*Name: **Solution **

*Registration: _____

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design
CSE 308

Finalterm Examination Spring 2022

21 July 2022, Duration: 120 Minutes

Exam Rules

Please read carefully before proceeding.

- This exam is CLOSED books/notes.
- No calculators/phones of any kind are allowed.
- Attempt all problems on the problem sheet. Use the answer sheet for scratch space and write a neat copy of your final answer in the provided space on the problem sheet. **Very Important!**
- Be precise and concise in your answers (no extra explanatory text).
- Some problems are harder than others. Answer the easy ones first to maximize your score.
- Problems will not be interpreted during exam. **Please note!**
- This exam booklet contains 7 pages, excluding this cover page. Count them to be sure you have them all.

Problem 1	_____	(20 pts.)
Problem 2	_____	(20 pts.)
Problem 3	_____	(30 pts.)
Total	_____	**(70 pts.)**

Don't Panic!

Problem 1.....(20 pts.)

As discussed in the class, it is possible to combine two chips to form a composite memory containing the same number of words but with twice as many bits in each word. Figure below shows the interconnection of two 1Kx32 RAM chips to form a 1Kx64 memory. The 64 input and output data lines are split between the two chips. Both receive the same 10-bit address and the common CS and RW control inputs.

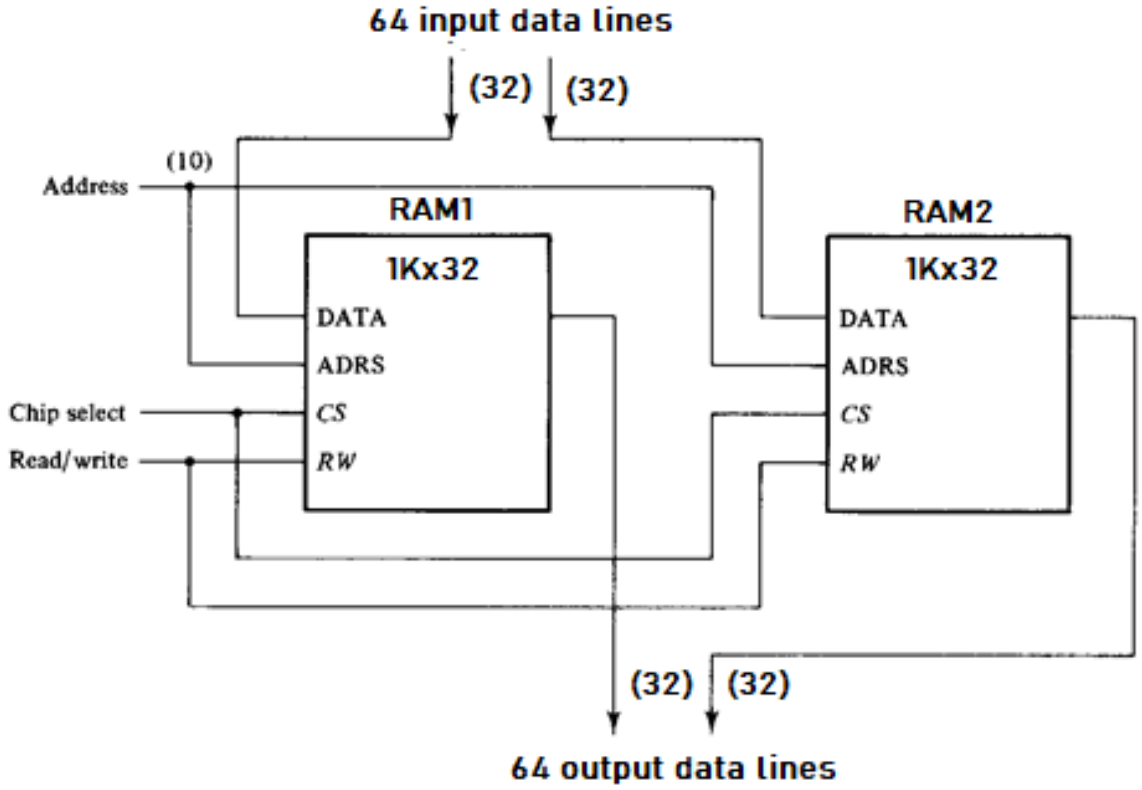


FIGURE
Block diagram of 1Kx64 RAM

1(a) (10 pts.) Given the following partial modules for two 1Kx32 RAM chips, fill in the blanks to complete the modules.

```
module RAM1_1Kx32 (adrs, CS, RW, idata, odata);
      input CS, RW;
      input [9:0] adrs;
      input [31:0] idata;
      output [31:0] odata;
      reg [31:0] d_out;
      reg [31:0] Mem1 [0:1023] ;
      assign odata = (CS && RW) ? d_out : 32'bz;
      always @ (adrs or idata or CS or RW)
            if (CS && !RW)
                  Mem1 [adrs] = idata ;
      always @ (adrs or CS or RW)
            if (CS && RW)
                  d_out = Mem1[adrs] ;
endmodule
```

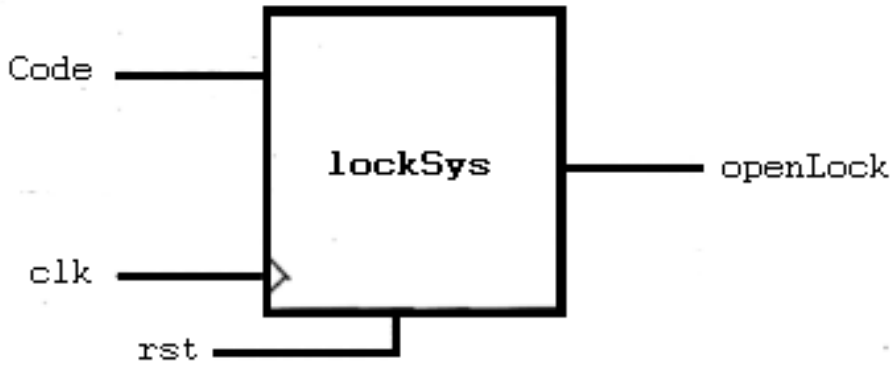
```
module RAM2_1Kx32 (odata, CS, RW, idata, adrs);
      input CS, RW;
      reg [31:0] Mem2 [0:1023] ;
      input [9:0] adrs;
      input [31:0] idata;
      output [31:0] odata;
      reg [31:0] d_out;
      assign odata = (CS && RW) ? d_out : 32'bz;
      always @ (adrs or idata or CS or RW)
            if (CS && !RW)
                  Mem2 [adrs] = idata ;
      always @ (adrs or CS or RW)
            if (CS && RW)
                  d_out = Mem2[adrs] ;
endmodule
```

1(b) (10 pts.) Write a top-level module to combine the two 1Kx32 RAM chips (given in 1(a)) to form a 1Kx64 RAM.

