

Final 23-12-20

Monday, July 19, 2021 3:35 PM

Problema 1 (4 puntos).

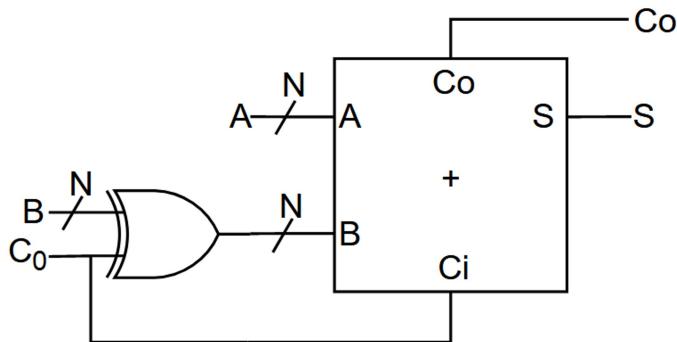
Diseñe un circuito a nivel RTL que tenga dos entradas (**A** y **B**) de N bits una entrada **C** (C_1, C_0) de 2 bits que se la utilizará para controlar el comportamiento del circuito. El circuito tendrá dos salidas, **S** de N bits y **SVAL** de un bit. El funcionamiento del circuito es el siguiente:

- Si, $C = 00$ se genera $S = A+B$ considerando A y B magnitudes.
- Si, $C = 01$ se genera $S = A-B$ considerando A y B magnitudes.
- Si, $C = 10$ se genera $S = A+B$ considerando A y B enteros en complemento a 2
- Si, $C = 11$ se genera $S = A-B$ considerando A y B enteros en complemento a 2.

En todos los casos **SVAL** debe estar en uno cuando el resultado de la operación es correcto, en caso contrario en cero.

a) Diseñe a nivel RTL para poder implementar la funcionalidad descrita, utilizando **un único sumador**, multiplexores y compuertas (cantidad necesaria).

b) Determine la salida **S** y **SVAL** para todos los casos de **C** con **A** = "1101" y **B** = "0110" (N = 4 bits)



MAGNITUDES - $C_1 = 0$

OP	C_0	A	B	C_{out}	OV
$A + B$	0	-	-	1	1
$A - B$	1	-	-	0	1
OTRAS				0	

$$OV_{A2} = C_{out\ n-1} \oplus C_{out\ n}$$

$$OV_{A2} = C_{out} \oplus (S_n \cdot \bar{B}_n \cdot \bar{A}_n + \bar{S}_n \cdot A_n \cdot B_n)$$

$$OV_{MAG} = C_{out} \oplus C_0$$

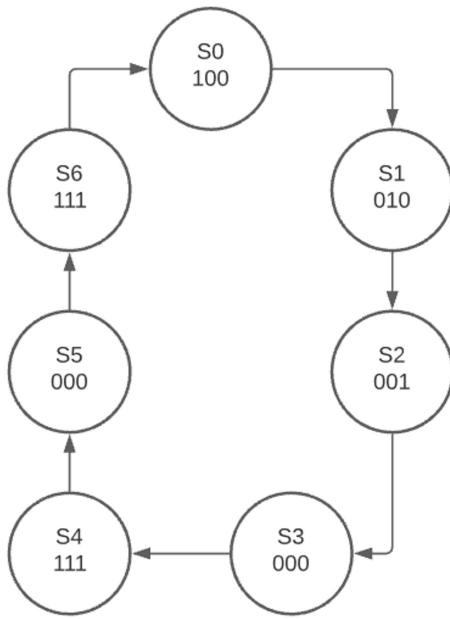
$$OV = OV_{MAG} \cdot \bar{C}_1 + OV_{A2} \cdot C_1$$

La expresión de OV es un multiplexor manejado por C_1

Problema 3 (4 puntos).

Dada la secuencia 100-010-001-000-111-000-111

- Determine el diagrama de estados, las tablas de estado futuro y de salidas.
- Implemente con flip flops D codificando los estados en one-hot¹. De las ecuaciones de excitación y las ecuaciones de las salidas.
- ¿Cuántos estados no utilizados tiene esta implementación?
- Implemente en VHDL con reset asincrónico. Para este punto puede utilizar la codificación de estados que desee.



q	q*	o		
S0	S1	1	0	0
S1	S2	0	1	0
S2	S3	0	0	1
S3	S4	0	0	0
S4	S5	1	1	1
S5	S6	0	0	0
S6	S0	1	1	1

q
0000001
0000010
0000100
0001000
0010000
0100000
1000000

$$o(2) = S0 + S4 + S6 = q(0) + q(4) + q(6)$$

$$o(1) = S1 + S4 + S6 = q(1) + q(4) + q(6)$$

$$o(0) = S2 + S4 + S6 = q(2) + q(4) + q(6)$$

B) Esta implementación se realiza mediante un contador anillo

C) En la implementación one hot se simplifican las expresiones de salida a costa de emplear una mayor cantidad de FF, con otra implementación podríamos haber resuelto el problema con solo 3 FF. En este caso en particular usamos 7 FF, lo cual nos da una cantidad de $2^7 = 128$ combinaciones posibles, de las cuales solo 7 están en uso, por lo que tenemos 121 estados no utilizados.

```

--You can specify the type of encoding, some options are:
--BINAY, GRAY, ONE_HOT, ONE_COLD or AUTO
type state is (s0, s1, s2, s3, s4, s5, s6);
attribute enum_encoding : string;
attribute enum_encoding of state : type is "ONE_HOT"; -- encoding style of the enumerated type

signal s_reg, s_next : state;

state_register: process(clk, rst)
begin
    if rst = '1' then --EN EL CASO DE QUE SE TUVIESE UN RESET ASINCRONICO
        s_reg <= s0;

    elsif rising_edge(clk) then
        s_reg <= s_next;
    end if;

end process state_register;

--NEXT STATE LOGIC
with s_reg select
    s_next <= s1 when s0,
    s2 when s1,
    s3 when s2,
    s4 when s3,
    s5 when s4,
    s6 when s5,
    s0 when s6;

--OUTPUT LOGIC
with s_reg select
    o(2) <= '1' when s0 | s4 | s6,
    '0' when others;

with s_reg select
    o(1) <= '1' when s1 | s4 | s6,
    '0' when others;

with s_reg select
    o(0) <= '1' when s2 | s4 | s6,
    '0' when others;

