 – 0x40015FFF
	eFROM	eFROM	0x40014000 – 0x40014FFF
	RTC	RTC	0x40013000 – 0x40013FFF
	MSS GPIO	MSS GPIO	0x40012000 – 0x40012FFF
	I2C_1	I2C_1	0x40011000 – 0x40011FFF
	SPI_1	SPI_1	0x40010000 – 0x40010FFF
	UART_1	UART_1	0x40008000 – 0x4000FFFF
	Fabrik Interface Interrupt Controller	Fabrik Interface Interrupt Controller	0x40007000 – 0x40007FFF
	Watchdog	Watchdog	0x40006000 – 0x40006FFF
	Timer	Timer	0x40005000 – 0x40005FFF
	Peripheral DMA	Peripheral DMA	0x40004000 – 0x40004FFF
	Ethernet MAC	Ethernet MAC	0x40003000 – 0x40003FFF
	I2C_0	I2C_0	0x40002000 – 0x40002FFF
	SPI_0	SPI_0	0x40001000 – 0x40001FFF
	UART_0	UART_0	0x40000000 – 0x40000FFF
			0x24000000 – 0x3FFFFFFF
	eSRAM_0 / eSRAM_1 (BB view)		0x22000000 – 0x23FFFFFF
			0x20010000 – 0x21FFFFFF
	eSRAM_1	eSRAM_1	0x20008000 – 0x2000FFFF
	eSRAM_0	eSRAM_0	0x20000000 – 0x20007FFF
			0x00088200 – 0x1FFFFFFF
			0x000881FF
	eNVM (Cortex-M3) Virtual View	eNVM (fabrik) Virtual View	0x00000000

Visible only to FPGA Fabric Master

Visible only to FPGA Fabric Master

Figure 2-4 • System Memory Map with 64 Kbytes of SRAM

Accessing memory locations from C



- Memory has an address and value
- Can equate a pointer to desired address
- Can set/get de-referenced value to change memory

```
#define  SYSREG_SOFT_RST_CR  0xE0042030

uint32_t *reg = (uint32_t *) (SYSREG_SOFT_RST_CR);

main () {
    *reg |= 0x00004000; // Reset GPIO hardware
    *reg &= ~(0x00004000);
}
```



Some useful C keywords

- **const**
 - Makes variable value or pointer parameter unmodifiable
 - `const foo = 32;`
- **register**
 - Tells compiler to locate variables in a CPU register if possible
 - `register int x;`
- **static**
 - Preserve variable value after its scope ends
 - Does not go on the stack
 - `static int x;`
- **volatile**
 - Opposite of `const`
 - Can be changed in the background
 - `volatile int I;`

What happens when **this** “instruction” executes?



```
#include <stdio.h>
#include <inttypes.h>

#define REG_FOO 0x40000140

main () {
    uint32_t *reg = (uint32_t *) (REG_FOO);
    *reg += 3;

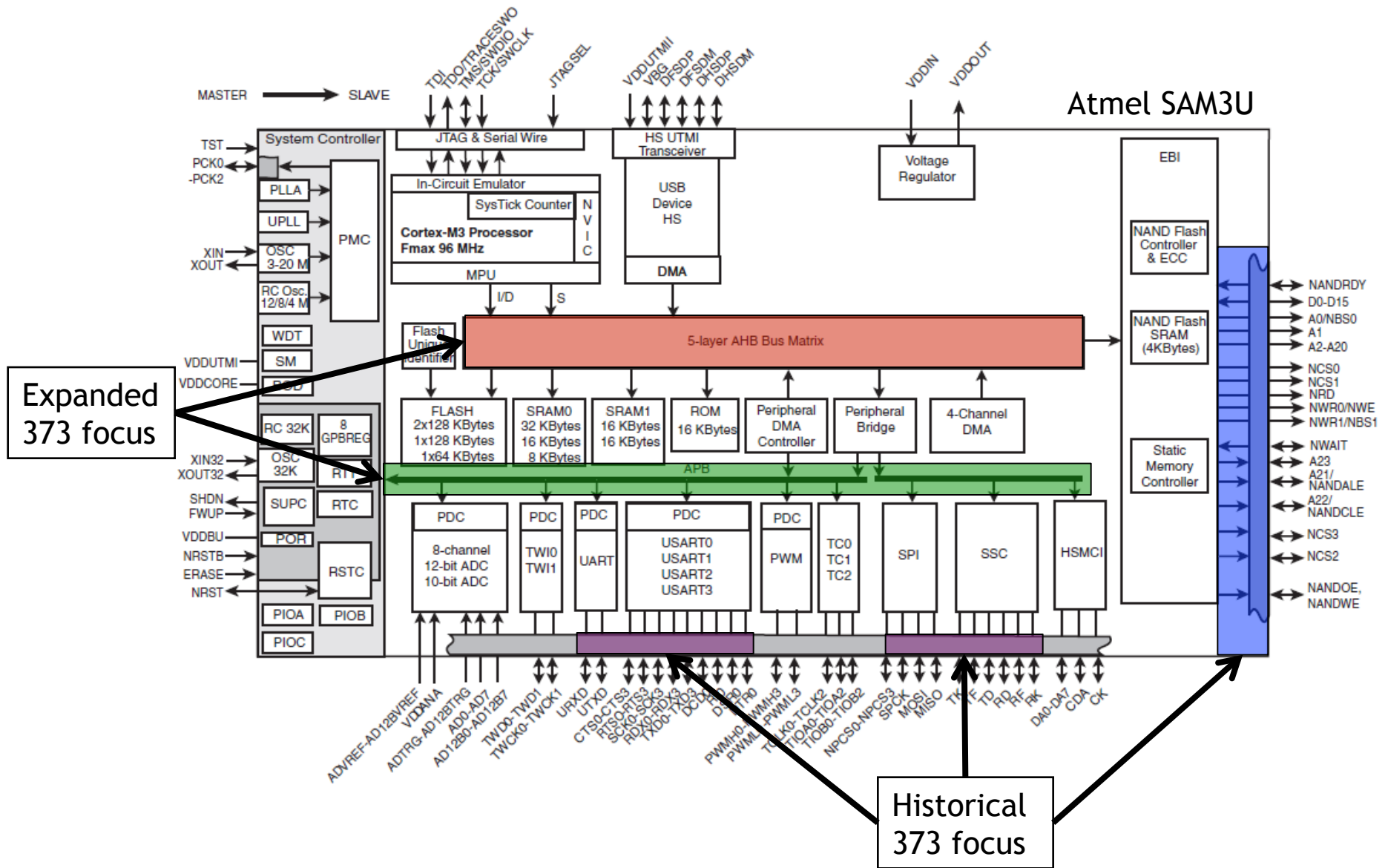
    printf("0x%x\n", *reg); // Prints out new value
}
```

“*reg += 3” is turned into a ld, add, str sequence



- Load instruction
 - A bus read operation commences
 - The CPU drives the address “reg” onto the address bus
 - The CPU indicated a read operation is in process (e.g. R/W#)
 - Some “handshaking” occurs
 - The target drives the contents of “reg” onto the data lines
 - The contents of “reg” is loaded into a CPU register (e.g. r0)
- Add instruction
 - An immediate add (e.g. add r0, #3) adds three to this value
- Store instruction
 - A bus write operation commences
 - The CPU drives the address “reg” onto the address bus
 - The CPU indicated a write operation is in process (e.g. R/W#)
 - Some “handshaking” occurs
 - The CPU drives the contents of “r0” onto the data lines
 - The target stores the data value into address “reg”

Modern embedded systems have multiple busses





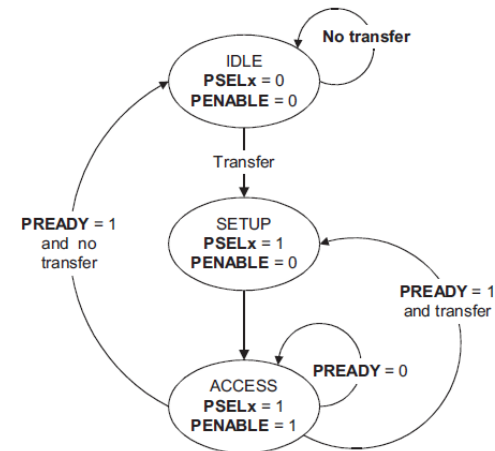
Why have so many busses?

