ECE 3401 Lecture 16

Memory & Timing Issues

Memories Overview

- Memories
  - Memory categories and timing
  - Random access memory (RAM):
    - SRAM
    - DRAM
- Timing issues
  - Sequential system timing requirements
  - Clock skew and clock jitter
  - Clock distribution

Memories

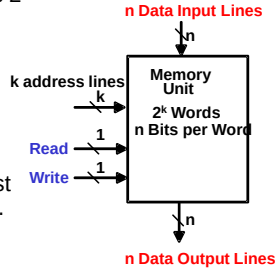
Read-Write Memory			Read-Only Memory
Volatile Memory		Non-volatile Memory	Mask-Programmed ROM (PROM) (nonvolatile)
Random Access	Sequential Access	EPROM EEPROM FLASH	
DRAM SRAM	FIFO LIFO Shift Register CAM		

- **Volatile: need electrical power**
- **Nonvolatile: magnetic disk, retains its stored information after the removal of power**
- **Random access: memory locations can be read or written in a random order**
- **EPROM: erasable programmable read-only memory**
- **EEPROM: electrically erasable programmable read-only memory**
- **FLASH: memory stick, USB disk**
- **Access pattern: sequential access: (video memory streaming) first-in-first-out (buffer), last-in-first-out (stack), shift register, content-addressable memory**

Memories

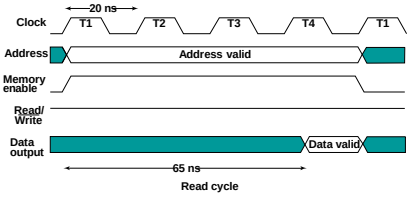
- k address lines are decoded to address 2^k words of memory.
- Each word is n bits.
- Read and Write are single control lines defining the simplest memory operations.

A basic memory system:



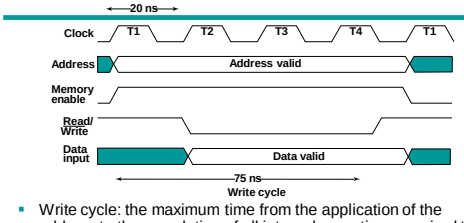
Memory Operation Timing - Reading

- Most basic memories are asynchronous
  - Storage in latches or storage of electrical charge
  - No clock
  - Controlled by control inputs and address, which are controlled by CPU and synchronized by its own clock
- Timing of signal changes/data observation is critical to the operation



- Read cycle: the access time, the maximum time from the application of the address to the appearance of the data at the Data output.

Memory Operation Timing - Writing



- Write cycle: the maximum time from the application of the address to the completion of all internal operations required to store a word
- Critical times measured with respect to edges of write pulse (1-0-1):
  - Address must be established at least a specified time before 1-0 and held for at least a specified time after 0-1 to avoid disturbing stored contents of other addresses
  - Data must be established at least a specified time before 0-1 and held for at least a specified time after 0-1 to write correctly

VHDL code for ROM

```
library IEEE;
use IEEE.std_logic_1164.all;

ENTITY rom8x4 IS
 PORT (
 addr: in std_logic_vector(2 downto 0);
 q: out std_logic_vector(3 downto 0));
END rom8x4;
```

```
ARCHITECTURE behav OF rom8x4 IS
BEGIN

PROCESS(addr)
BEGIN
 CASE addr IS
   when "000" => q <= "0001";
   when "001" => q <= "0000";
   when "010" => q <= "0111";
   when "011" => q <= "1101";
   when "100" => q <= "1000";
   when "101" => q <= "1100";
   when "110" => q <= "0110";
   when "111" => q <= "1011";
   when others  =>  NULL;
 END case;
 END process;
END behav;
```

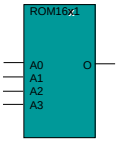

ROMS in Xilinx library

- **ROM16x1 (16-word by 1 bit ROM)**
- **ROM32x1**
- **ROM64x1**
- **ROM128x1**
- **ROM256x1**

ROMS in Xilinx library

```
component ROM16X1
-- synthesis translate_off
 generic (INIT : bit_vector := X"0016");
-- synthesis translate_on
 port (O : out STD_ULOGIC;
          A0 : in STD_ULOGIC;
       A1 : in STD_ULOGIC;
       A2 : in STD_ULOGIC;
       A3 : in STD_ULOGIC);
end component;

architecture beh of rom is
begin
ROM16X1_INSTANCE_NAME : ROM16X1
-- synthesis translate_off
 generic map (INIT => hex_value )
-- synthesis translate_on
 port map (O => user_O, A0 => user_A0 , A1 => user_A1,
          A2 => user_A2, A3 => user_A3 );
end architecture
```



