University of Tehran
College of Engineering
School of Electrical and Computer Engineering

Digital Systems 1

Dr.Navabi

Homework 8

Soroush Mesforush Mashhad

SN:810198472

Khordad 00

Question 1

The system verilog code for the wanted upcounter is as follows:

```
`timescale 1ns/1ns
module Q18BitUpCounter(inout [7:0] PIO,input clk,rst,cnt,ld,ci,oe,output co);
    logic[7:0] LogInt;
    always@(negedge clk,negedge rst)begin
        if(~rst)
           LogInt<=8'd0;
           else begin
                if(ld)
                   LogInt<=PIO;
                        else if (cnt) LogInt<=ci ? (LogInt+1):LogInt;
           end
     end
     assign PIO=oe?LogInt:8'bz;
     assign co=(cnt&oe)? &{PIO,ci}:1'b0;
endmodule
```

Figure 1: 8-bit upcounter

Question 2

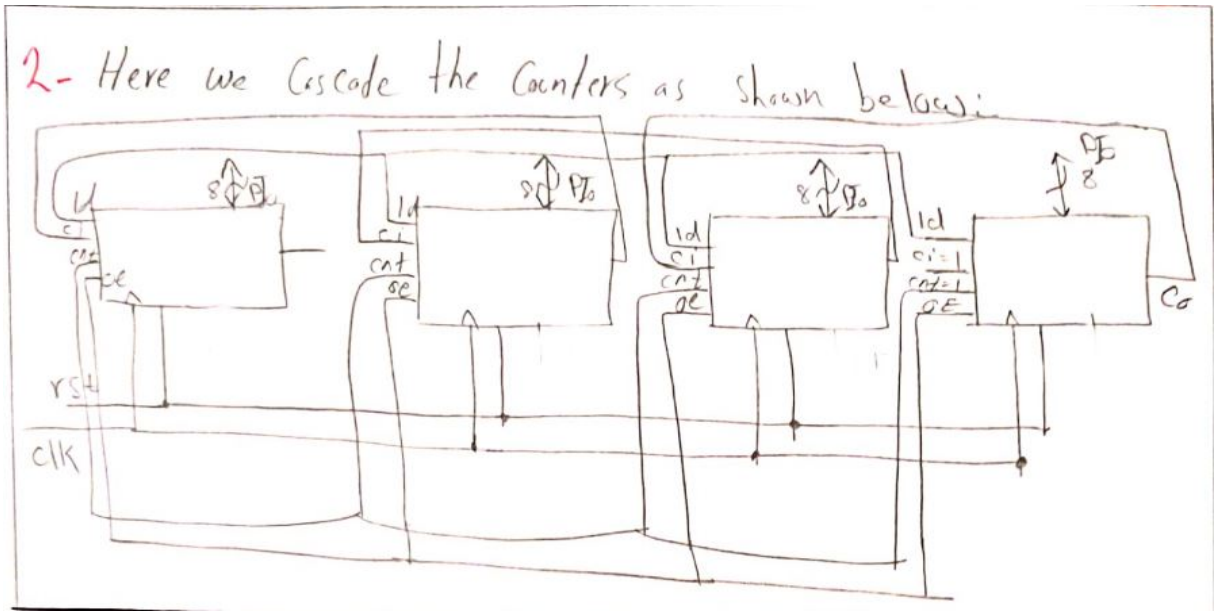


Figure 2: Cascading of counters

Question 3

The system verilog code for the wanted shift register is as follows:

```
`timescale 1ns/1ns
module Q38BitShiftRegister(input [7:0] PI,input clk,rst,shen,ld,si,output logic [7:0] PO,output so);
    assign so=PO[0];
    always@(negedge clk,negedge rst)begin
      if(~rst)
          PO<=8'd0;
      else if(ld)
         PO<=PI;
      else
        PO<=(shen)?{si,PO[7:1]}:PO;
     end
endmodule
```