```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity p4 is
  Port ( clk : in STD_LOGIC;
         rst : in STD_LOGIC;
         a : in STD_LOGIC;
         b : in STD_LOGIC;
         s : out STD_LOGIC);
end p4;
architecture Behavioral of p4 is
  signal ci,co : std_logic;
begin
  process(clk)
  begin
    if(rising_edge(clk)) then
      if(rst = '1') then
        ci <= '0';
      else
        ci <= co;
      end if;
    end if;
  end process;
  s <= a xor b xor ci;
  co <= (a and b) or (a and ci) or (b and ci);
end Behavioral;

```

Dado el VHDL de la figura determine la salida (s) para todos los ciclos de reloj de la tabla, empezando por t_0 y considerando que previamente a t_0 el sistema fue reseteado.

#	a	b	C_i^*	C_i	C_o	S
t_0	0	1	0	0	0	1
t_1	1	1	1	0	1	0
t_2	0	1	1	1	1	0
t_3	1	0	1	1	1	0
t_4	0	0	0	1	0	1
t_5	0	1	0	0	0	1

Final 02-26-20

Sunday, July 25, 2021 5:06 PM

1. Implemente un conversor gray a binario utilizando xor y for generate

```
entity grayBinario is
    generic (N: integer := 4);
    port ( gray : in std_logic_vector (N-1 downto 0);
           binario : out std_logic_vector (N-1 downto 0));
end grayBinario;
```

Utilice las siguientes expresiones para realizar la conversión:

- $binario(i) = gray(i)$; para $n = N - 1$
- $binario(i) = binario(i + 1) \text{ xor } gray(i)$; para $(N - 2) \leq i \leq 0$

```
--Test vectors
type input_array is array (0 to 7) of std_logic_vector(2 downto 0);
constant input_vectors: input_array := ("000", "001", "011", "010", "110", "111", "101", "100");

begin
    uut:
        entity work.grayBinary
            --SI ES QUE SE TIENE UNA ENTIDAD GENERICA
            generic map(N => N)
            --Relacionar las signals del testbench con las de la entidad ... señal => tb_señal
            port map(gray => t_gray, binario => t_binario);

    -----
    -- ESTIMULOS --
    -----
proceso: process
begin
    for i in input_vectors'range loop
        t_gray <= input_vectors(i);
        wait for INTERVALO;
    end loop;
end process proceso;
```

2. Implemente en VHDL un circuito con la siguiente entidad que realice operaciones aritméticas y lógicas

```
entity myALU is
    generic (N : integer := 8);
    port (a: in std_logic_vector (N-1 downto 0);
          b: in std_logic_vector (N-1 downto 0);
          r : out std_logic_vector (N-1 downto 0);
          ov: out std_logic;
          op: in std_logic_vector (1 downto 0));
end myALU;
```

Donde:

- a y b: son los operandos
- r: es el resultado de las operaciones.
- ov: salida que indica si hubo overflow en la operación.
- op: es la operación a realizar
 - 00: a + b signado (setea la señal de overflow)
 - 01: a - b signado (setea la señal de overflow)
 - 10: a and b bit a bit (setea overflow en cero)
 - 11: a or b bit a bit (setea overflow en cero)

No está permitido utilizar process en este ejercicio.

```
signal aux_a,aux_b: signed(N-1 downto 0);
signal aux_r: std_logic_vector (N-1 downto 0);
signal aux_ov: std_logic_vector (2 downto 0);
signal ov_s, ov_r: std_logic;

begin
    aux_a <= signed(a);
    aux_b <= signed(b);

    --r logic
    r    <= std_logic_vector(aux_a + aux_b) when op = "00" else
        std_logic_vector(aux_a - aux_b) when op = "01" else
        ((a) and (b))                      when op = "10" else
        ((a) or (b));

    --ov logic
    aux_ov <= aux_r(N-1) & a(N-1) & b(N-1);

    with aux_ov select
        ov_s <= '1' when "100",
        '1' when "011",
        '0' when others;

    with aux_ov select
        ov_r <= '1' when "101",
        '1' when "010",
        '0' when others;

    ov <= ov_s when op = "00" else
        ov_r when op = "01" else
        '0';

end architecture data_flow;
```

4. Realice una descripción en VHDL de un registro de desplazamiento con entrada serie, salida paralelo y reset sincrónico. La entidad es la siguiente.

```
entity registro is
  generic (N : integer := 8);
  port ( clk :          in std_logic;
         rst :          in std_logic;
         entradaSerie : in std_logic;
         salidaParalelo : out std_logic_vector (N-1 downto 0));
end registro;
```

Final 14-10-20

Tuesday, July 20, 2021 4:34 PM

Problema 1

Implemente un circuito que genere la siguiente secuencia 000 - 100 - 010 - 111 cada valor de la secuencia debe permanecer por 100ns. Suponga que el clock del sistema tiene un periodo de 10ns e implemente con rst sincrónico.

```
entity myGenCnt is
    port ( clk: in std_logic;
           rst: in std_logic;
           q : out std_logic_vector (2 downto 0));
end myGenCnt;
```

```
You, 3 hours ago • first commit
attribute enum_encoding : string;

type s_state is (s0, s1, s2, s3);
attribute enum_encoding of s_state : type is "AUTO"; -- encoding style of the enumerated type
signal s_reg, s_next : s_state;

type c_state is (c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10);
attribute enum_encoding of c_state : type is "AUTO"; -- encoding style of the enumerated type
signal c_reg, c_next: c_state;

begin

state_register: process(clk, rst)
begin
    if rising_edge(clk) then
        if rst = '1' then
            s_reg <= s0;
            c_reg <= c0;
        else
            s_reg <= s_next;
            c_reg <= c_next;
        end if;
    end if;
end process state_register;

--NEXT STATE LOGIC
next_state_logic: process(s_reg, c_reg)
begin
    if c_reg = c10 then
        case s_reg is
            when s0 => s_next <= s1;
            when s1 => s_next <= s2;
            when s2 => s_next <= s3;
            when s3 => s_next <= s0;
        end case;
    end if;

    case c_reg is
        when c0 => c_next <= c1;
        when c1 => c_next <= c2;
        when c2 => c_next <= c3;
        when c3 => c_next <= c4;
        when c4 => c_next <= c5;
        when c5 => c_next <= c6;
        when c6 => c_next <= c7;
        when c7 => c_next <= c8;
        when c8 => c_next <= c9;
        when c9 => c_next <= c10;
        when c10 => c_next <= c0;
    end case;
end process next_state_logic;

--OUTPUT LOGIC
with s_reg select
    q <= "000" when s0,
               "100" when s1,
               "010" when s2,
               "111" when s3,
               "000" when others;

end architecture data_flow;
```

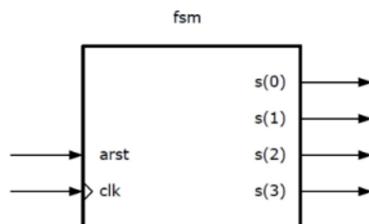
```

    "111" when s3,
    "000" when others;
end architecture data_flow;

