DSLab. 15 Memory

Lab. 15 Memory

- Design and Verify the following circuits using Verilog HDL
  - 4 x 4 RAM (Structural modeling)
  - 8 x 8 Register File (Behavioral modeling)

- Please write and upload the lab report (Lab15) -- Due on 2022/12/16 23:59

Example 1: 4-bit ripple counter (1/2)

Table 6.4
Binary Count Sequence

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

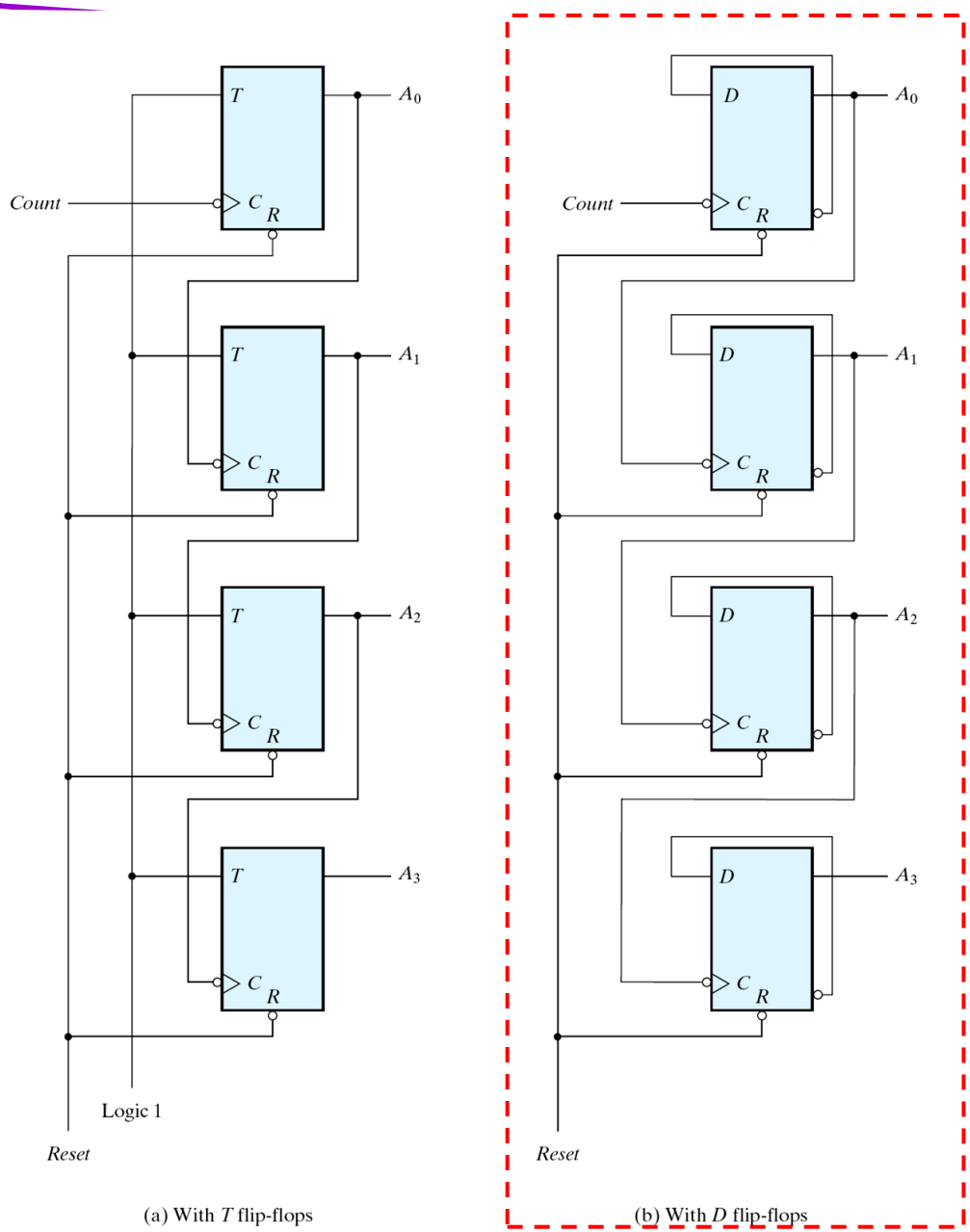


Fig. 6.8
Four-bit binary ripple counter

Example 1: 4-bit ripple counter (2/2)

