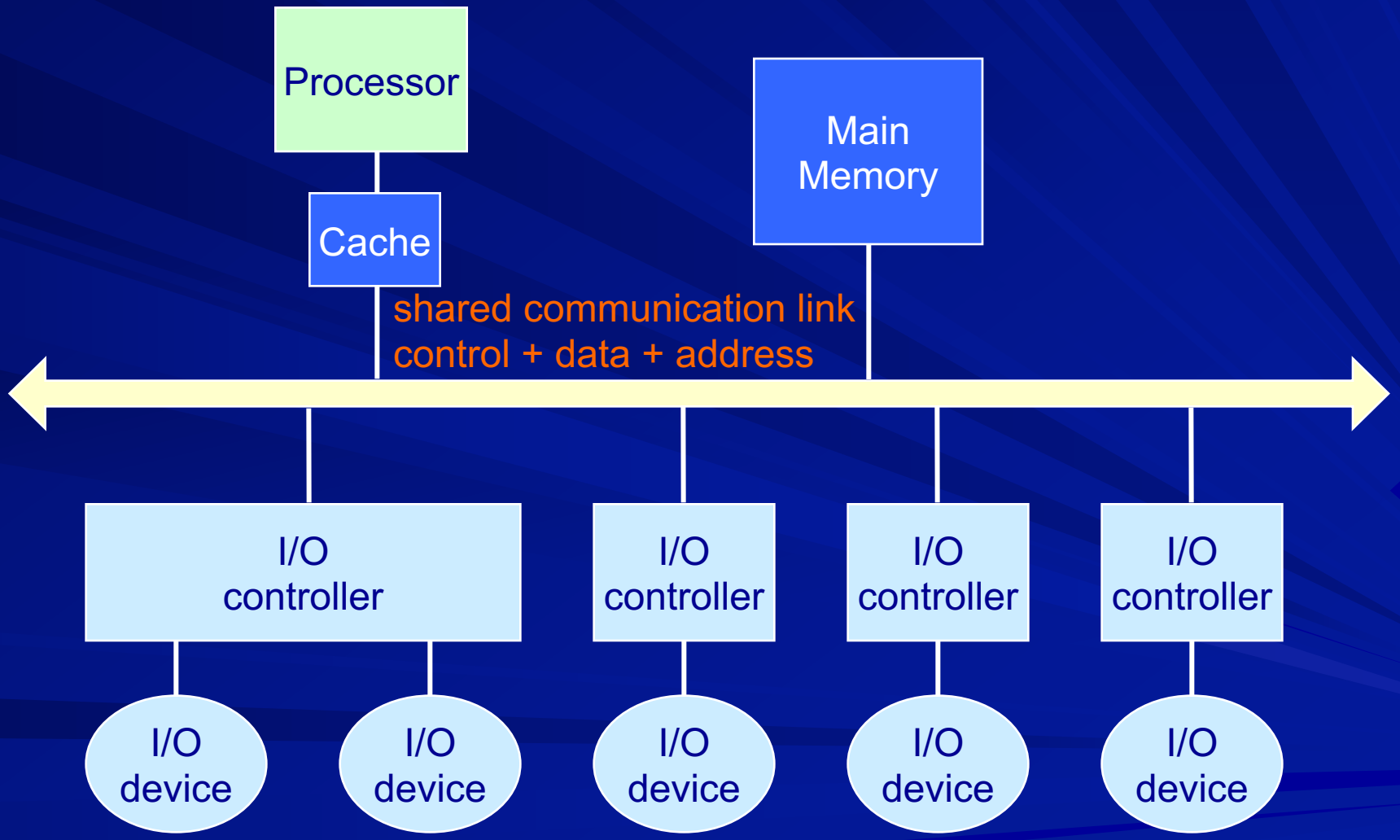
COL216
Computer Architecture

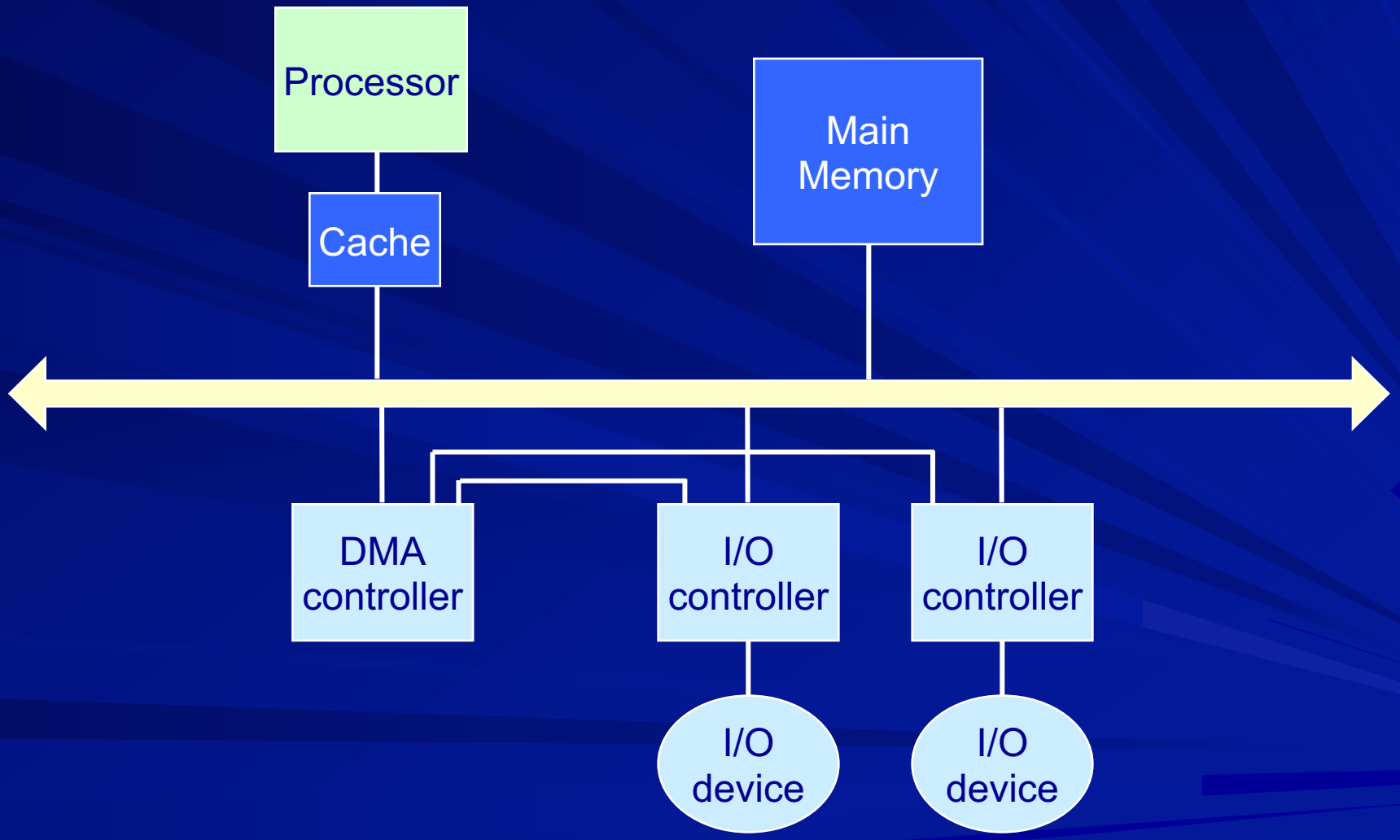
Input/Output – 4

24th March 2022

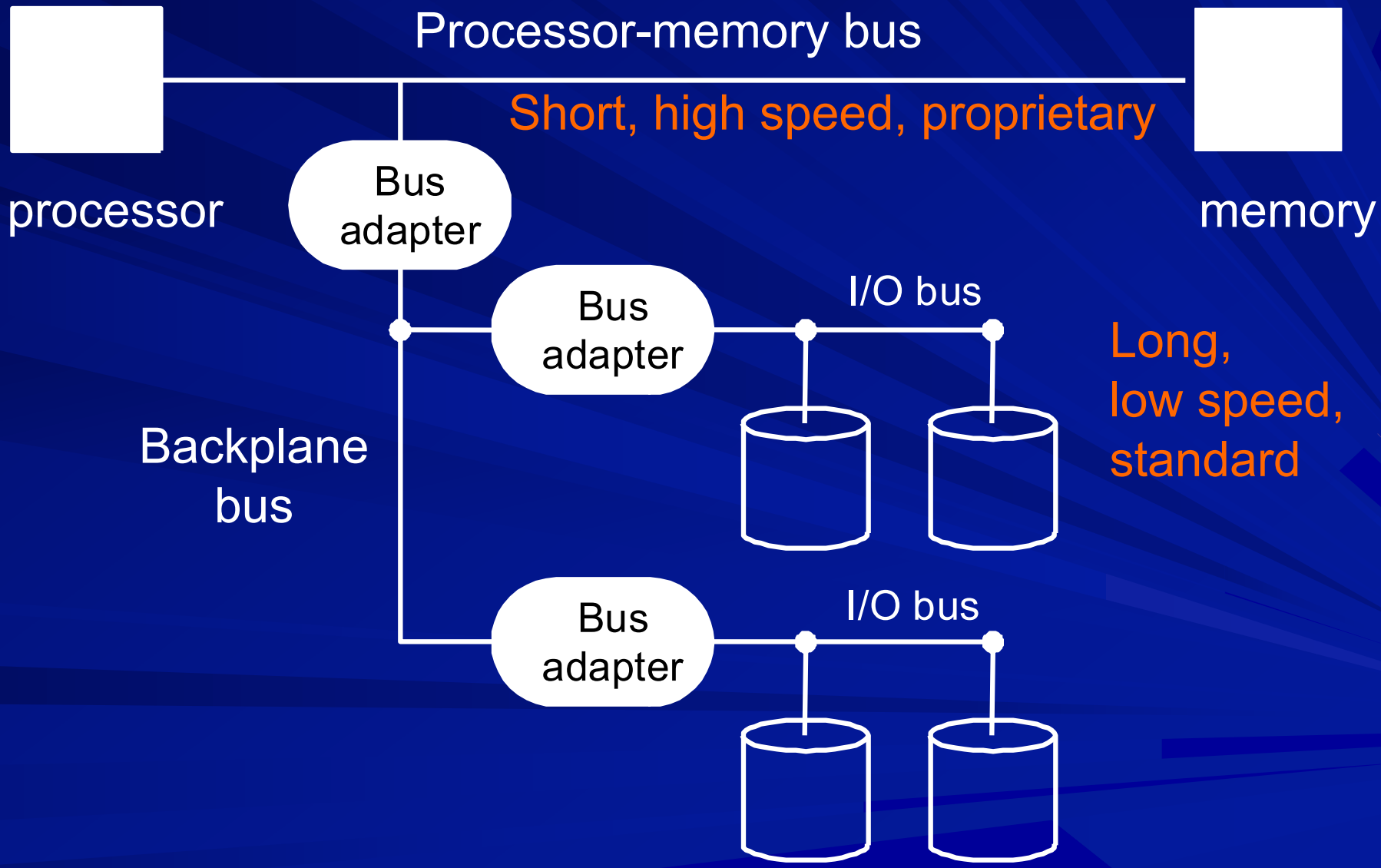
Single bus connecting all



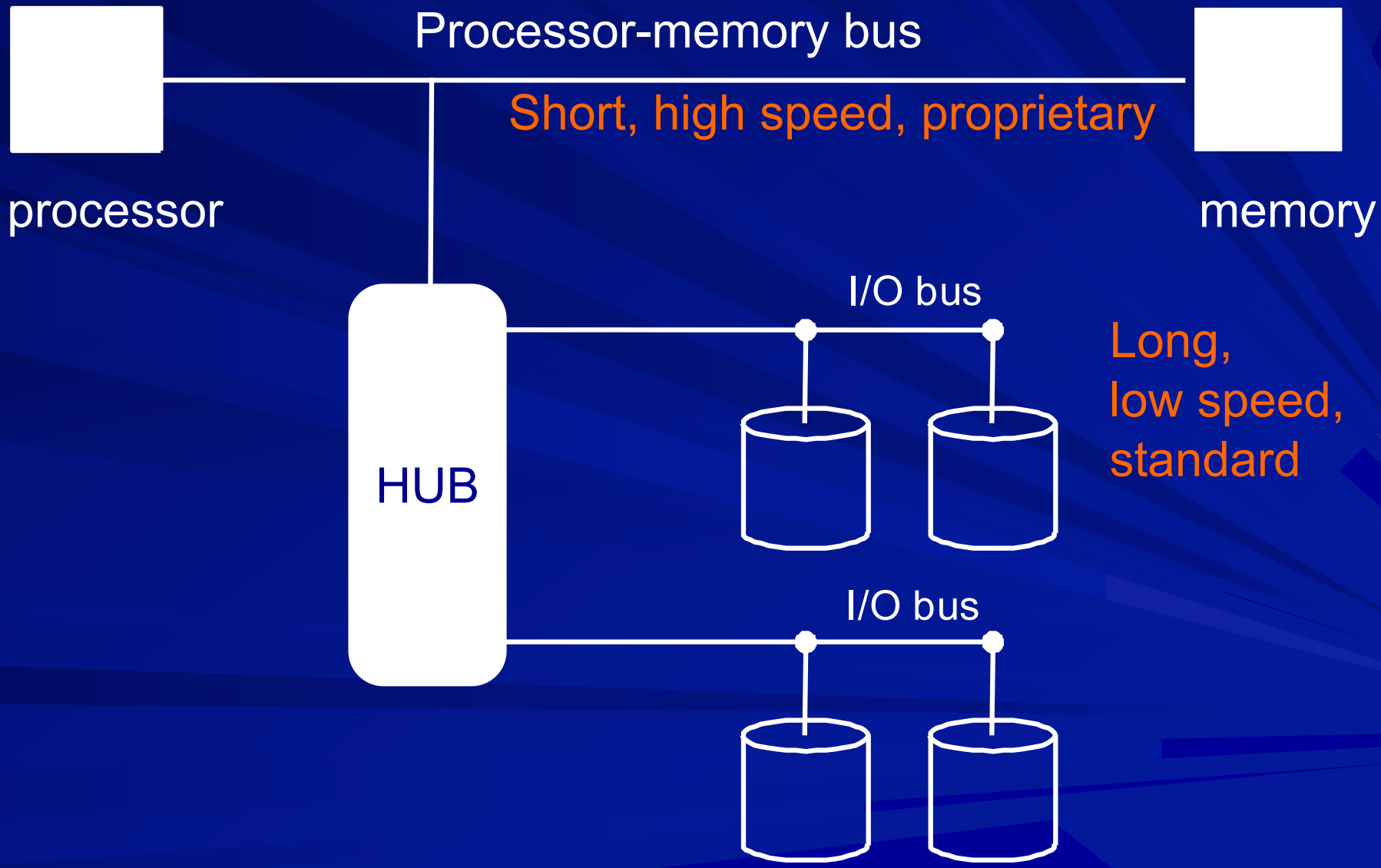
Direct memory access



System with multiple buses



System with multiple buses



Bus Standards

- Physical / mechanical
  - Pins, connectors, cables
- Electrical
  - Voltage / current levels, impedances
- Logical
  - Definition of signals
  - Timings and protocols