Figure 3: 8-bit shift register

Question 4

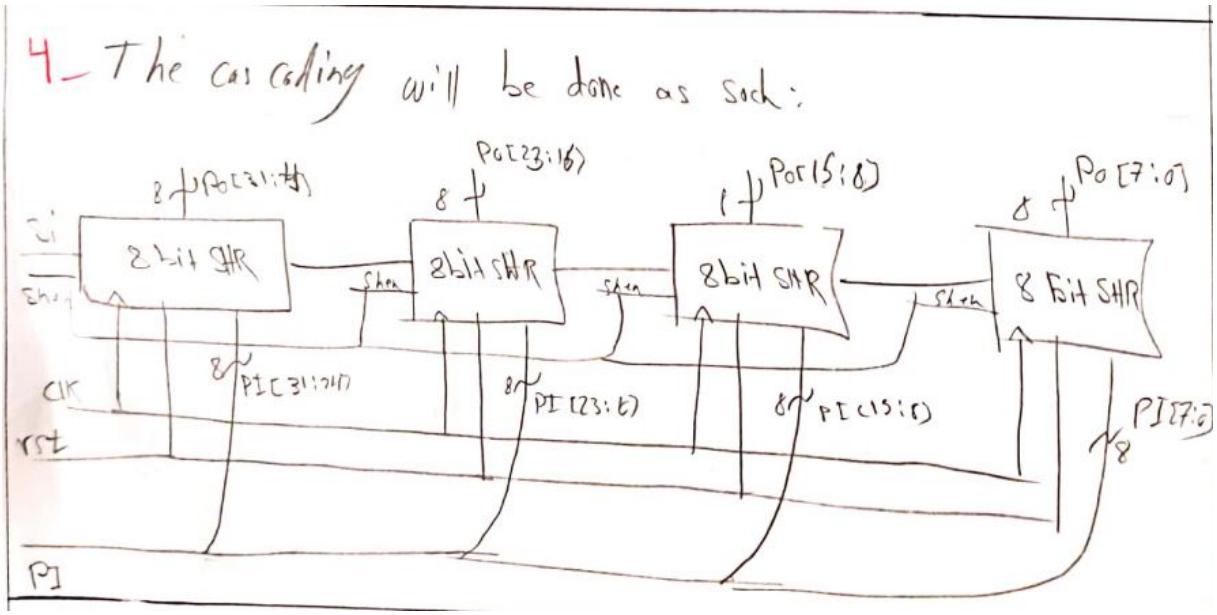
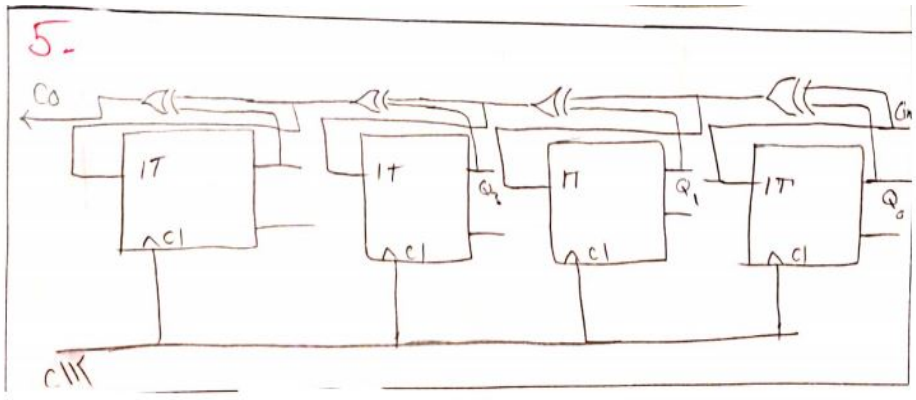


Figure 4: Cascading Of Shift Registers

Question 5



We assume that the initial values are 0000

So if we put cin=0 we have

0000 ⟶ 0000 ⟶ 0000 ⟶ Nothing happens, counting doesn't occur.