Structural model

```
`timescale 1ns / 100 ps
module Ripple_Counter_4bit (A3,A2,A1,A0, Count, Reset);
  output A3,A2,A1,A0;
  input Count,Reset;
//Instantiate complementing flip-flop
   Comp_D_flip_flop F0 (A0, Count, Reset);
   Comp_D_flip_flop F1 (A1, A0, Reset);
   Comp_D_flip_flop F2 (A2, A1, Reset);
   Comp_D_flip_flop F3 (A3, A2, Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module Comp_D_flip_flop (Q, CLK, Reset);
   output Q;
   input CLK, Reset;
   reg Q;
   always @ (negedge CLK, posedge Reset)
   if (Reset) Q <= 1'b0; else Q <= #2 ~Q;
endmodule
```

```
//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   Ripple_Counter_4bit M0 (A3, A2, A1, A0, Count, Reset);
   always
   #5 Count = ~Count;
   initial
     begin
        Count = 1'b0;
        Reset = 1'b1;
        #4 Reset = 1'b0;
     end

   initial #170 $finish;

endmodule
```

Example 2: 64 x 4 memory (1/3)

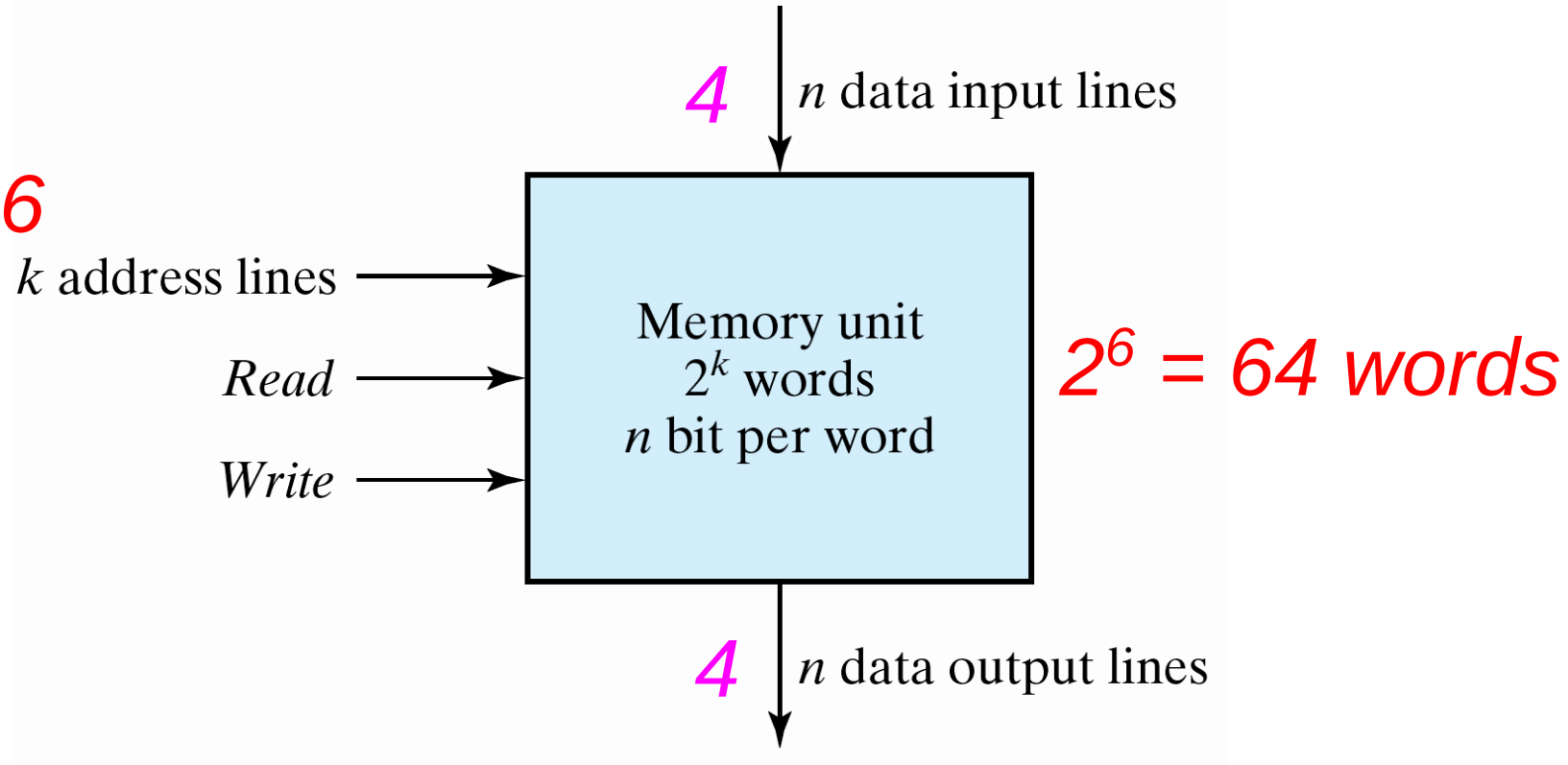


Fig. 7.2 Block diagrams of a memory unit

Example 2: 64 x 4 memory (2/3)

Behavioral model

```
module memory (Enable, ReadWrite, Address, DataIn, DataOut);
  input             Enable, ReadWrite;
  input    [3: 0]   DataIn;
  input    [5: 0]   Address;
  output   [3:0]    DataOut;
  reg      [3: 0]   DataOut;
