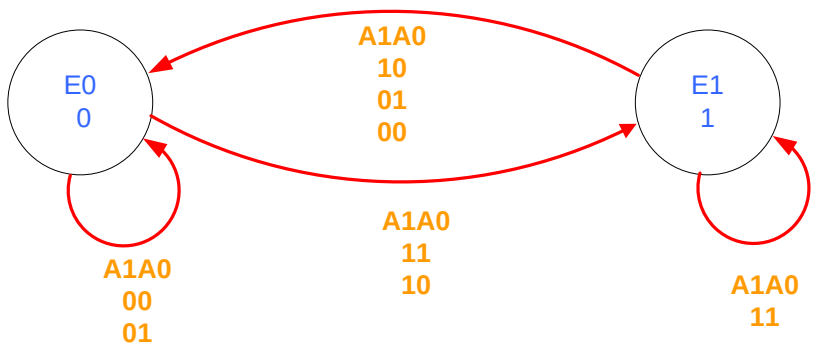
Soluciones Tema 5

Ejercicio 1.

$$D_0 = A_1 \cdot A_0 + A_1 \cdot \overline{Q_0}$$

Entadas		Estado Actual		Salidas	Estado siguiente		
A1	A0	Q0	Estado	O0	D0	$Q0^+$	Estado
0	0	0	E0	0	0	0	E0
0	0	1	E1	1	0	0	E0
0	1	0	E0	0	0	0	E0
0	1	1	E1	1	0	0	E0
1	0	0	E0	0	1	1	E1
1	0	1	E1	1	0	0	E0
1	1	0	E0	0	1	1	E1
1	1	1	E1	1	1	1	E1



Ejercicio 2.

$J_0 = A \cdot Q_1$

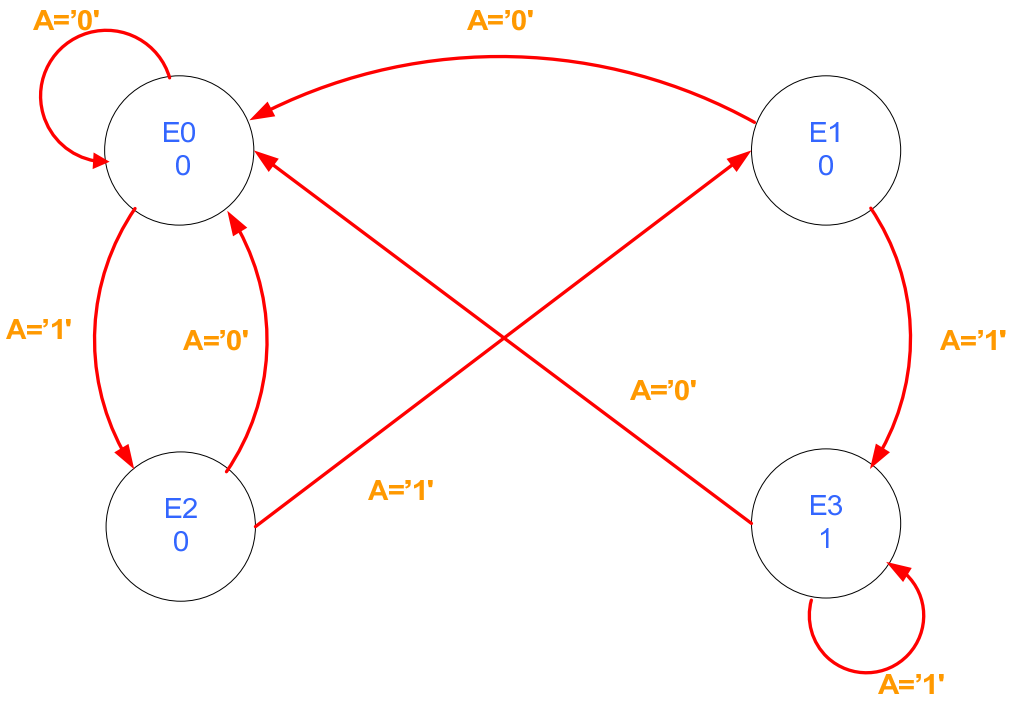
$K_0 = \overline{A}$

$J1 = A$

$K1 = \overline{Q_0} + \overline{A}$

$S = Q_1 \cdot Q_0$

Entadas	Estado Actual			Salidas	Estado siguiente						
A	Q1	Q0	Estado	S	J1	K1	J0	K0	$Q1^+$	$Q0^+$	Estado
0	0	0	E0	0	0	1	0	1	0	0	E0
0	0	1	E1	0	0	1	0	1	0	0	E0
0	1	0	E2	0	0	1	0	1	0	0	E0
0	1	1	E3	1	0	1	0	1	0	0	E0
1	0	0	E0	0	1	1	0	0	1	0	E2
1	0	1	E1	0	1	0	0	0	1	1	E3
1	1	0	E2	0	1	1	1	0	0	1	E1
1	1	1	E3	1	1	0	1	0	1	1	E3



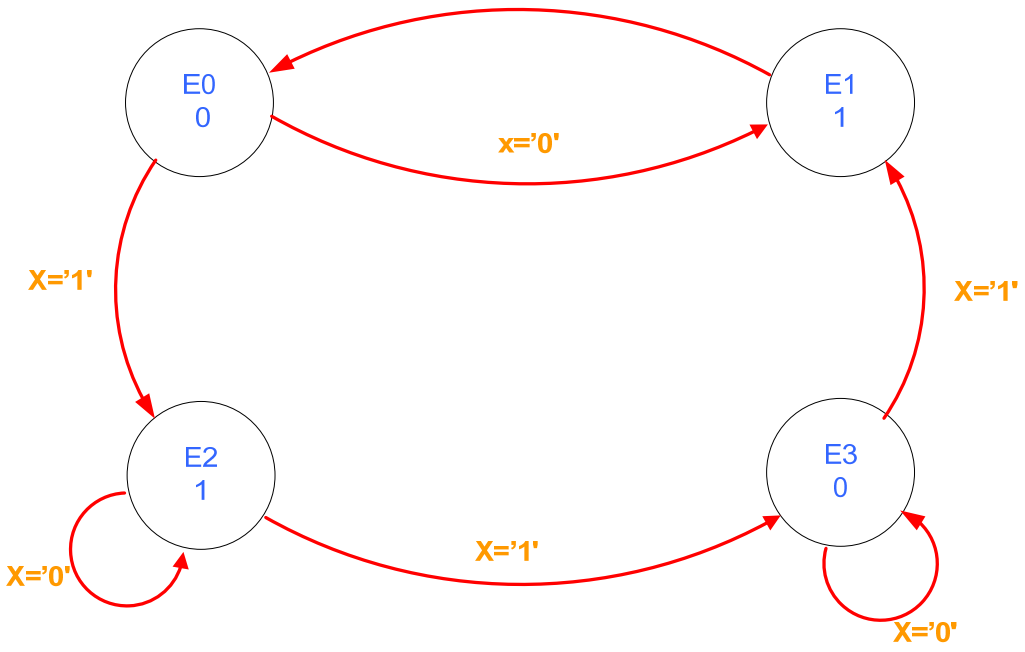
Ejercicio 3.

$$D_0 = \overline{Q_1}\cdot\overline{Q_0}\cdot\overline{X} + Q_1\cdot\overline{Q_0}\cdot X + Q_1\cdot Q_0$$

$$D_1 = \overline{Q_1}\cdot\overline{Q_0}\cdot X + Q_1\cdot\overline{Q_0} + Q_1\cdot Q_0\cdot\overline{X}$$

$$Y = Q_1 \oplus Q_0$$

Entadas	Estado Actual			Salidas	Estado siguiente				
X	Q1	Q0	Estado	Y	D1	D0	$Q1^+$	$Q0^+$	Estado
0	0	0	E0	0	0	1	0	1	E1
0	0	1	E1	1	0	0	0	0	E0
0	1	0	E2	1	1	0	1	0	E2
0	1	1	E3	0	1	1	1	1	E3
1	0	0	E0	0	1	0	1	0	E2
1	0	1	E1	1	0	0	0	0	E0
1	1	0	E2	1	1	1	1	1	E3
1	1	1	E3	0	0	1	0	1	E1



Ejercicio 4.

a)