- Many designs considerations
 - Master vs Slave
 - Internal vs External
 - Bridged vs Flat
 - Memory vs Peripheral
 - Synchronous vs Asynchronous
 - High-speed vs low-speed
 - Serial vs Parallel
 - Single master vs multi master
 - Single layer vs multi layer
 - Multiplexed A/D vs demultiplexed A/D
- Discussion: what are some of the tradeoffs?

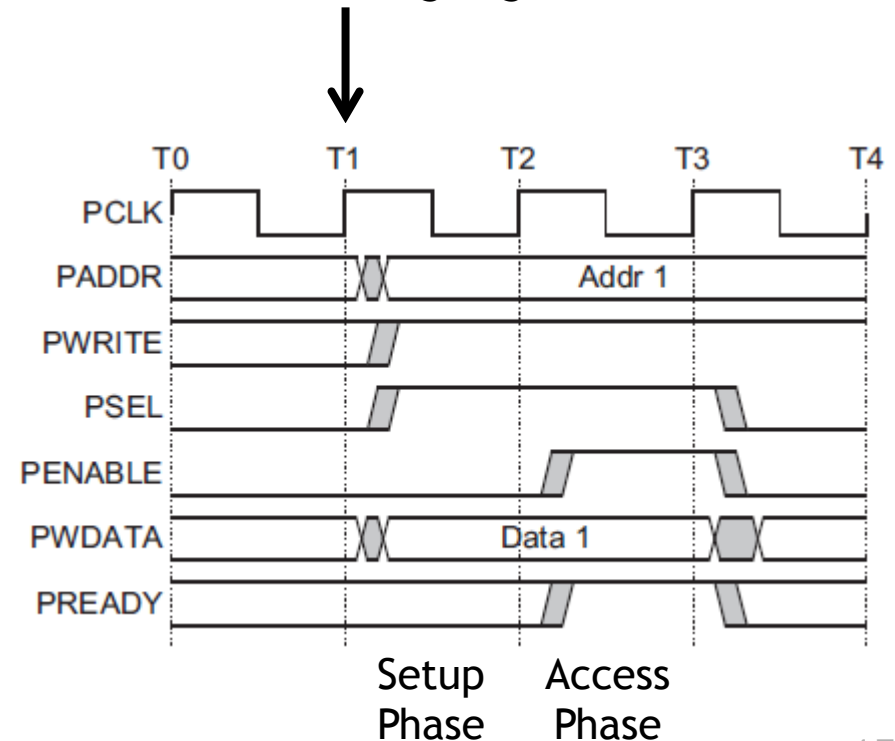
APB



- IDLE
 - Default APB state
- SETUP
 - When transfer required
 - PSELx is asserted
 - Only one cycle
- ACCESS
 - PENABLE is asserted
 - Addr, write, select, and write data remain stable
 - Stay if PREADY = L
 - Goto IDLE if PREADY = H and no more data
 - Goto SETUP is PREADY = H and more data pending



Setup phase begins with this rising edge



APB signal definitions



- PCLK: the bus clock source (rising-edge triggered)
- PRESETn: the bus (and typically system) reset signal (active low)
- PADDR: the APB address bus (can be up to 32-bits wide)
- PSELx: the select line for each slave device
- PENABLE: indicates the 2nd and subsequent cycles of an APB xfer
- PWRITE: indicates transfer direction (Write=H, Read=L)
- PWDATA: the write data bus (can be up to 32-bits wide)
- PREADY: used to extend a transfer
- PRDATA: the read data bus (can be up to 32-bits wide)
- PSLVERR: indicates a transfer error (OKAY=L, ERROR=H)

Let's say we want a device that provides data from a switch on a read to any address it is assigned.
(so returns a 0 or 1)



PREADY

PRDATA[32:0]

PWRITE

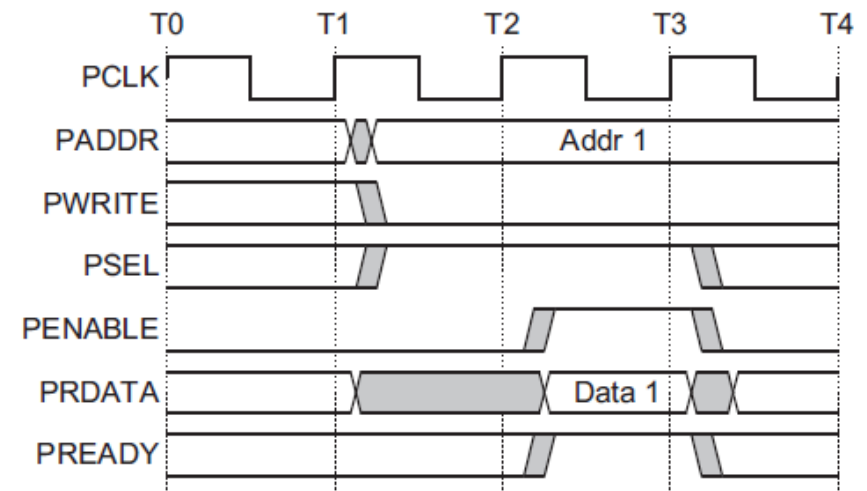
PENABLE

PSEL

PADDR[7:0]

PCLK

Mr.
Switch



Device provides data from switch A if address 0x00001000 is read from. B if address 0x00001004 is read from



PREADY

PRDATA[32:0]

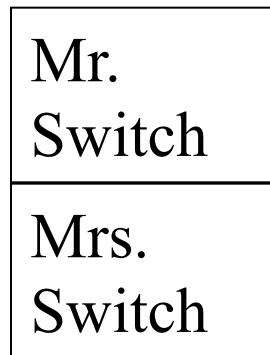
PWRITE

PENABLE

PSEL

PADDR[7:0]

PCLK



All reads read from register, all writes write...



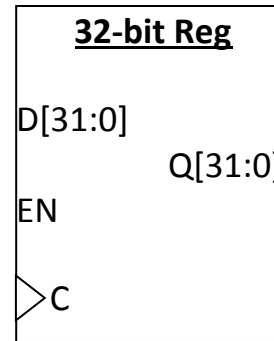
PWDATA[31:0]

PREADY

PWRITE

PRDATA[32:0]

PENABLE

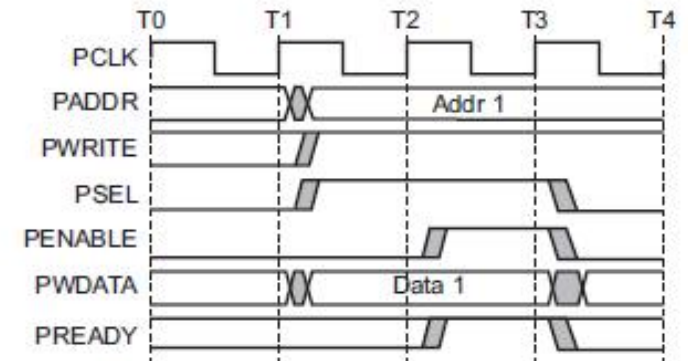


PSEL

PADDR[7:0]

PCLK

PREADY



We are assuming APB only gets lowest 8 bits of address here...