Standard buses/ports/interfaces

- Standardization required for subsystems from multiple sources to work together
- Technological developments lead to continuous improvement of specs, standardization implies freezing the specs
- Standards need to be revised and improved periodically
- Standardization is done by consortium of several organizations or professional bodies

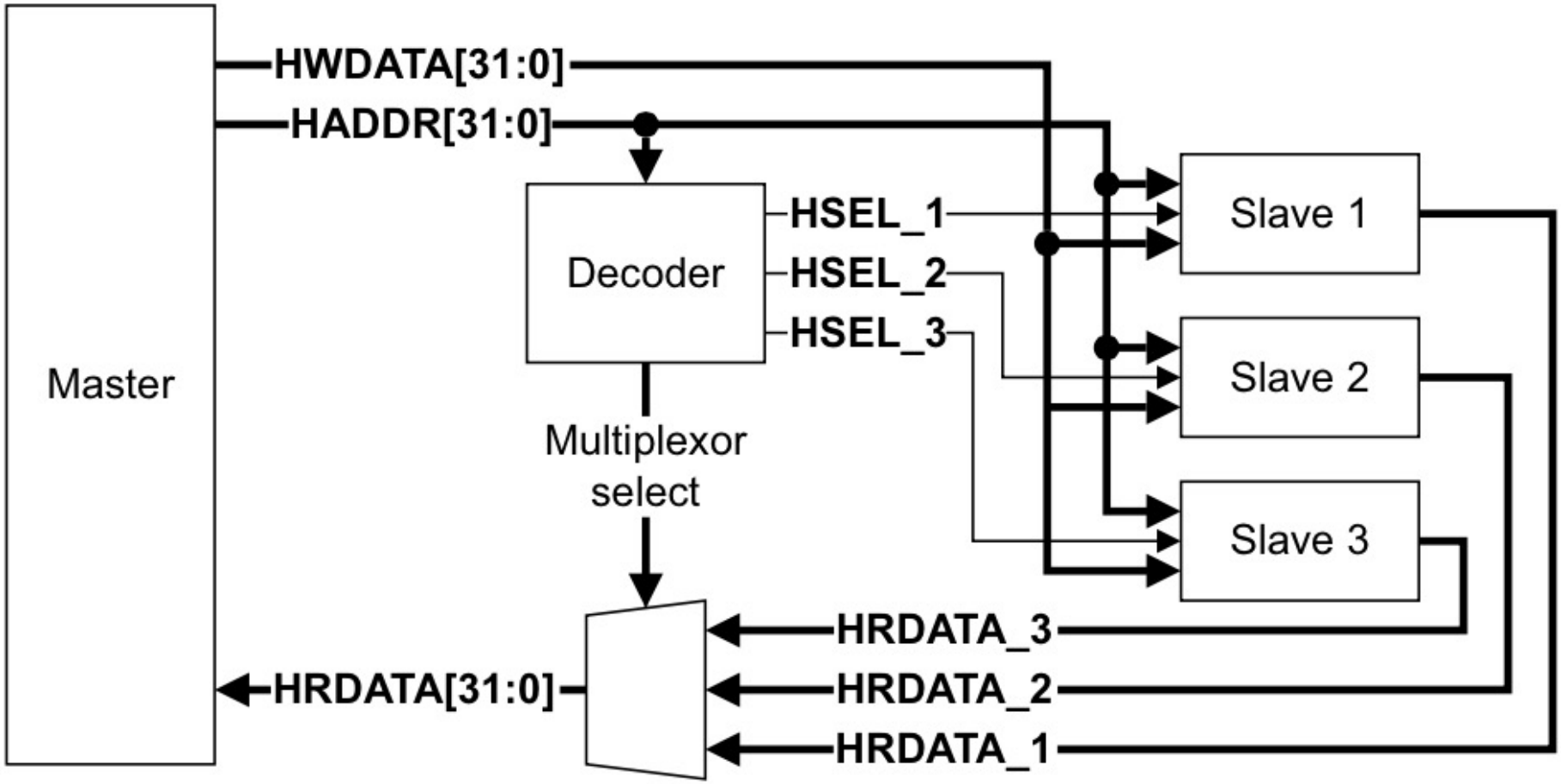
AMBA : Advanced Microcontroller Bus Architecture

- ARM open standard for on-chip buses
- For System-on-Chip (SoC)
- Variants
  - AHB : Advanced High speed Bus
  - APB : Advanced Peripheral Bus
  - AXI : Advanced eXtensible Interface
  - AHB-Lite : subset of AHB

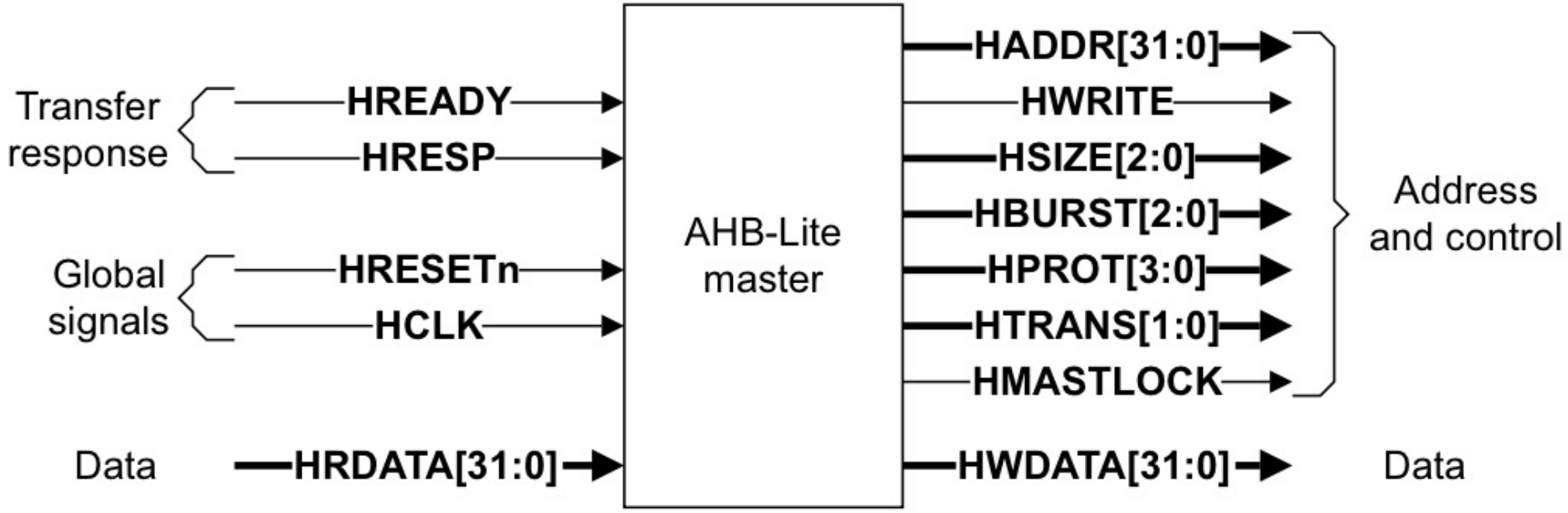
AHB-Lite

- Single master, multiple slaves
- Parallel
- Multiplexed (not tri-state)
- Separate data, address, control
- Synchronous
- Pipelined
- Burst transfer supported
- Split transaction not supported