Entadas	Estado Actual			Salidas	Estado siguiente				
X	Q1	Q0	Estado	S	D1	D0	$Q1^+$	$Q0^+$	Estado
0	0	0	E0	0	0	0	0	0	E0
0	0	1	E1	0	1	0	1	0	E2
0	1	0	E2	1	1	0	1	0	E2
0	1	1	E3	1	0	0	0	0	E0
1	0	0	E0	0	0	1	0	1	E1
1	0	1	E1	0	0	1	0	1	E1
1	1	0	E2	1	1	1	1	1	E3
1	1	1	E3	1	1	1	1	1	E3

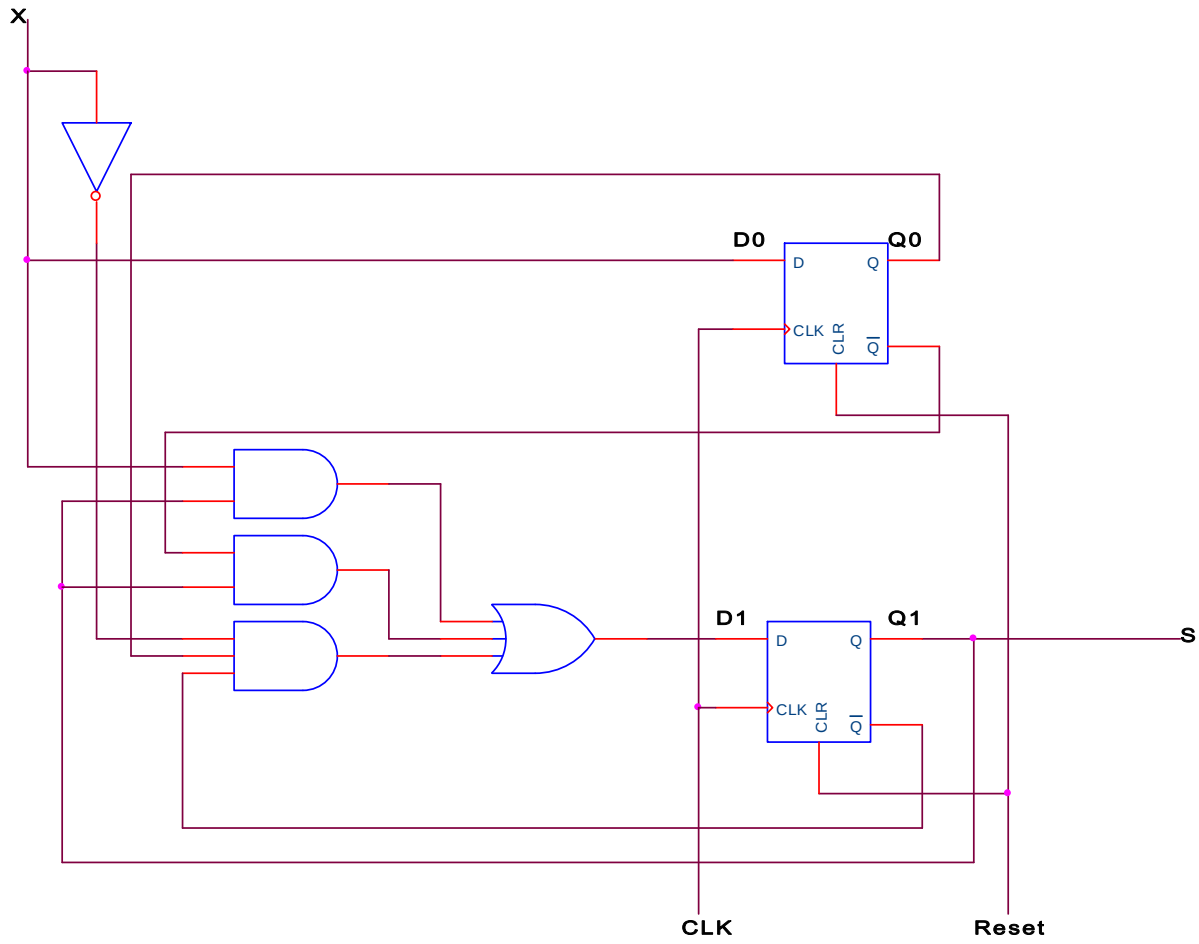
Q_1Q_0 / X	00	01	11	10
0	0	0	0	0
1	1	1	1	1

Q_1Q_0 / X	00	01	11	10
0	0	1	0	1
1	0	0	1	1

$D_0 = X$

$D_1 = \overline{X}\cdot\overline{Q_1}\cdot Q_0 + Q_1\cdot\overline{Q_0} + X\cdot Q_1$

$S = Q_1$



b)

Entadas	Estado Actual			Salidas	Estado siguiente				
UP/$\overline{DOWN}$	Q1	Q0	Estado	S3S2S1S0	D1	D0	$Q1^+$	$Q0^+$	Estado
0	0	0	E0	0101	1	1	1	1	E3
0	0	1	E1	0110	0	0	0	0	E0
0	1	0	E2	0111	0	1	0	1	E1
0	1	1	E3	1000	1	0	1	0	E2
1	0	0	E0	0101	0	1	0	1	E1
1	0	1	E1	0110	1	0	1	0	E2
1	1	0	E2	0111	1	1	1	1	E3
1	1	1	E3	1000	0	0	0	0	E0

Q_1Q_0 / UP/$\overline{DOWN}$	00	01	11	10
0	1	0	0	1
1	1	0	1	1

Q_1Q_0 / UP/$\overline{DOWN}$	00	01	11	10
0	1	0	1	0
1	0	1	0	1

$D_0 = \overline{Q_0}$

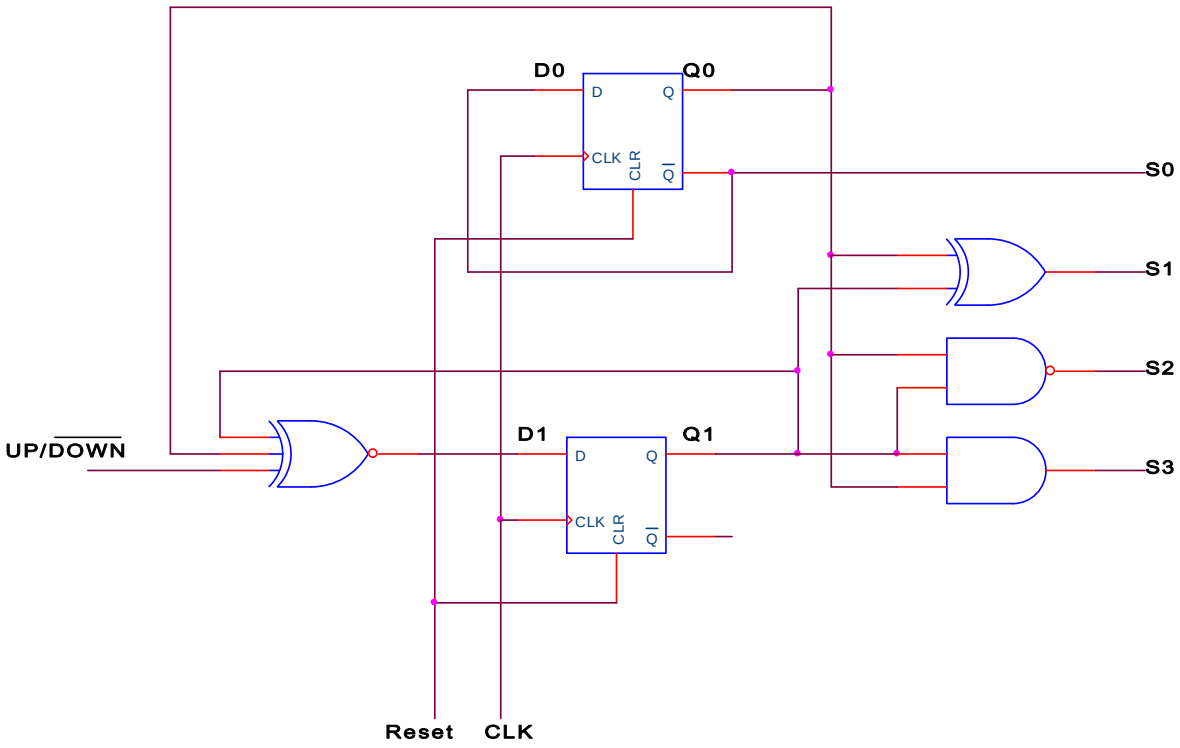
$D_1 = \overline{UP/\overline{DOWN} \oplus Q_1 \oplus Q_0}$