Outline

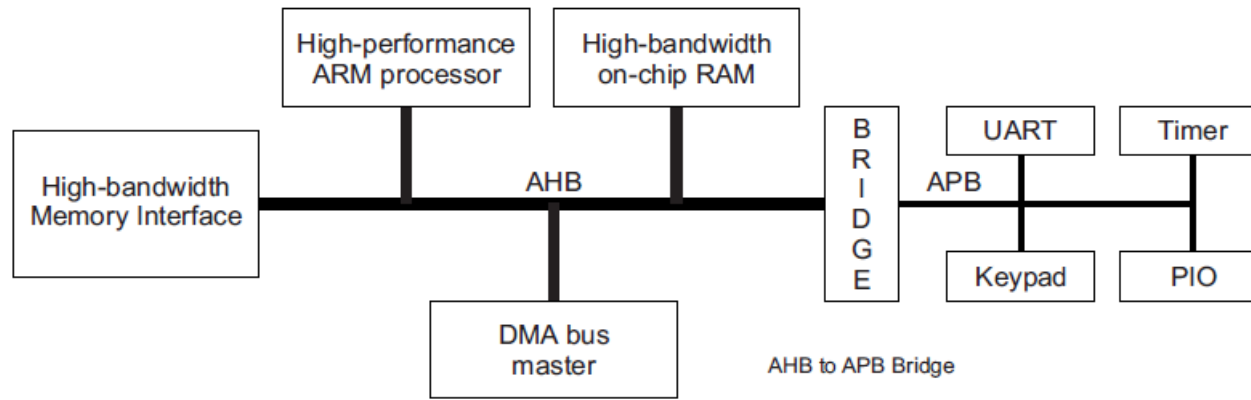


- Announcements
- Review
- ARM AHB-Lite



Advanced Microcontroller Bus Architecture (AMBA)

- Advanced High-performance Bus (AHB)
- Advanced Peripheral Bus (APB)



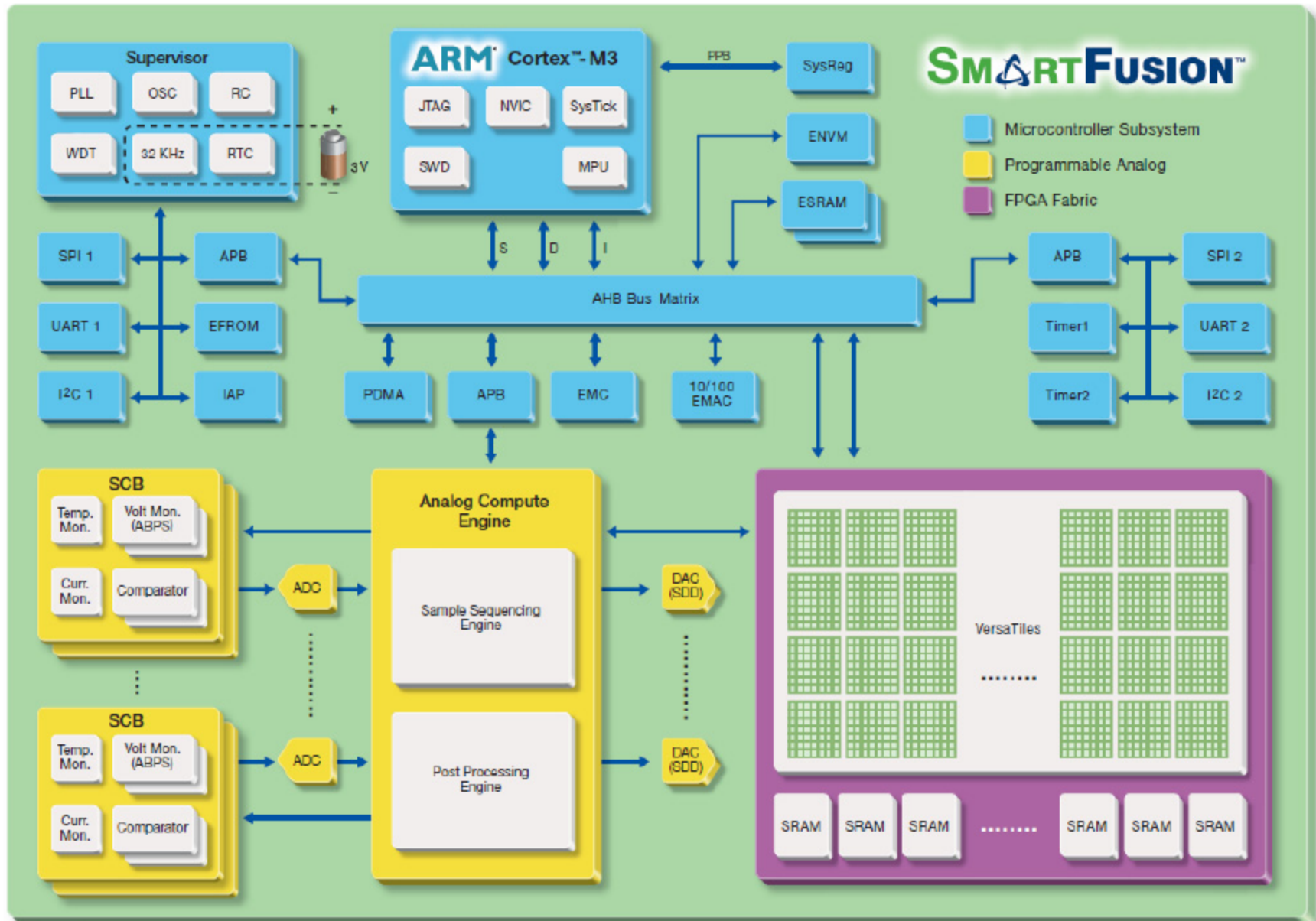
AHB

- High performance
- Pipelined operation
- Burst transfers
- Multiple bus masters
- Split transactions

APB

- Low power
- Latched address/control
- Simple interface
- Suitable of many peripherals

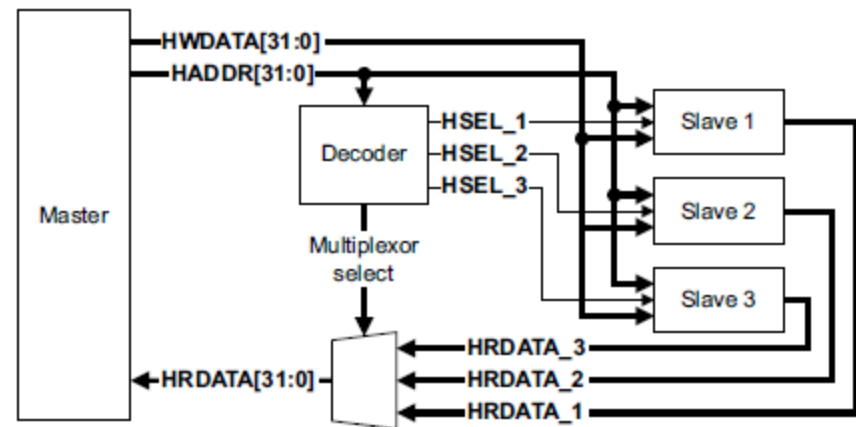
Actel SmartFusion system/bus architecture





AHB-Lite supports single bus master and provides high-bandwidth operation

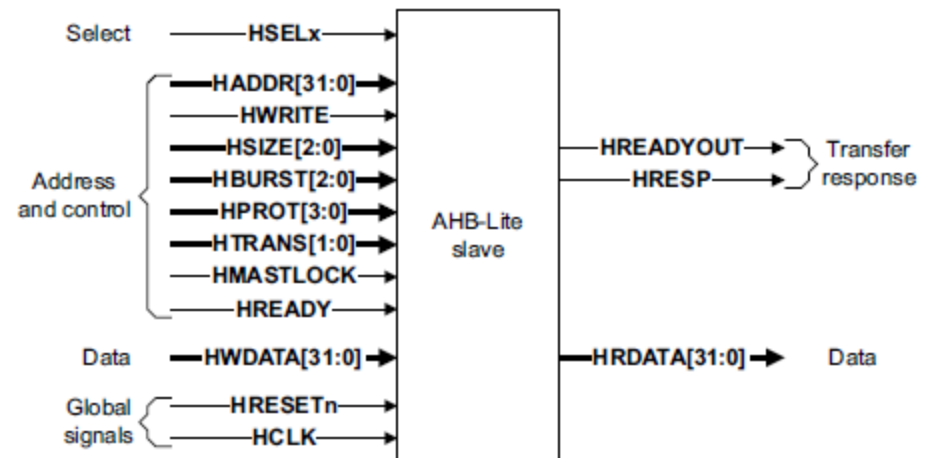
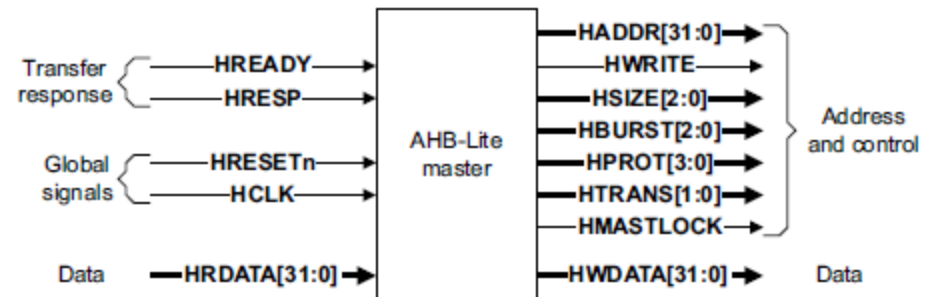
- Burst transfers
- Single clock-edge operation
- Non-tri-state implementation
- Configurable bus width