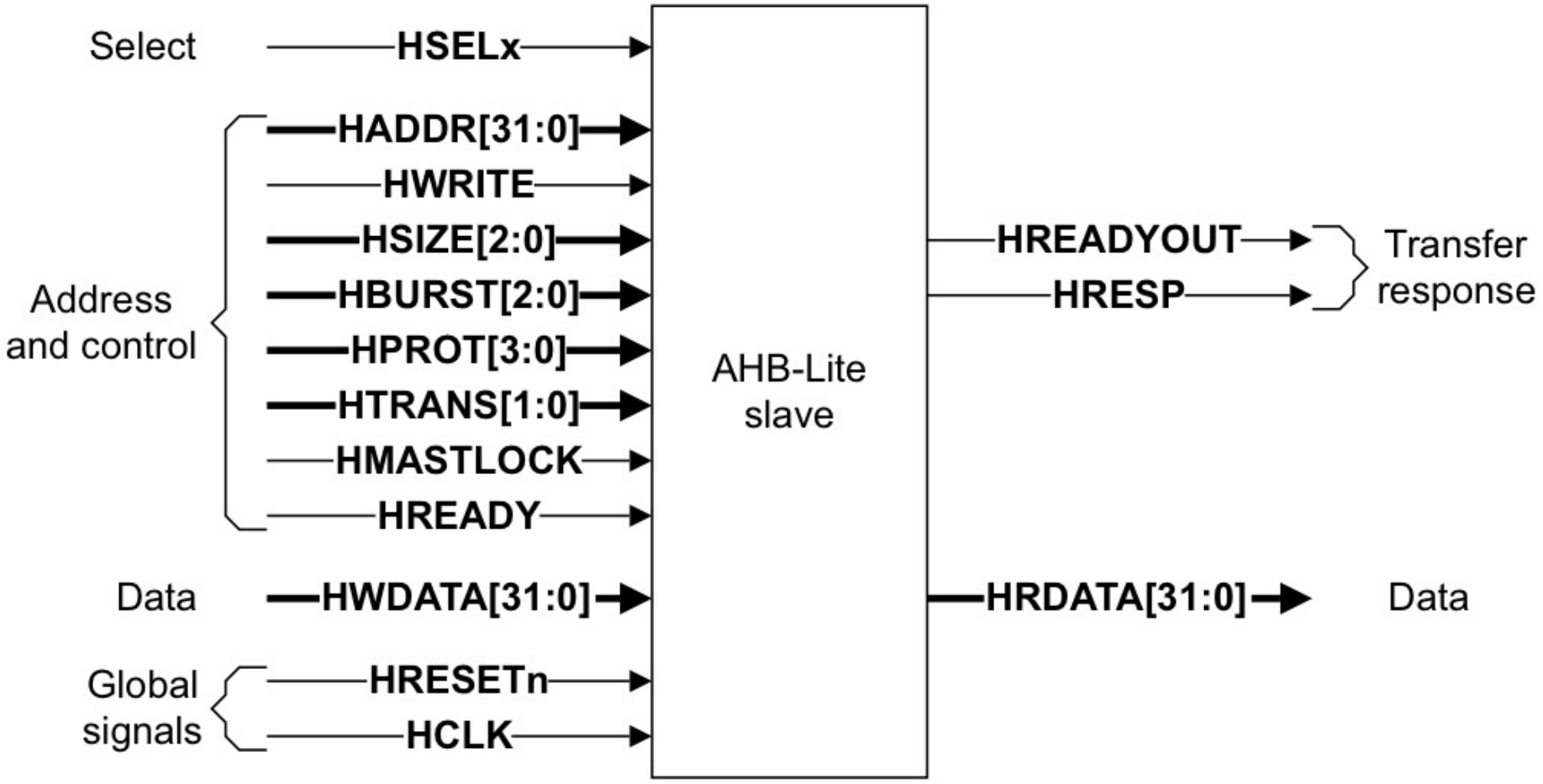
AHB-Lite Block Diagram



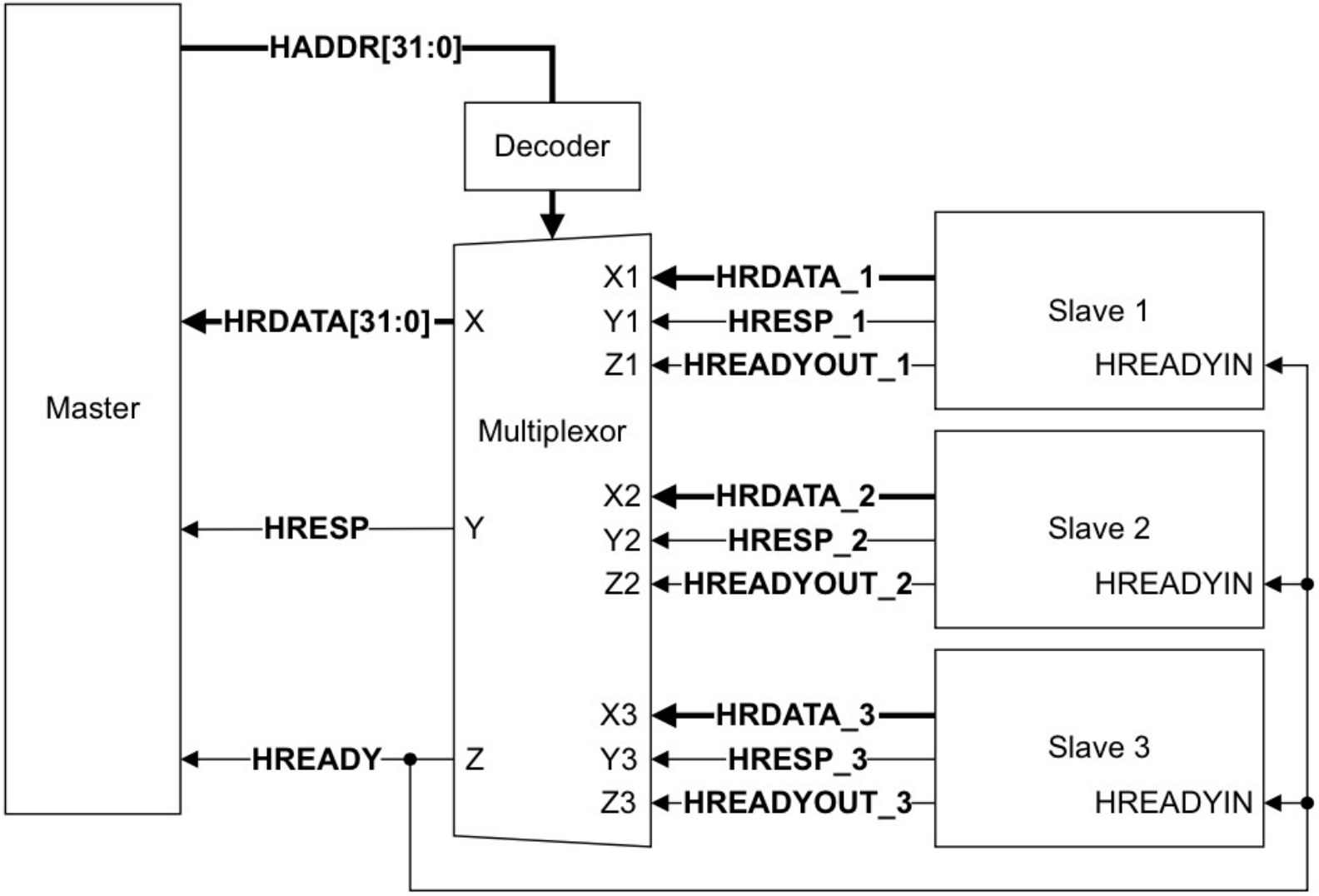
Master Interface



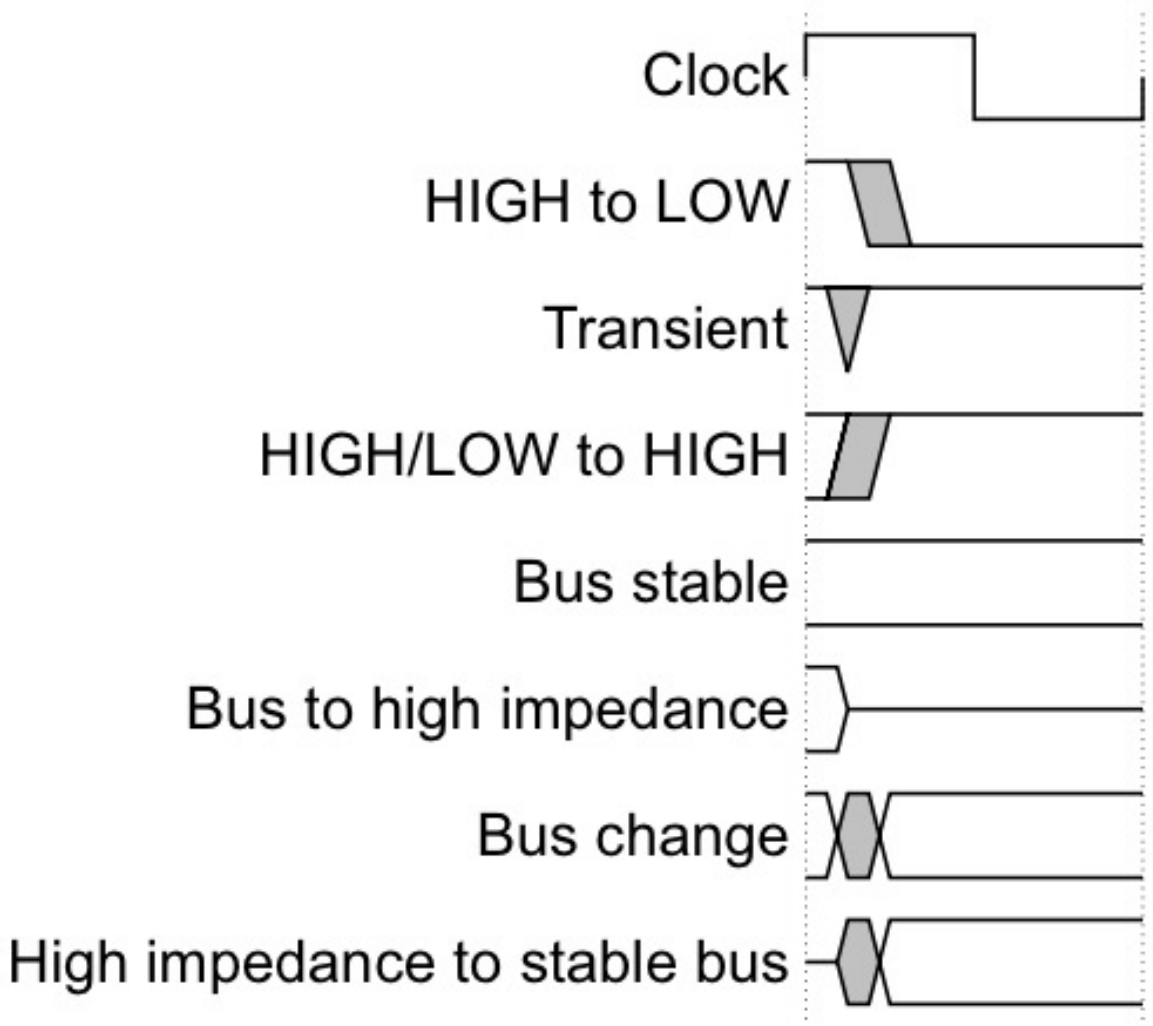
Slave Interface



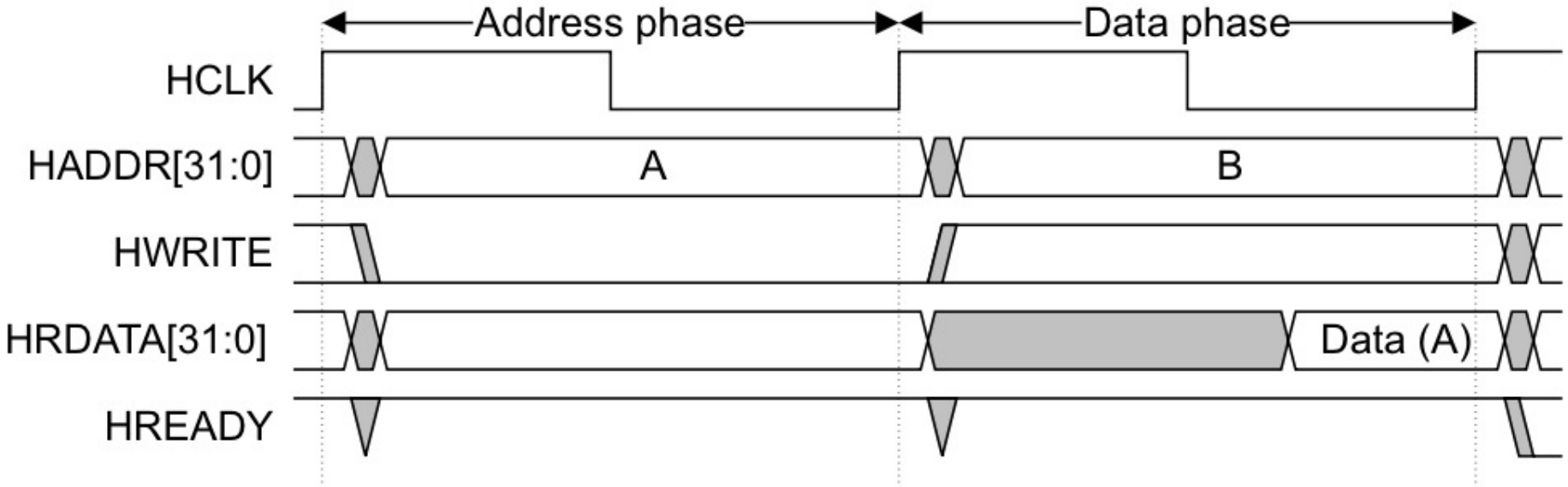
Multiplexor Interconnection



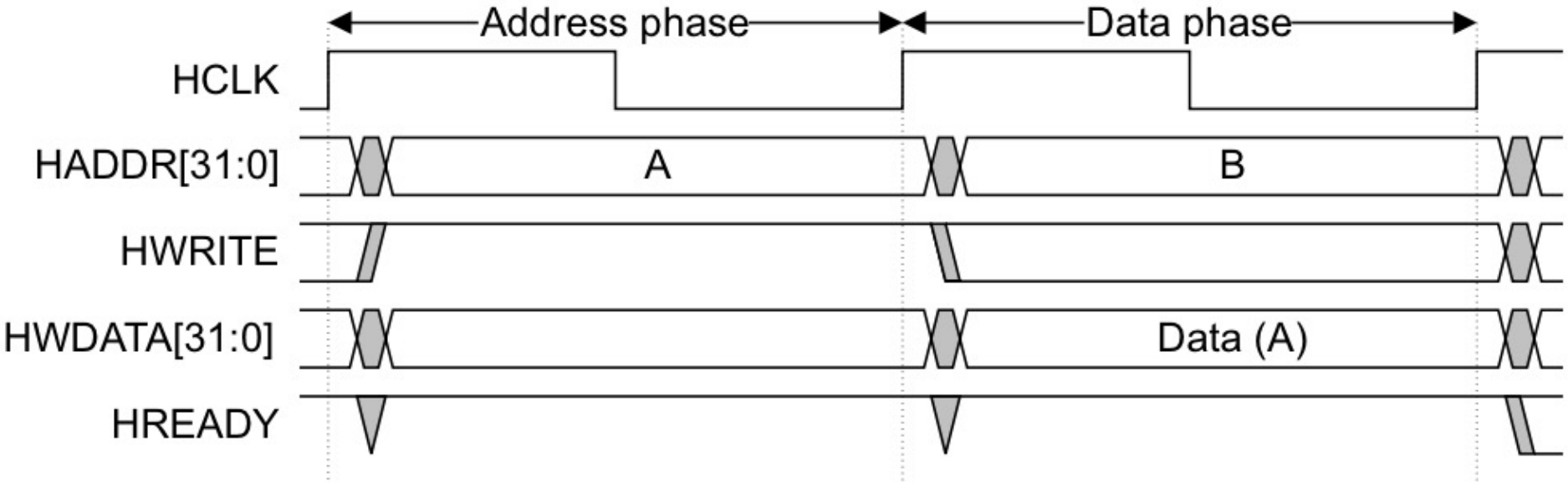
Timing Diagram Conventions



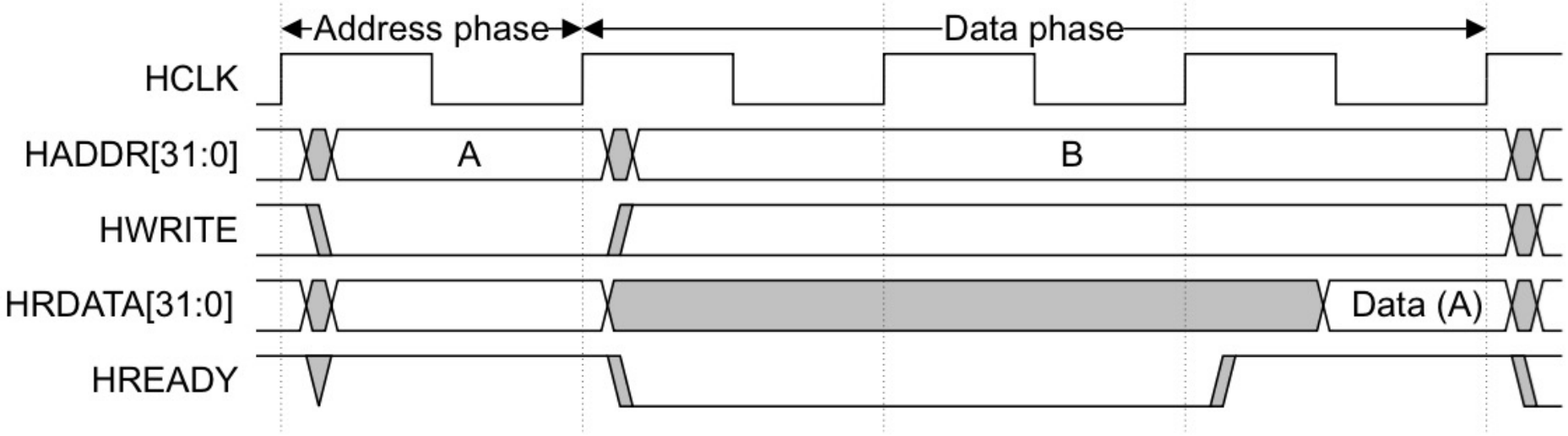
Read Transfer



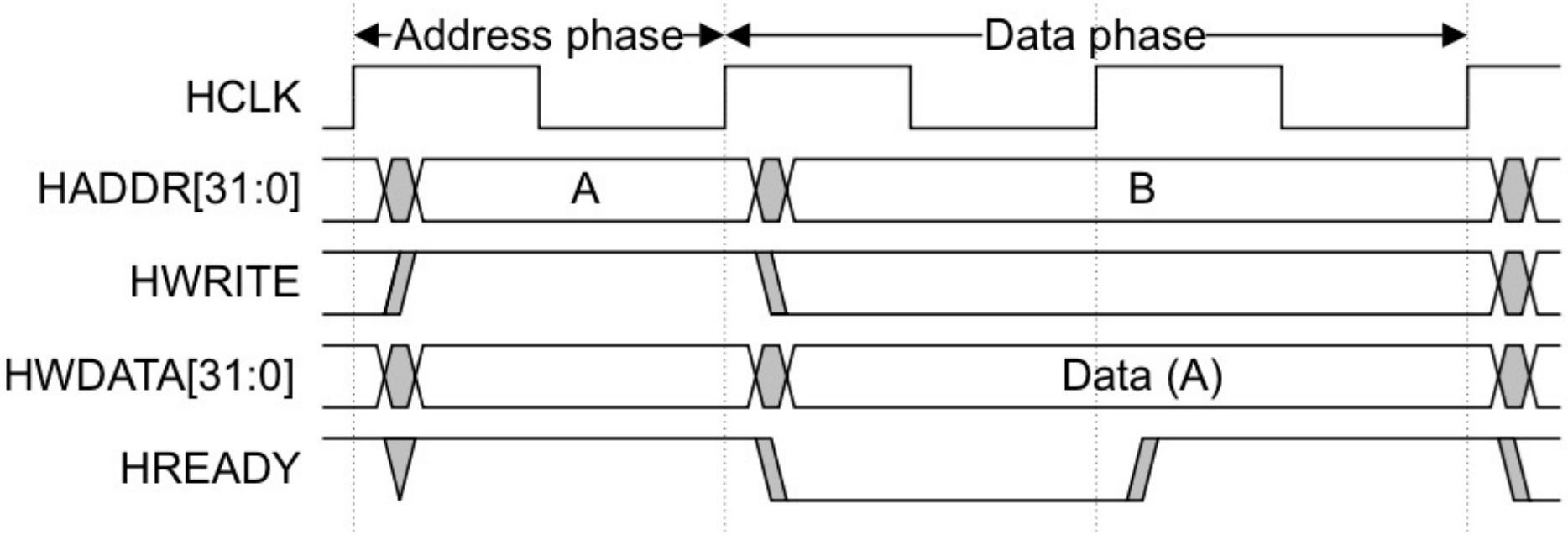
Write Transfer



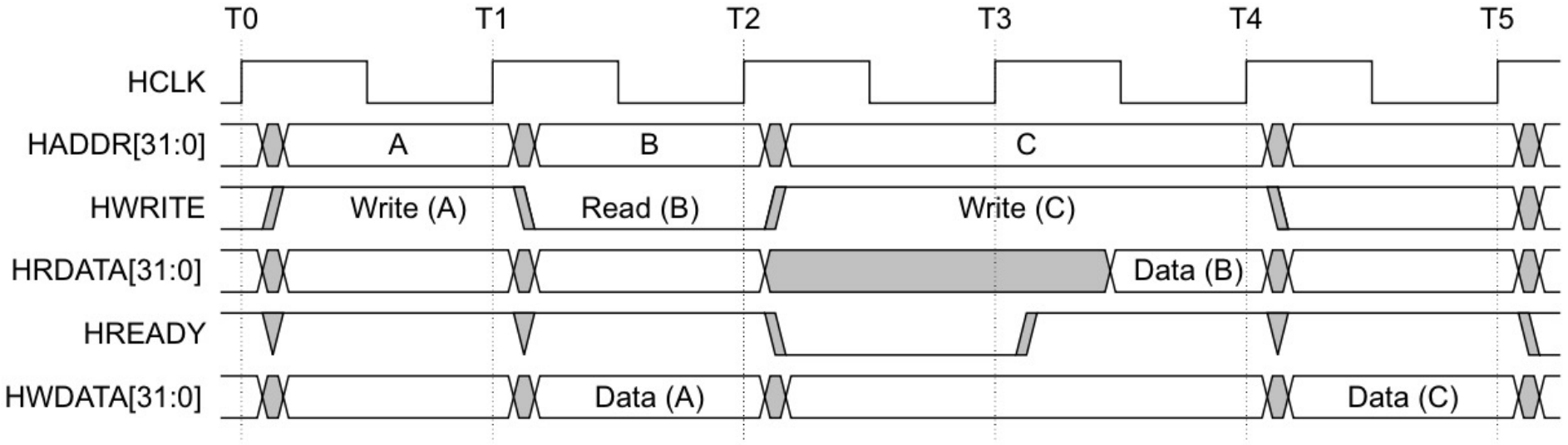
Read Transfer with Wait States