$S_0 = \overline{Q_0}$

$S_1 = Q_1 \oplus Q_0$

$S_2 = \overline{Q_1} + \overline{Q_0}$

$S_3 = Q_1 \cdot Q_0$



Ejercicio 5.

a)

Entadas	Estado Actual			Salidas	Estado siguiente						
X	Q1	Q0	Estado	S	J1	K1	J0	k0	$Q1^+$	$Q0^+$	Estado
0	0	0	E0	1	0	X	0	X	0	0	E0
0	0	1	E1	0	0	X	X	0	0	1	E1
0	1	0	E2	0	X	0	0	X	1	0	E2
1	0	0	E0	1	0	X	1	X	0	1	E1
1	0	1	E1	0	1	X	X	1	1	0	E2
1	1	0	E2	0	X	1	0	X	0	0	E0

Q_1Q_0 / X	00	01	11	10
0	X	0	X	X
1	X	1	X	X

$K_0 = X$

Q_1Q_0 / X	00	01	11	10
0	0	X	X	0
1	1	X	X	0

$J_0 = X \cdot \overline{Q_1}$

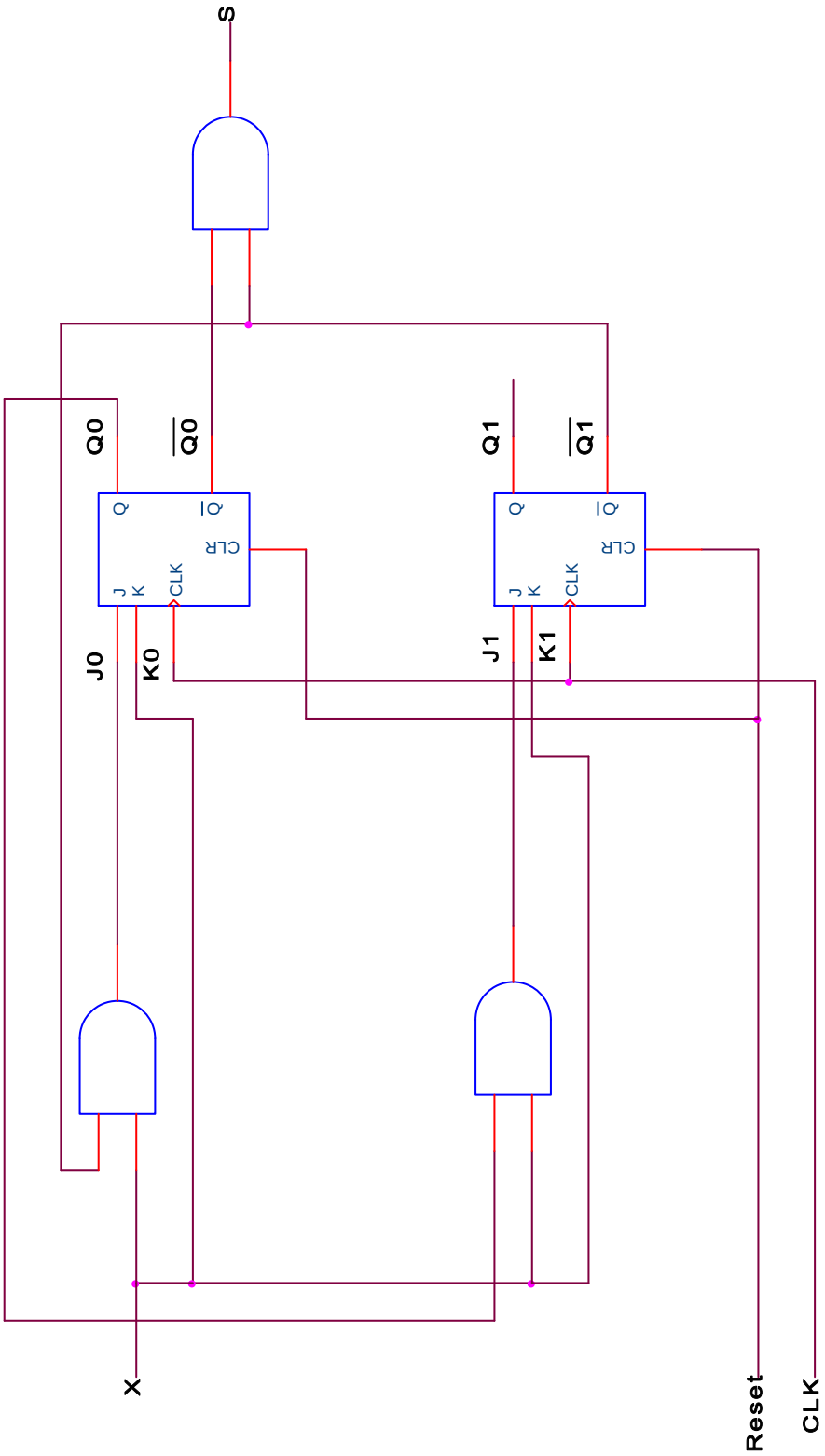
Q_1Q_0 / X	00	01	11	10
0	X	X	X	0
1	X	X	X	1

$K_1 = X$

Q_1Q_0 / X	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_1 = X \cdot Q_0$

$S = \overline{Q_1} \cdot \overline{Q_0}$


S
Q0
$\overline{Q0}$
Q1
$\overline{Q1}$
Q
IQ
CLR
J
K
CLK
J0
K0
J1
K1
X
Reset
CLK

b)

Entadas		Estado Actual			Salidas	Estado siguiente						
A	B	Q1	Q0	Estado	S	J1	K1	J0	K0	$Q1^+$	$Q0^+$	Estado
0	0	0	0	E0	0	0	X	0	X	0	0	E0
0	0	0	1	E1	0	1	X	X	0	1	1	E3
0	0	1	0	E2	0	X	1	0	X	0	0	E0
0	0	1	1	E3	1	X	0	X	0	1	1	E3
0	1	0	0	E0	0	0	X	0	X	0	0	E0
0	1	0	1	E1	0	0	X	X	1	0	0	E0
0	1	1	0	E2	0	X	0	1	X	1	1	E3
0	1	1	1	E3	1	X	0	X	0	1	1	E3
1	0	0	0	E0	0	0	X	1	X	0	1	E1
1	0	0	1	E1	0	1	X	X	0	1	1	E3
1	0	1	0	E2	0	X	1	0	X	0	0	E0
1	0	1	1	E3	1	X	1	X	1	0	0	E0
1	1	0	0	E0	0	1	X	0	X	1	0	E2
1	1	0	1	E1	0	0	X	X	1	0	0	E0
1	1	1	0	E2	0	X	0	1	X	1	1	E3
1	1	1	1	E3	1	X	1	X	1	0	0	E0