AHB-Lite bus master/slave interface



- Global signals
 - HCLK
 - HRESETn
- Master out/slave in
 - HADDR (address)
 - HWDATA (write data)
 - Control
 - HWRITE
 - HSIZE
 - HBURST
 - HPROT
 - HTRANS
 - HMASTLOCK
- Slave out/master in
 - HRDATA (read data)
 - HREADY
 - HRESP



AHB-Lite signal definitions



- Global signals
 - HCLK: the bus clock source (rising-edge triggered)
 - HRESETn: the bus (and system) reset signal (active low)
- Master out/slave in
 - HADDR[31:0]: the 32-bit system address bus
 - HWDATA[31:0]: the system write data bus
 - Control
 - HWRITE: indicates transfer direction (Write=1, Read=0)
 - HSIZE[2:0]: indicates size of transfer (byte, halfword, or word)
 - HBURST[2:0]: indicates single or burst transfer (1, 4, 8, 16 beats)
 - HPROT[3:0]: provides protection information (e.g. I or D; user or handler)
 - HTRANS: indicates current transfer type (e.g. idle, busy, nonseq, seq)
 - HMASTLOCK: indicates a locked (atomic) transfer sequence
- Slave out/master in
 - HRDATA[31:0]: the slave read data bus
 - HREADY: indicates previous transfer is complete
 - HRESP: the transfer response (OKAY=0, ERROR=1)

Key to timing diagram conventions

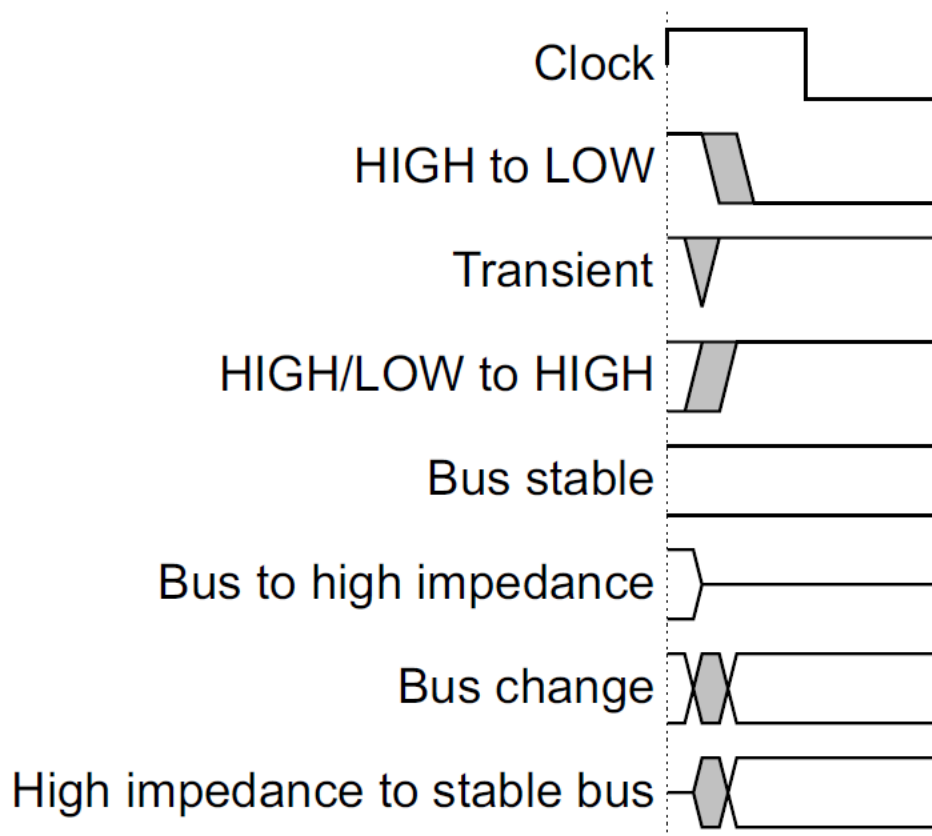


- Timing diagrams

- Clock
- Stable values
- Transitions
- High-impedance

- Signal conventions

- Lower case 'n' denote active low (e.g. RESETn)
- Prefix 'H' denotes AHB
- Prefix 'P' denotes APB



Basic read and write transfers with no wait states

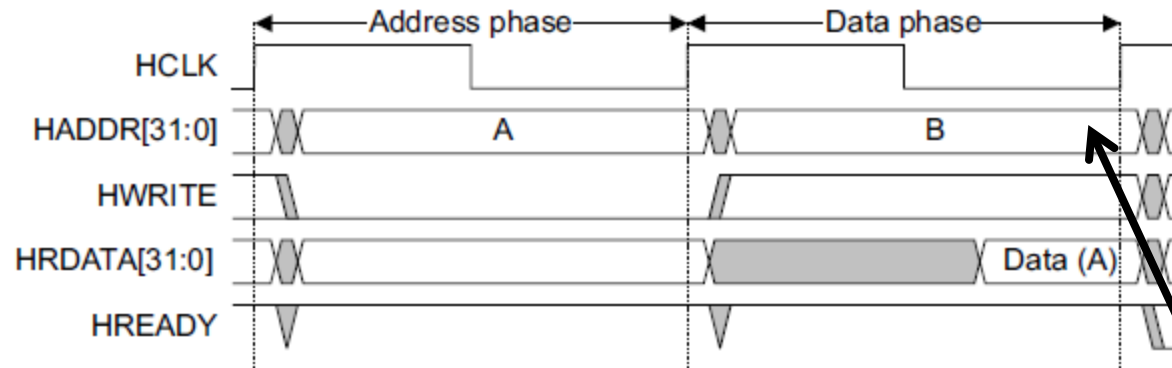


Figure 3-1 Read transfer

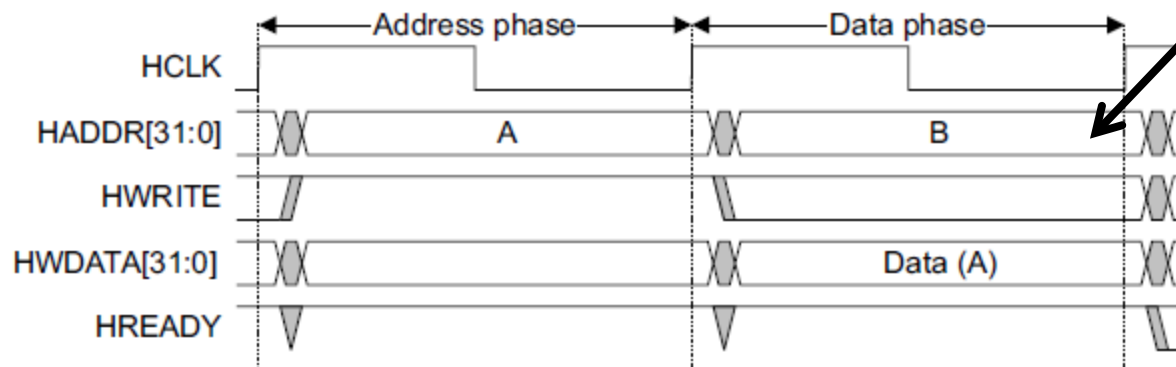
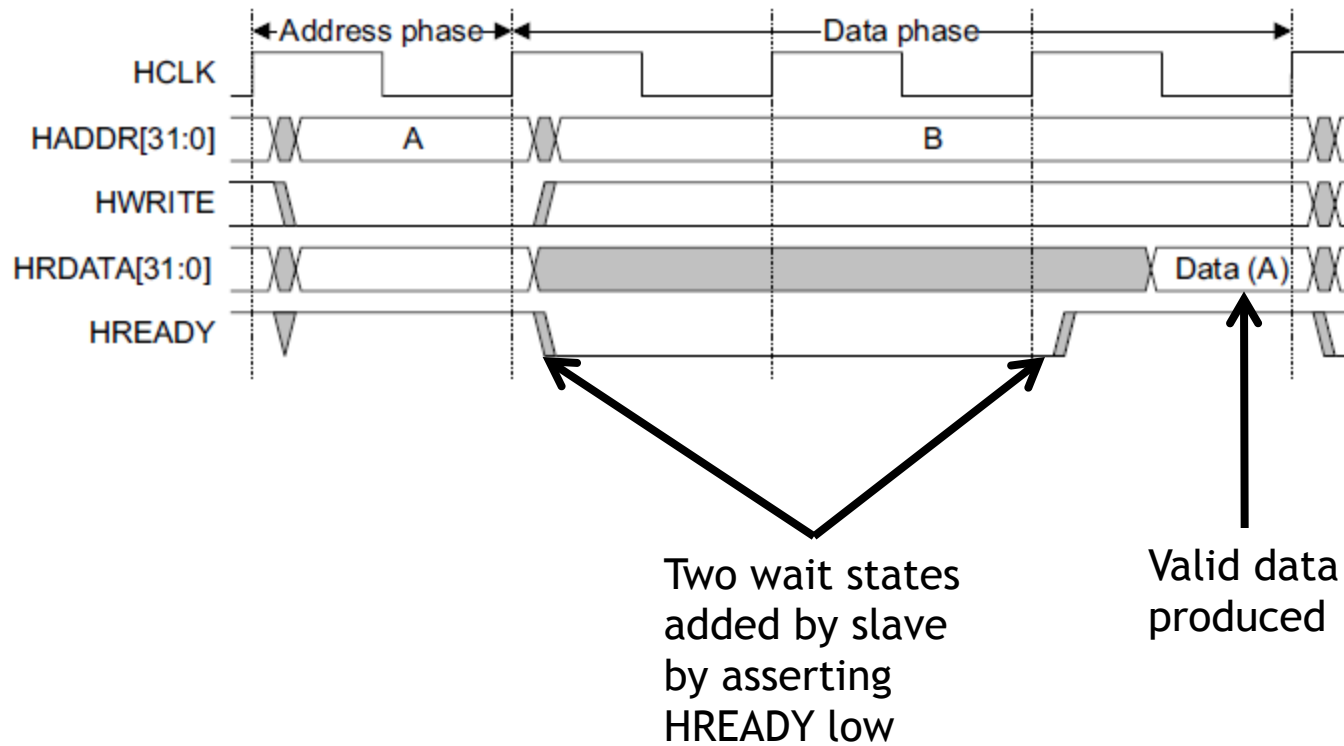