Write Transfer with Wait States



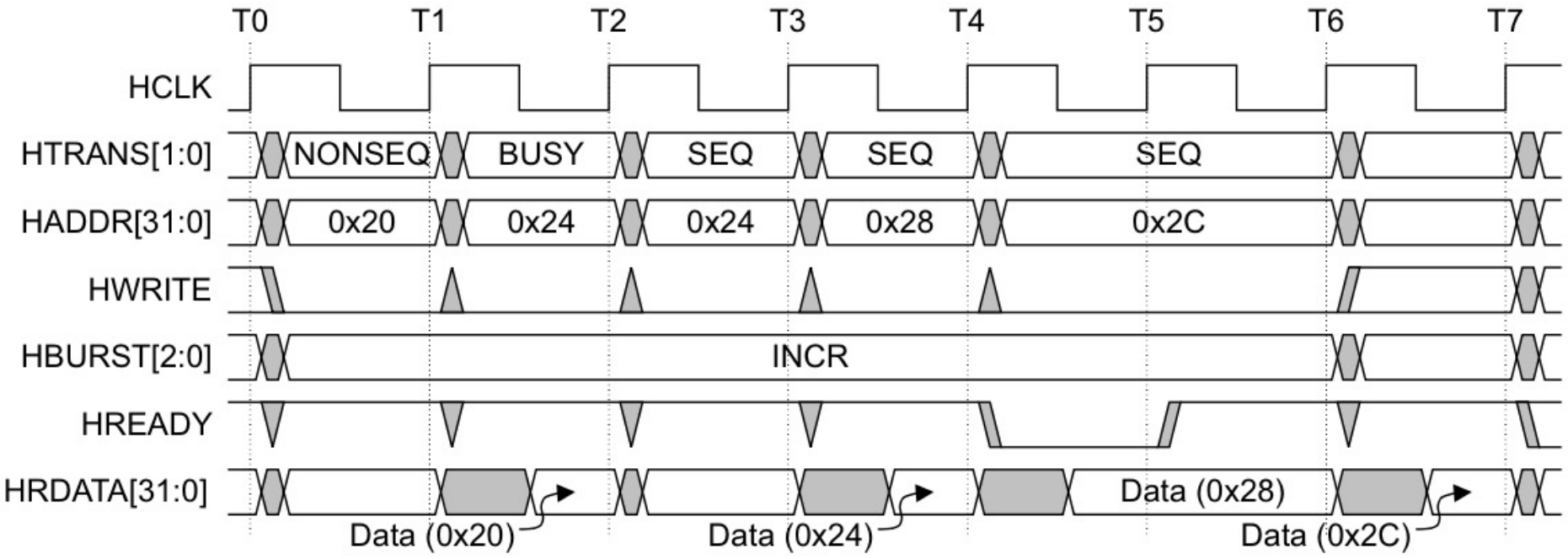
Multiple Transfers



Transfer Types

HTRANS[1:0]	Type	
00	IDLE	Master does not want transfer
01	BUSY	Master inserts wait cycle
10	NONSEQ	Single transfer or first transfer of a burst
11	SEQ	Remaining transfers of a burst

Transfer Type Examples



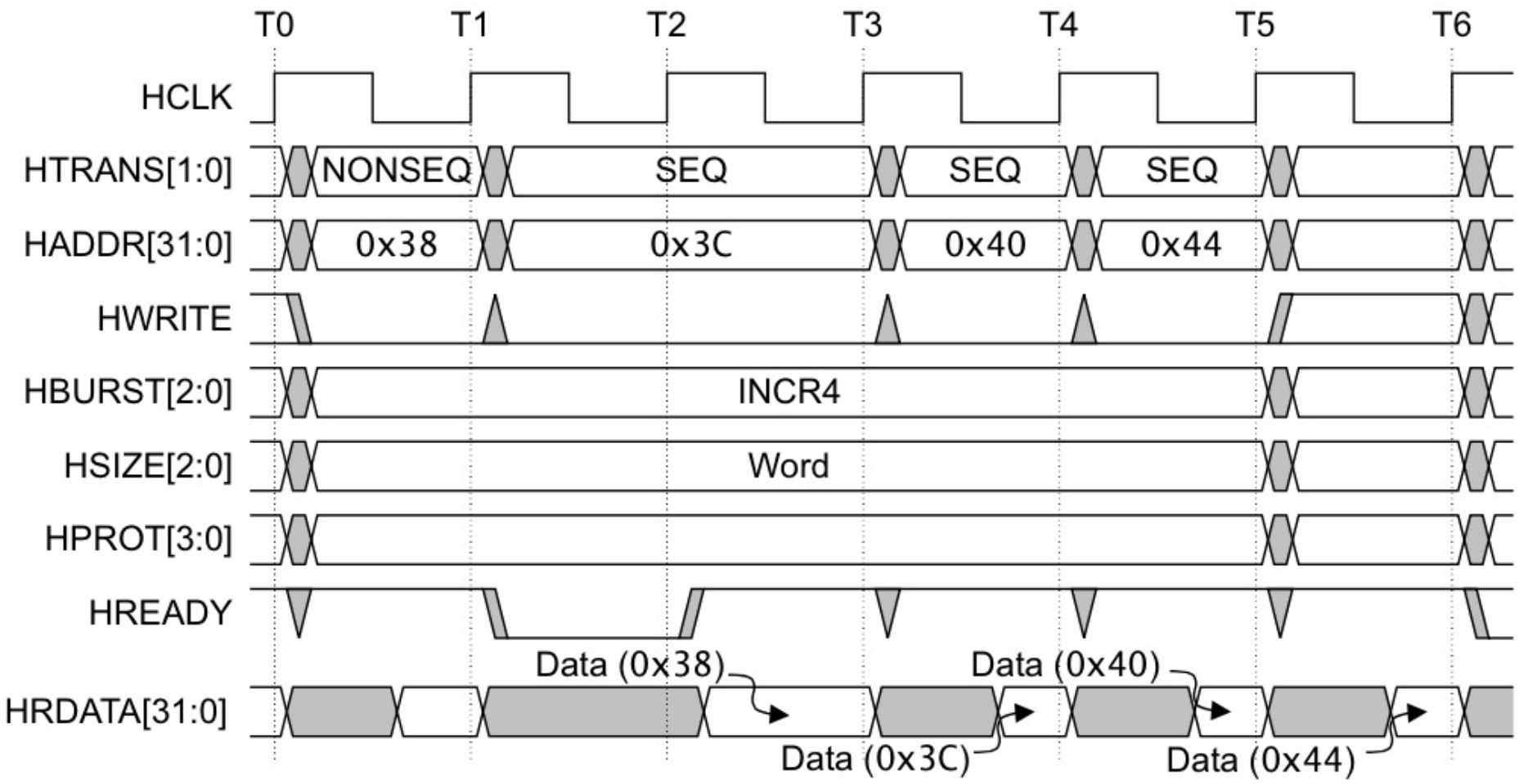
Transfer Size

HSIZE[2:0]	Size (bits)	Size (bytes)
000	8	1
001	16	2
010	32	4
011	64	8
100	128	16
101	256	32
110	512	64
111	1024	128