A B \ Q_1Q_0	00	01	11	10
00	X	0	0	X
01	X	1	0	X
11	X	1	1	X
10	X	0	1	X

$$K_0 = B\cdot\overline{Q_1} + A\cdot Q_1$$

A B \ Q_1Q_0	00	01	11	10
00	0	X	X	0
01	0	X	X	1
11	0	X	X	1
10	1	X	X	0

$$J_0 = A\cdot\overline{B}\cdot\overline{Q_1} + B\cdot Q_1$$

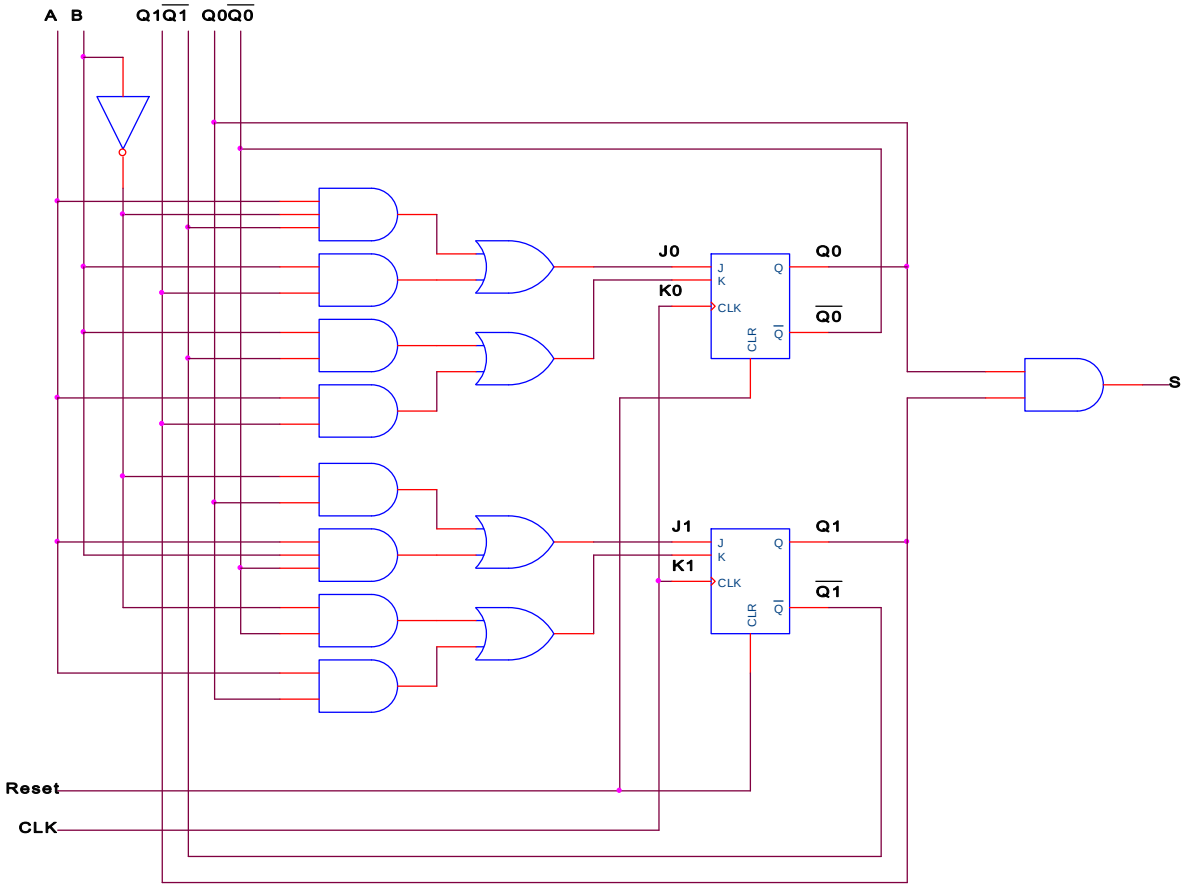
A B \ Q_1Q_0	00	01	11	10
00	X	X	0	1
01	X	X	0	0
11	X	X	1	0
10	X	X	1	1

$$K_1 = \overline{B}\cdot\overline{Q_0} + A\cdot Q_0$$

A B \ Q_1Q_0	00	01	11	10
00	0	1	X	X
01	0	0	X	X
11	1	0	X	X
10	0	1	X	X

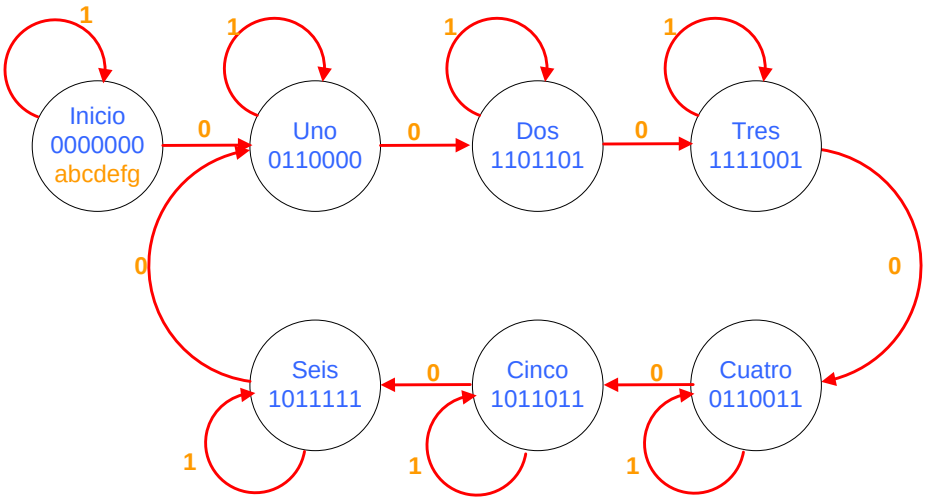
$$J_1 = A\cdot B\cdot\overline{Q_0} + \overline{B}\cdot Q_0$$

$$S = Q_1\cdot Q_0$$



Ejercicio 6.

a)



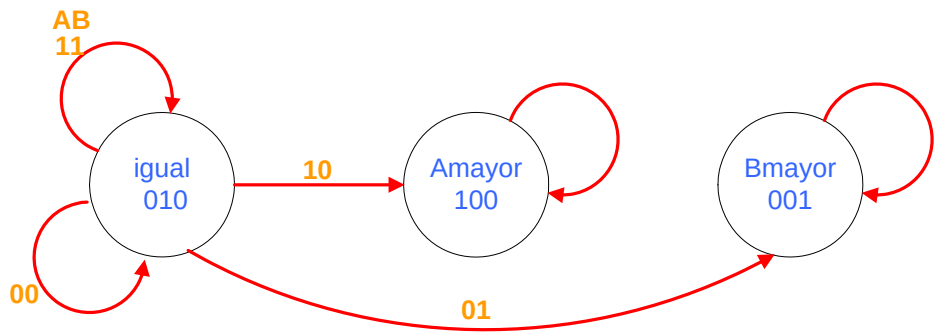
b)

```
----------------------------------------------------------------------
-- Dado electrónico
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dado is
    Port (  P, CLK, Reset : in std_logic;
                  display : out std_logic_vector (6 downto 0));
end dado;

architecture Behavioral of dado is
      --Definimos un tipo con los estados posibles
      type estados_posibles IS (inicio, uno, dos, tres, cuatro, cinco, seis);
      --Declaramos una señal que puede tomar cualquiera de los estados
      --posibles.
      signal estado: estados_posibles;
begin
      --Definimos un proceso sensible al reloj y al reset para modificar
      --el estado.
      process(CLK, Reset)
      begin
            --Reset asíncrono de nivel activo alto.
            if (Reset='1') then  estado <= inicio;
                  elsif (CLK 'event and CLK='1') then
                  case estado is
                        when inicio => -- Estado inicial.
                              if (P='0') then estado <= uno;
                              end if;
                        when uno => --Estado uno.
                              if (P='0') then estado <= dos;
                              end if;

                  when dos => --Estado dos.
                         if (P='0') then estado <= tres;
                        end if;
                  when tres => --Estado tres.
```

```
                        if (P='0') then estado <= cuatro;
                        end if;
                  when cuatro => --Estado cuatro.
                        if (P='0') then estado <= cinco;
                        end if;
                  when cinco => --Estado cinco.
                        if (P='0') then estado <= seis;
                        end if;
                  when seis => --Estado seis.
                         if (P='0') then estado <= uno;
                        end if;
            end case;
      end if;
end process;

--Ahora en función del estado, generamos la salida

display <= (others =>'0') when (estado= inicio) else
            "0110000" when (estado = uno ) else
            "1101101" when (estado = dos ) else
            "1111001" when (estado = tres ) else
            "0110011" when (estado = cuatro ) else
            "1011011" when (estado = cinco ) else
            "1011111";
end Behavioral;
```

Ejercicio 7.

a)



b)

Entadas		Estado Actual			Salidas			Estado siguiente				
A	B	Q1	Q0	Estado	AmayorB	AigualB	AmenorB	D1	D0	$Q1^+$	$Q0^+$	Estado
0	0	0	0	Igual	0	1	0	0	0	0	0	Igual
0	0	0	1	Amayor	1	0	0	0	1	0	1	Amayor
0	0	1	0	Bmayor	0	0	1	1	0	1	0	Bmayor
0	1	0	0	Igual	0	1	0	1	0	1	0	Bmayor
0	1	0	1	Amayor	1	0	0	0	1	0	1	Amayor
0	1	1	0	Bmayor	0	0	1	1	0	1	0	Bmayor
1	0	0	0	Igual	0	1	0	0	1	0	1	Amayor
1	0	0	1	Amayor	1	0	0	0	1	0	1	Amayor
1	0	1	0	Bmayor	0	0	1	1	0	1	0	Bmayor
1	1	0	0	Igual	0	1	0	0	0	0	0	Igual
1	1	0	1	Amayor	1	0	0	0	1	0	1	Amayor
1	1	1	0	Bmayor	0	0	1	1	0	1	0	Bmayor