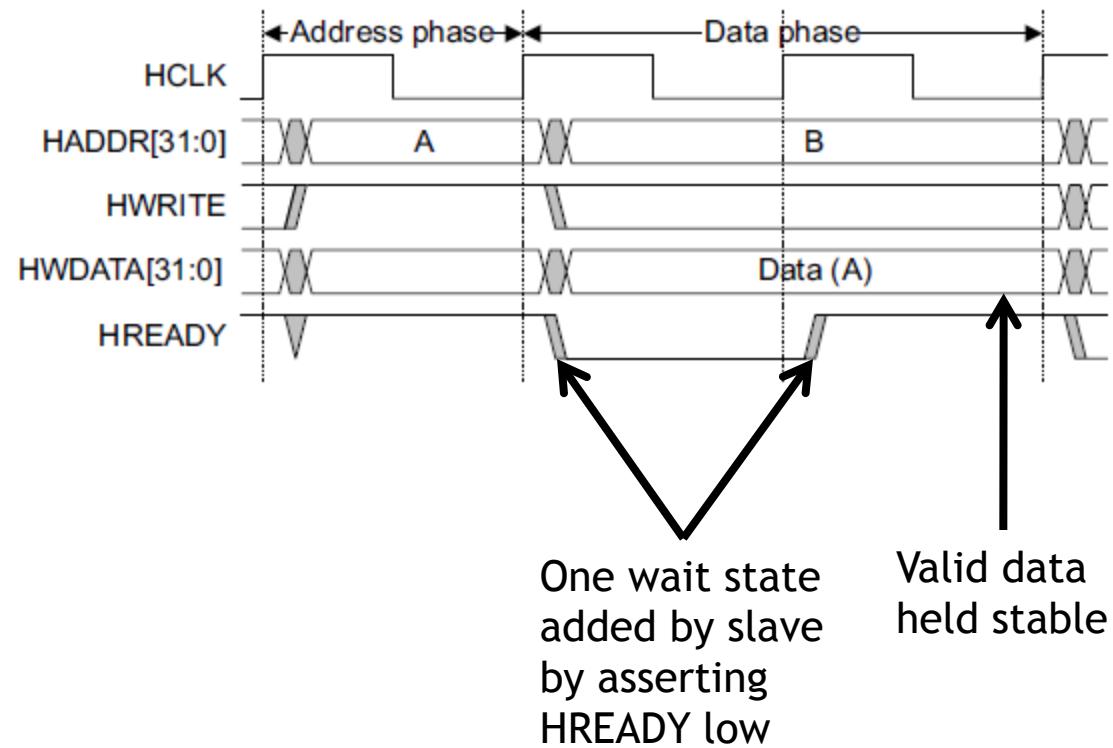
Figure 3-2 Write transfer

Pipelined
Address
& Data
Transfer

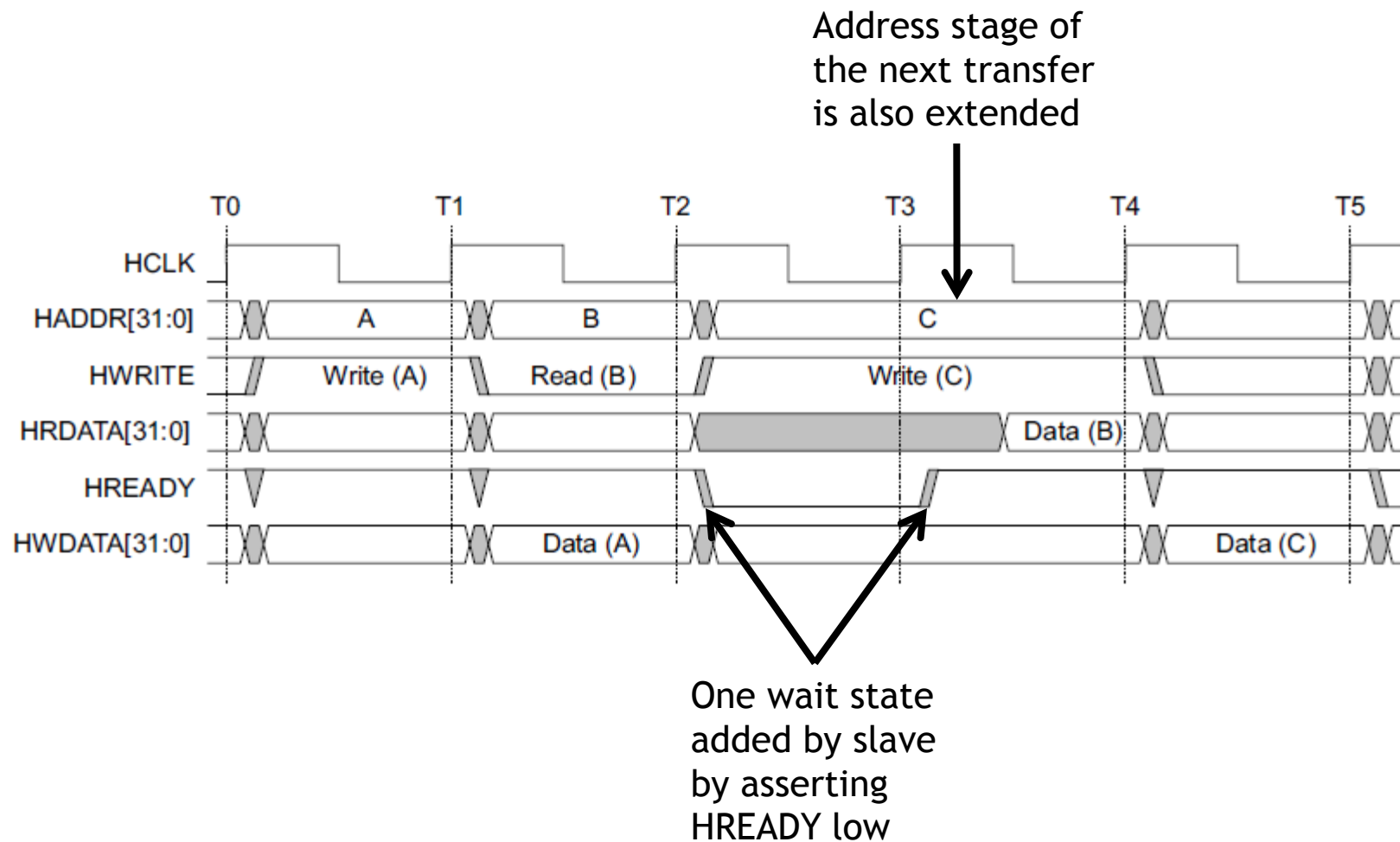
Read transfer with two wait states



Write transfer with one wait state



Wait states extend the address phase of next transfer

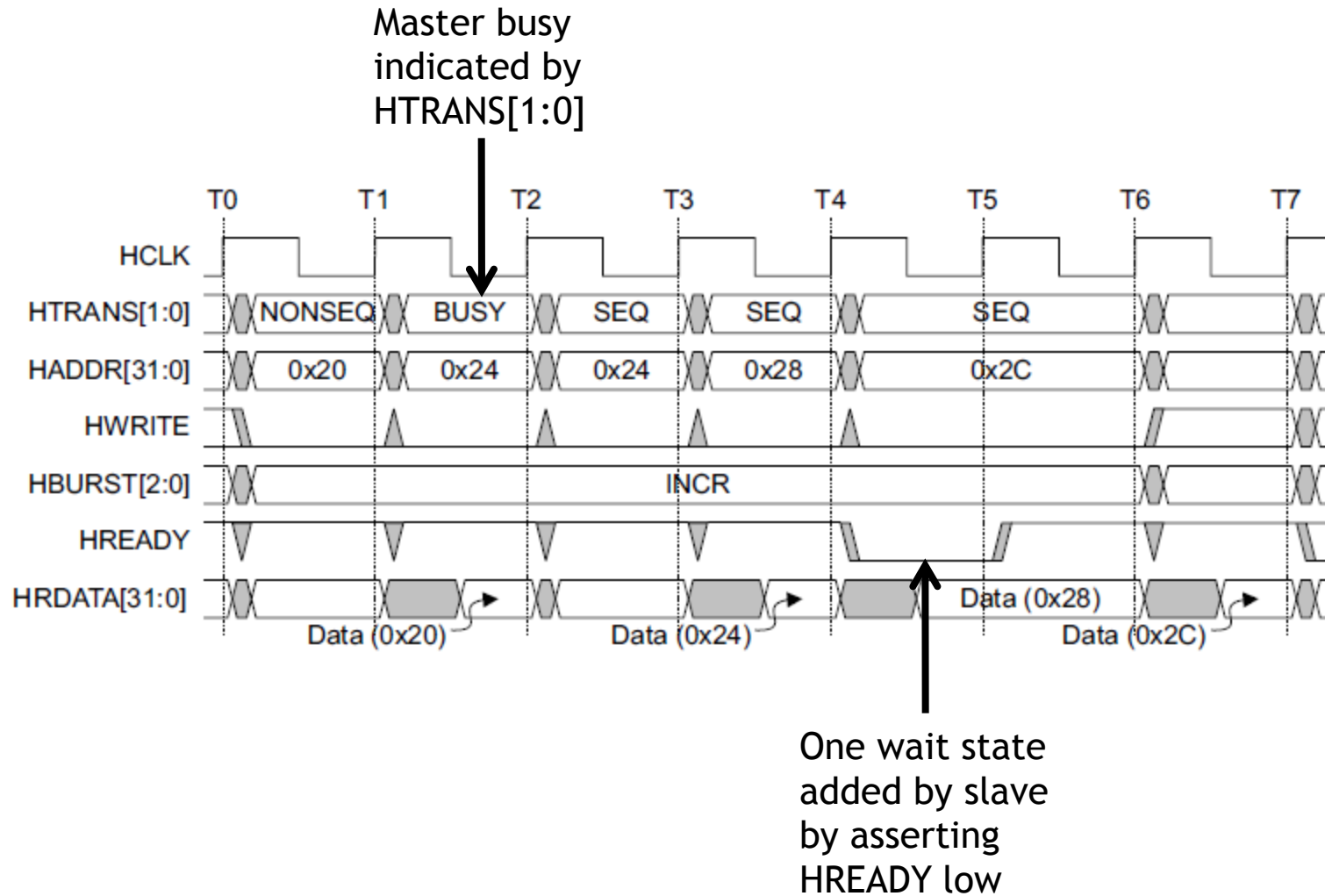


Transfers can be of four types (HTRANS[1:0])



- IDLE (b00)
 - No data transfer is required
 - Slave must OKAY w/o waiting
 - Slave must ignore IDLE
- BUSY (b01)
 - Insert idle cycles in a burst
 - Burst will continue afterward
 - Address/control reflects next transfer in burst
 - Slave must OKAY w/o waiting
 - Slave must ignore BUSY
- NONSEQ (b10)
 - Indicates single transfer or first transfer of a burst