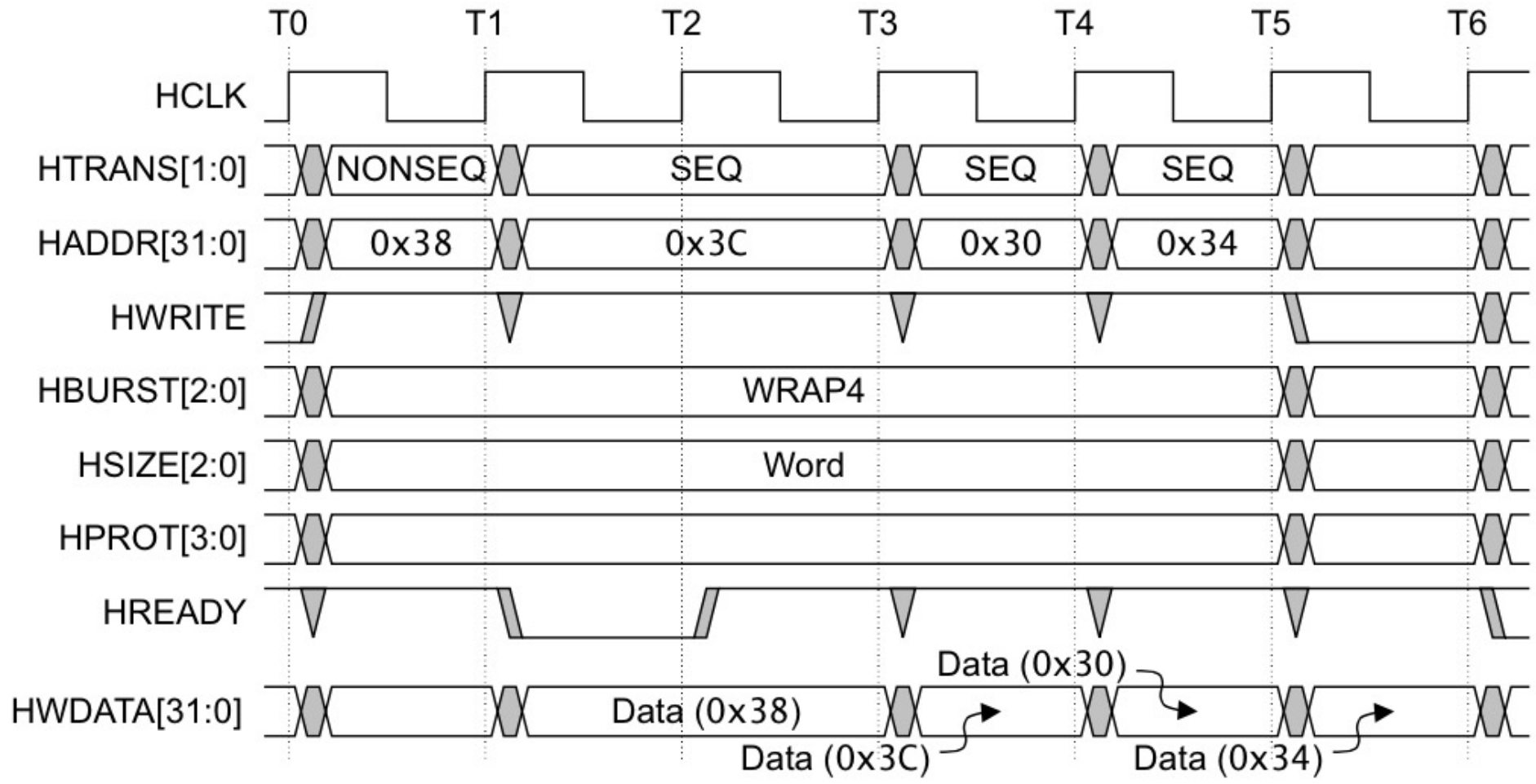
Burst type and size

HBURST[2:0]	Type and size
000	SINGLE
001	INCR
010	WRAP4
011	INCR4
100	WRAP8
101	INCR8
110	WRAP16
111	INCR16

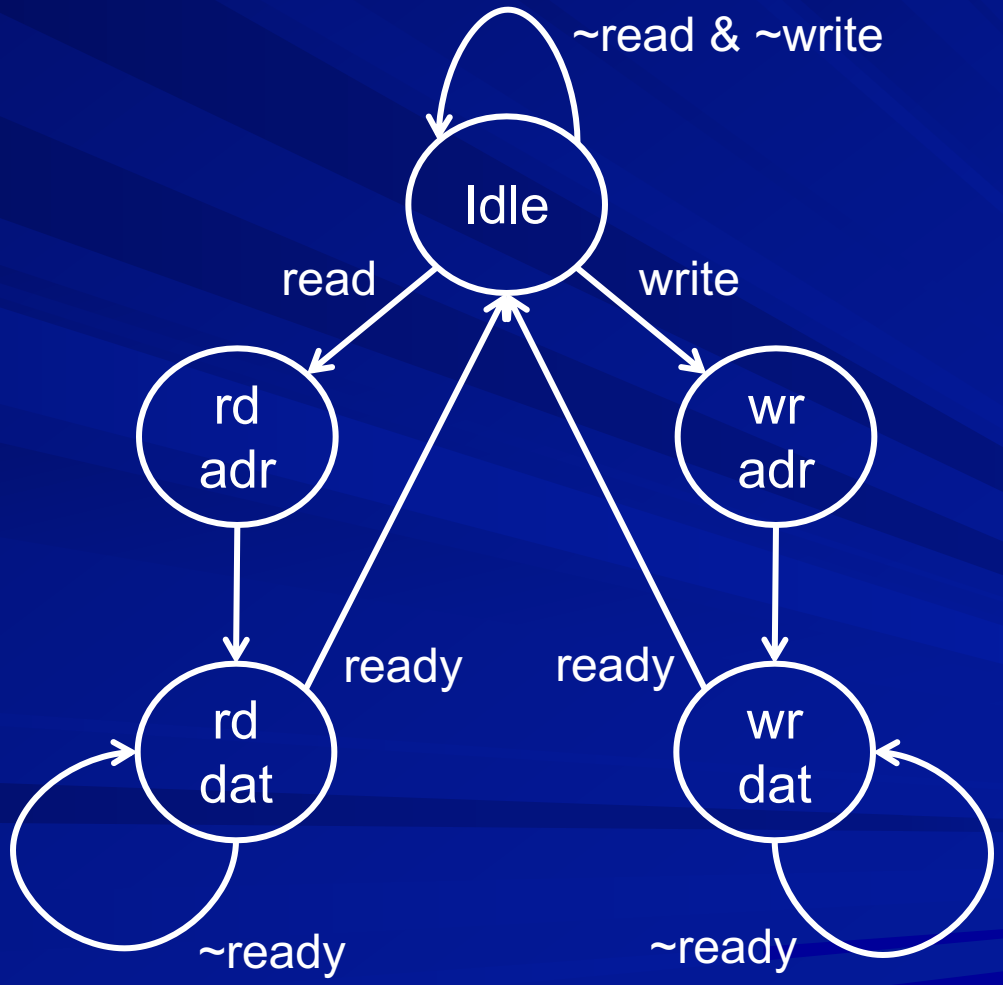
Four-Beat Incrementing Burst



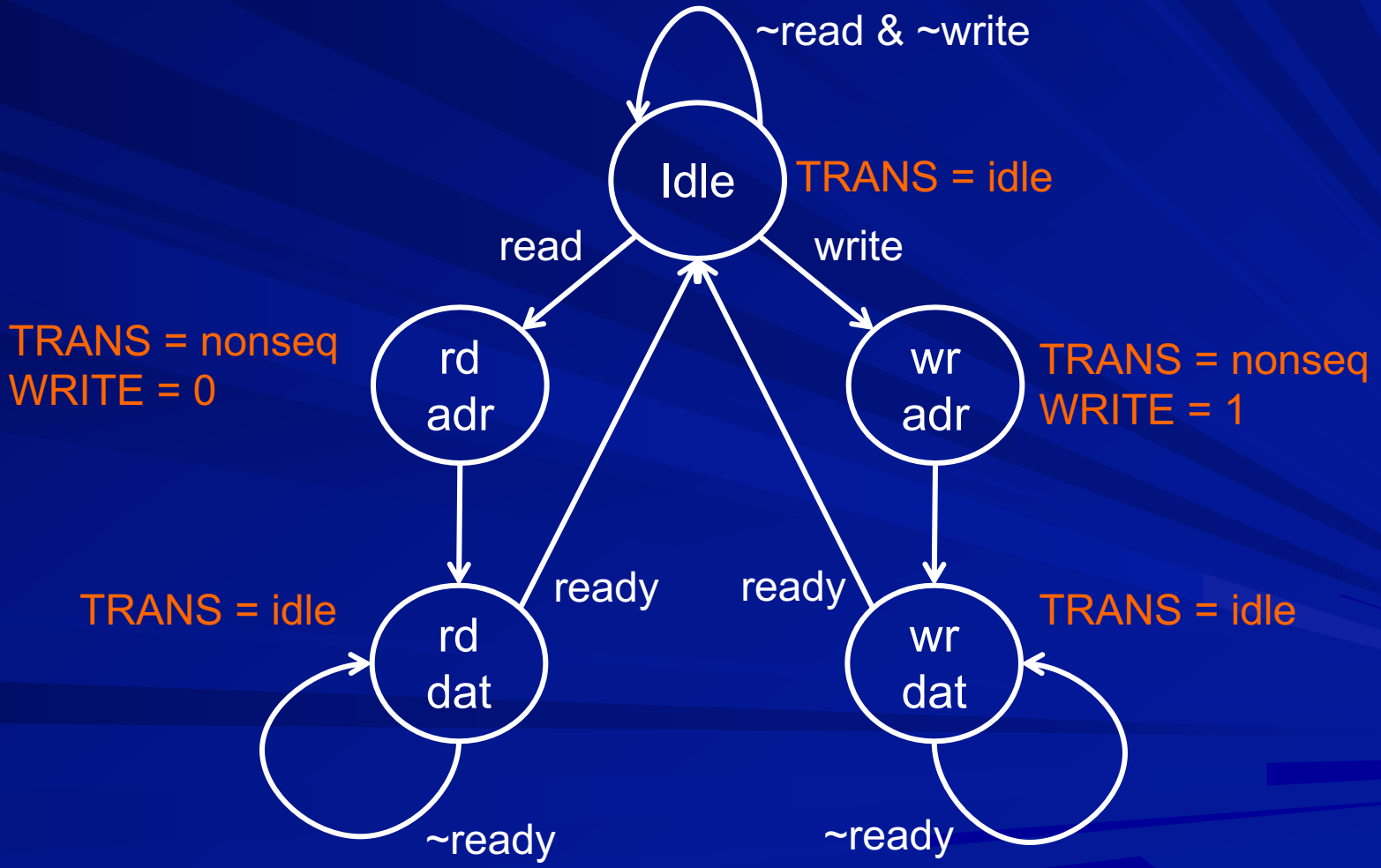
Four-Beat Wrapping Burst



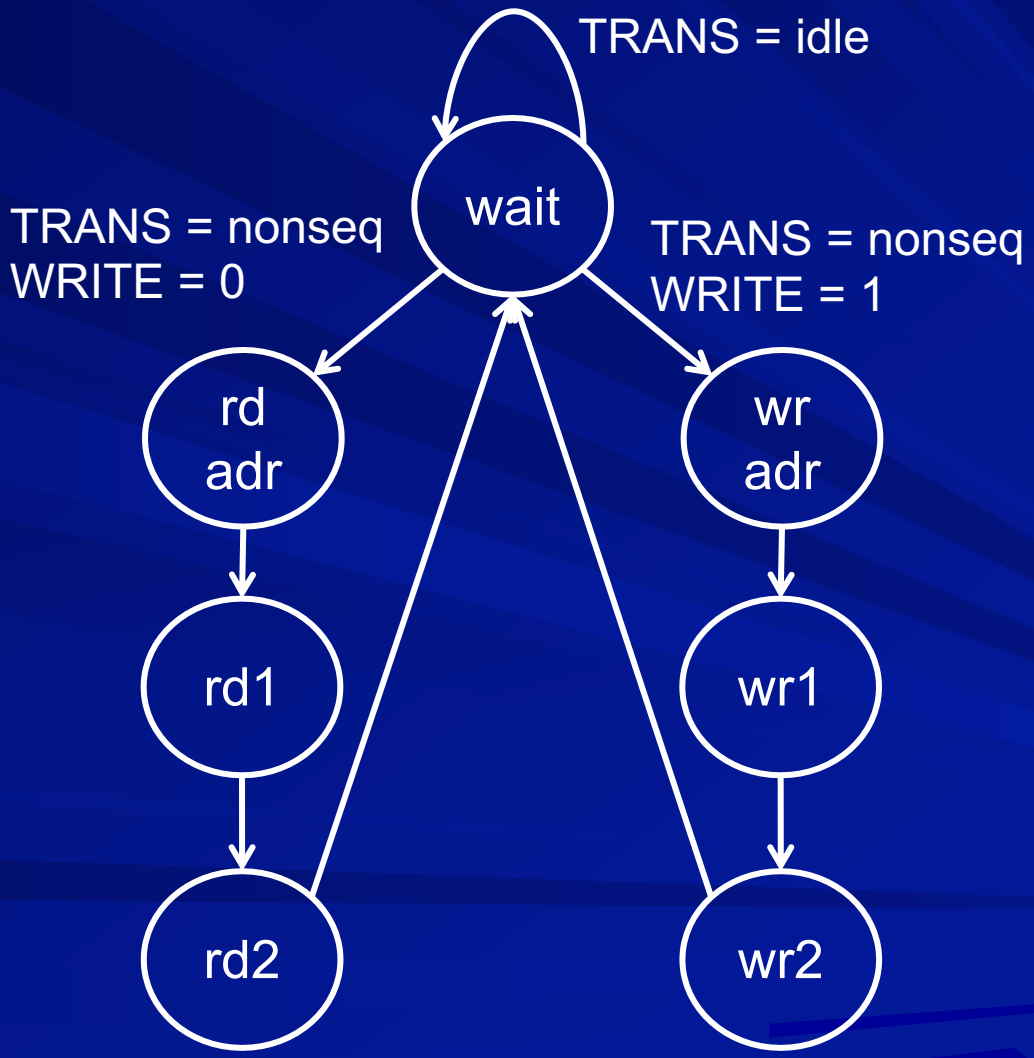
Master interface



Master interface



Slave interface



Obtaining access to a bus

Masters and slaves on a bus

- masters (processors/peripherals) initiate transactions
- slaves (peripherals/memories) respond to the masters

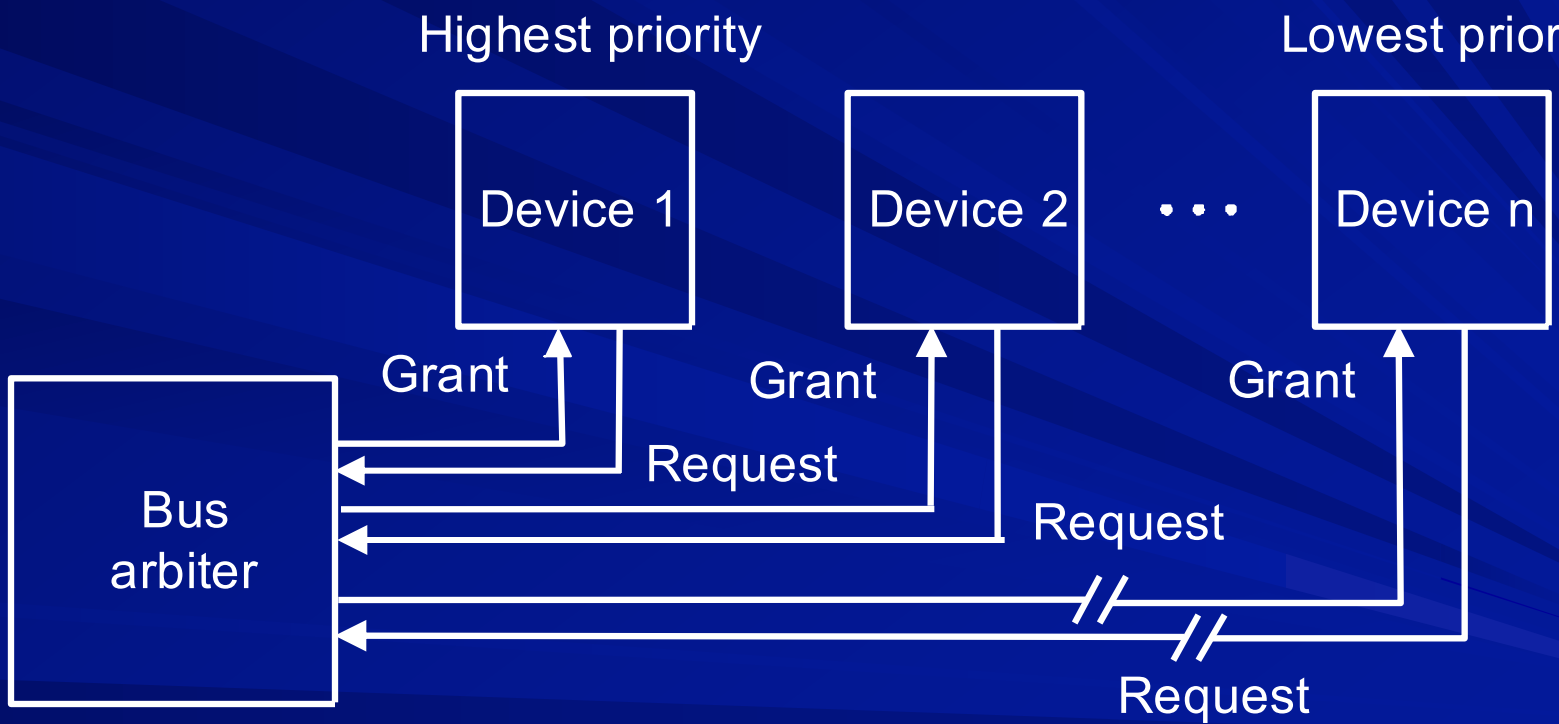
How does a master get control of a bus?

Priorities may be assigned. Fairness is essential.

