

# Memory Systems for Embedded Applications

Chapter 4 (Sections 4.1-4.4)

# Platform components

---

- ▶ CPUs.
- ▶ Interconnect buses.
- ▶ Memory.
- ▶ Input/output devices.
  
- ▶ Implementations:
  - ▶ System-on-Chip (SoC) vs. Multi-Chip
    - ▶ Microcontroller vs. microprocessor
  - ▶ Commercial off-the-shelf (COTS) vs. custom
  - ▶ FPGA & Platform FPGA

# CPU Buses

---

- ▶ Mechanism for communication with memories and I/O devices
- ▶ Bus components:
  - ▶ signal wires with designated functions
  - ▶ protocol for data transfers
  - ▶ electrical parameters (voltage, current, capacitance, etc.)
  - ▶ physical design (connectors, cables, etc.)

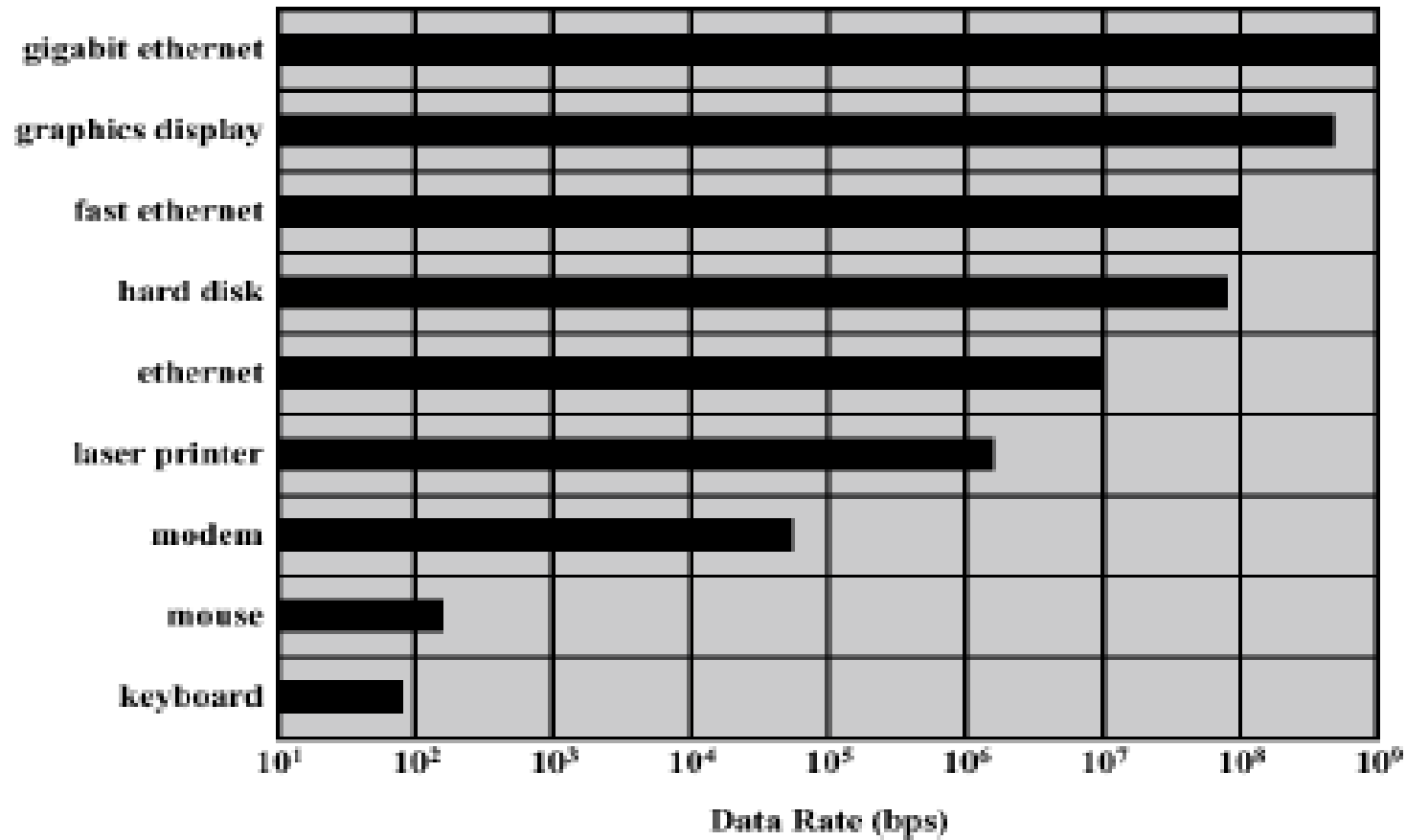
# Bus Types

---

- ▶ **Synchronous vs. Asynchronous**
  - ▶ Sync: all op's synchronized to a clock
  - ▶ Async: devices signal each other to indicate start/stop of operations
    - ▶ May combine sync/async (80x86 "Ready" signal)
- ▶ **Data transfer types:**
  - ▶ Processor to/from memory
  - ▶ Processor to/from I/O device
  - ▶ I/O device to/from memory (DMA)
- ▶ **Data bus types**
  - ▶ Parallel (data bits transferred in parallel)
  - ▶ Serial (data bits transferred serially)

# Typical bus data rates

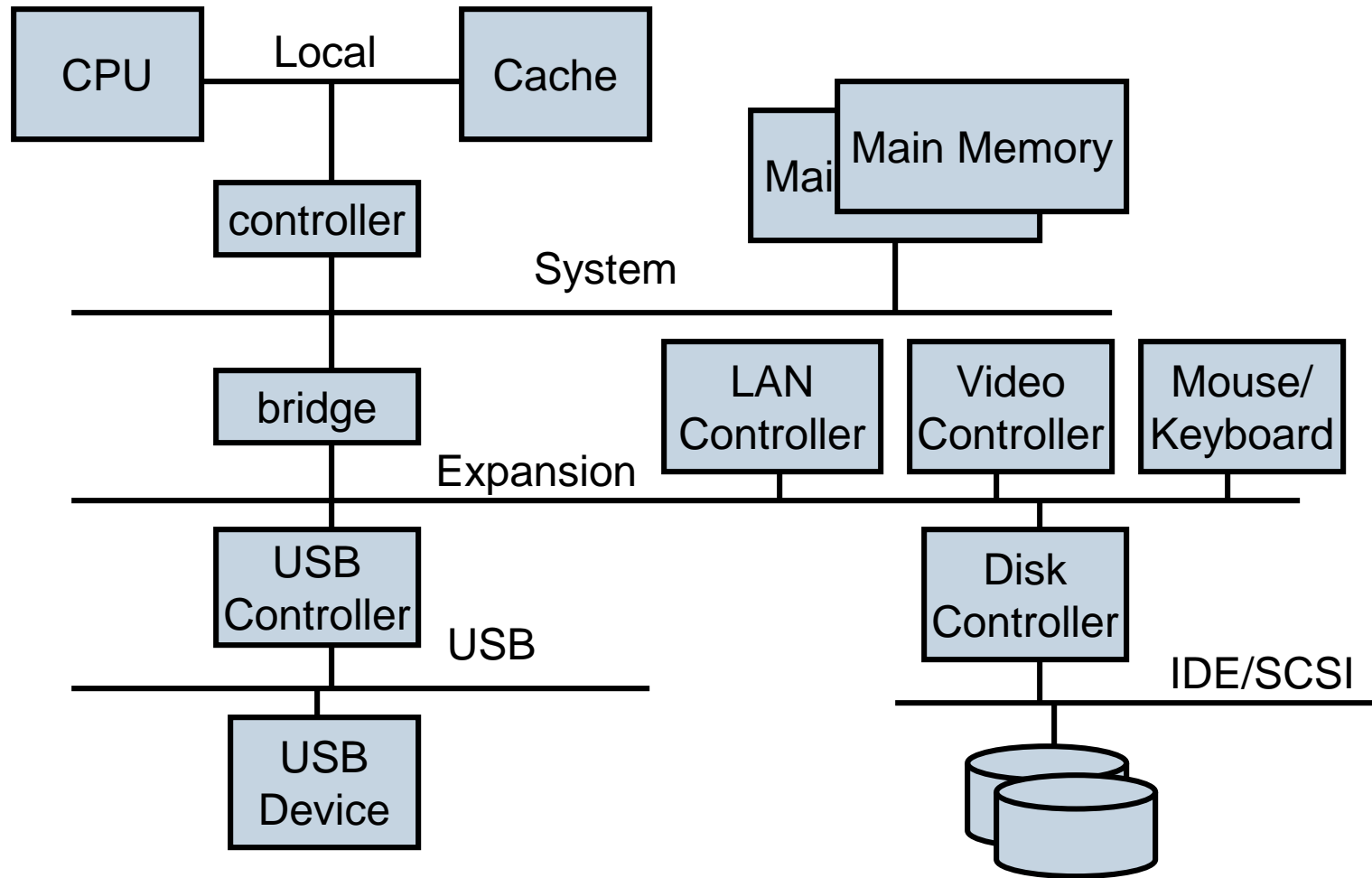
---



Source: Peter Cheung "Computer Architecture & Systems Course Notes"

---

# Hierarchical Bus Architecture

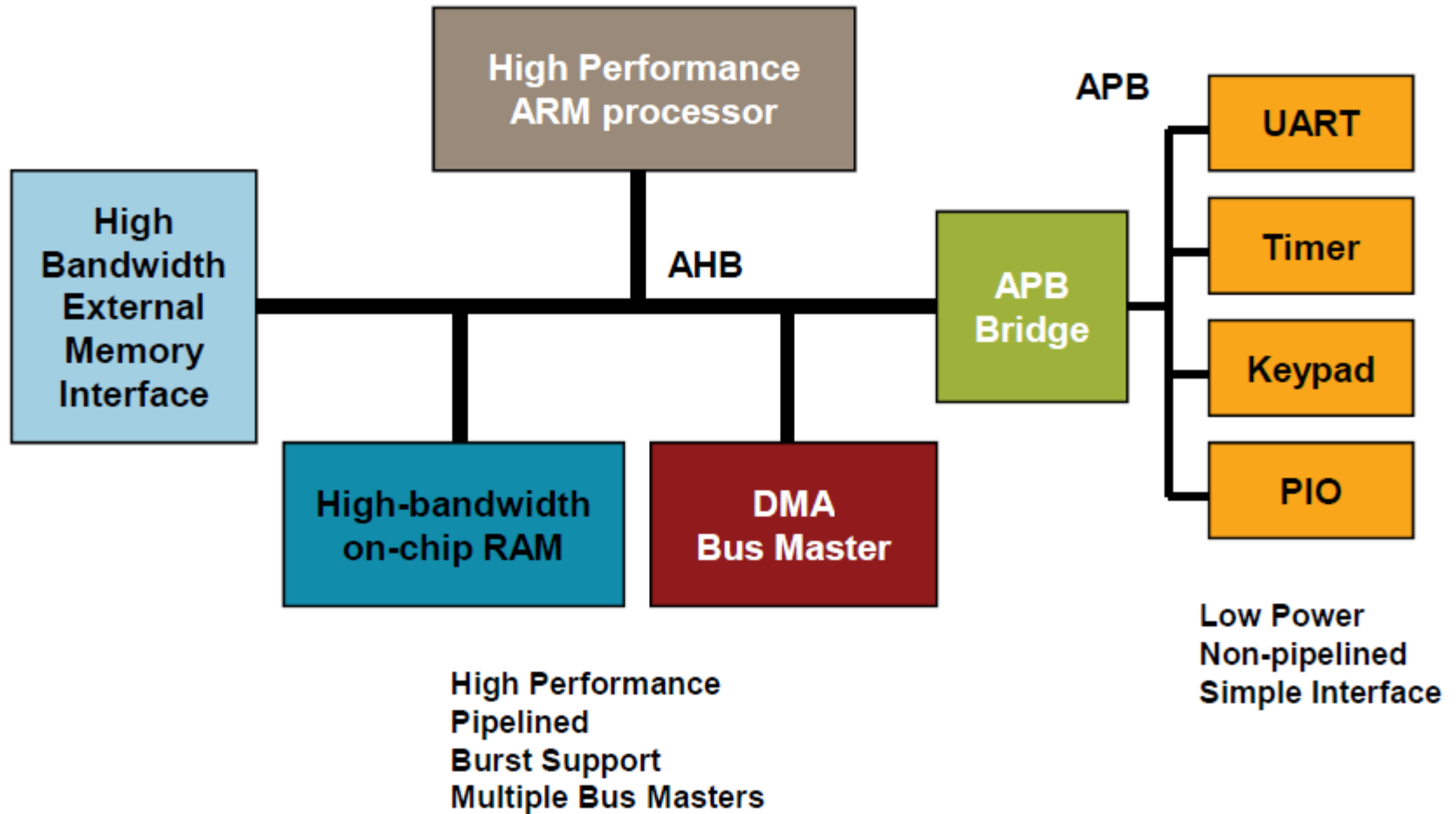


# ARM Advanced Microcontroller Bus Architecture (AMBA)

---

- ▶ **On-chip** interconnect specification for SoC
- ▶ Promote **re-use** by defining a common backbone for SoC modules using standard bus architectures
  - ▶ **AHB** – Advanced High-performance Bus (system backbone)
    - ▶ High-performance, high clock freq. modules
    - ▶ Processors to on-chip memory, off-chip memory interfaces
  - ▶ **APB** – Advanced Peripheral Bus
    - ▶ Lower performance requirements
    - ▶ Low-power peripherals
    - ▶ Reduced interface complexity
  - ▶ Others:
    - ▶ **ASB** – Advanced System Bus (high performance alternate to AHB)
    - ▶ **AXI** – Advanced eXtensible Interface
    - ▶ **ACE** – AXI Coherency Extension
    - ▶ **ATB** – Advanced Trace Bus

# Example AMBA System

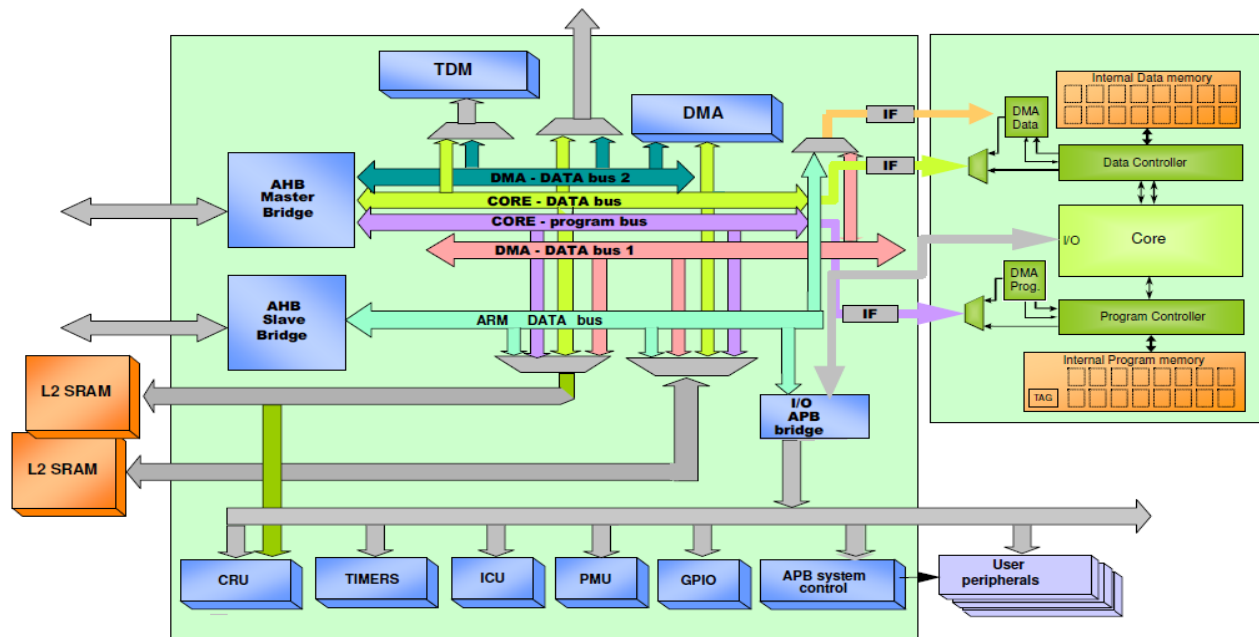




# Communication Architecture Standards

## Why do we need communication standards?

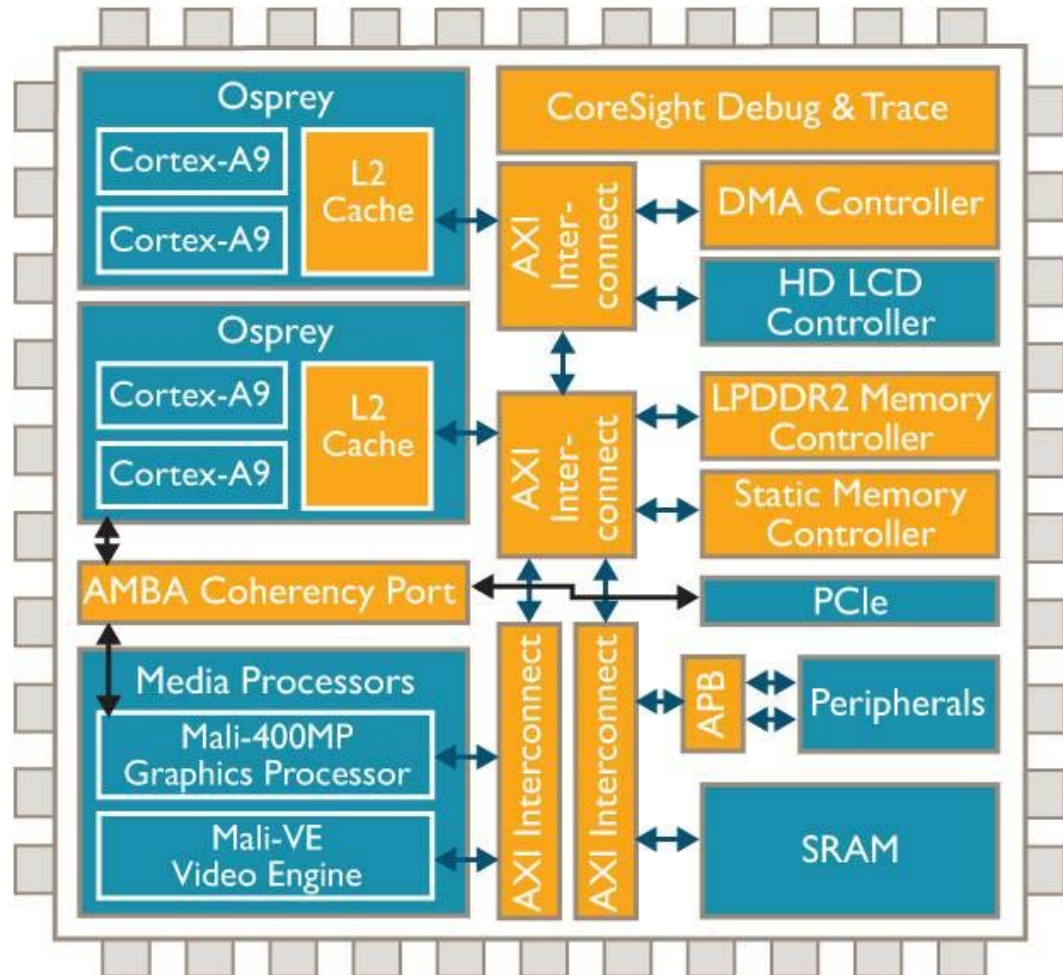
- ▶ Modular design approach
- ▶ Allows design reuse
- ▶ Facilitates IP integration into an SoC design