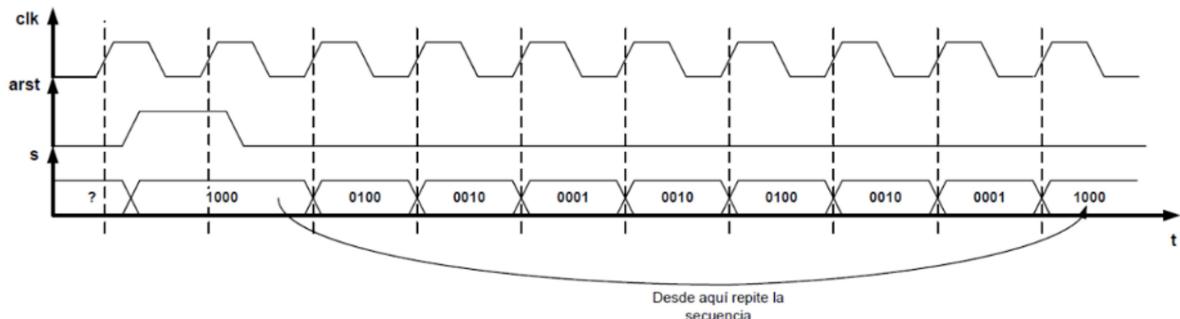
```

Problema 2

Diseñe un circuito secuencial cuyas entradas y salidas vienen dadas por la figura



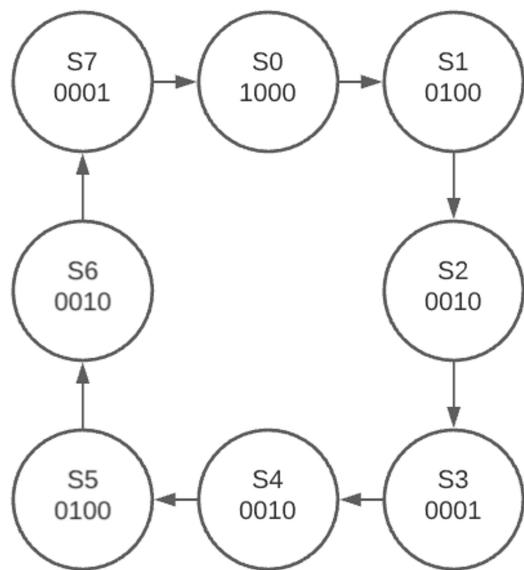
Construya el circuito usando flip-flops disparados por flanco ascendente y con reset asincrónico. El circuito evoluciona siguiendo la secuencia dada por el diagrama temporal



Muy importante: La secuencia se repite cada 8 ciclos de reloj.

- Confeccione un diagrama de estados que satisfaga el comportamiento buscado.
- Construya las tablas de transiciones y salidas.
- Obtenga las ecuaciones de transiciones y salidas.

No es necesario que dibuje el circuito resultante



q	q*	S
s0	s1	1000
s1	s2	0100
s2	s3	0010
s3	s4	0001
s4	s5	0010
s5	s6	0100
s6	s7	0010
s7	s0	0001

$$\begin{aligned}
 S(0) &= q(3) + q(7) \\
 S(1) &= q(2) + q(4) + q(6) \\
 S(2) &= q(1) + q(5) \\
 S(3) &= q(0)
 \end{aligned}$$

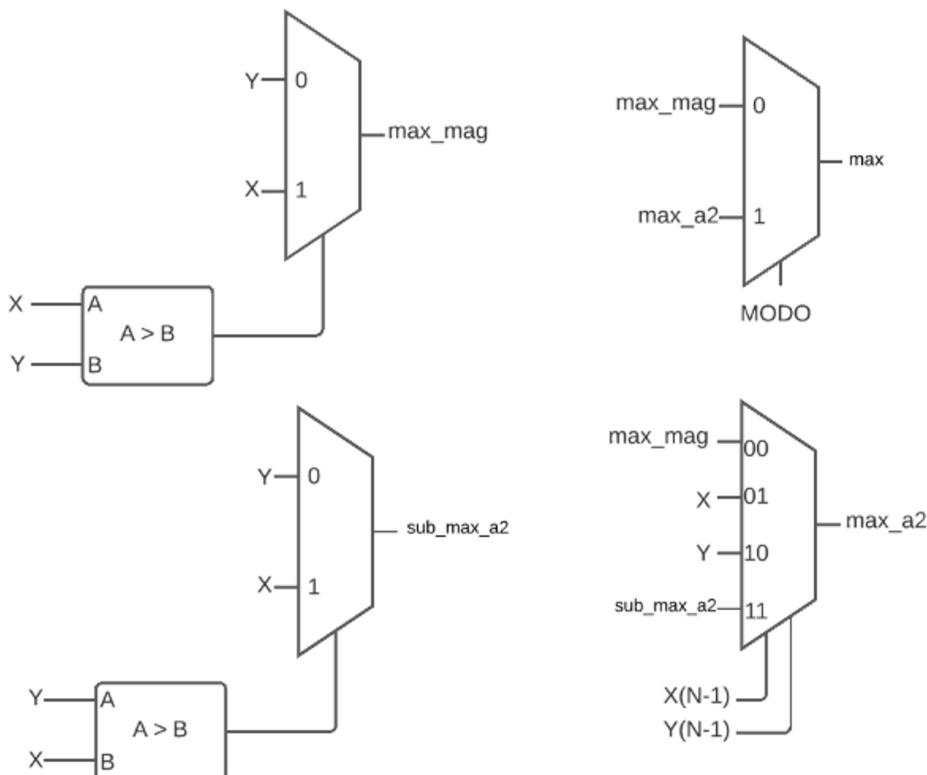
Empleando codificación one-hot

Problema 3

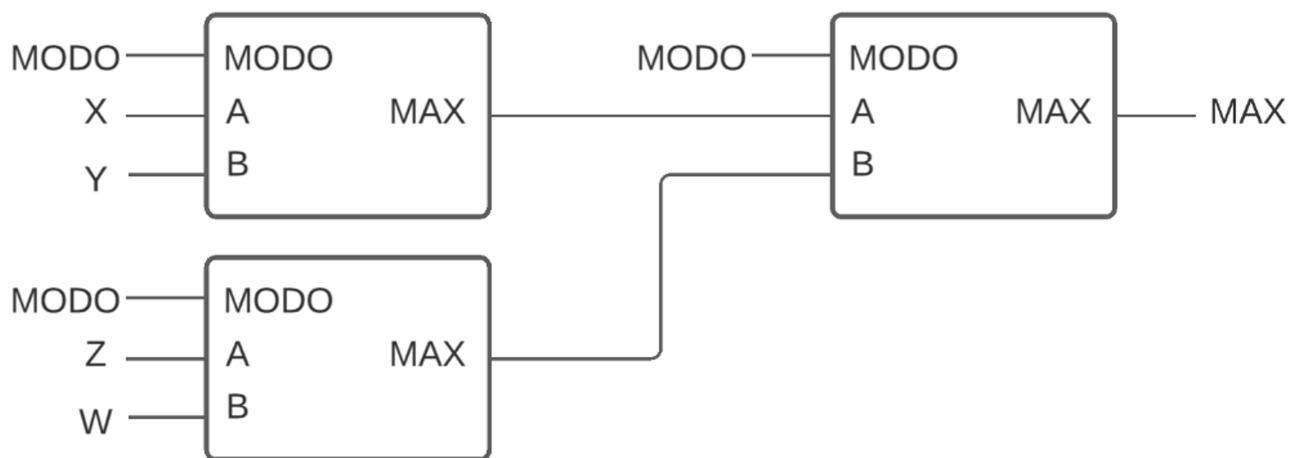
Un circuito combinacional tiene dos entradas (X e Y) de N bits, una entrada MODO de 1 bit y una salida Z de N bits. Se comporta de la siguiente manera. Si MODO es 1 considera a las entradas como números en complemento a 2 y entrega a su salida la mayor de las dos entradas. Si MODO es 0 considera las entradas como magnitudes binarias de N bits y entrega a su salida la mayor de las dos entradas.

- Diseñe a nivel RTL un circuito que implemente el comportamiento predicho. Para implementarlo posee comparadores de magnitudes (solamente), multiplexores, inversores y algunas compuertas de su preferencia
- A partir de la celda diseñada en el punto anterior genere un circuito que dada cuatro magnitudes o enteros codificados en complemento a 2 ponga en su salida al mayor de ellos.