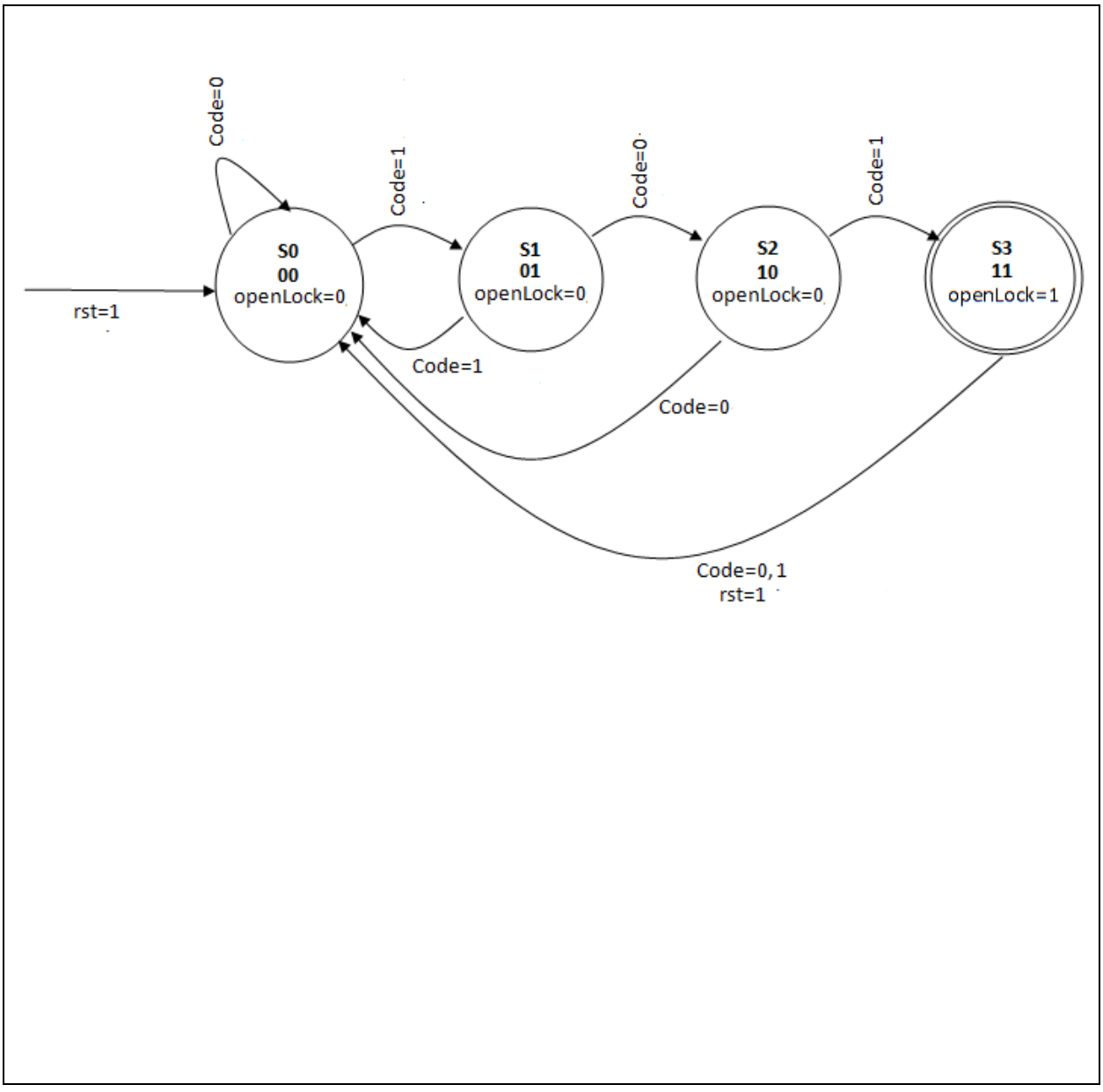
```
module RAM_1Kx64 (adrs, CS, RW, idata, odata);

      input CS, RW;
      input [9:0] adrs;
      input [63:0] idata;
      output [63:0] odata;

      RAM1_1Kx32 rc1 (adrs, CS, RW, idata[63:32], odata[63:32]);
      RAM2_1Kx32 rc2 (odata[31:0], CS, RW, idata[31:0], adrs);









endmodule
```

Problem 2.....(20 pts.)

In this problem, design an electronic lock system (**lockSys**, see Figure) for a garage door lock. The electronic lock accepts a 3-bit user code input, one bit at a time. If the input code sequence exactly matches 101, the electronic lock is opened (**openLock** is asserted). If any part of the 3-bit code input sequence is incorrect, the user is forced to restart code entry i.e. all backward arcs go back to the initial state if a "wrong" bit is entered. When **openLock** is asserted after the correct code has been entered, the lock stays open. And the lock shuts itself (**openLock** is de-asserted) if the asynchronous input signal **rst** is asserted or a 0/1 is entered after the correct input code sequence until the correct input code is again entered.