 - Address/control unrelated to prior transfers
- SEQ (b11)
 - Remaining transfers in a burst
 - Addr = prior addr + transfer size

A four beat burst with master busy and slave wait



Controlling the size (width) of a transfer



- HSIZE[2:0] encodes the size
- The cannot exceed the data bus width (e.g. 32-bits)
- HSIZE + HBURST is determines wrapping boundary for wrapping bursts
- HSIZE must remain constant throughout a burst transfer

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size (bits)	Description
0	0	0	8	Byte
0	0	1	16	Halfword
0	1	0	32	Word
0	1	1	64	Doubleword
1	0	0	128	4-word line
1	0	1	256	8-word line
1	1	0	512	-
1	1	1	1024	-

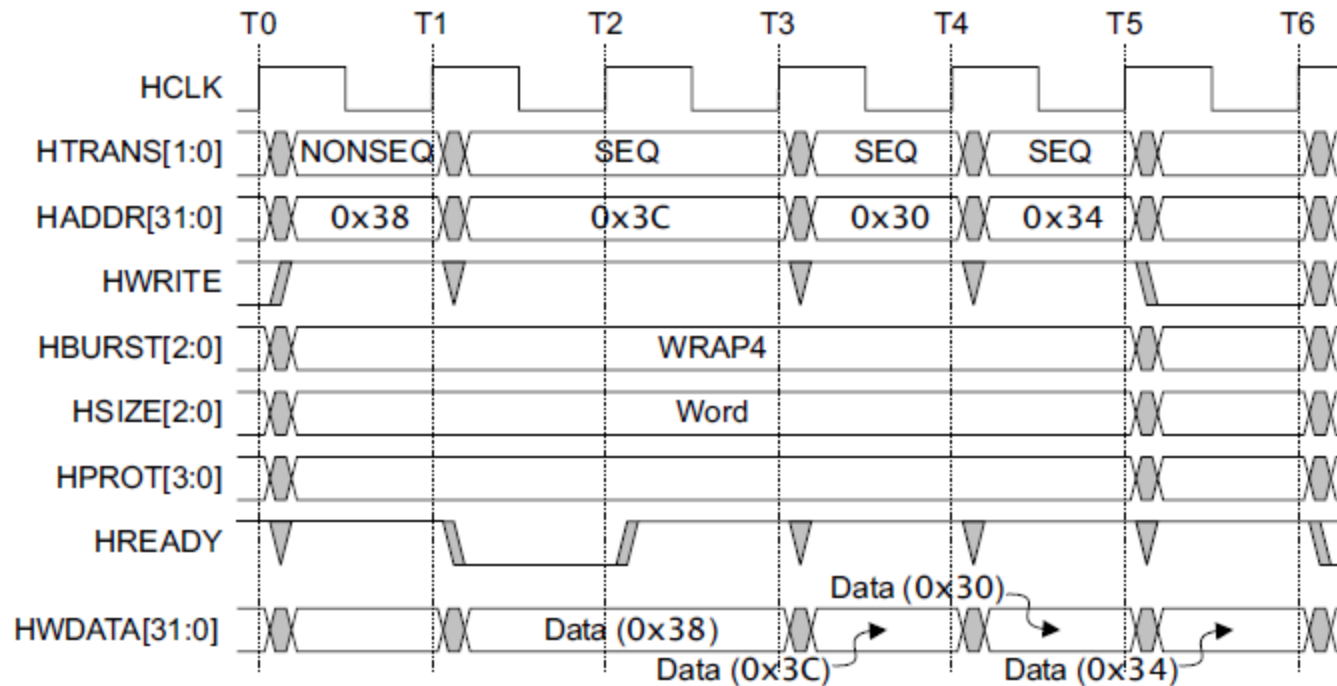
Controlling the burst beats (length) of a transfer