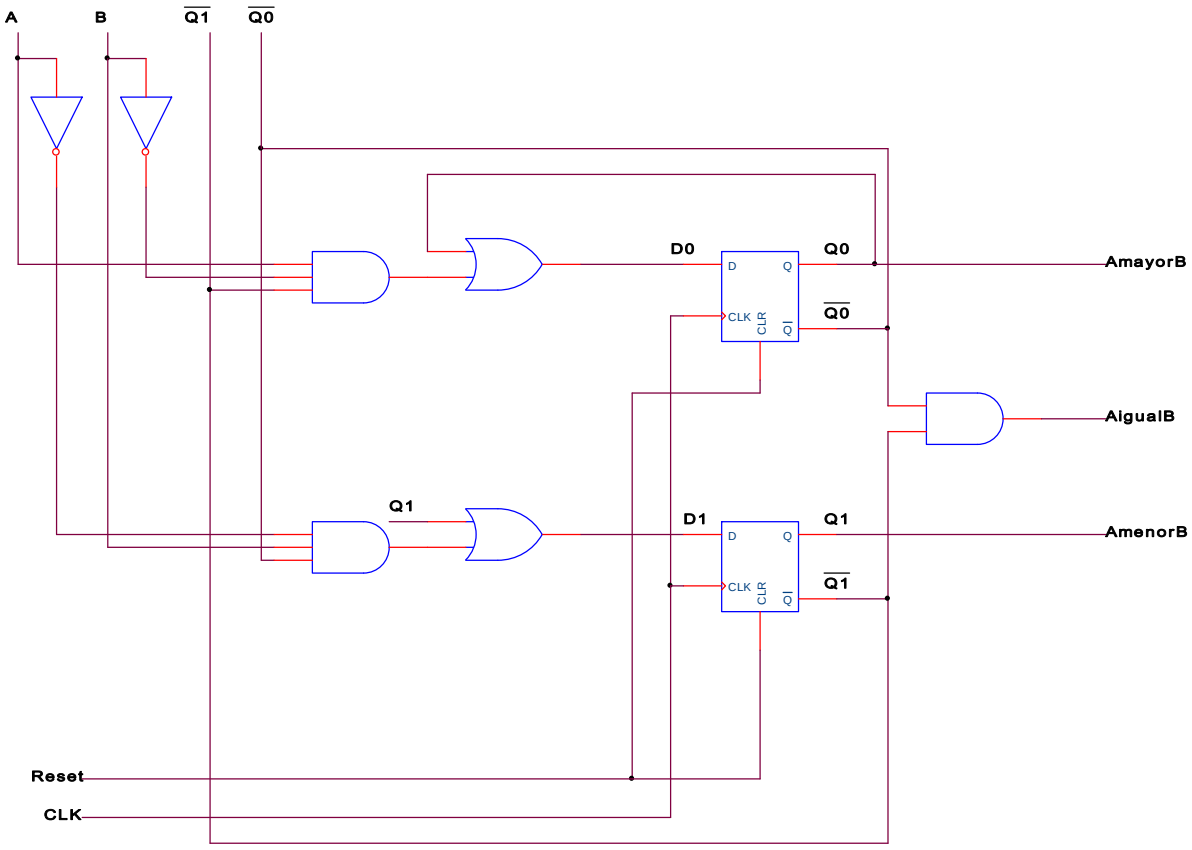
$\text{AmayorB} = Q_0$ $\quad \text{AigualB} = \overline{Q_1} \cdot \overline{Q_0}$ $\quad \text{AmenorB} = Q_1$

Q1Q0 / AB	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	0	1	X	0
10	1	1	X	0

$$D_0 = Q_0 + \overline{Q_1} \cdot A \cdot \overline{B}$$

Q1Q0 / AB	00	01	11	10
00	0	0	X	1
01	1	0	X	1
11	0	0	X	1
10	0	0	X	1

$$D_1 = Q_1 + \overline{Q_0} \cdot \overline{A} \cdot B$$


A
B
Q1
Q0
D0
Q0
D
Q
CLK
CLR
Q0
AmayorB
AigualB
Q1
D1
Q1
D
Q
CLK
CLR
Q1
AmenorB
Reset
CLK

Una variante con biestables o flip-flops tipo J-K

Entadas		Estado Actual			Salidas			Estado siguiente						
A	B	Q1	Q0	Estado	AmayorB	AigualB	AmenorB	J1	K1	J0	K0	$Q1^+$	$Q0^+$	Estado
0	0	0	0	Igual	0	1	0	0	X	0	X	0	0	Igual
0	0	0	1	Amayor	1	0	0	0	X	X	0	0	1	Amayor
0	0	1	0	Bmayor	0	0	1	X	0	0	X	1	0	Bmayor
0	1	0	0	Igual	0	1	0	1	X	0	X	1	0	Bmayor
0	1	0	1	Amayor	1	0	0	0	X	X	0	0	1	Amayor
0	1	1	0	Bmayor	0	0	1	X	0	0	X	1	0	Bmayor
1	0	0	0	Igual	0	1	0	0	X	1	X	0	1	Amayor
1	0	0	1	Amayor	1	0	0	0	X	X	0	0	1	Amayor
1	0	1	0	Bmayor	0	0	1	X	0	0	X	1	0	Bmayor
1	1	0	0	Igual	0	1	0	0	X	0	X	0	0	Igual
1	1	0	1	Amayor	1	0	0	0	X	X	0	0	1	Amayor
1	1	1	0	Bmayor	0	0	1	X	0	0	X	1	0	Bmayor

A B \ Q_1Q_0	00	01	11	10
00	X	0	X	X
01	X	0	X	X
11	X	0	X	X
10	X	0	X	X

K_0 ='0'

A B \ Q_1Q_0	00	01	11	10
00	0	X	X	0
01	0	X	X	0
11	0	X	X	0
10	1	X	X	0

$J_0 = A \cdot \overline{B} \cdot \overline{Q_1}$

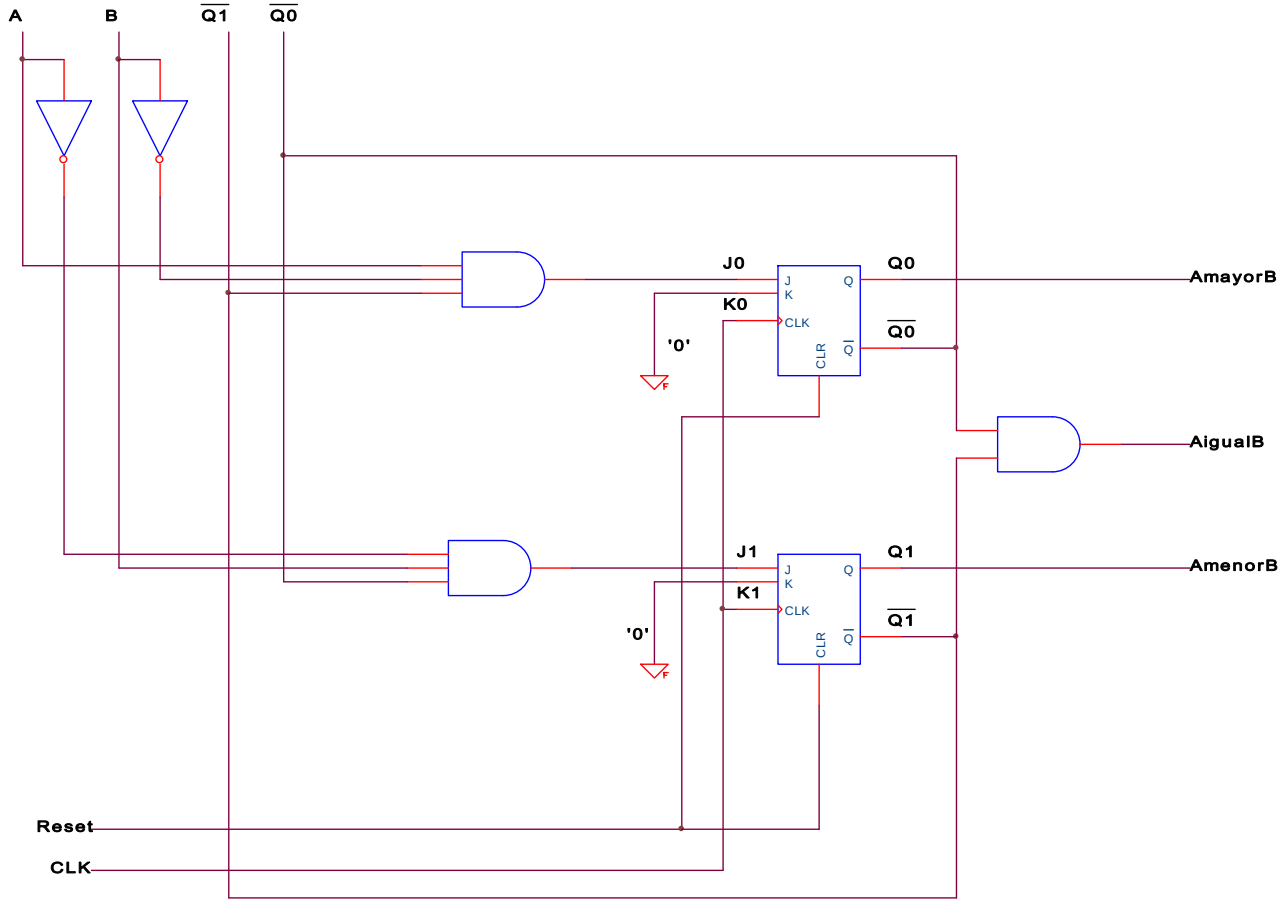
A B \ Q_1Q_0	00	01	11	10
00	X	X	X	0
01	X	X	X	0
11	X	X	X	0
10	X	X	X	0

