Managment and Analysis of Physical Data – part I
a.a. 2019-20

Memories and Buses

G.Collazuol

Overview

- Registers and Memories
- Buses and Interconnections

Part I
Buses & I/O interconnections

Registers

Registers

Flip-flops are used in a variety of application circuits:

- counting circuits
- frequency division
- data storage and transfer (data movement) circuits

Counters and Registers

- Belong to the category of medium scale (<100 gates) sequential logic circuits
- Similar architecture: both comprise a cascaded arrangement of more than one FFs
- Both constitute very important building blocks of sequential logic

Registers

A register, unlike a counter, has no specified sequence of states (except in certain very specialized applications)

Application

- **Counters** are mainly used in counting applications, where they either measure the time interval between two unknown time instants or measure the frequency of a given signal
- **Registers** are primarily used for the temporary storage of data present at the output of a digital circuit before they are fed to another digital circuit

Register and Shift Register

The **storage capacity** of a register is the total number of bits (1s and 0s) of digital data it can retain. Since each FF can store one bit of data, the storage capacity of the shift register equals the number of FFs used

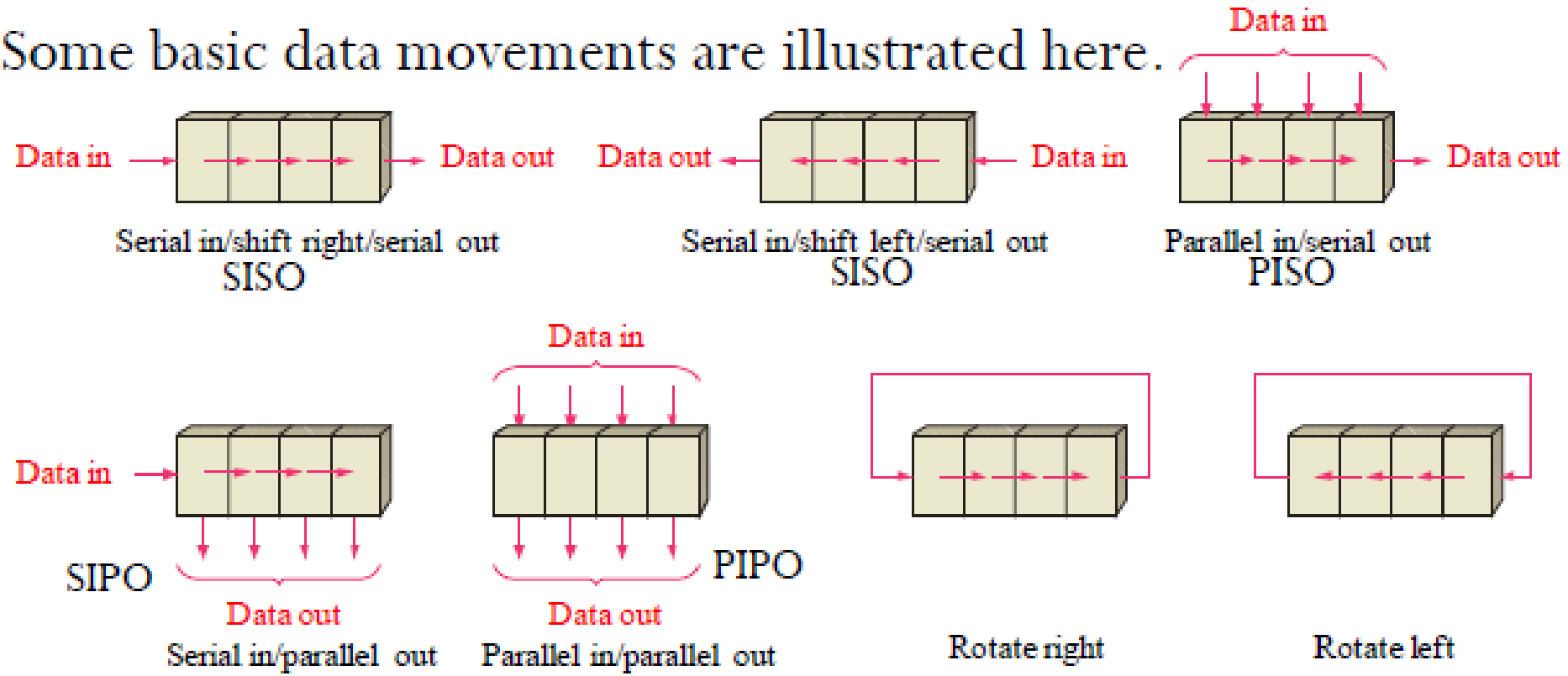
The **shift capability** of a register permits the movement of data from stage to stage within the register or into or out of the register upon application of clock pulses. Two very useful applications:

1) sequential memories (FIFO, …)

2) parallel ↔ serial data type transformation

Shift Register

- A shift register is an arrangement of flip-flops with important applications in storage and movement of data.
- Some basic data movements are illustrated here.

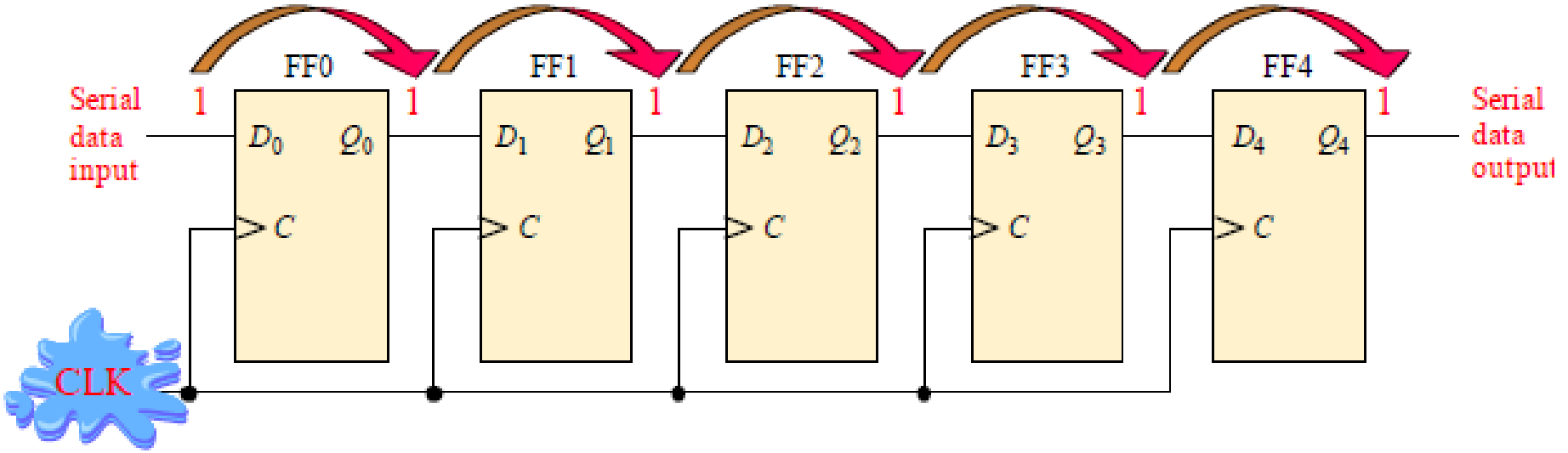


- The basic building block in all shift registers is the FF, mainly a D-FF.

Although in many of the commercial shift register ICs their internal circuit diagram might indicate the use of S-R FFs, a careful examination will reveal that these S-R FFs have been wired as D FFs only.

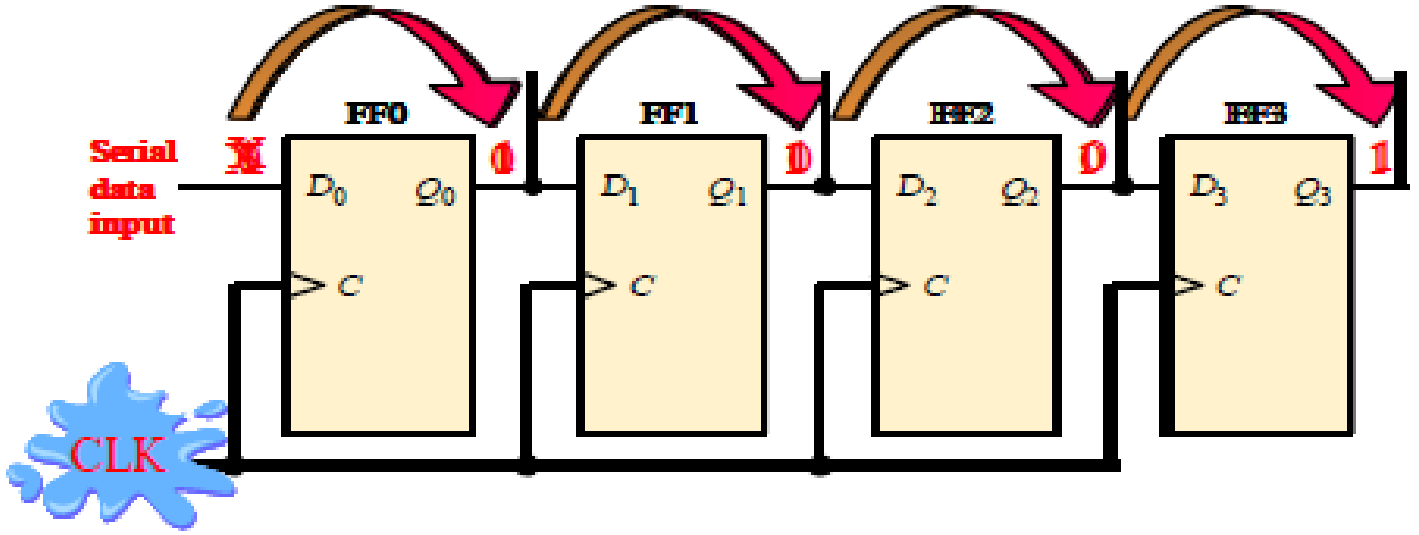
SISO Shift Register

- Accept data serially: one bit at a time on a single line.
- Each clock pulse will move an input bit to the next FF. For example, a 1 is shown as it moves across.
- Five-bit serial-in serial-out register.



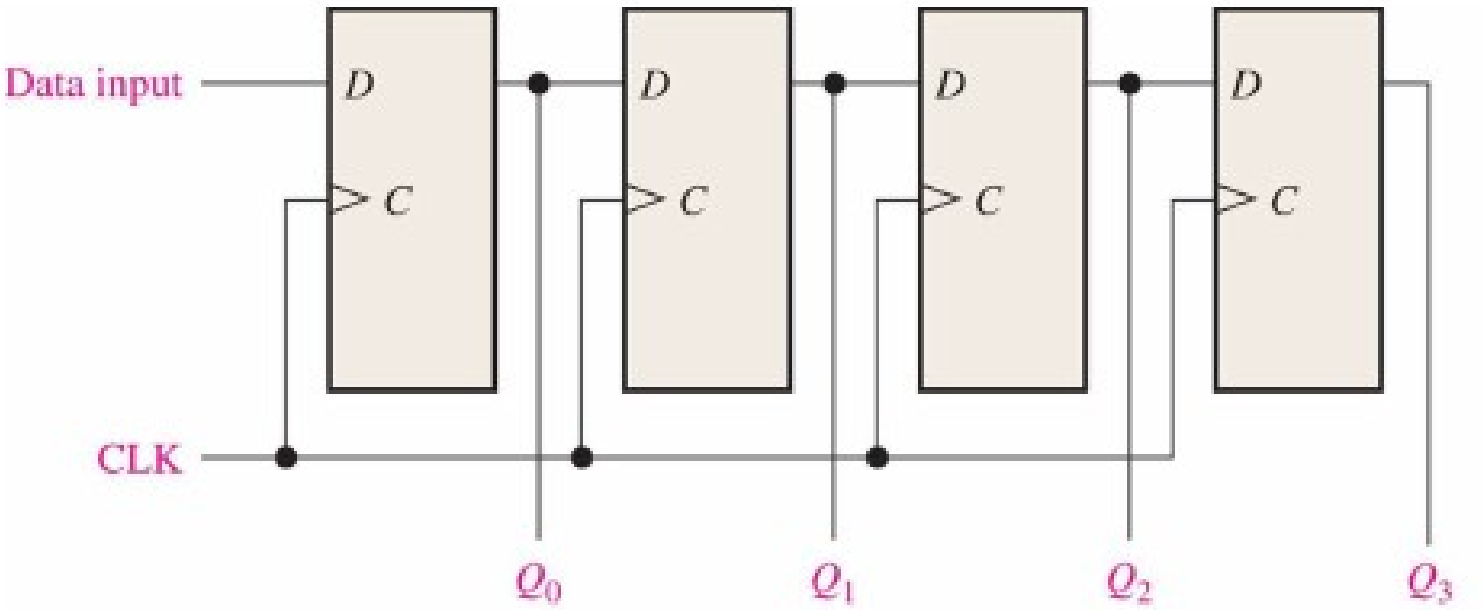
SIPO application

- An application of shift registers is conversion of serial data to parallel form.
- For example, assume the binary number 1101 is loaded sequentially, one bit at each clock pulse.

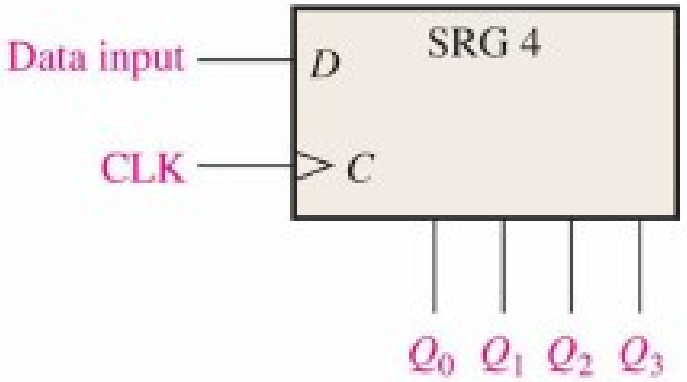


- After 4 clock pulses, the data is available at the parallel output.
  - They can be stored for any length of time as long as the FFs have dc power.

SIPO shift register

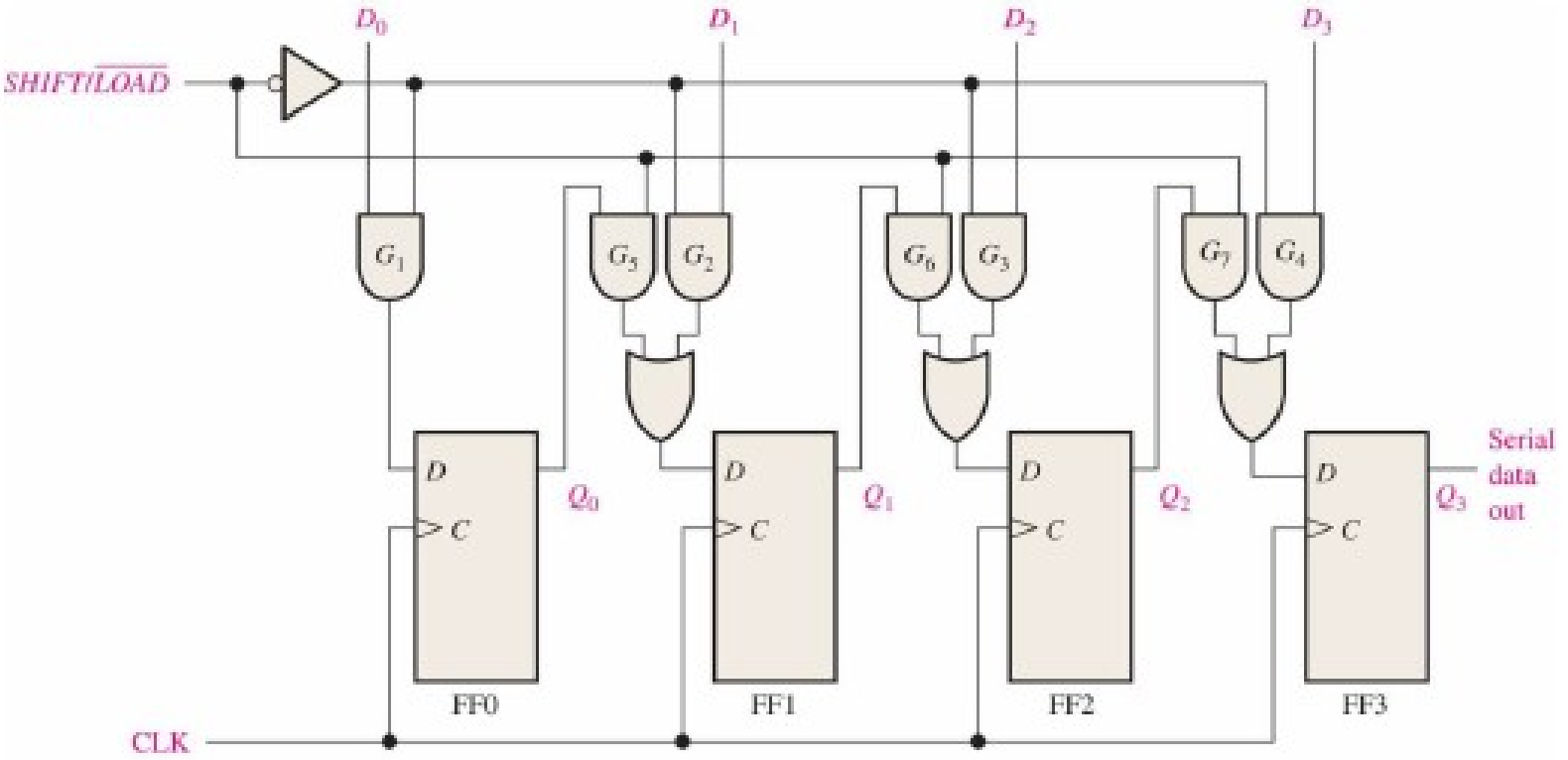


(a)

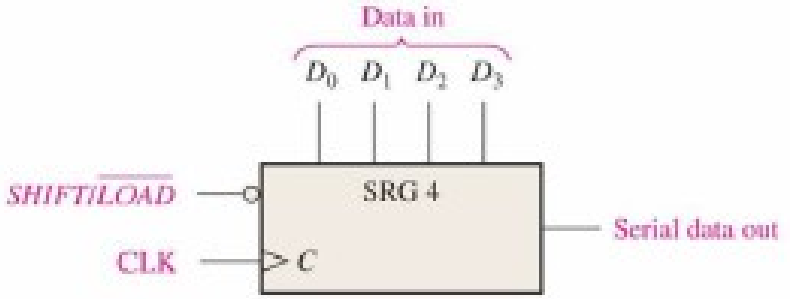


(b)

PISO shift register

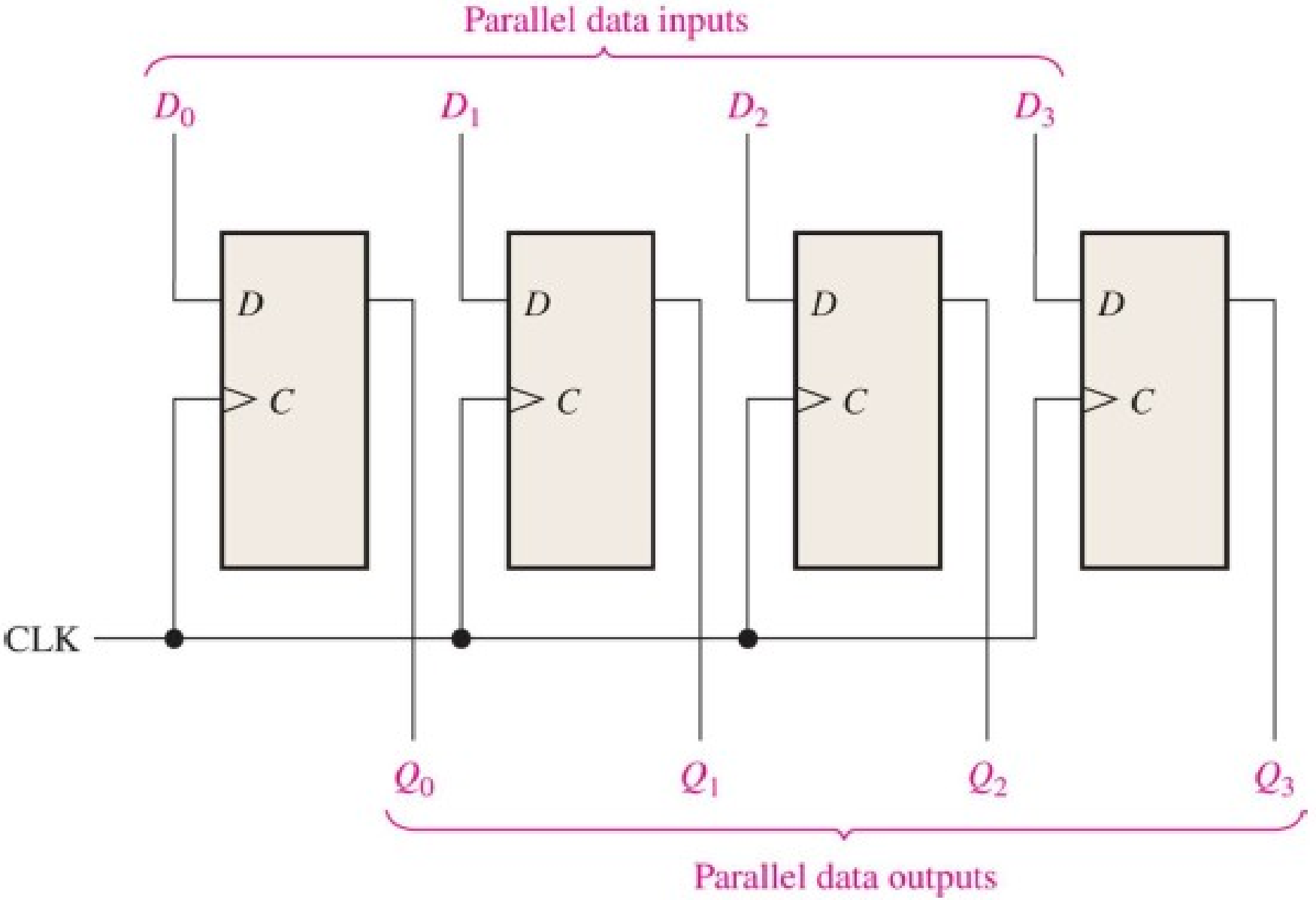


(a) Logic diagram



(b) Logic symbol

PIPO shift register



An SR derivate → FIFO

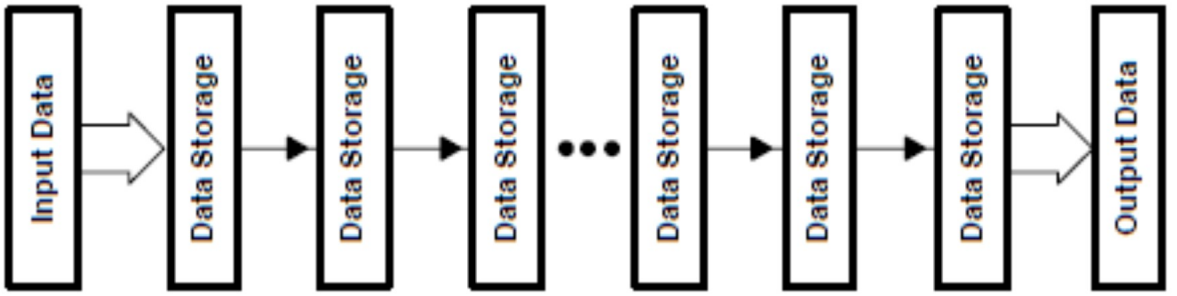
Every memory in which the data word that is written in first also comes out first when the memory is read is a first-in first-out memory (FIFO)

There are three kinds of FIFO:

- Shift register – FIFO with an invariable number of stored data words and, thus, the necessary synchronism betweenthe read and the write operations because a data word must be read every time one is written

- Exclusive read/write FIFO – FIFO with a variable number of stored data words and, because of the internal structure,the necessary synchronism between the read and the write operations

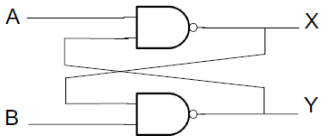
- Concurrent read/write FIFO – FIFO with a variable number of stored data words and possible asynchronismbetween the read and the write operation



Memories

Simplest Memory block

Memory is the portion of a digital system that stores binary data:



- The smallest unit of binary data, as you know, is the **bit**

- In many applications, data are handled in an 8-bit unit called a **byte**

- Each storage element in a memory can retain either a 1 or a 0 and is called a **cell**

Memory capacity and speed

- The term **word** can have two meanings in digital systems terminology:

  1) in memories, it is defined as a group of bits or bytes that acts as a single entity that can be stored in one memory location

  2) in assembly language, a word is specifically defined as two bytes

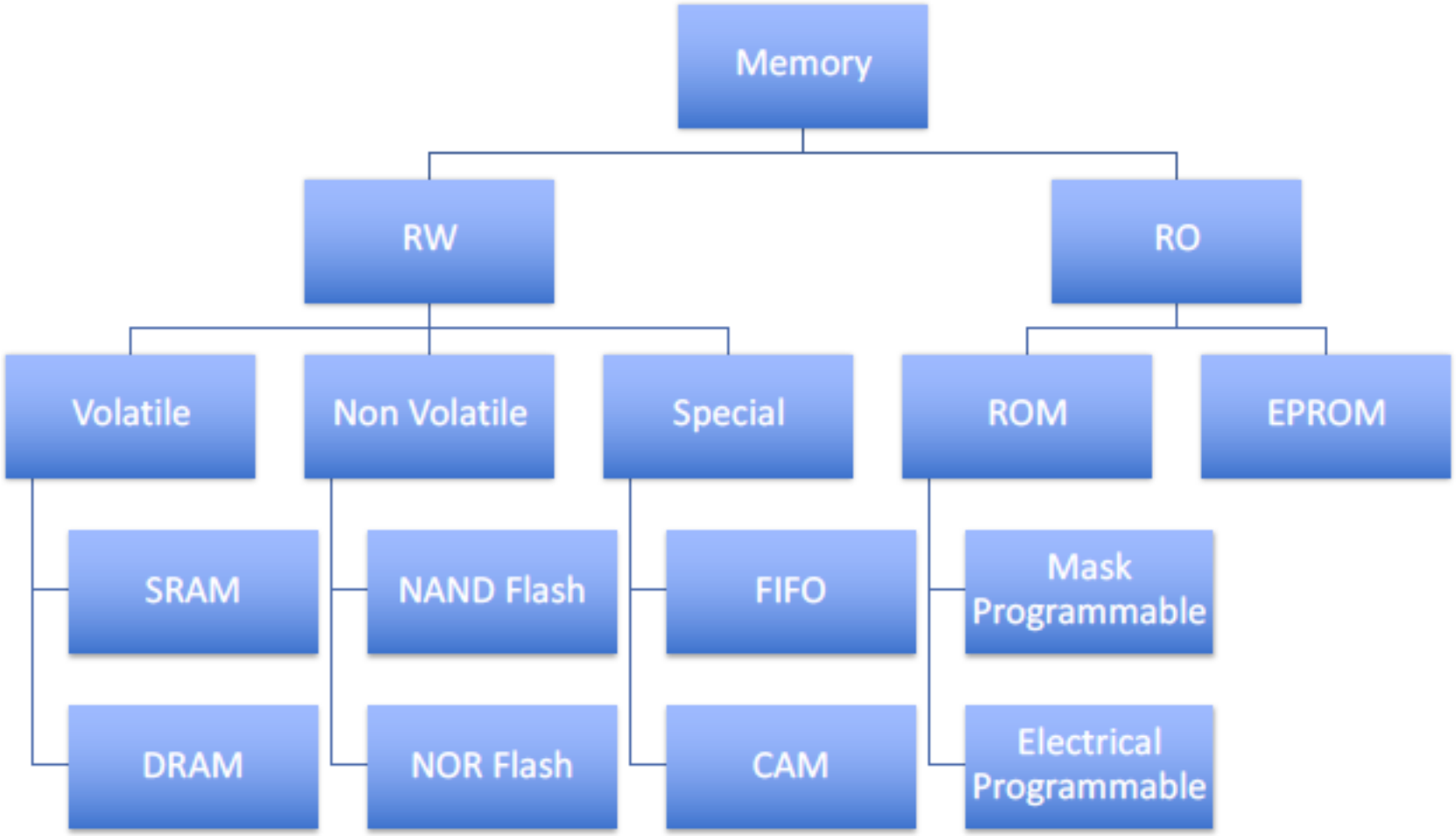
- A memory is **identified** by the number of words it can store times the word size

  → A 16k x 8 memory can store 16384 words of eight bits each

- The **capacity of a memory** is the total number of data units that can be stored

- **Data transfer speed** is the rate at which data can be transmitted in and out of the memory. This is often expressed in kbit per second or Mbit per second (Gbit, ...), abbreviated as kbps and Mbps respectively (Gbps…). In some cases the data transfer speed is expressed in kbytes or Mbytes per second, or kB/sec and MB/sec (GB/sec, …)

Memory zoology



Different storage mechanisms, max capacities and speed…

Example: Memory hierarchies in Computers

Level	Name	Speed	Size	Type
Level 1	Cache	1-2 Clock cycles (1-5 ns)	10KB-1MB	SRAM
Level 2	Main	10-50 Cycles (20-100 ns)	128 MB-100 GB	DRAM
Level 3	Disk	< 1ms	100GB-10TB	Magnetic- SSD
Level 4	Archive	< 1s	PB	Magnetic

Memory zoology: storage mechanism & type of access

Data storage mechanism: there are volatile and nonvolatile storage methods.