# ARM CoreLink peripherals for AMBA

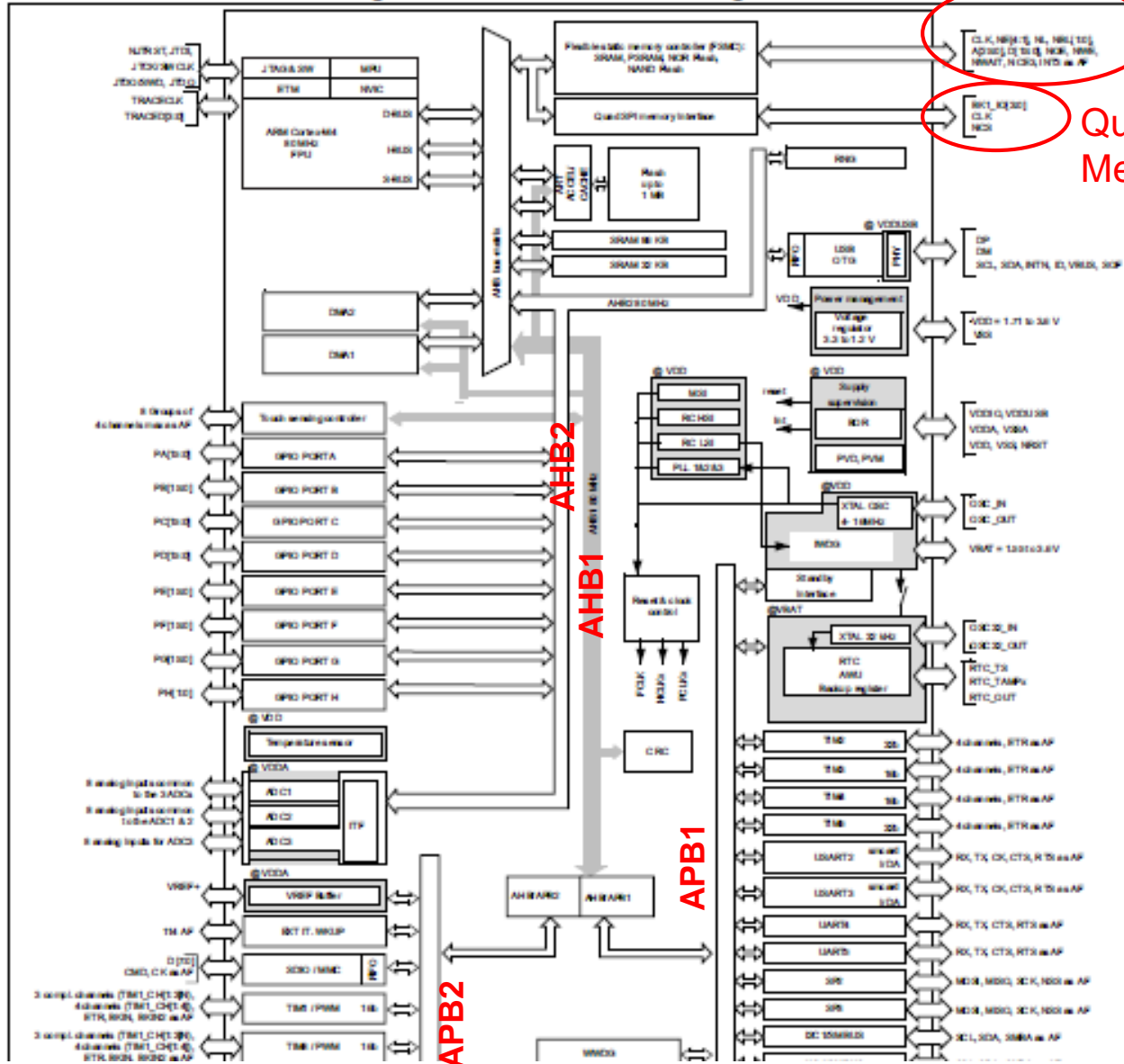
“CoreLink”  
(orange blocks)

Interconnect +  
memory controller  
IP for Cortex/Mali



# STM32L476G Microcontroller

Figure 1. STM32L476xx block diagram

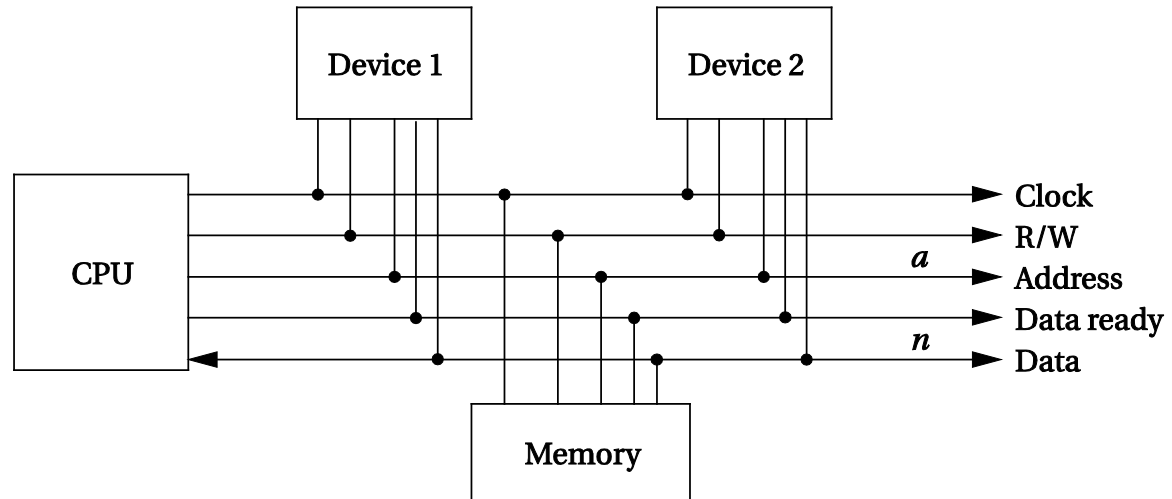


External Memory

Quad SPI Memory

# Microprocessor buses

- ▶ **Clock** provides synchronization.
- ▶ **R/W'** true when reading, false when writing.
  - ▶ May replace CLK and R/W with RD and WR strobes
- ▶ **Address** is a-bit bundle of address lines.
- ▶ **Data** is n-bit bundle of data lines.
- ▶ **Data ready** signals when n-bit data is ready.



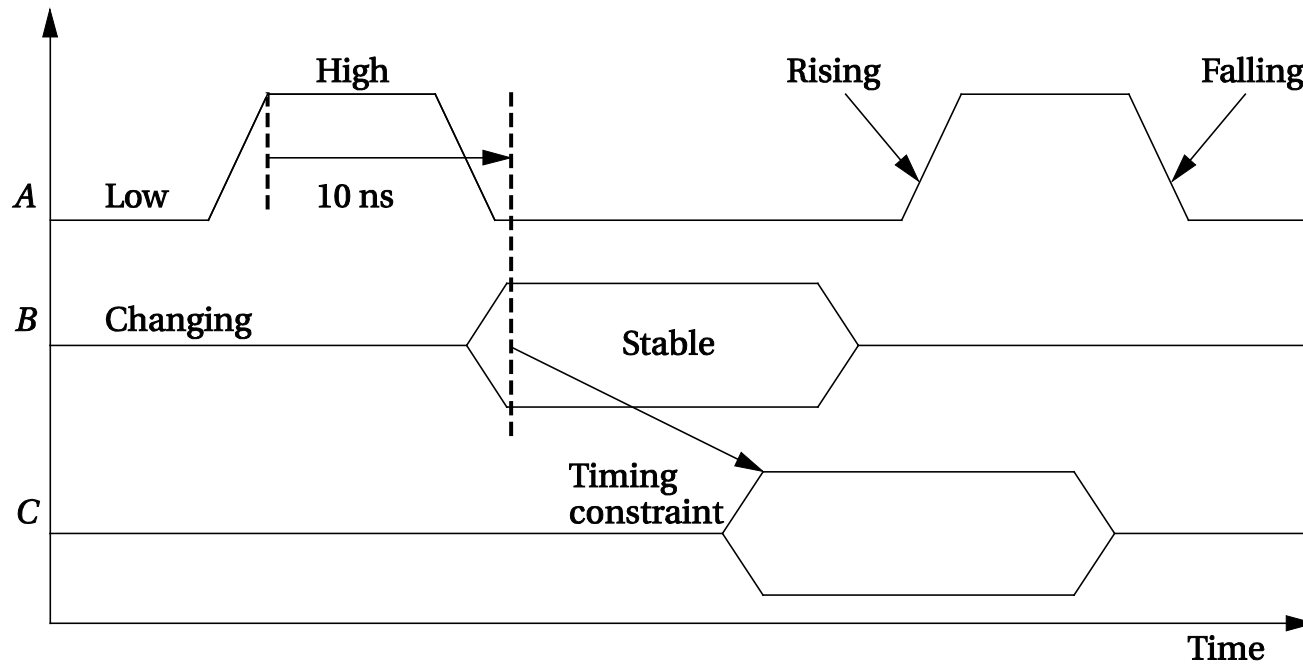
# Bus protocols

---

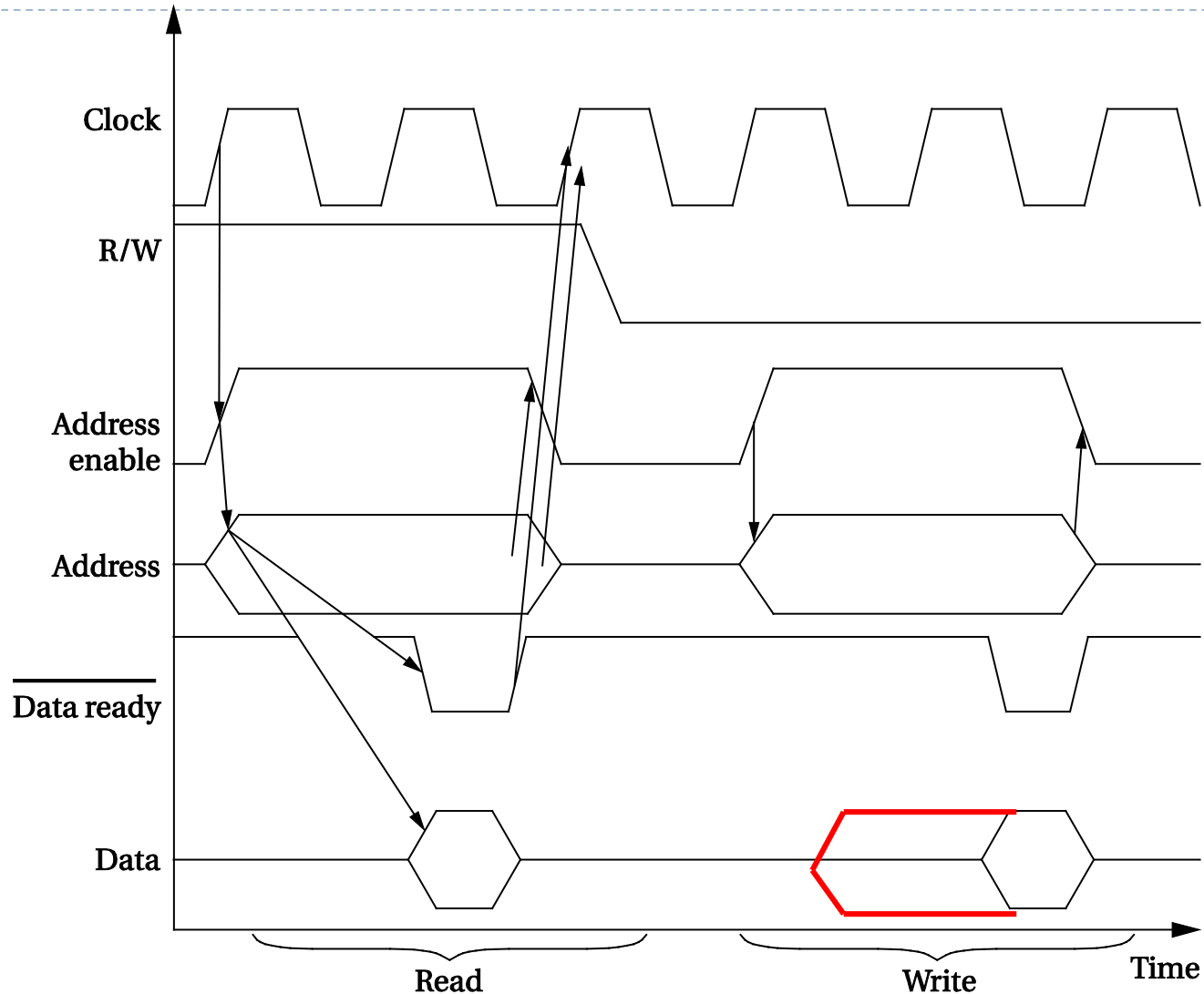
- ▶ Bus **protocol** determines how devices communicate.
- ▶ Devices on the bus go through sequences of **states**.
  - ▶ Protocols are specified by state machines,
  - ▶ One state machine per actor in the protocol.
- ▶ May contain synchronous and/or asynchronous logic behavior.
- ▶ Bus protocol often defined by **timing diagrams**

# Timing diagrams

---



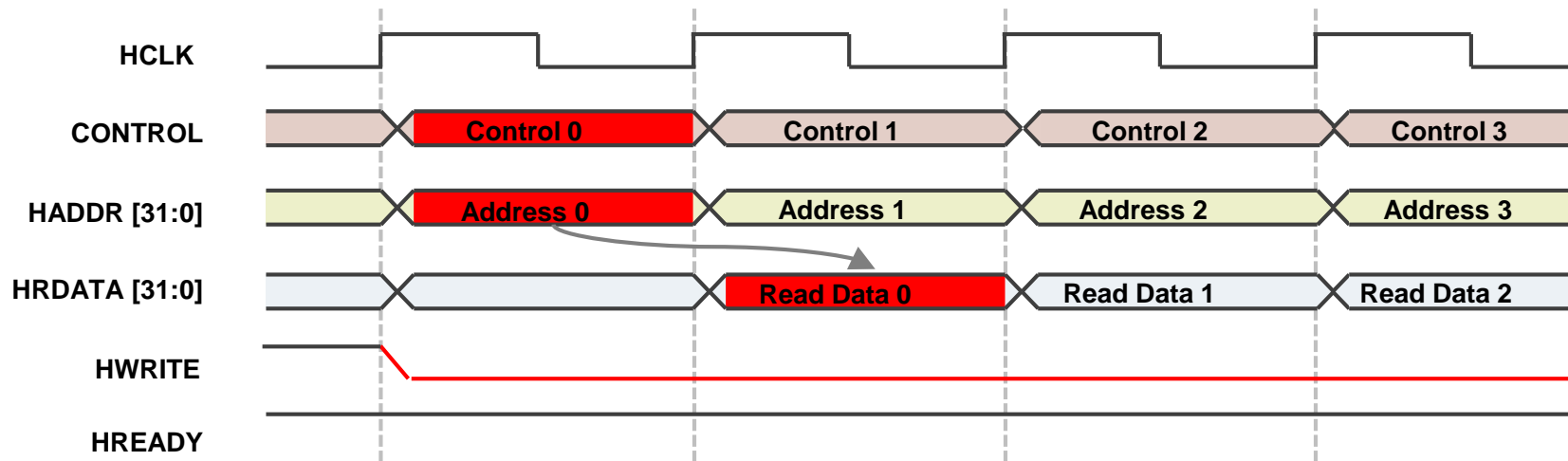
# Typical bus read and write timing



# Arm AHB: Basic Read Transfer

## Simple read transfer with no wait states:

- ▶ The address phase: The master drives the address and control signals onto the bus after the rising edge of HCLK.
- ▶ The data phase: The slave samples the address and control information and make data available at HRDATA before driving the appropriate HREADY response.

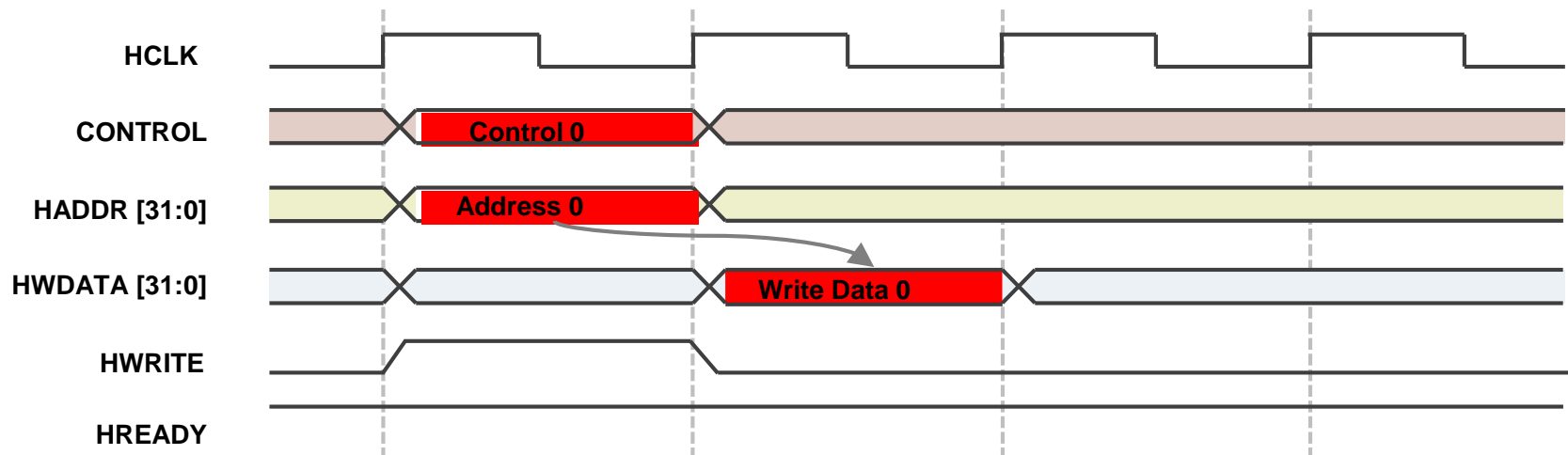




# Arm AHB: Basic Write Transfer

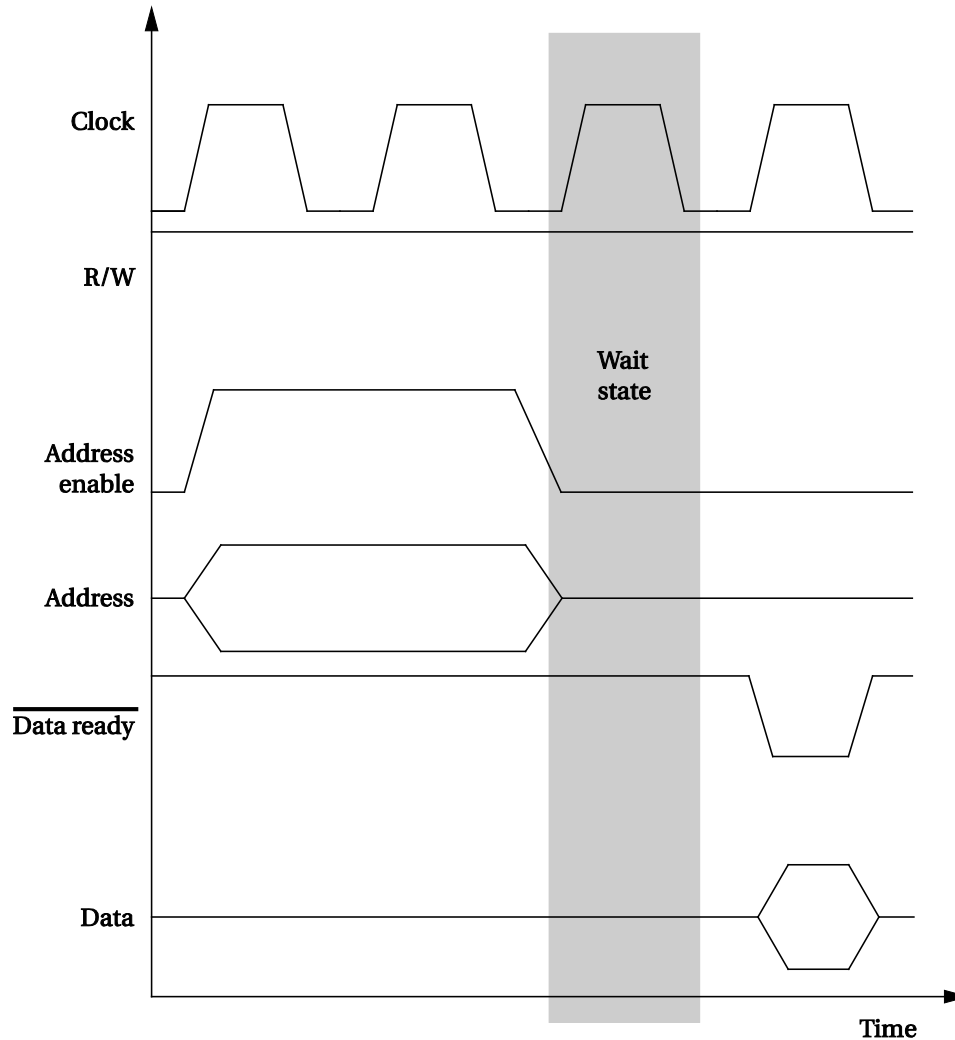
Simple write transfer with no wait states:

- ▶ The address phase: The master drives the address and control signals onto the bus after the rising edge of HCLK and sets HWRITE to one.
- ▶ The data phase: The slave samples the address and control information and captures the data from HWDATA before driving the appropriate HREADY response.