  reg      [3: 0]   Mem [0: 63];                  //64 x 4 memory

  always @ (Enable or ReadWrite or Address)
    if (Enable)
      if (ReadWrite) DataOut = Mem[Address];  //Read
      else Mem[Address] = DataIn;             //Write
    else DataOut = 4'bz;                      //High impedance state
endmodule
```

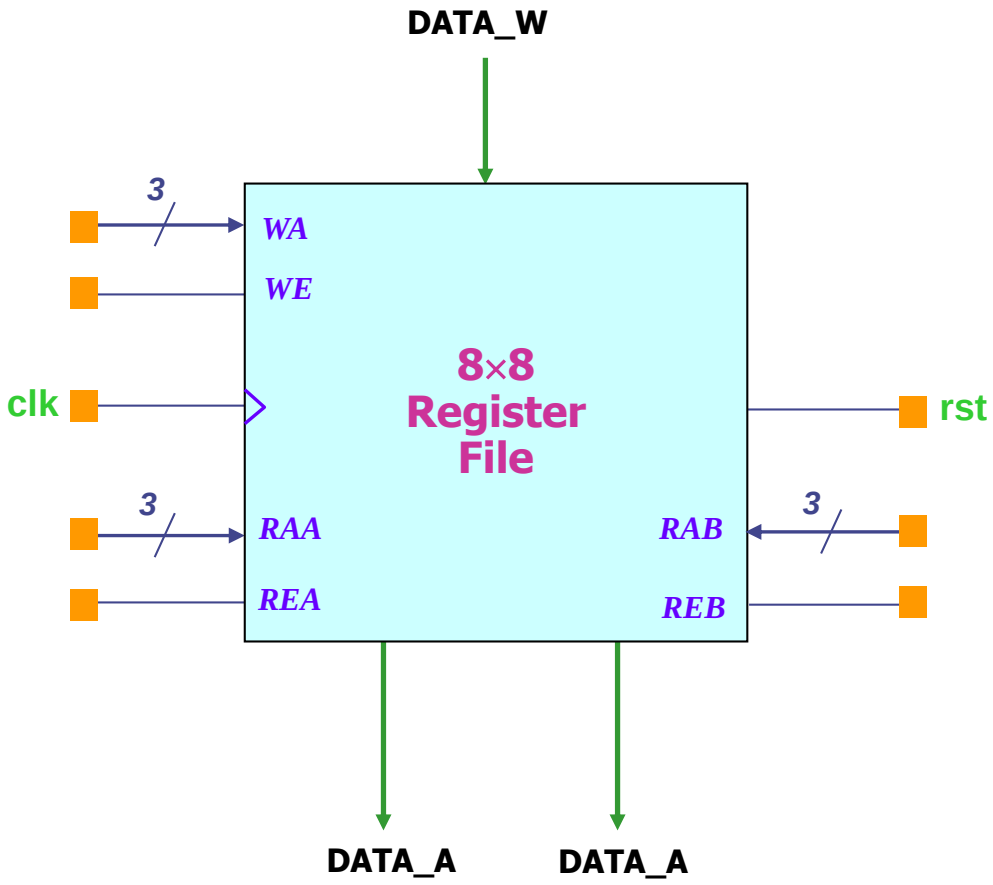
Example 2: 64 x 4 memory (3/3)

```
module t_memoryt;
  reg     t_Enable, t_ReadWrite;
  reg     [3: 0]   t_DataIn;
  reg     [5: 0]   t_Address;
  wire    [3:0]    t_DataOut;
  memory M0 (t_Enable, t_ReadWrite, t_Address, t_DataIn, t_DataOut);
  initial #150 $finish;

  initial begin
     t_Enable = 1;
     t_ReadWrite = 0;
     t_Address = 0;
     t_DataIn = 0;
     #10 t_ReadWrite=1'b0;       //Write
     #10 t_Address =5'b00001; t_DataIn =4'b0001;
     #10 t_Address =5'b00010; t_DataIn =4'b0010;
     #10 t_Address =5'b00011; t_DataIn =4'b0011;
     #10 t_Address =5'b00100; t_DataIn =4'b0100;
     #10 t_Address =5'b00101; t_DataIn =4'b0101;
     #10 t_Address =5'bxxxxx; t_DataIn =4'bxxxx;
     #10 t_ReadWrite=1'b1;       //Read
     #10 t_Address =5'b00001;
     #10 t_Address =5'b00010;
     #10 t_Address =5'b00011;
     #10 t_Address =5'b00100;
     #10 t_Address =5'b00101;
  end
endmodule
```