Volatile memories often store data as charge on a capacitor. The information can be destroyed when reading or deteriorated because of leakage, thus requiring a periodic refreshing

Non-volatile memories, also known as **ROM**, can be as follows:
- mask programmed, which are written during fabrication;
- programmable, written by burning out internal connections named fuses;
- erasable, where data is stored as charge on an isolated gate (the so-called floating gate). The cell can be erased using ultraviolet light. Electrically erasable, or flash, able to re-program the content by using high voltage

Type of access. There are two forms of access:

sequential access that features a group of elements, like data in a memory array or a disk file, which are accessed in a predefined, sequential order.

The second is random access denoting that information from any storage location is equally accessible in random order at a fixed rate, independent of physical location, for reading or writing. This kind of memory is denoted by the acronym **RAM**

Memory zoology - PROM

PROM (Programmable Read-Only Memory)
is programmable but data can be written only once. Therefore, the data is written in permanent form. This type of memories are sold in a blank format to be programmed with a special PROM programmer. Often the memory consists of an array of fuses “blown” to define the data pattern. This type of memory is often used to store fixed instructions and the set-up that must not change or be changed by the user

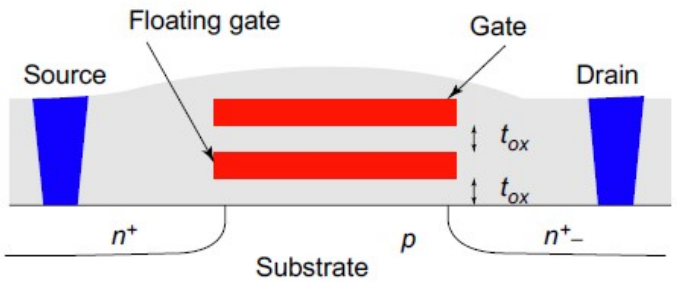
EPROM (Erasable Programmable Read-Only Memory). This is a memory that enables programmability but also erasability by exposure to an ultraviolet light for many minutes through a quartz window on the memory package. Each memory cell is made of a special type of transistor that has an isolated electrode, named floating gate, such that it can store information as a trapped charge.
- For erasing, ultraviolet exposure is exploied which favors release of a trapped charge. The operation totally erases the memory, which must be fully rewritten even to change a single bit.
- For programming, high voltages applied to the input of the cell cause large pulse currents and strong attraction of electrons toward the floating gate. Because of the very low oxide thickness the electrons penetrate and reach the floating gate. For use, the quartz window must be covered to preserve data for extended periods.

Memory zoology - PROM

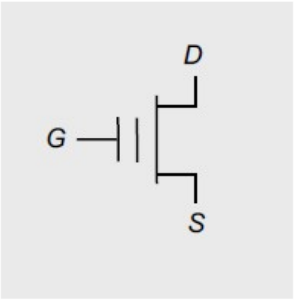
EEPROM (Electrically Erasable Programmable Read-Only Memory). This is a memory whose writing and erasing is done with electrical voltage. The memory cell also uses a floating gate but its erase operation is not with UV but with an electrical pulse. Unlike EPROM the EEPROM may be selectively erased. This type of memory is non-volatile, as it retains stored data when power turns off. The read and write cycles are relatively slow. Thus this type of memory is suitable for operations that do not affect the overall speed, typically when data must be downloaded at startup. Another important feature is that the write and erase operations are on a byte per byte basis. The number of rewrite cycles is very large (10 millions or more); this feature is called endurance. The retention time of data is limited because the floating gate insulation is not perfect and charge can be lost especially at high temperatures. Typical data retention is 10 years

Flash memory. Flash memory is an evolution of EEPROM technology. The key feature of non-volatile flash memory is that blocks of memory may be erased simultaneously. Data are read on an individual cell basis. There are two different technologies of flash memory, distinguished by the logic function exploited in the cells: NOR and NAND. The NAND type is preferred for memory cards because of the lower cost and higher storage capacity. The memory can store a single bit or multiple bits per cell. Many bits per cell augment the capacity but slow the transfer speed, increase power consumption and lower cell endurance. Often the assembling of flash memories uses chip stacking technology

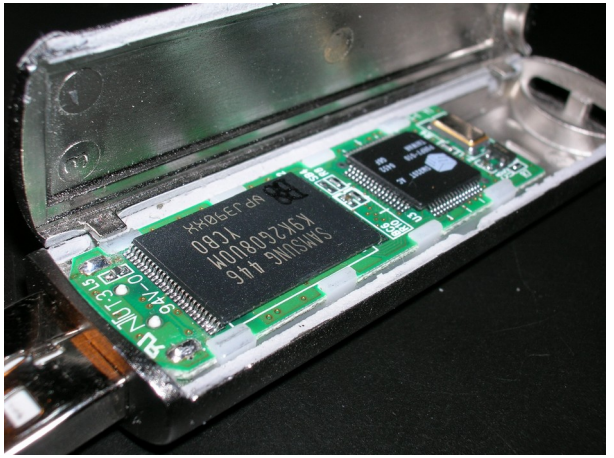
Flash memory – floating gate transistor



Device cross-section

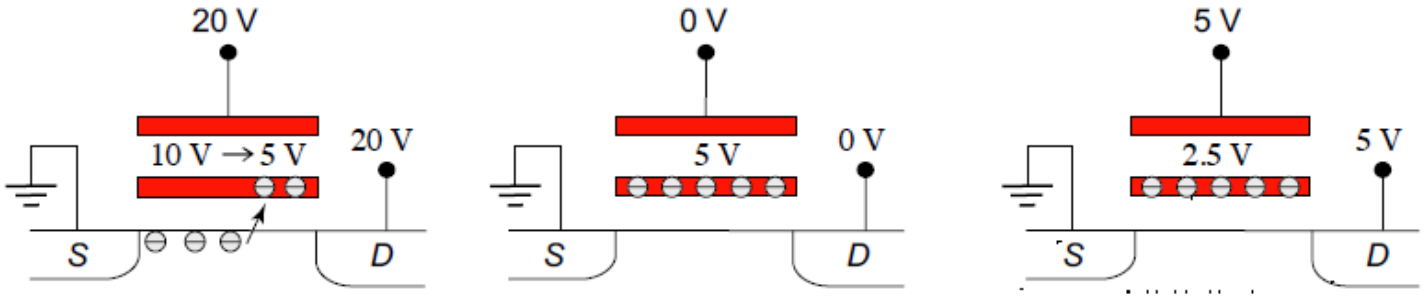


Schematic symbol



USB drive

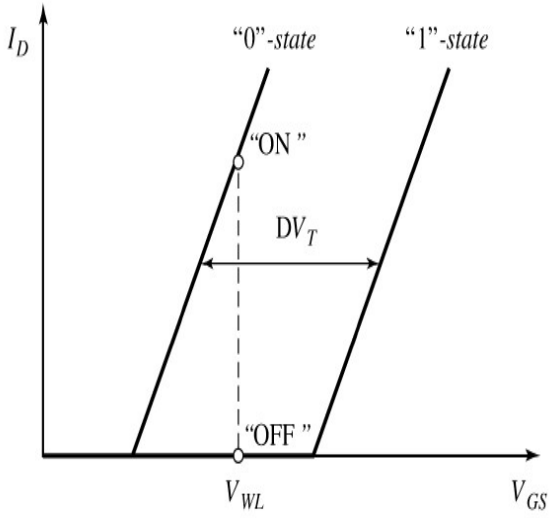
Floating-Gate Transistor Programming



Avalanche injection

Removing programming voltage leaves charge trapped

Programming results in higher V_T.



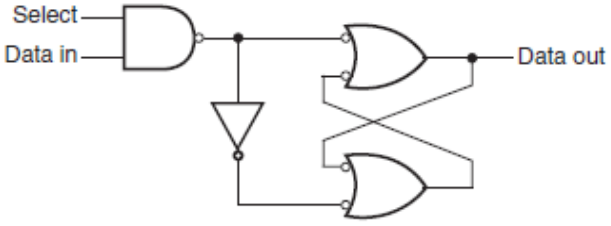
→ a programmable threshold transistor

Courtesy S.Marchioro (CERN)

Memory zoology - RAM

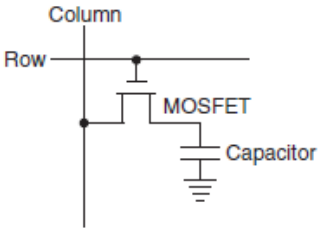
DRAM (Dynamic RAM) is a volatile form of random access memory. It uses a capacitor to store each bit of data with the charge on it used to determine the logical value “1” or “0”. The cell is very simple being made of a capacitor and a switch. However the capacitor does not hold its charge indefinitely, because of leakage. To overcome this problem the data is periodically refreshed. The sensing of data allows its re-instate of the value on the same cell. Typically, refreshing occurs every 64ms. This activity is accomplished over single rows or portions of the memory, by relatively simple refresh circuitry and minimum interference with system operation. DRAM in its various types makes most system memories because it is cheap and small. The access time is relatively long, typically not less than 60 ns

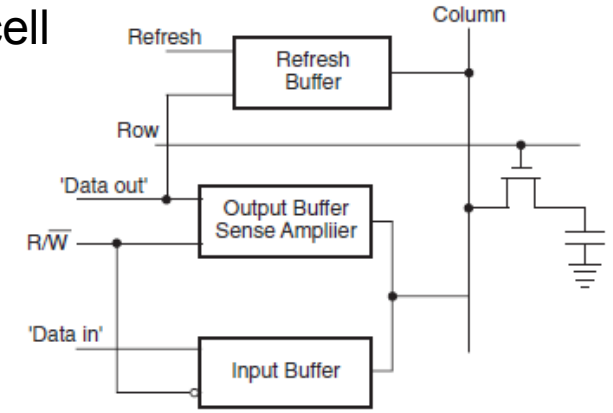
SRAM (Static Random Access Memory). Static memory cells do not need refreshing the content. The access time is fast (10 ns against 60 ns for DRAM). Also, the cycle time is short because pauses between accesses are not necessary. The power consumed can be competitive with DRAM or even give rise to better performance. However, the SRAM is less dense and more expensive than DRAM. In computers, SRAMs are normally used for caches.



Basic Static RAM cell

Basic Dynamic RAM cell





Memory zoology - RAM

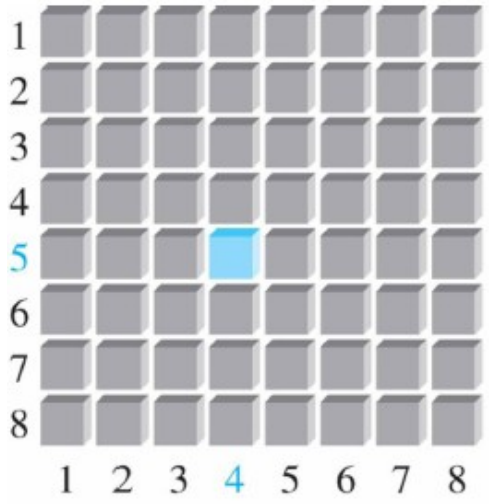
SDRAM (Synchronous DRAM) is a special DRAM architecture that obtains fast speeds. The key to this architecture is the synchronization of operation with the clock of the signal processor that uses it. The control of the memory keeps two sets of memory addresses opened at the same time and transfers data alternately from the two sets. Overlapped operation avoids the delay caused by needing to close one address bank before opening the next one.

MRAM (Magneto-resistive RAM, or Magnetic RAM). This is a type of memory that uses magnetic features instead of charges to store data. They use materials with magneto-resistive properties. MRAM can provide high-density non-volatile memories that are extremely fast (access time less than 10 ns). Its commercial use depends on the maturity level of the technology used.

Memory arrangement

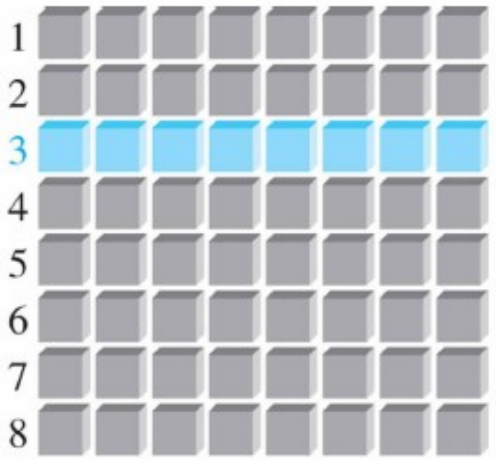
The location of a unit of data in a memory is called the **address**

64x1



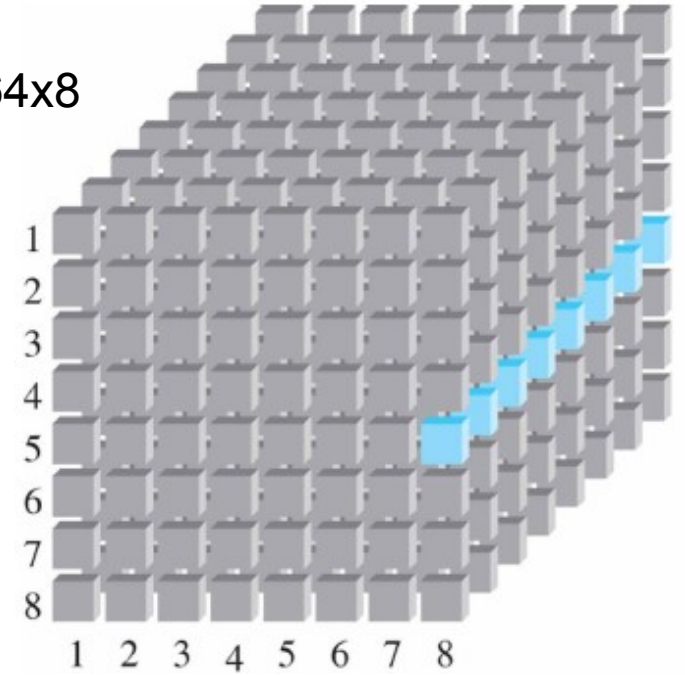
Address of blue bit
is (row 5, col 4)

8x8



Address of blue byte
is row 3

64x8



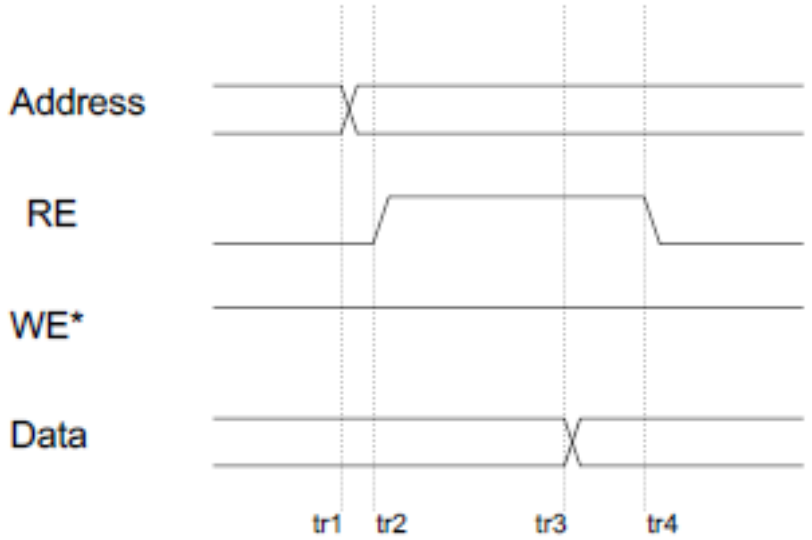
Address of blue byte
is (row 5, col 8)

The address depends on how the memory is organized into units of data

- Personal computers have random-access memories organized in bytes
- The smallest group of bits that can be addressed is eight

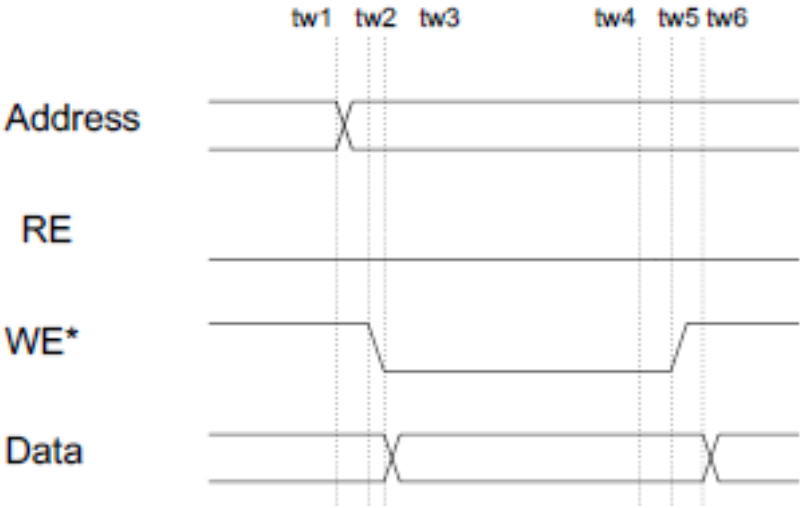
Memory access and timing

Read Cycle



The **read operation copies** data out of a specified address in the memory

Write Cycle



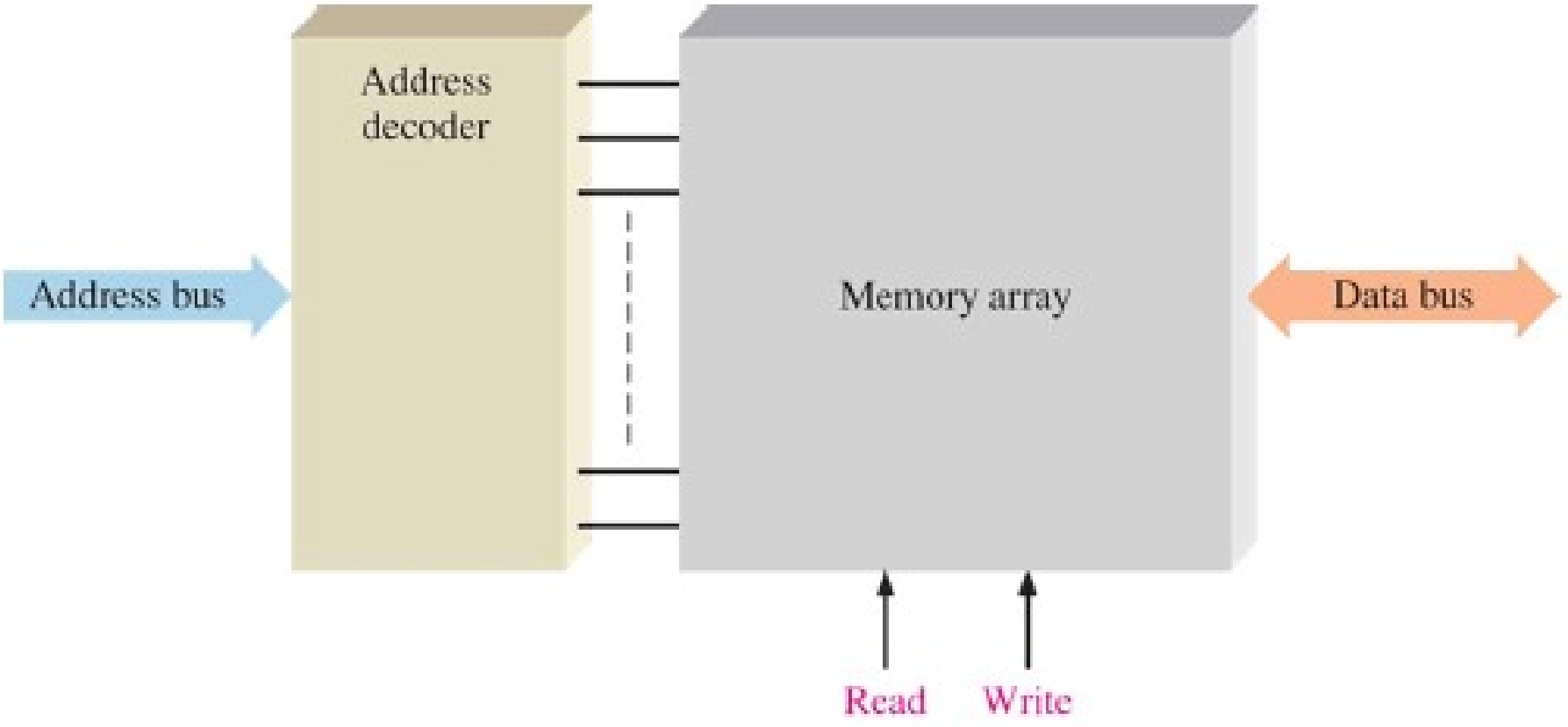
The **write operation** puts data into a specified address in the memory

The **addressing operation**, which is part of both the write and the read operations, selects the specified memory address

Data Bus

Data units go into the memory during a write operation and come out of the memory during a read operation on a set of lines called the **data bus**

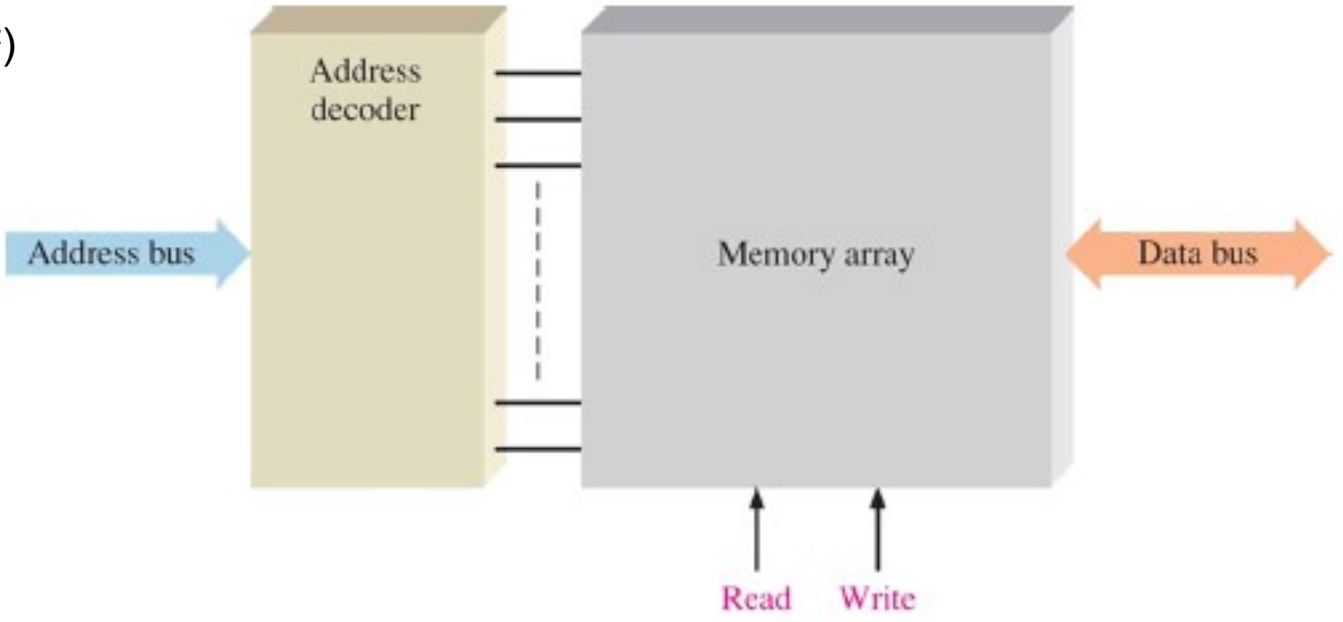
The data bus is **bidirectional**, which means that data can go in either direction (into the memory or out of the memory)



Data Bus

- For a write or a read operation, an address is selected by placing a **binary code representing** the desired address on a set of lines called the address bus
- The **number of lines in the address** bus depends on the capacity of the memory

- 16-bit address code can select 65,536 locations (2^{16}) in the memory
- 32-bit address code can select 4,294,967,296 (4G) locations (2^{32}) in the memory