# Bus wait state

Extend  
read/write  
cycle if  
memory  
slower than  
CPU



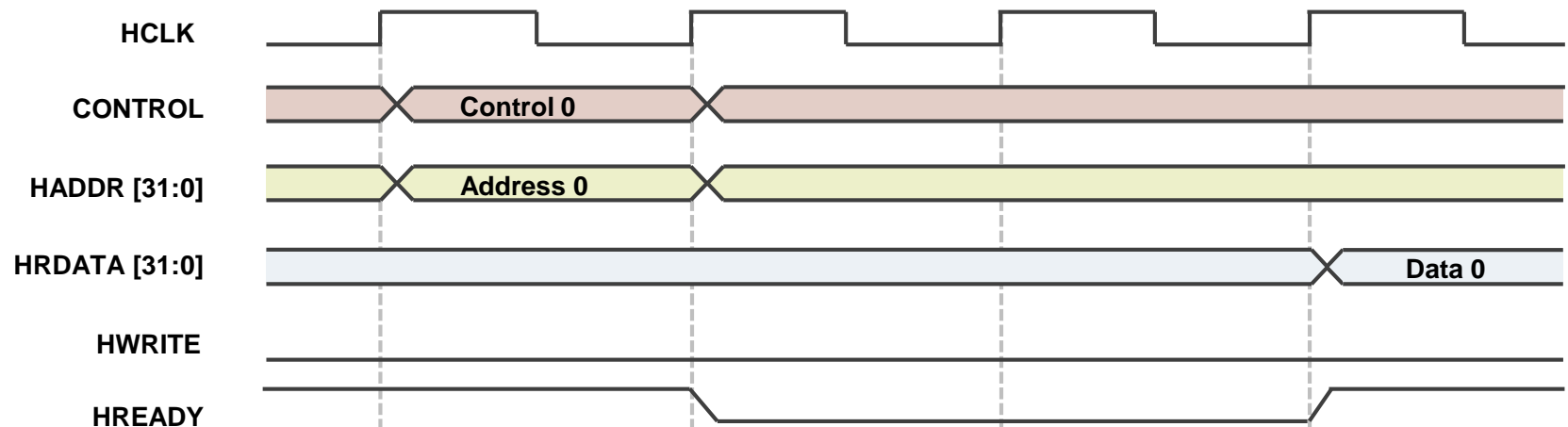
# Arm AHB: Read Transfer with Wait State

## Address phase (first clock cycle)

- ▶ Give address and control signals; set HWRITE to one.

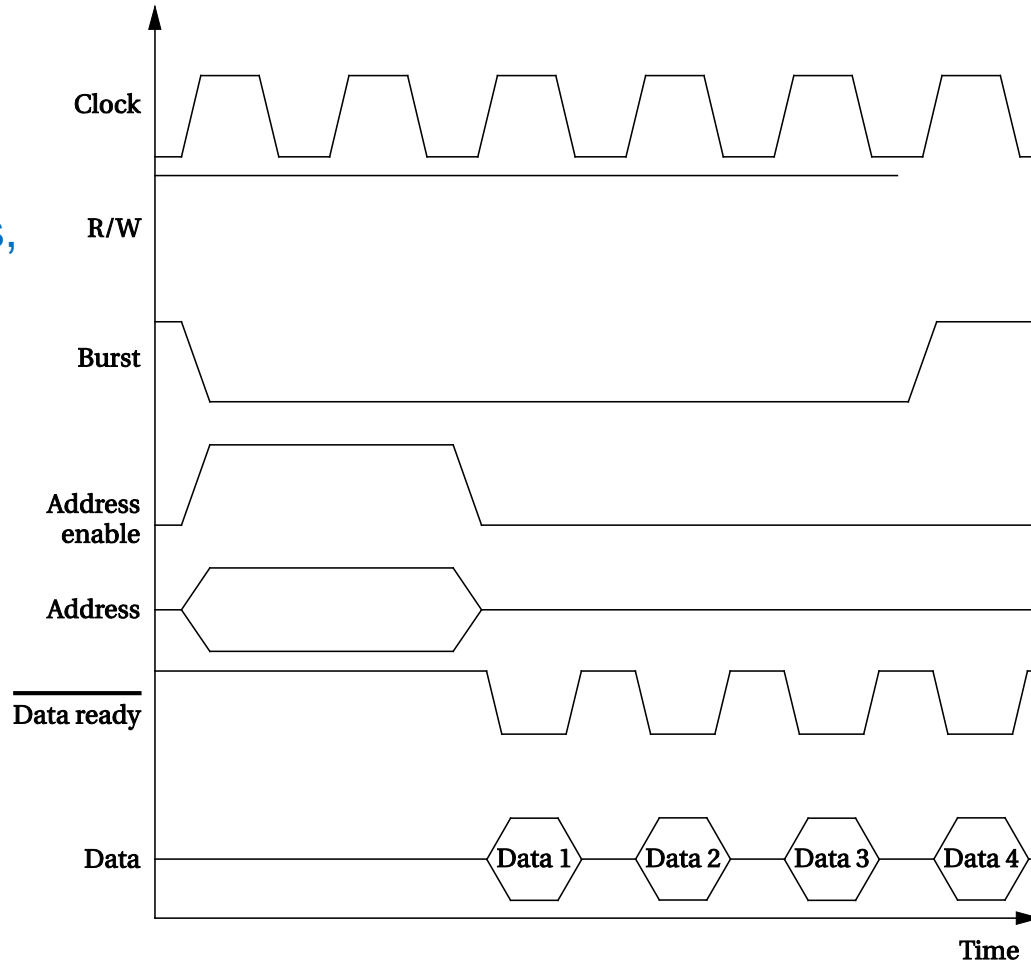
## Data phase (multiple clock cycles)

- ▶ The slave holds HREADY to zero if it is not ready to provide its data; the master delays its next transaction.
- ▶ When the slave is ready, the data will be given at HRDATA; at the same time, HREADY is set to one. The master will then continue its next transaction.

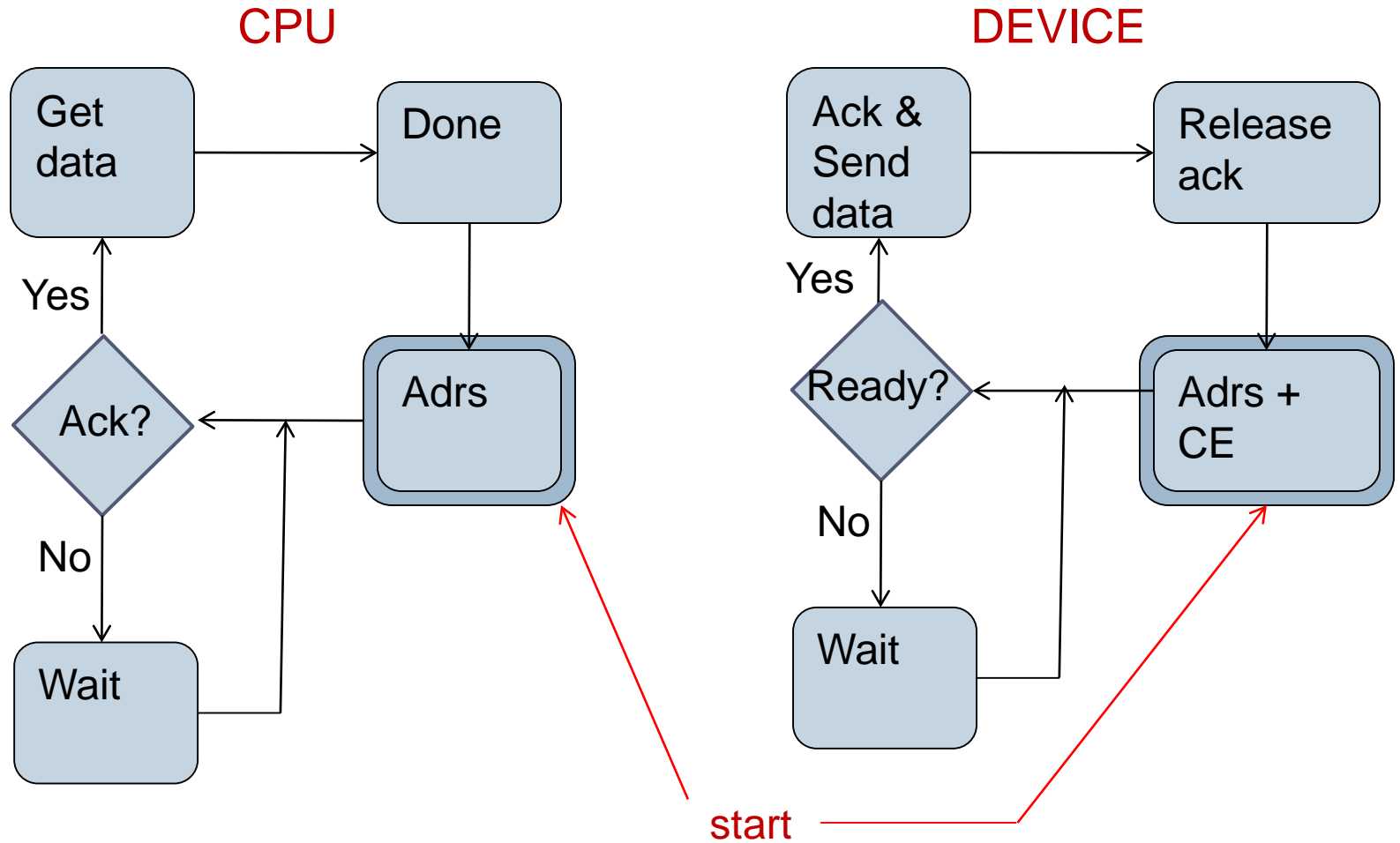


# Bus burst read

CPU sends start address, followed by burst of data from consecutive addresses

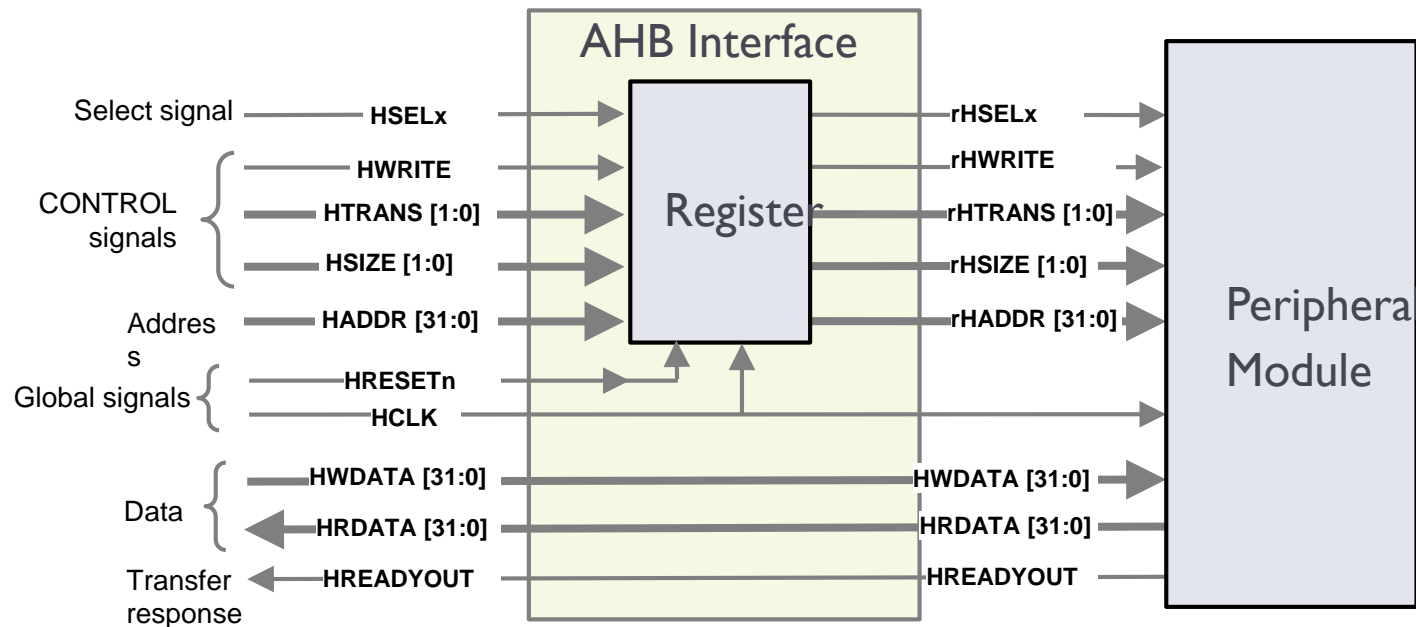


# State diagrams for bus read



# Arm AHB Interface

- Capture address and control signals in registers in one HCLK cycle.
- Transfer the corresponding data in the next HCLK cycle.



# Read-only memory types

---

- ▶ Mask-programmed ROM
  - ▶ Programmed at factory (*high NRE cost*)
- ▶ PROM (Programmable ROM)
  - ▶ Programmable once by users (*low NRE cost*)
    - ▶ Electric pulses selectively applied to “fuses” or “antifuses”
- ▶ EPROM (Erasable PROM)
  - ▶ Repeatedly programmable/reprogrammable
    - Electric pulses for programming (seconds)
    - Ultraviolet light for erasing (15-20 minutes)
- ▶ EEPROM (Electrically Erasable PROM)
  - ▶ Electrically programmable and erasable at the single-byte level (*msec*)
- ▶ Flash EPROM
  - ▶ Electrically programmable ( $\mu$ sec) and
  - ▶ Electrically erasable (block-by-block: *msec* to *sec*)
  - ▶ Structures: NOR (random access); NAND (sequential access)
  - ▶ **Most common program memory in embedded applications**
  - ▶ **Widely used in digital cameras, multimedia players, smart phones, etc.**



# Read-write memory types

---

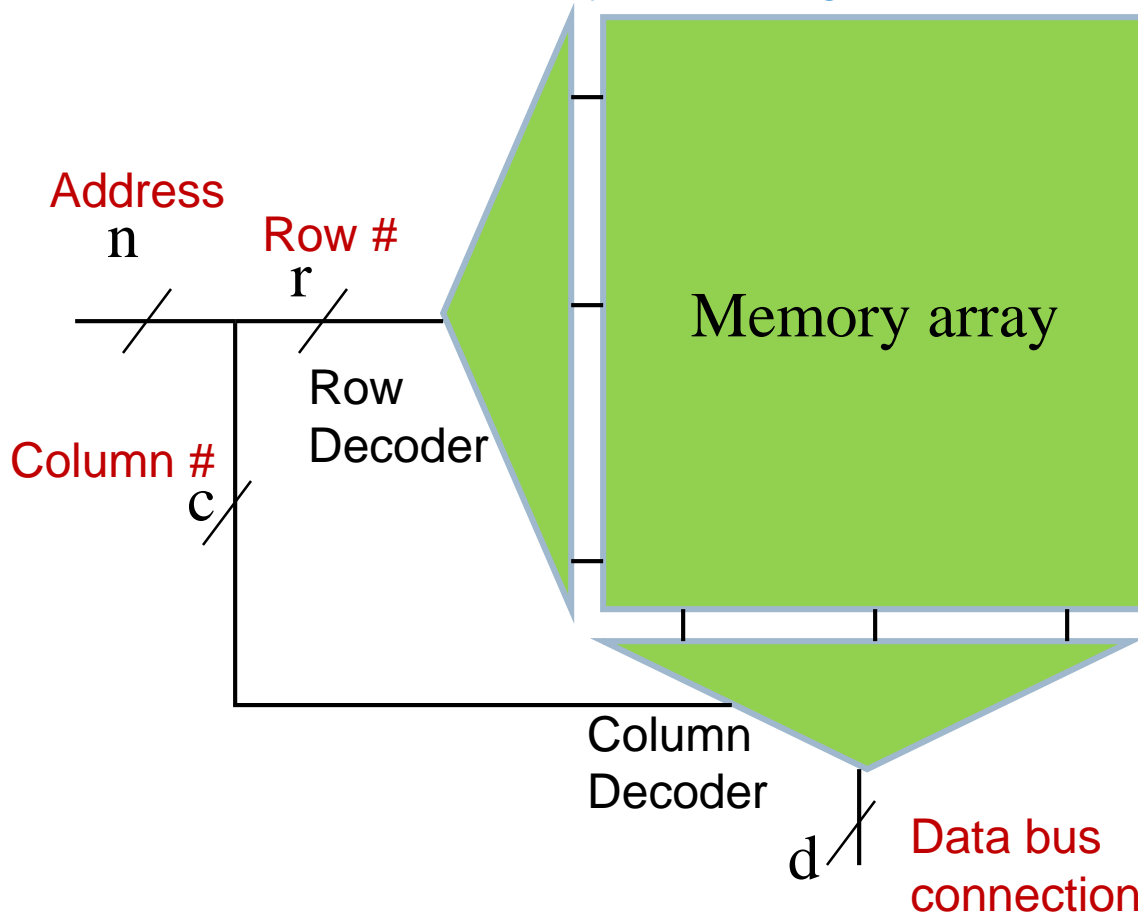
- ▶ **Static RAM (SRAM)**
  - ▶ Each cell is a flip-flop, storing 1-bit information which is retained as long as power is on
  - ▶ Faster than DRAM
  - ▶ Requires a larger area per cell than DRAM
- ▶ **Dynamic RAM (DRAM)**
  - ▶ Each cell is a capacitor, which needs to be refreshed periodically to retain the 1-bit information
  - ▶ A refresh consists of reading followed by writing back
  - ▶ Refresh overhead





# ROM/RAM device organization

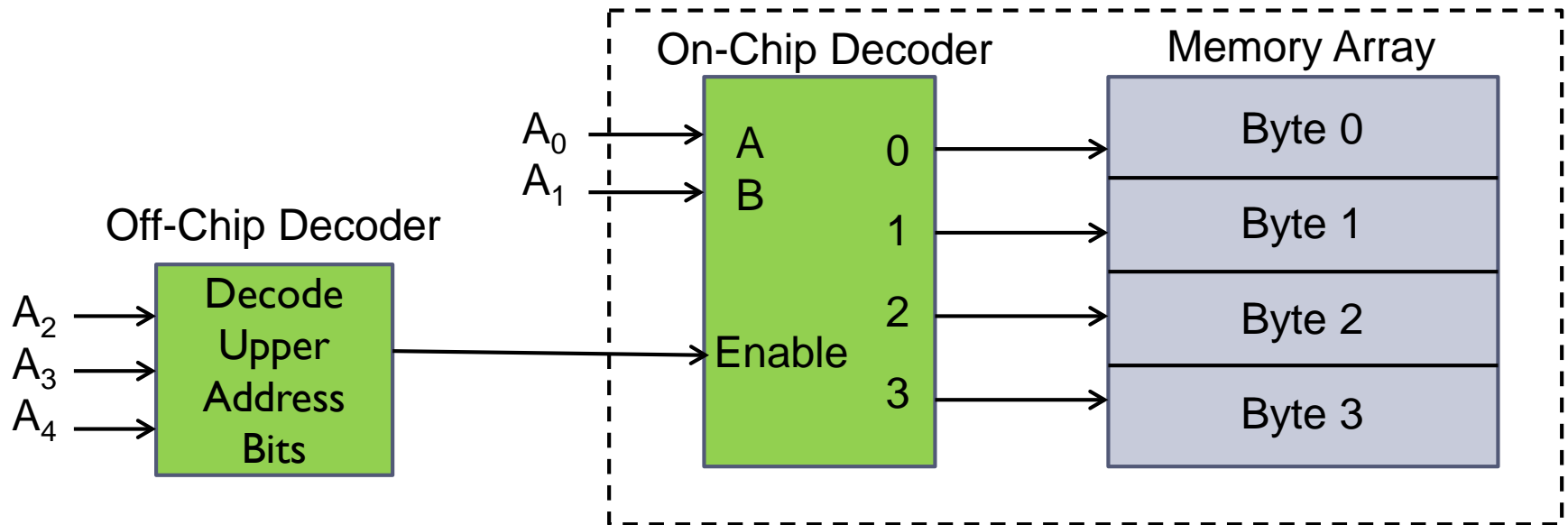
Memory "organization" =  $2^n \times d$   
(from system designer's perspective)



- ▶ **Size.**
  - ▶  $2^n$  addressable words
  - ▶ Address width =  $n = r + c$
- ▶ **Aspect ratio.**
  - ▶ Data width  $d$ .

