Solution

*Name:__________________________________

*Registration:__________________________

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design
CSE 308

Finalterm Examination Spring 2023

22 June 2023, Duration: 120 Minutes

Exam Rules

Please read carefully before proceeding.

- This exam is CLOSED books/notes/Internet/laptops/phones.
- No calculators/phones of any kind are allowed.
- Attempt all problems on the answer sheet.
- 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 6 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.)**

Good luck!

PROBLEM 1 (MEMORIES)..........................(20 pts., CLO-2)

RAM chips are available in a variety of sizes. If the memory unit needed for an application is larger than the capacity of one chip, it is necessary to combine a number of chips in an array to form the required memory size. The capacity of the memory depends on two parameters: the number of words and the number of bits per word. An increase in the number of words requires that we increase the address length. Every bit added to the length of the address doubles the number of words in memory. The increase in the number of bits per word requires that we increase the length of the data input and output lines, but the address length remains the same.

Suppose that we have many 2Kx16 RAM chips available with us but we need a memory of size 4Kx16 for our application. This can be constructed with two 2Kx16 RAM chips as shown in the Figure below. The 16 input data lines go to both the chips. The outputs must be ORed together to form the common 16 output data lines. (The OR gates are not shown in the diagram.) The 4K word memory requires a 12-bit address. The 11 least significant bits of the address are applied to the address inputs of both the chips.

The most significant bit is applied to a 1x2 decoder. The two outputs of the decoder are applied to the CS inputs of each chip. The memory is disabled when the memory-enable input of the decoder is equal to 0. This causes both the outputs of the decoder to be in the 0 state and none of the chips are selected. When the decoder is enabled, address bit 12 determines the particular chip that is selected. If bit 12 is equal to 0, the first RAM chip is selected. The remaining eleven address bits select a word within the chip in the range from 0 to 2047. The next 2048 words are selected from the second RAM chip with a 12-bit address that starts with 1 and follows by the eleven bits from the common address lines. The address range for each chip is listed in decimal over its block diagram in the Figure.

