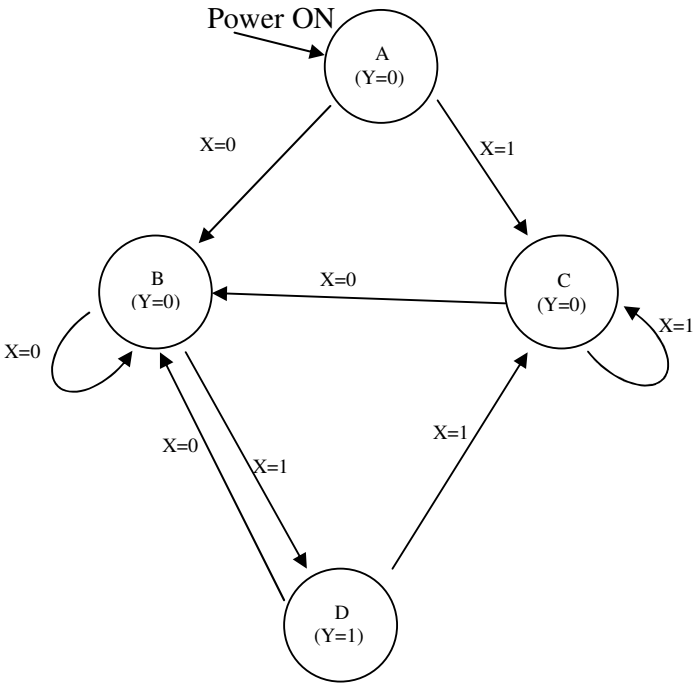
4-5-2008
BILKENT UNIVERSITY
Department of Electrical and Electronics Engineering
EEE102 Introduction to Digital Circuit Design
Midterm Exam II SOLUTION

Surname: _____
Name: _____
ID-Number: _____
Signature: _____

Duration is 120 minutes. Solve all 5 questions. Show all your work.

Q1 (20 points	
Q2 (20 points)	
Q3 (20 points)	
Q4 (20 points)	
Q5 (20 points)	
Total	

Q1. A synchronous FSM, with one input X and one output Y, has 4 states A, B, C, and D and has the following State/Output diagram. Design and draw this FSM using D flip flops with active low Preset and Clear controls. What does this FSM do?



Solution:

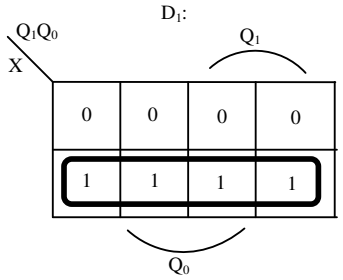
State encoding

State	Q_1	Q_0
A	0	0
B	0	1
C	1	0
D	1	1

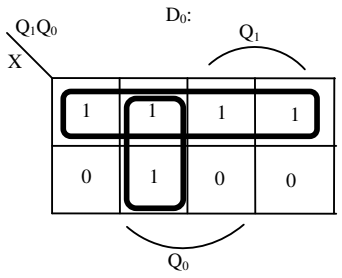
Next state and excitation table

Q_1	Q_0	X	Q_1*=D_1	Q_0*=D_0
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