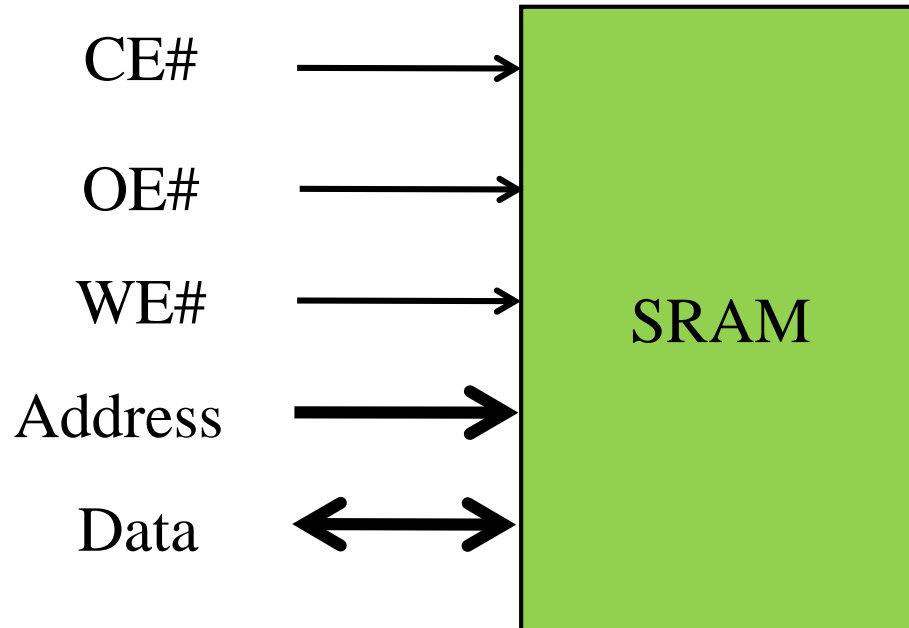
# Memory address decoding

- ▶ Select a sub-space of memory addresses
- ▶ A simple example
  - ▶ Microprocessor with 5 address bits ( $A_4 \dots A_0$ )  $\rightarrow 2^5 = 32$  bytes addressable
  - ▶ Assume 4 byte ( $4 \times 8$ ) memory chip  $\rightarrow$  Decodes two address bits ( $A_1 A_0$ )
  - ▶  $\mu$ P can address up to 8 chips (decode address bits ( $A_4 A_3 A_2$ )) for chip enable



# Typical generic SRAM

---



CE# = **chip enable**: initiate memory access when active

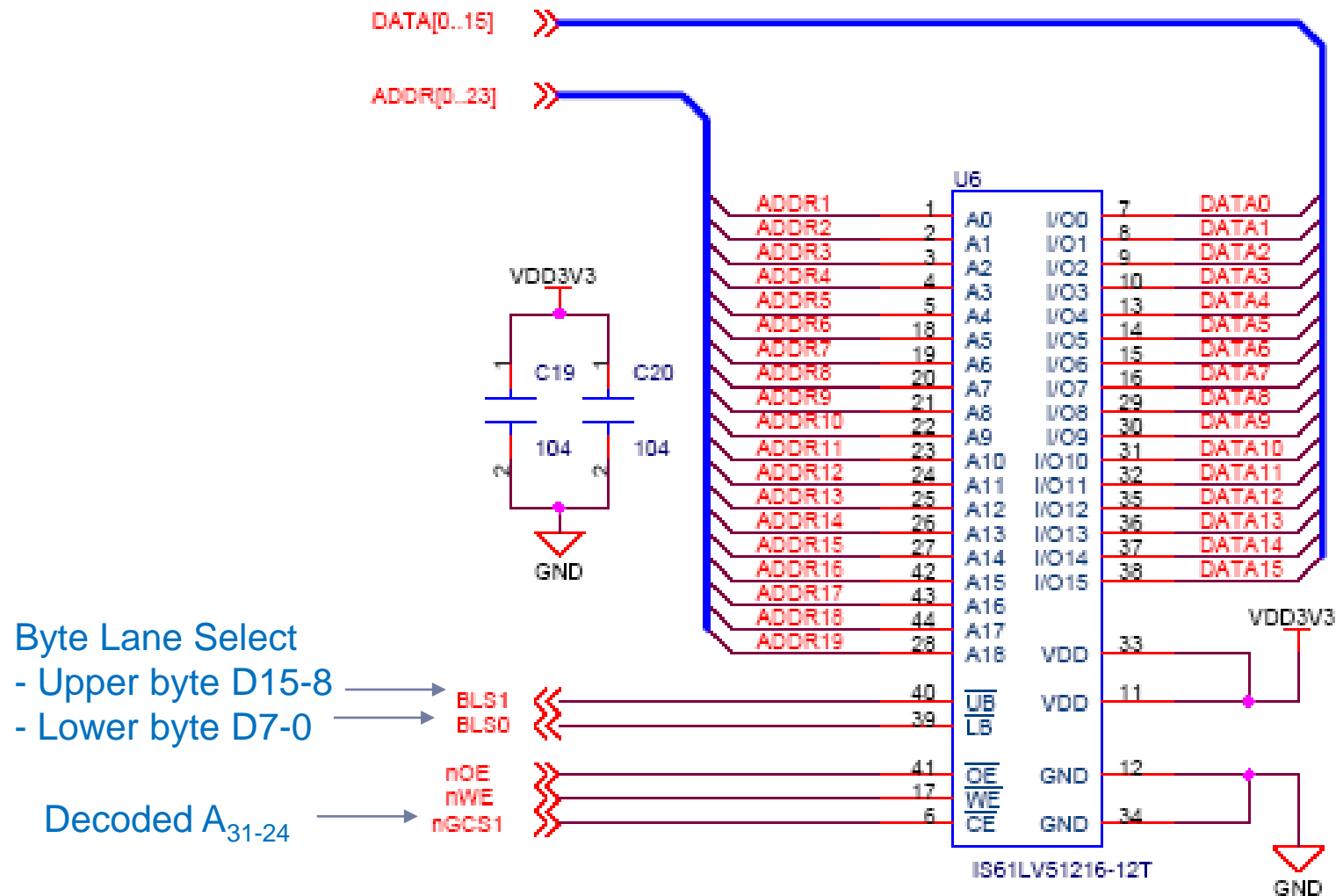
OE# = **output enable**: drive Data lines when active

WE# = **write enable**: update SRAM contents with Data

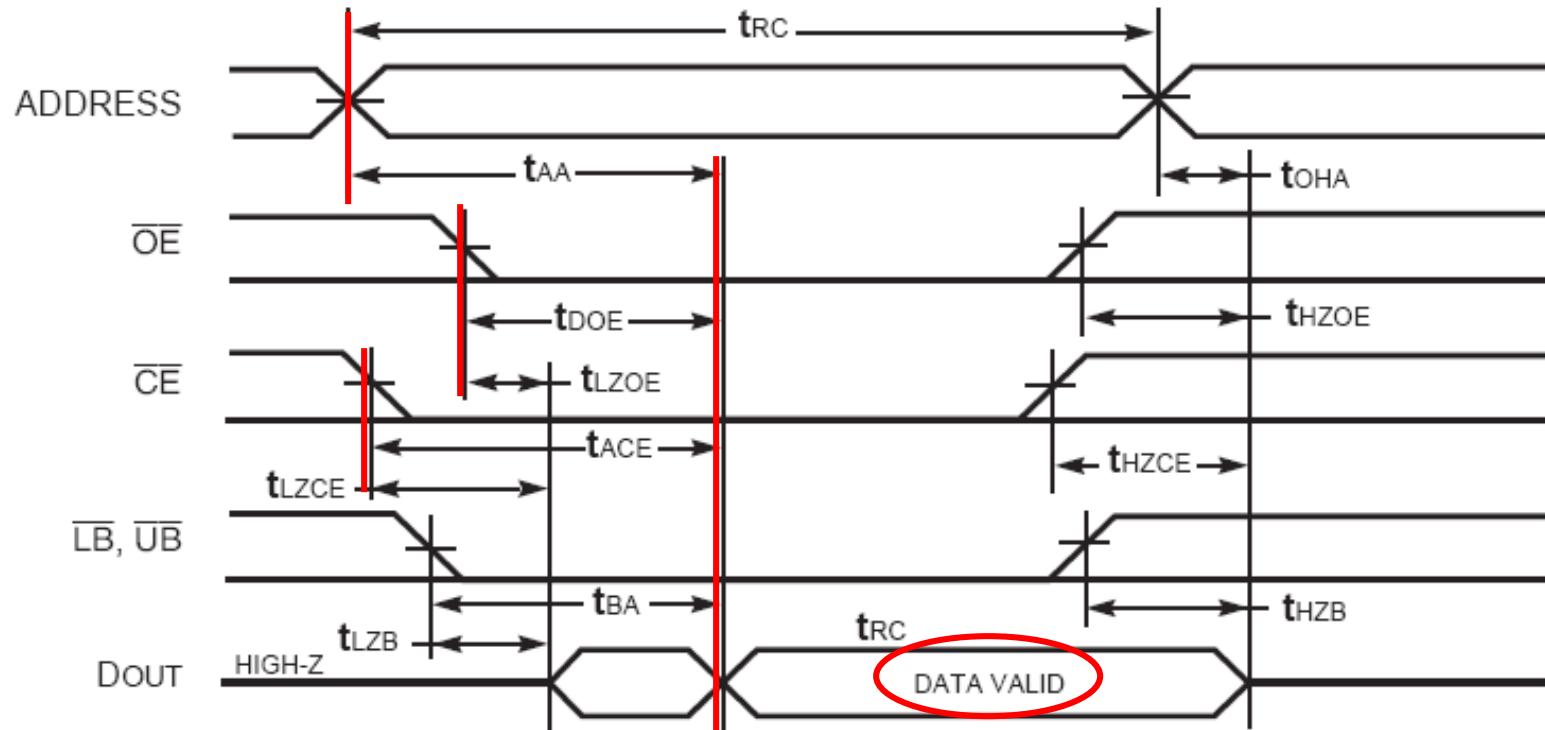
(May have one R/W# signal instead of OE# and WE#)

Multi-byte data bus devices have a **byte-enable** signal for each byte.

# IS61LV51216-12T: 512K x 16 SRAM (on uCdragon board)



# ISSI IS61LV51216 SRAM read cycle



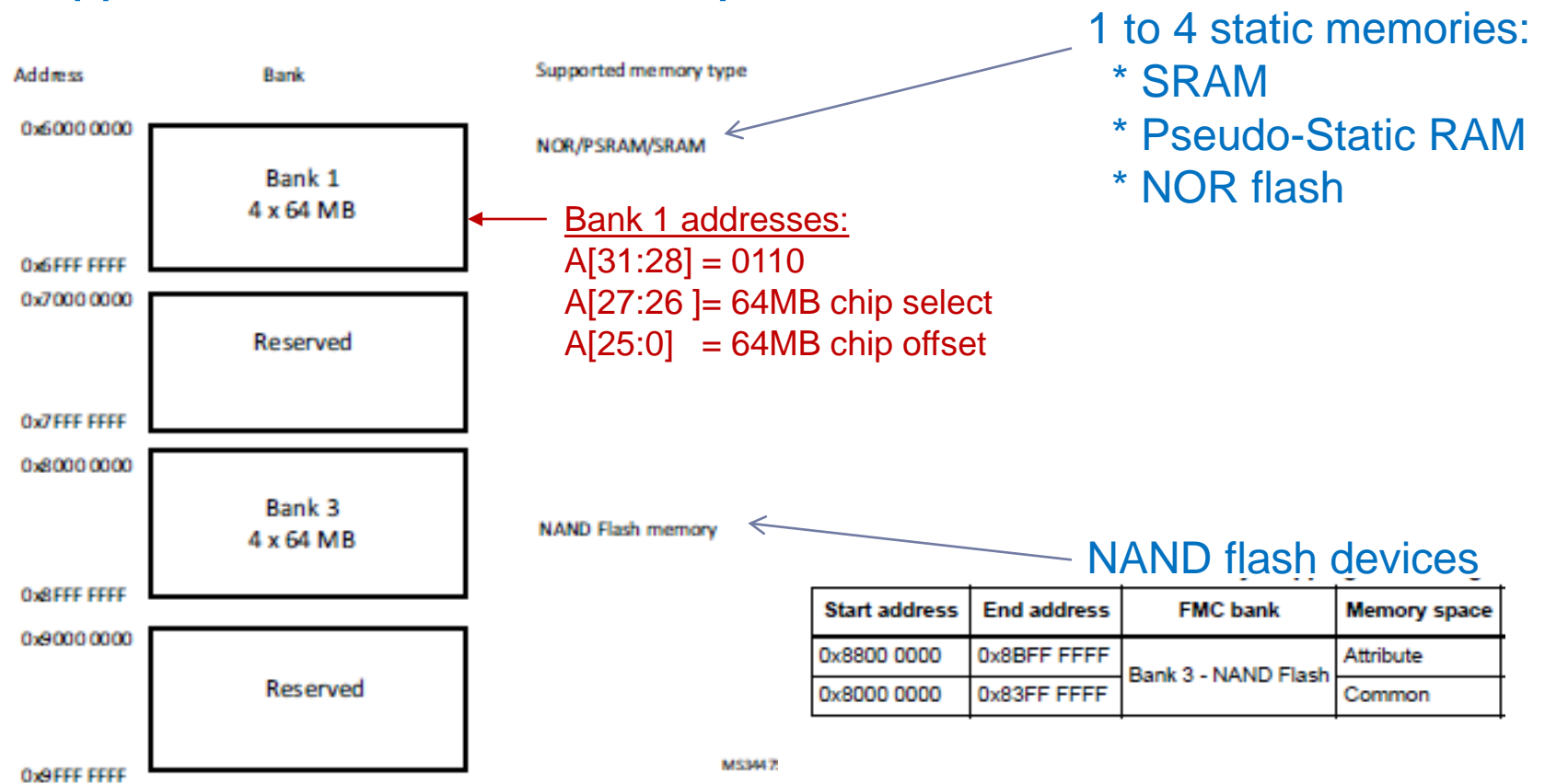
Timing Parameters:  
Max data valid times  
following activation of  
Address, CE, OE

Symbol	Parameter	-8		-10		-12		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>RC</sub>	Read Cycle Time	8	—	10	—	12	—	ns
t <sub>AA</sub>	Address Access Time	—	8	—	10	—	12	ns
t <sub>OHA</sub>	Output Hold Time	3	—	3	—	3	—	ns
t <sub>ACE</sub>	$\overline{\text{CE}}$ Access Time	—	8	—	10	—	12	ns
t <sub>DOE</sub>	$\overline{\text{OE}}$ Access Time	—	3.5	—	4	—	5	ns

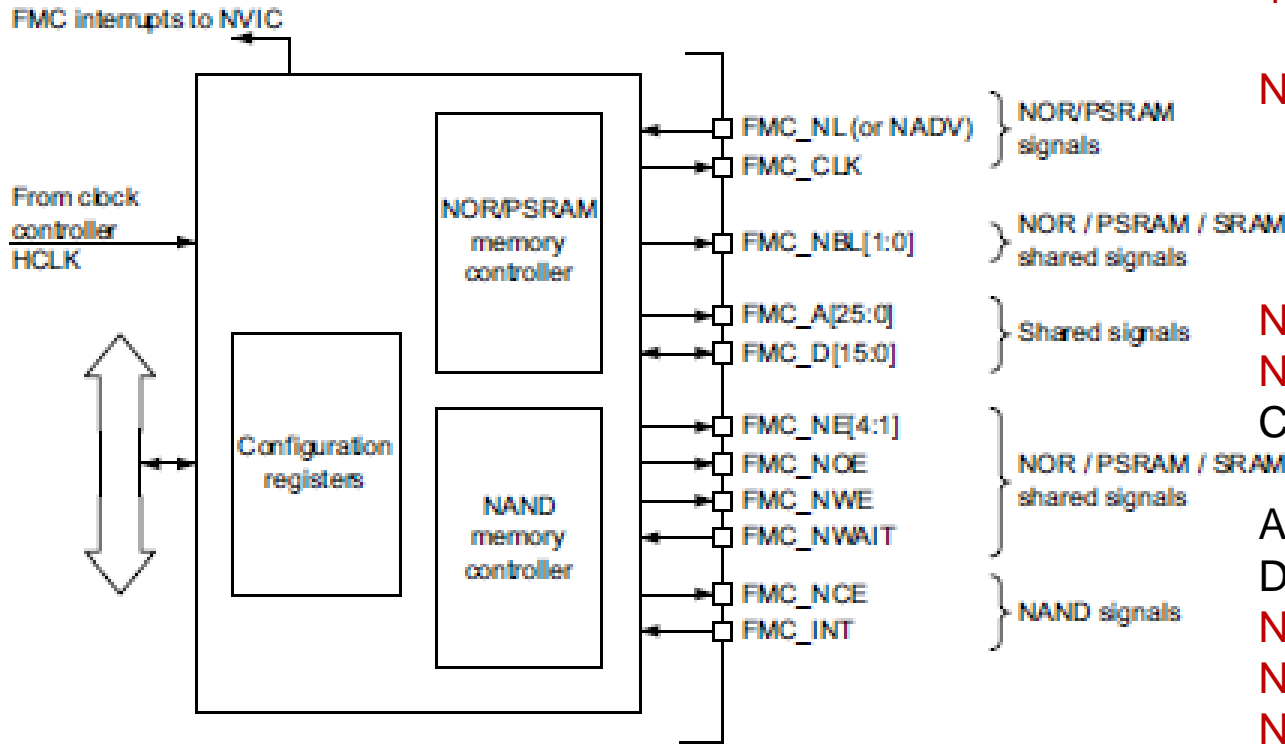
# STM32 Flexible Static Memory Controller (FSMC)

## STM32L4x6 Tech. Ref. Manual, Chap. 16

- ▶ Control external memory on AHB bus in four 256M banks
- ▶ Upper address bits decoded by the FSMC



# FSMC block diagram



“N” = “negative” (active low)

NE[4:1] = NOR/PSRAM enable

NE[1]: A[27:26]=00

NE[2]: A[27:26]=01

NE[3]: A[27:26]=10

NE[4]: A[27:26]=11

NL = address latch/advance

NBL = byte lane

CLK for sync. Burst

A[25:0] = Address bus

D[15:0] = Data bus\*\*

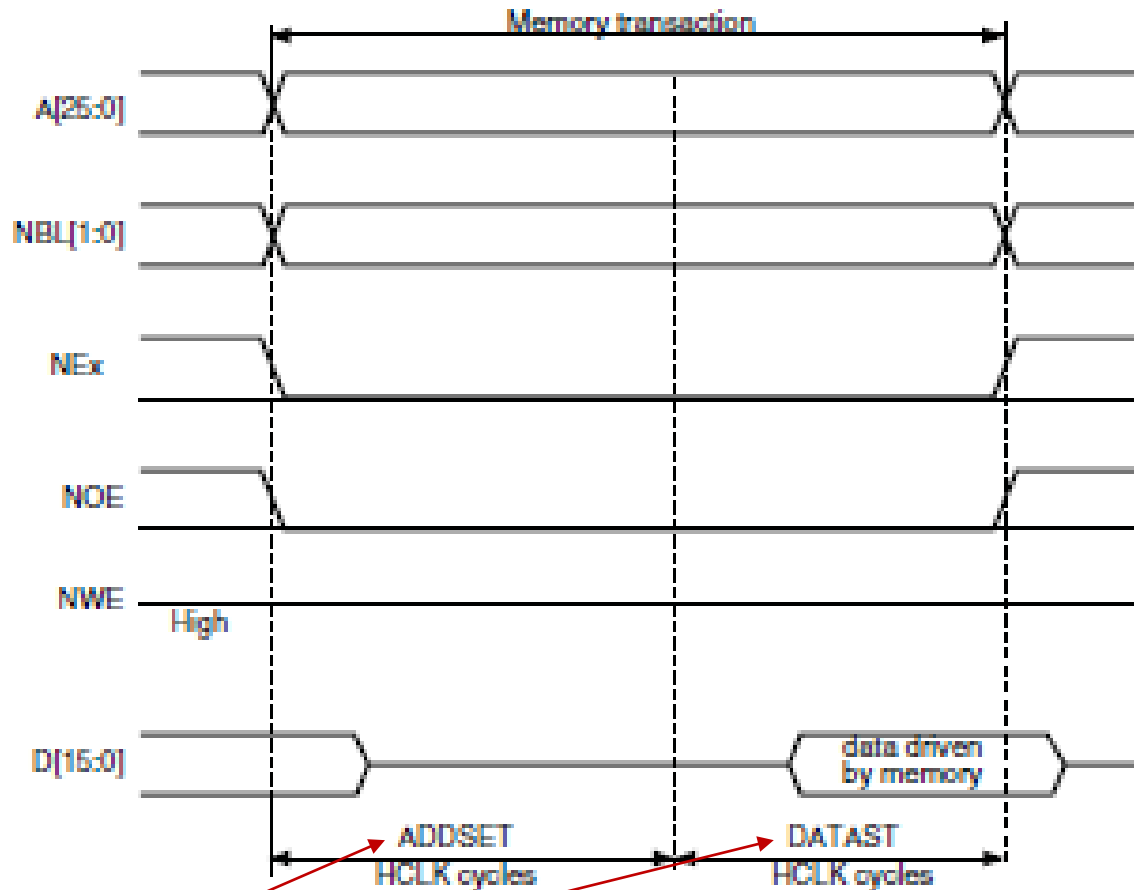
NOE = output enable

NWE = write enable

NWAIT = wait request

\*\* Data bus = 8 or 16 bits

# FSMC “Mode 1” memory read



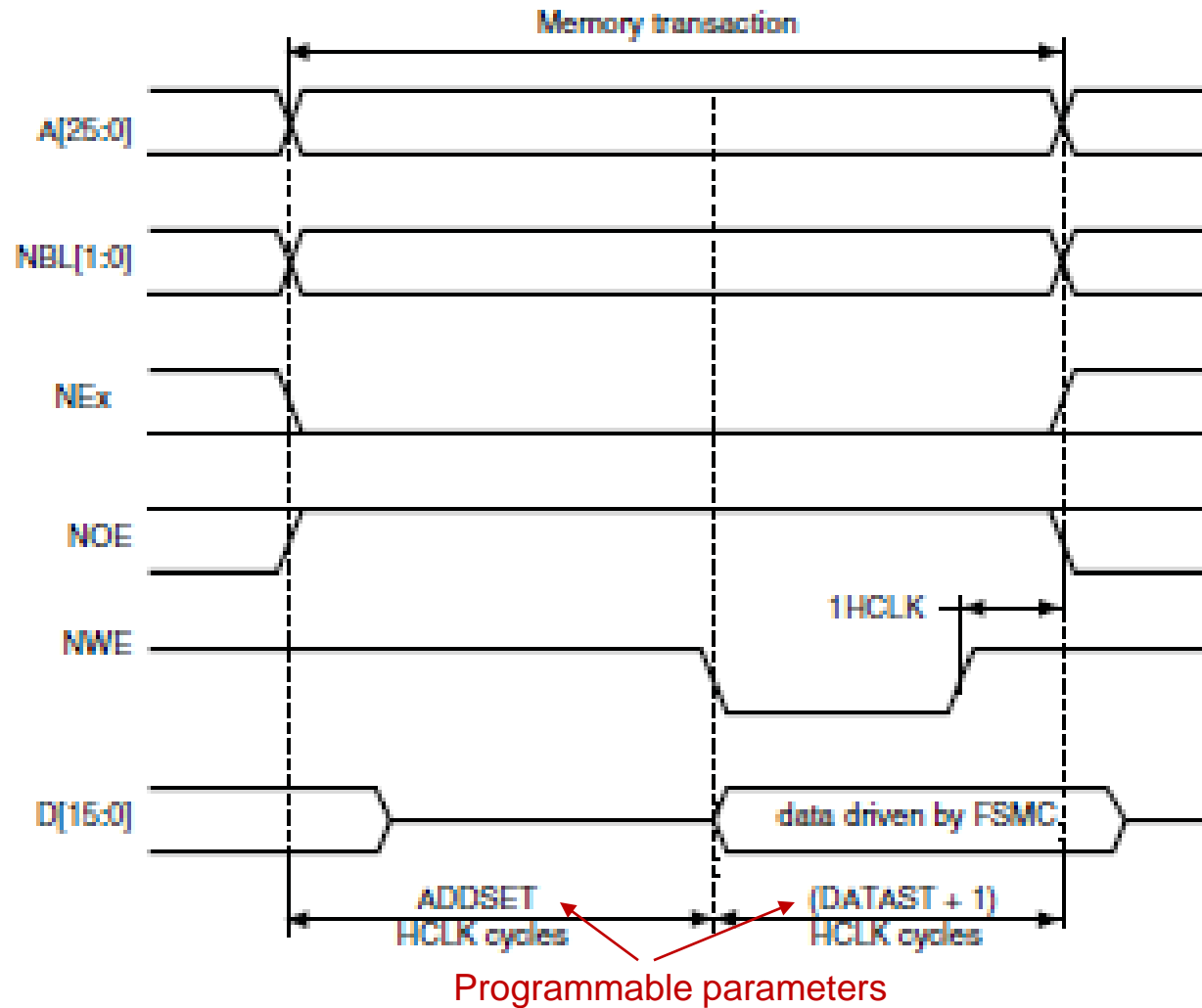
Other modes:

- \* Provide ADV (address latch/advance)
- \* Activate OE and WE only in DATAST
- \* Multiplex A/D bits 15-0
- \* Allow WAIT to extend DATAST

ADDSET/DATAST programmed in chip-select timing register (HCLK = AHB clock)



# FSMC “Mode 1” memory write



# Flash memory devices

---

- ▶ Flash memory is programmed at system voltages.
- ▶ Erasure time is long.
- ▶ Must be erased in blocks.
- ▶ Available in NAND or NOR structures
  - ▶ **NOR**: memory cells in parallel – allows random access
  - ▶ **NAND**: memory cells in series – sequential access/60% smaller

	<b>SLC NAND Flash (x8)</b>	<b>MLC NAND Flash (x8)</b>	<b>MLC NOR Flash (x16)</b>
<b>Density</b>	512 Mbits <sup>1</sup> – 4 Gbits <sup>2</sup>	1Gbit to 16Gbit	16Mbit to 1Gbit
<b>Read Speed</b>	24 MB/s <sup>3</sup>	18.6 MB/s	103MB/s
<b>Write Speed</b>	8.0 MB/s	2.4 MB/s	0.47 MB/s
<b>Erase Time</b>	2.0 mSec	2.0mSec	900mSec
<b>Interface</b>	Serial access	Serial access	Random access
<b>Application</b>	Program/Data mass storage	Program/Data mass storage	Program memory

SLC = Single-Level Cell, MLC = Multi-Level Cell

---

# NAND and NOR flash comparison

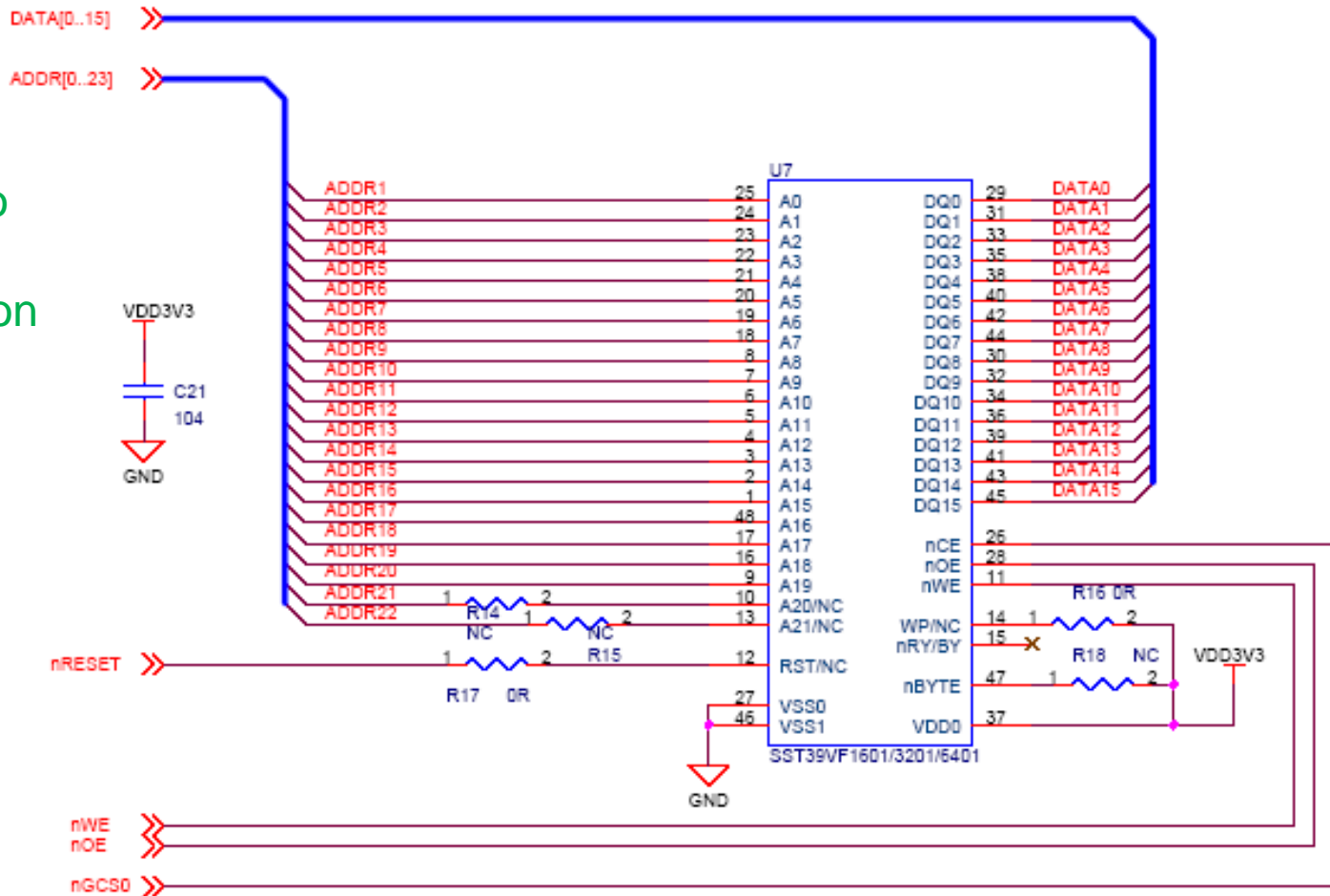
NAND flash similar to a hard disk drive (sequential access to bits of a sector)

NOR flash similar to a Random-access memory (ROM/RAM)

	NAND	NOR
Cell Array	<p>Diagram showing the NAND cell array structure. It features a vertical blue bit line, horizontal orange word lines, and a horizontal green source line. A dashed box highlights a single unit cell. Labels include 'Contact', 'Word line', 'Unit Cell', 'Bit line', and 'Source line'.</p>	<p>Diagram showing the NOR cell array structure. It features a vertical blue bit line, horizontal orange word lines, and a horizontal green source line. A dashed box highlights a single unit cell. Labels include 'Bit line', 'Contact', 'Word line', 'Unit Cell', 'Bit line', and 'Source line'.</p>
Layout	<p>Layout diagram of NAND flash showing a unit cell with dimensions <math>2F</math> by <math>2F</math>.</p>	<p>Layout diagram of NOR flash showing a unit cell with dimensions <math>5F</math> by <math>2F</math>.</p>
Cross Section	<p>Cross-section diagram of NAND flash showing a vertical stack of cells.</p>	<p>Cross-section diagram of NOR flash showing a horizontal stack of cells.</p>
Cell Size	$4F^2$	$10F^2$

# SST39VF1601- 1M x 16 NOR Flash (on uCdragon board)

Similar to  
SRAM  
connection

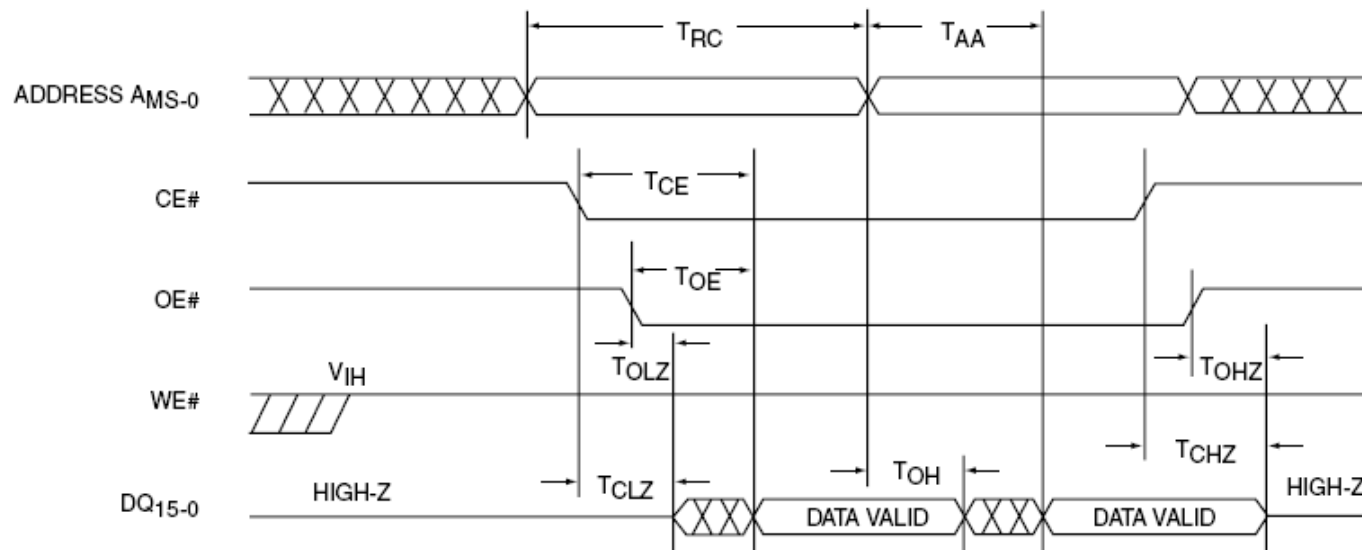


# SST39VF1601 characteristics

---

- ▶ **Organized as 1M x 16**
  - ▶ 2K word sectors, 32K word blocks
- ▶ **Performance:**
  - ▶ Read access time = 70ns or 90ns
  - ▶ Word program time = 7us
  - ▶ Sector/block erase time = 18ms
  - ▶ Chip erase time = 40ms
- ▶ **Check status of write/erase operation via read**
  - ▶ DQ7 = complement of written value until write complete
  - ▶ DQ7=0 during erase, DQ7=1 when erase done

# SST39VF1601 read cycle timing



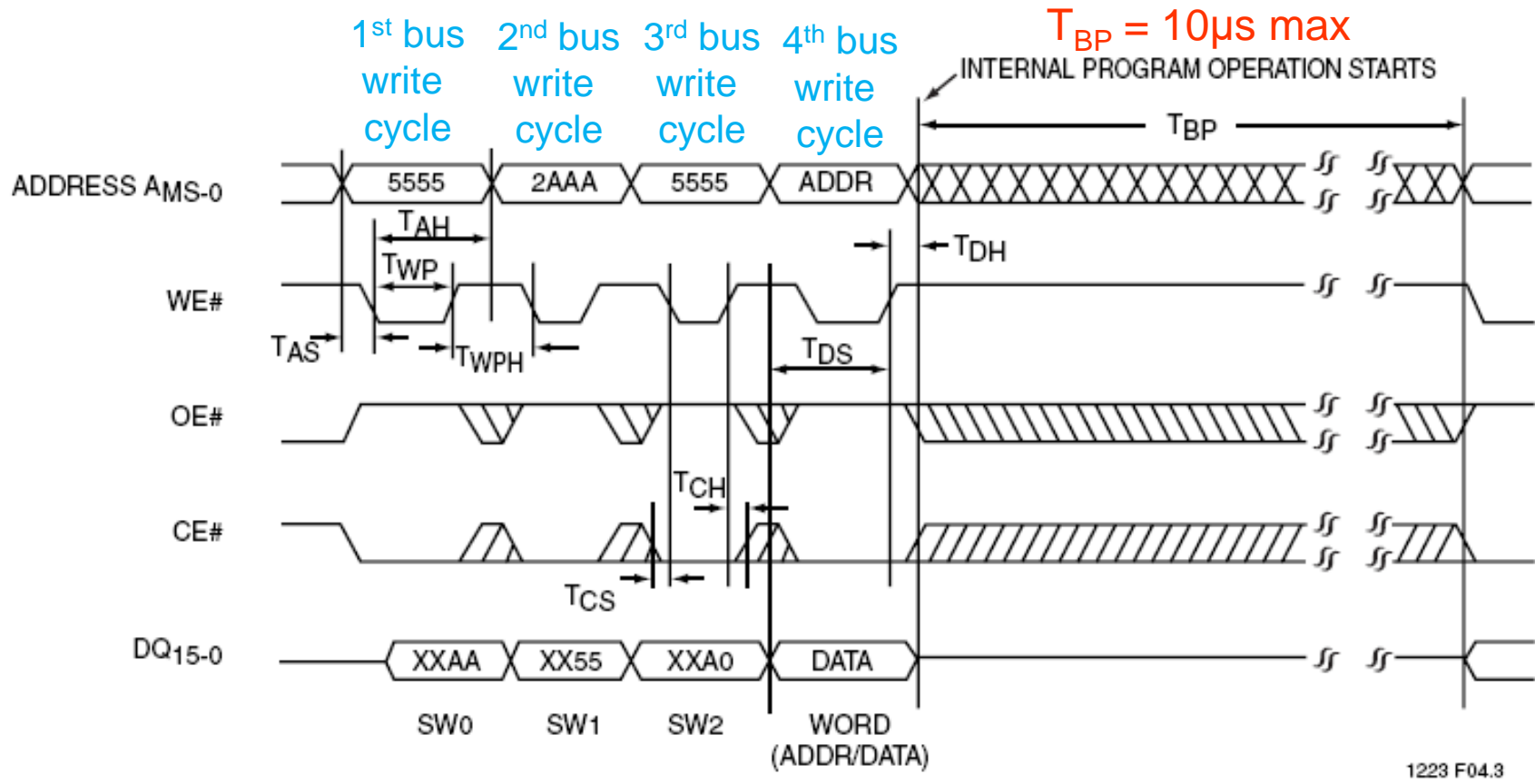
Symbol	Parameter	SST39VFxx01/xx02-70		SST39VFxx01/xx02-90		Units
		Min	Max	Min	Max	
$T_{RC}$	Read Cycle Time	70		90		ns
$T_{CE}$	Chip Enable Access Time		70		90	ns
$T_{AA}$	Address Access Time		70		90	ns
$T_{OE}$	Output Enable Access Time		35		45	ns
$T_{CLZ}^1$	CE# Low to Active Output	0		0		ns
$T_{OLZ}^1$	OE# Low to Active Output	0		0		ns
$T_{CHZ}^1$	CE# High to High-Z Output		20		30	ns
$T_{OHZ}^1$	OE# High to High-Z Output		20		30	ns
$T_{OH}^1$	Output Hold from Address Change	0		0		ns
$T_{RP}^1$	RST# Pulse Width	500		500		ns
$T_{RHR}^1$	RST# High before Read	50		50		ns
$T_{RY}^{1,2}$	RST# Pin Low to Read Mode		20		20	$\mu$ s

# SST39VF1601 command sequences

Assert Address, Data, WE# and CE# to write a command

Command Sequence	1st Bus Write Cycle		2nd Bus Write Cycle		3rd Bus Write Cycle		4th Bus Write Cycle		5th Bus Write Cycle		6th Bus Write Cycle	
	Addr <sup>1</sup>	Data <sup>2</sup>	Addr <sup>1</sup>	Data <sup>2</sup>	Addr <sup>1</sup>	Data <sup>2</sup>	Addr <sup>1</sup>	Data <sup>2</sup>	Addr <sup>1</sup>	Data <sup>2</sup>	Addr <sup>1</sup>	Data <sup>2</sup>
Word-Program	5555H	AAH	2AAAH	55H	5555H	A0H	WA <sup>3</sup>	Data				
Sector-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	SA <sub>x</sub> <sup>4</sup>	30H
Block-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	BA <sub>x</sub> <sup>4</sup>	50H
Chip-Erase	5555H	AAH	2AAAH	55H	5555H	80H	5555H	AAH	2AAAH	55H	5555H	10H
Erase-Suspend	XXXXH	B0H										
Erase-Resume	XXXXH	30H										
Query Sec ID <sup>5</sup>	5555H	AAH	2AAAH	55H	5555H	88H						
User Security ID Word-Program	5555H	AAH	2AAAH	55H	5555H	A5H	WA <sup>6</sup>	Data				
User Security ID Program Lock-Out	5555H	AAH	2AAAH	55H	5555H	85H	XXH <sup>6</sup>	0000H				
Software ID Entry <sup>7,8</sup>	5555H	AAH	2AAAH	55H	5555H	90H						
CFI Query Entry	5555H	AAH	2AAAH	55H	5555H	98H						
Software ID Exit <sup>9,10</sup> /CFI Exit/Sec ID Exit	5555H	AAH	2AAAH	55H	5555H	F0H						
Software ID Exit <sup>9,10</sup> /CFI Exit/Sec ID Exit	XXH	F0H										

# SST39VF1601 word program





# Micron 2Gbit NAND flash organization

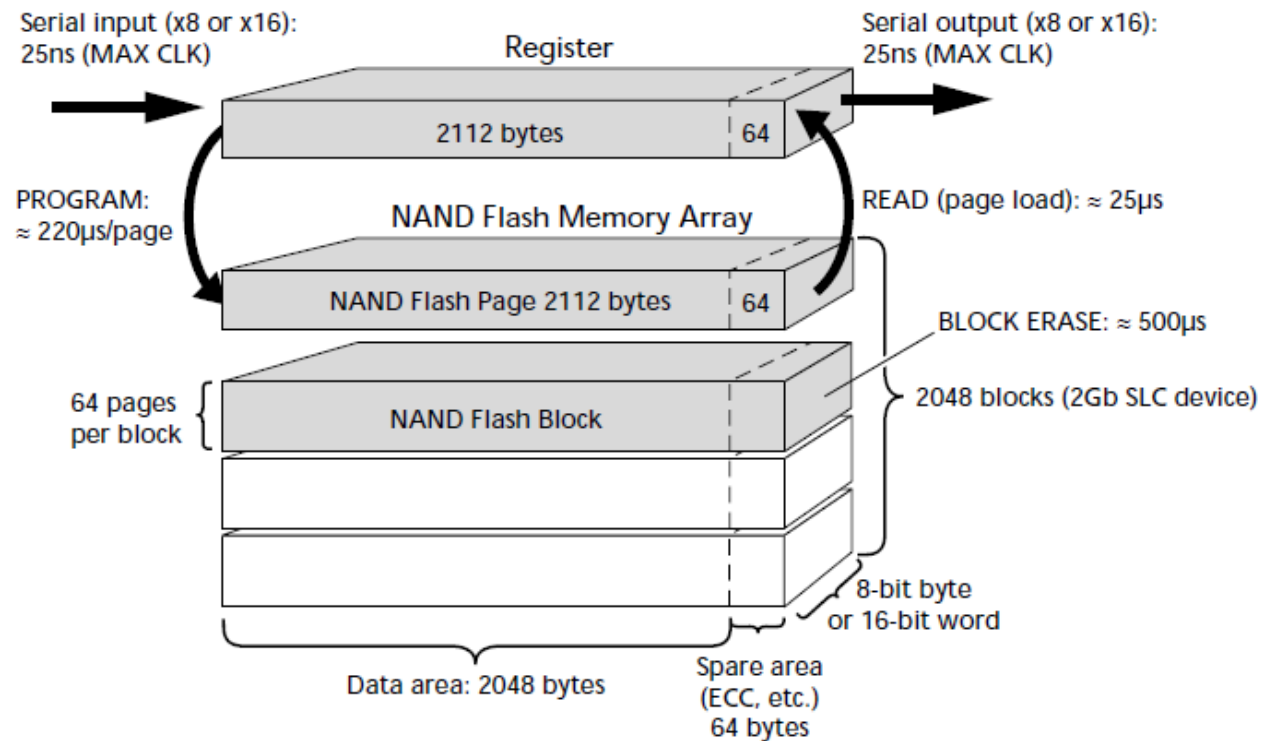
System transfers data to/from the “Register”  
Internal: page copied to Register