- Usando el circuito del punto b obtenga un circuito que se comporte de la misma manera pero que en su salida el resultado siempre esté codificado en complemento a 2

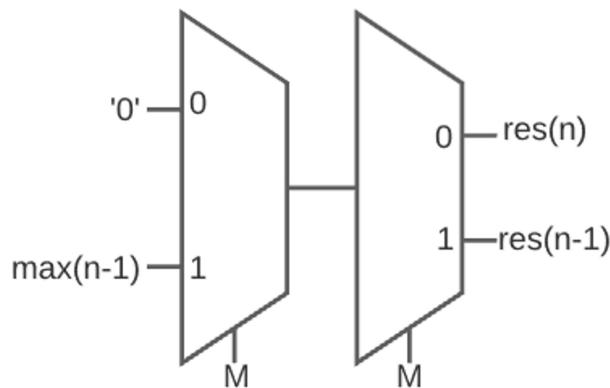
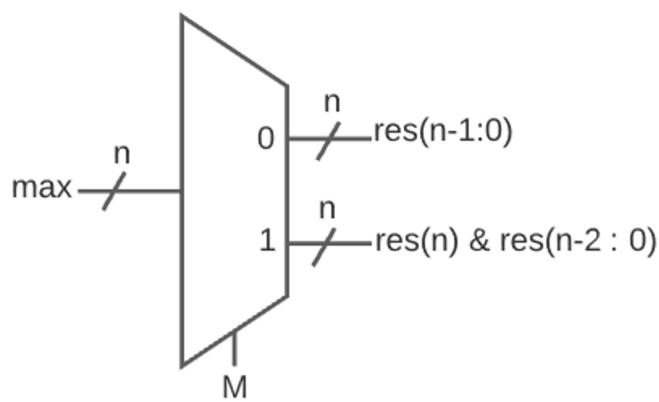
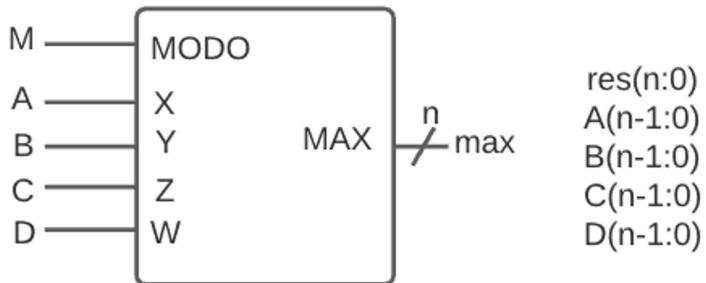
A)



B)

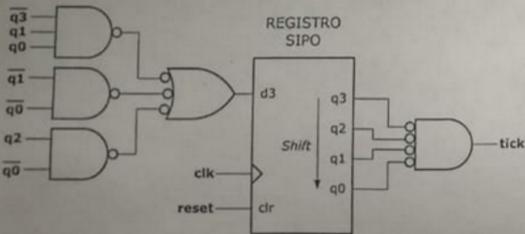


C)



Problema 3

Para el circuito secuencial de la figura P3, realizado en base a un registro SIPO con entrada clear asincrónica (clr):



- Obtenga la tabla de transición/salida en el formato adecuado al tipo de máquina.
- Dibuje el *diagrama de estados completo* con el formato adecuado, identificando a los estados con los valores de las señales de estado $q_3 q_2 q_1 q_0$.
- En base al análisis del diagrama de estados, determine que función realiza el circuito.

Fig.P3

	d3 = 0				d3 = 1							
	q3	q2	q1	q0	q3*	q2*	q1*	q0*	q3*	q2*	q1*	q0*
s0	0	0	0	0	0	0	0	0	1	0	0	0
s1	0	0	0	1	0	0	0	0	1	0	0	0
s2	0	0	1	0	0	0	0	1	1	0	0	1
s3	0	0	1	1	0	0	0	1	1	0	0	1
s4	0	1	0	0	0	0	1	0	1	0	1	0
s5	0	1	0	1	0	0	1	0	1	0	1	0
s6	0	1	1	0	0	0	1	1	1	0	1	1
s7	0	1	1	1	0	0	1	1	1	0	1	1
s8	1	0	0	0	0	1	0	0	1	1	0	0
s9	1	0	0	1	0	1	0	0	1	1	0	0
s10	1	0	1	0	0	1	0	1	1	1	0	1
s11	1	0	1	1	0	1	0	1	1	1	0	1
s12	1	1	0	0	0	1	1	0	1	1	1	0
s13	1	1	0	1	0	1	1	0	1	1	1	0
s14	1	1	1	0	0	1	1	1	1	1	1	1
s15	1	1	1	1	0	1	1	1	1	1	1	1

s	s*	d	tick
s0	s8	1	1
s1	s0	0	0
s2	s1	0	0
s3	s9	1	0
s4	s10	1	0
s5	s2	0	0
s6	s11	1	0
s7	s11	1	0
s8	s12	1	0
s9	s8	0	0
s10	s5	0	0
s11	s5	0	0
s12	s14	1	0
s13	s14	0	0
s14	s15	1	0
s15	s7	0	0

Pareciera un contador de ciclos de clock. Una vez presionado el reset la salida tick toma el valor '1' y después de 10 ciclos (si no se presiona el reset durante el conteo), se vuelve a poner la misma en '1', durante todo el conteo la salida tick vale '0'.

Problema 4

Formato del Diagrama

Si (Estado)
salida \leq valor
 $Exp[d]$

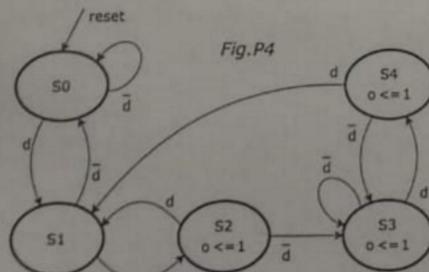


Fig.P4