Example 3: 8 x 8 Register file with 2 read and 1 write ports

```
module regfile(WA, WE, RAA, REA, RAB, REB, DATA_W, DATA_A, DATA_B, clk, rst);

input [2:0] WA, RAA, RAB;
input WE, REA, REB, clk, rst;
input [7:0] DATA_W;
output [7:0] DATA_A, DATA_B;

reg [7:0] DATA_A, DATA_B;
reg [7:0] registers[7:0];

 assign DATA_A = registers[RAA];
 assign DATA_B = registers[RAB];

always @ (posedge clk or negedge rst)
begin
  if (~rst)
     registers[0] = 0;
  else
  begin
      if (WE) registers[WA] <= DATA_W;
  end
end

endmodule
```

Behavioral model



Example 4: 4 x 16 decoder

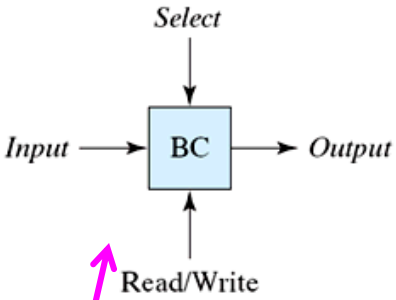
```
module decoder_case (binary_in,  decoder_out , enable);
   input [3:0] binary_in  ;            //  4 bit binary input
   input  enable ;                     //  Enable for the decoder
   output [15:0] decoder_out ;         //  16-bit out

   reg [15:0] decoder_out ;

 always @ (enable or binary_in)
 begin
    decoder_out = 0;
    if (enable) begin
       case (binary_in)
           4'h0 : decoder_out = 16'h0001;
           4'h1 : decoder_out = 16'h0002;
           4'h2 : decoder_out = 16'h0004;
           4'h3 : decoder_out = 16'h0008;
           4'h4 : decoder_out = 16'h0010;
           4'h5 : decoder_out = 16'h0020;
           4'h6 : decoder_out = 16'h0040;
           4'h7 : decoder_out = 16'h0080;
           4'h8 : decoder_out = 16'h0100;
           4'h9 : decoder_out = 16'h0200;
           4'hA : decoder_out = 16'h0400;
           4'hB : decoder_out = 16'h0800;
           4'hC : decoder_out = 16'h1000;
           4'hD : decoder_out = 16'h2000;
           4'hE : decoder_out = 16'h4000;
           4'hF : decoder_out = 16'h8000;
        endcase
   end
 end
endmodule
```

Behavioral model

```
module decoder_assign (binary_in,  decoder_out , enable);
   input [3:0] binary_in  ;            //  4 bit binary input
   input  enable ;                     //  Enable for the decoder
   output [15:0] decoder_out ;         //  16-bit out

  wire [15:0] decoder_out ;

   assign decoder_out = (enable) ? (1 << binary_in) : 16'b0 ;

endmodule
```