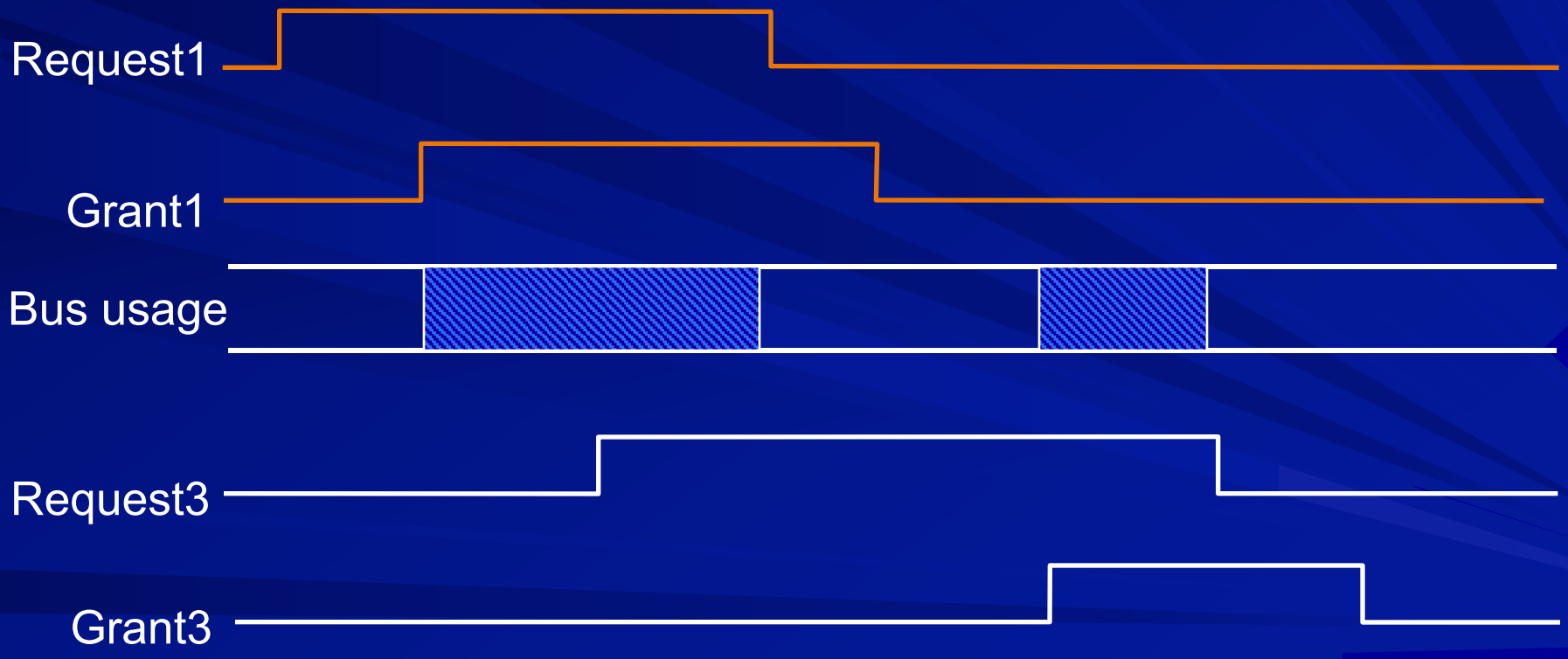
- Daisy chaining
- Centralized parallel arbitration
- Distributed arbitration by self selection
- Distributed arbitration by collision detection

Centralized parallel arbitration

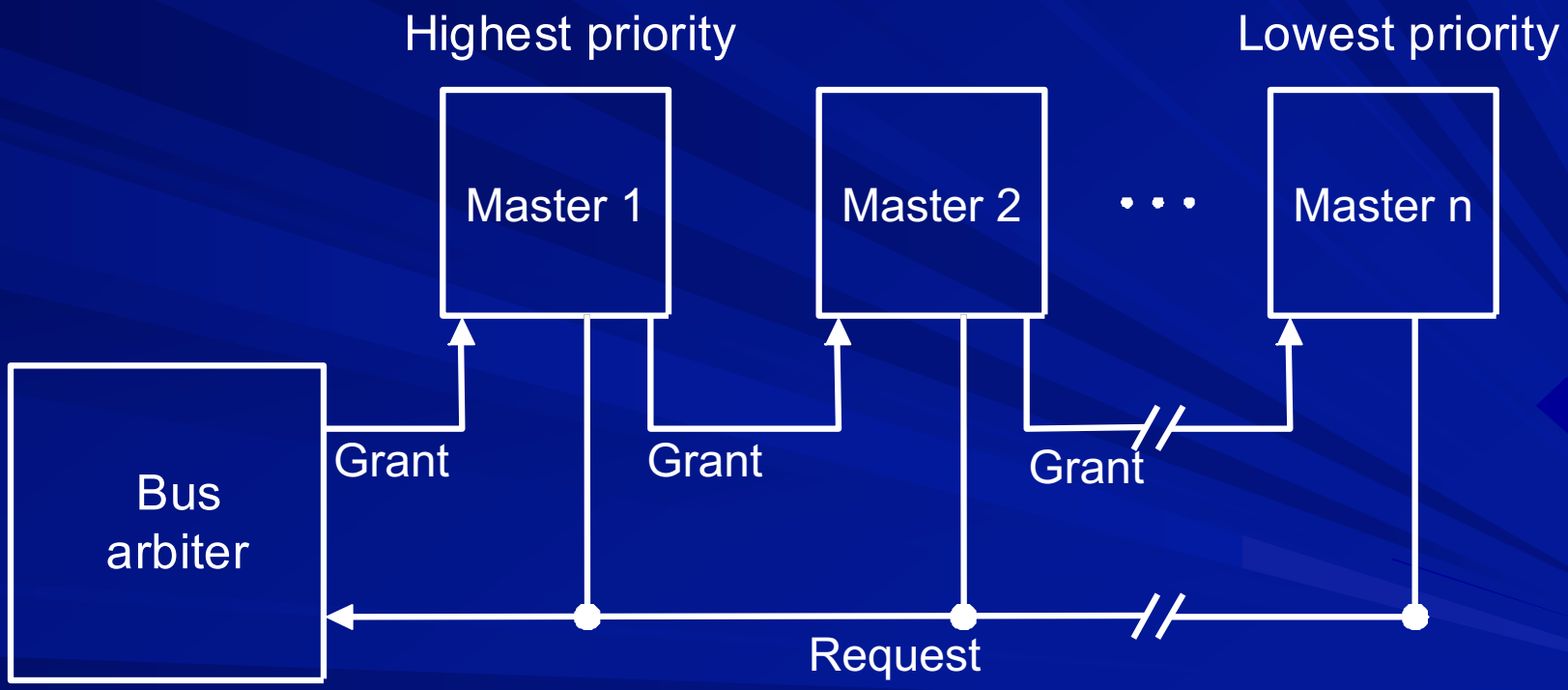
Example: PCI bus (a backplane bus)