La figura P4 describe la operación de una máquina de estados sincrónico con una entrada *reset asincrónica*, una entrada *d* y una salida *o* *síncronas*.

- En base al diagrama de estados, realice un cronograma con secuencias de entrada/salida sincronizadas con el flanco ascendente de *clk*, que le permita expresar en palabras que función realiza esta máquina de estados.
- Realice la descripción en código VHDL de la entidad *fsm*, usando *procesos separados* para la parte sensible a *clk* y para las partes combinacionales.

	d=0	d=1	
s	s*	s*	o
s0	s0	s1	0
s1	s0	s2	0
s2	s3	s1	1
s3	s3	s4	1
s4	s3	s1	1

El estado de la salida cambia cuando si la entrada se mantuvo en '1' durante los últimos dos ciclos de clock

Problema 2

Se requiere diseñar el *circuito combinacional genérico* mostrado en la figura P2, donde N es la cantidad de bits en la entrada de datos. Los datos de entrada **data_in** y salida **data_out** son *números enteros representados en el código de complemento a dos*; la salida **data_out** debe ser el producto aritmético entre la entrada **data_in** y los factores **+1, +2, +4 y +8** cuando la señal de control **ctrl(1:0)** sea "00", "01", "10" y "11" respectivamente.

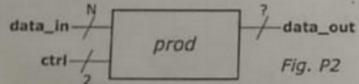


Fig. P2

- Dibuje un esquema conceptual usando bloques RTL para la solución propuesta.
- Realice la descripción VHDL *genérica* de la entidad **prod** en base al diagrama RTL del punto a).

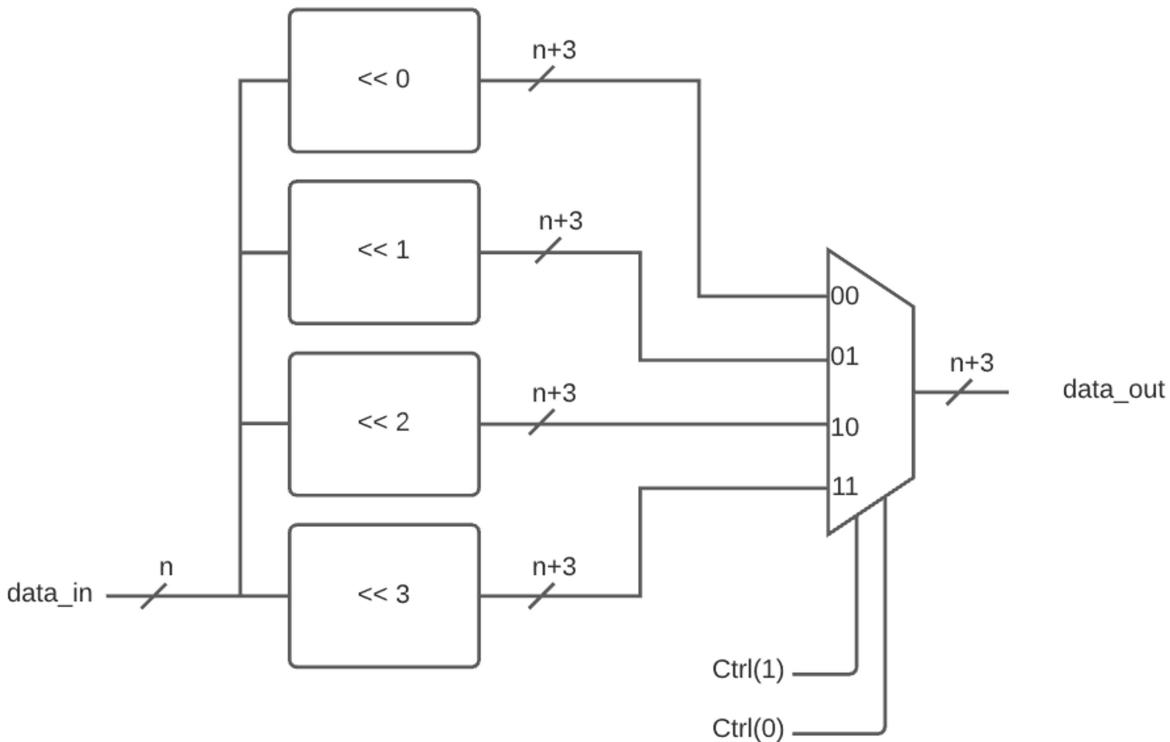
Tengo la siguiente operatoria

$$\text{Data_out} = \text{data_in} * \text{ctrl}$$

Que puede ser traducido de la siguiente manera:

$$\text{Data_out} = \text{data_in} \ll \text{ctrl}$$

Para poder representar a la salida cualquier combinación entre **data_in** * **ctrl**, tomamos el caso límite en el cual **ctrl = "11"**, lo cual significa un desplazamiento de 3 bits. Por lo que **data_out** tendrá que tener un tamaño de $n + 3$ bits.