Exercise 1: 4 x 4 RAM (1/2)

- Design and verify the 4 x 4 RAM using Verilog HDL
  - Structural modeling



Behavioral modeling

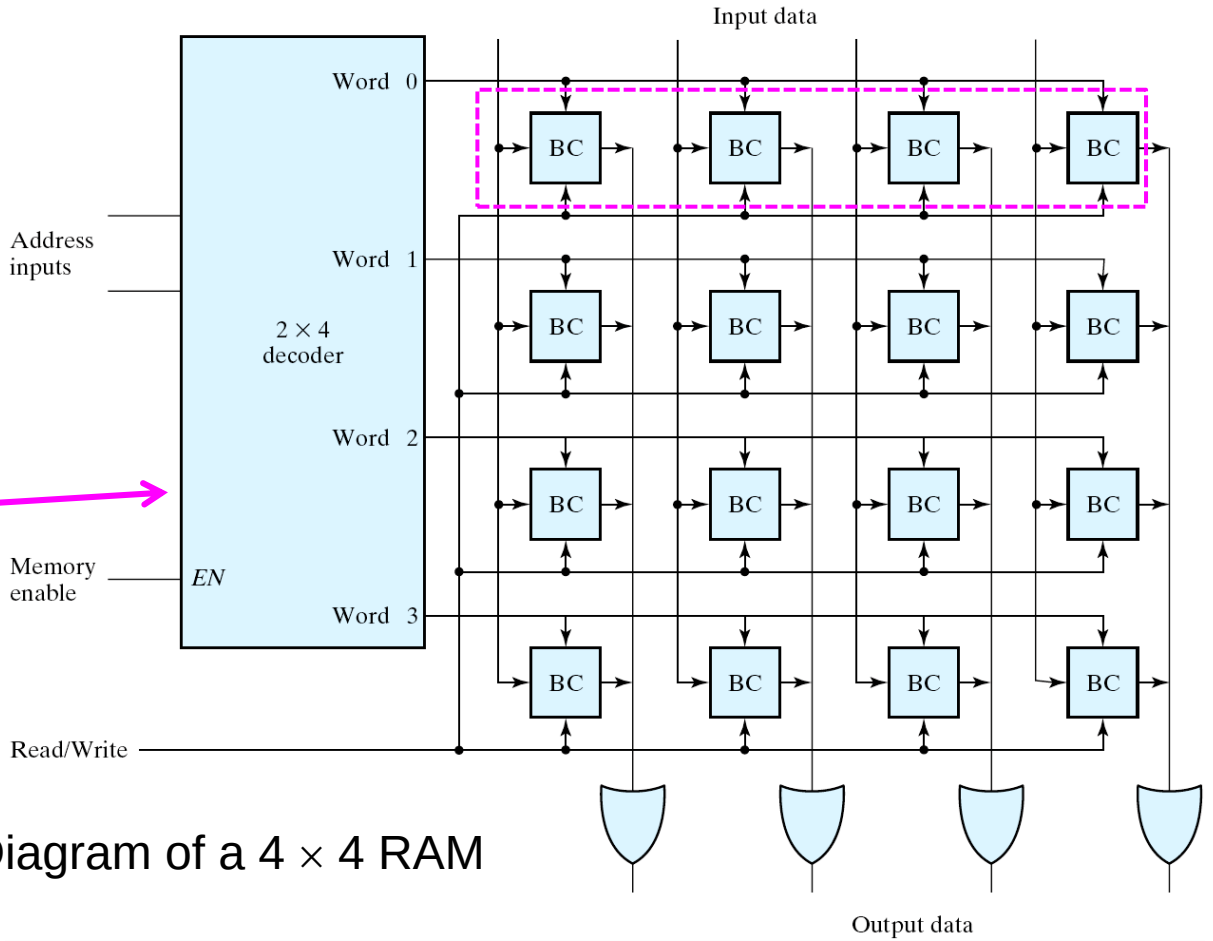


Fig. 7.6 Diagram of a 4 × 4 RAM

Exercise 1: 4 x 4 RAM (2/2)