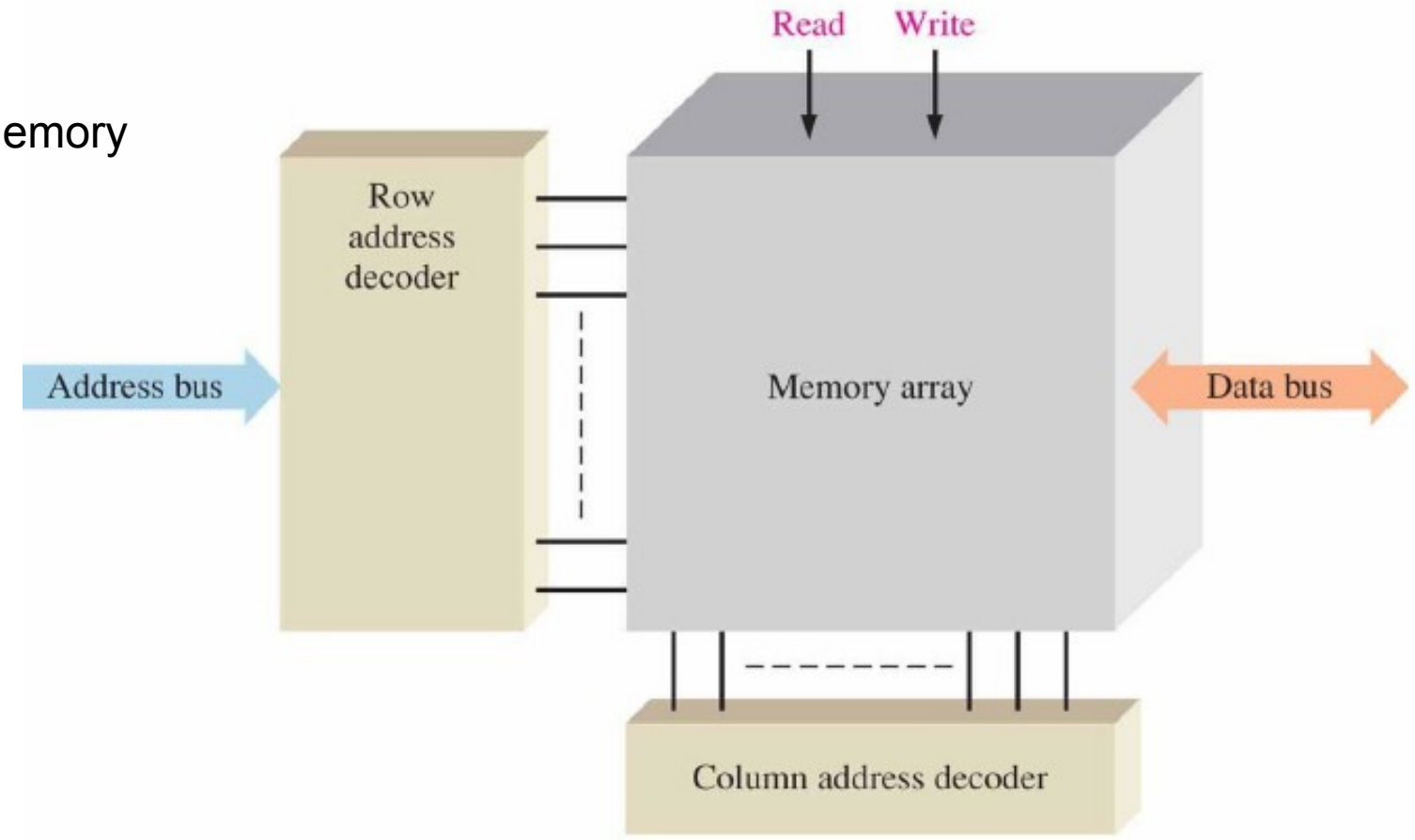


Data Bus

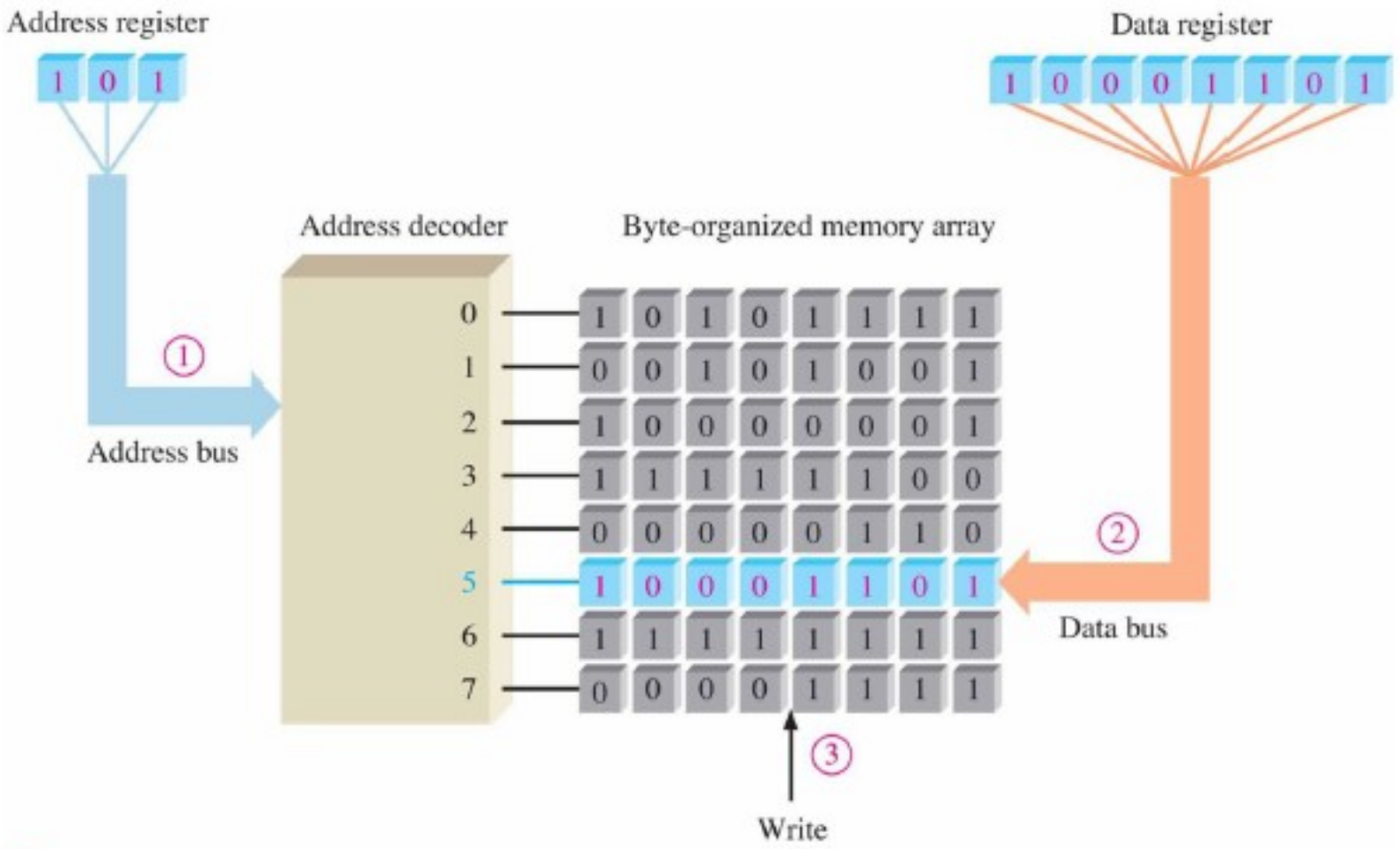
• For a write or a read operation, an address is selected by placing a **binary code representing** the desired address on a set of lines called the address bus

• The **number of lines in the address** bus depends on the capacity of the memory

3D memory array



Simplified Write operation

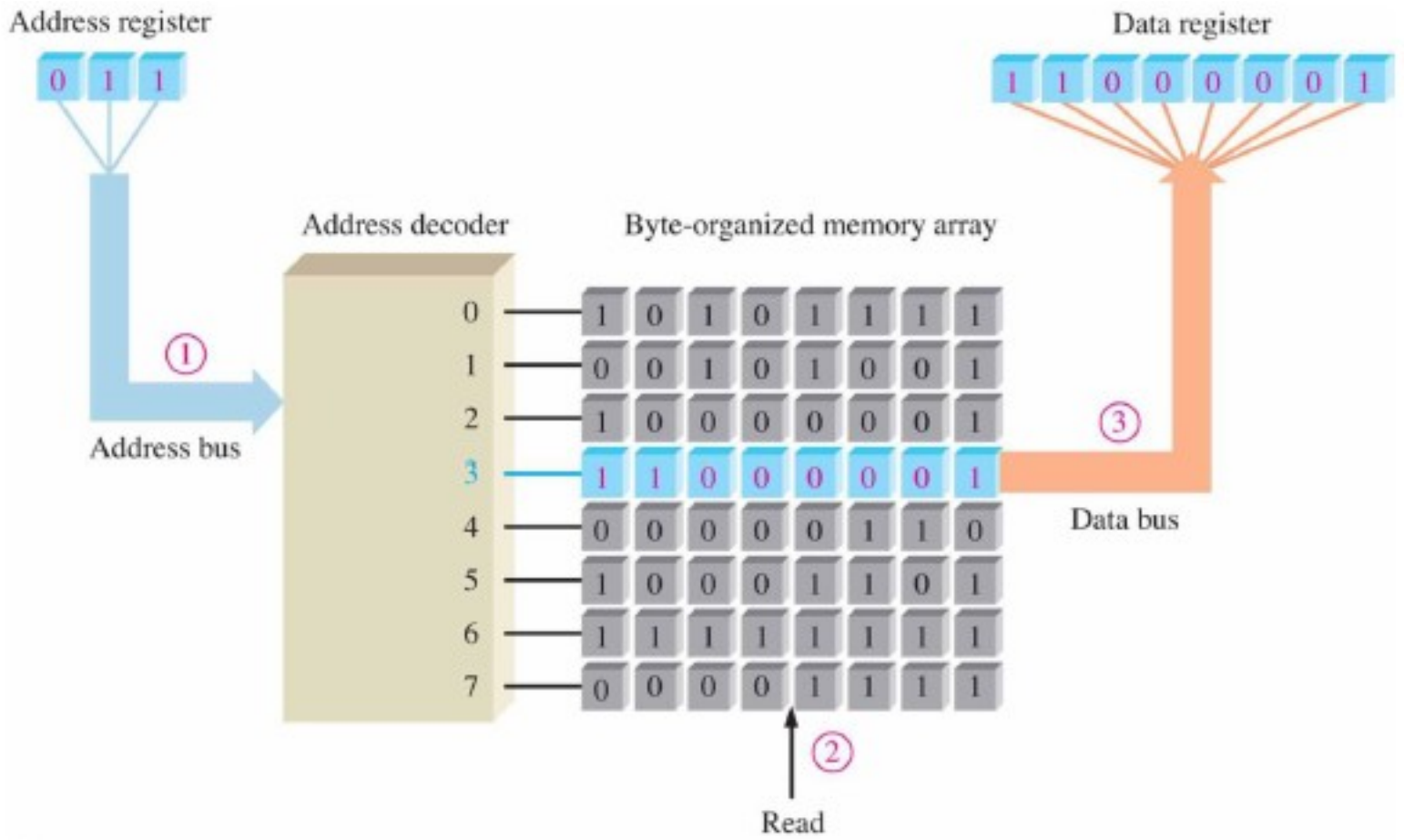


① Address code 101 is placed on the address bus and address 5 is selected.

② Data byte is placed on the data bus.

③ Write command causes the data byte to be stored in address 5, replacing previous data.

Simplified Read operation



① Address code 011 is placed on the address bus and address 3 is selected.

② Read command is applied.

③ The contents of address 3 is placed on the data bus and shifted into data register. The contents of address 3 is not erased by the read operation.

Address multiplexing

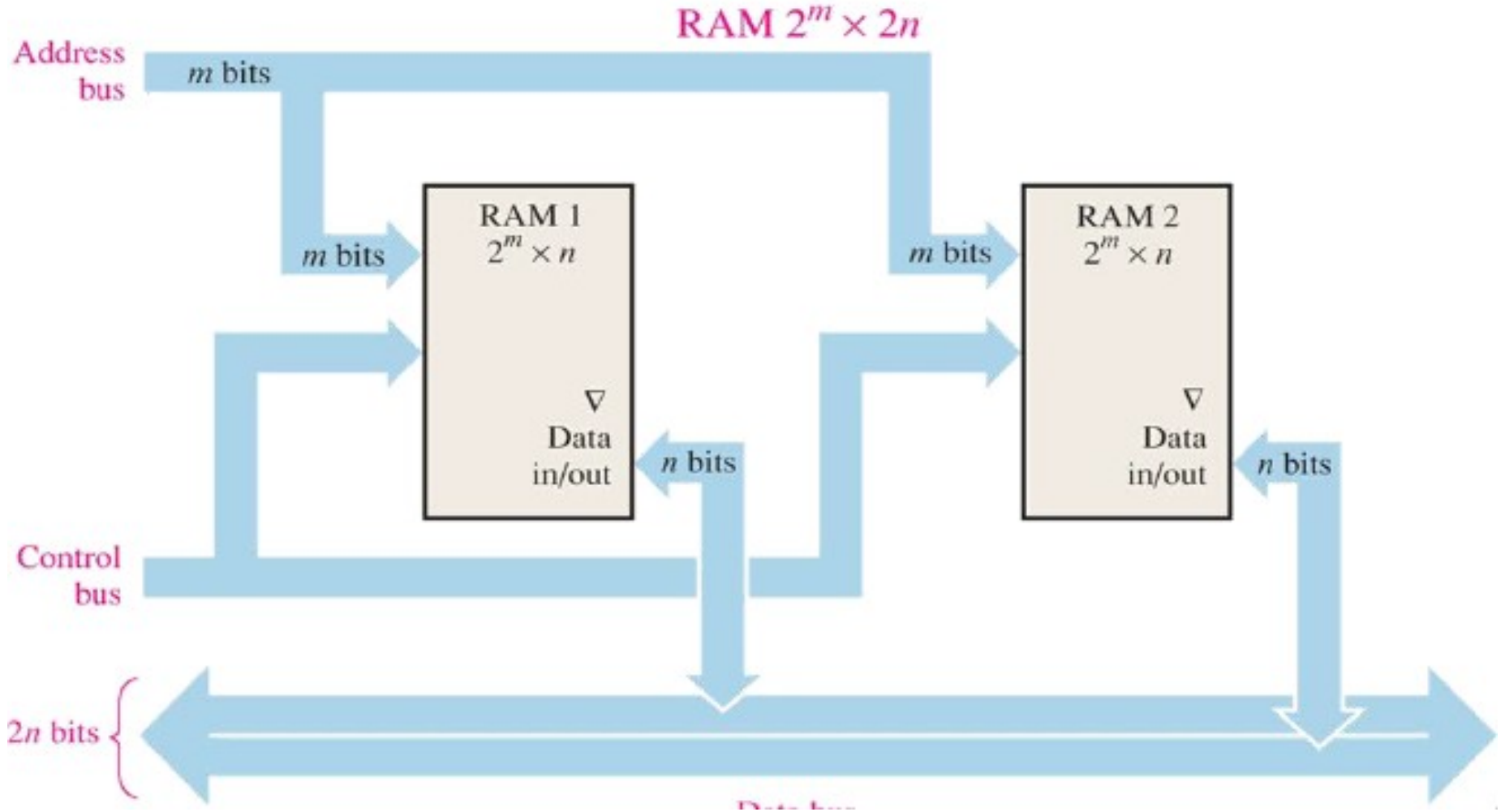
DRAMs use a technique called **address multiplexing** to reduce the number of address lines

For example, consider a DRAM which have 10 address lines

- The ten address lines are time multiplexed at the beginning of a memory cycle into two separate 10-bit address fields: the row address and the column address
- First, the l0-bit row address is latched into the row address latch. Next, the l0-bit column address is latched into the column address latch
- The row address and the column address are decoded to select one of the 1048576 addresses (2^{20}) in the memory array

Memory expansion

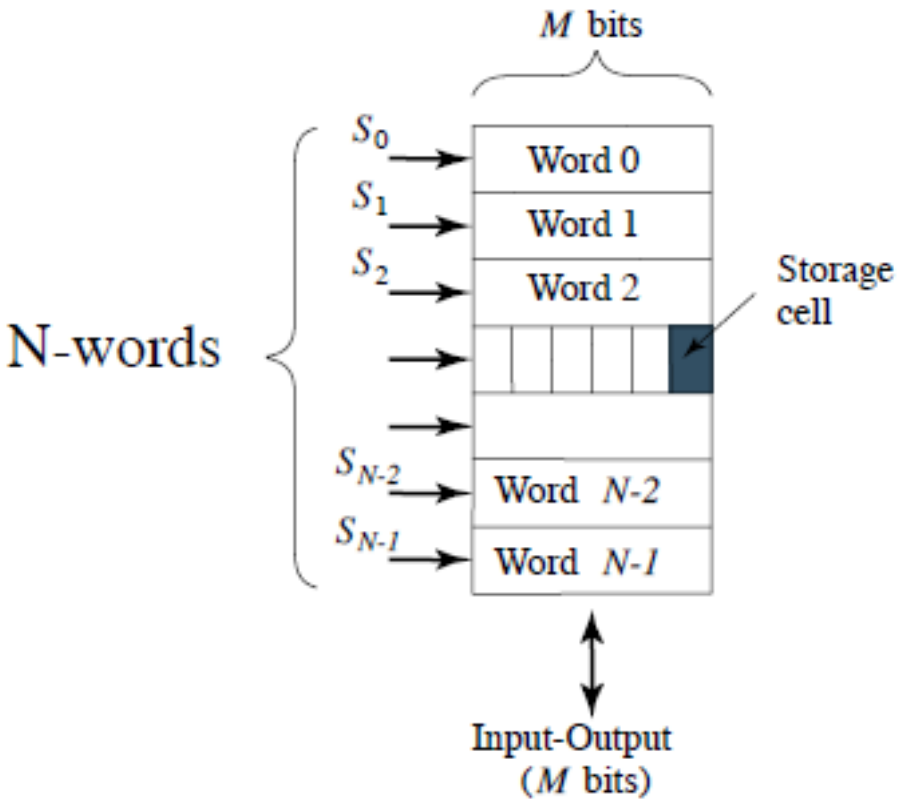
- Example: One $2^m \times 2n$ RAM from two $2^m \times n$ RAMs

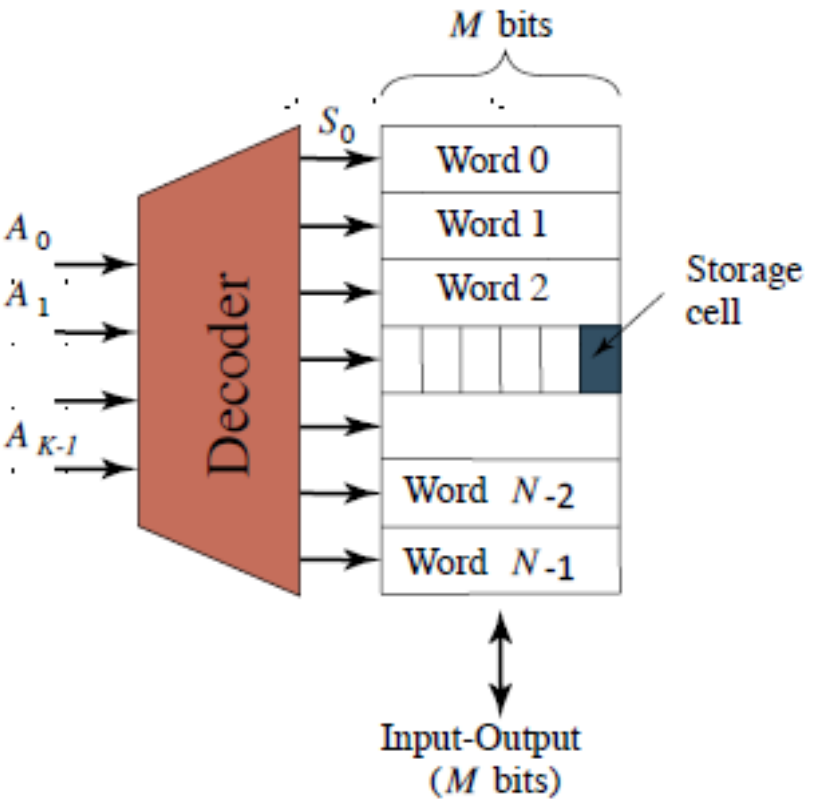


Memories Architecture

- Random Access
- Contents Addressable
- Sequential Access

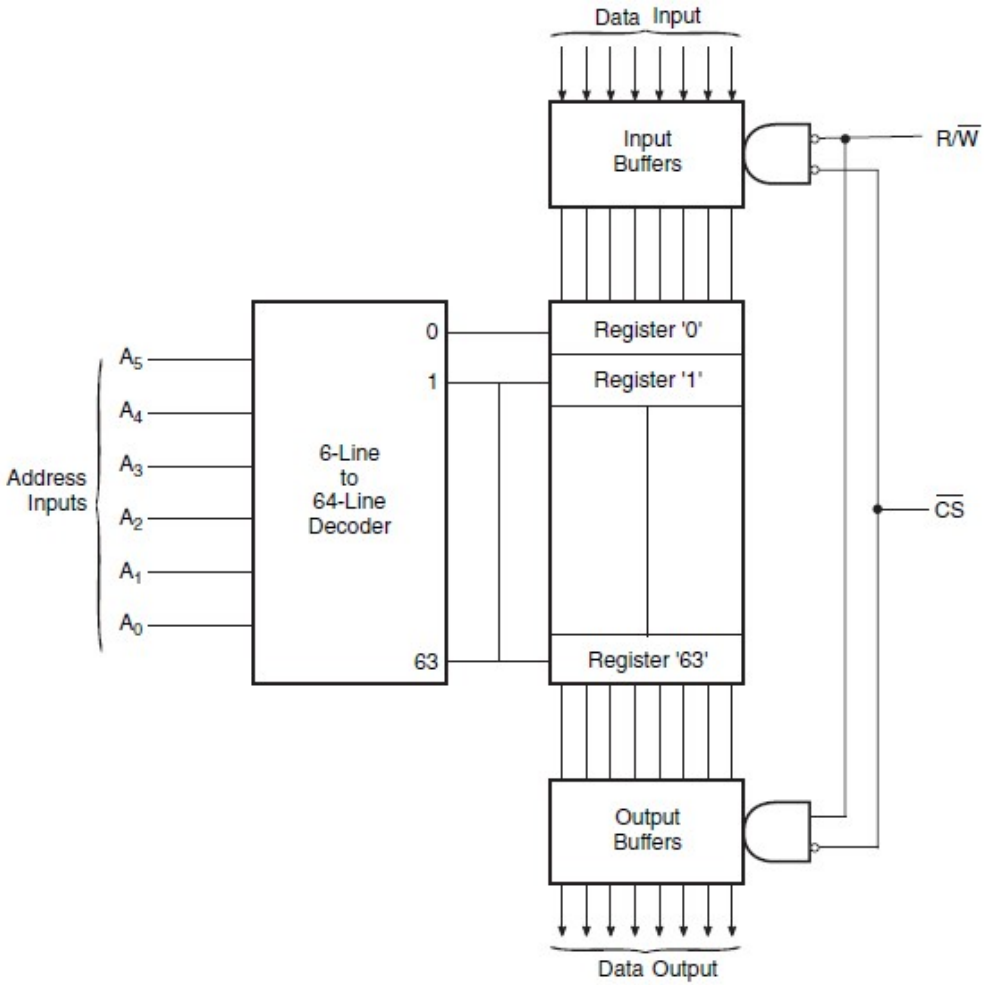
Random Access Memory architecture

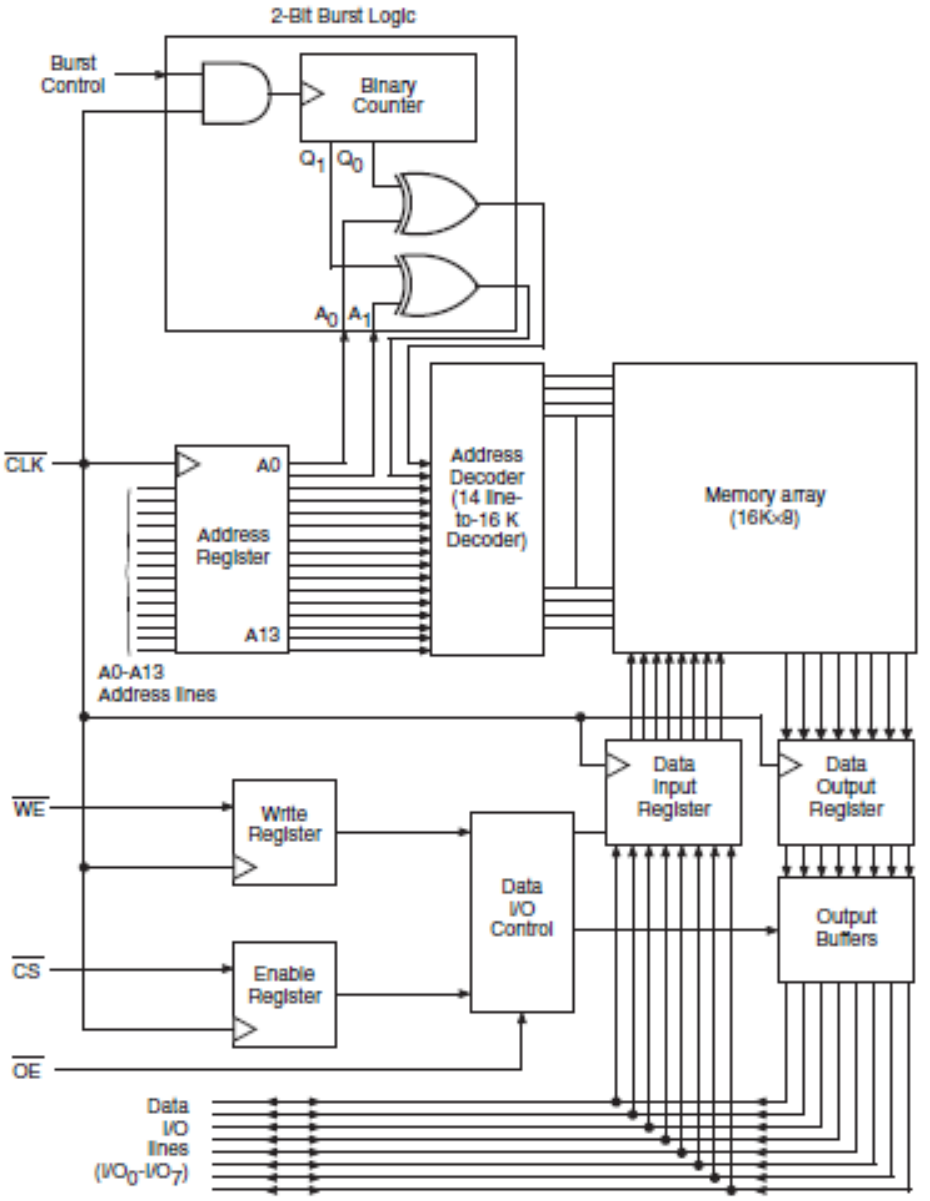




Static RAM - examples

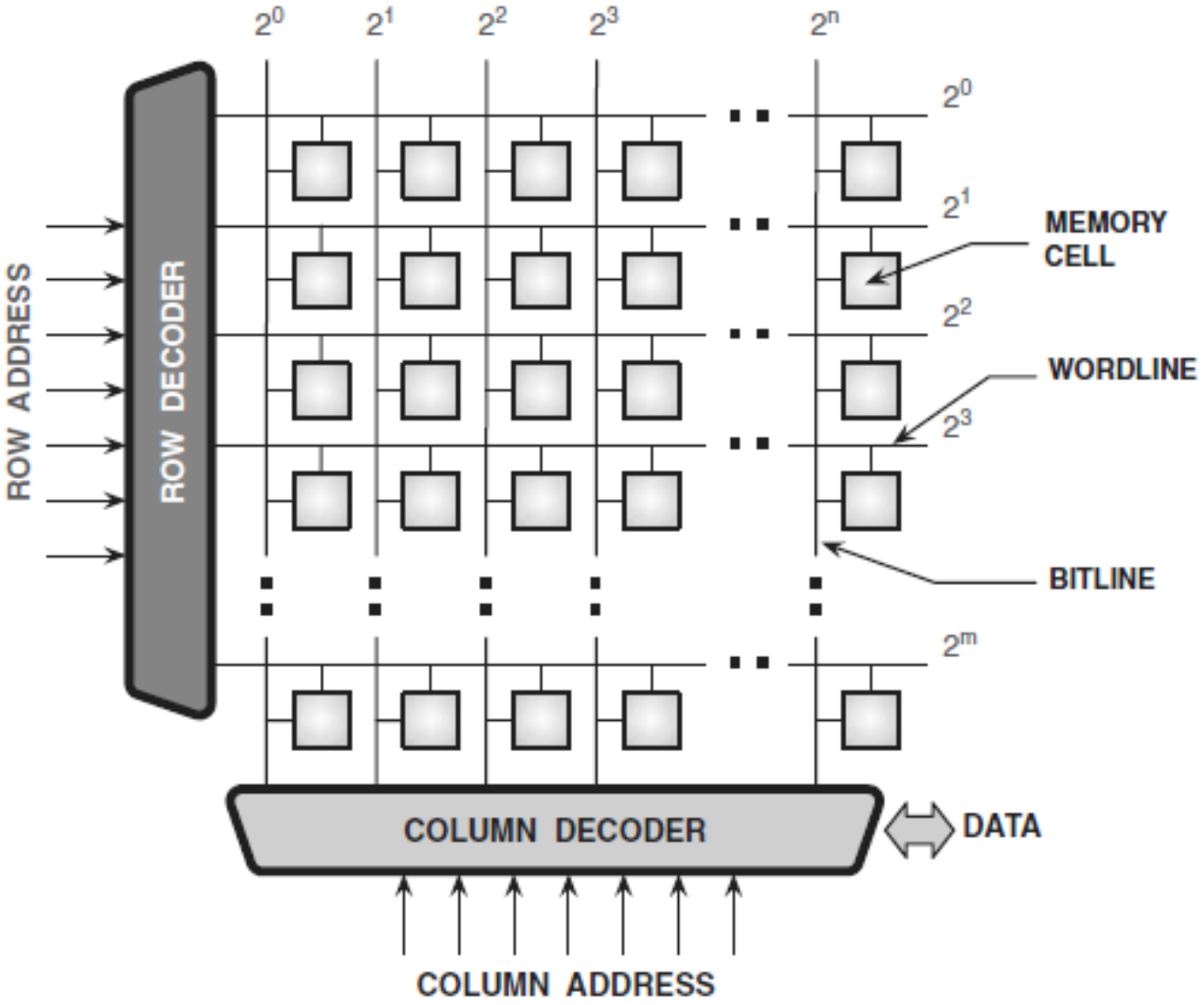
Asynchronous (64x8)