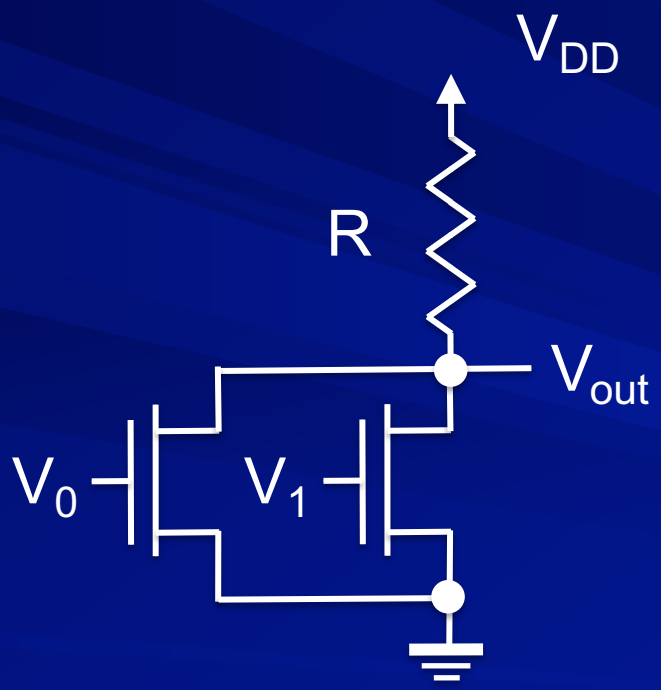
Request and Grant signals



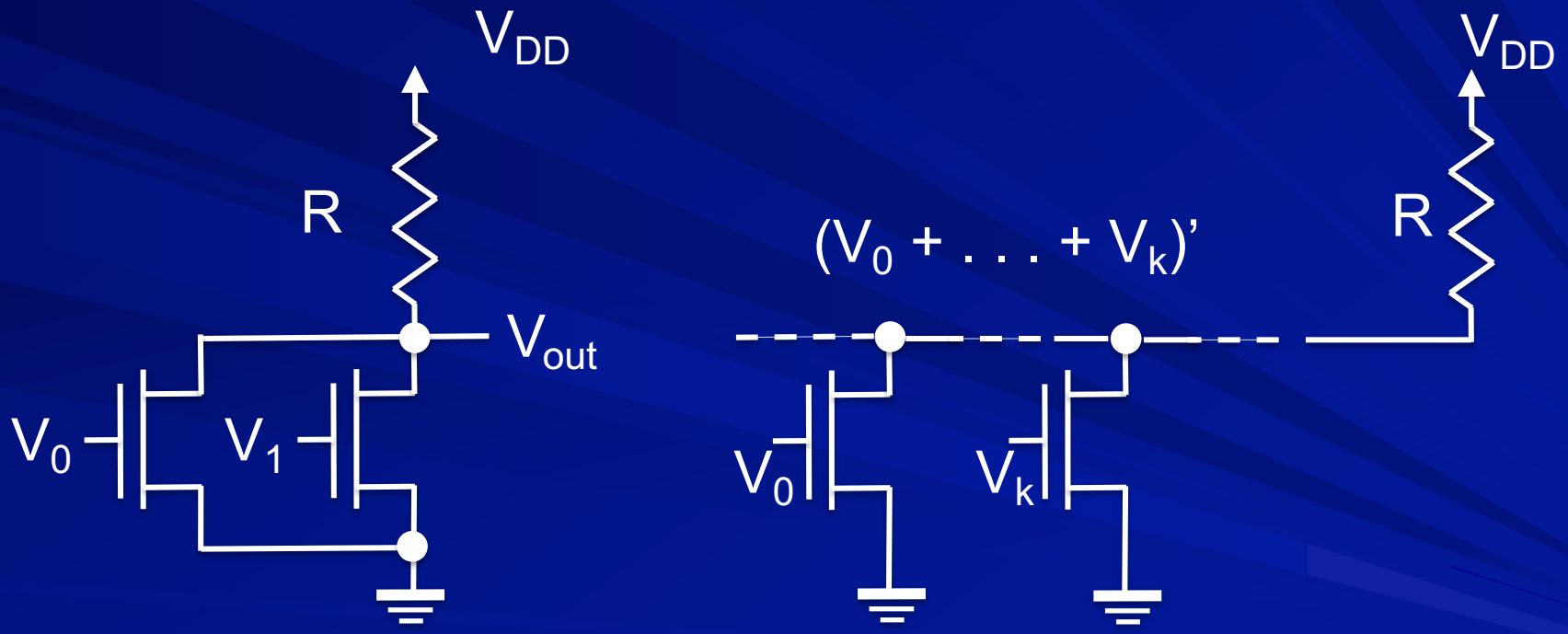
Combining/chaining signals



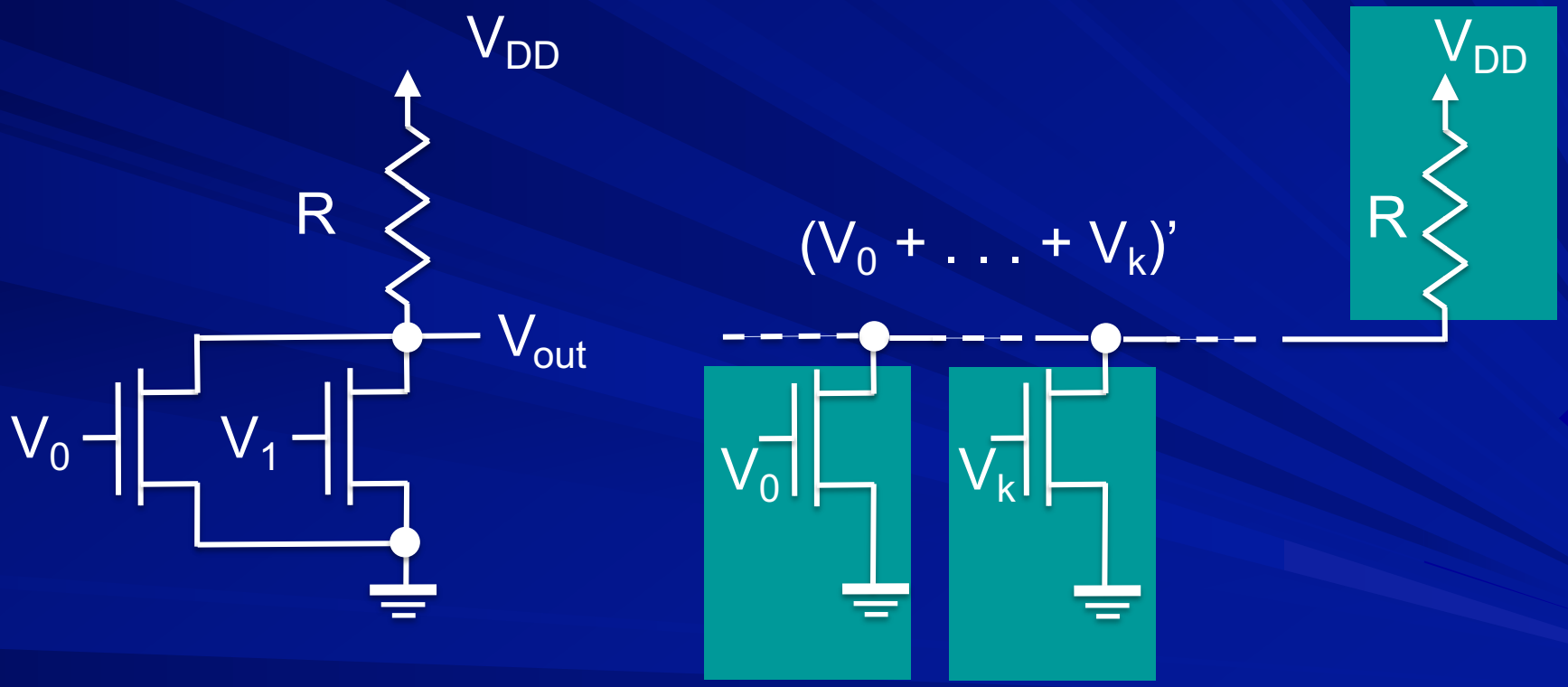
Wired OR



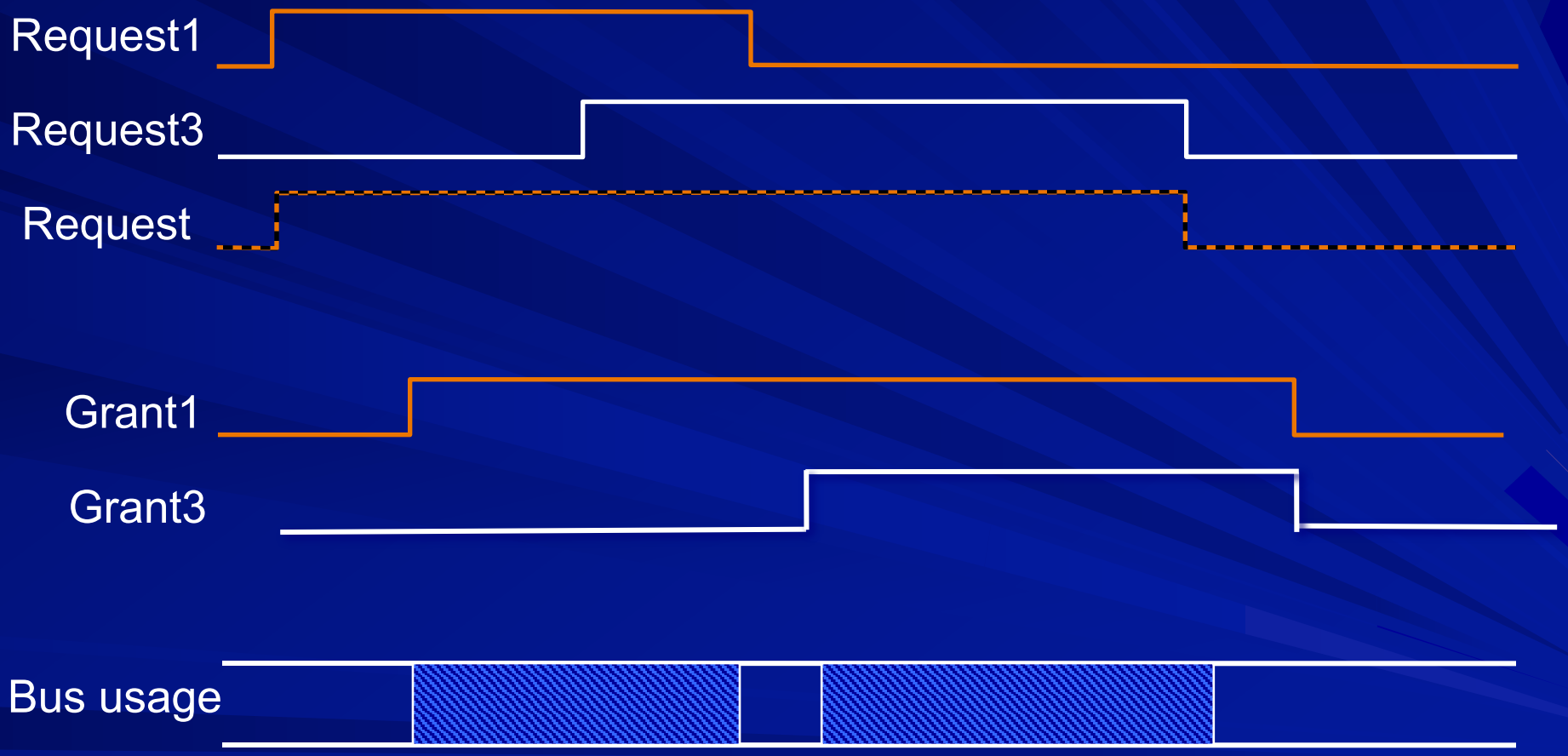
Wired OR



Wired OR

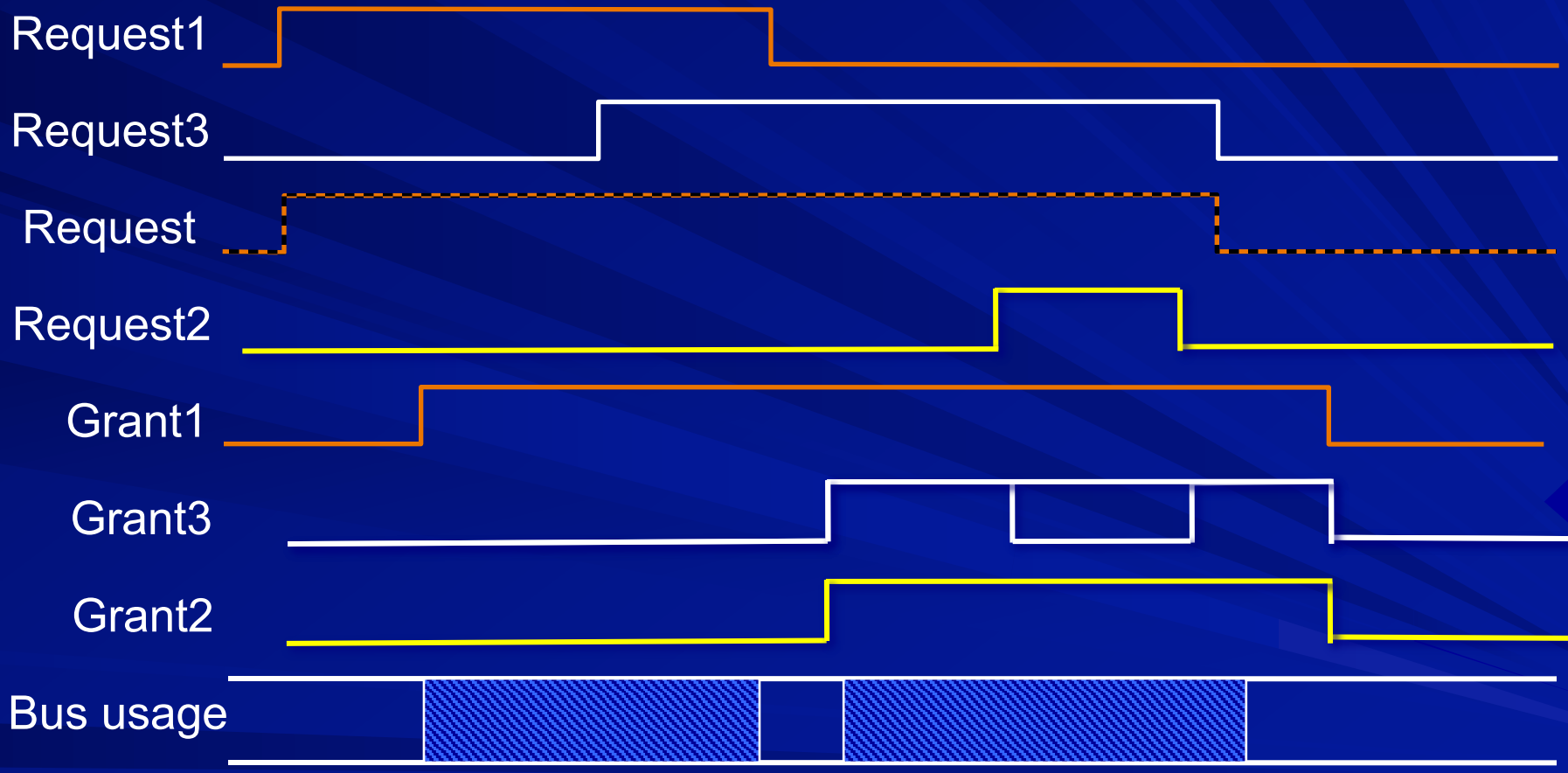


Request and Grant signals



Problem: High priority device can usurp the grant signal

Request and Grant signals



Problem: High priority device can usurp the grant signal
Solution: Daisy chain – request, grant, release