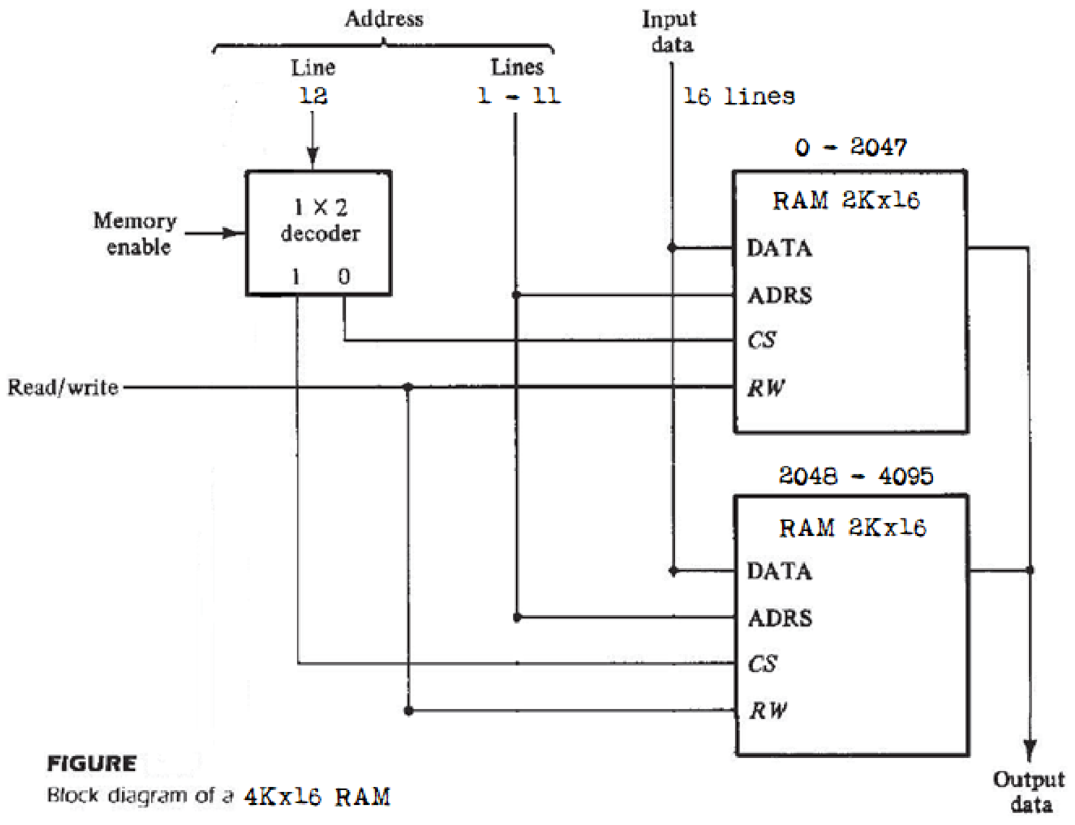


FIGURE
Block diagram of a 4Kx16 RAM

1(a) (10 pts.) Given the following partial modules for two 2Kx16 RAM chips, fill in the blanks to complete the modules.

```
 module RAM1_2Kx16 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [10:0] addr;
     input [15:0] idata;
     output [15:0] odata;
     reg [15:0] d_out;
     reg [15:0] Mem1 [0:2047];

     assign odata = (CS && RW)?d_out:16'b0;

     always @(addr or idata or CS or RW)
          if (CS && !RW)
               Mem1 [addr] = idata;
     always @(addr or CS or RW)
          if (CS && RW)
               d_out = Mem1 [addr];

     initial
          $readmemh ("memory1.dat", Mem1);

endmodule
```

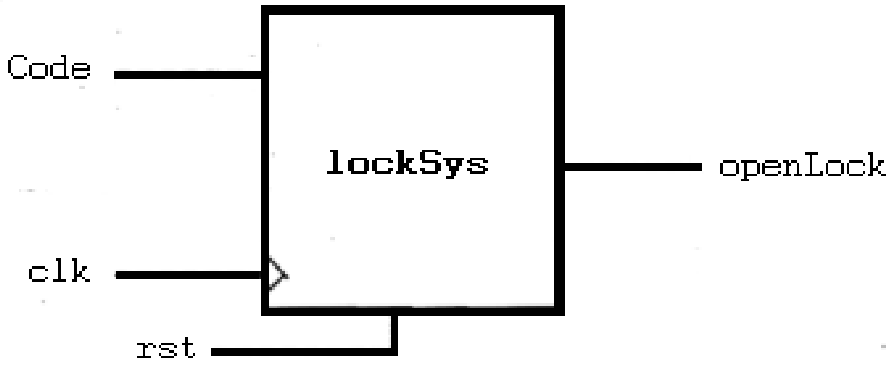
```
module RAM2_2Kx16 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [10:0] addr;
     input [15:0] idata;
     output [15:0] odata;
     reg [15:0] d_out;
     reg [15:0] Mem2 [0:2047];

     assign odata = (CS && RW)?d_out:16'b0;

     always @(addr or idata or CS or RW)
          if (CS && !RW)
               Mem2 [addr] = idata;
     always @(addr or CS or RW)
          if (CS && RW)
               d_out = Mem2 [addr];

     initial
          $readmemh ("memory2.dat", Mem2);

endmodule
```

1(b) (10 pts.) Write a top-level module to combine the two 2Kx16 RAM chips (given in **1(a)**) to form a 4Kx16 RAM. Use the below skeleton module.

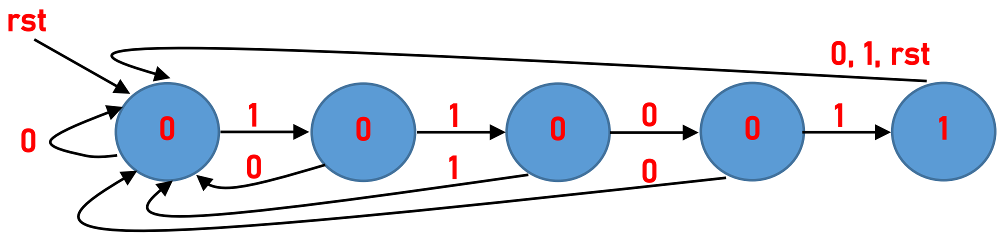
```
module RAM_4Kx16 (addr, CS, RW, idata, odata);

     input CS, RW;
     input [11:0] addr;
     input [15:0] idata;
     output [15:0] odata;
     wire [15:0] odata1, odata2;

     wire D0, D1;

     assign D0 = CS && ~addr[11]; //Output D0 of the 2x1 decoder
     assign D1 = CS && addr[11];  //Output D1 of the 2x1 decoder

     RAM1_2Kx16 rc1 (addr[10:0], D0, RW, idata, odata1);
     RAM2_2Kx16 rc2 (addr[10:0], D1, RW, idata, odata2);

     assign odata = (CS && RW)?odata1 | odata2:16'bz;


endmodule
```

