Universidad
Carlos III de Madrid

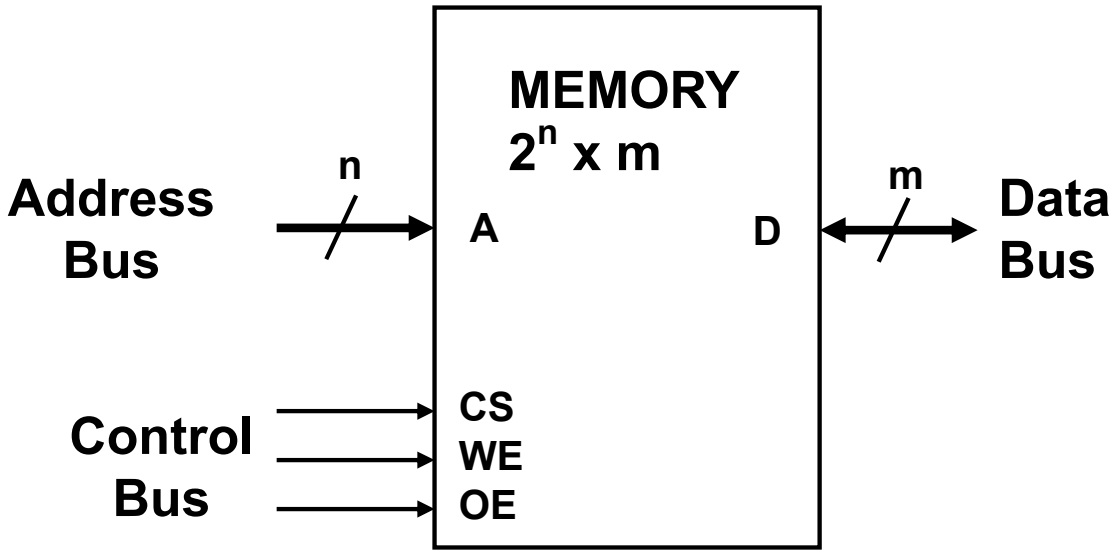
VHDL: Digital Circuits

MEMORY CIRCUITS

Outline

- Memory circuits review
- Memory circuits design
- Memory circuits synthesis

Basic Interface



Memory circuits review

- Address bus (A): Sets the position to access
  - n bits wide for a memory with 2^n positions
  - Example: 20 bits for 1M, 30 bits for 1G
- Data bus (D): Provides the datum
  - Data bus width (m) is equal to data size
  - Input data for writing
  - Output data for reading
  - Can be a single bidirectional bus, or two buses, one input bus and one output bus

Memory circuits review

- Control bus: signals that control memory operation. Some typical signals:
  - CS (Chip Select) or CE (Chip Enable): when deasserted, data bus is typically in tristate mode
  - R/W (Read/Write) or WE (Write Enable): selects the memory operation (read or write)
  - OE (Output Enable): Enables data output. When deasserted, data bus is typically in tristate mode
  - RAS (Row Address Strobe) and CAS (Column Address Strobe) in memories with 3D organization
  - Other signals: CLK in synchronous memories, BE (Byte Enable) to select particular bytes in a memory word, etc.

Memory circuits review

Semiconductor memory circuits

- RAM memory (Random Access Memory)
  - Memories for reading and writing
  - Example: computer main memory
- ROM memory (Read Only Memory)
  - Read only memory
  - Contents fixed during manufacturing or programmable
  - Example: Flash memory

Memory circuits review

RAM Memory

- Read & Write memory
- Two main types:
  - Static RAM (SRAM): every bit is stored in a bistable
  - Dynamic RAM (DRAM): every bit is stored in a capacitor
- Currently, the most widely used type of DRAM is DDR (Double Data Rate) SDRAM
  - 3D organization
  - Clocked memory (Synchronous DRAM)
  - Data access in both clock edges (Double Data Rate)
  - 64 bits wide data
  - Versions evolving on time: DDR, DDR2, DDR3, DDR4