Daisy chain

Arbiter Device

Request ↑

Grant ↑

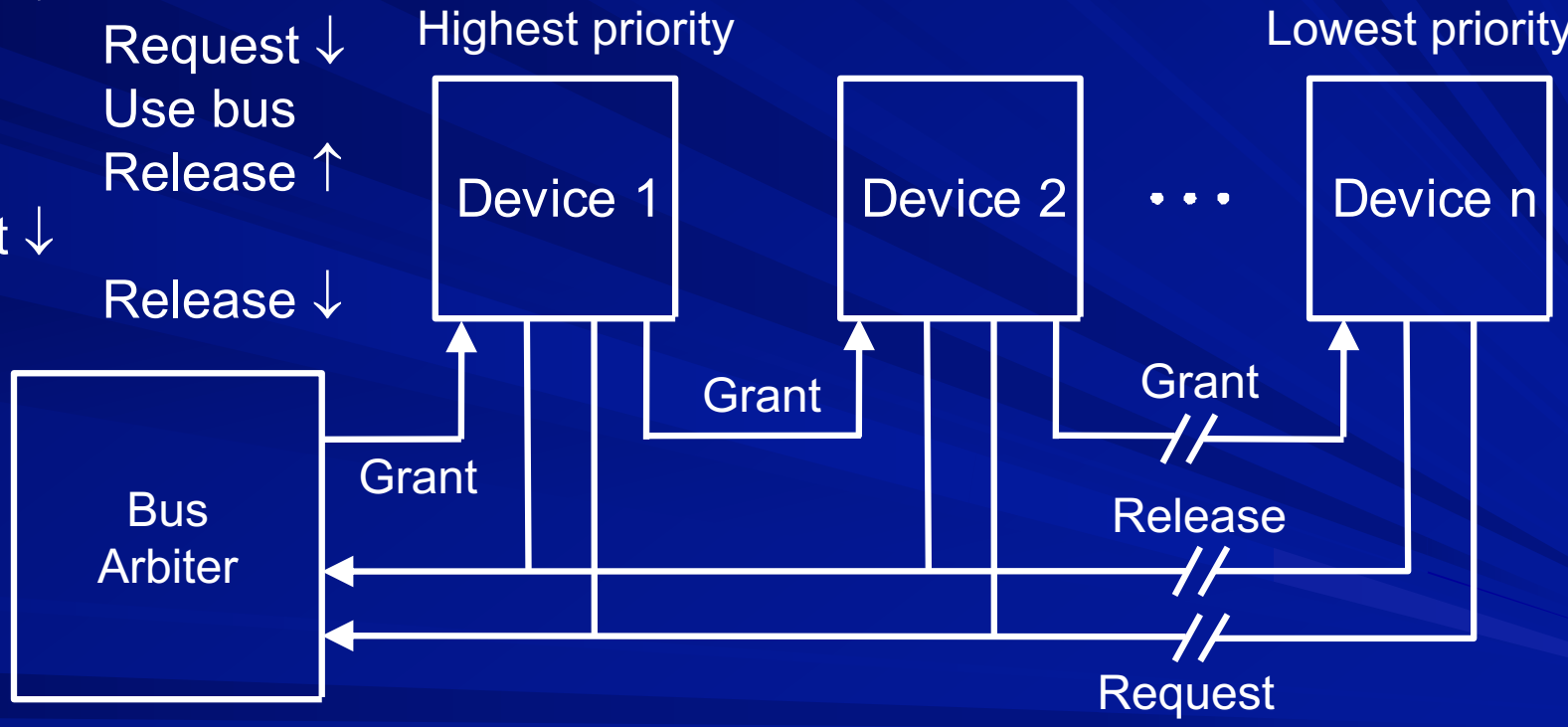
Request ↓

Use bus

Release ↑

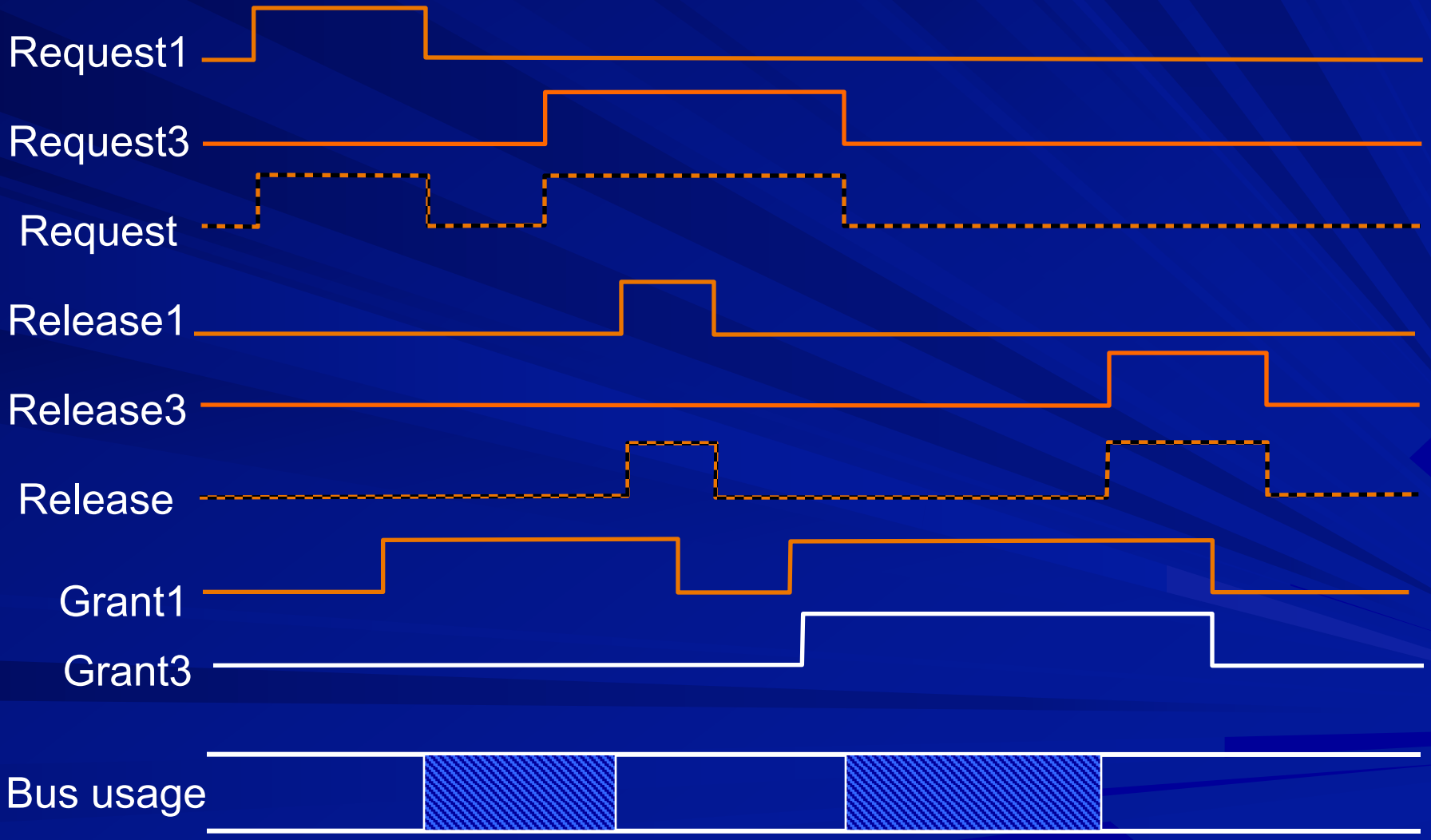
Grant ↓

Release ↓



Note: Rising edge on Grant signal is significant, not level

Request, release and grant signals



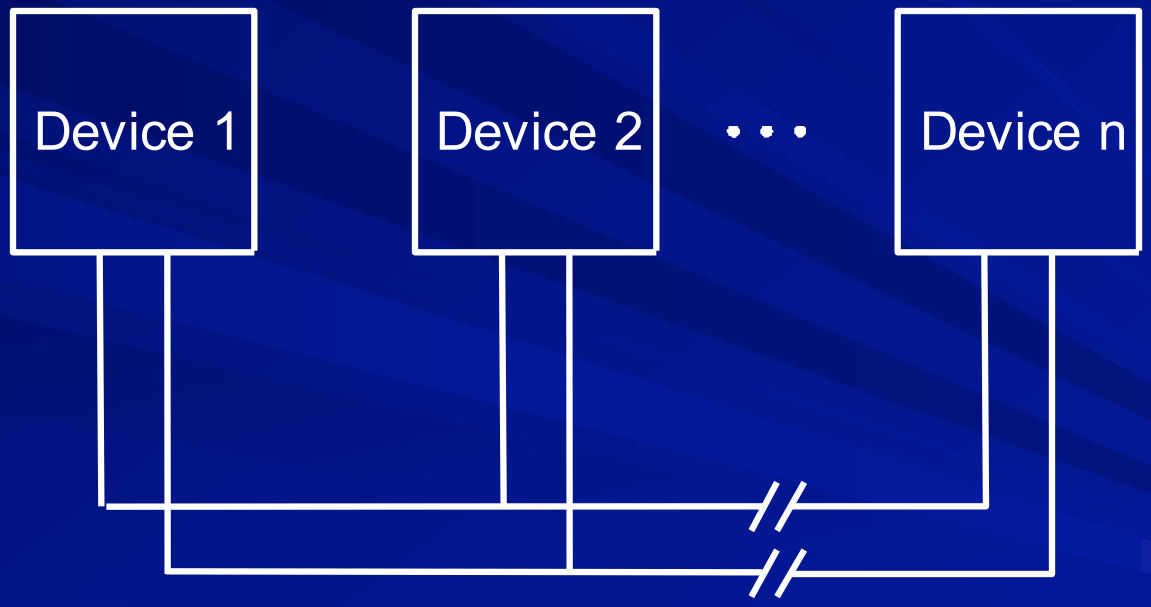
Daisy chain characteristics

- Simple and inexpensive
- Speed is limited due to chaining of grant signal
- Starvation of low priority devices may not be prevented

Rule to improve fairness: A device that has just used the bus cannot reacquire it until it sees the request line go low

Distributed arbitration

Example: NuBus (a backplane bus) in Apple Macintosh



Request lines and identity codes

Distributed arbitration in NuBus

- $req = req^0 + req^1 + .. + req^{n-1}$ (wired OR)
- $ID^i_3 .. ID^i_0$ = ID of i^{th} requester
- $arb^i_3 .. arb^i_0$ = arb output of i^{th} requester
- $arb_{3..0} = arb^0_{3..0} + .. + arb^{n-1}_{3..0}$ (wired OR)
- $arb^i_3 = ID^i_3 . req^i$
- $arb^i_2 = ID^i_2 . req^i . (ID^i_3 + arb_3')$
- $arb^i_1 = ID^i_1 . req^i . (ID^i_3 + arb_3') . (ID^i_2 + arb_2')$
- $arb^i_0 = ID^i_0 . req^i . (ID^i_3 + arb_3') . (ID^i_2 + arb_2') . (ID^i_1 + arb_1')$

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1				
2	0	1	1	0				
3	1	1	0	0				
4	0	0	1	1				

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	0	1
2	0	1	1	0	0	1	1	0
3	1	1	0	0	1	1	0	0
4	0	0	1	1	0	0	1	1
			OR		1	1	1	1

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1			
2	0	1	1	0	0			
3	1	1	0	0	1			
4	0	0	1	1	0			
					1			

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0		
2	0	1	1	0	0	x		
3	1	1	0	0	1	1		
4	0	0	1	1	0	x		
					1	1		

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	x	
2	0	1	1	0	0	x	x	
3	1	1	0	0	1	1	0	
4	0	0	1	1	0	x	x	
					1	1	0	

NuBus arbitration example-1

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	x	x
2	0	1	1	0	0	x	x	x
3	1	1	0	0	1	1	0	0
4	0	0	1	1	0	x	x	x
					1	1	0	0

NuBus arbitration example-2

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1				
2	0	1	1	0				
3	1	0	1	0				
4	0	0	1	1				

NuBus arbitration example-2

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	0	1
2	0	1	1	0	0	1	1	0
3	1	0	1	0	1	0	1	0
4	0	0	1	1	0	0	1	1
			OR		1	1	1	1

NuBus arbitration example-2

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1			
2	0	1	1	0	0			
3	1	0	1	0	1			
4	0	0	1	1	0			
					1			

NuBus arbitration example-2

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0		
2	0	1	1	0	0	x		
3	1	0	1	0	1	0		
4	0	0	1	1	0	x		
					1	0		

NuBus arbitration example-2

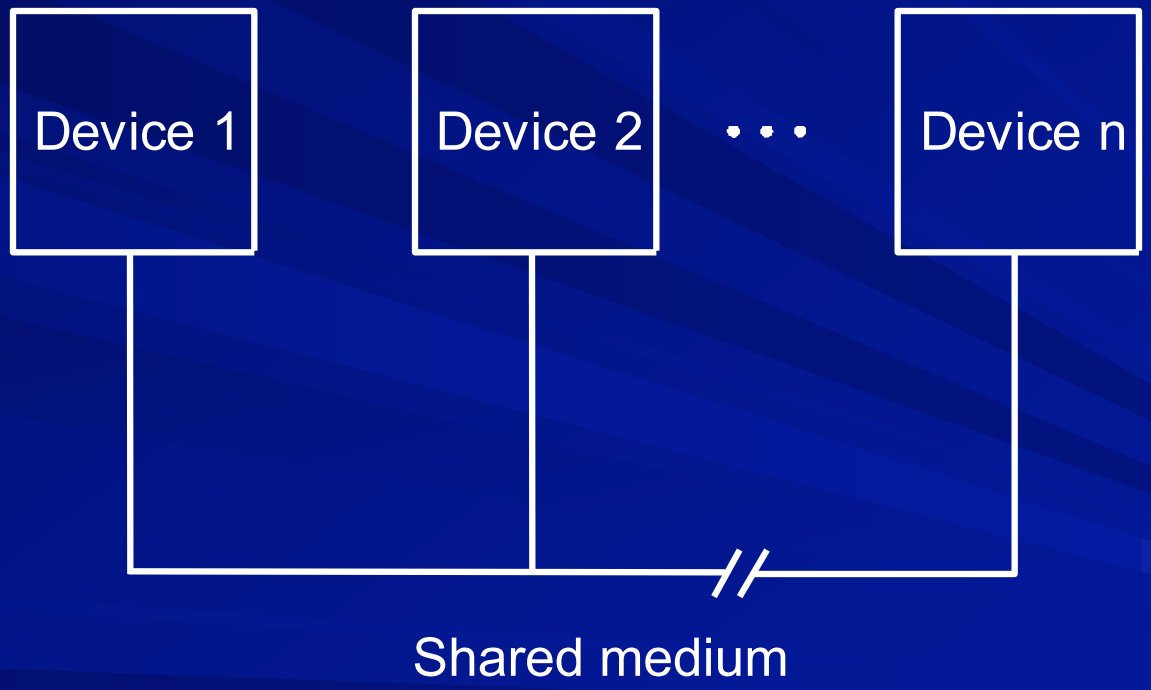
k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	0	
2	0	1	1	0	0	x	x	
3	1	0	1	0	1	0	1	
4	0	0	1	1	0	x	x	
					1	0	1	

NuBus arbitration example-2

k	ID^k_3	ID^k_2	ID^k_1	ID^k_0	arb^k_3	arb^k_2	arb^k_1	arb^k_0
1	1	0	0	1	1	0	0	x
2	0	1	1	0	0	x	x	x
3	1	0	1	0	1	0	1	0
4	0	0	1	1	0	x	x	x
					1	0	1	0

Arbitration by collision detection

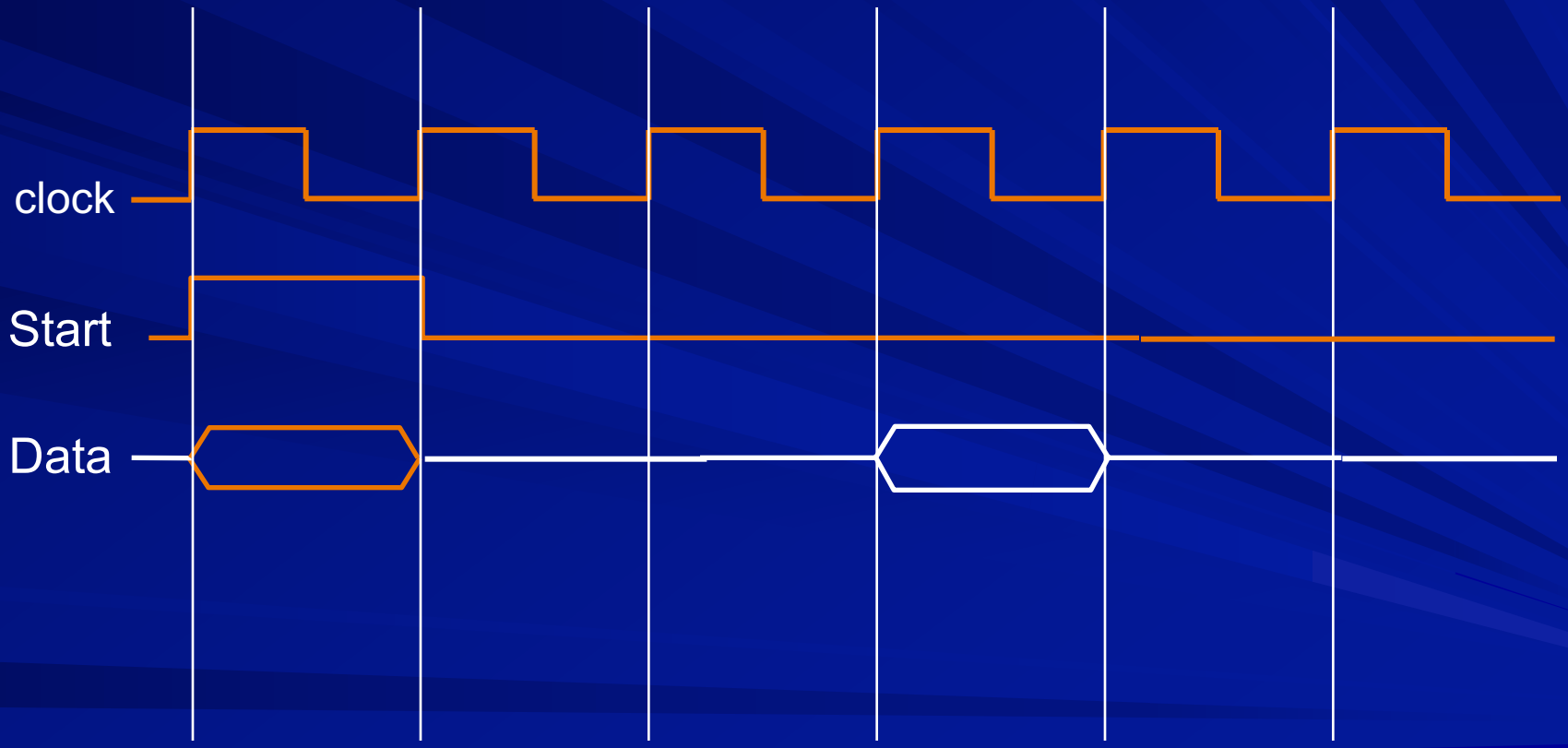
Example: Ethernet



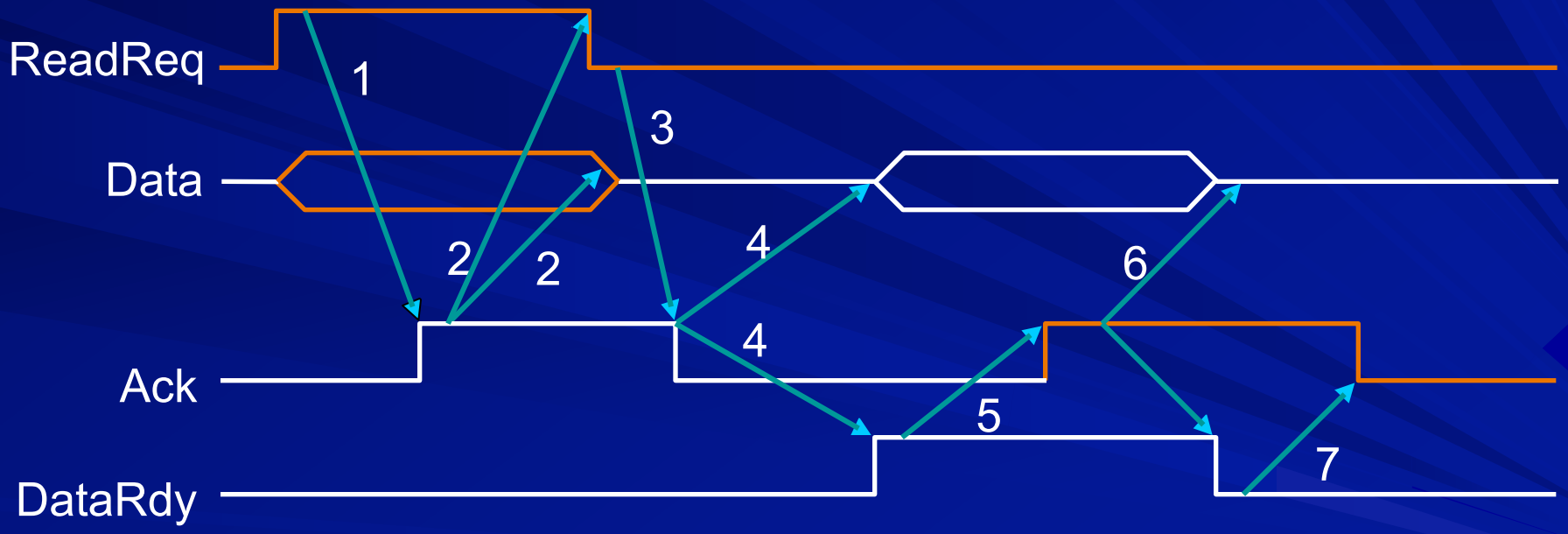
Increasing the bus bandwidth

- increase bus width
- separate data and address lines
- pipelining
- multiple word blocks
- synchronous
- split transaction

Synchronous transfer



Asynchronous handshaking



THANKS