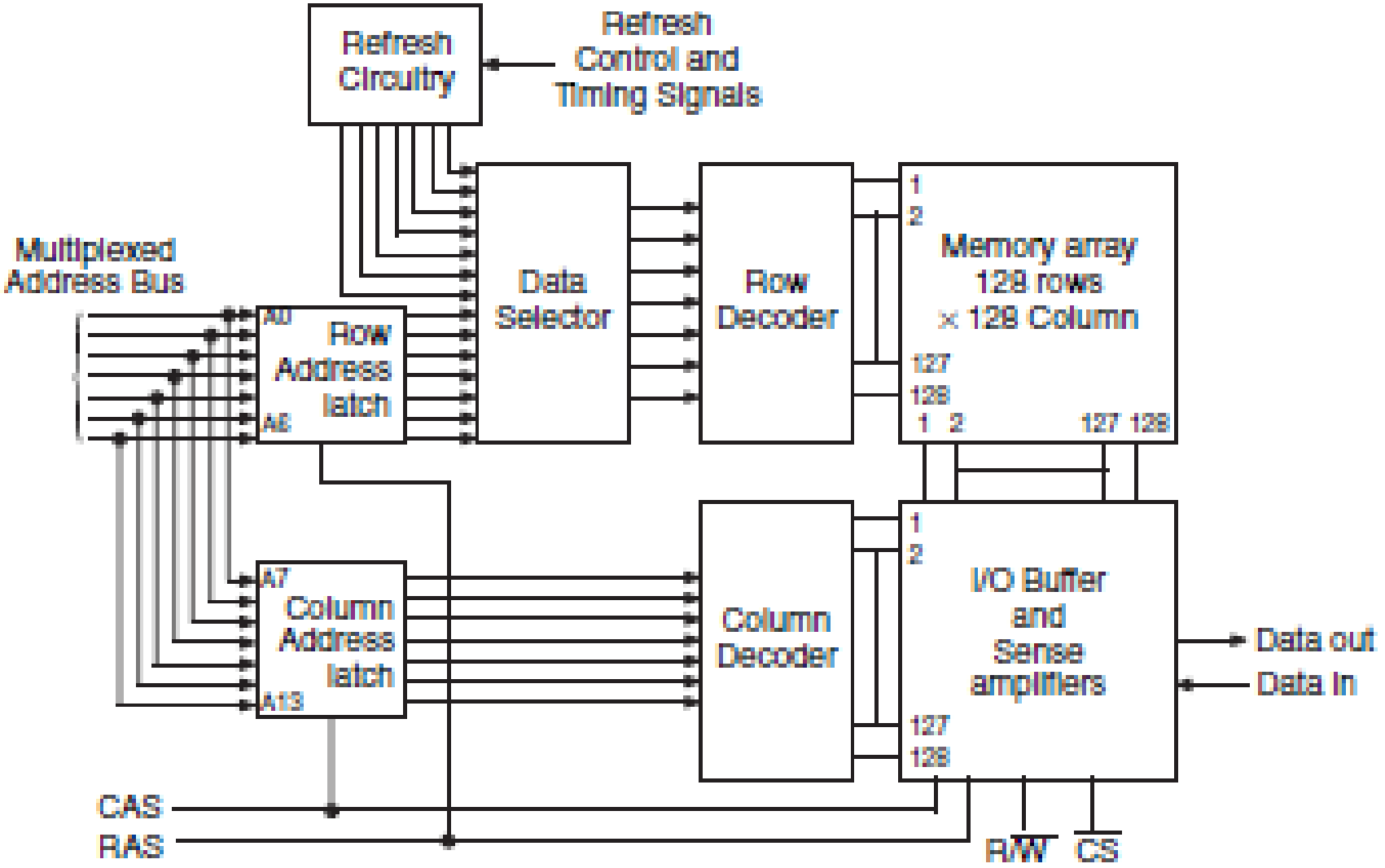


Synchronous (16kx8)

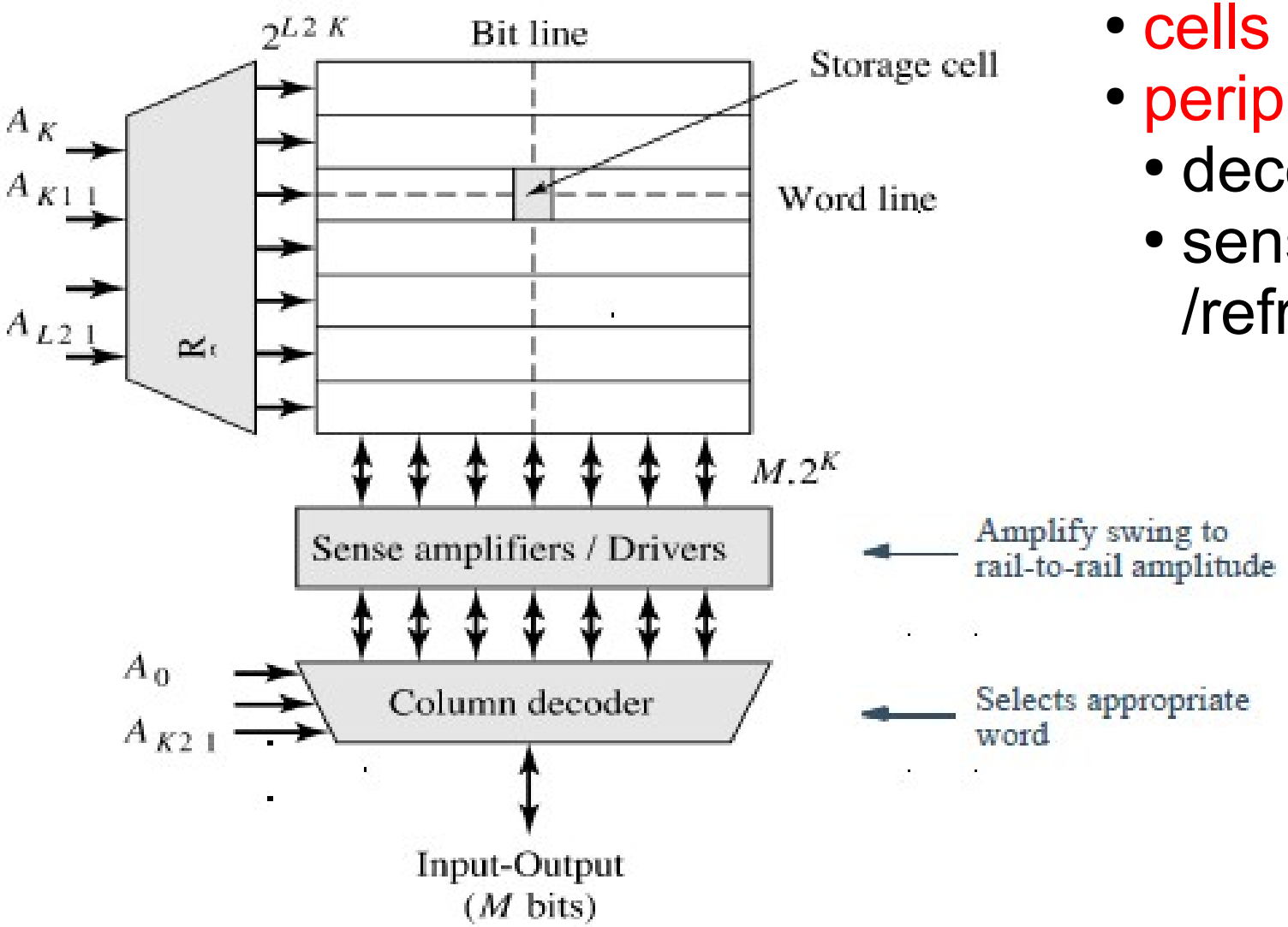
Array-structured RAM architecture



Dynamic RAM - examples

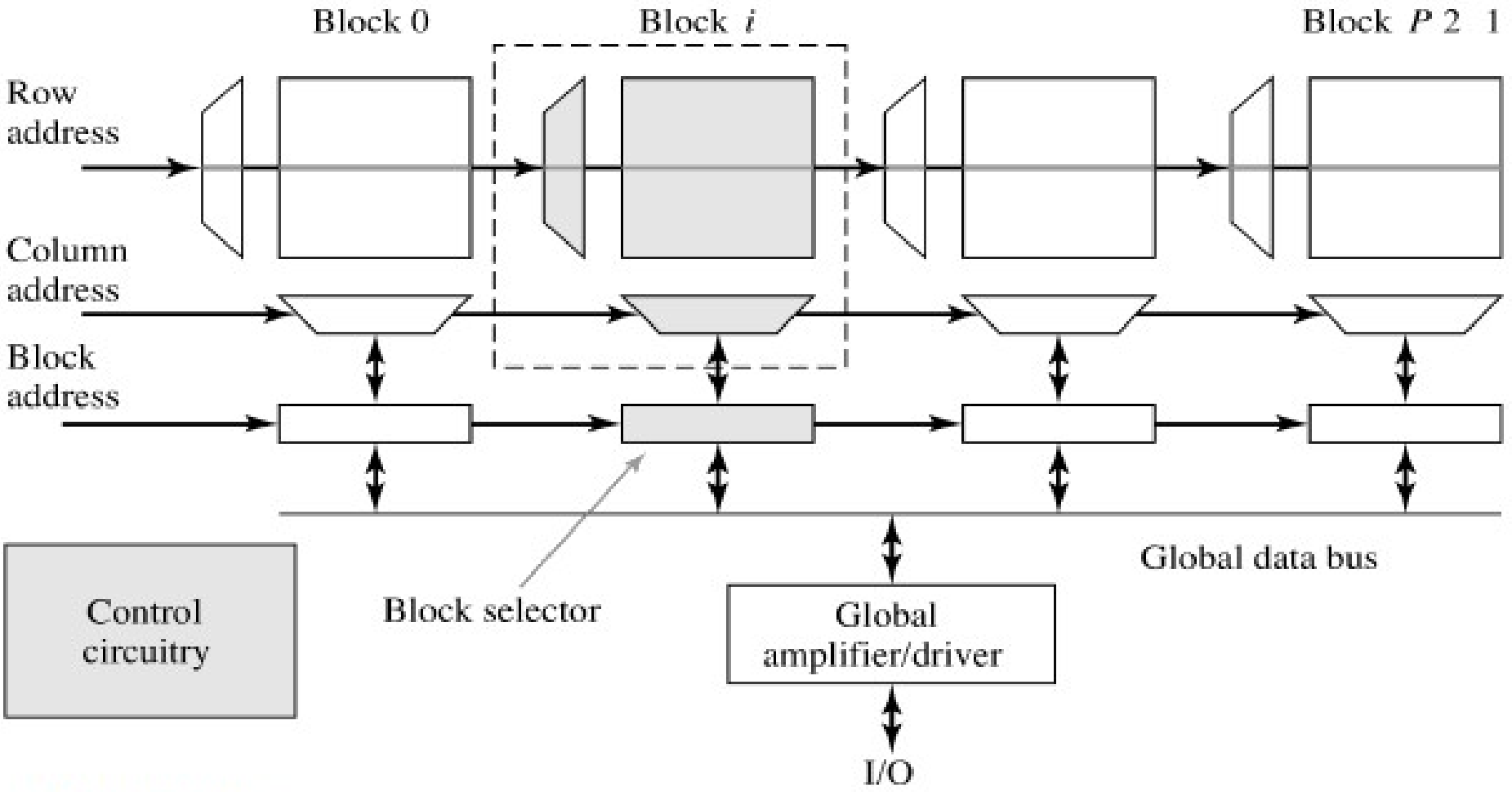


Array-structured RAM architecture



- cells
- periphery
  - decoder
  - sense amplifier /refreshing circuit

Hierarchical RAM architecture

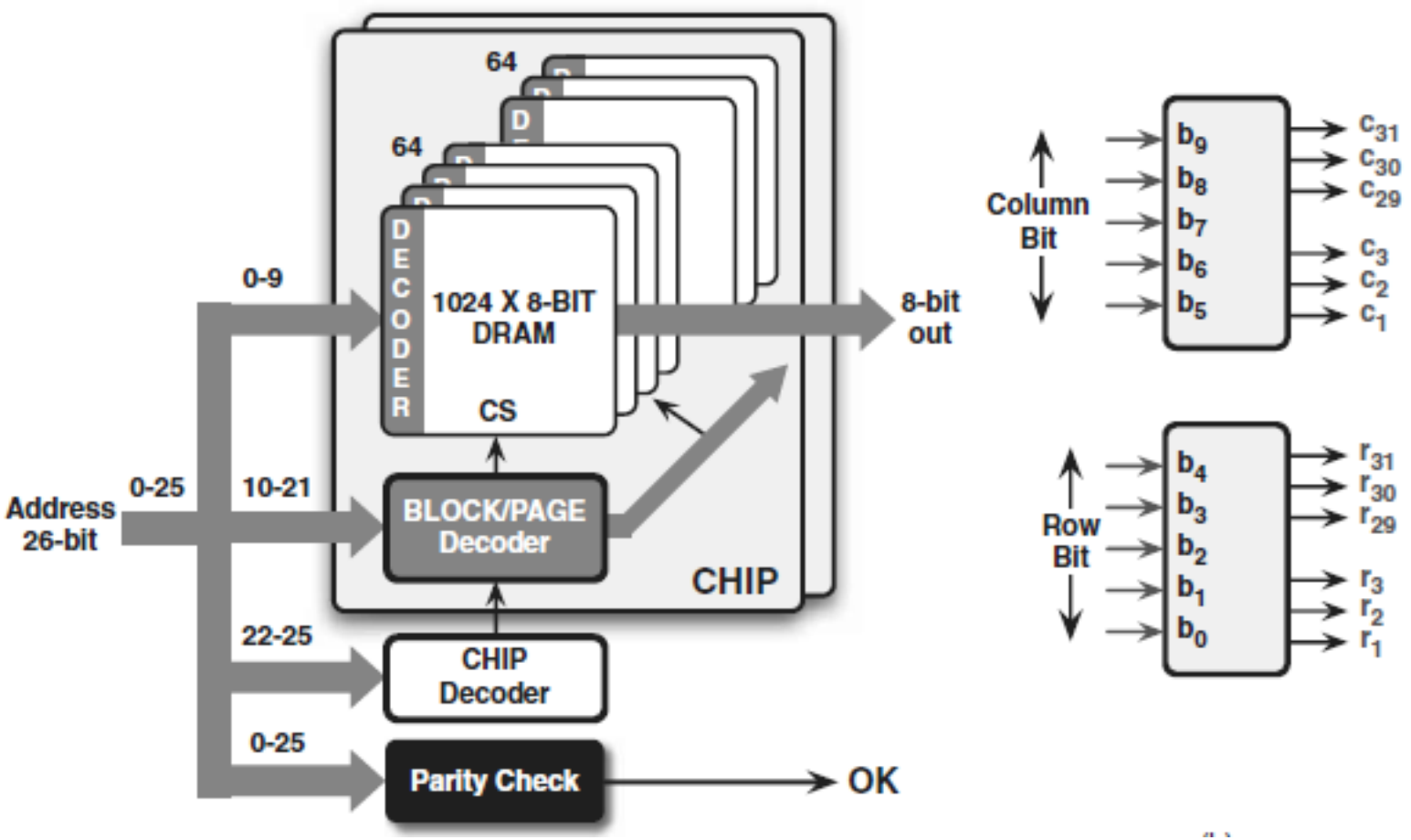


Advantages:

1. Shorter wires within blocks
2. Block address activates only 1 block => power savings

Periphery: decoders

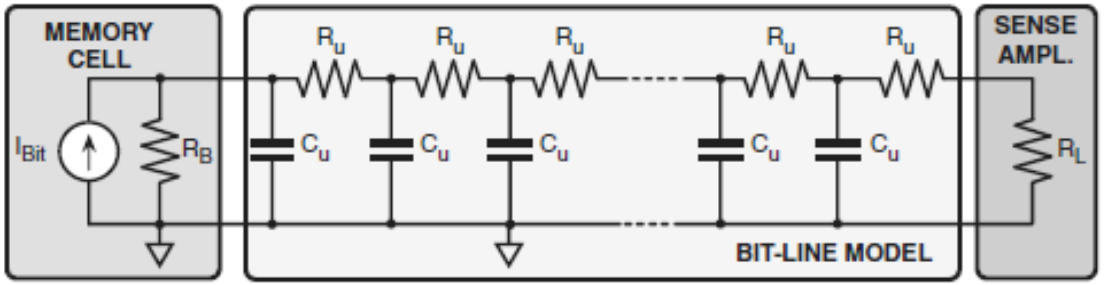
The circuit that implements the address decoder can be combinational logic made by suitable interconnection of boolean cells (AND, NAND, OR, NOR), a matrix of switches, or a programmable logic device with volatile or non-volatile definitions of addresses



Periphery: sense amplifiers

A sense amplifier, as the name implies, is used to sense or detect stored data from a selected memory cell.

It is a critical peripheral block, because the signals that travel over long bit-lines can be very small. The bit-line behaves like a resistive delay line, causing attenuation and delay. Therefore, other than accounting for the delay in column access time, it is necessary to amplify the small signal received to attain a full logic level



There are two types, the voltage sense amplifier and the current sense amplifier. The former focuses on measured voltage, and the latter assumes that the signal from the bit-line is mainly caused by current.

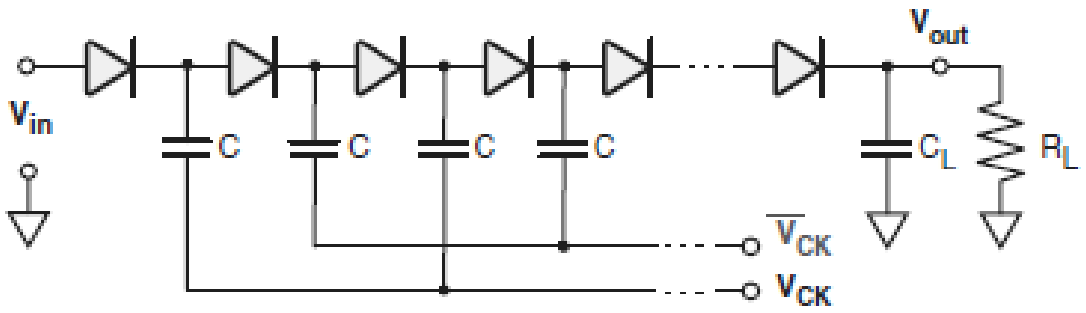
The choice depends, obviously, on the type of memory cell. In order to reduce noise sensitivity memory cells often provide differential signals so that the voltage amplifier senses differential voltages. The challenge is to find the best trade-off between gain and speed. A frequent solution is to use a fast amplifier with medium gain followed by a latch.
A positive feedback circuit is a common solution: indeed, it is not necessary to amplify a signal linearly but to rebuild a logic signal from its attenuated differential version.

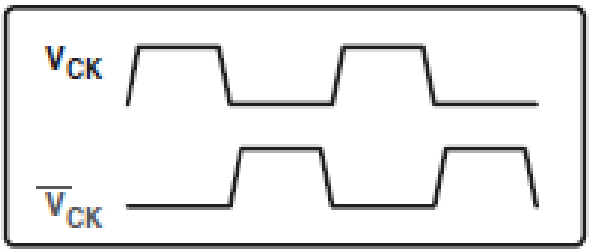
Using current-sensing amplifiers does not suffer the limitation of low voltage signals. What matters is to have current in the bit-line to indicate a measured high/low status or high/low voltage.

Periphery: refresher circuits (non-volatile memories)

Electrical programmability of memories requires high voltages to inject or remove charges in floating electrodes. Since the operational voltage is lower than that required for memory programming, it would be necessary to use two supply voltages.

This is a limitation that onchip high-voltage generators avoid. Fortunately, programming memories requires high voltage but limited current. Thus the so-called voltage multiplier, or charge pump, based on a scheme proposed by Dickson, is the solution that is mostly used. The circuit uses capacitors and diodes with complementary clock signals



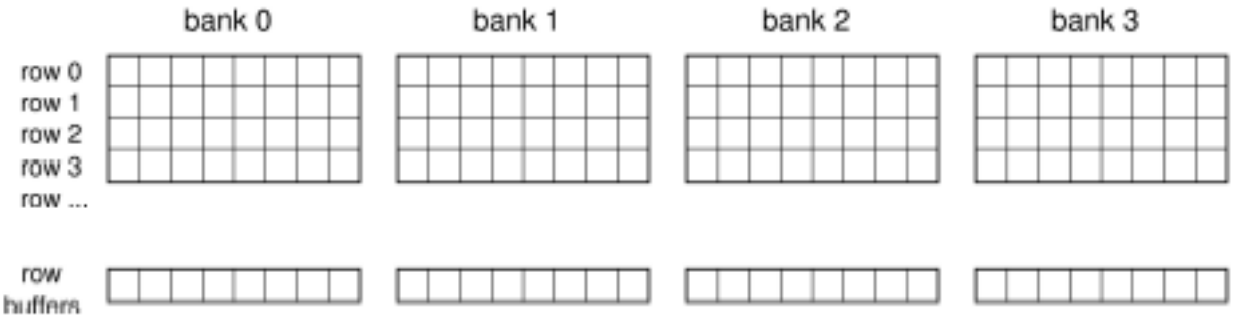


SDRAM evolution

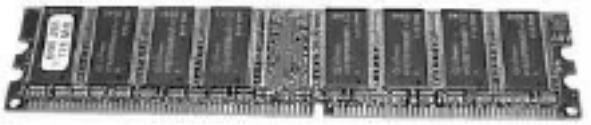
There has been multiple improvements to the DRAM design in the past 20 years
– A clock signal was added making the design synchronous → SDRAM
– The data bus transfers data on both rising and falling edge of the clock → DDR SDRAM
– Second generation of DDR memory (DDR2) scales to higher clock frequencies
– DDR3 … DDR4 … DDR5 standardization

- SDRAMs have a multi-bank architecture and is organized in banks, rows and columns.

Typical figures:
4 – 8 banks
16K rows / bank
1024 columns / row
4 – 16 bits / column

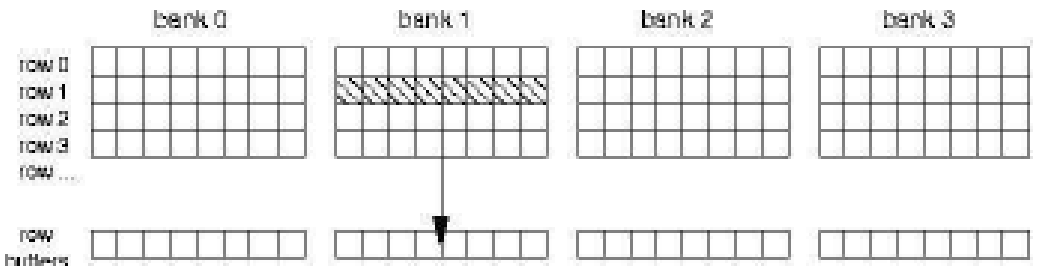


- Many chips are combined on a memory module to increase the word width. This is called the *memory configuration*.

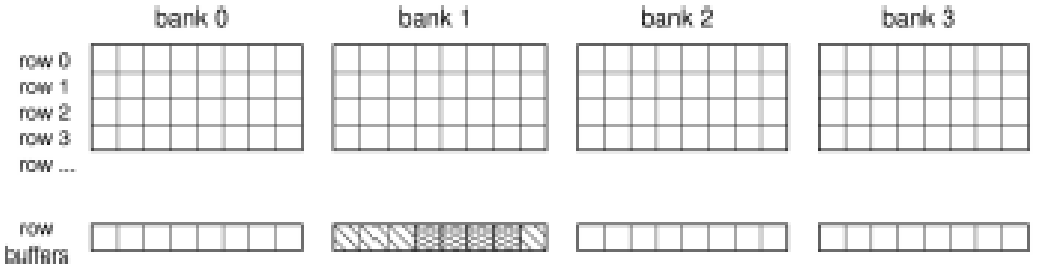


SDRAM Naive memory access model

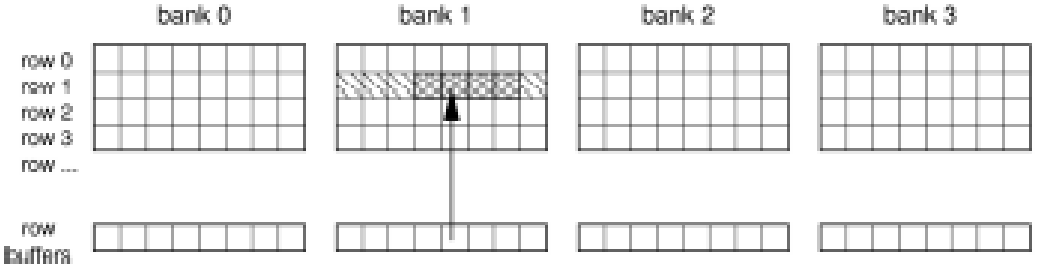
- The requested row is *activated* and copied to the row buffer of the corresponding bank.



- *Read* and/or *write* bursts are issued to the active row.

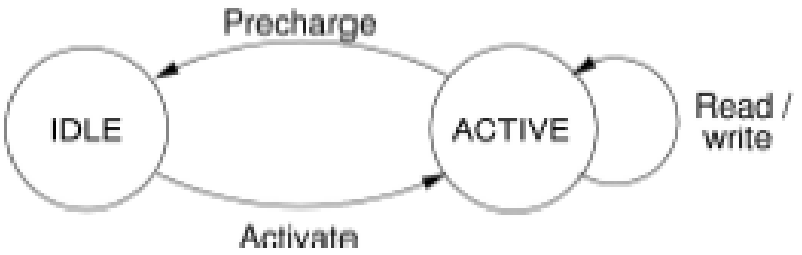


- The row is *precharged* and stored back into the memory array.



Note: all banks need to be pre-charged when refresh is issued

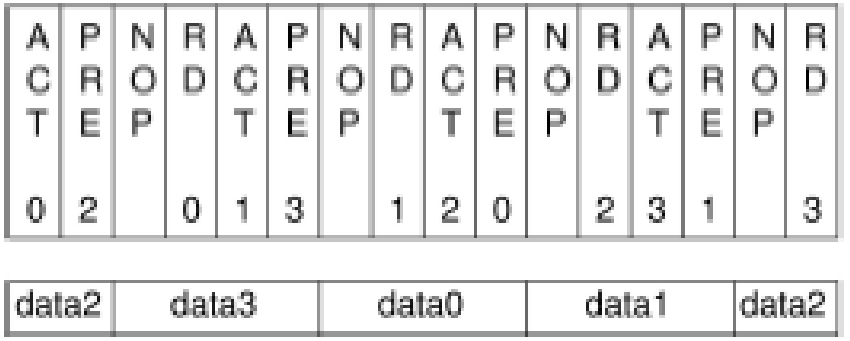
SDRAM Command summary



No operation	NOP	Ignores all inputs
Activate	ACT	Activate a row in a particular bank
Read	RD	Initiate a read burst to an active row
Write	WR	Initiate a write burst to an active row
Precharge	PRE	Close a row in a particular bank
Refresh	REF	Start a refresh operation

SDRAM Pieplined memory access

- The SDRAM interface has a separate data and command bus.
- When data is transferred to or from a bank other banks are activated and precharged (*bank preparation*).
- Pipelining memory accesses increases efficiency and throughput.