K_1 ='0'

A B \ Q_1Q_0	00	01	11	10
00	0	0	X	X
01	1	0	X	X
11	0	0	X	X
10	0	0	X	X

$J_1 = \overline{A} \cdot B \cdot \overline{Q_0}$

AmayorB= Q_0 AigualB= $\overline{Q_1} \cdot \overline{Q_0}$ AmenorB= Q_1

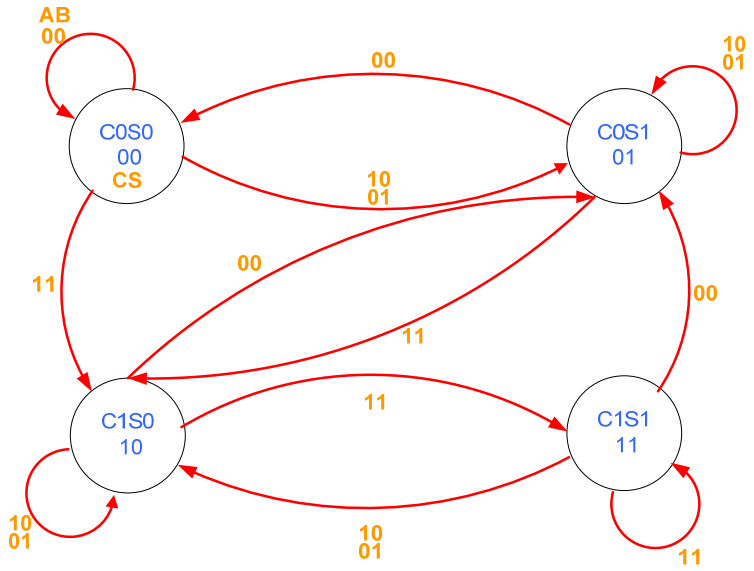


c)

```
----------------------------------------------------------------------

-- Comparador secuencial
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comparador_secuencial is
    Port (  A, B, CLK, Reset : in std_logic;
           AmayorB,AigualB,AmenorB : out std_logic);
end comparador_secuencial;

architecture Behavioral of comparador_secuencial is
      --Definimos un tipo con los estados posibles
      type estados_posibles IS (igual,Amayor,Bmayor);

      --Declaramos una señal que puede tomar cualquiera de los estados
      --posibles
      signal estado: estados_posibles;

Begin
      --Definimos un proceso sensible al reloj y al reset para modificar el
estado.
      process(CLK, Reset)
      begin
            --Reset asíncrono de nivel activo alto.
            if (Reset='1') then  estado <= igual;
            elsif (CLK 'event and CLK='1') then
                  case estado is
                        when igual =>
                              if (A='1' and B='0') then estado <=Amayor;
                              elsif (A='0' and B='1') then estado <=Bmayor;
                              end if;
                        when Amayor =>
                              estado <=Amayor;
                        when Bmayor =>
                              estado <=Bmayor;
                  end case;
            end if;
      end process;

      --Ahora en función del estado, generamos la salida
      AmayorB <= '1' when (estado = Amayor) else '0';
      AigualB <= '1' when (estado = igual) else '0';
      AmenorB <= '1' when (estado = Bmayor) else '0';

end Behavioral;
```

Ejercicio 8.

a)



b)

Entadas		Estado Actual			Salidas		Estado siguiente				
A	B	Q1	Q0	Estado	C	S	D1	D0	$Q1^+$	$Q0^+$	Estado
0	0	0	0	C0S0	0	0	0	0	0	0	C0S0
0	0	0	1	C0S1	0	1	0	0	0	0	C0S0
0	0	1	0	C1S0	1	0	0	1	0	1	C0S1
0	0	1	1	C1S1	1	1	0	1	0	1	C0S1
0	1	0	0	C0S0	0	0	0	1	0	1	C0S1
0	1	0	1	C0S1	0	1	0	1	0	1	C0S1
0	1	1	0	C1S0	1	0	1	0	1	0	C1S0
0	1	1	1	C1S1	1	1	1	0	1	0	C1S0
1	0	0	0	C0S0	0	0	0	1	0	1	C0S1
1	0	0	1	C0S1	0	1	0	1	0	1	C0S1
1	0	1	0	C1S0	1	0	1	0	1	0	C1S0
1	0	1	1	C1S1	1	1	1	0	1	0	C1S0
1	1	0	0	C0S0	0	0	1	0	1	0	C1S0
1	1	0	1	C0S1	0	1	1	0	1	0	C1S0
1	1	1	0	C1S0	1	0	1	1	1	1	C1S1
1	1	1	1	C1S1	1	1	1	1	1	1	C1S1

A B \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	1	1	1
10	0	0	1	1

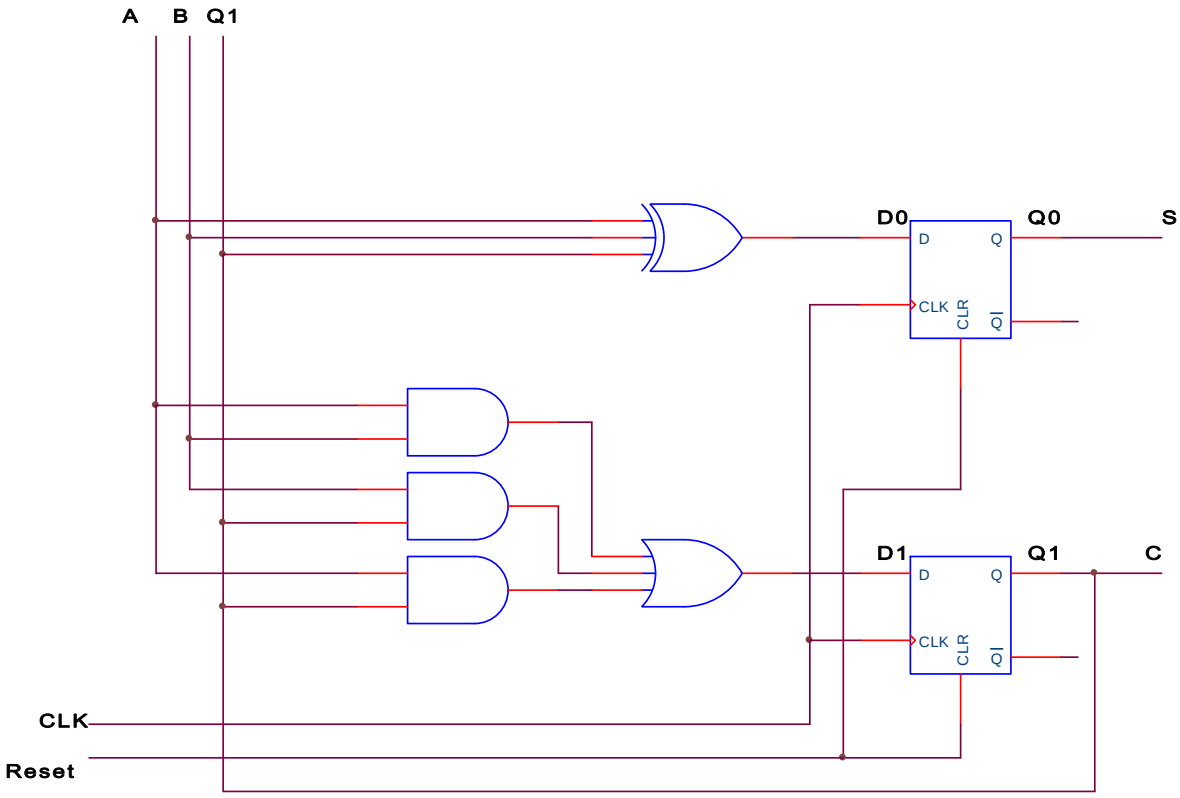
$$D_1 = A{\cdot}B + B{\cdot}Q_1 + A{\cdot}Q_1$$