```

shifter: process (ctrl, data_in)
begin
    case( ctrl ) is
        when "00" =>
            data_out <= data_in(N-1) & data_in(N-1) & data_in(N-1) & data_in(N-1) & data_in(N-2 downto 0);

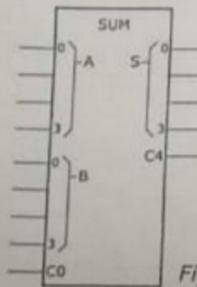
        when "01" =>
            data_out <= data_in(N-1) & data_in(N-1) & data_in(N-1) & data_in(N-2 downto 0) & "0";

        when "10" =>
            data_out <= data_in(N-1) & data_in(N-1) & data_in(N-2 downto 0) & "00";

        when others =>
            data_out <= data_in(N-1) & data_in(N-2 downto 0) & "000";
    end case ;
end process shifter;

```

Problema 1



Se busca hacer un comparador de magnitudes binarias de cuatro bits $p(3:0)$ y $q(3:0)$ usando como bloque constructivo el sumador binario de cuatro bits mostrado en la figura P1, con el agregado de mínima lógica adicional. El comparador debe proveer las salidas p_{ma_q} , p_{ig_q} y p_{me_q} .

- Indique y realice todos los pasos de diseño que permitan obtener las expresiones de las tres salidas del comparador.
- Dibuje el circuito del comparador de magnitudes de cuatro bits *indicando los nombres de todos los puertos de entrada/salida y sus conexiones*.
- Indique justificando como debe modificarse el circuito obtenido en b), para realizar un comparador de números enteros de cuatro bits representados en código de complemento a dos sin que se vea afectado el tiempo de propagación original del circuito.

Puedo modificar una de las entradas para convertir al sumador en un restador ya que los resultados van a dar mayor información respecto al valor de los datos de entrada.

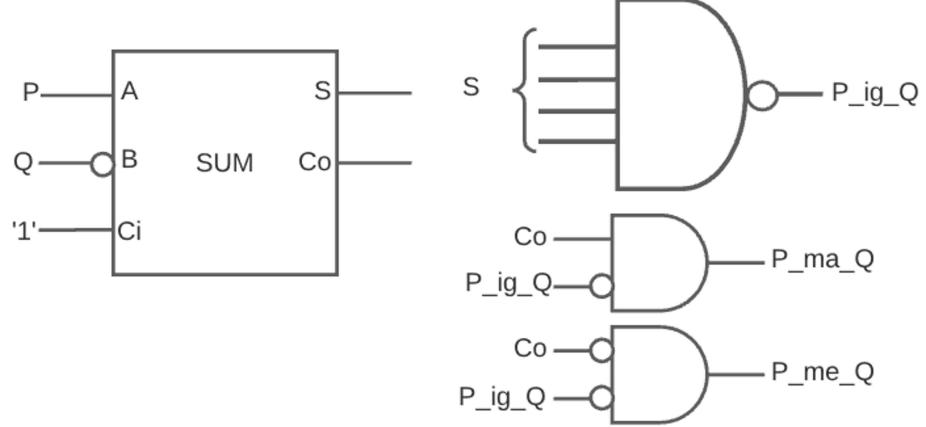
Para la resta solo necesitamos aplicar el complemento A2 a una de nuestras entradas, por ejemplo Q, para lo cual invertimos todos sus bits sumamos 1 ($Cin = 1$).

Para determinar la salida usaremos la siguiente metodología

```

if S = 0
    Q = P
else if Co = 1
    Q < P
else if Co = 0
    Q > P

```



Final 20-02-19

Saturday, July 24, 2021 5:44 PM

4. Realice una descripción en VHDL que acumule números de 8 bits, las cuales son recibidos cuando la entrada enable se encuentra en uno y hay un flanco ascendente del clock.

```
entity acumulador is
  Port ( clk : in std_logic;
         rst : in std_logic;      -- Sincronico, activo bajo
         enable: in std_logic;   -- Activo alto
         entrada :in std_logic_vector (7 down to 0);
         salida: out std_logic_vector (15 down to 0))
end acumulador;
```

- a. Implemente la arquitectura que corresponda con el diagrama de bloques anterior.

```
architecture data_flow of accumulator is

  signal accum : unsigned (15 downto 0);

begin

  accumulator: process(clk, rst, enable)
  begin
    if rst = '1' then
      accum <= (others => '0');
      salida <= (others => '0');
    elsif rising_edge(clk) then
      if enable = '1' then
        accum <= accum + unsigned(entrada);
        salida <= std_logic_vector(accum);
      else
        accum <= accum;
      end if;
    end if;
  end process accumulator;

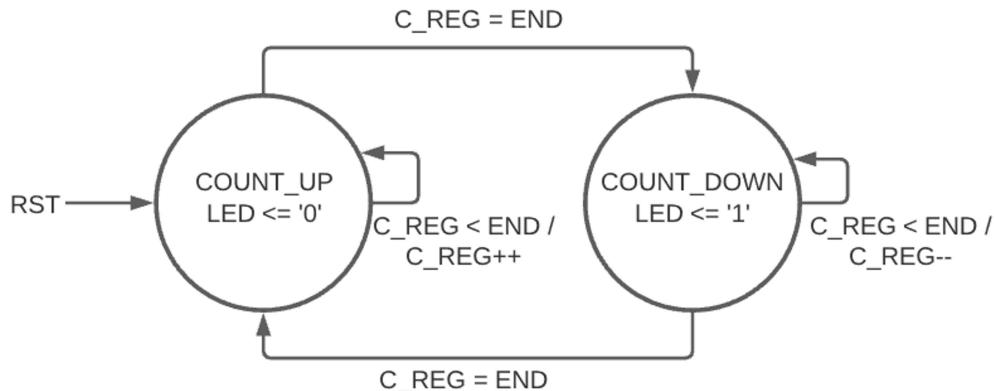
end architecture data_flow;
```

3. Realice una descripción en VHDL que haga destellar un led con una cadencia de 1 segundo, sabiendo que la frecuencia de clock es 50MHz. La entidad es la siguiente.

```
entity parpadeo1S is
  Port ( clk : in std_logic;
         rst : in std_logic;      -- Sincronico, activo bajo
         led : out std_logic);  -- Enciende con cero.
end parpadeo1S;
```

- a. Realice un diagrama en bloques del circuito a implementar. Indicando claramente como obtiene la base de tiempo de 1 segundo.
b. Implemente la arquitectura que corresponda con el diagrama de bloques anterior.

El clk tiene una frecuencia de 50MHz que es igual a un periodo de 20ns, por lo que debemos contar 50.10^6 ciclos para obtener un segundo. Para esto necesitaríamos un registro con N = 23 bits, el cual usaremos como contador.