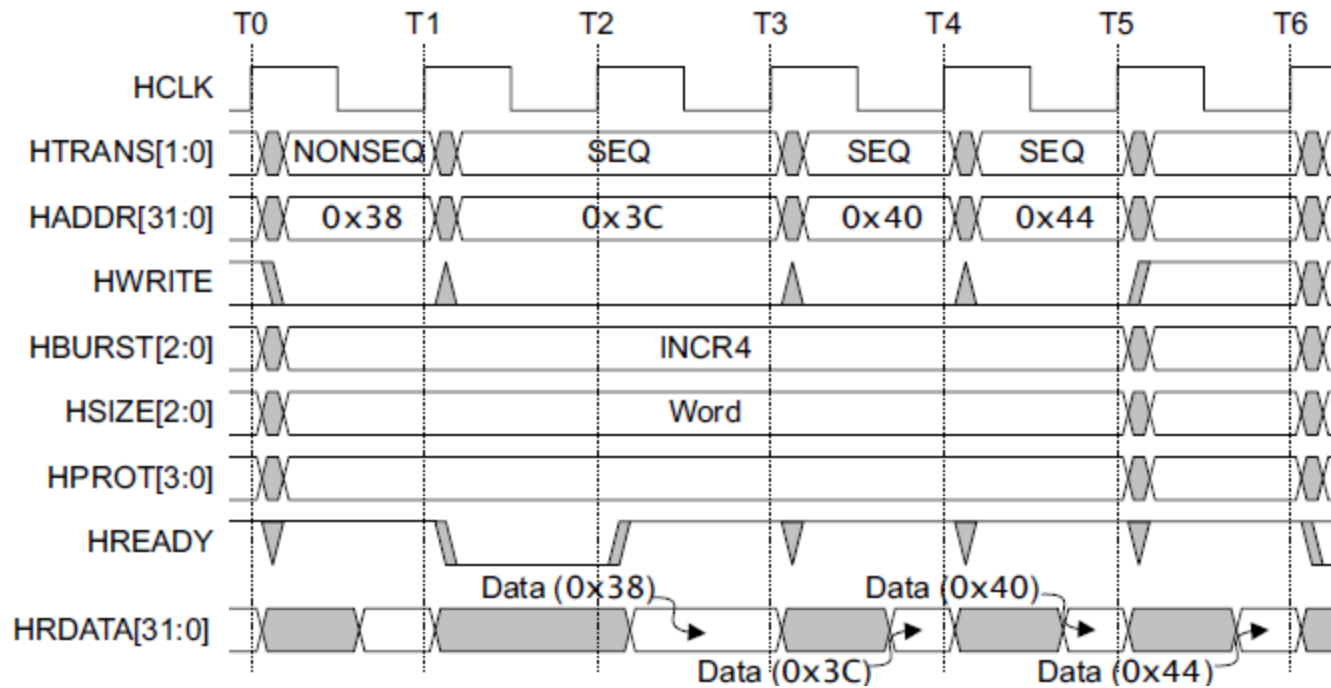
- Burst of 1, 4, 8, 16, and undef
- HBURST[2:0] encodes the type
- Incremental burst
- Wrapping bursts
 - 4 beats x 4-byte words wrapping
 - Wraps at 16 byte boundary
 - E.g. 0x34, 0x38, 0x3c, 0x30,...
- Bursts must not cross 1KB address boundaries

HBURST[2:0]	Type	Description
b000	SINGLE	Single burst
b001	INCR	Incrementing burst of undefined length
b010	WRAP4	4-beat wrapping burst
b011	INCR4	4-beat incrementing burst
b100	WRAP8	8-beat wrapping burst
b101	INCR8	8-beat incrementing burst
b110	WRAP16	16-beat wrapping burst
b111	INCR16	16-beat incrementing burst

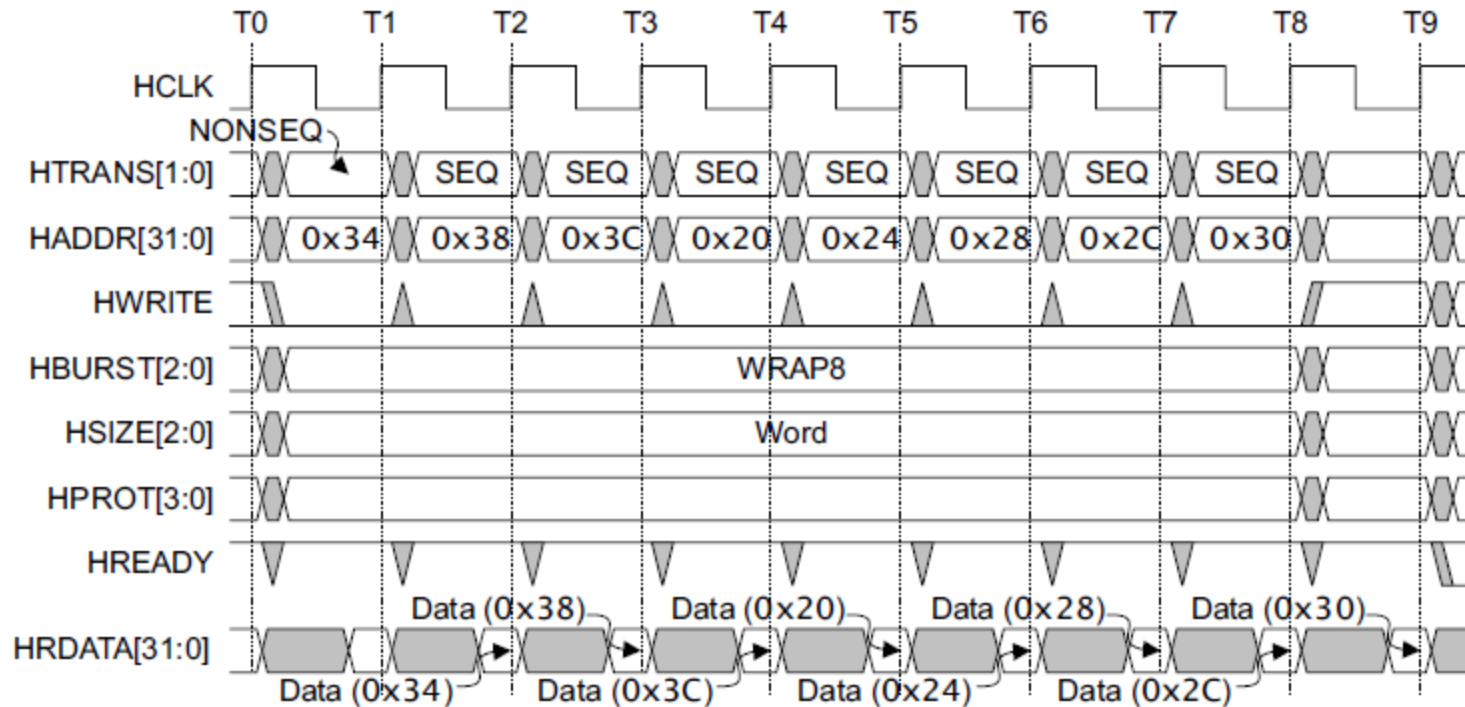
A four beat wrapping burst (WRAP4)