A B \ Q_1Q_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

$$D0 = \overline{A}{\cdot}B{\cdot}\overline{Q_1} + \overline{A}{\cdot}\overline{B}{\cdot}Q_1 + A{\cdot}B{\cdot}Q_1 + A{\cdot}\overline{B}{\cdot}\overline{Q_1} = (A \oplus B)\overline{Q_1} + \overline{(A \oplus B)}Q_1 = A \oplus B \oplus Q_1$$

$C = Q_1$

$S = Q_0$

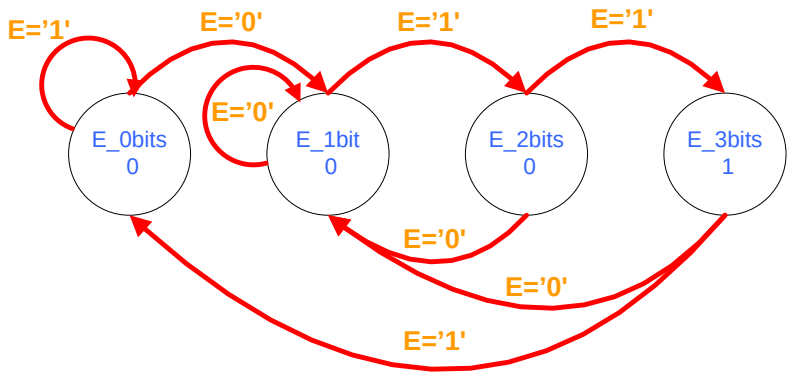


c)

```
----------------------------------------------------------------------
-- Sumador secuencial
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sumador_secuencial is
    Port (  A, B, CLK, Reset : in std_logic;
                                        S, C : out std_logic);
end sumador_secuencial;

architecture Behavioral of sumador_secuencial is
      --Definimos un tipo con los estados posibles
      type estados_posibles IS (C0S0, C0S1, C1S0, C1S1);

      --Declaramos una señal que puede tomar cualquiera de los estados
      --posibles
      signal estado: estados_posibles;
Begin
      --Definimos un proceso sensible al reloj y al reset para modificar el
estado.
      process(CLK, Reset)
      begin
            --Reset asíncrono de nivel activo alto.
            if (Reset='1') then  estado <= C0S0;
            elsif (CLK 'event and CLK='1') then
                  case estado is
                        when C0S0 =>
                              if ((A xor B)='1') then estado <=C0S1;
                              elsif ((A and B)='1') then estado <=C1S0;
                              end if;
                        when C0S1 =>
                              if ((A nor B)='1') then estado <=C0S0;
                              elsif ((A and B)='1') then estado <=C1S0;
                              end if;
                        when C1S0 =>
                              if ((A nor B)='1') then estado <=C0S1;
                              elsif ((A and B)='1') then estado <=C1S1;
                              end if;
                        when C1S1 =>
                              if ((A xor B)='1') then estado <=C1S0;
                              elsif ((A nor B)='1') then estado <=C0S1;
                              end if;
                  end case;
            end if;
      end process;

      --Ahora en función del estado, generamos la salida
      S<= '1' when (estado = C0S1 or estado = C1S1) else '0';
      C<='1' when (estado =  C1S1 or estado = C1S0) else '0';
end Behavioral;
```

Ejercicio 9.

a)



b)

Entadas	Estado Actual			Salidas	Estado siguiente						
E	Q1	Q0	Estado	O	J1	K1	J0	K0	$Q1^+$	$Q0^+$	Estado
0	0	0	0_bits	0	0	X	1	X	0	1	1_bit
0	0	1	1_bit	0	0	X	X	0	0	1	1_bit
0	1	0	2_bits	0	X	1	1	X	0	1	1_bit
0	1	1	3_bits	1	X	1	X	0	0	1	1_bit
1	0	0	0_bits	0	0	X	0	X	0	0	0_bits
1	0	1	1_bit	0	1	X	X	1	1	0	2_bits
1	1	0	2_bits	0	X	0	1	X	1	1	3_bits
1	1	1	3_bits	1	X	1	X	1	0	0	0_bits

Q_1Q_0 / E	00	01	11	10
0	X	0	0	X
1	X	1	1	X

$$K_0 = E$$

Q_1Q_0 / E	00	01	11	10
0	1	X	X	1
1	0	X	X	1

$$J_0 = \overline{E} + Q_1$$

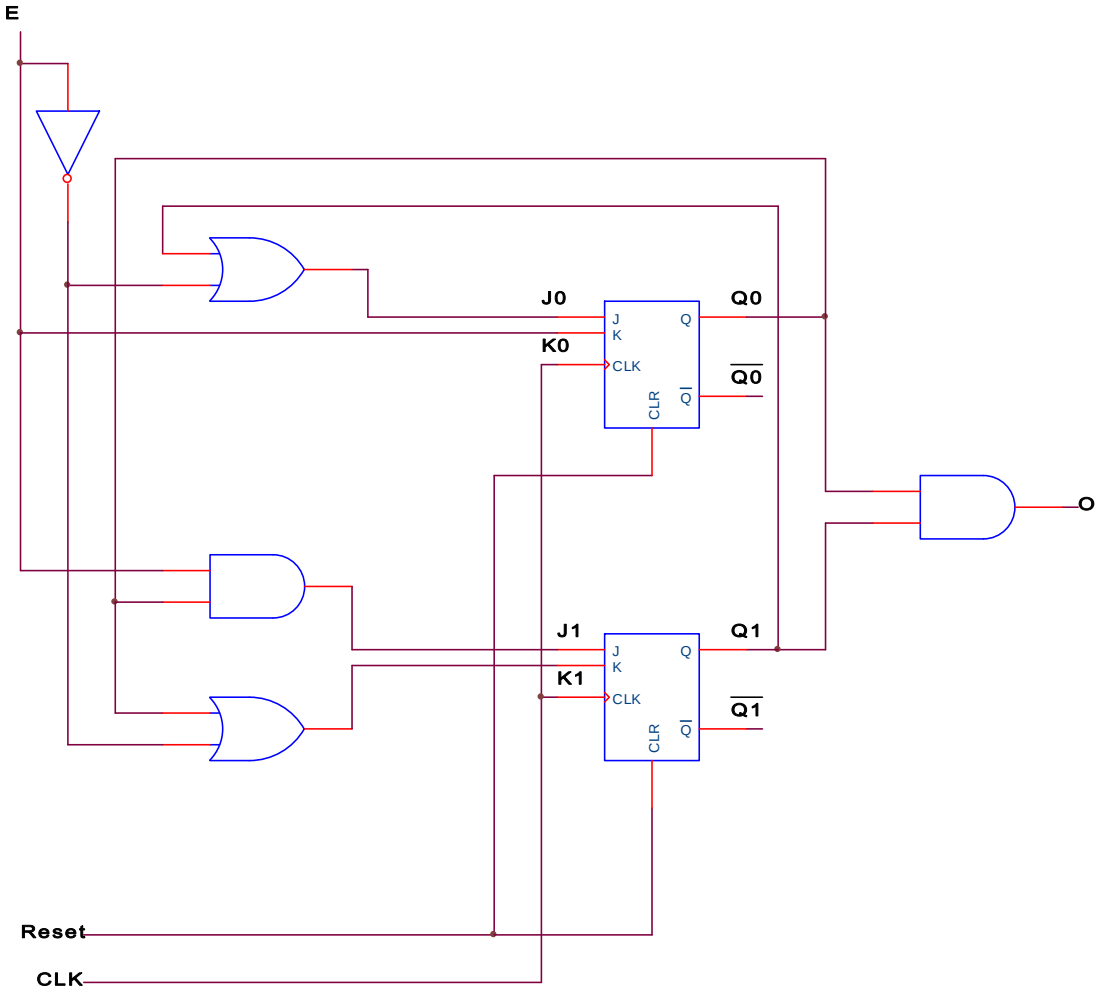
E \ Q_1Q_0	00	01	11	10
0	X	X	1	1
1	X	X	1	0