How do we use ROM

- VHDL components in a VHDL design
- Graphical components in a schematic

```
architecture beh of proc is
 signal ABUS : std_logic_vector(3 downto 0);
 signal DBUS : std_logic_vector(1 downto 0);
begin

bit0 : ROM16X1
-- synthesis translate_off
 generic map (INIT => "1010 1110 0001 0001" )
-- synthesis translate_on
 port map (O => DBUS(0), A0 => ABUS(0),
  A1 => ABUS(1), A2 => ABUS(2), A3 => ABUS(3) );

bit1 : ROM16X1
-- synthesis translate_off
 generic map (INIT => "1101 1010 1111 0101")
-- synthesis translate_on
 port map (O => DBUS(1), A0 => ABUS(0),
  A1 => ABUS(1), A2 => ABUS(2), A3 => ABUS(3) );
```

```
rom_inc : process(CLK) is
begin
  if (CLK'event and CLK='1') then
     ABUS <= ABUS + 1;
  end if;
end process;

jump : process(DBUS) is
begin
  if (DBUS="11") then
     ABUS <= "0000";
  end if;
end;

end architecture
```


Random Access Memories (RAMs)

- Read/Write memory
- Types:
  - Static RAM (SRAM):
    - Once a word is written at a location, it remains stored as long as power is applied to the chip, unless the same location is written again.
    - Fast speed, but their cost per bit higher.
    - Application: Caches memories in Microprocessor
  - Dynamic RAM (DRAM):
    - The data stored at each location must be periodically refreshed by reading it and then writing it back again, otherwise it disappears.
    - Their density is greater and their cost per bit lower, but the speed is slower.

RAMS in Xilinx library

- Static Block RAMs
- Single port
  - RAMB4_S1, RAMB4_S2, RAMB4_S8, RAMB4_S16
- Dual port
  - RAMB4_S1_S1, RAMB4_S1_S2, … RAMB4_S16_S16

- **The difference between single port RAM and dual port RAM is that single port RAM can be accessed at one address at one time, thus you can read/write only one memory cell during each clock cycle. Dual port RAM has ability to simultaneously read and write different memory cells at different addresses.**
- **SPRAM uses a 6 transistor basic ram cell, while the dual port ram cell uses 8 transistor cell for memory.**

RAMS in Xilinx library

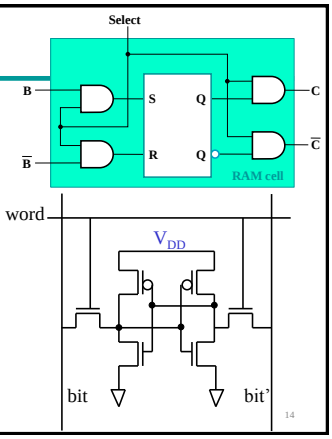
```
component RAMB4_S1
-- synthesis translate_off
 generic ( INIT_00 : bit_vector :=
      X"0000000000000000000000000000000000000000000000000000000000000000";
        INIT_01 : bit_vector :=
      X"0000000000000000000000000000000000000000000000000000000000000000";
        . . .
        );
-- synthesis translate_on
 port (DO : out STD_LOGIC_VECTOR (0 downto 0);
       ADDR : in STD_LOGIC_VECTOR (11 downto 0);
       CLK : in STD_ULOGIC;
       DI : in STD_LOGIC_VECTOR (0 downto 0);
       EN : in STD_ULOGIC;
       RST : in STD_ULOGIC;
       WE : in STD_ULOGIC);
end component;

RAMB4_Sn_INSTANCE_NAME : RAMB4_Sn
-- synthesis translate_off
 generic map ( INIT_00 => 64bit_hex_value,
               INIT_01 => 64bit_hex_value,
                 . . .
      INIT_0F => 64bit_hex_value)
-- synthesis translate_on
 port map (DO => user_DO, ADDR => user_ADDR, CLK => user_CLK, DI => user_DI,
           EN => user_EN, RST => user_RST, WE => user_WE );
```

13

SRAM Cell

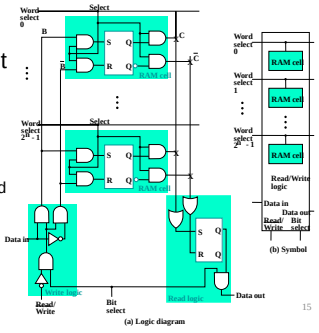
- Array of storage cells used to implement static RAM
- Storage Cell
  - SR Latch
  - Input "Select" for control
  - Dual Rail Data
    - Inputs B and $\overline{B}$
    - Outputs C and $\overline{C}$