PROBLEM 2 (FINITE STATE MACHINES)............(20 pts., CLO-2)

In this problem, design an electronic lock system (**lockSys**, see Figure) for a garage door lock. The electronic lock accepts a 4-bit user code input, one bit at a time. If the input code sequence exactly matches 1101, the electronic lock is opened (**openLock** is asserted). If any part of the 4-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 synchronous 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**.

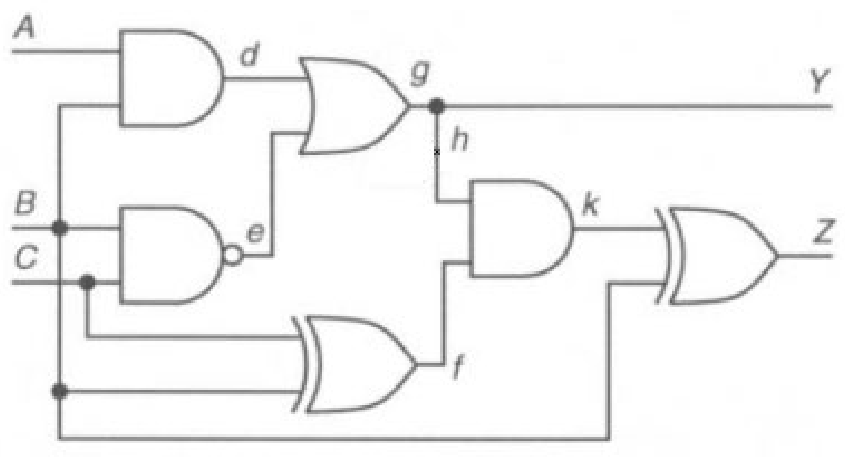


2(b) (10 pts.) Implement the FSM in **2(a)** in Verilog. Use the below skeleton module and the same standard format as was presented in the class. (Define your states; use one **always** block for next state; use one **always** block for state transitions; **assign** statement for output.)

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

     input Code, clk, rst;
     output openLock;
     reg openLock;

     parameter A = 0, B = 1, C = 2, D = 3, E=4;
     reg [2:0] PS, NS;

     always @(posedge clk)
          if (rst)
               PS <= A;
          else
               PS <= NS;

     always @(Code, PS)
          case (PS)
               A: NS = Code?B:A;
               B: NS = Code?C:A;
               C: NS = Code?A:D;
               D: NS = Code?E:A;
               E: NS = Code?A:A;
               default: NS = A;
          endcase

     assign openLock = (PS==E);