Special Memories:

- Associative
- Sequential
- …

Content Addressable Memory

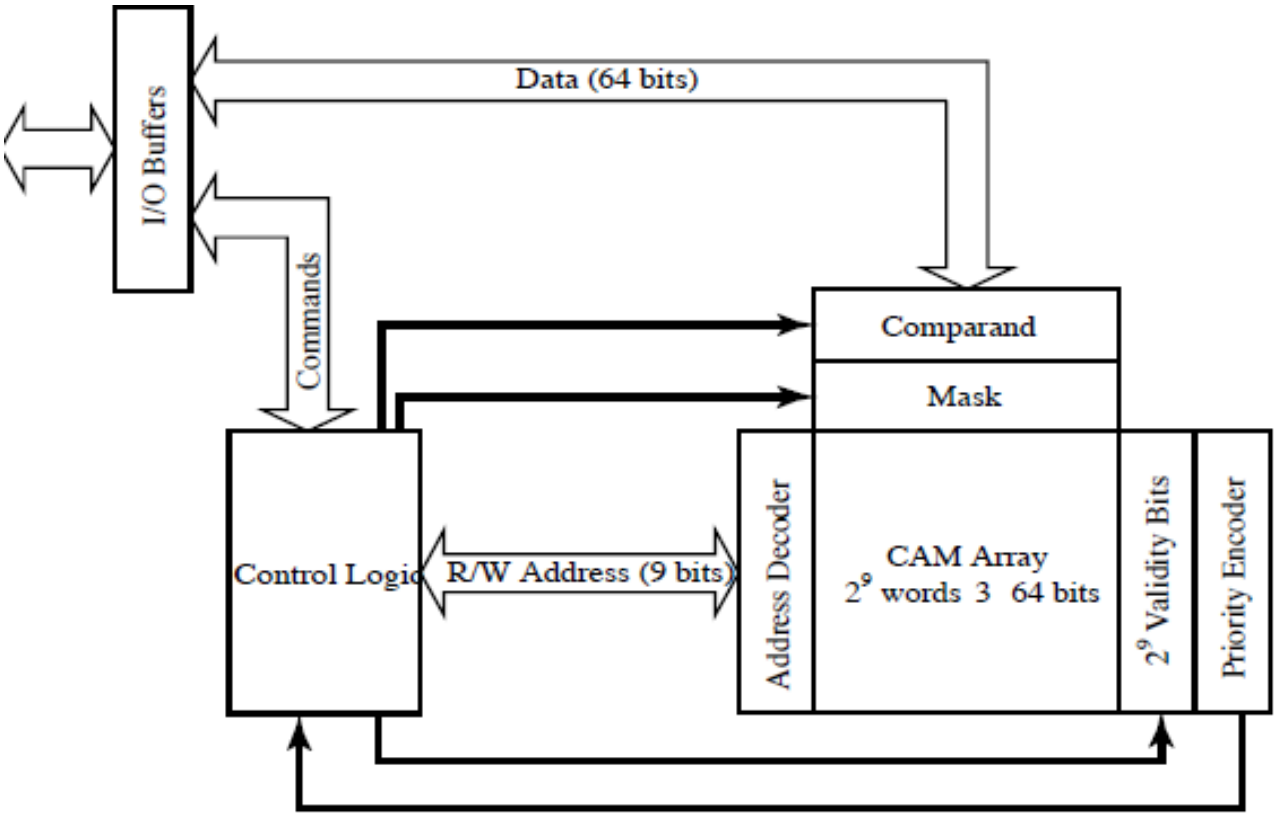
A CAM is designed such that the user supplies a data word and the CAM searches its entire memory to see if that data word is stored anywhere in it

If the data word is found, the CAM returns a list of one or more storage addresses where the word was found (and in some architectures, it also returns the contents of that storage address, or other associated pieces of data)

A CAM is the hardware embodiment of what in software terms would be called an **associative memory** (or array)

CAM is frequently used in networking devices where it speeds forwarding information base and routing table operations

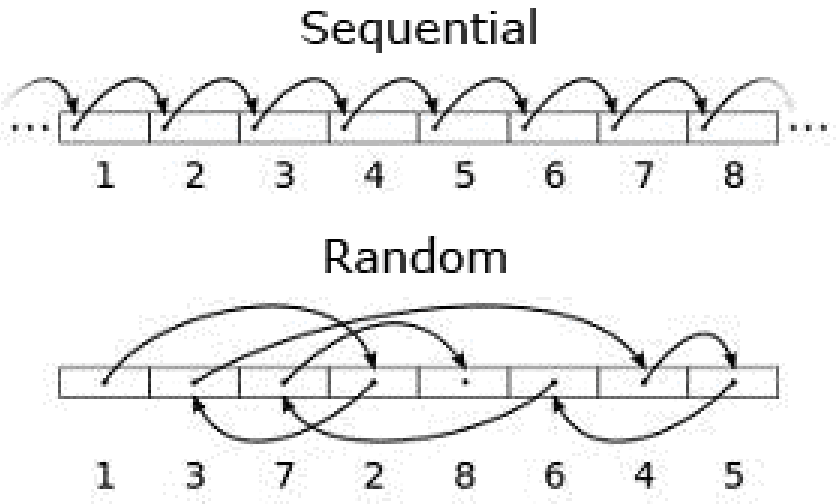
In general CAM are used in very-high-speed searching application



Sequential Memory

Sequential access memory (SAM) is a class of data storage devices that read stored data in a sequence. This is in contrast to random access memory (RAM) where data can be accessed in any order

Access Method Diagram



A common example of sequential access is with a tape drive, where the device must move the tape's ribbon forward or backward to reach the desired information.

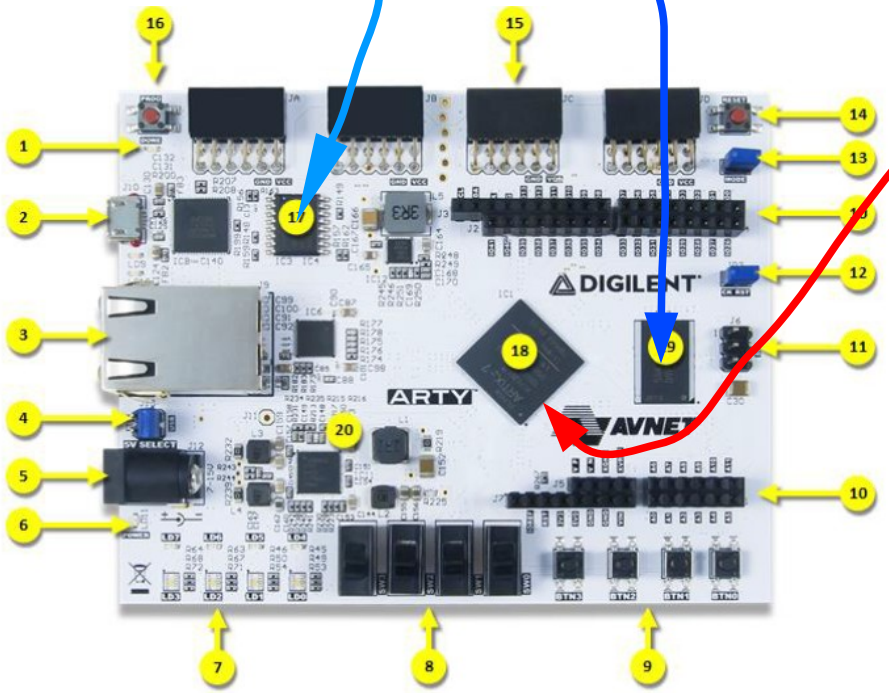
Note: even if not strictly classified as Sequential Access,
Shift Registers and their derivates (FIFOs, etc…) are infact sequential access memories

Memory on the ARTY7 board

Memory on ARTY7 board

- Xilinx Artix-35T FPGA
  → ~ 40k FF
  → 1800 kbits of fast block RAM

- System
  → 256MB DDR3L with a 16-bit bus @ 667MHz
  → 16MB Quad-SPI Flash



FPGA – block RAM → Dual Port RAM

7 Series FPGAs
Memory Resources

XILINX

User Guide

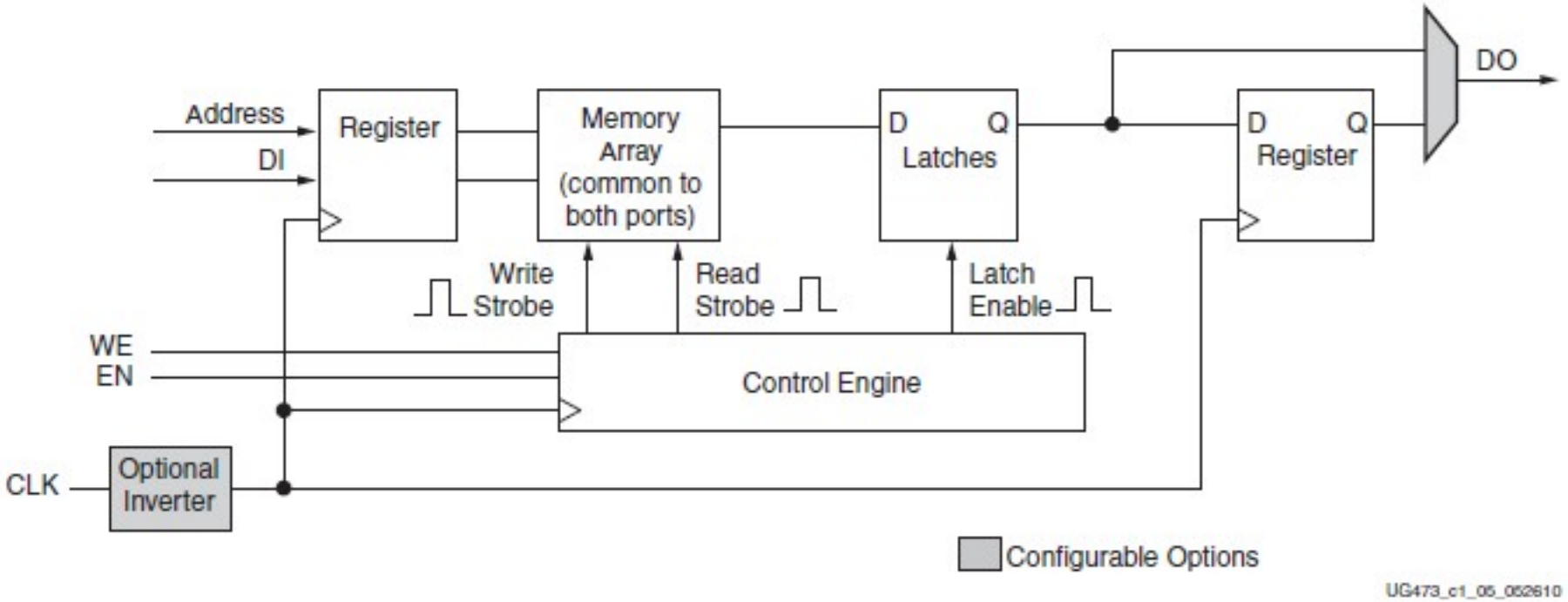


Figure 1-5: **Block RAM Logic Diagram (One Port Shown)**

FPGA – block RAM → Dual Port RAM

7 Series FPGAs Memory Resources

XILINX

User Guide

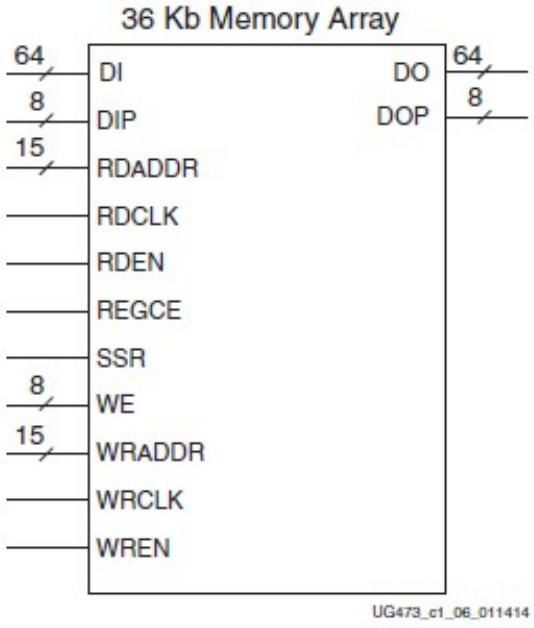


Figure 1-6: **RAMB36 in the Simple Dual-Port Data Flow**

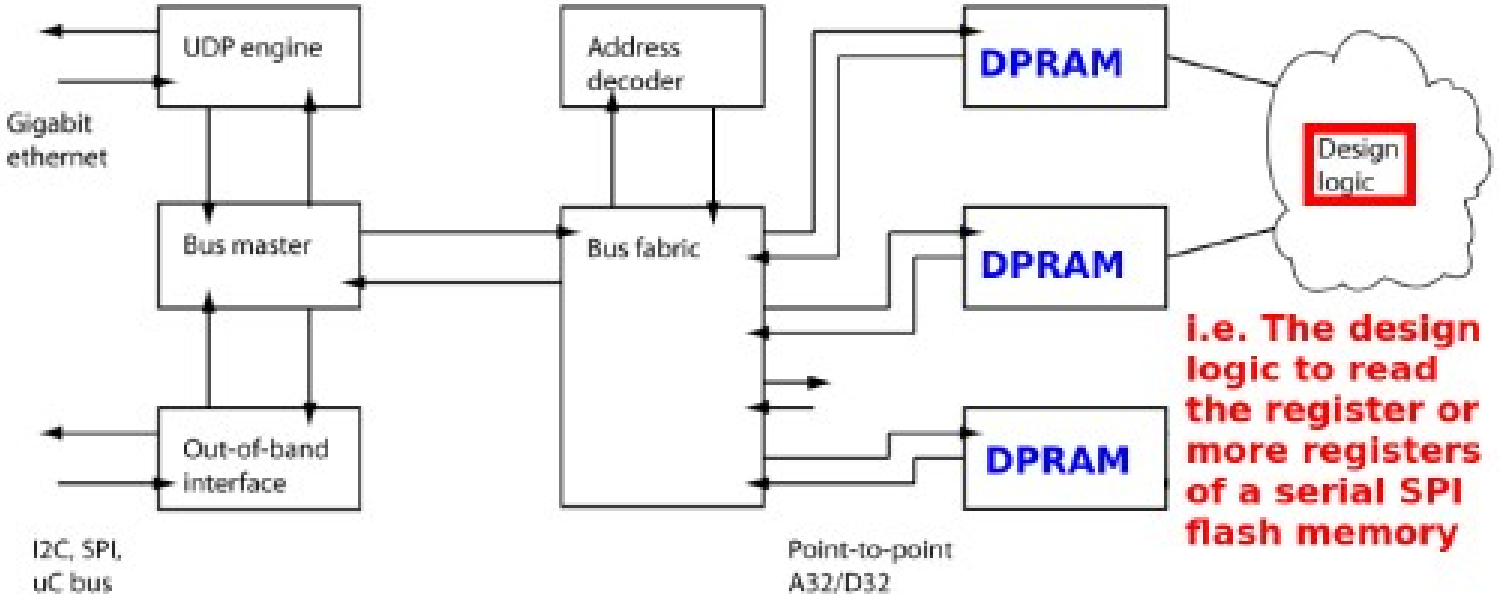
Table 1-6: **Simple Dual-Port Functions and Descriptions**

Port Function	Description
DO	Data Output Bus
DOP	Data Output Parity Bus
DI	Data Input Bus
DIP	Data Input Parity Bus
RDADDR	Read Data Address Bus
RDCLK	Read Data Clock
RDEN	Read Port Enable
REGCE	Output Register Clock Enable
SBITERR	Single Bit Error Status
DBITERR	Double Bit Error Status
ECCPARITY	ECC Encoder Output Bus
SSR	Synchronous Set or Reset of Output Registers or Latches
WE	Byte-wide Write Enable
WRADDR	Write Data Address Bus
WRCLK	Write Data Clock
WREN	Write Port Enable

Notes:

1. Block RAM primitive port names can be different from the port function names.

DPRAM in our Lab9



- DPRAM : Dual Port RAM.It enables independent, simultaneous memory access from two buses with an easy processor arbitration required.
- It is implemented inside the file *ipbus_dpram.vhd*.
- You don't have to write code inside here, but you have to understand how it works, in order to interface it with the design logic. (Not in this lab but in the next).

DPRAM in our Lab9

```
dpram: ipbus_dpram
  generic map( ADDR_WIDTH => 4)
  port map(
          clk         => ipb_clk,
          rst         => rst_ipb,
          ipb_in      => ipbw(0),
          ipb_out     => ipbr(0),
          rclk        => ipb_clk,
          we          => we_s,
          d           => x"000000" & rxd_s,
          q           => open,
          addr        => addr_s
  );
```

```
type ram_array is array(2 ** ADDR_WIDTH - 1 downto 0) of std_logic_vector(31 downto 0);
shared variable ram: ram_array;
signal sel, rsel: integer; signal ack: std_logic;
begin
sel <= to_integer(unsigned(ipb_in.ipb_addr(addr_width-1 downto 0)));

process(clk)
begin
    if rising_edge(clk) then
        ipb_out.ipb_rdata <= ram(sel);
        if ipb_in.ipb_strobe='1' and ipb_in.ipb_write='1' then
            ram(sel) := ipb_in.ipb_vdata;
        end if;
        ack <= ipb_in.ipb_strobe and not ack;
    end if;
end process;

ipb_out.ipb_ack <= ack;
ipb_out.ipb_err <= '0';
```

Code to interface with the host. i.e. PC, Laptop ...

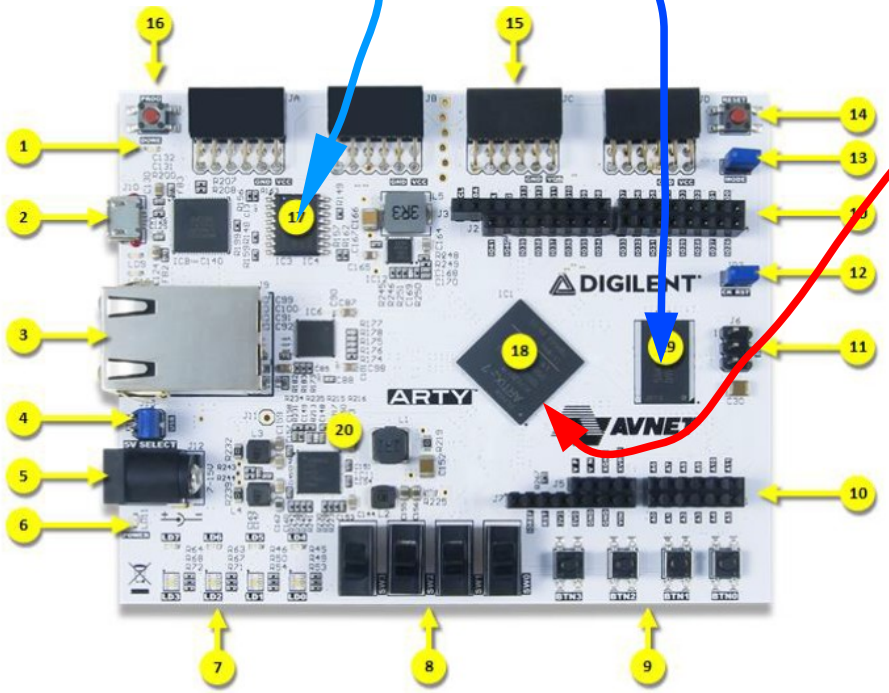
```
rsel <= to_integer(unsigned(addr));

process(rclk)
begin
    if rising_edge(rclk) then
        q <= ram(rsel);
        if we = '1' then
            ram(rsel) := d;
        end if;
    end if;
end process;
```

Code to interface with the design logic. i.e. the code to read the registers of the serial SPI flash memory

Memory on ARTY7 board

- Xilinx Artix-35T FPGA
  → ~ 40k FF
  → 1800 kbits of fast block RAM

- System
  → 256MB DDR3L with a 16-bit bus @ 667MHz
  → 16MB Quad-SPI Flash



Quad-SPI Flash chip

Micron Data-sheet

numonyx

N25Q128

128-Mbit 3 V, multiple I/O, 4-Kbyte subsector erase on boot sectors, XiP enabled, serial flash memory with 108 MHz SPI bus interface

Features

- SPI-compatible serial bus interface
- 108 MHz (maximum) clock frequency
- 2.7 V to 3.6 V single supply voltage
- Supports legacy SPI protocol and new Quad I/O or Dual I/O SPI protocol
  - Additional smart protections available upon customer request
- Electronic signature
  - JEDEC standard two-byte signature (BA18h)
  - Additional 2 Extended Device ID (EDID) bytes to identify device factory options

8 Memory organization

The memory is organized as:

- 16,777,216 bytes (8 bits each)
- 256 sectors (64 Kbytes each)
- In Bottom and Top versions: 8 bottom (top) 64 Kbytes boot sectors with 16 subsectors (4 Kbytes) and 248 standard 64 KB sectors
- 65,536 pages (256 bytes each)
- 64 OTP bytes located outside the main memory array

- $16777216 = 2^{24}$;
- Therefore the address of each 8-bit register is a 24-bit address.

Quad-SPI Flash access

To read the memory content in SPI protocol different instructions are available: READ, Fast Read, Dual Output Fast Read, Dual Input Output Fast Read, Quad Output Fast Read and Quad Input Output Fast read, allowing the application to choose an instruction to send addresses and receive data by one, two or four data lines.

Table 14. Instruction set: extended SPI protocol (page 1 of 2)

Instruction	Description	One-byte Instruction Code (BIN)	One-byte Instruction Code (HEX)	Address bytes	Dummy clock cycle	Data bytes
RDID	Read Identification	1001 111x	9Eh / 9Fh	0	0	1 to 20
READ	Read Data Bytes	0000 0011	03h	3	0	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0Bh	3	8 (1)	1 to ∞
DOFR	Dual Output Fast Read	0011 1011	3Bh	3	8 (1)	1 to ∞
DIOFR	Dual Input/Output Fast Read	1011 1011	BB	3	8 (1)	1 to ∞
QOFR	Quad Output Fast Read	0110 1011	6Bh	3	8 (1)	1 to ∞
QIOFR	Quad Input/Output Fast Read	1110 1011	EBh	3	10 (1	1 to ∞
ROTP	Read OTP (Read of OTP area)	0100 1011	4Bh	3	8 (1)	1 to 65

Quad-SPI Flash read (by FPGA)

- The device is selected by driving the CS low.
- The instruction code for the READ instruction (00000011_2) is sent. Each bit is latched-in (evaluated) on the rising edge of the clock.
- The 3 bytes address (23,22,...,1,0) are then sent. Each bit is latched-in (evaluated) on the rising edge of the clock.
- Then the memory contents, at that address, is shifted out on the MISO, each bit is sent, on the falling edge of Serial Clock (C). But evaluated by the Master on the following rising-edge.
- The READ instruction is terminated by driving the CS high.

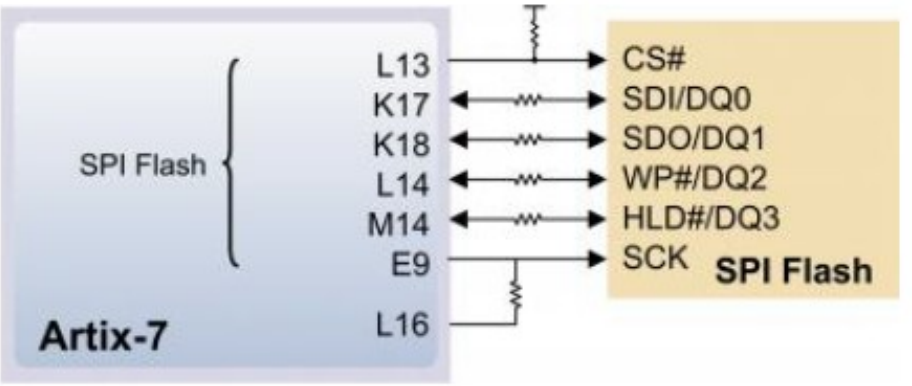
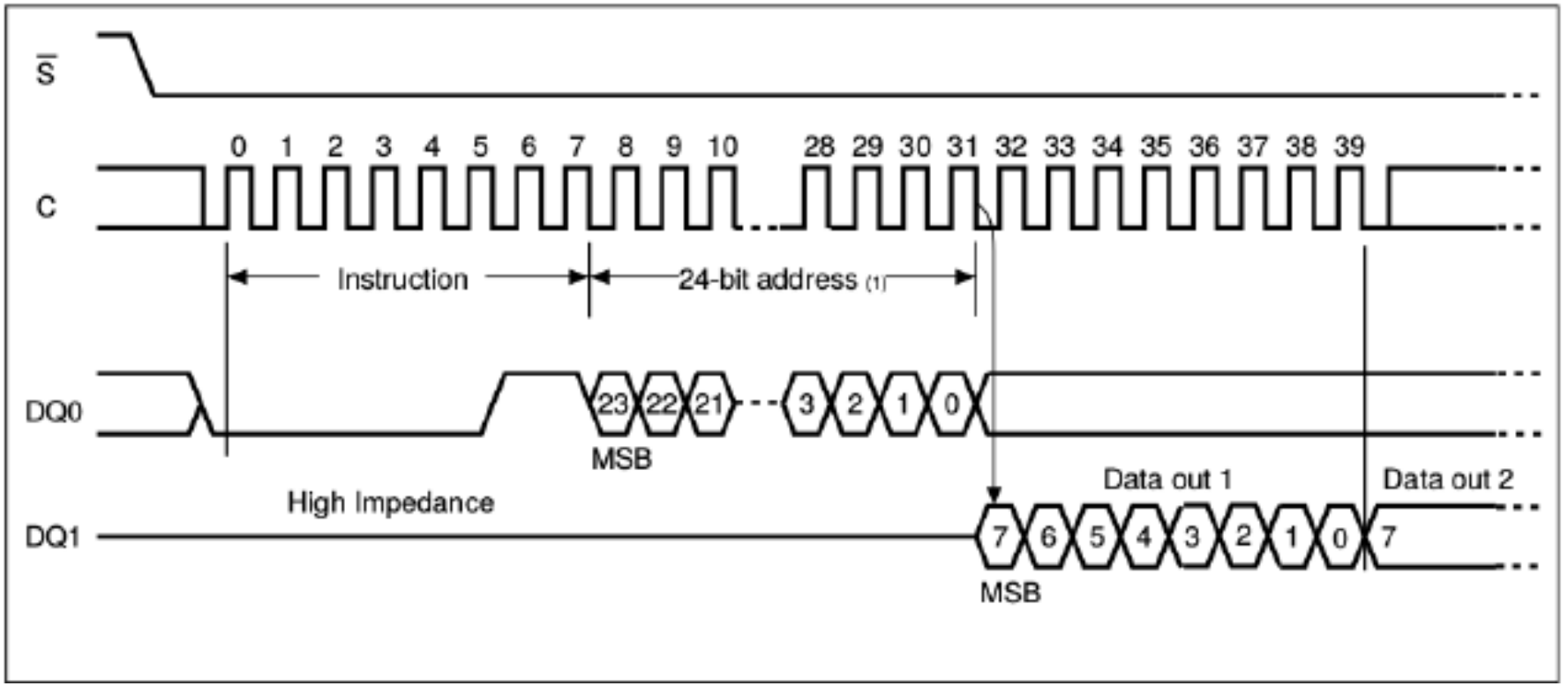


Figure 11. Read Data Bytes instruction and data-out sequence



Note about FLASH in ARTY-7

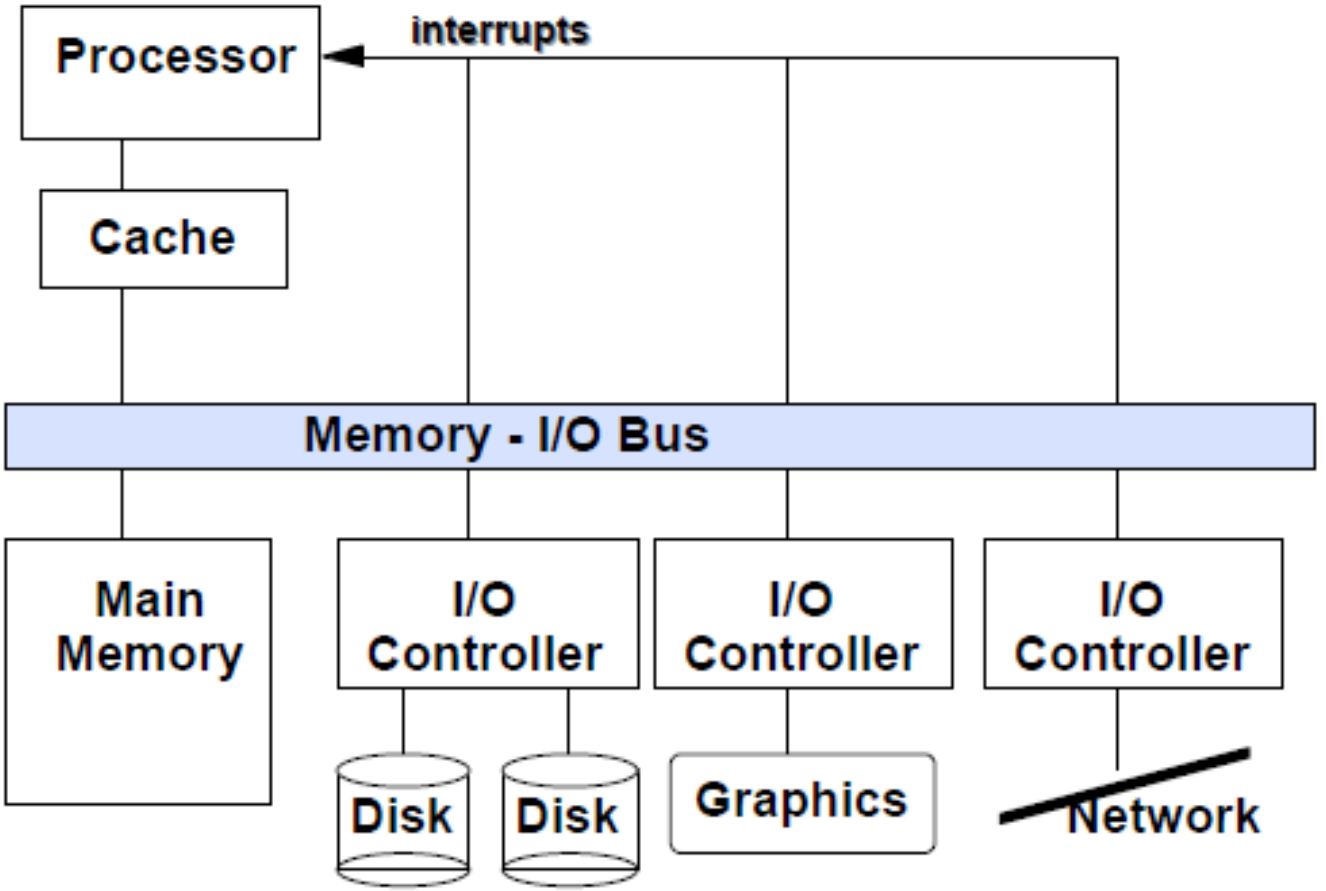
FPGA configuration files can be written to the Quad-SPI Flash (Micron part number N25Q128A13ESF40),
and setting the mode jumper will cause the FPGA to automatically read a configuration from this device at power on.