2(a) (10 pts.) Design a Moore FSM for **lockSys.**

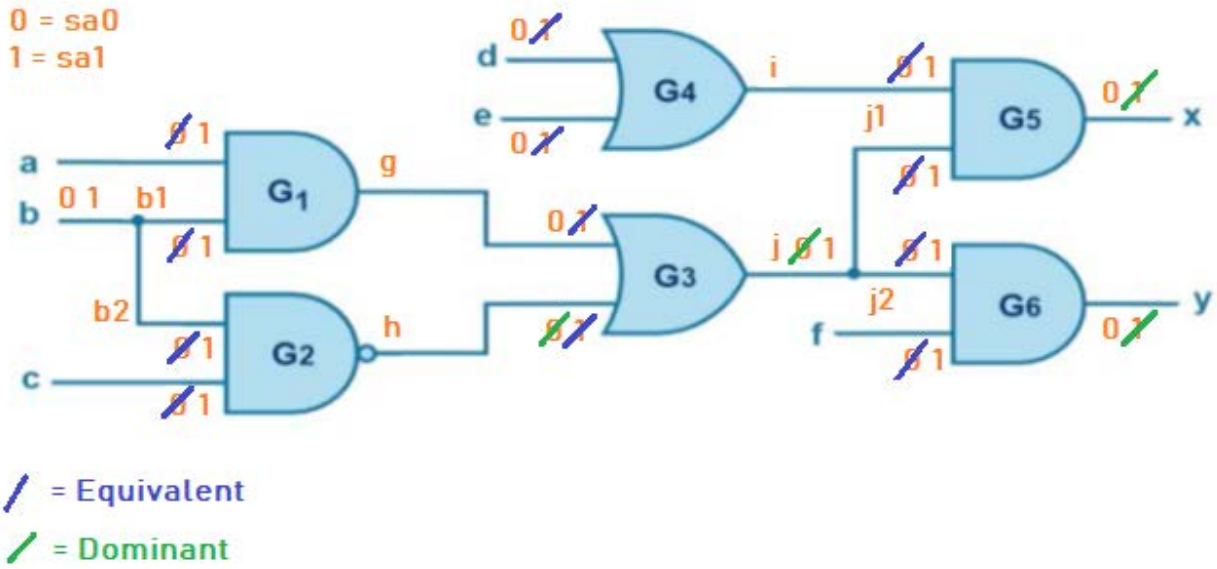


2(b) (10 **pts.**) Implement the FSM in **2(a)** in Verilog. Use the same standard format as was presented in the class. (Define your states; use one **always** block for next state; use one **always** block for state transitions; **assign** statement for output.)

```
module sysLock (Code, clk, rst, openLock);

     input Code, clk, rst;
     output openLock;

     parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
     reg [1:0] PS, NS;

     always @(posedge clk, rst)
          if (rst) begin
               openLock <= 0;
               PS <= S0; end
          else
               PS <= NS;


     always @(PS, Code)
          case (PS)
               S0: NS = Code?S1:S0;
               S1: NS = Code?S0:S2;
               S2: NS = Code?S3:S0;
               S3: NS = Code?S0:S0;
          endcase


     assign openLock = (PS==S3);

endmodule
```

Problem 3.....(30 pts.)

Consider the below given circuit based on the single stuck-at fault assumption and answer the questions related to this.



3(a) (2 pts.) What is the number of all possible faults?

Total fault sites = #PIs + #Fanouts + #Gates
= 6 + 4 + 6 = 16
Total faults = 16 x 2 = 32

3(b) (3 pts.) How many checkpoint faults are in the circuit?

Total checkpoints = #PIs + #Fanouts
= 6 + 4 = 10
Total checkpoint faults = 10 x 2 = 20

3(c) (15 pts.) Write the reduced fault list using the method of fault equivalence and fault dominance reduction. What is the collapse ratio after you perform fault collapsing (based on the equivalent and dominant faults you find)?

RFL = {a sa1, b sa0, b sa1, b1 sa1, b2 sa1, c sa1, d sa0, e sa0, f sa1, g sa0, i sa1, j sa1, j1 sa1, j2 sa1, x sa0, y sa0}

CR = 16/32 = 50%

3(d) (10 pts.) Label the sequence of assignments made by the D algorithm to generate a test for the **sa1** fault on **line c** indicated below. Be sure to give the order and show the values for all nodes in the circuit.