Register:  
Holds 1 page

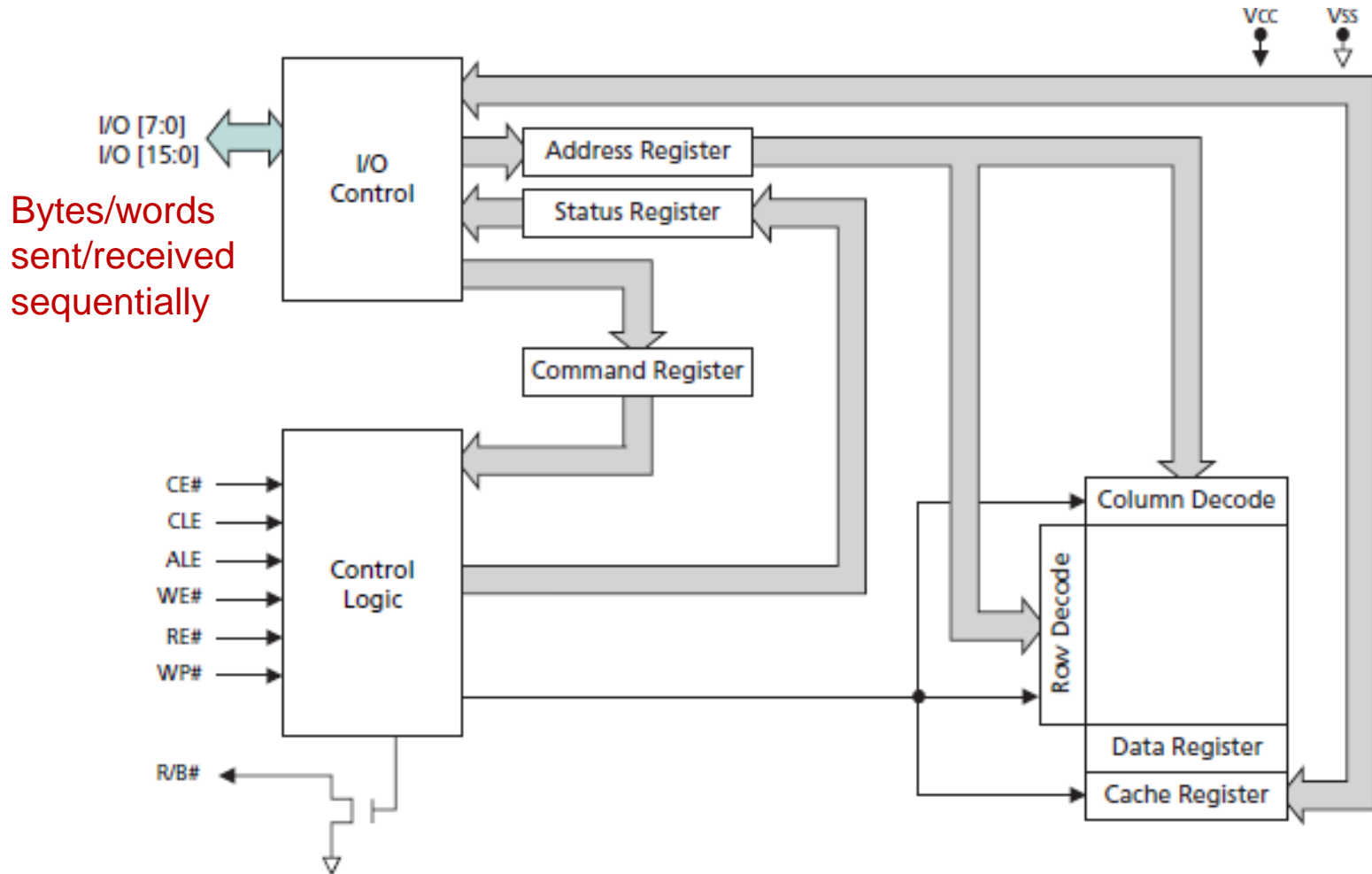
Page:  
2048 + 64 bytes

Block:  
64 pages

Chip:  
2048 blocks









# NAND flash functional block diagram



Micron: 2/4/8 Gbit, x8/x16 multiplexed NAND flash

# Micron Flash Mode Selection

CLE = command latch enable; ALE = address latch enable

CLE	ALE	CE#	WE#	RE#	WP# <sup>1</sup>	PRE <sup>2</sup>	Mode
H	L	L		H	X	X	Read mode Command input
L	H	L		H	X	X	
H	L	L		H	H	X	Write mode Command input
L	H	L		H	H	X	
L	L	L		H	H	X	Data input
L	L	L	H		X	X	Sequential read and data output
L	L	L	H	H	X	X	During read (busy)
X	X	X	X	X	H	X	During program (busy)
X	X	X	X	X	H	X	During erase (busy)
X	X	X	X	X	L	X	Write protect
X	X	H	X	X	0V/Vcc	0V/Vcc	Standby

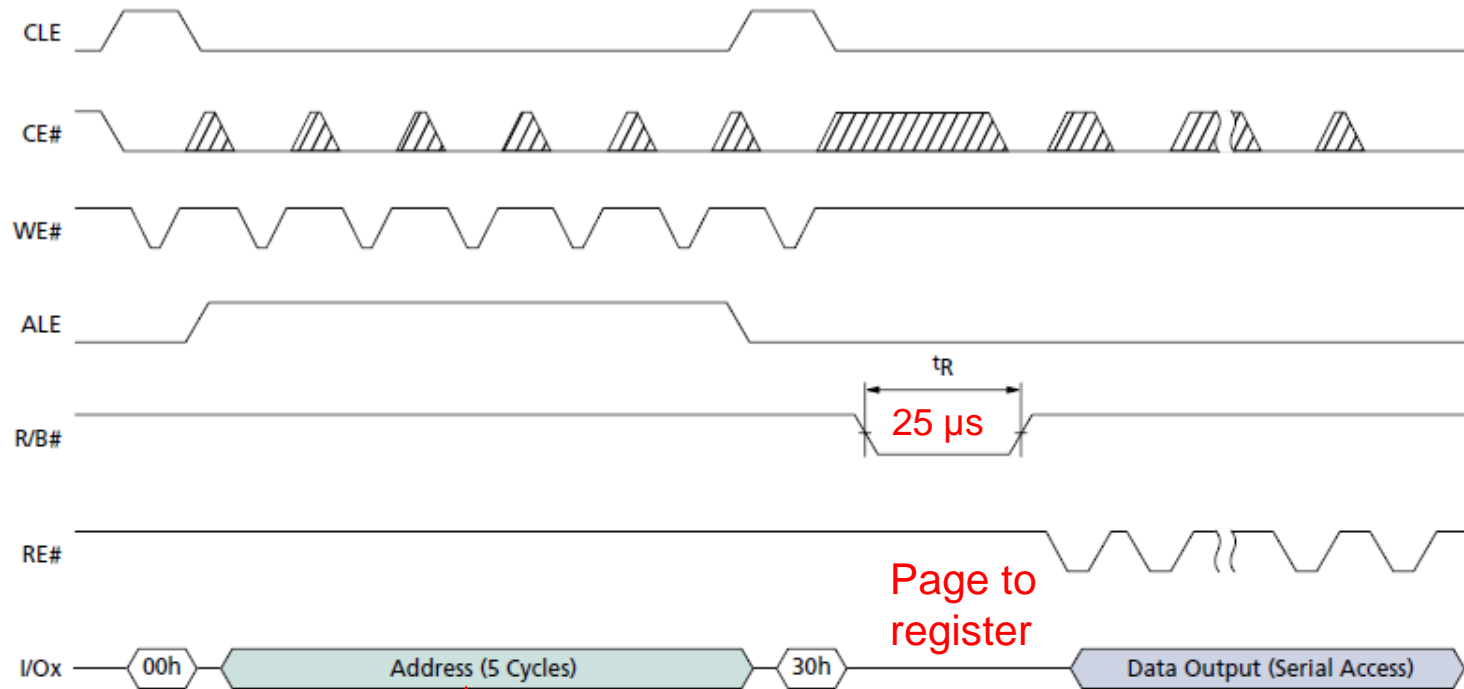
- Notes:
1. WP# should be biased to CMOS HIGH or LOW for standby.
  2. PRE should be tied to VCC or ground. Do not transition PRE during device operations. The PRE function is not supported on extended-temperature devices.
  3. Mode selection settings for this table: H = Logic level HIGH; L = Logic level LOW; X = VIH or VIL.

# Micron Flash Command Set

---

Operation	Cycle 1	Cycle 2	Valid During Busy
PAGE READ	00h	30h	No
PAGE READ CACHE MODE START <sup>1</sup>	31h	–	No
PAGE READ CACHE MODE START LAST <sup>1</sup>	3Fh	–	No
READ for INTERNAL DATA MOVE <sup>2</sup>	00h	35h	No
RANDOM DATA READ <sup>3</sup>	05h	E0h	No
READ ID	90h	–	No
READ STATUS	70h	–	Yes
PROGRAM PAGE	80h	10h	No
PROGRAM PAGE CACHE <sup>1</sup>	80h	15h	No
PROGRAM for INTERNAL DATA MOVE <sup>2</sup>	85h	10h	No
RANDOM DATA INPUT for PROGRAM <sup>4</sup>	85h	–	No
BLOCK ERASE	60h	D0h	No
RESET	FFh	–	Yes

# Micron NAND Flash Page Read Operation



Five address cycles

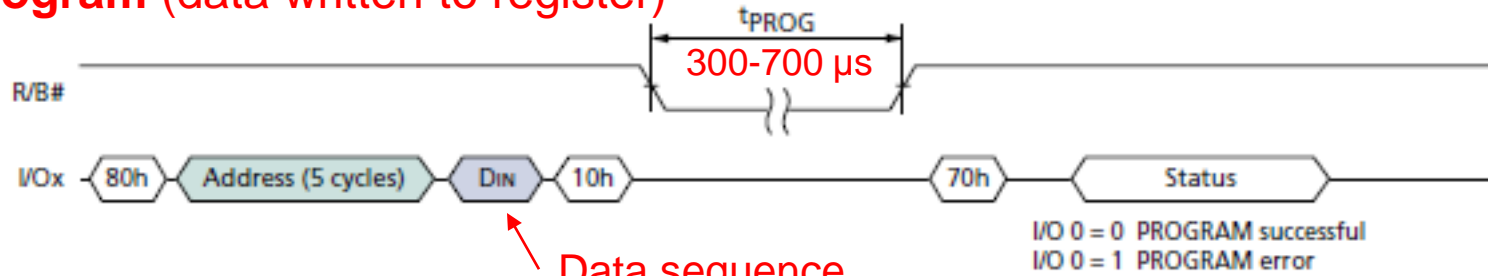
Cycle	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
First	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Second	LOW	LOW	LOW	LOW	CA11	CA10	CA9	CA8
Third	RA19	RA18	RA17	RA16	RA15	RA14	RA13	RA12
Fourth	RA27	RA26	RA25	RA24	RA23	RA22	RA21	RA20
Fifth	LOW	LOW	LOW	LOW	LOW	LOW	LOW	RA28

Note: CAx = column address; RAx = row address.