An Artix-7 35T configuration file requires 17,536,096 bits of memory, leaving about 87% of the flash device (or ~14MB)
available for user data. A common use for this extra memory is to store MicroBlaze programs
too big to fit in the onboard Block memory (typically 128 KB). These programs are then loaded and executed using a smaller
bootloader program that can fit in the block memory. It is possible to automatically generate this bootloader, roll it into a single
file (called an .mcs file) that also contains the bitstream and your custom MicroBlaze application, and program this file into SP
Flash using Xilinx SDK and Vivado.
Xilinx Answer Record 63605 explains how to do this.

The contents of the memory can be manipulated by issuing certain commands on the SPI bus.
All signals in the SPI bus are general-purpose user I/O pins after FPGA configuration.
On other boards, SCK is an exception because it remains a dedicated pin even after configuration,
however, on Arty the SCK signal is routed to an additional general purpose pin that can be accessed
after configuration. This allows access to this pin without having to instantiate the special FPGA primitive
called STARTUPE2

Part II
Buses and I/O interconnections

Buses – example → PC

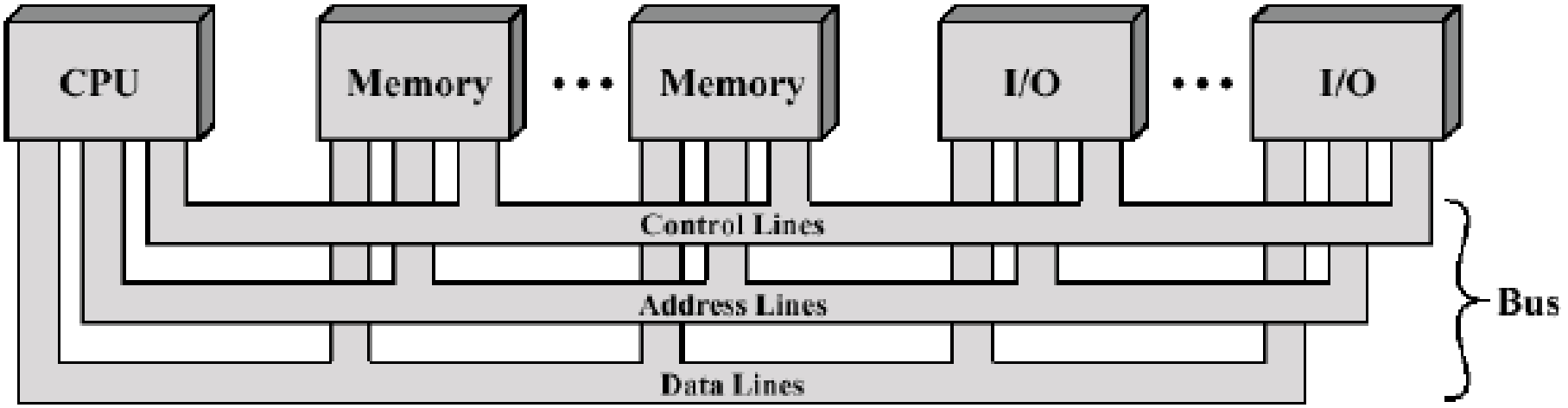


Time(workload) = Time(CPU) + Time(I/O) - Time(Overlap)

Buses – example → PC

System interconnection structures may support these transfers:

- **Memory to processor**: the processor reads instructions and data from memory
- **Processor to memory**: the processor writes data to memory
- **I/O to processor**: the processor reads data from I/O device
- **Processor to I/O**: the processor writes data to I/O device
- **I/O to or from memory**: I/O module allowed to exchange data directly with memory without going through the processor - Direct Memory Access (DMA)

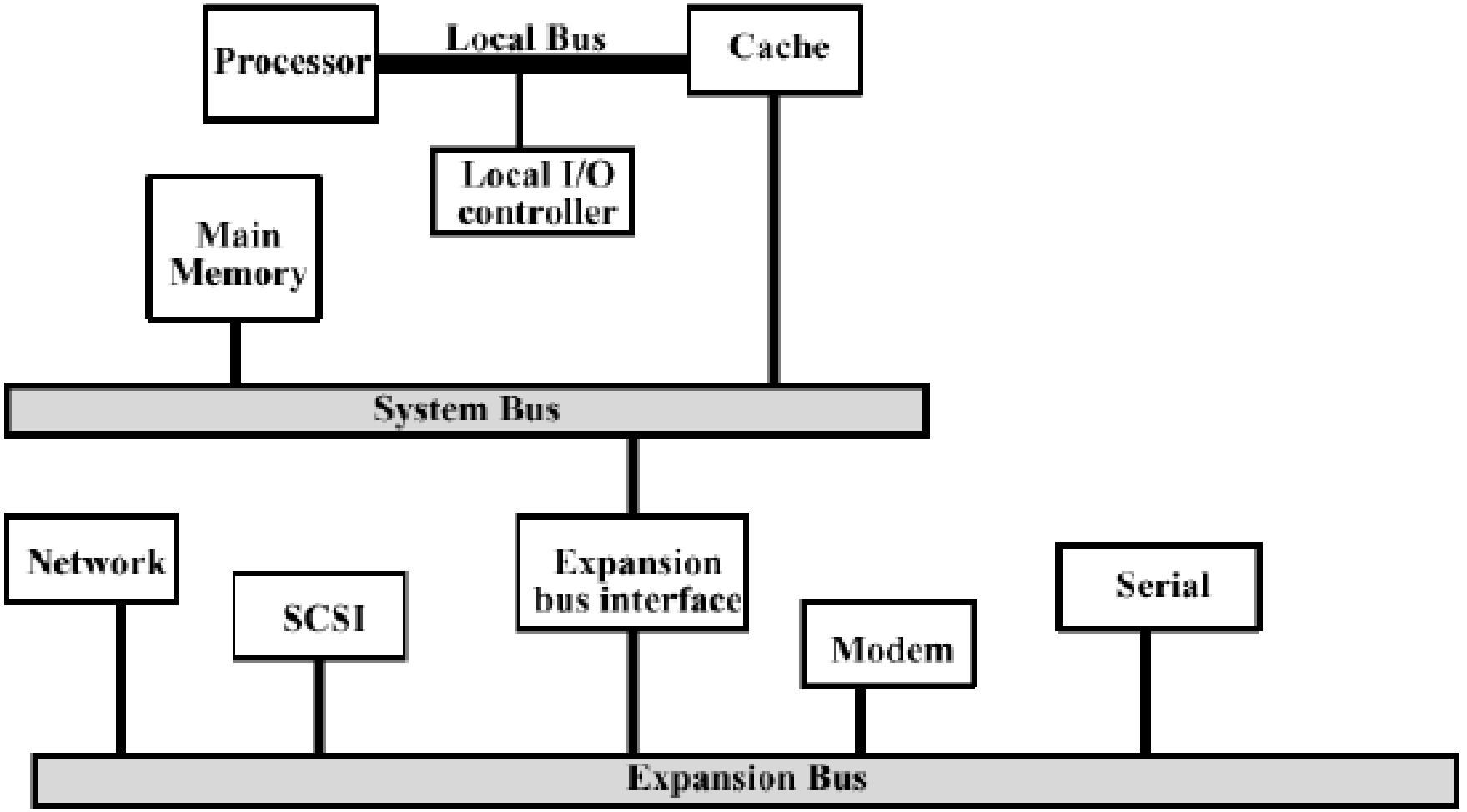


Reference: Computer Organization & Architecture (5th Ed), William Stallings

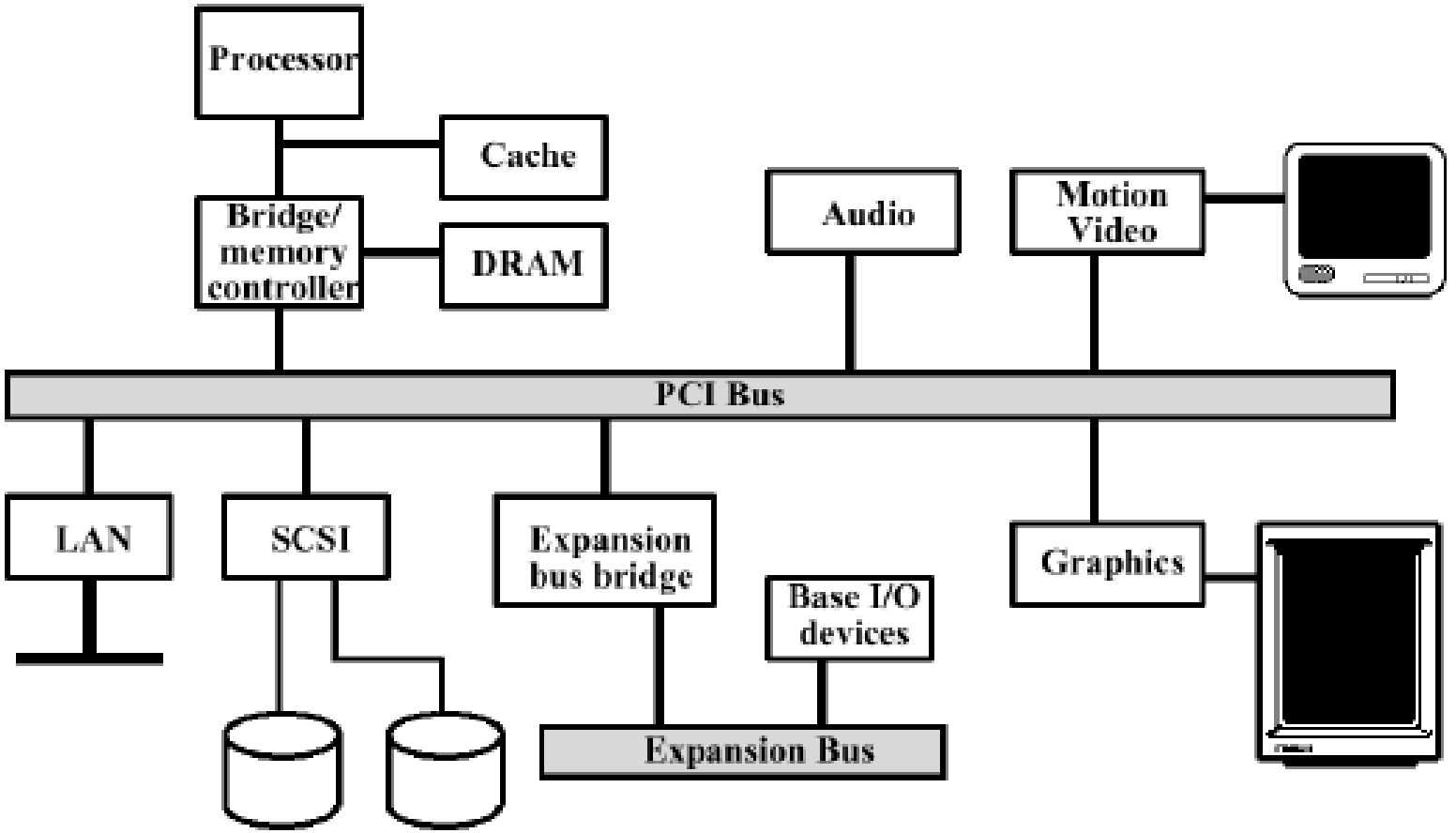
Signals on a bus

- **Address lines**
- **Data lines**
- **Control lines:**
  - **Memory write**: data on the bus written into the addressed location
  - **Memory read**: data from the addressed location placed on the bus
  - **I/O write**: data on the bus output to the addressed I/O port
  - **I/O read**: data from the addressed I/O port placed on the bus
  - **Bus REQ**: indicates a module needs to gain control of the bus
  - **Bus GRANT**: indicates that requesting module has been granted control of the bus
  - **Interrupt REQ**: indicates that an interrupt is pending
  - **Interrupt ACK**: Acknowledges that pending interrupt has been recognized
  - **Reset**: Initializes everything connected to the bus
  - **Clock**: on a synchronous bus, everything is synchronized to this signal

Traditional PC bus architecture

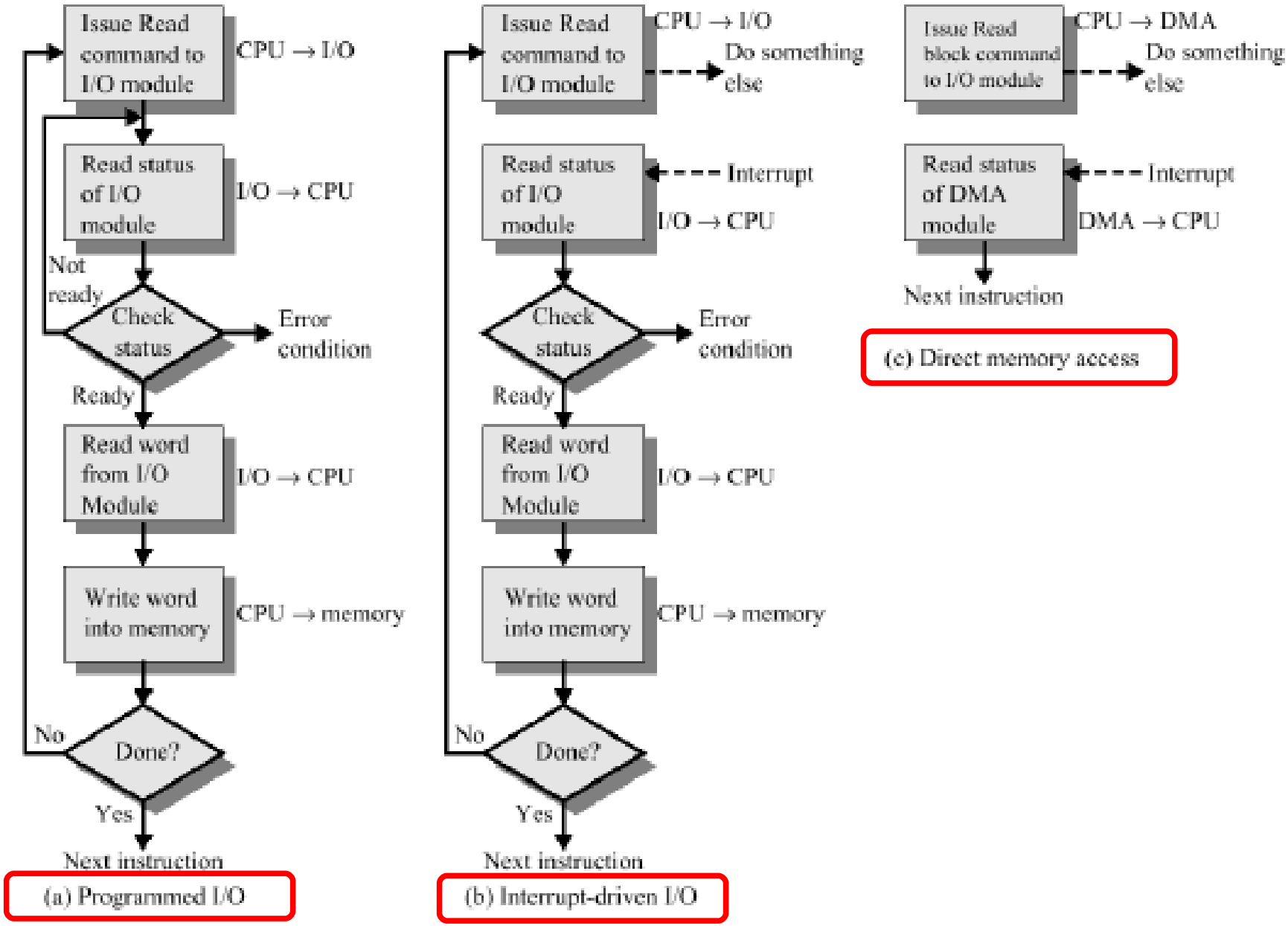


High Performance PC bus architecture



- Separate out fast local bus (PCI bus) and slower I/O expansion bus (ISA bus)

Techniques for inputting a block of data



Techniques for inputting a block of data → polling vs interrupt

- Both are methods to notify processor that I/O device needs attention
- **Polling**
  - simple, but slow
  - processor check status of I/O device regularly to see if it needs attention
  - similar to checking a telephone without bells!
- **Interrupt**
  - fast, but more complicated
  - processor is notified by I/O device (interrupted) when device needs attention
  - similar to a telephone with bells

Direct Memory Access

- Interrupt & programmed I/O requires processor to transfer data between memory and I/O module.
- Processor is tied up in performing the transfer by executing a number of instructions - SLOW!!!
- I/O module can contain autonomous hardware to perform transfer – DMA
- DMA is stealing bus cycles from processor - a technique known as cycle stealing

Direct Memory Access

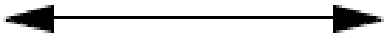
- DMA is initiated by a processor. The following information must be sent by the processor to the DMA module:
  - Tell DMA module whether a read or write is required
  - Send to the DMA module the address of the I/O device involved in the transfer
  - The starting location in memory to read from or write to - DMA module will store this locally in the address register
  - The number of word to read or write - DMA module will store this in the data count register
- Once DMA is initiated, the processor can continue with other work.
- Since processor has cache memory, it can work concurrently with transfer between I/O device and memory. (But there is extra work needed to ensure *cache coherency*)

Interconnection Buses

- **Interconnect = glue that interfaces computer system components**
- **High speed hardware interfaces + logical protocols**
- **Networks, channels, backplanes**

	Network	Channel	Backplane
Distance	>1000 m	10 - 100 m	1 m
Bandwidth	10 - 100 Mb/s	40 - 1000 Mb/s	320 - 1000+ Mb/s ++
Latency	high (>ms)	medium	low (<μs)
Reliability	low Extensive CRC	medium Byte Parity	high Byte Parity

message-based
narrow pathways
distributed arb



memory-mapped
wide pathways
centralized arb

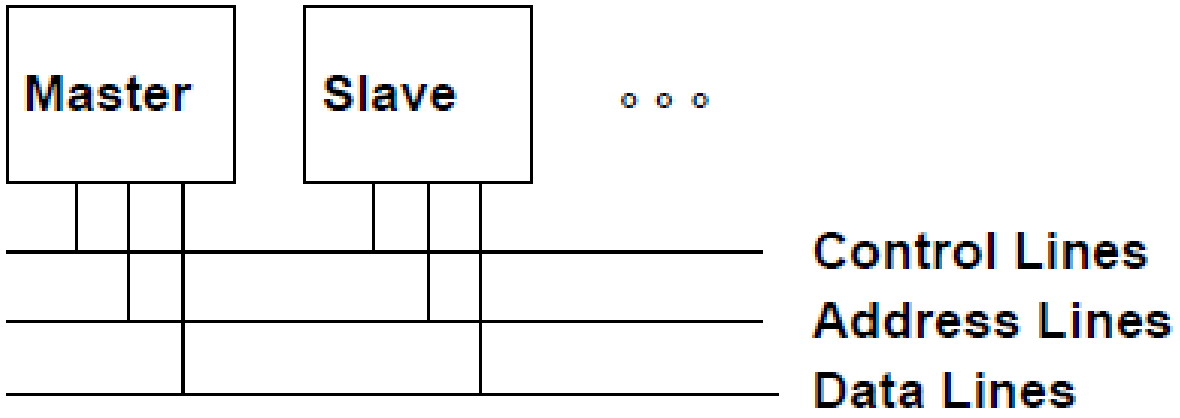
Bus based interconneciton

- **Bus: a shared communication link between subsystems**
  - Low cost: a single set of wires is shared multiple ways
  - Versatility: Easy to add new devices & peripherals may even be ported between computers using common bus
- **Disadvantage**
  - A communication bottleneck, possibly limiting the maximum I/O throughput
- **Bus speed is limited by physical factors**
  - the bus length
  - the number of devices (and, hence, bus loading).
  - these physical limits prevent arbitrary bus speedup.
- **Two generic types of busses:**
  - I/O busses: lengthy, many types of devices connected, wide range in the data bandwidth), and follow a bus standard (sometimes called a *channel*)
  - CPU–memory buses: high speed, matched to the memory system to maximize memory–CPU bandwidth, single device (sometimes called a *backplane*)
  - To lower costs, low cost (older) systems combine together
- **Bus transaction**
  - Sending address & receiving or sending data

Backplane Buses (old…) examples

Metric	VME	FutureBus	MultiBus II	SCSI-I
Bus Width (signals)	128	96	96	25
Address/Data Multiplexed?	No	Yes	Yes	na
Data Width	16 - 32	32	32	8
Xfer Size	Single/Multiple	Single/Multiple	Single/Multiple	Single/Multiple
# of Bus Masters	Multiple	Multiple	Multiple	Multiple
Split Transactions	No	Optional	Optional	Optional
Clocking	Async	Async	Sync	Either
Bandwidth, Single Word (0 ns mem)	25	37	20	5, 1.5
Bandwidth, Single Word (150 ns mem)	12.9	15.5	10	5, 1.5
Bandwidth Multiple Word (0 ns mem)	27.9	95.2	40	5, 1.5
Bandwidth Multiple Word (150 ns mem)	13.6	20.8	13.3	5, 1.5
Max # of devices	21	20	21	7
Max Bus Length	.5 m	.5 m	.5 m	25 m
Standard	IEEE 1014	IEEE 896	ANSI/IEEE 1296	ANSI X3.131

Bus protocols



Multibus: 20 address, 16 data, 5 control, 50ns Pause

Bus Master: has ability to control the bus, initiates transaction

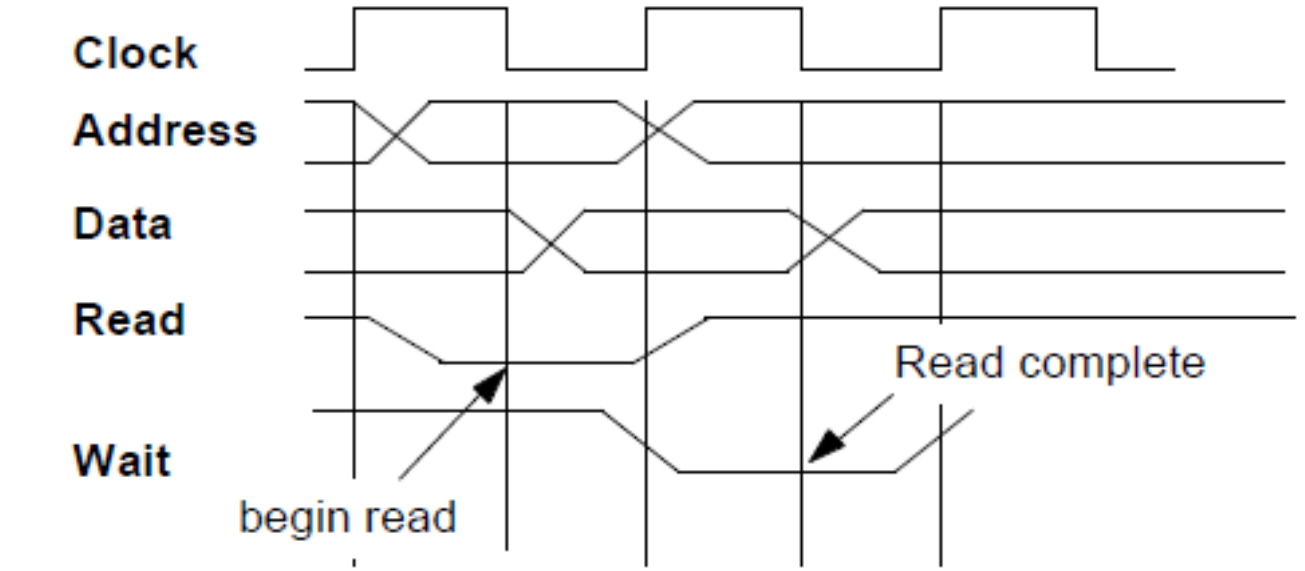
Bus Slave: module activated by the transaction

Bus Communication Protocol: specification of sequence of events and timing requirements in transferring information.