Memory circuits review

ROM Memory

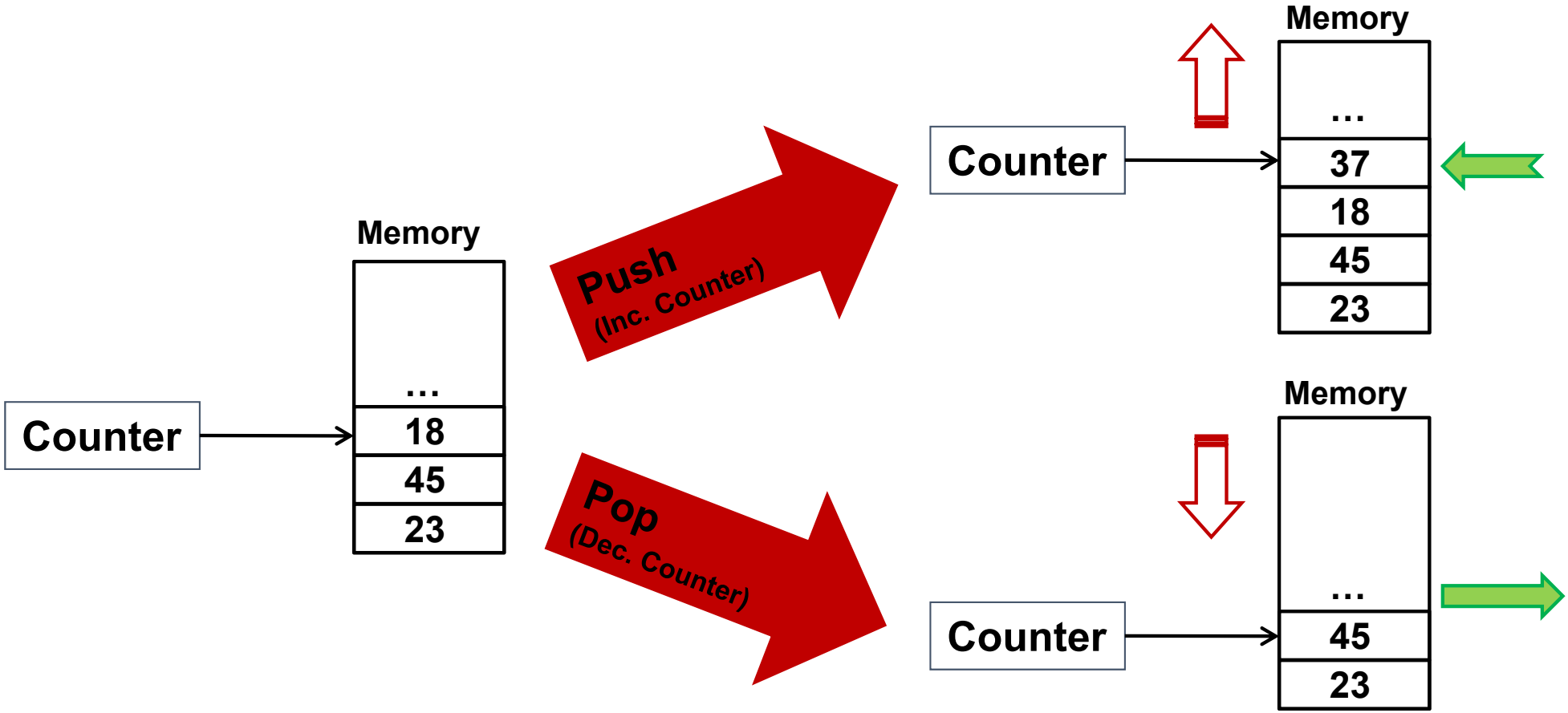
- Read Only Memory
- Non volatile
- Types
  - Mask programmable: contents fixed by manufacturing
  - One-Time Programmable: contents fixed by the user
  - Erasable or Reprogrammable: can be erased to store new contents
- The concepts of “writing” and “programming” should be distinguished, although the frontier is becoming blurred:
  - Writing is an operation similar to reading
  - Programming uses physical means different from reading, it is generally slower and it must be applied on a block basis or even for the entire chip

Memory circuits review

- Synchronous/asynchronous
- Single Port: Only one access (read/write) per clock cycle.
- Double Port: Two address and control buses. Two addresses can be accessed simultaneously.
- LIFO (Last Input First Output) and FIFO (First Input First Output). They may be single or double port.

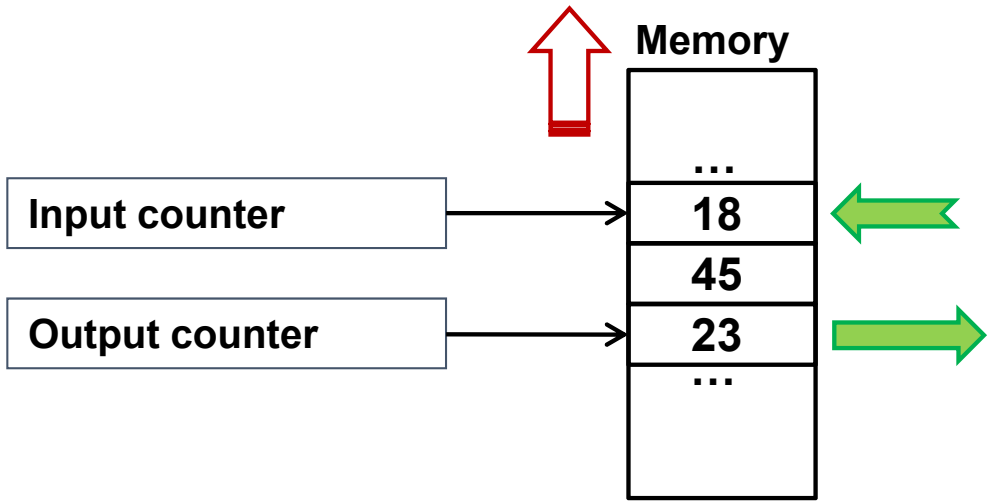
Last Input First Output Memory (LIFO)

- LIFO (Last In First Out): Implements a stack.