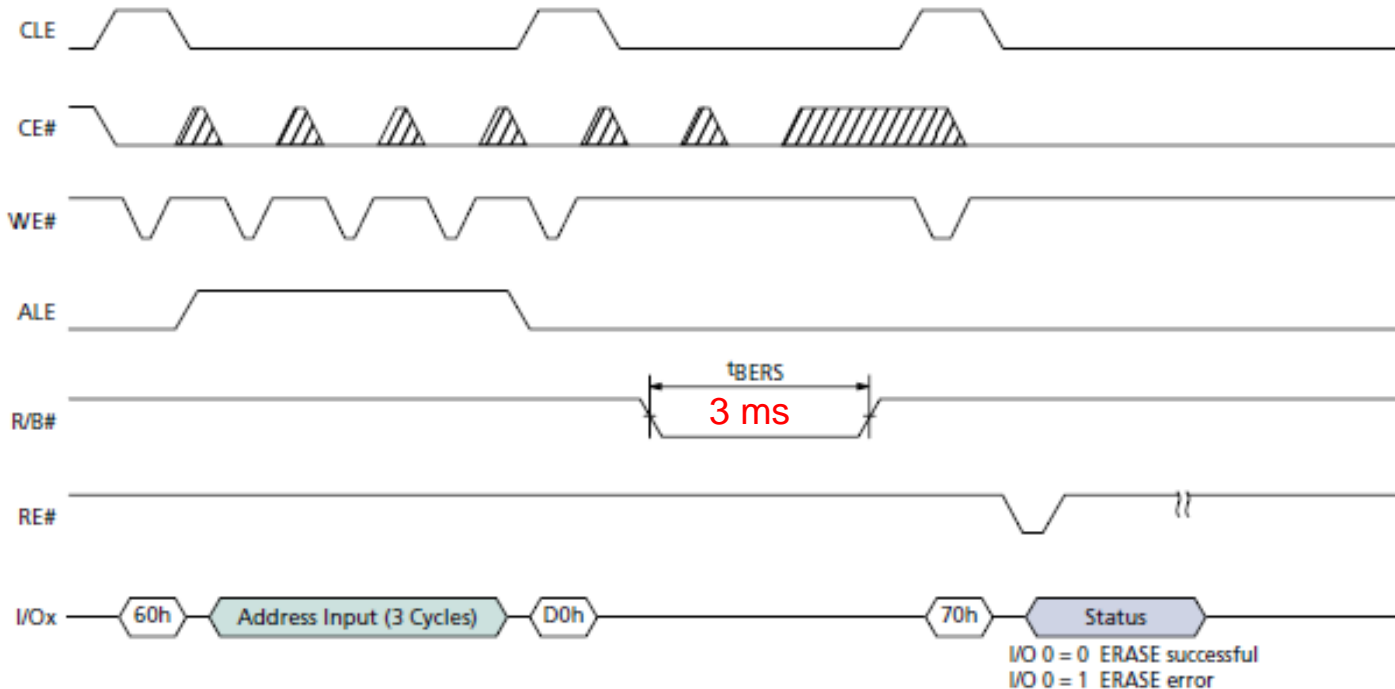
Capacity-dependent

# Micron NAND Flash: Program & Erase Op's

## Program (data written to register)

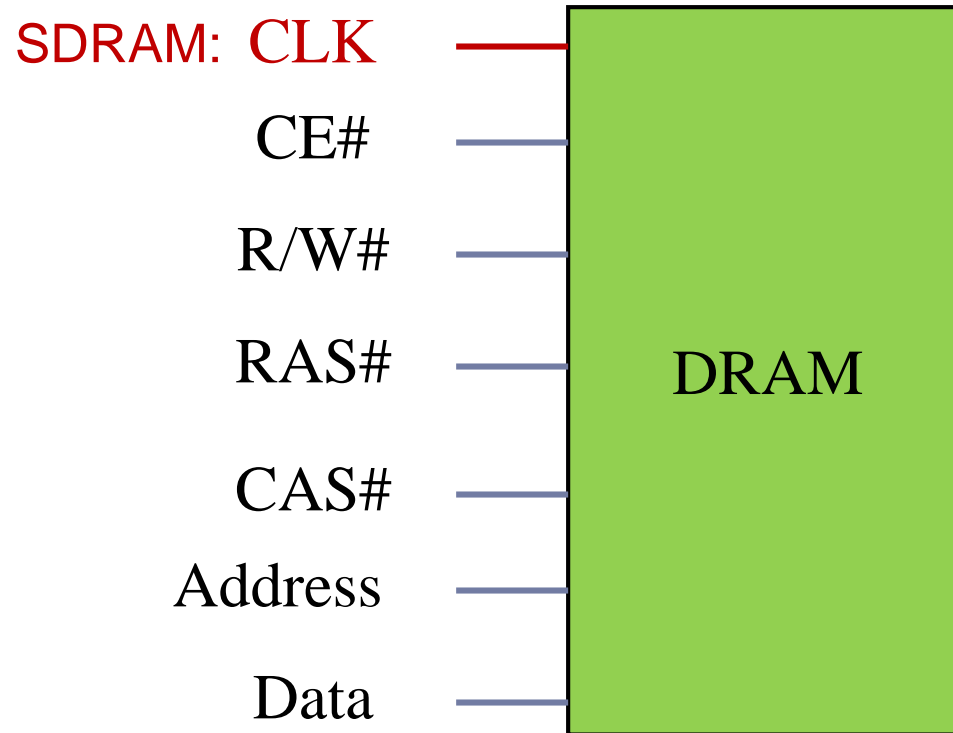


## Erase selected block



# Generic DRAM device

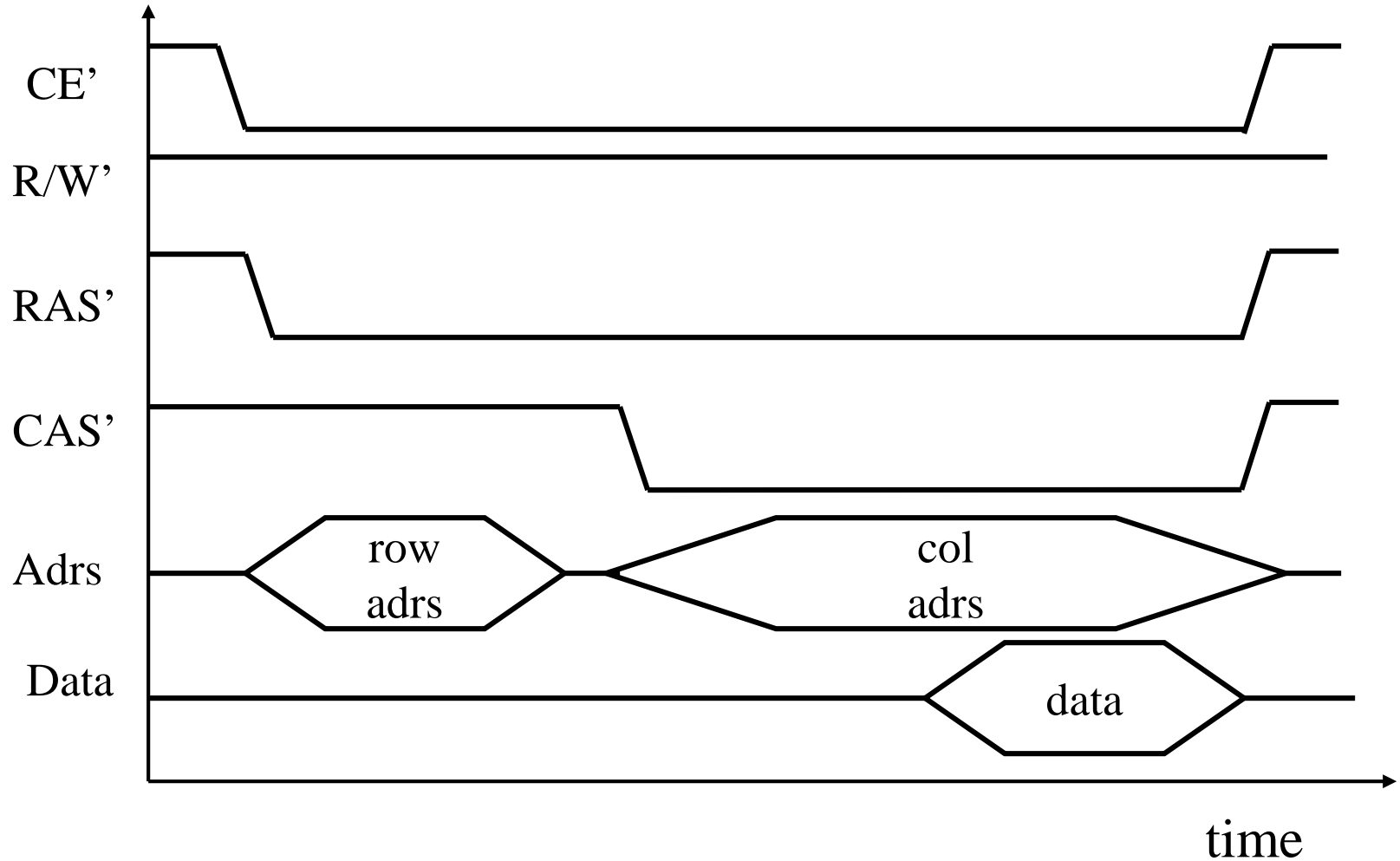
---



RAS# = **Row Address Strobe**: row# on Address inputs

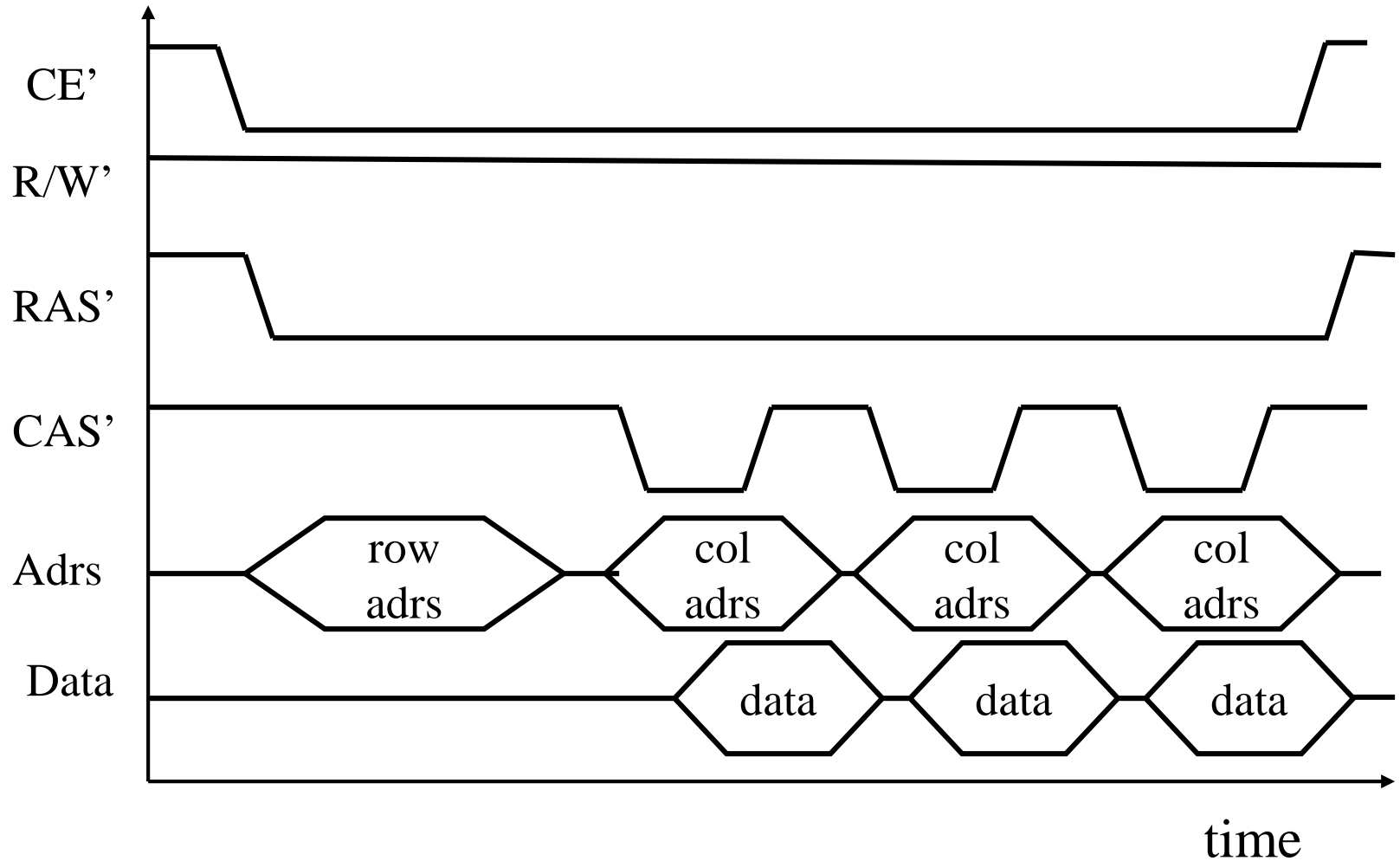
CAS# = **Column Address Strobe**: column# on Address inputs

# Asynchronous DRAM timing

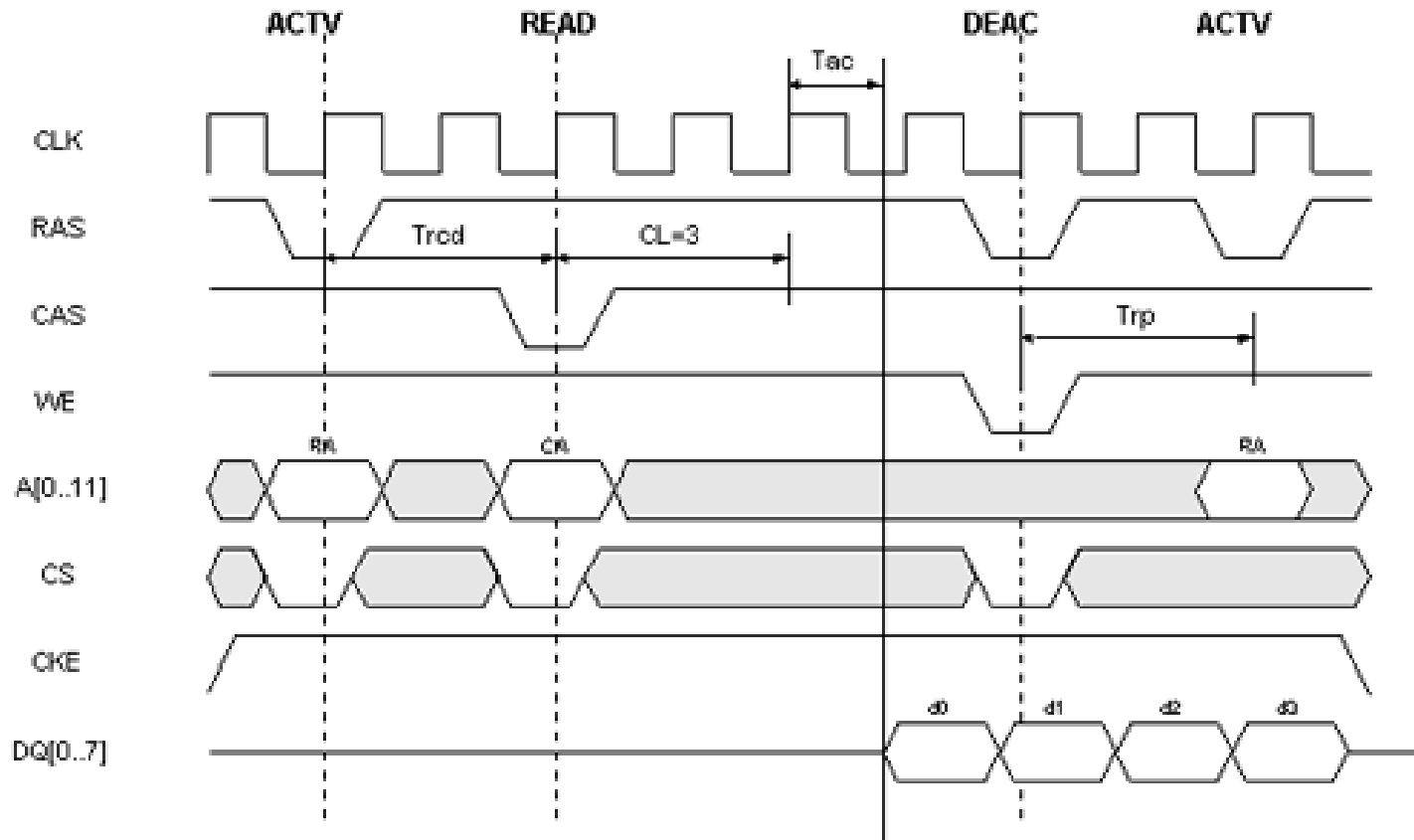




# Asynchronous DRAM page mode access



# SDRAM burst read (burst length 4)



*Trcd = RAS-to-CAS delay*  
*CL = CAS latency (CAS to data ready)*  
*Tac = access time*

# Dynamic RAM refresh

---

- ▶ Value decays in approx. 1 ms.
- ▶ Refresh value by reading it.
  - ▶ Can't access memory during refresh.
- ▶ RAS-only refresh
- ▶ CAS-before-RAS refresh.
- ▶ Hidden refresh.

## Example: 4 Mbyte DRAM

Refreshed every 4 msec (one row at a time)

Organized as 2048 rows x 2048 columns → 2048 refreshes

Assume 1 refresh → 80 nsec

$$\frac{2048 \times 80 \times 10^{-9}}{4 \times 10^{-3}} \cong 0.041 \rightarrow 4.1\% \text{ of time spent refreshing}$$

# Other DRAM forms

---

- ▶ Extended data out (EDO): improved page mode access.
- ▶ Synchronous DRAM: clocked access for pipelining.
  - ▶ All operations clocked
    - ▶ Row address
    - ▶ Column address - increments on clock for each data transfer
    - ▶ Data transfer – burst transfers (one per clock) after initial latency
- ▶ Double Data Rate (DDR) – transfer on both edges of clock
  - ▶ Effectively doubles the bandwidth
  - ▶ DDR-2: doubles the clock rate of DDR
  - ▶ DDR-3, DDR-4 support increasingly higher bandwidths
- ▶ Rambus: highly pipelined DRAM.

# DDR2 bank activate

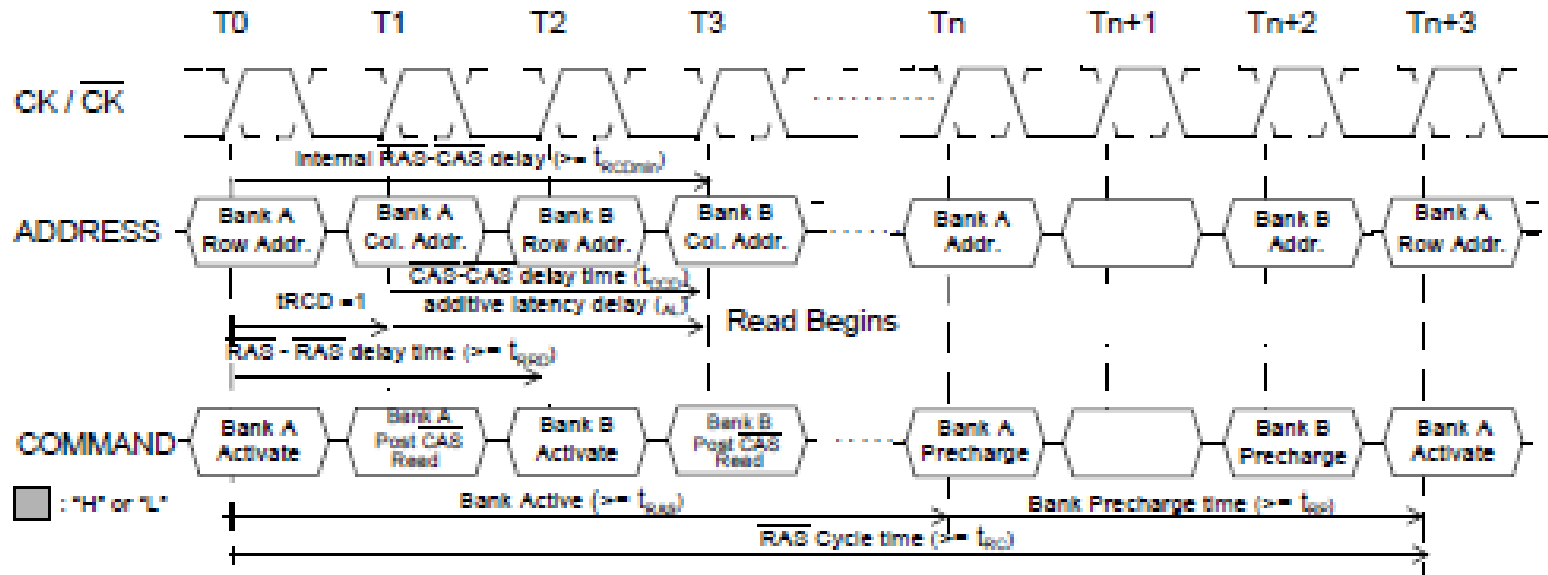


Figure 15. Bank active command cycle:  $t_{RCD} = 3$ ,  $AL = 2$ ,  $t_{RP} = 3$ ,  $t_{RRD} = 2$ ,  $t_{CCD} = 2$

Memory partitioned into 8 separate arrays called "banks"

Bank Activate command =  $CS\#$  low,  $RAS\#$  low,  $CAS\#$  high,  $WE\#$  high (and  $CKE$  high)

- Bank address BA2-BA0 selects bank

- Row address A15-A0 selects a row in the bank

Follow with read/write command in next clock cycle

Concurrent Bank Activate commands permitted (up to 8)

# DDR2 burst read (burst length 4)

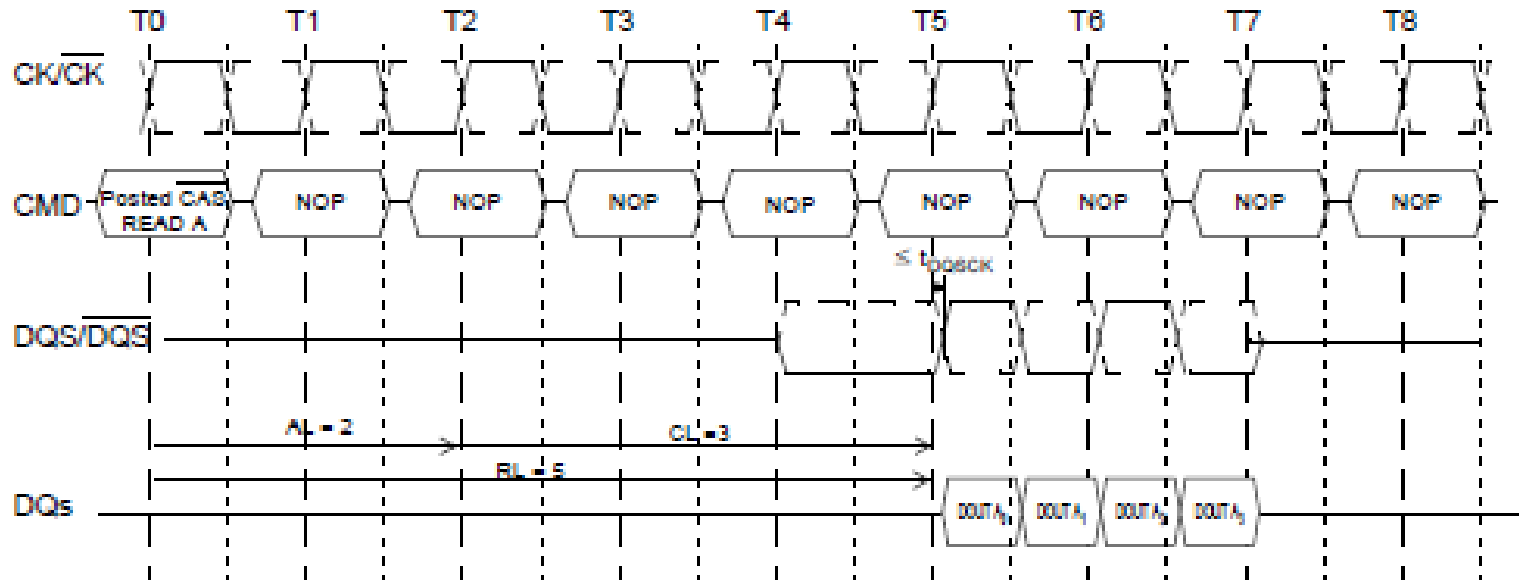


Figure 19. Burst read operation: RL = 5 (AL=2, CL=3, BL=4)

*Burst read command = CS# low, CAS# low, RAS# high, WE# high (and CKE high)*

*Read Latency  $RL = AL + CL$*

*CL (programmable) = CAS latency (CAS to data ready)*

*AL (programmable) = "Additive" Latency*

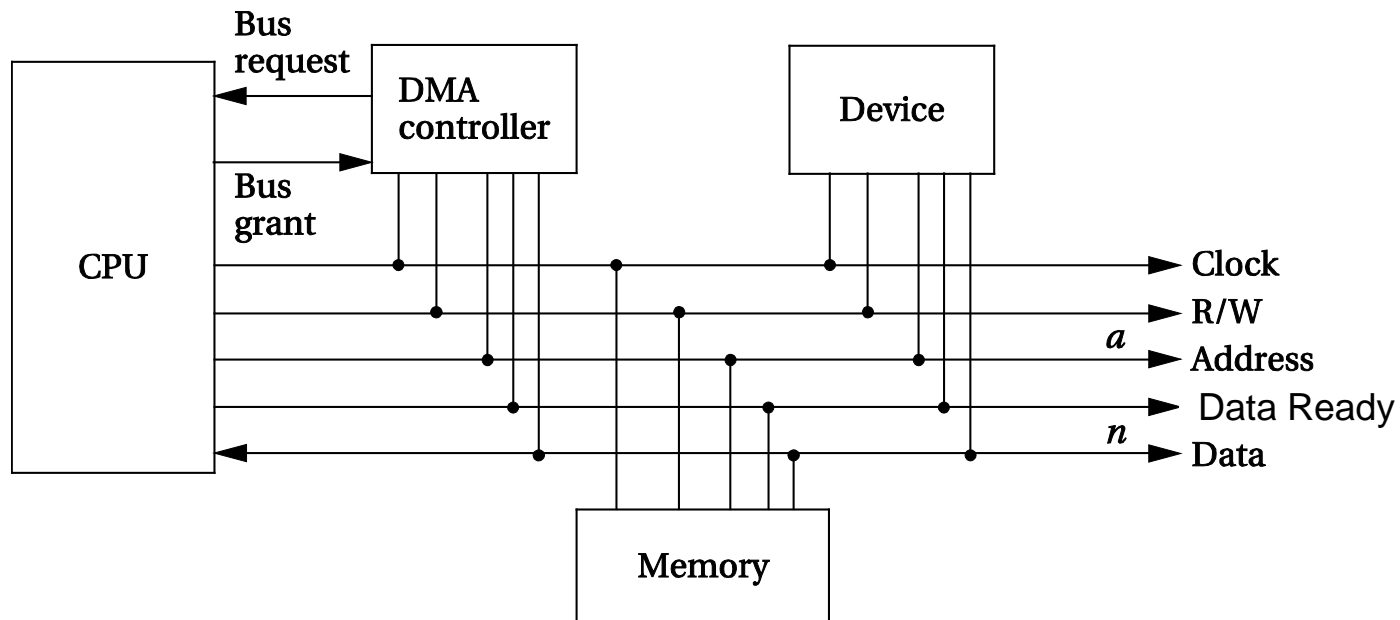
# Systems with multiple bus masters

---

- ▶ **Bus master** controls operations on the bus.
- ▶ CPU is default bus master.
- ▶ Other devices may request bus mastership.
  - ▶ Request mastership via separate handshaking lines.
  - ▶ Main CPU can't use bus when it is not master.
- ▶ Situations for multiple bus masters:
  - ▶ **DMA** data transfers
  - ▶ Multiple CPUs/Cores with shared memory
  - ▶ Separate graphics/network processor

# Direct Memory Access (DMA)

- ▶ DMA data transfers done without executing CPU instructions.
  - ▶ CPU sets up transfer.
  - ▶ DMA engine fetches, writes.
- ▶ DMA controller is a separate unit.