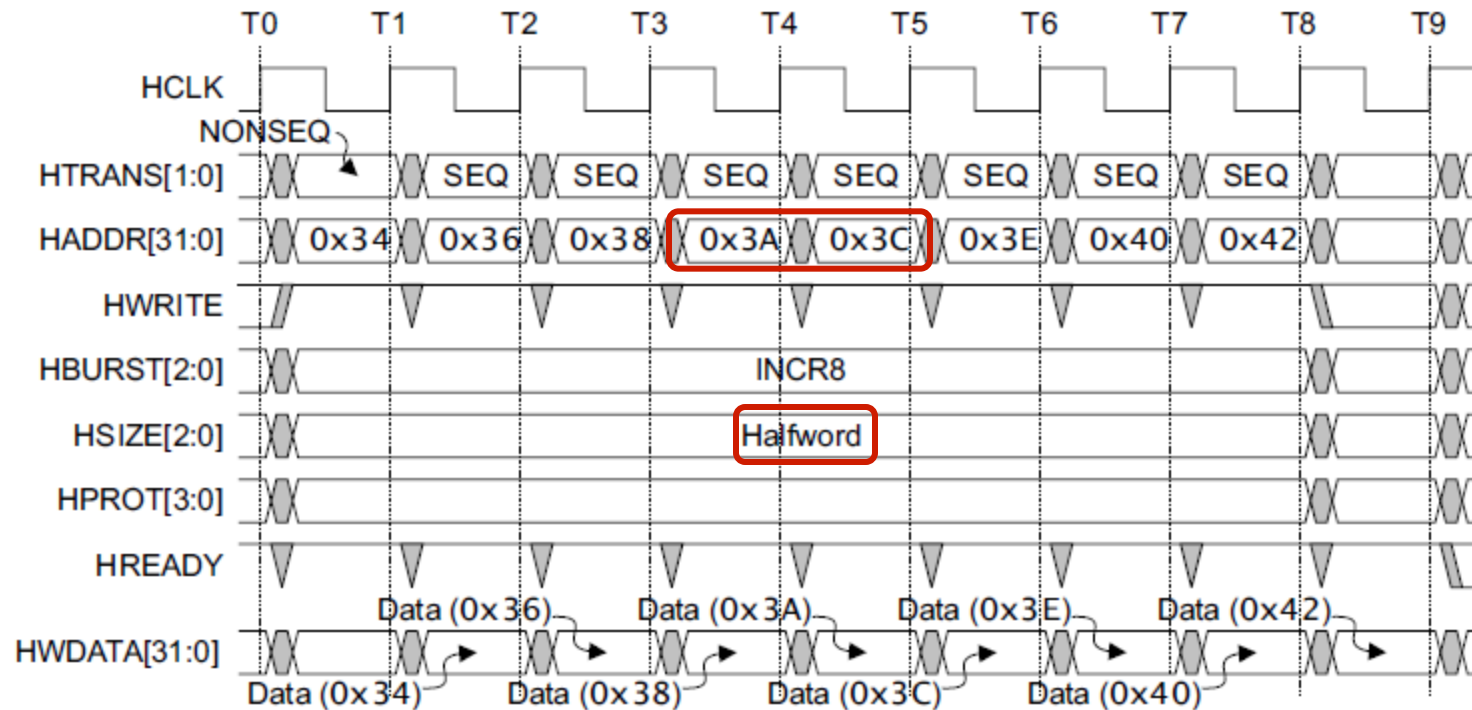
A four beat incrementing burst (INCR4)



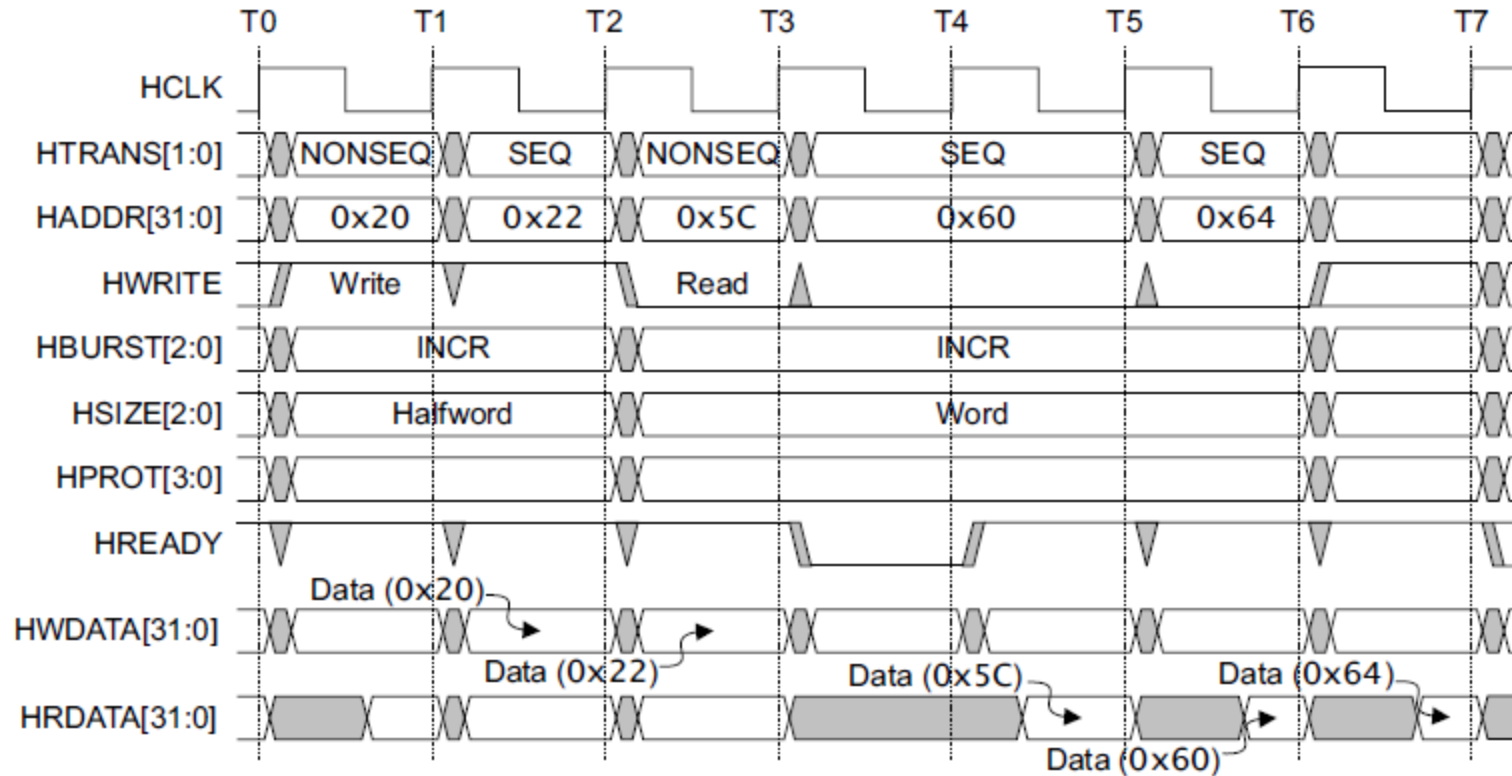
An eight beat wrapping burst (WRAP8)



An eight beat incrementing burst (INCR8) using half-word transfers



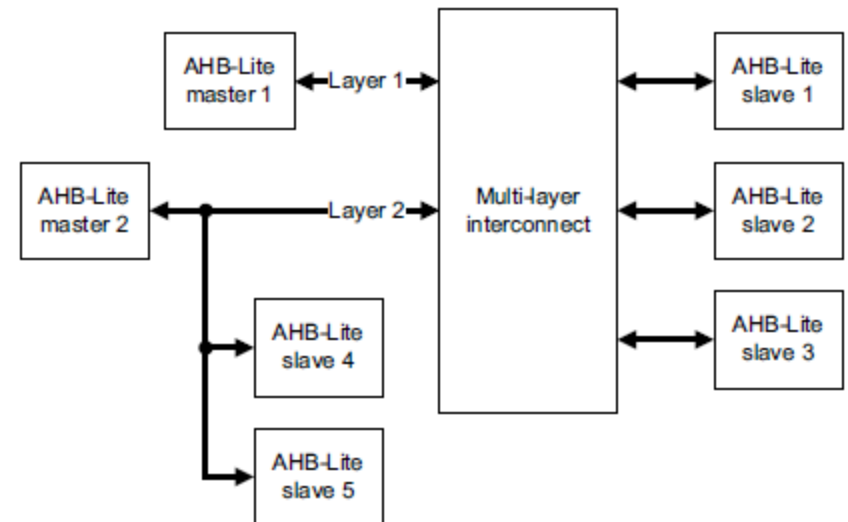
An undefined length incrementing burst (INCR)





Multi-master AHB-Lite requires a multi-layer interconnect

- AHB-Lite is single-master
- Multi-master operation
 - Must isolate masters
 - Each master assigned to layer
 - Interconnect arbitrates slave accesses
- Full crossbar switch often unneeded
 - Slaves 1, 2, 3 are shared
 - Slaves 4, 5 are local to Master 1





Questions?

Comments?

Discussion?