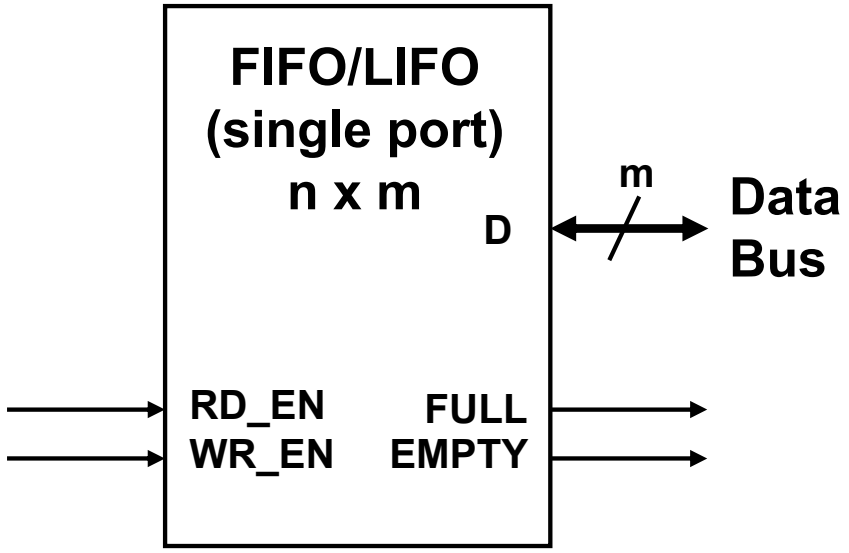


First Input First Output Memory (FIFO)

- FIFO (First In First Out): Data get in and out in the same sort order, but at different instants
- Application: Temporary storage for reading a data burst



FIFO/LIFO example: Interface



- n: size, maximum number of data that can be writen in the FIFO/LIFO
- RD_EN/WR_EN: Used to activate a read/write.
- FULL/EMPTY: Indicate wether the memory is empty (no more data can be read) or full (no more data can be written)

Memory circuits synthesis

- Distributed memory
  - Using flip-flops
    - Independent of the technology.
    - Synchronous writing, asynchronous reading
    - They may have high area requirements
  - Using LUTs (Look-up tables)
    - Dependent of the techonolgy (they need to be available in the target device).
    - Synchronous writing, asynchronous reading
- Dedicated memory blocks
  - Dependent of the techonology
  - Synchronous reading and writing
- Each manufacturer provides the necessary means:
  - Synthesis tools
  - Synthesis directives
  - Libraries and code generators (example: Xilinx Core Generator)

Example: Asynchronous ROM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity rom_example is
  port (address : in  integer range 0 to 255;
        data_o  : out std_logic_vector(7 downto 0)
     );
end rom_example;

architecture behavioral of rom_example is

  type rom_t is array (0 to 255) of std_logic_vector(7 downto 0);
  signal rom : rom_t := (
     X”FF”, -- address = 0
     X”01”, -- address = 1
     X”34”, -- address = 2
     X”87”,
     . . .
    );

begin

  data_o <= rom(address);

end behavioral;
```

- It can be implemented only as distributed memory

Example: RAM with asynchronous reading

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ram_example is
  port (Clk     : in  std_logic;
        address : in  integer range 0 to 255;
        we      : in  std_logic;
        data_i  : in  std_logic_vector(7 downto 0);
        data_o  : out std_logic_vector(7 downto 0)
     );
end ram_example;

architecture behavioral of ram_example is

  type ram_t is array (0 to 255) of std_logic_vector(7 downto 0);
  signal ram : ram_t := (others => (others => '0'));

begin

  data_o <= ram(address);

  process(Clk)
  begin
    if rising_edge(Clk) then
      if (we='1') then
        ram(address) <= data_i;
      end if;
    end if;
  end process;

end behavioral;
```

- Asynchronous reading, synchronous writing: it can be only distributed memory

Example: RAM with synchronous reading

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ram_example is
  port (Clk     : in  std_logic;
        address : in  integer range 0 to 255;
        we      : in  std_logic;
        data_i  : in  std_logic_vector(7 downto 0);
        data_o  : out std_logic_vector(7 downto 0)
     );
end ram_example;

architecture behavioral of ram_example is

  type ram_t is array (0 to 255) of std_logic_vector(7 downto 0);
  signal ram : ram_t := (others => (others => '0'));

begin

  process(Clk)
  begin
    if rising_edge(Clk) then
      if (we='1') then
        ram(address) <= data_i;
      end if;
      data_o <= ram(address);
    end if;
  end process;

end behavioral;
```

- Synchronous reading and writing: it can be distributed or block
- "Read first": reading returns the previous value to the one that is being written