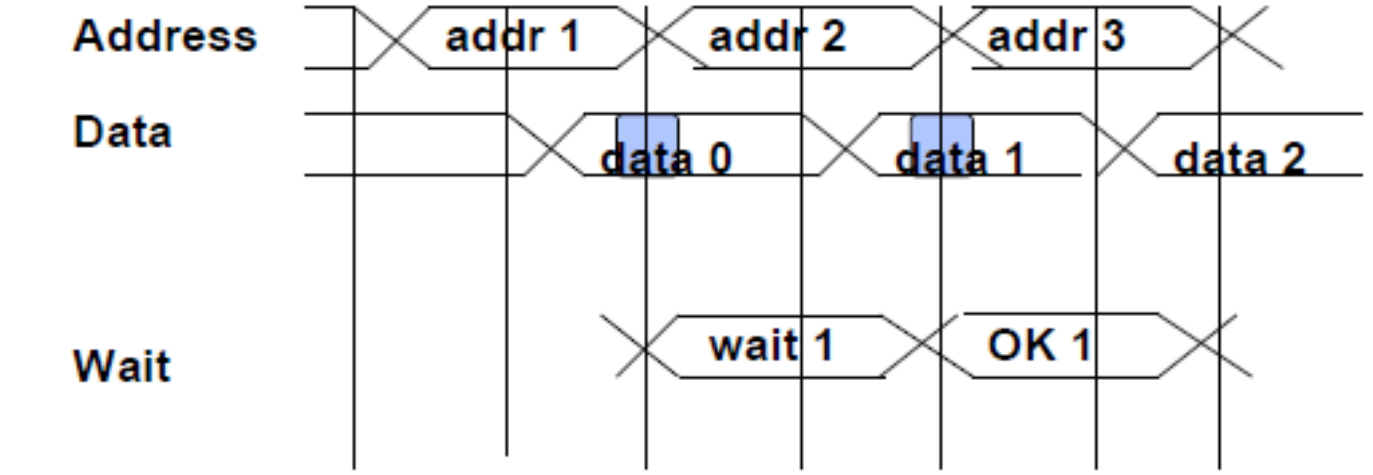
Asynchronous Bus Transfers: control lines (req., ack.) serve to orchestrate sequencing

Synchronous Bus Transfers: sequence relative to common clock

Synchrnonous Bus protocols

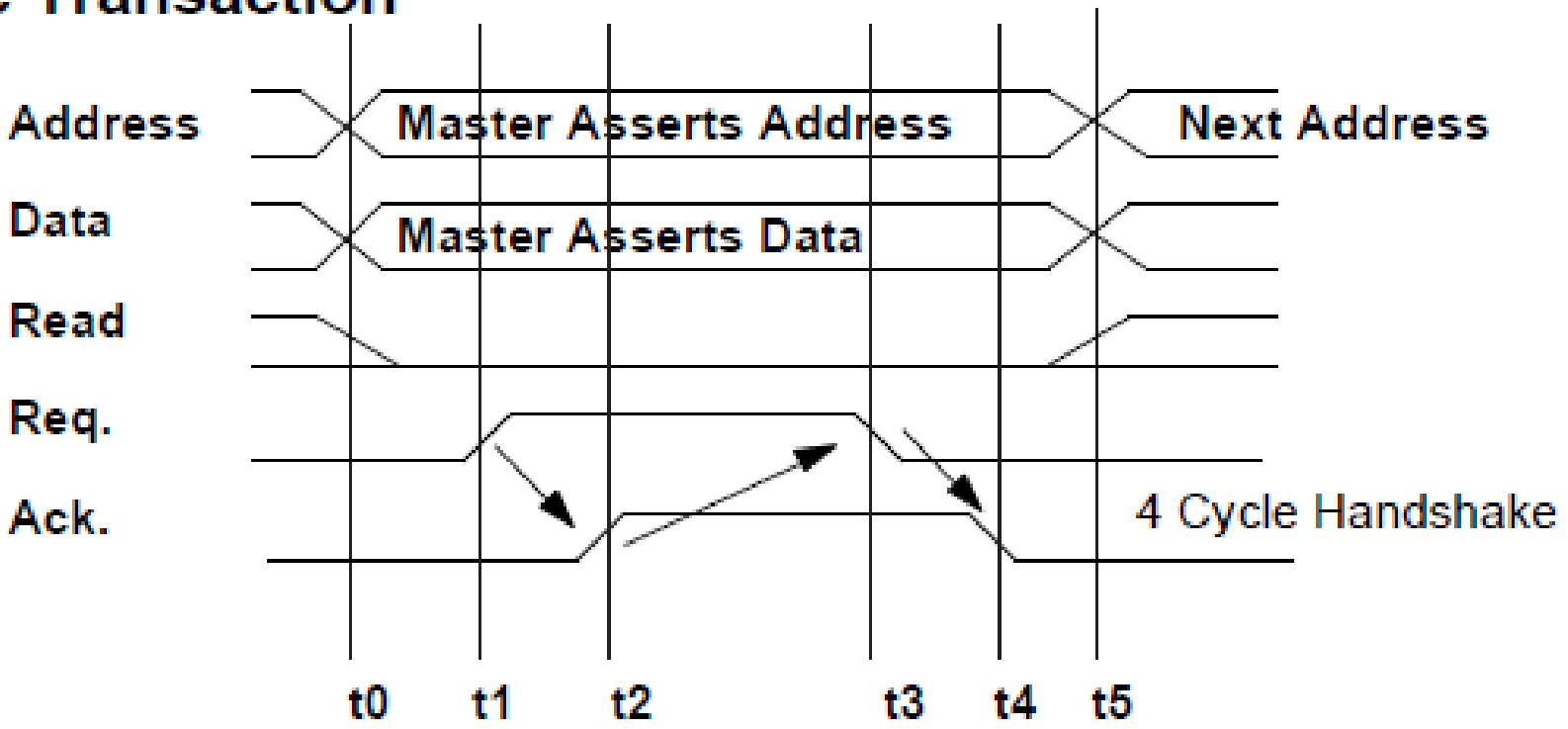


Pipelined/Split transaction Bus Protocol



Asynchrnonous handshake

Write Transaction



t0 : Master has obtained control and asserts address, direction, data
Waits a specified amount of time for slaves to decode target\

t1: Master asserts request line

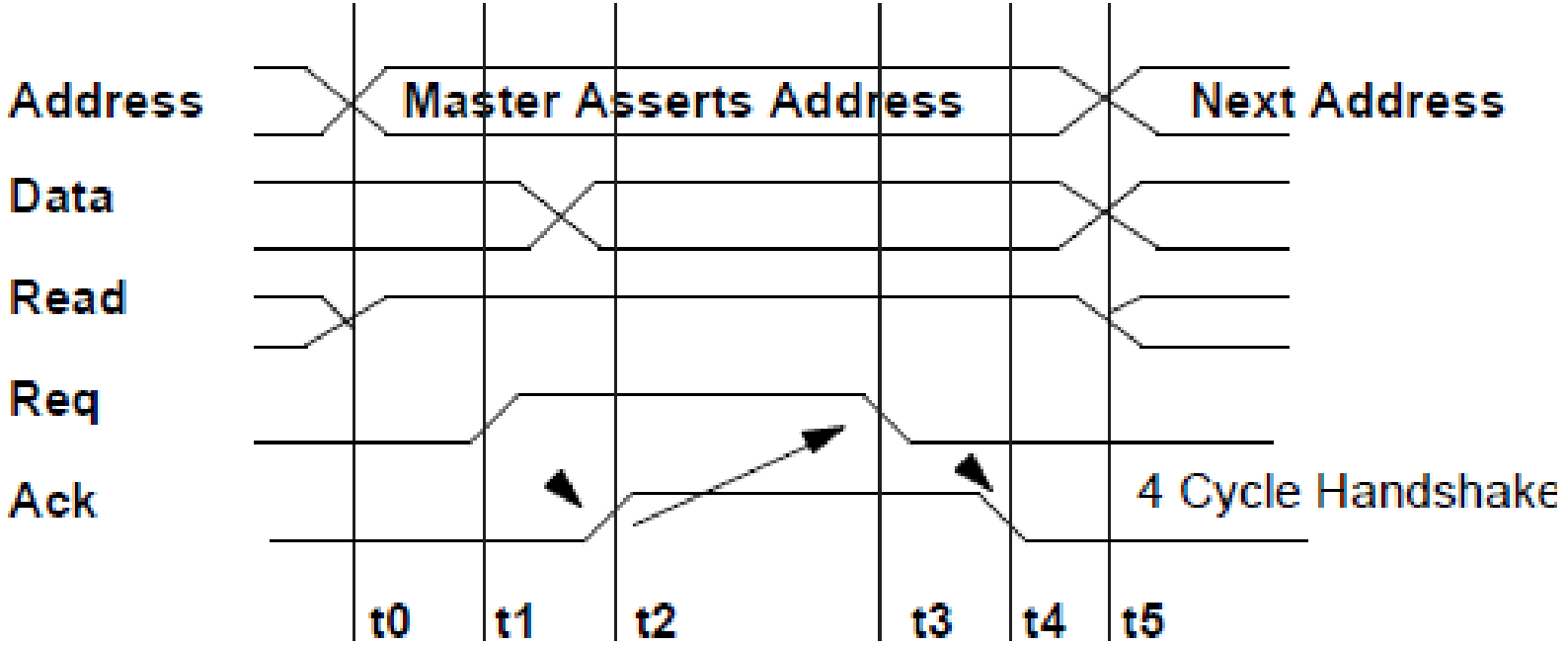
t2: Slave asserts ack, indicating data received

t3: Master releases req

t4: Slave releases ack

Asynchrnonous handshake

Read transaction



t0 : Master has obtained control and asserts address, direction, data
Waits a specified amount of time for slaves to decode target\

t1: Master asserts request line

t2: Slave asserts ack, indicating ready to transmit data

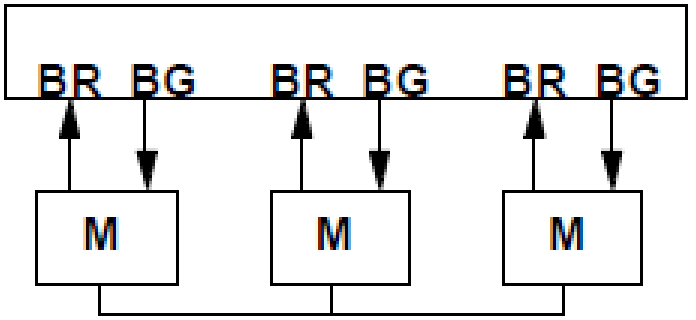
t3: Master releases req, data received

t4: Slave releases ack

Time Multiplexed Bus: address and data share lines

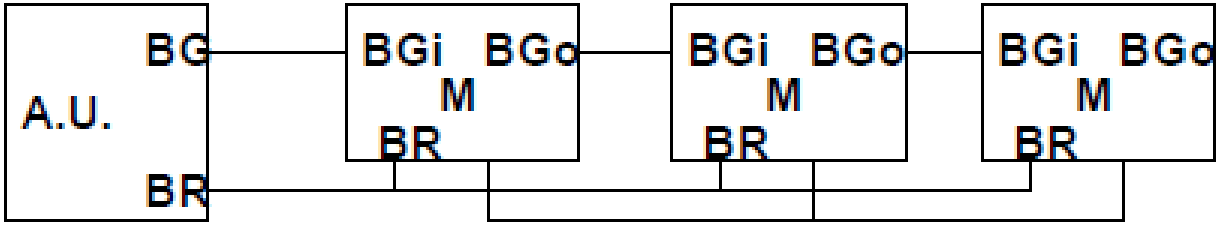
Bus Arbitration

Parallel (Centralized) Arbitration

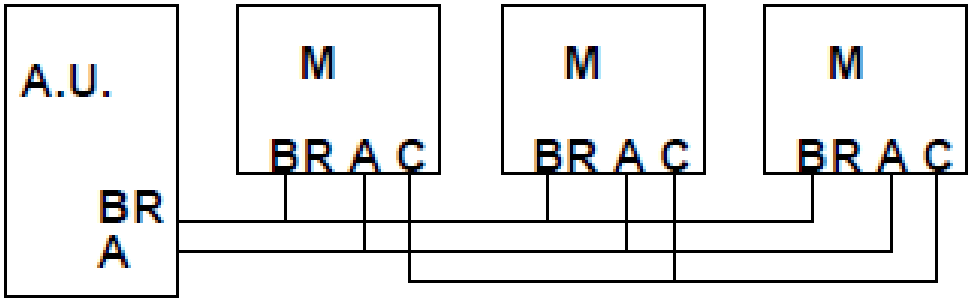


Bus Request
Bus Grant

Serial Arbitration (daisy chaining)



Polling



Bus Options

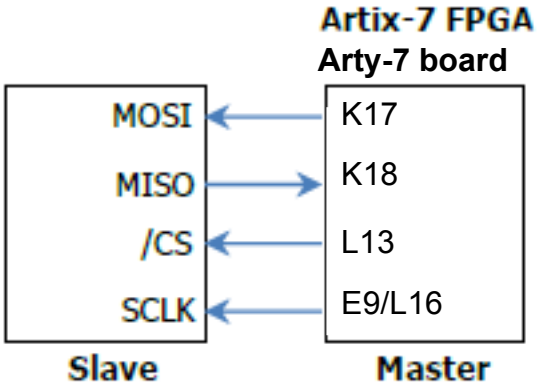
Option	*High performance*	*Low cost*
Bus width	Separate address & data lines	Multiplex address & data lines
Data width	Wider is faster (e.g., 32 bits)	Narrower is cheaper (e.g., 8 bits)
Transfer size	Multiple words has less bus overhead	Single-word transfer is simpler
Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
Split transaction?	Yes—separate Request and Reply packets gets higher bandwidth (needs multiple masters)	No—continuous connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

Data transfer w/ FPGA…
… a few Examples

Serial Sync. Communication with SPI

SPI INTERFACE

- Simple 4-wired synchronous (clock is transmitted) serial interface.
- SPI logic signals:
  - ✓ SCLK: Serial clock. Generated by Master.
  - ✓ MOSI: Master Output, Slave Input. Generated by Master.
  - ✓ MISO: Master Input, Slave Output. Generated by Slave.
  - ✓ /CS: Chip select (or Slave Select). Generated by Master.
- Messages are supported that are multiple of 8 bits.
- Clock polarity (CPOL) and Phase (CPHA):



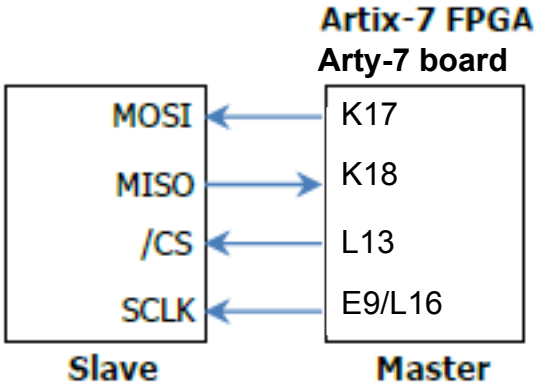
- SPI (Serial Peripheral Interface) :
  - it is serial: the data is transmitted one bit at a time;
  - it is synchronous: the transmission of the data is imposed by the clock;
  - it has an architecture master/slave(s);
  - it is used for the communications between devices.
- It is a basic serial protocol (not the most).
- It is mostly used for the communication between μC or FPGA and ICs like: A/D, D/A converters, sensors, memories ...

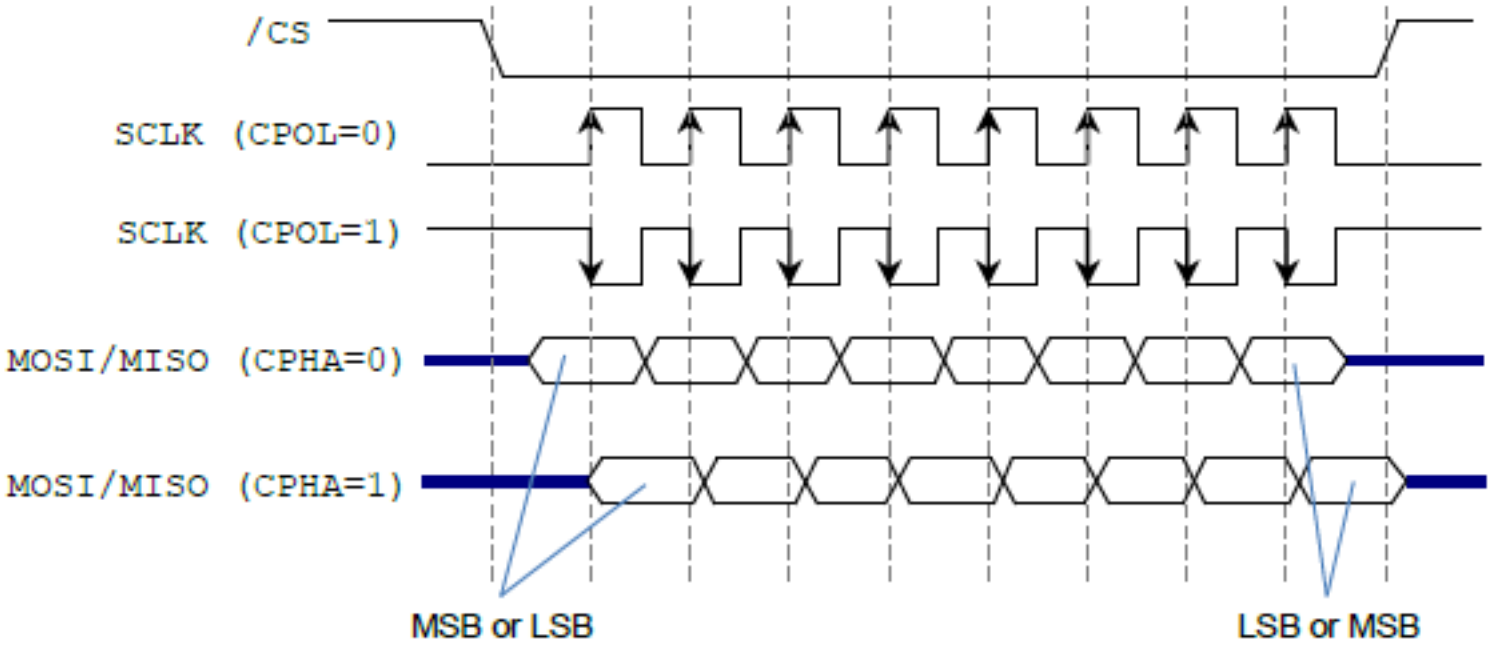
- It is commonly used for short distance communications within embedded systems. Microcontrollers and FPGA designs use SPI to communicate with internal/external peripherals. Large variety of SPI-capable peripherals available: sensors (e.g.: temperature, pressure), ADCs, DACs, touchscreens, memories, LCDs, SD cards.

Serial Sync. Communication with SPI

SPI INTERFACE

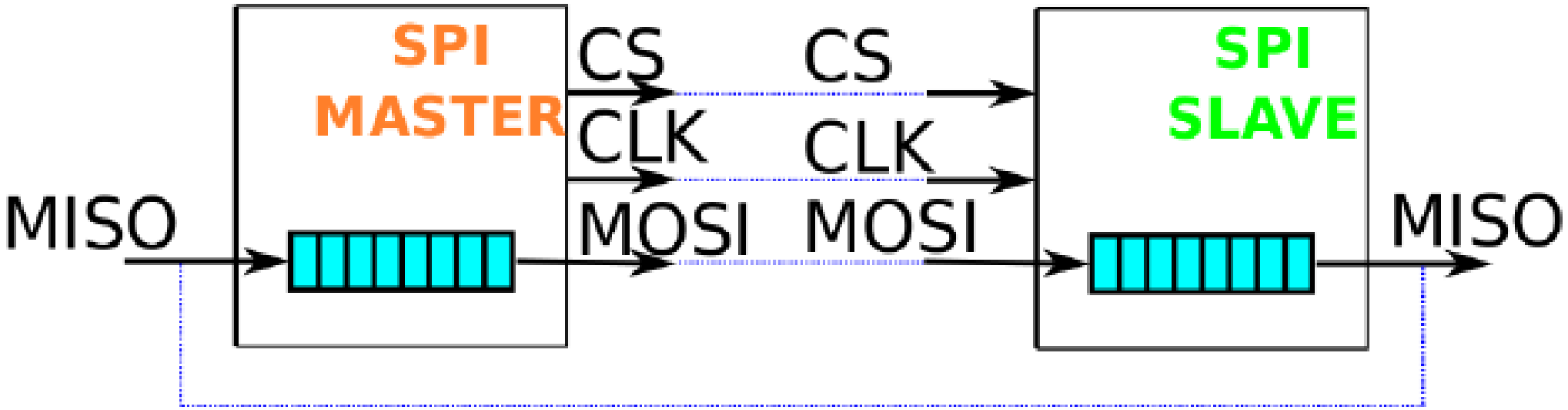
- Simple 4-wired synchronous (clock is transmitted) serial interface.
- SPI logic signals:
  - ✓ SCLK: Serial clock. Generated by Master.
  - ✓ MOSI: Master Output, Slave Input. Generated by Master.
  - ✓ MISO: Master Input, Slave Output. Generated by Slave.
  - ✓ /CS: Chip select (or Slave Select). Generated by Master.
- Messages are supported that are multiple of 8 bits.
- Clock polarity (CPOL) and Phase (CPHA):





- It is commonly used for short distance communications within embedded systems. Microcontrollers and FPGA designs use SPI to communicate with internal/external peripherals. Large variety of SPI-capable peripherals available: sensors (e.g.: temperature, pressure), ADCs, DACs, touchscreens, memories, LCDs, SD cards.

Serial Sync. Communication with SPI



- The data are exchanged between a master and a slave.
- Essentially each device has inside it a shift register with the data. The data transmission "exchanges" the data between the shift registers.
- The Master addresses the Slave and it manages the data transmission with the clock (on the **rising edge**).

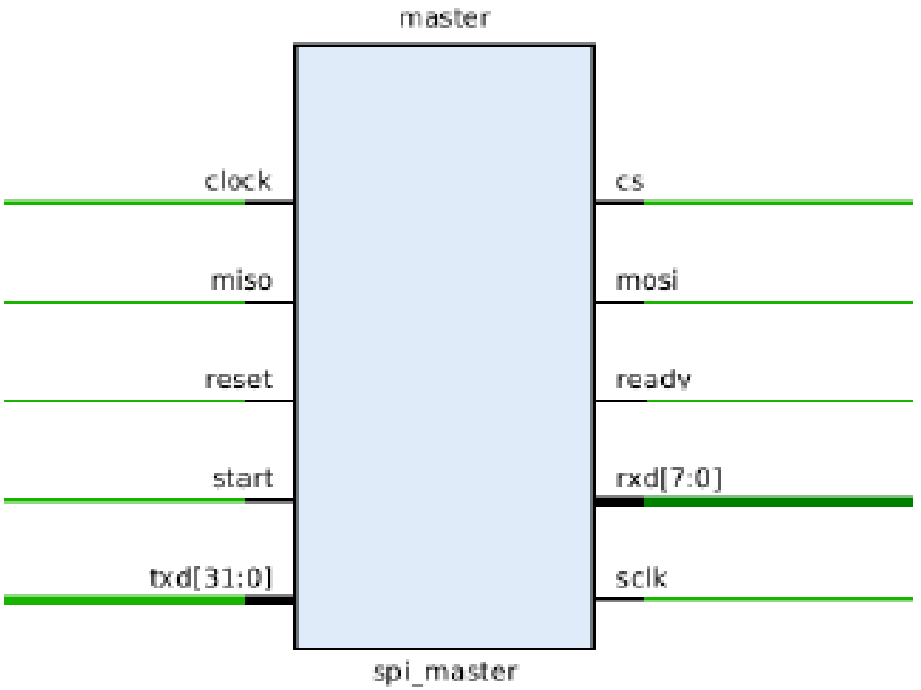
Serial Sync. Communication with SPI

SPI signals (from the slave side):

- **CS** or **SS**: Chip (Slave) Select. When this input signal is high, the device is deselected.
- **CLK**: Clock. This input signal provides the timing for the serial interface. Instructions, addresses, or data present at the MOSI are latched on the rising edge of the clock.
- **MOSI**: Master Output Slave Input. This input signal is used to transfer data serially into the device. It receives instructions, addresses, and the data to be programmed.
- **MISO**: Master Input Slave Output. This output signal is used to transfer data serially out of the device. Data are shifted out on the falling edge of the clock.

Lab. about SPI (aiming at communicating w/ FLASH)

- You have to write the VHDL code to implement an SPI master, in order to read the data stored in a 8-bit register of the serial flash memory.



Serial Sync. Communication with SPI example w/ accelerometer

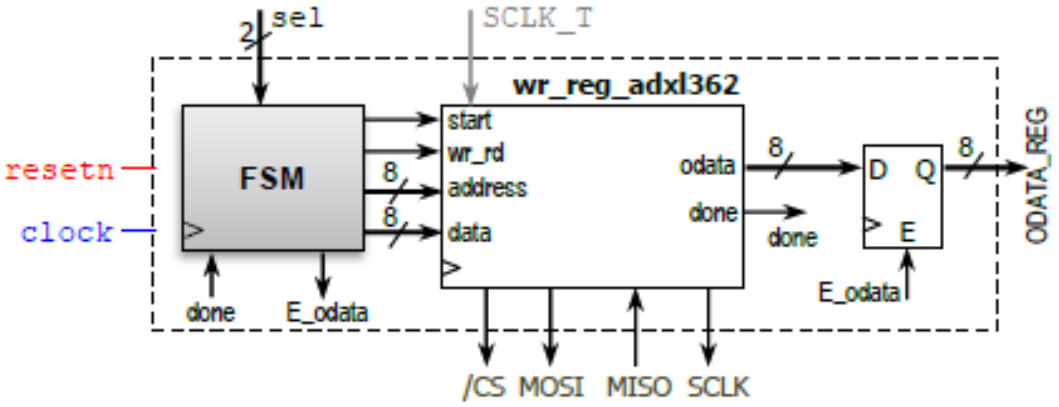
ACCELEROMETER ADXL362

- This 3-axis MEMS device operates as a SPI slave device. We read/write data via a register-based interface: we can write/read a byte or many bytes per bus transaction.
- ADXL362 parameters (range, resolution, ODR are selectable):
  - ✓ Range: ± 2g (default at reset), ± 4g, ± 8g.
  - ✓ Resolution: 1mg/LSB (default at reset), 2 mg/LSB, 4 mg/LSB
  - ✓ Output data rate (ODR): 12.5 – 400 Hz. Default at reset: 100 Hz.
  - ✓ Output resolution: 12 bits. Representation: signed.
- CPOL = 0, CPHA = 0. Many SPI devices work very similarly, although we need to comply with specific timing parameters.

Accelerometer: Basic Controller

- **Operation:** We first configure the appropriate ADXL362 registers and then proceed to read 8-bit registers. A simple operation mode is listed here. Refer to the ADXL362 datasheet for a complete list of registers and operation modes.
  - ✓ Reset the ADXL362. Write `0x52` on SOFT_RESET (`0x1F`) register.
  - ✓ Activate measurement mode. Write `0x02` on POWER_CTL (`0x2D`) register.
  - ✓ Read any 8-bit register (one per bus transaction). See ADXL362 datasheet for complete list.

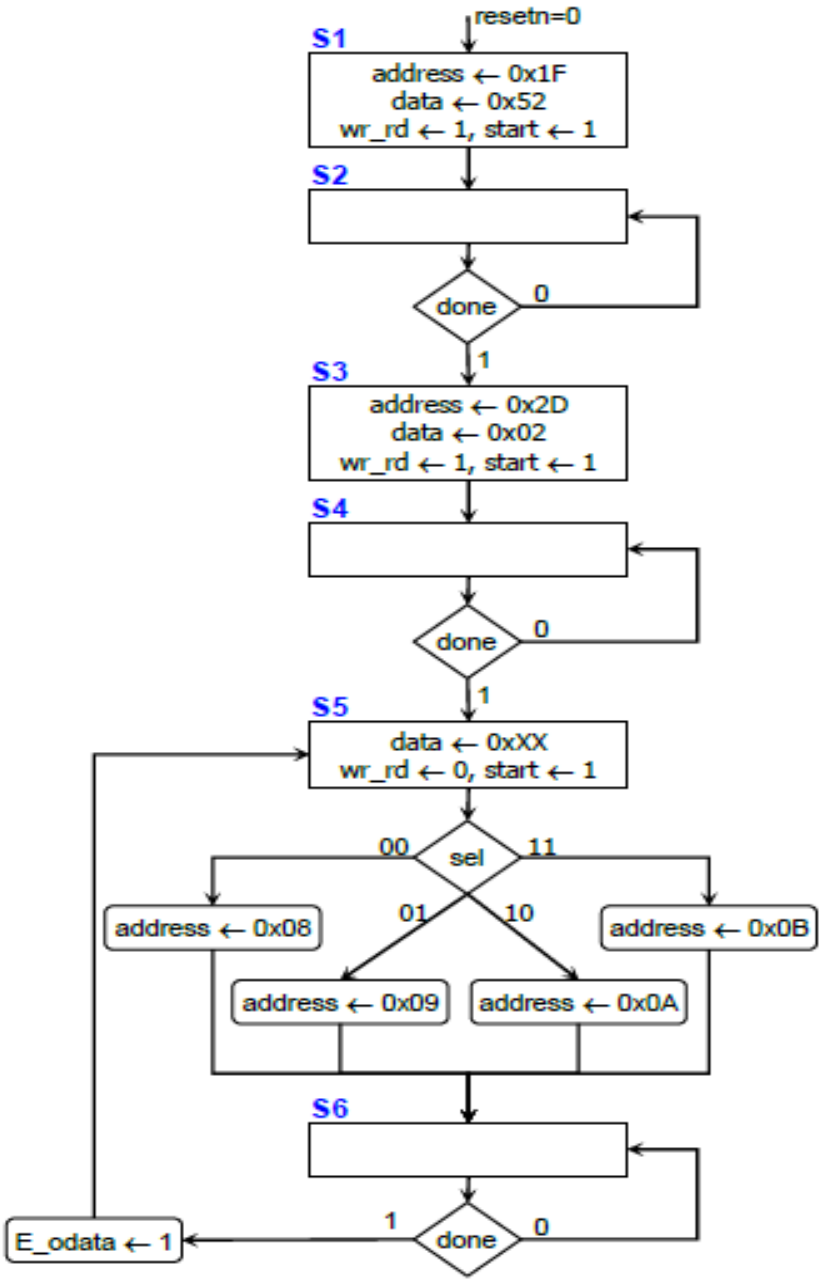
- The basic controller is depicted on the right. The block **wr_reg_adxl362** is the most important: it handles the SPI communication based on address, data and write/read decision. Asserting the *start* signal initiates a transaction. When the operation is completed, the signal *done* is asserted for one clock cycle. If reading data, it appears on *odata*. A new transaction can be started on the next cycle after $done = 1$.