endmodule
```

PROBLEM 3 (TESTING)...........................(30 pts., CLO-3)

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



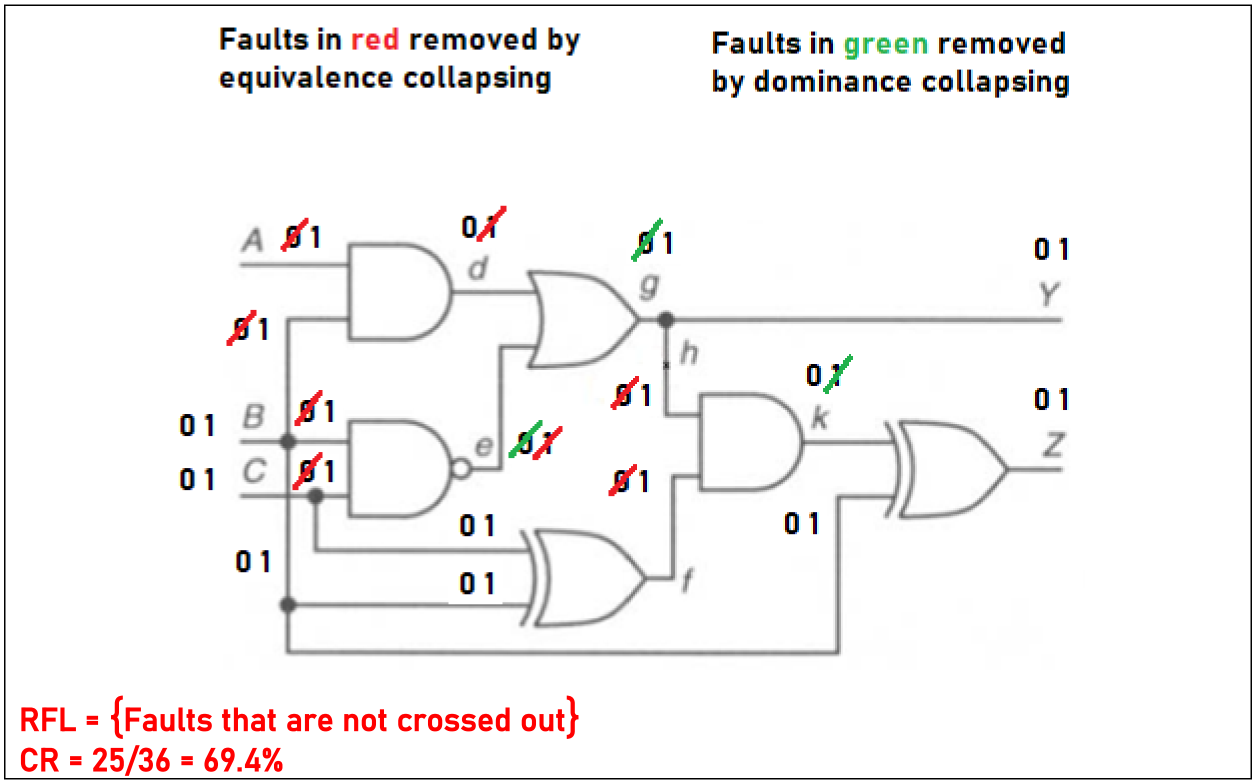
3(a) (2 pts.) What is the number of all possible faults?

Total fault sites = #PIs + #Fanouts + #Gates
= 3 + 9 + 6 = 18
Total faults = 18 x 2 = 36

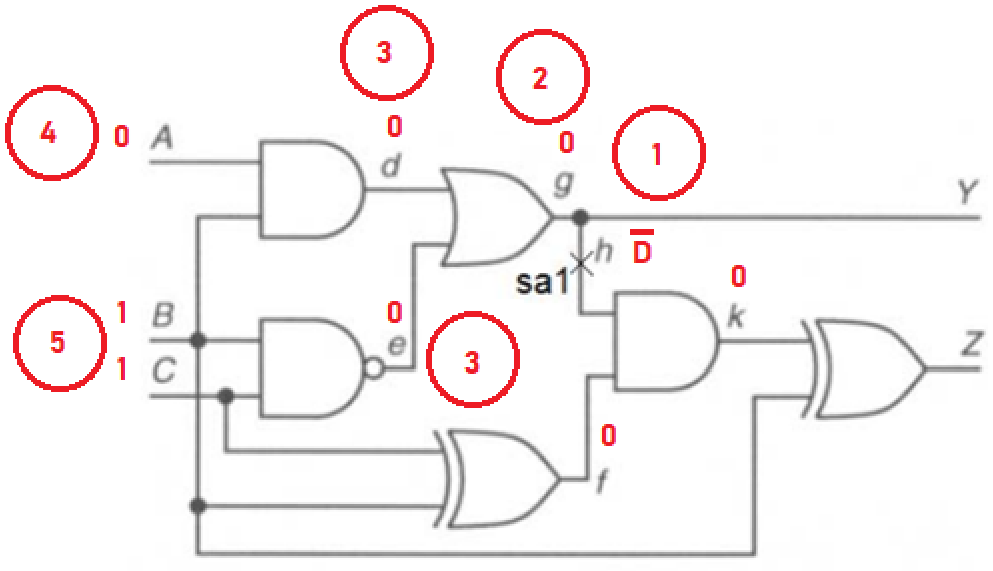
3(b) (3 pts.) How many checkpoint faults are in the circuit?

Total checkpoints = #PIs + #Fanouts
= 3 + 9 = 12
Total faults = 12 x 2 = 24

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)?


Faults in red removed by equivalence collapsing
Faults in green removed by dominance collapsing
A
d
g
h
Y
k
Z
B
C
e
f
0 1
RFL = {Faults that are not crossed out}
CR = 25/36 = 69.4%

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



It is not possible to generate a test for h sa1, as f=0 blocks the fault.