$$K_1 = \overline{E} + Q_0$$

E \ Q_1Q_0	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$$J_1 = E \cdot Q_0$$

$$S = Q_1 \cdot Q_0$$



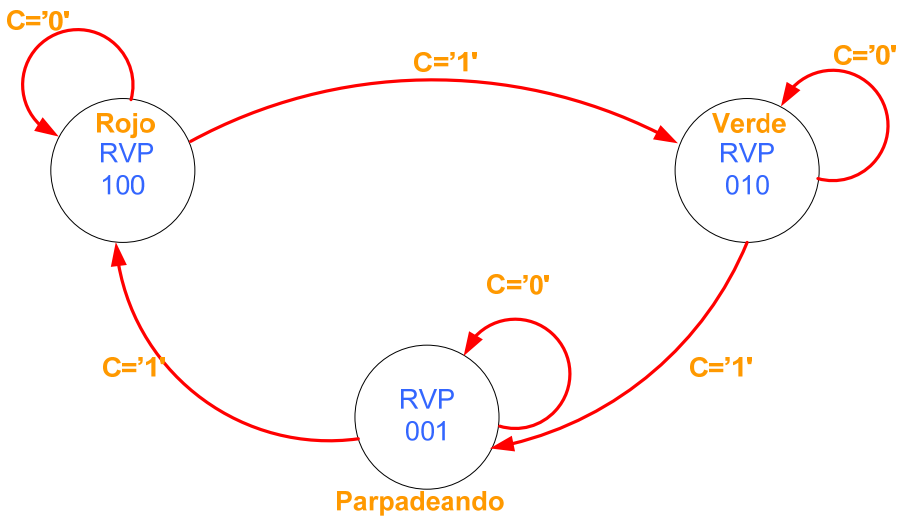
c)

```
----------------------------------------------------------------------
-- Buscador de secuencia
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity buscador_secuencia is
    Port (  E,CLK, Reset : in std_logic;
                      O : out std_logic);
end buscador_secuencia;

architecture Behavioral of buscador_secuencia is
      --Definimos un tipo con los estados posibles
      type estados_posibles IS (0_bits,1_bit, 2_bits, 3_bits);

      --Declaramos una señal que puede tomar cualquiera de los estados
      --posibles
      signal estado: estados_posibles;
Begin
      --Definimos un proceso sensible al reloj y al reset para modificar el
estado.
      process(CLK, Reset)
      begin
            --Reset asíncrono de nivel activo alto.
            if (Reset='1') then  estado <= 0_bits;
            elsif (CLK 'event and CLK='1') then
                  case estado is
                        when 0_bits =>
                              if (E='0') then estado <=1_bit;
                              end if;
                        when 1_bit =>
                              if (E='1') then estado <=2_bits;
                              end if;
                        when 2_bits =>
                              if (E='1') then estado <=3_bits;
                              end if;;
                        when 3_bits =>
                              if (E='1') then estado <=0_bits;
                              else estado <=1_bit;
                              end if;
                  end case;
            end if;
      end process;

      --Ahora en función del estado, generamos la salida
      O<= '1' when (estado = 3_bits) else '0';
end Behavioral;
```

Ejercicio 10.

a)



b)

Entadas	Estado Actual			Salidas			Estado siguiente						
C	Q1	Q0	Estado	R	V	P	J1	K1	J0	k0	$Q1^+$	$Q0^+$	Estado
0	0	0	Rojo	1	0	0	0	X	0	X	0	0	Rojo
0	0	1	Verde	0	1	0	0	X	X	0	0	1	Verde
0	1	0	Parpadeando	0	0	1	X	0	0	X	1	0	Parpadeando
1	0	0	Rojo	1	0	0	0	X	1	X	0	1	Verde
1	0	1	Verde	0	1	0	1	X	X	1	1	0	Parpadeando
1	1	0	Parpadeando	0	0	1	X	1	0	X	0	0	Rojo

Q_1Q_0 / C	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$$J_1 = C \cdot Q_0$$

Q_1Q_0 / C	00	01	11	10
0	X	X	X	0
1	X	X	X	1

$$K_1 = C$$

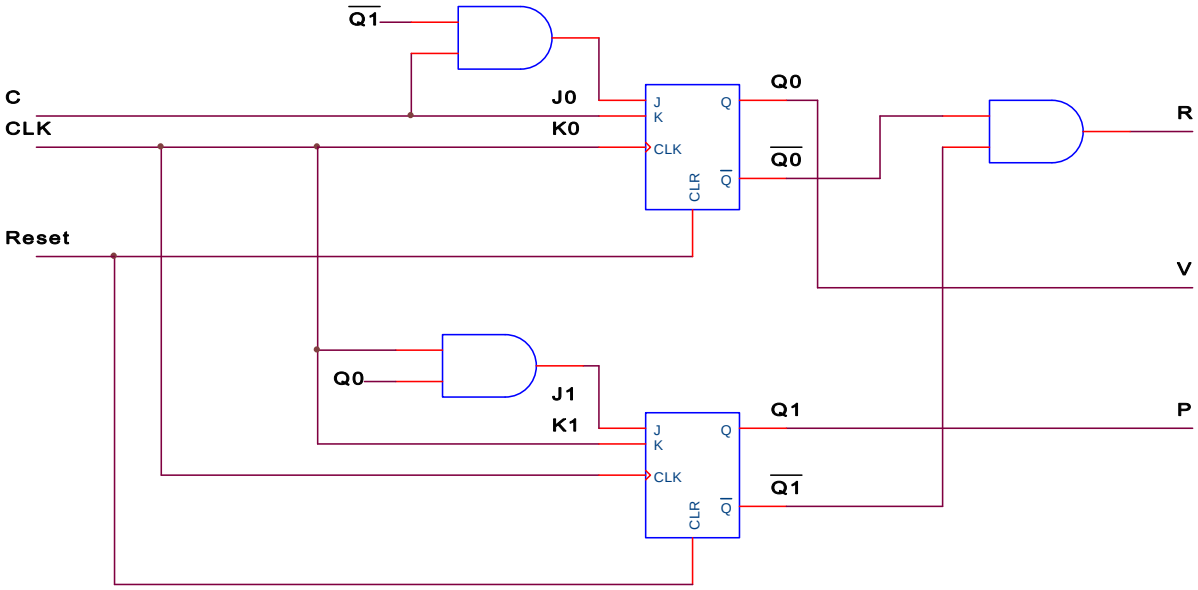
Q_1Q_0 / C	00	01	11	10
0	0	X	X	0
1	1	X	X	0

$$J_0 = C \cdot \overline{Q_1}$$

Q_1Q_0 / C	00	01	11	10
0	X	0	X	X
1	X	1	X	X

$$K_0 = C$$

$$R = \overline{Q_1} \cdot \overline{Q_0} \qquad V = Q_0 \qquad P = Q_1$$



c)

```
--------------------------------------------------------------------------
-- Semáforo Peatonal
--------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity semaforo is
    Port (  C,CLK, Reset : in std_logic;
                   R,V,P : out std_logic);
end semaforo;

architecture Behavioral of semaforo  is
      --Definimos un tipo con los estados posibles
      type estados_posibles IS (Rojo, Verde, Parpadeando);

      --Declaramos una señal que puede tomar cualquiera de los estados
      --posibles
      signal estado: estados_posibles;
Begin
      --Definimos un proceso sensible al reloj y al reset para modificar el
estado.
      process(CLK, Reset)
      begin
            --Reset asíncrono de nivel activo alto.
            if (Reset='1') then  estado <= Rojo;
            elsif (CLK 'event and CLK='1') then
                  case estado is
                        when Rojo =>
                              if (C='1') then estado <=Verde;
                              end if;
                        when Verde =>
                              if (C='1') then estado <=Parpadeando;
                              end if;
                        when Parpadeando =>
                              if (C='1') then estado <=Rojo;
                              end if;
                  end case;
```

```
            end if;
      end process;

      --Ahora en función del estado, generamos la salida
      R<= '1' when (estado = Rojo) else '0';
      V<='1' when (estado = Verde) else '0';
      P<='1' when (estado = Parpadeando) else '0';
end Behavioral;
```