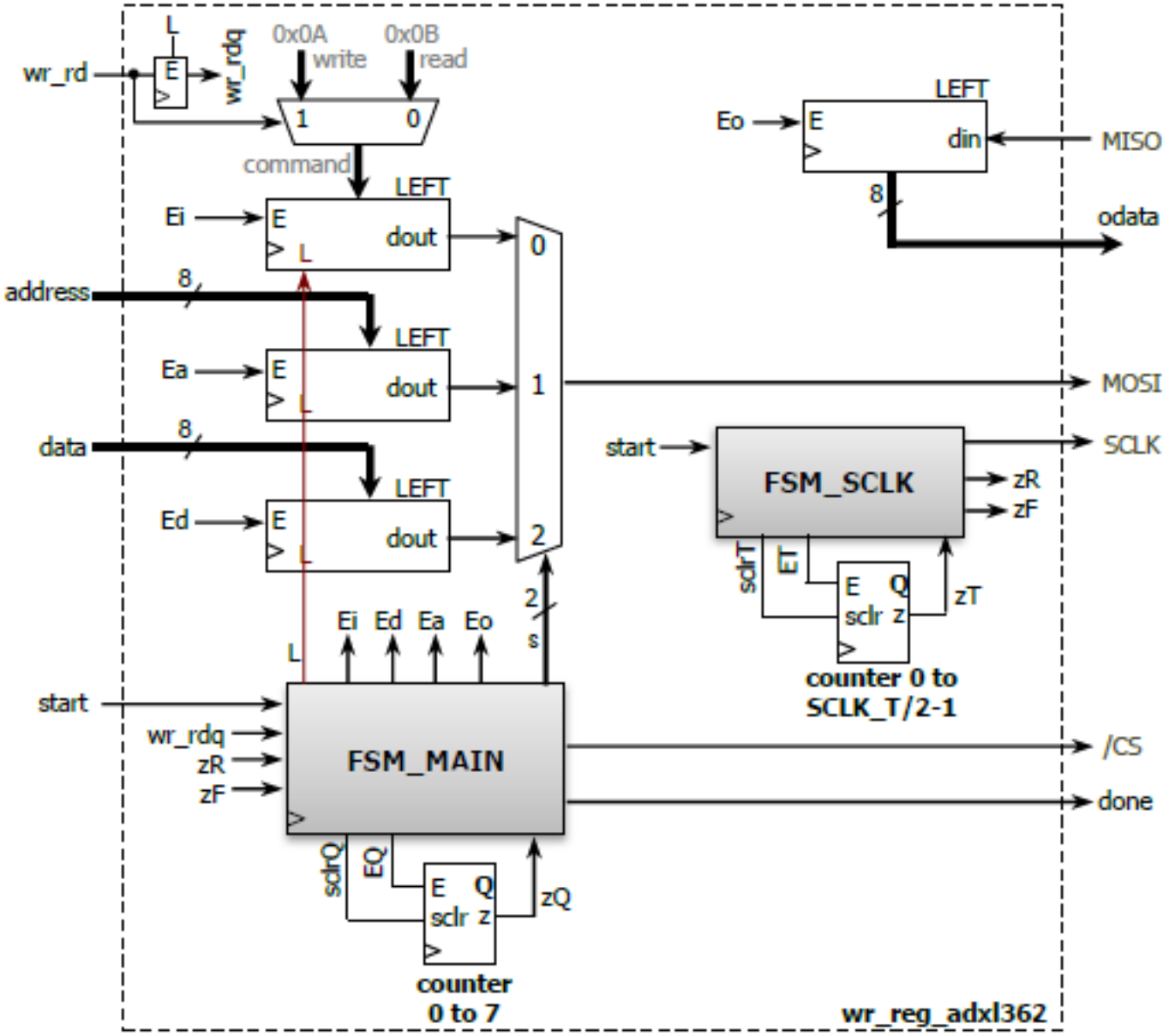
Blocks FSM and wr_reg description

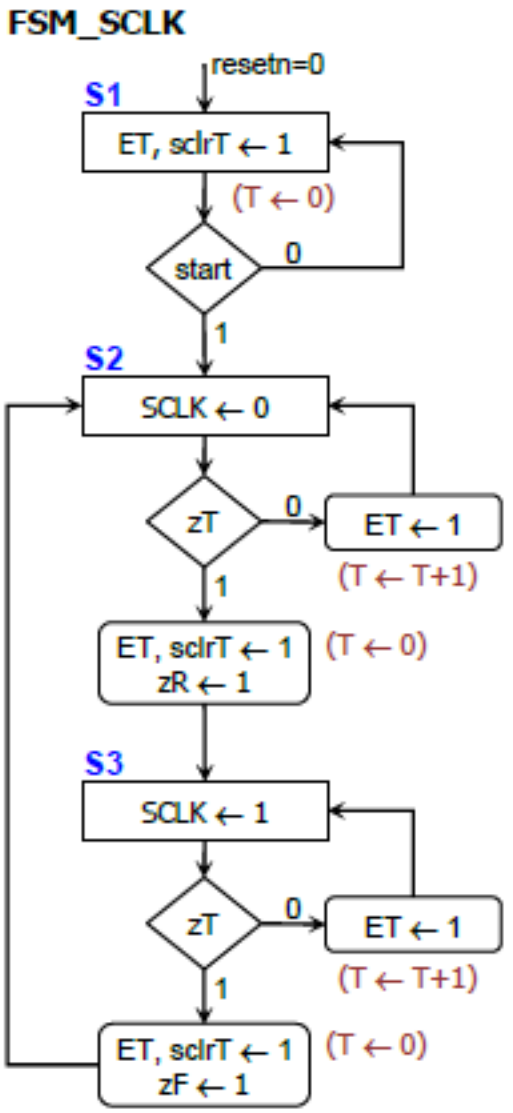
- **FSM**: It issues commands to configure the 2 ADXL362 registers and then read (cyclically) from one of four 8-bit ADXL362 registers (selected by the `sel` input). Data is fetched on the output register. Here, we can read the low-precision 8-bit X, Y, Z measurements (`0x08`, `0x09`, `0x0A`) and the Status Register (`0x0B`).

- **wr_reg_adxl362:** This circuit handles the SPI communication with the ADXL362. The user provides address, data, and read/write. Then, a read/write SPI transaction is executed. At every transaction, we write or retrieve 8 bits of data. When writing to the ADXL362, 3 bytes are transmitted: `|command|address|data|`. When reading from the ADXL362, 2 bytes are transmitted `|command|address|`, and 1 byte is read (data) and placed on `odata`.
  - ✓ Circuit Design: It involves implementing the SPI protocol and complying with the ADXL362 timing parameters (see datasheet):
    - C_{SS} (/CS Setup Time): 100 ns
    - t_{CSH} (/CS Hold Time): 20 ns
    - t_{CSD} (/CS Disable Time): 20 ns
    - t_{SU} (Data Setup Time) = t_{HD} (Data Hold Time): 20 ns
    - f_{SCLK}: 2.4 (only when using FIFO) – 8000 KHz.
    - t_{HIGH} (SCLK High Time) = t_{LOW} (SCLK Low Time) = 50 ns. Note that these times only constrain the duty cycle when using large frequencies. The maximum frequency is 8 MHz.
  - ✓ SCLK: not defined by the standard (usually a few MHz). This is specified by the Slave Device (ADXL362: $f_{SCLK} \leq 8000$ KHz).
  - ✓ This design uses a free running SCLK. To comply with the timing parameters: $T_{SCLK}-(t_{CSD}+t_{CSH}) \geq C_{SS} \rightarrow T_{SCLK} \geq 280$ ns ($f_{SCLK} \leq 3.57$ MHz). For T_{SCLK}=280 ns, we have `SCLK_T = 28` (at clock=100 MHz) as the minimum possible value.
  - ✓ To display data on LEDs or 7-segment displays, you need an appropriate refreshment rate. We can choose T_{SCLK}=1 ms (f_{SCLK}=1 KHz) → `SCLK_T`=10^6. Since there are 24 SCLK periods in a reading transaction, data is refreshed at 24 ms per sample.
  - ✓ FSM_SCLK: It generates a free running clock of period `SCLK_T` and 50% DC along with rising and falling edge detectors.



Serial Sync. Communication with SPI example w/ accelerometer



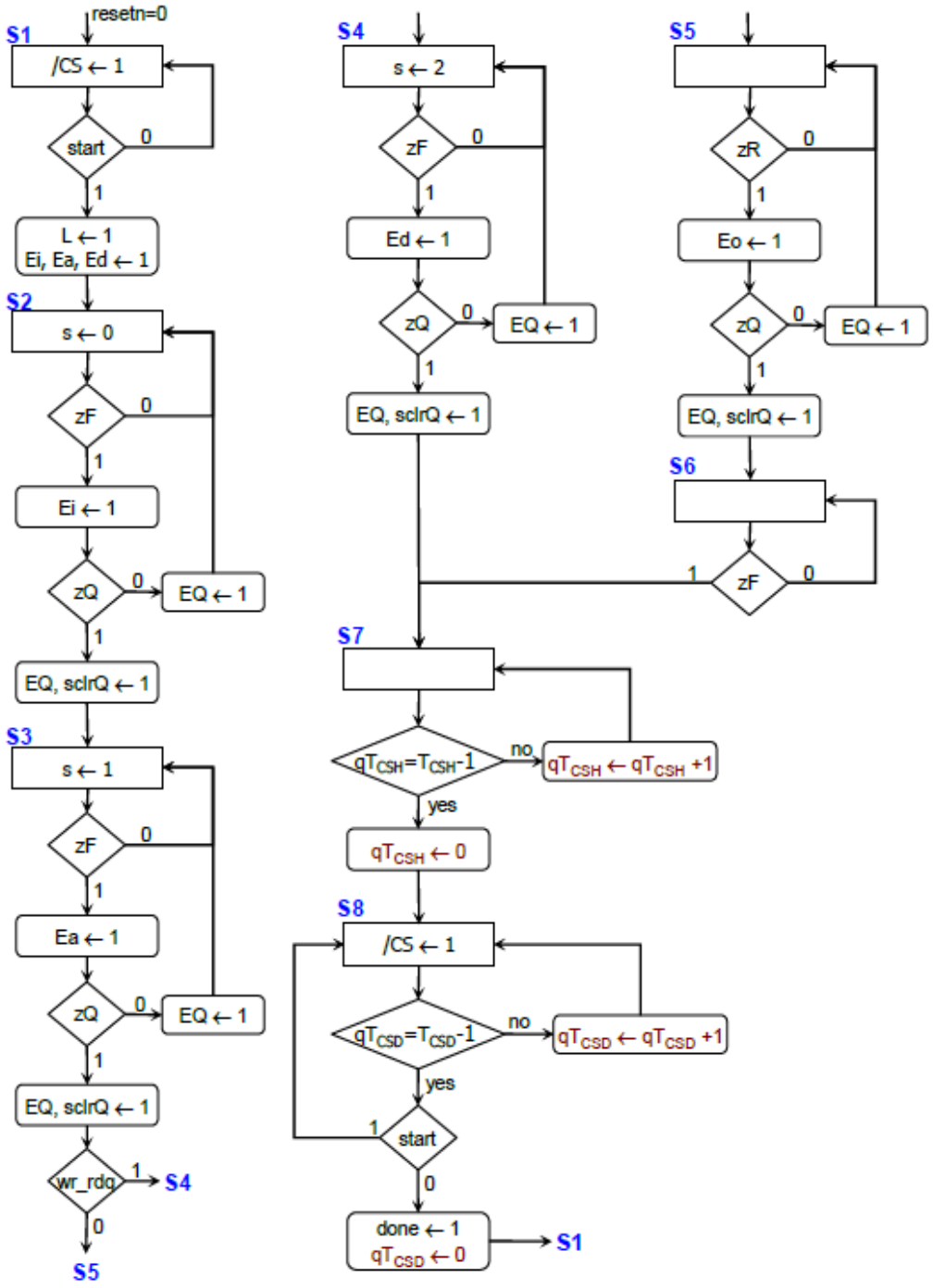


Serial Sync. Communication with SPI example w/ accelerometer

FSM_MAIN: handles the SPI communication and complies with the ADXL362 timing parameters. Command, address, data (MSB is sent first), same when reading.
Note that TCSD = TCSH = 2 (20 ns).

To comply with the timing parameters, we always wait until the last falling edge in a reading or writing cycle, then wait for TCSH cycles, set /CS=1 for TCSD cycles and then we are back in State S1 for a new transaction.
Note how we embed the counters for qTCSD and qTCSH inside the FSM.

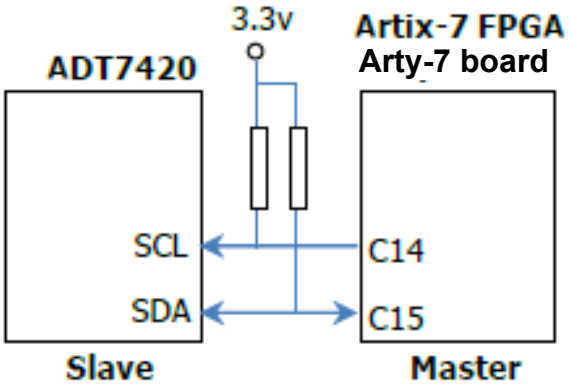
Other approaches do not have a free running SCLK, but instead they only activate it when /CS=0. This approach might make the controlling of the timing parameters simpler (depending on the timing parameters).



Serial Sync. Communication with I2C

I²C (Inter-Integrated Circuit) INTERFACE

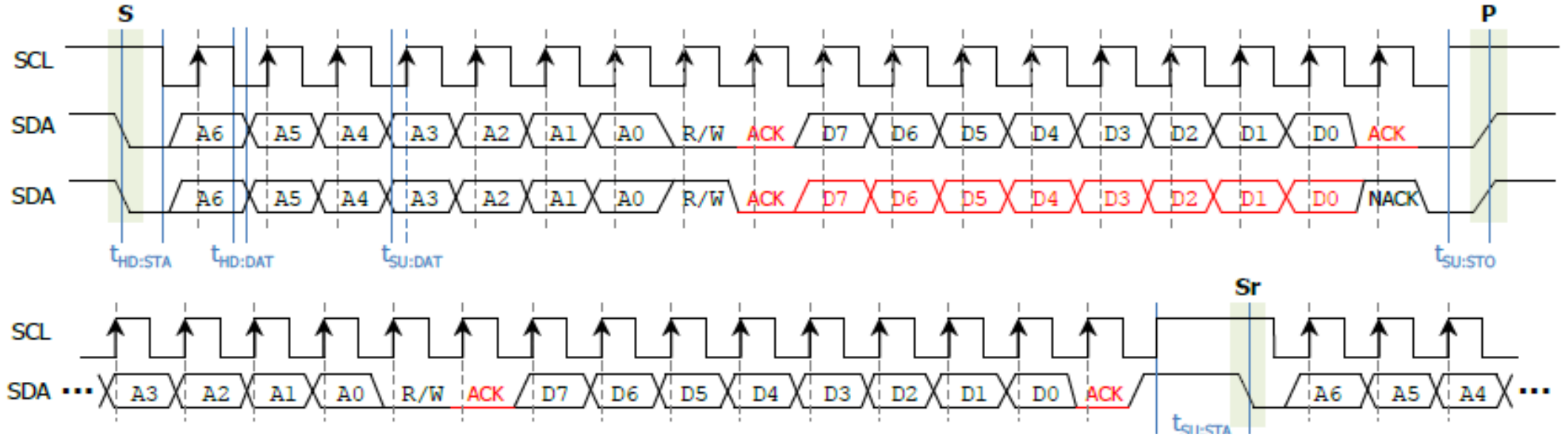
- Simple 2-wired synchronous (clock is transmitted) serial interface.
- I²C logic signals:
  - ✓ SCL: Serial clock. Generated by Master, defined by the Slave device. The standard specifies a Fast Mode (up to 400 KHz), a High Speed Mode (up to 3.4 MHz), and an Ultra-Fast Mode (up to 5 MHz).
  - ✓ SDA: Bi-directional serial data.
- In general, SCL and SDA are open-drain. There can be one Master and many Slaves. The Master device puts the slave address on the bus, and the slave device with the matching address acknowledges the Mater.
- Slave Address: Unique identifier of a device. 7-bits wide.



- I²C is commonly used for attaching lower-speed devices to processors and microcontrollers in short-distance, intra-board communication. Large variety of I²C-capable peripherals available: sensors (e.g.: temperature, acceleration, pressure), ADCs, DACs, touchscreens, memories, LCDs, SD cards.

Serial Sync. Communication with I2C

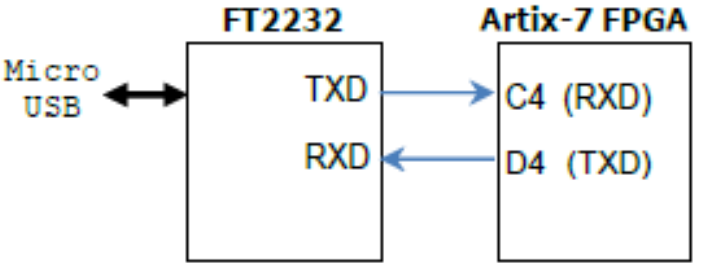
- Communication on the I^2C bus:
  - ✓ It starts when the master puts the START condition (S) on the bus (a high-to-low transition on SDA while SCL is high). The bus is considered to be busy until the Master puts a STOP condition (P) on the bus (a low-to-high transition on SDA while SCL is high).
  - ✓ I^2C data: Transferred in 8-bit packets. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge signal (ACK). ACK (0) is generated by the Slave.
  - ✓ After a START condition (S), the Master writes the 7-bit Slave Address followed by a Read/Write bit, then ACK. Then, the Master writes/reads bytes of data, each byte followed by an ACK. When writing, after the last written byte (followed by ACK), data transmission is terminated by the Master with a STOP condition (P). When reading, only on the last byte, the Master must generate a NACK (Not acknowledge) bit, and then a STOP condition (P). The Master can also generate a repeated START condition (Sr) without first generating a STOP condition (P) to signal that the bus is still busy.
  - ✓ Data bits are read on the SCL rising edge. We must comply with the Slave device timing parameters: $t_{SU:DAT}$, $t_{HD:DAT}$, $t_{SU:STA}$, $t_{HD:STA}$, $t_{SU:STO}$. Unlike a flip flop, $t_{HD:DAT}$ (hold time) is defined as the time the data bit should be on the bus after SCL is high (i.e., after the falling edge).

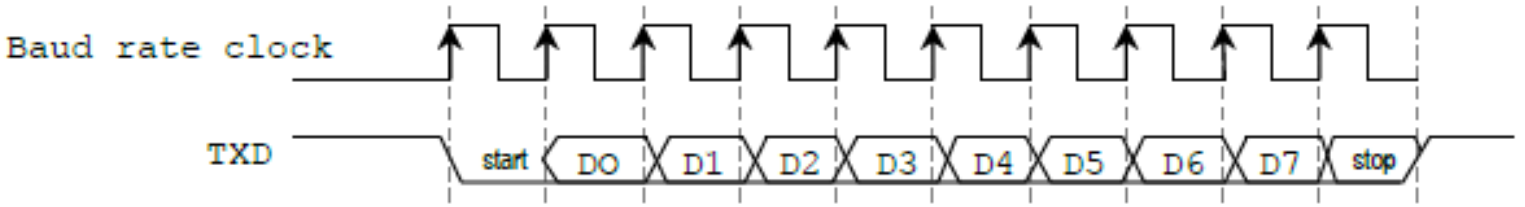


Serial Async. Communication with UART

UART INTERFACE

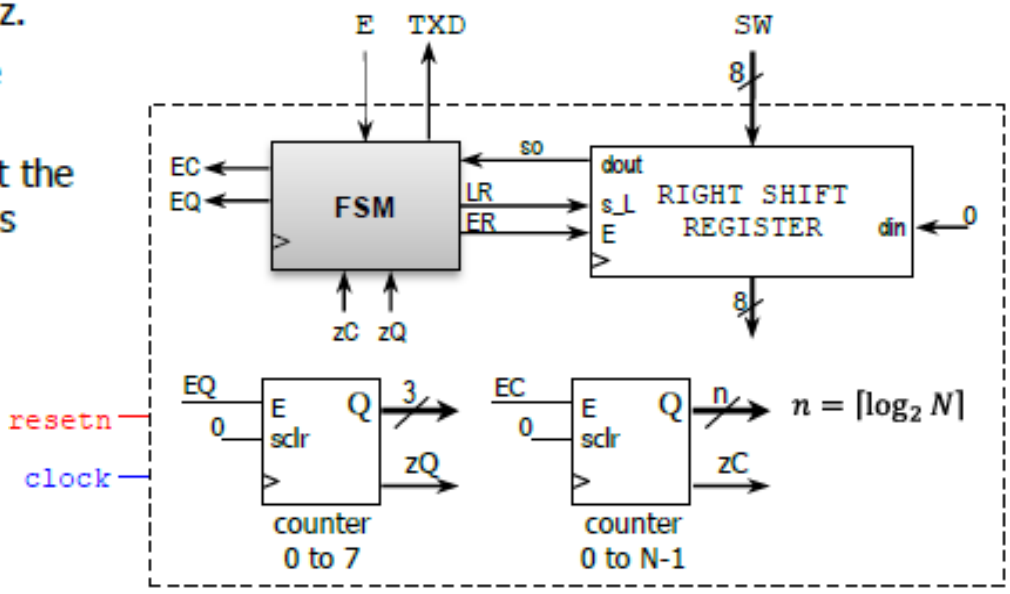
- This interface transfers data asynchronously (clock is not transmitted, transmitter and receiver use their own clocks).
- Data communication: RXD (receive pin), TXD transmit pin). The FT2232 chip inside the Arty-7 board handles the USB communication with a computer.
- Format of a Frame: Start bit ('0'), 8 to 9 data bits (LSB transmitted first), optional parity bit, and a stop bit ('1').
- Transmitter: Simple design that transmit the data frame at the Baud rate (or bit rate in bps).
- Receiver: It uses a clock signal whose frequency is a multiple (usually 16) of the incoming data rate.



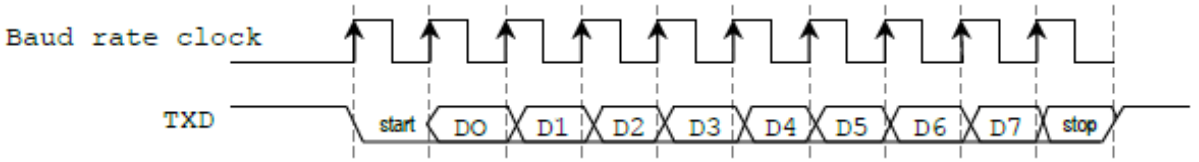


DIGITAL SYSTEM: UART TRANSMITTER (FSM + Datapath circuit)

- For a baud rate of 9600 bps, the Baud rate clock is 9600 Hz.
- $N = \frac{1/9600}{10\ ns} = 10416$. This number changes according to the desired baud rate.
- Counters: E=1 → Q ← Q+1. E=sclr=1 → Q ← 0. Note that the way the counters are designed, once the maximum count is reached, asserting the enable to '1' resets the count to 0.

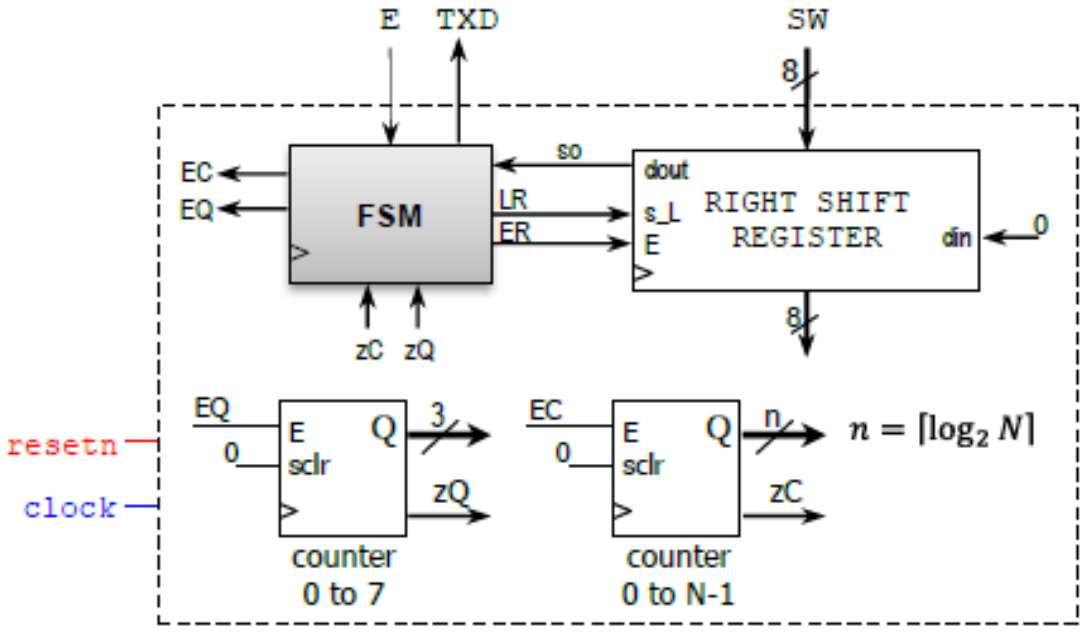


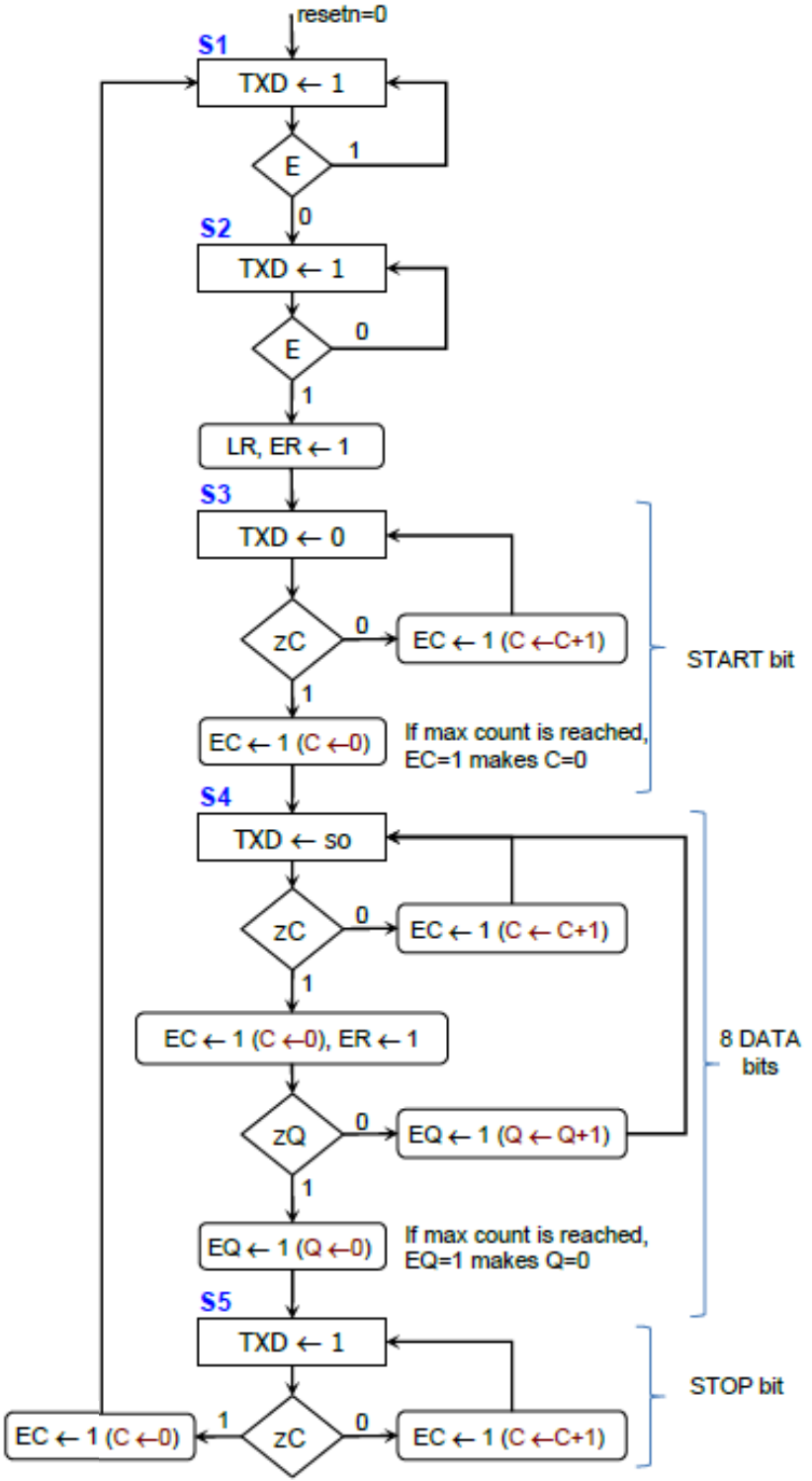
Serial Communication with UART



DIGITAL SYSTEM: UART TRANSMITTER (FSM + Datapath circuit)

- For a baud rate of 9600 bps, the Baud rate clock is 9600 Hz.
- $N = \frac{1/9600}{10\ ns} = 10416$. This number changes according to the desired baud rate.
- Counters: E=1 → Q ← Q+1. E=sclr=1 → Q ← 0. Note that the way the counters are designed, once the maximum count is reached, asserting the enable to '1' resets the count to 0.





Additional Materials

AXI Bus

Ethernet