Minimized functions for the next state logic



$D_1 = X$

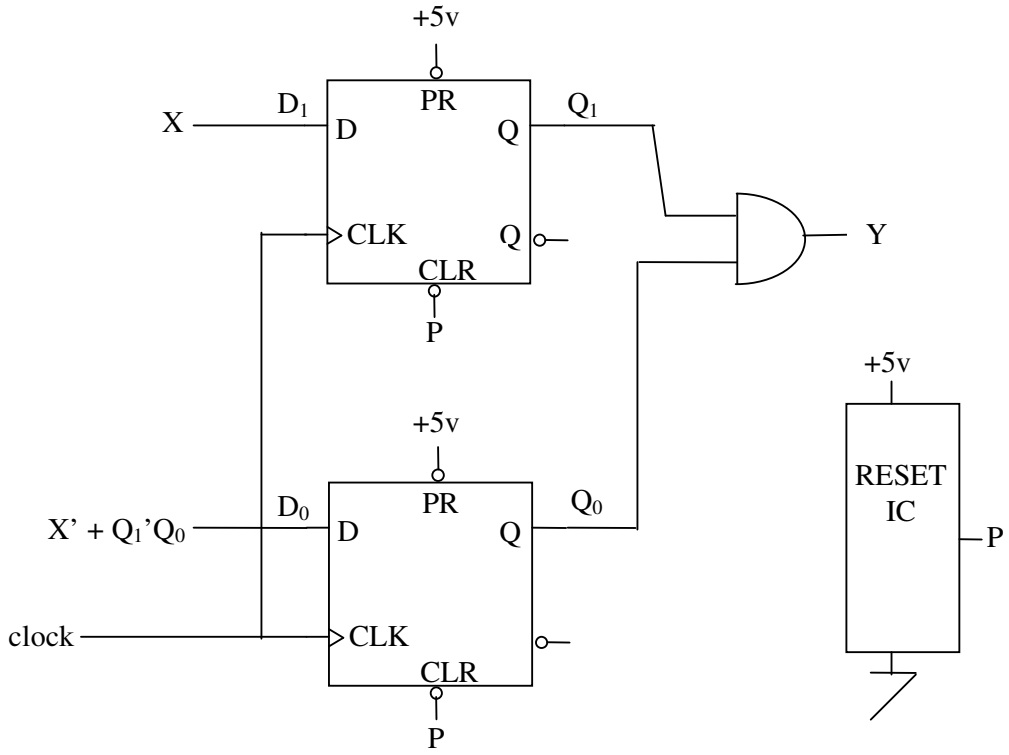


$D_0 = X' + Q_1'Q_0$

Output table

Q_1	Q_0	Y
0	0	0
0	1	0
1	0	0
1	1	1

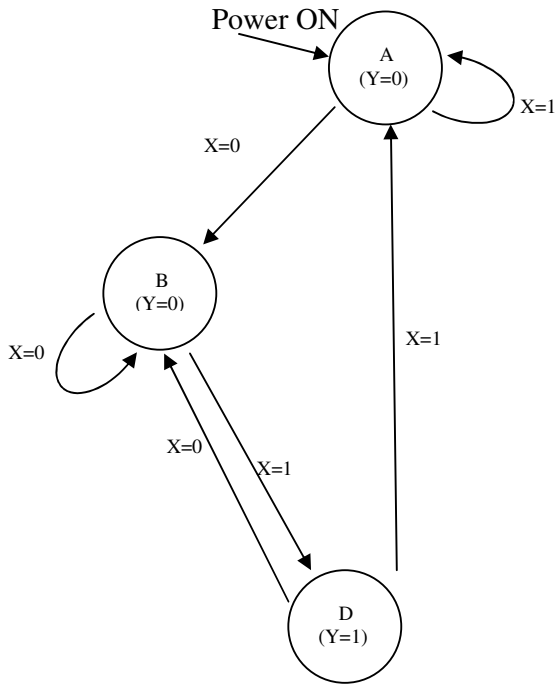
$Y = Q_1Q_0$



This FSM makes its output 1 when the last received X is 1 and the previously received X is 0. Thus it detects the "01" sequence in the input.

Alternative solution:

It is observed that states A and C are equivalent. Therefore we can draw the State/Output diagram again.



State	Q_1	Q_0
A	0	0
B	0	1
D	1	0

Q_1	Q_0	X	Q_1*=D_1	Q_0*=D_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	d	d
1	1	1	d	d

Q_1	Q_0	Y
0	0	0
0	1	0
1	0	1
1	1	d

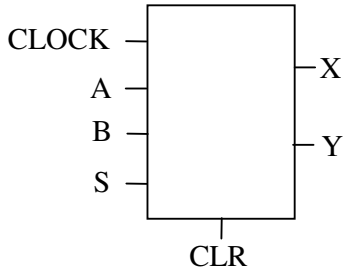
Minimized functions: $D_1 = Q_0X$, $D_0 = X$, $Y = Q_1$

Q2. A circuit, shown below as a block, realizes the following functions:

(i) At the rising edge of the CLOCK signal
it loads A to X and B to Y if S = 0, and
it loads B to X and A to Y if S = 1.