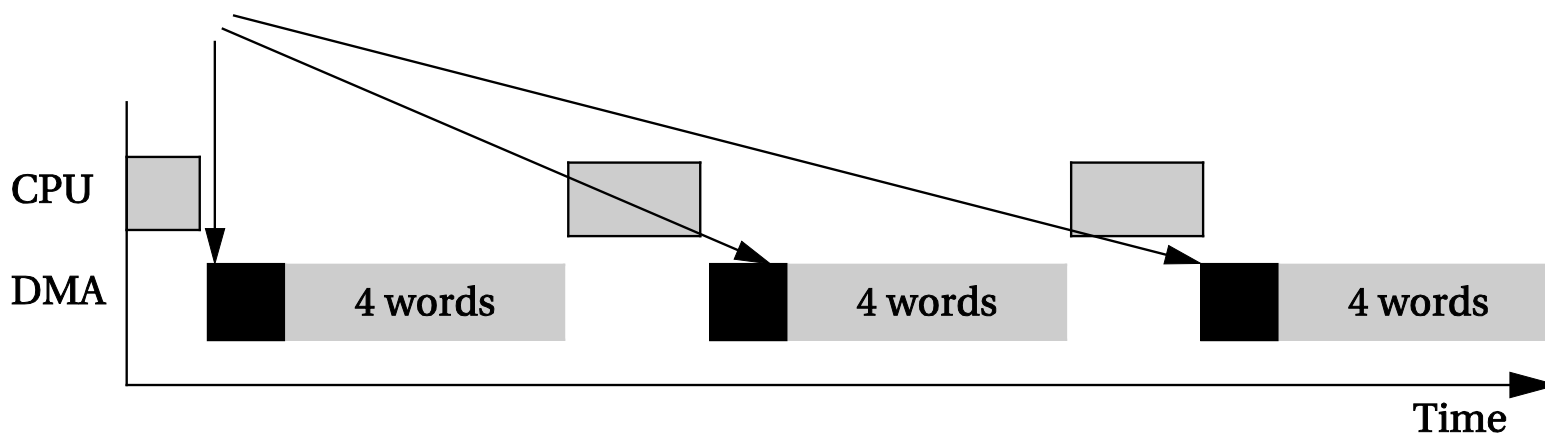




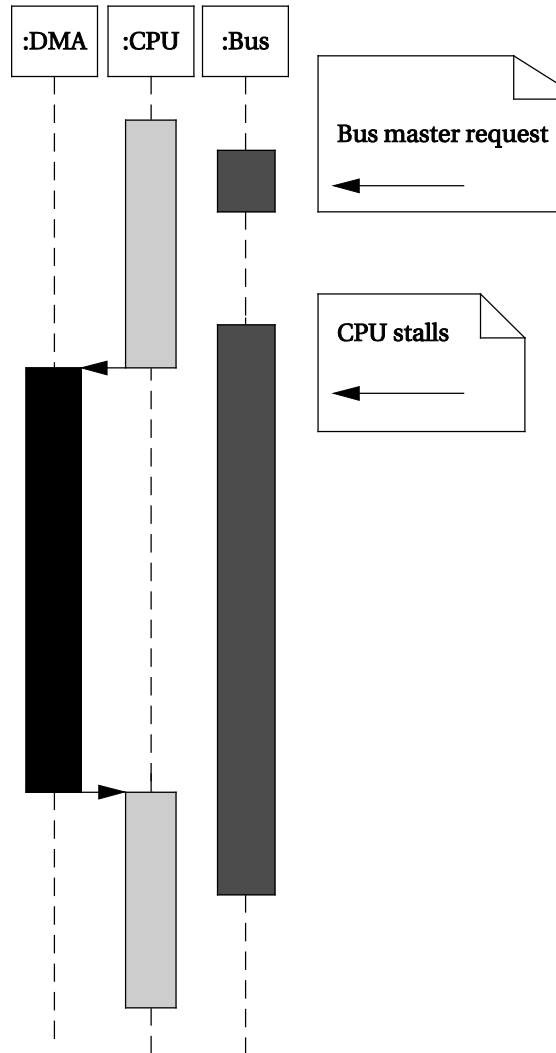
# DMA operation

- ▶ CPU sets DMA registers for start address, length.
- ▶ DMA status register controls the unit.
  - ▶ Bus request to CPU – Bus grant back from CPU
- ▶ DMA controller requests bus mastership from CPU
- ▶ Once DMA is bus master, it transfers automatically.
  - ▶ May run continuously until complete.
  - ▶ May use every  $n^{\text{th}}$  bus cycle.

Bus master request



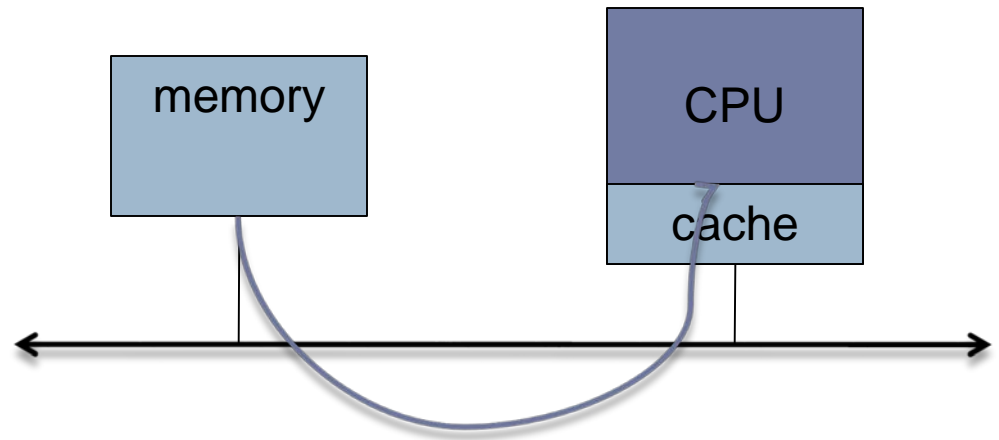
# Bus transfer sequence diagram



# System-level performance analysis

---

- ▶ Performance depends on all the elements of the system:
  - ▶ CPU.
  - ▶ Cache.
  - ▶ Bus.
  - ▶ Main memory.
  - ▶ I/O device.



# Bandwidth as performance

---

- ▶ **Bandwidth applies to several components:**
  - ▶ Memory.
  - ▶ Bus.
  - ▶ CPU fetches.
- ▶ Different parts of the system run at different clock rates.
- ▶ Components may have different widths (bus, memory).

# Bandwidth and data transfers

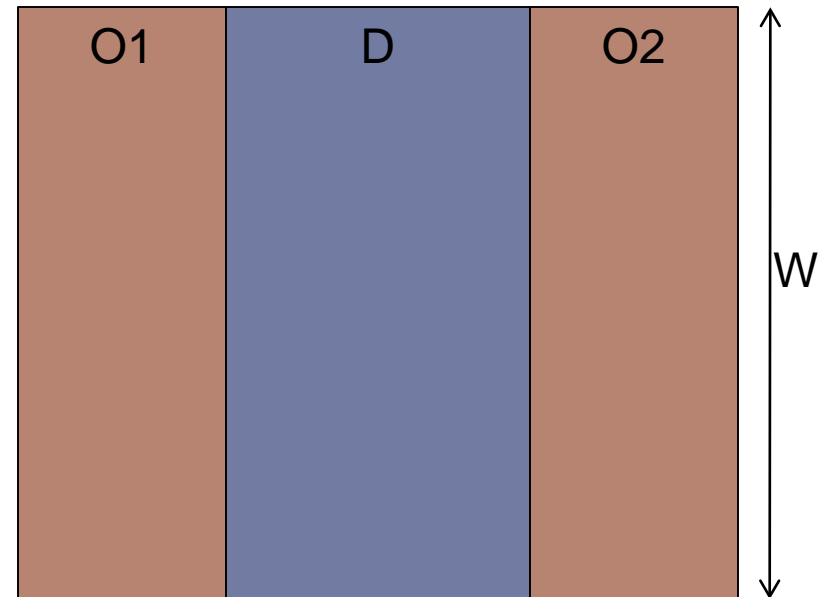
---

- ▶ Video frame:  $320 \times 240 \times 3 = 230,400$  bytes.
  - ▶ Need to transfer in  $1/30$  sec = 0.033 sec
- ▶ Transfer 1 byte/ $\mu$ sec, 0.23 sec per frame.
  - ▶ **Too slow.**
- ▶ To increase bandwidth:
  - ▶ Increase bus width.
  - ▶ Increase bus clock rate.
  - ▶ Minimize overhead (do burst transfers)

# Bus bandwidth

---

- ▶  $T$ : # bus cycles.
- ▶  $P$ : bus clock period.
- ▶ Total time for transfer:
  - ▶  $t = TP$ .
- ▶  $D$ : data payload length.
- ▶  $O = O1 + O2 =$  overhead.  
(before & after data)
- ▶  $N =$  total # data payloads.
- ▶  $W =$  bus width (bits/xfer)

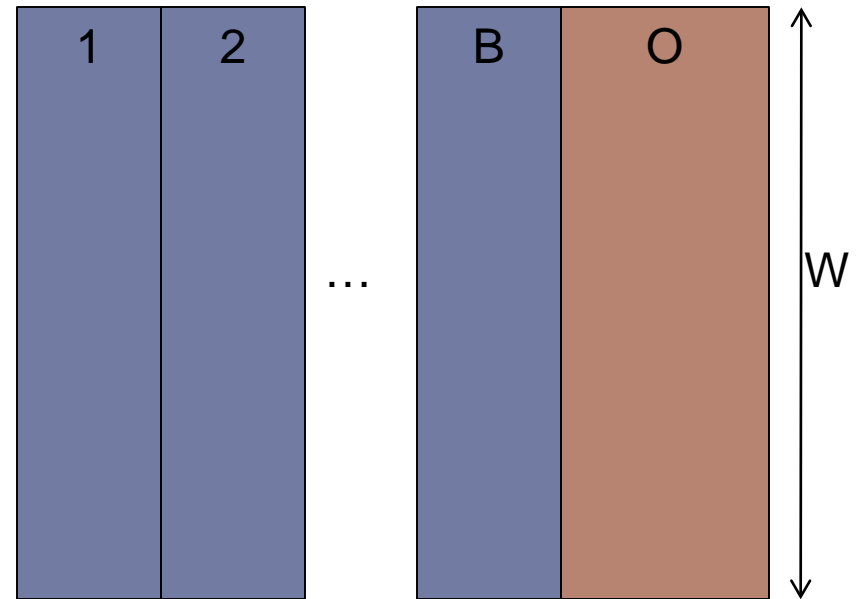


$$T_{\text{basic}}(N) = (D+O)N/W$$

Transfer  $ND$  bits

# Bus burst transfer bandwidth

- ▶ T: # bus cycles.
- ▶ P: time/bus cycle.
- ▶ Total time for transfer:
  - ▶  $t = TP$ .
- ▶ D: data payload length.
- ▶ B: burst size  
(#transfers of size D)
- ▶  $O1 + O2 = \text{overhead } O$ .
- ▶ N = total # data payloads

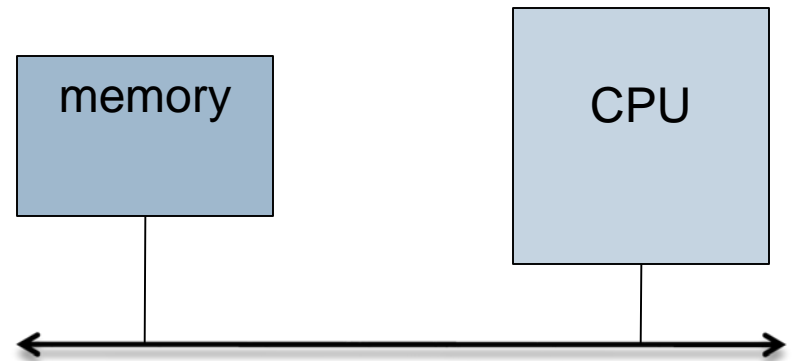


$$T_{\text{burst}}(N) = (BD+O)*N/(BW)$$

# Bus performance bottlenecks

---

- ▶ Transfer 320 x 240 video frame @ 30 frames/sec = 612,000 bytes/sec.
- ▶ Is performance bottleneck bus or memory?



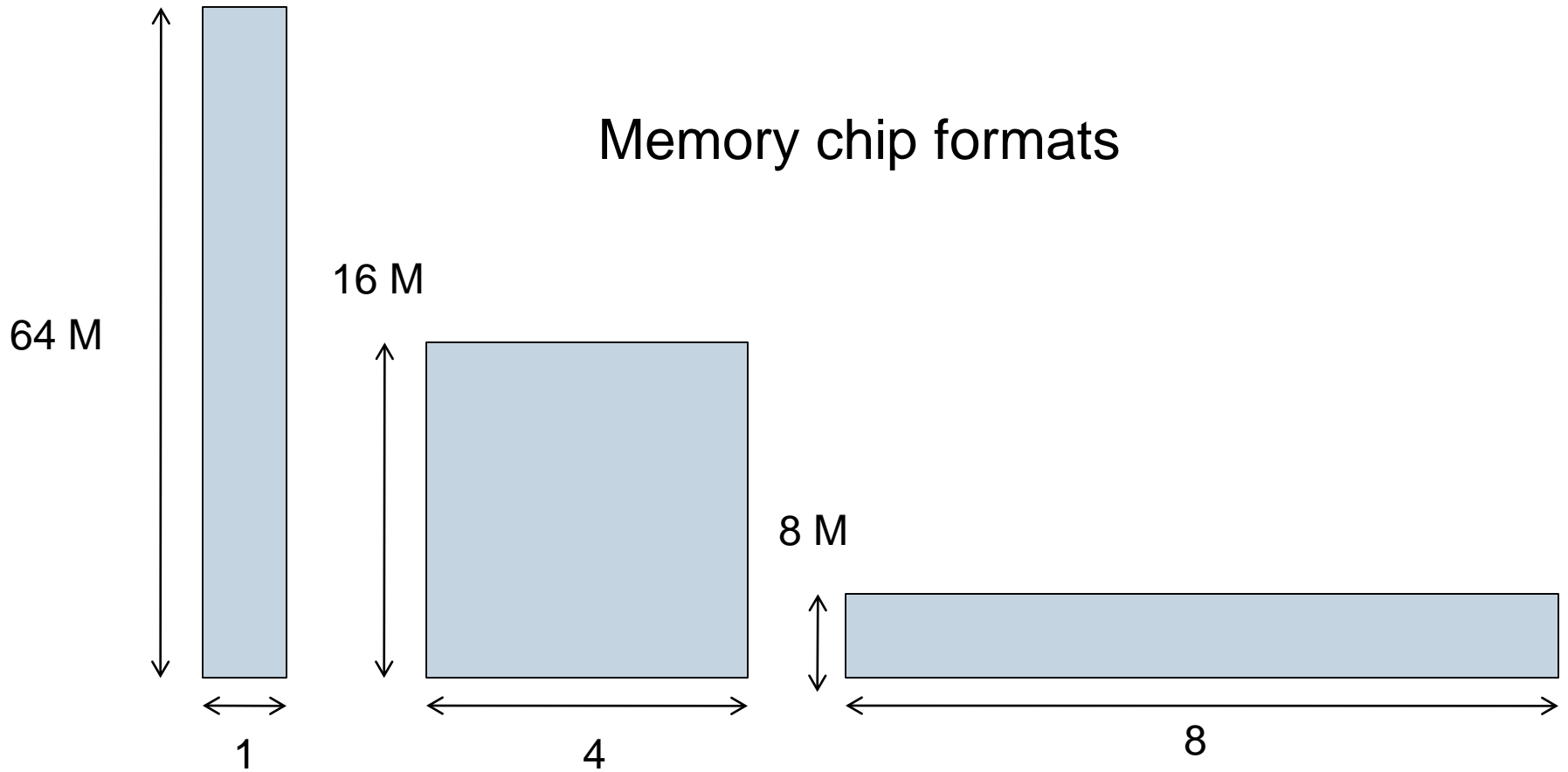
- 
- ▶ **Bus:** assume 1 MHz bus,  $D=1$ ,  $O=3$ :
    - ▶  $T_{\text{basic}} = (1+3)612,000/2 = 1,224,000$  cycles = 1.224 sec.
  - ▶ **Memory:** try burst mode  $B=4$ , width  $w=0.5$ .
    - ▶  $T_{\text{mem}} = (4*1+4)612,000/(4*0.5) = 2,448,000$  cycles = 0.2448 sec.



# Memory aspect ratios

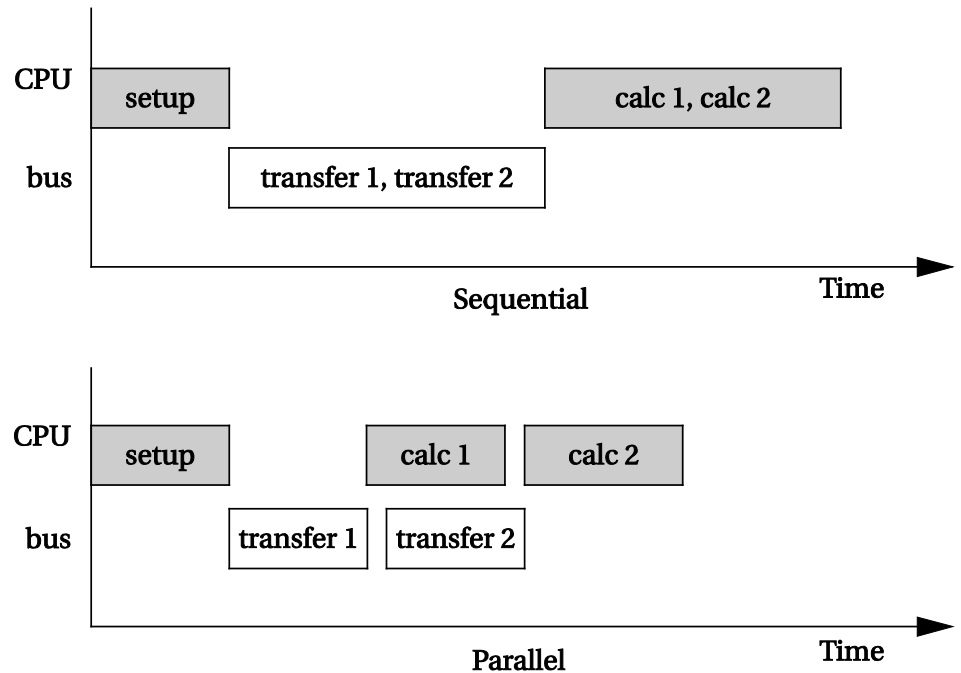
---

## Memory chip formats



# Parallelism

- ▶ Speed things up by running several units at once.
- ▶ DMA provides parallelism if CPU doesn't need the bus:
  - ▶ DMA + bus.
  - ▶ CPU.

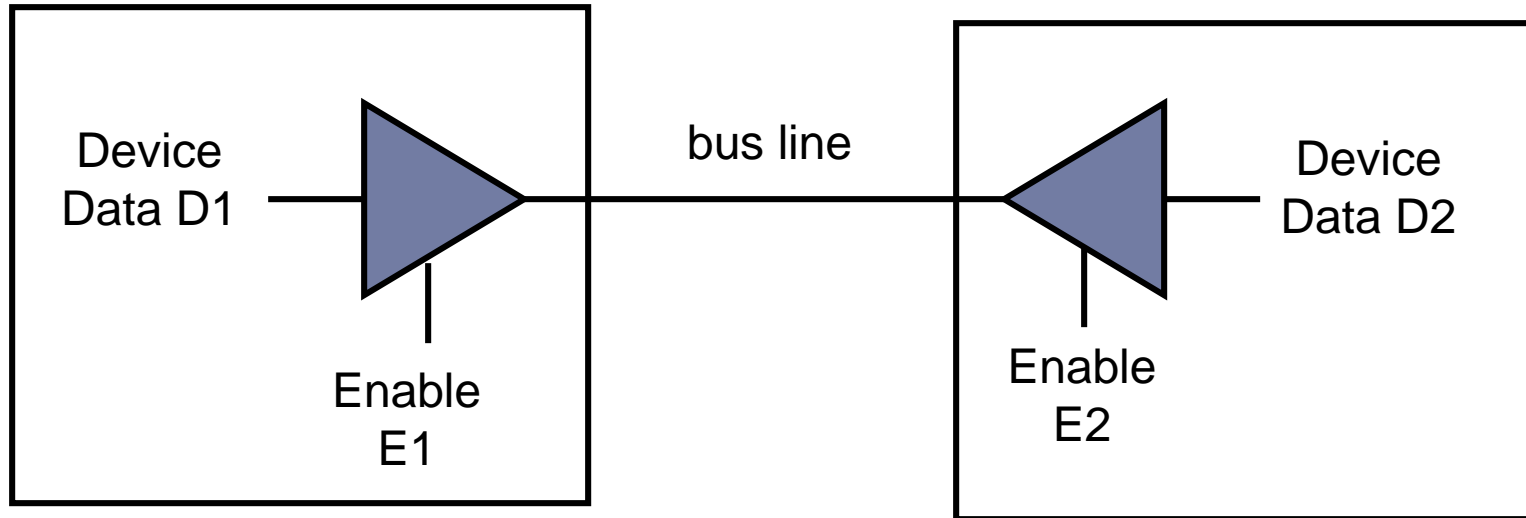


# Electrical bus design

---

- ▶ **Bus signals are usually tri-stated.**
- ▶ **Address and data lines may be multiplexed.**
- ▶ **Every device on the bus must be able to drive the maximum bus load:**
  - ▶ Bus wires.
  - ▶ Other bus devices.
  - ▶ Resistive and capacitive loads.
  - ▶ Bus specification may limit loads
- ▶ **Bus may include clock signal.**
  - ▶ Timing is relative to clock.

# Tristate operation



	E2=0	E2=1
E1=0	float	D2
E1=1	D1	conflict

← Must prevent E1=E2=1