14

SRAM Bit Slice

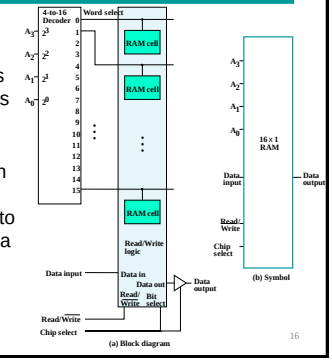
- Represents all circuitry that is required for 2^n 1-bit words
  - Multiple RAM cells
  - Control Lines:
    - Word select i – one for each word
    - Read/$\overline{\text{Write}}$
    - Bit Select
  - Data Lines:
    - Data in
    - Data out



15

2^n-Word by 1-Bit RAM IC

- To build a RAM IC from a RAM slice:
  - Decoder decodes the n address lines to 2^n word select lines
  - A 3-state buffer on the data output permits RAM ICs to be combined into a RAM with c × 2^n words



16

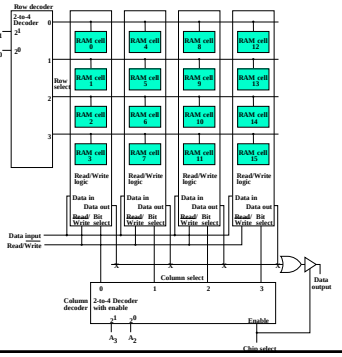
Cell Arrays and Coincident Selection

- Memory arrays can be very large =>
  - Large decoders
  - Large fanouts for the input bit lines
  - The decoder size and fanouts can be reduced by approximately $\sqrt{n}$ using a coincident selection in a 2-D array: uses two decoders, one for words and one for bits:
    - Word select becomes Row select
    - Bit select becomes Column select
- See next slide for example

17

Cell Arrays and Coincident Selection (Contd.)

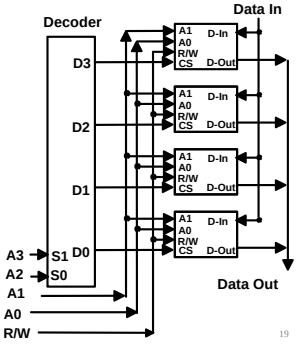
A_1 and A_0 used for Row select
A_3 and A_2 for Column select



18

Making Larger Memories

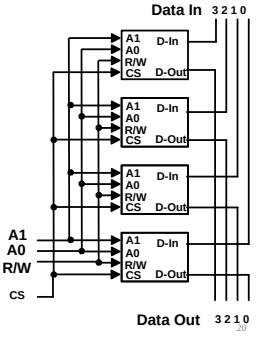
- We can make larger memories from smaller ones by using the decoded higher order address bits to control CS (chip select) lines, tying all address, data, and R/W lines in parallel.
- A 16-Word by1-Bit memory constructed using 4-Word by 1-Bit memory.



19

Making Wider Memories

- Tie the address and control lines in parallel and keep the data lines separate.
- Example: make a 4-word by 4-bit memory from 4, 4-word by 1-bit memories
- Note: Both 16x1 and 4x4 memories take 4-chips and hold 16 bits of data.



20

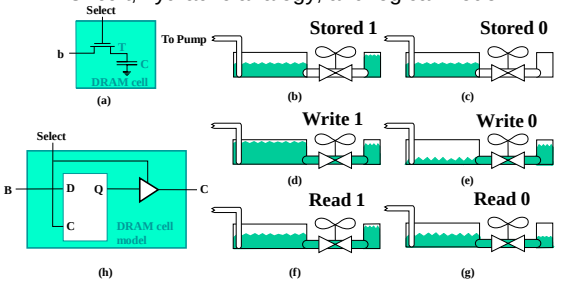
DRAM

- Basic Principle: Storage of information on capacitors.
- Charge and discharge of capacitor to change stored value
- Use of transistor as "switch" to:
  - Store charges
  - Charge or discharge

21

Dynamic RAM (Contd.)

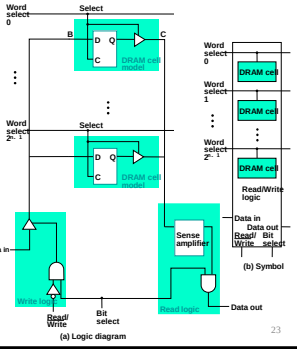
- Circuit, hydraulic analogy, and logical model.



22

Dynamic RAM - Bit Slice

- C driven by 3-state drivers
- Sense amplifier is used to change the small voltage change on C into H or L
- In the electronics, B, C, and the sense amplifier output are connected to make destructive read into non-destructive read



23