(ii) If the asynchronous CLR input is 1 then it makes X = 0 and Y = 0.

"Asynchronous" means "independent of CLOCK edges". Write VHDL code to describe this circuit.



Solution:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AdvancedDff is
   Port ( CLOCK : in std_logic;
          CLR : in std_logic;
          A : in std_logic;
          B : in std_logic;
```

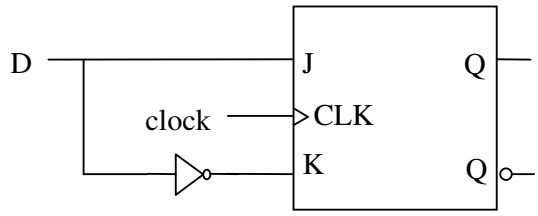
```
        S : in std_logic;
        X : out std_logic;
        Y : out std_logic);
end AdvancedDff;

architecture Behavioral of AdvancedDff is
begin
process (CLOCK,CLR)
        begin
                if CLR='1' then X<='0';Y<='0';
                elsif CLOCK'event and CLOCK='1' then
                        case S is
                                when '0' => X<=A;Y<=B;
                                when '1' =>     X<=B;Y<=A;
                                when others => X<=A;Y<=B;
                        end case;
                end if;
end process;
end Behavioral;
```

Q3. a) Draw how you can obtain a D flip flop using a single JK flip flop and minimum number of additional simple gates.

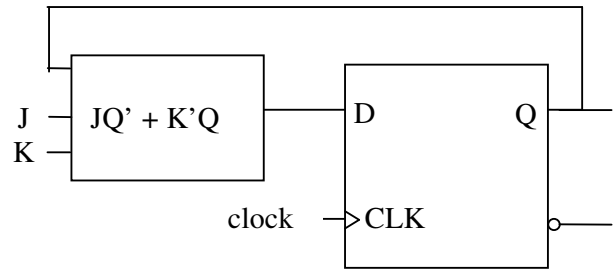
b) Draw how you can obtain a JK flip flop using a single D flip flop and minimum number of additional simple gates.