```

architecture data_flow of parpadeo is
type state is (COUNT_UP, COUNT_DOWN);
attribute enum_encoding : string;
signal s_reg, s_next : state;

signal c_next, c_reg: unsigned (7 downto 0) := (others => '0');
constant COUNTER_END: unsigned (7 downto 0) := x"1F"; -- Count 0x1F = 37 cycles

signal tc: std_logic;
begin

state_register: process(rst, clk)
begin
    if rst = '1' then --EN EL CASO DE QUE SE TUVIESE UN RESET ASINCRONICO
        s_reg <= COUNT_UP;
    elsif rising_edge(clk) then
        s_reg <= s_next;
    end if;
end process state_register;

counter_register: process(rst, clk)
begin
    if rst = '1' then --EN EL CASO DE QUE SE TUVIESE UN RESET ASINCRONICO
        c_reg <= (others => '0');
    elsif rising_edge(clk) then
        c_reg <= c_next;
    end if;
end process counter_register;

--NEXT STATE LOGIC
s_next <= COUNT_DOWN when (tc = '1' and s_reg = COUNT_UP) else
    COUNT_UP when (tc = '1' and s_reg = COUNT_DOWN) else
    s_reg;

c_next <= (c_reg + 1) when (tc = '0') else (others => '0');

--MISC LOGIC
tc <= '1' when c_reg = COUNTER_END else '0';

--OUTPUT LOGIC
with s_reg select
    led <= '0' when COUNT_UP,
    '1' when COUNT_DOWN,
    '0' when others;

end architecture data_flow;

```

2. Implemente un circuito a nivel de compuertas que realice la siguiente operación de forma correcta

$$R = (A / 2) + B - (C * 4) - D$$

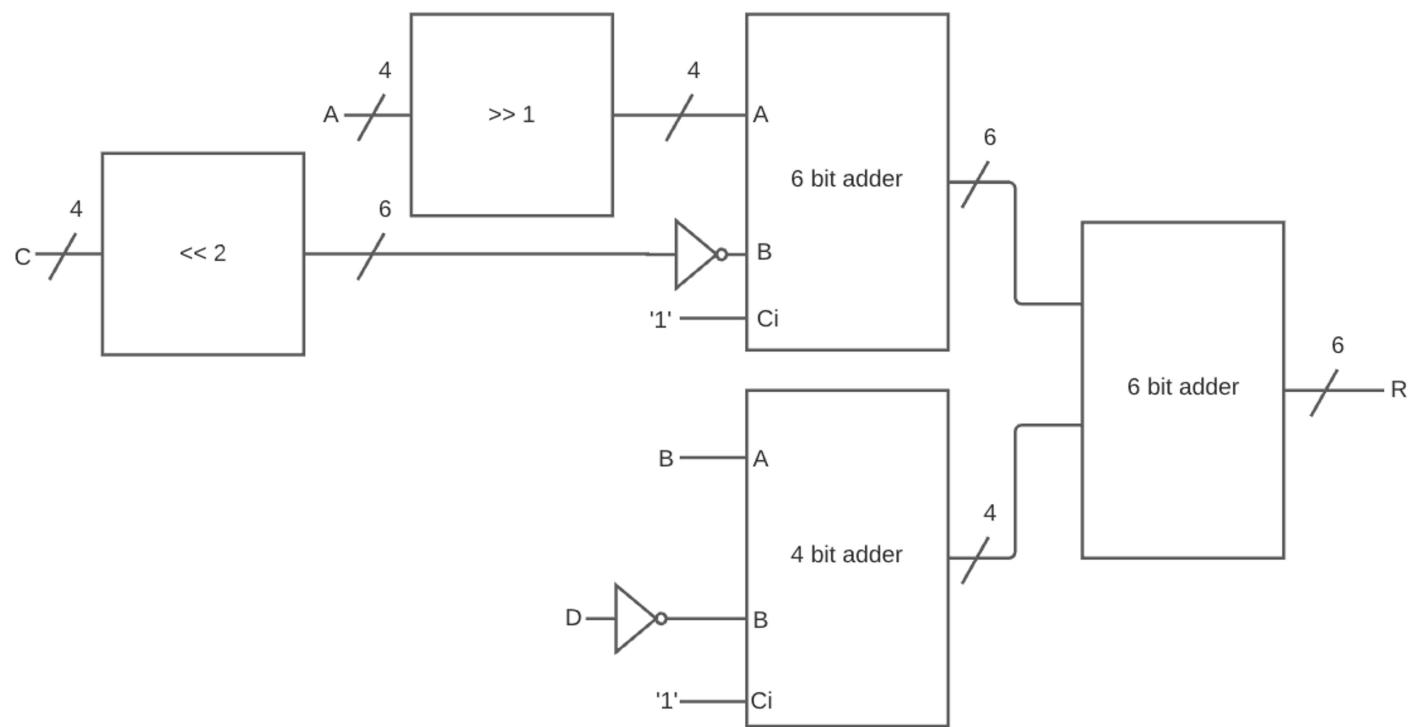
Donde, A,B,C y D son números signados en complemento a dos de 4 bits. R es el resultado de la operación de M bits

- Dibuje el diagrama en bloques de circuito. Indique claramente el valor de M.
- Dibuje el circuito a nivel de compuertas que se corresponda con el diagrama en bloques.
- Determine el tiempo de propagación máximo en función del tiempo de propagación de una compuerta.

Vamos a reescribir mediante el uso de operadores desplazamiento:

$$R = (A \gg 1) + B - (C \ll 2) - D$$

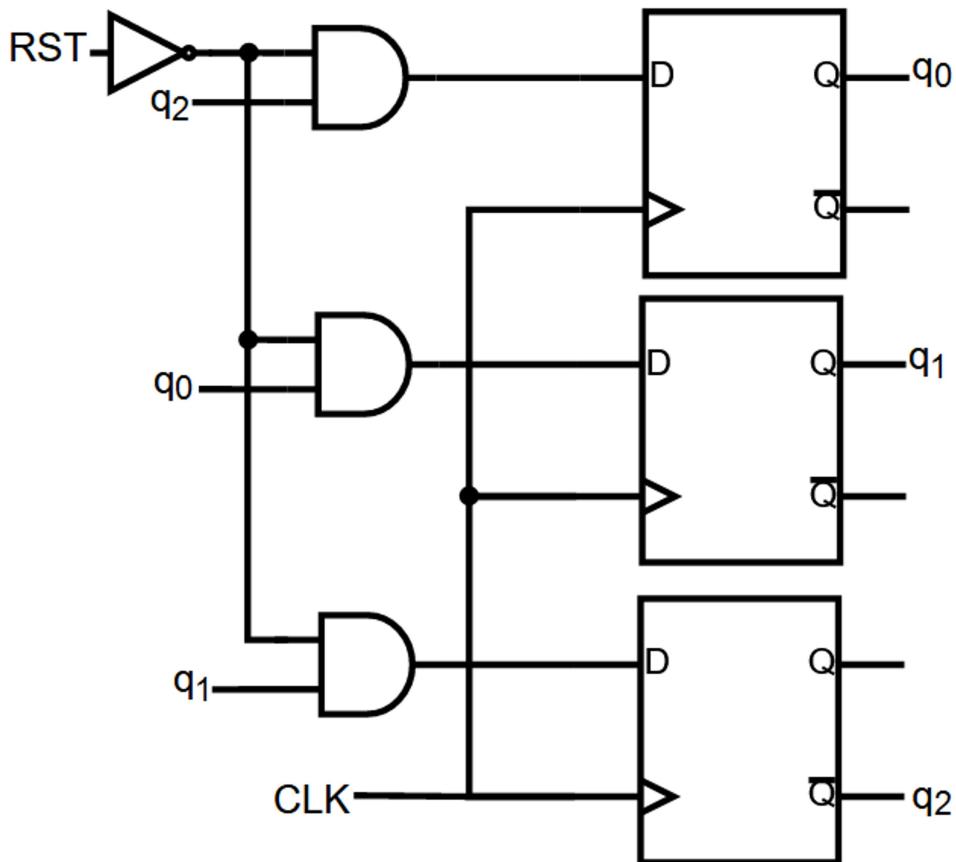
El valor de M estará dado por C ya que por tratarse de dos desplazamientos a izquierda, el resultado final será de 6 bits.



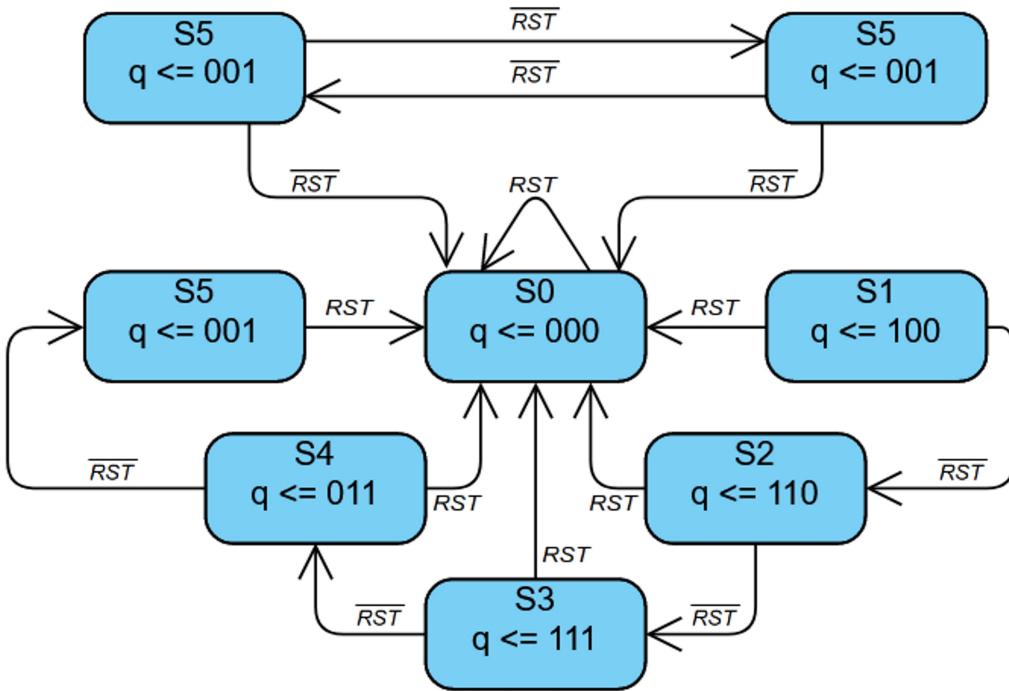
Final 20-12-17

Sunday, July 18, 2021 8:49 PM

1. Se dispone de un contador Johnson de 3 bits con reset sincrónico.
 - a. Dibuje el circuito completo.
 - b. Indique si es Mealy o Moore.
 - c. Dibuje el diagrama de estados completo.
 - d. Indique la frecuencia máxima de clock si los FFD son de tipo A



Es una maquina Moore ya que su salida solo esta dada por el estado de los flip-flop



Teniendo en cuenta el camino critico, q2 + inversor + and:

$$T_{clk_m} = t_{cq_{M(A)}} + t_{s_{M(A)}} + t_{next_M}$$

$$T_{clk_m} = t_{cq_{M(A)}} + t_{s_{M(A)}} + 2 \cdot t_{g_M}$$

$$T_{clk_m} = (4+4+2) \text{ ns}$$

$$T_{clk_m} = 10 \text{ ns}$$

$$f_{clk_M} = \frac{1}{T_{clk_m}} = \frac{1}{10 \text{ ns}} = 100 \text{ MHz}$$

3. Implemente en VHDL un circuito que invierta los bits de una palabra de entrada entity invertir is