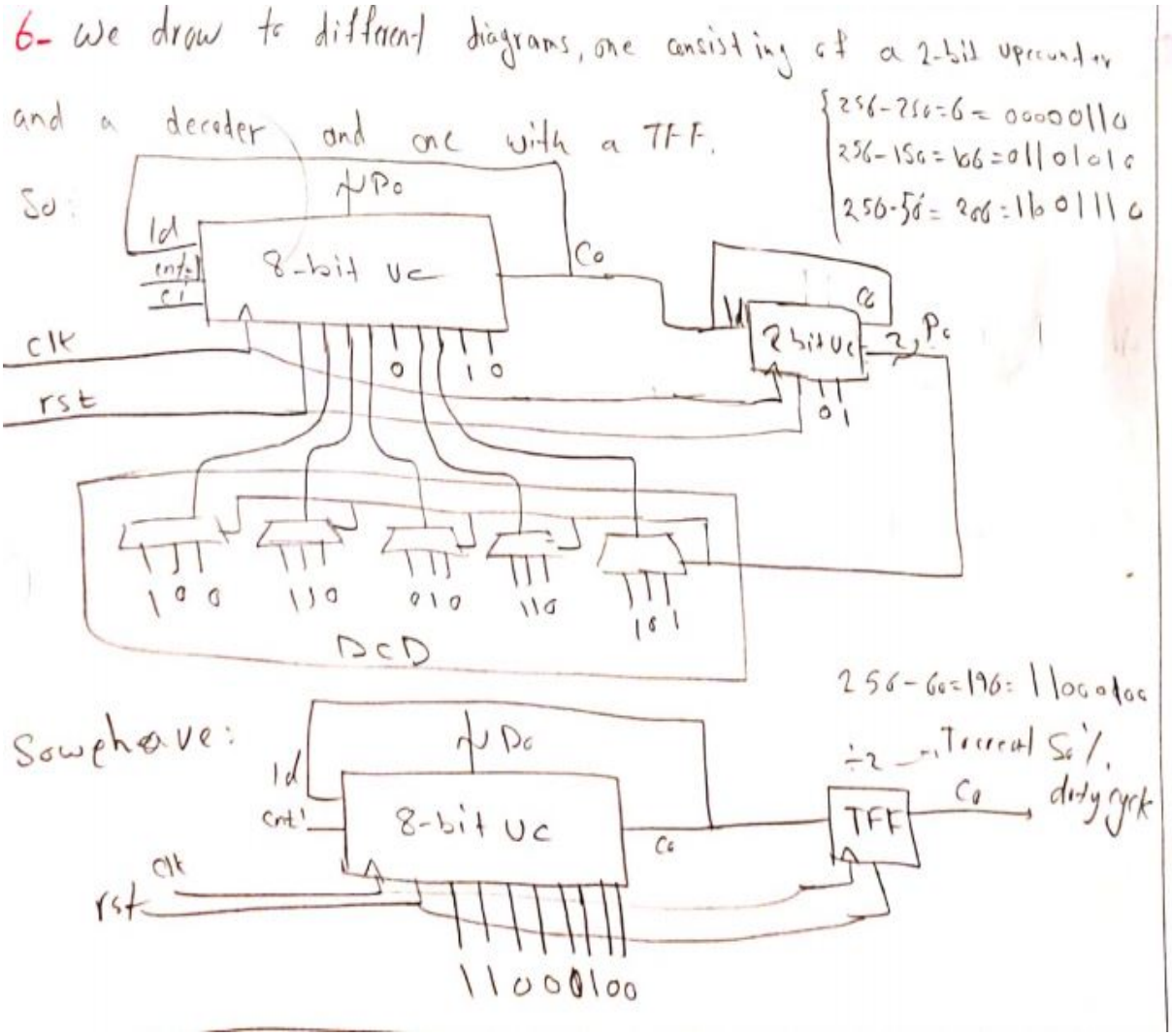
if Cin=1 we shall have:

0000 ⟶ 1111 ⟶ 1110 ⟶ 1001 ⟶ 1000

⟶ 0111 ⟶ 0010 ⟶ 0001 ⟶ 0000

⟶ So after some transitions we get back to the starting point

Question 6


6- we draw to different diagrams, one consisting of a 2-bit upcounter and a decoder and one with a TFF.
So:
8-bit uc
2 bit uc
DCD
So we have:
8-bit uc
TFF
clk
rst
ld
cnt
Co
dirty cyrk

Question 7

The system verilog code for the wanted upcounter and the synthesis results are as follows:

```
`timescale 1ns/1ns
module Q18BitUpCounter(input [3:0] PI,input clk,rst,cnt,ld,ci,output logic[3:0] PO,output co);
    always@(posedge clk,posedge rst)begin
        if(rst)
           PO<=4'd0;
           else begin
                if(ld)
                   PO<=PI;
                        else if (cnt) PO<=ci ? (PO+1):PO;
           end
     end
     assign co=(cnt)? &{PO,ci}:1'b0;
endmodule
```

Figure 5: 4-bit upcounter

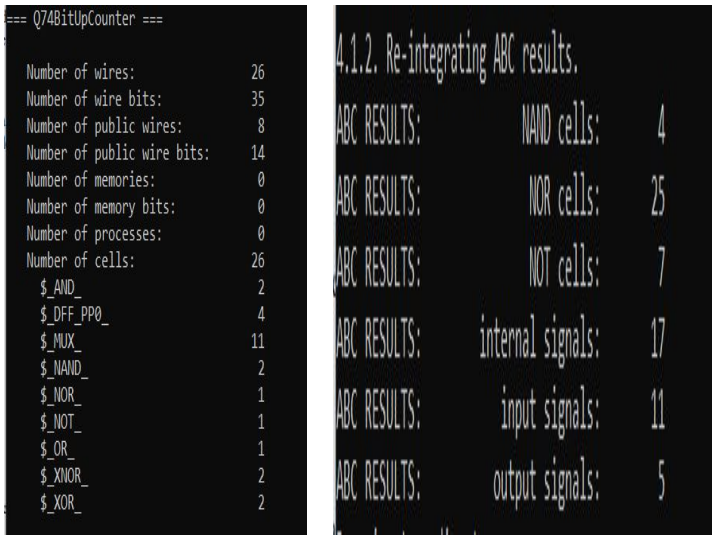


Figure 6: Synthesis

As we can see 4 DFFs are used which are what we expected, some other

needed gates such as NAND,NOR and Not have been made to implement our other needs.