Solution:

a)



b)



Q4. Implement F = AB + C'

a) using a generic 8-to-1 multiplexer and minimum number of additional simple gates,

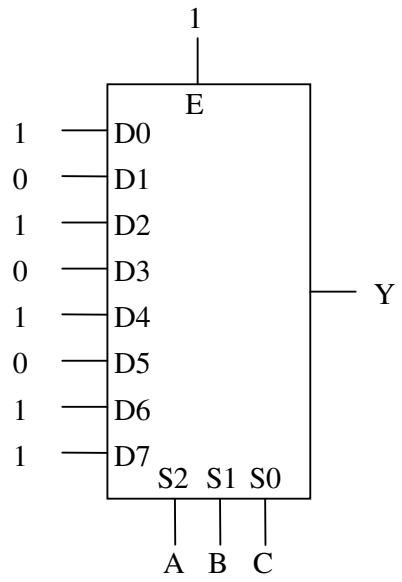
b) using a generic 4-to-1 multiplexer and minimum number of additional simple gates,

c) using a generic 2-to-1 multiplexer and minimum number of additional simple gates,

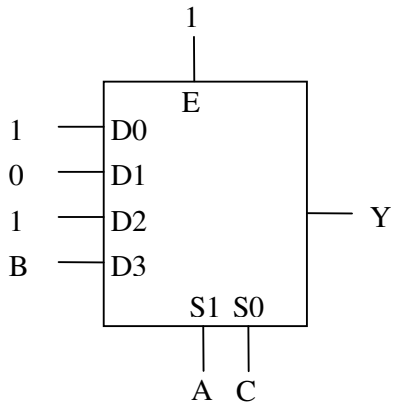
d) using one 74XX138 decoder and minimum number of additional simple gates. (note that 74XX138 is a 3-to-8 binary decoder with active low outputs and with three enables, one active high and two active low),
and
e) using two 2-to-4 generic decoders and minimum number of additional simple gates.
Solution:

A	B	C	AB+C'
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

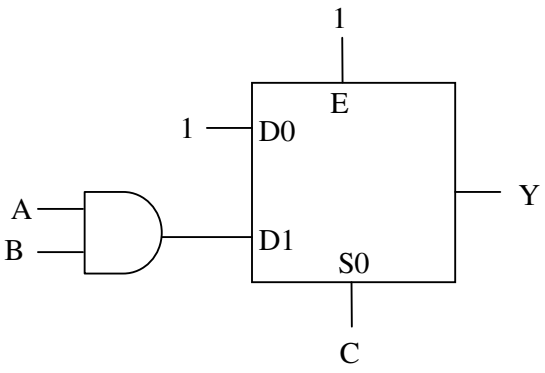
a) using a generic 8-to-1 multiplexer



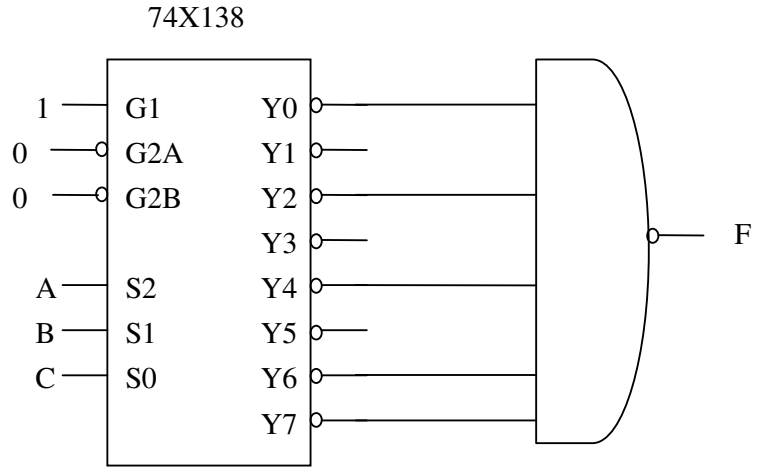
b) using a generic 4-to-1 multiplexer



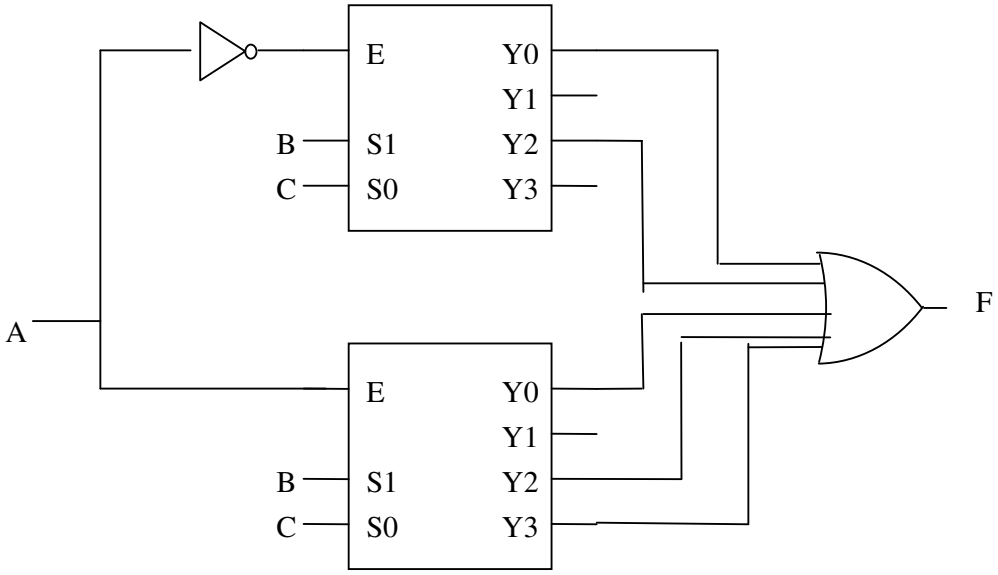
c) using a generic 2-to-1 multiplexer



d) using a 74XX138 decoder.



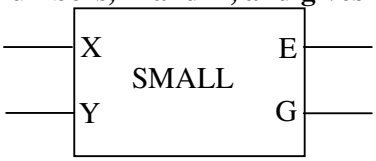
e) using two 2-to-4 generic decoders.



Q5. We need to design a comparator (called BIG) which has two 2-bit binary numbers as input, A and B, and two outputs EQ (meaning "equal") and GT (meaning "greater") such that the following table is implemented:

Condition	EQ	GT
A = B	1	0
A > B	0	1
A < B	0	0

We have already designed a simple comparator (called SMALL) which receives two 1-bit numbers, X and Y, and gives outputs E and G such that



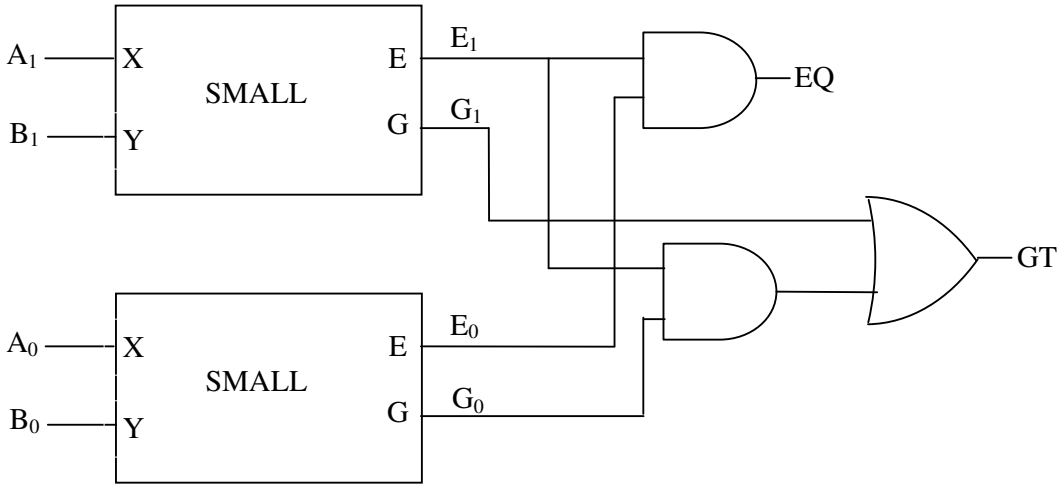
Condition	E	G
X = Y	1	0
X > Y	0	1
X < Y	0	0

a) (5 points) Use two SMALLs and minimum number of additonal simple gates to design one BIG. Draw your circuit.

b) (15 points) Assume now that the VHDL code for SMALL has already been written and included in the library. Write VHDL code for BIG using two SMALLs as components.

Solution:

a)



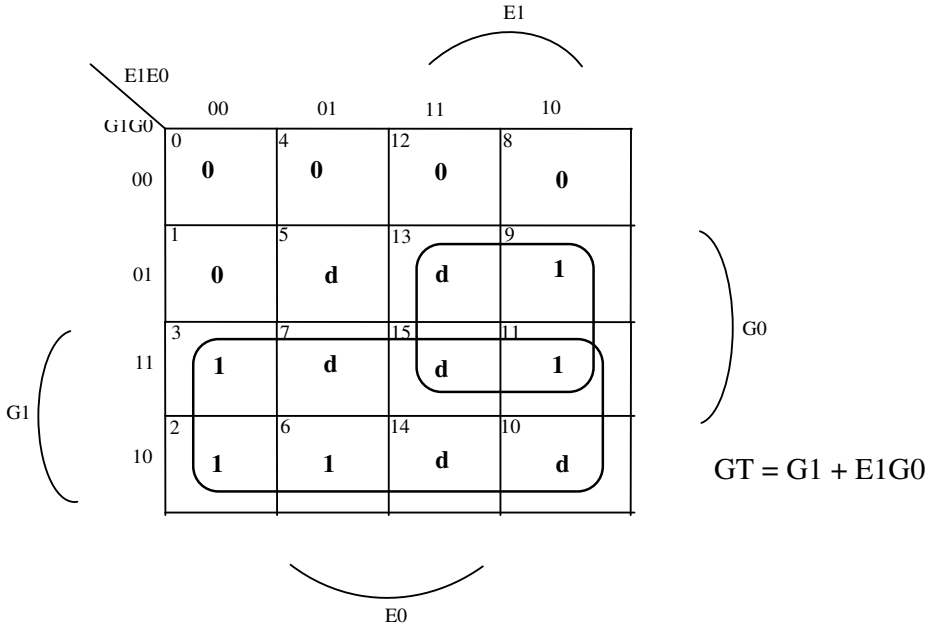
The logic behind this solution is as follows: EQ is 1 if and only if $A_1 = B_1$ and $A_0 = B_0$.
GT is 1 if $A_1 > B_1$ (in this case we do not have to compare A_0 and B_0),
or $A_1 = B_1$ and $A_0 > B_0$.
Thus $GT = G_1 + E_1G_0$

More systematic solution:

A1	B1	A0	B0		E1	E0	G1	G0		GT
0	0	0	0		1	1	0	0		0
0	0	0	1		1	0	0	0		0
0	0	1	0		1	0	0	1		1
0	0	1	1		1	1	0	0		0
0	1	0	0		0	1	0	0		0
0	1	0	1		0	0	0	0		0

0	1	1	0		0	0	0	1	0
0	1	1	1		0	1	0	0	0
1	0	0	0		0	1	1	0	1
1	0	0	1		0	0	1	0	1
1	0	1	0		0	0	1	1	1
1	0	1	1		0	1	1	0	1
1	1	0	0		1	1	0	0	0
1	1	0	1		1	0	0	0	0
1	1	1	0		1	0	0	1	1
1	1	1	1		1	1	0	0	0

E1	E0	G1	G0		GT