- The memory unit BC can be described using behavioral modeling

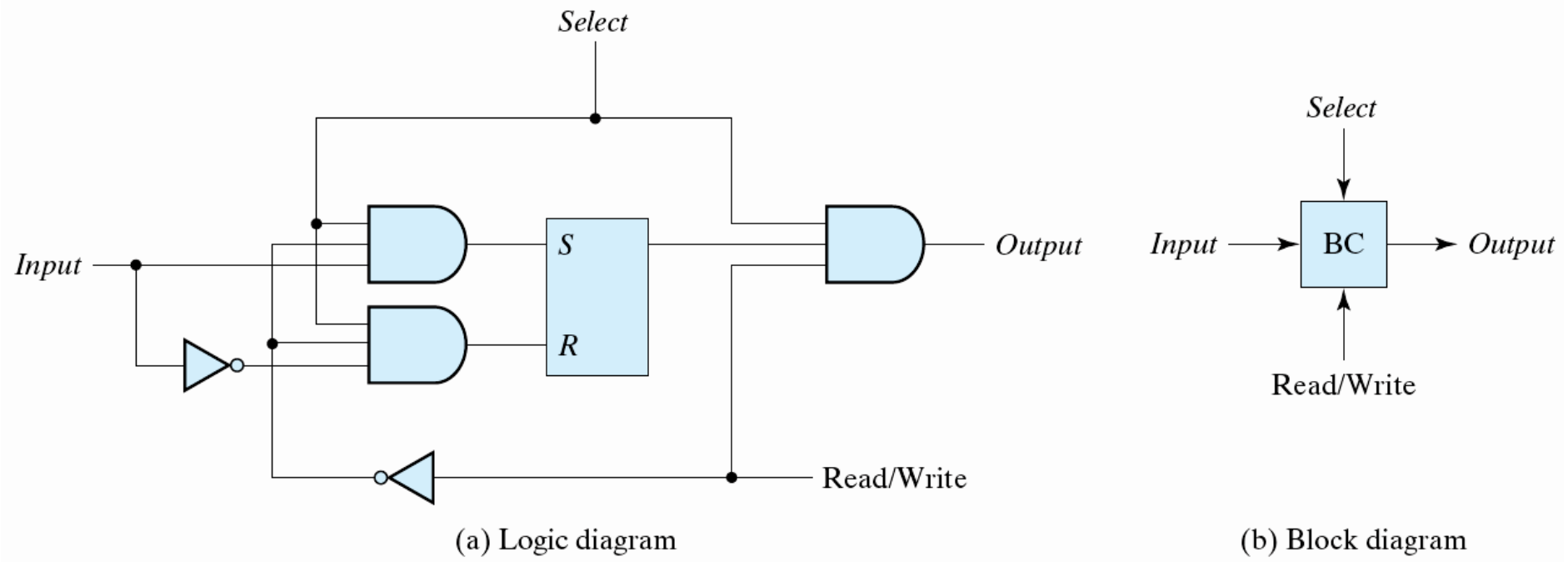
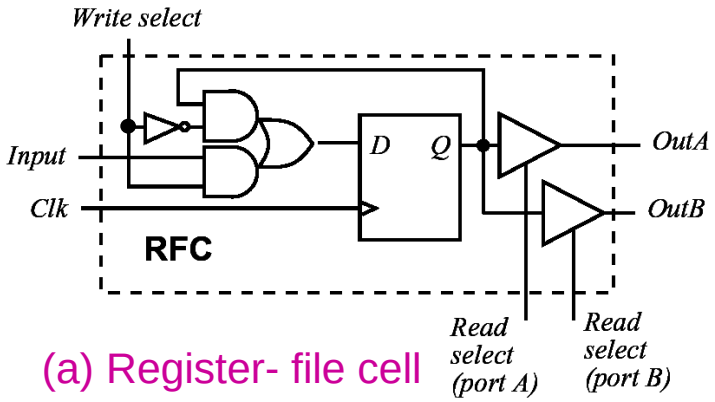