0	0	0	0		0
0	0	0	1		0
0	0	1	0		1
0	0	1	1		1
0	1	0	0		0
0	1	0	1		d
0	1	1	0		1
0	1	1	1		d
1	0	0	0		0
1	0	0	1		1
1	0	1	0		d
1	0	1	1		1
1	1	0	0		0
1	1	0	1		d
1	1	1	0		d
1	1	1	1		d



GT = G1 + E1G0

b)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity BIG is
   Port ( A : in std_logic_vector(1 downto 0);
          B : in std_logic_vector(1 downto 0);
          EQ : out std_logic;
          GT : out std_logic);
end BIG;

architecture Behavioral of BIG is
component SMALL
Port ( X : in std_logic;
       Y : in std_logic;
       E : out std_logic;
       G : out std_logic);
end component;
signal E1,E0,G1,G0:std_logic;
begin
C1:SMALL port map(A(1),B(1),E1,G1);
C2:SMALL port map(A(0),B(0),E0,G0);
EQ <= E1 and E0;
GT <= G1 or (E1 and G0);
end Behavioral;
```

You do not have to write the code for SMALL but I have included it below for your information:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity SMALL is
   Port ( X : in std_logic;
          Y : in std_logic;
          E : out std_logic;
          G : out std_logic);
end SMALL;
architecture Behavioral of SMALL is
begin
process(X,Y)
begin
          if X=Y then E<='1';G<='0';
          elsif X>Y then E<='0';G<='1';
          else E<='0';G<='0';
          end if;
end process;
end Behavioral;
```