Fig. 7.5 Memory cell

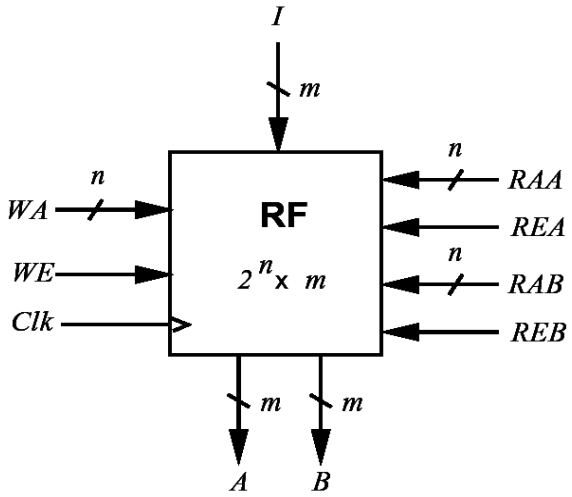
Exercise 2: 8 x 8 Register File (1/2)

- Register file is used as fast temporary storage
- Design and verify the 8 x 8 register file with 2 read ports and 1 write port using Verilog HDL
  - Behavioral modeling

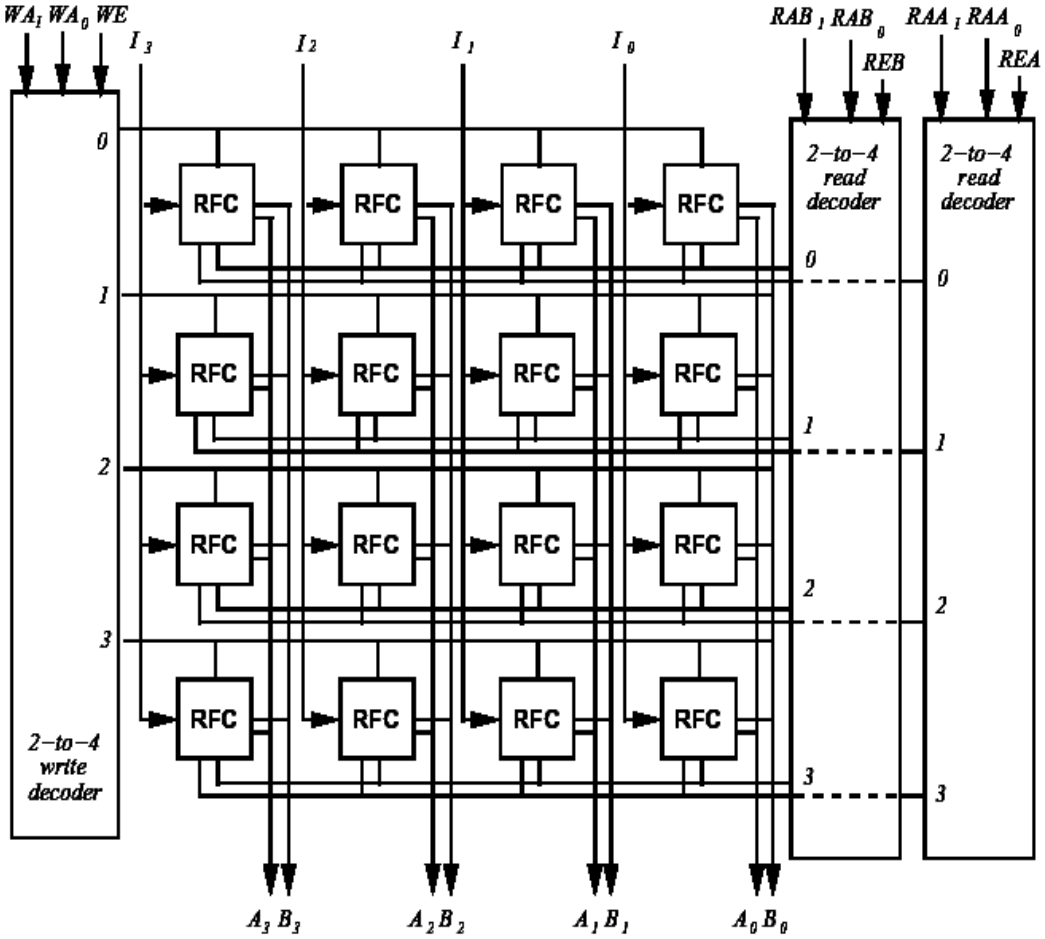
Exercise 2: 8 x 8 Register File (2/2)



(a) Register- file cell



(b) Graphic symbol



(c) Logic Schematic