```

generic (N : natural);
port (a: in std_logic_vector (N - 1 downto 0);
      b: out std_logic_vector (N - 1 downto 0));
end invertir;

```

Ejemplo: Si a <= "0001" entonces b <= "1000". No está permitido usar vectores con to

```

todo: for i in 0 to N-1 generate
    b(N-1-i) <= a(i);
end generate todo;

```

4. Implemente un circuito en VHDL que realice la resta ($b - a$) de dos números signados en CA2 de N bits con resultado de N bits. Si la operación produce overflow sature el resultado e indique dicha condición. Además realice un diagrama RTL

```
entity resta is
    generic (N : natural);
    port (a: in std_logic_vector (N - 1 downto 0);
          b: in std_logic_vector (N - 1 downto 0);
          r: out std_logic_vector (N - 1 downto 0);
          ov: out std_logic);
end resta;
```

```
auxA <= signed(a);
auxB <= signed(b);
auxR <= (auxB - auxA);

ovPos <= auxA(N-1) and (not (auxB(N-1))) and auxR(N-1); -- b > 0 - a < 0 -> r > 0

ovNeg <= (not (auxA(N-1))) and auxB(N-1) and (not (auxR(N-1))); -- b < 0 - a > 0 -> r < 0

r <= ('0', others => '1') when ovPos = '1' else
      ('1', others => '0') when ovNeg = '1' else
      STD_LOGIC_VECTOR(auxR);

ov <= (ovPos or ovNeg);
```

Final 08-02-17

Thursday, July 15, 2021 11:46 PM

1. Implemente un circuito que cuente la cantidad de unos en las entradas. El circuito dispone de 5 entradas.

- Determine la cantidad de salidas de circuito.
- Halle el circuito a nivel de compuertas y justifique su funcionamiento.

A. La cantidad máxima de unos es 5, para determinar la cantidad mínima de salidas que necesitamos podemos aplicar la siguiente expresión $2^N \geq 5$, por lo que $N = 3$.

B. Podemos hacer uso de sumadores completos binarios:

Ci	B	A	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$\begin{array}{c} B \\ A \\ C_i \\ \oplus \\ S \end{array}$

$\begin{array}{c} B \\ A \\ C_i \\ \oplus \\ S \end{array}$

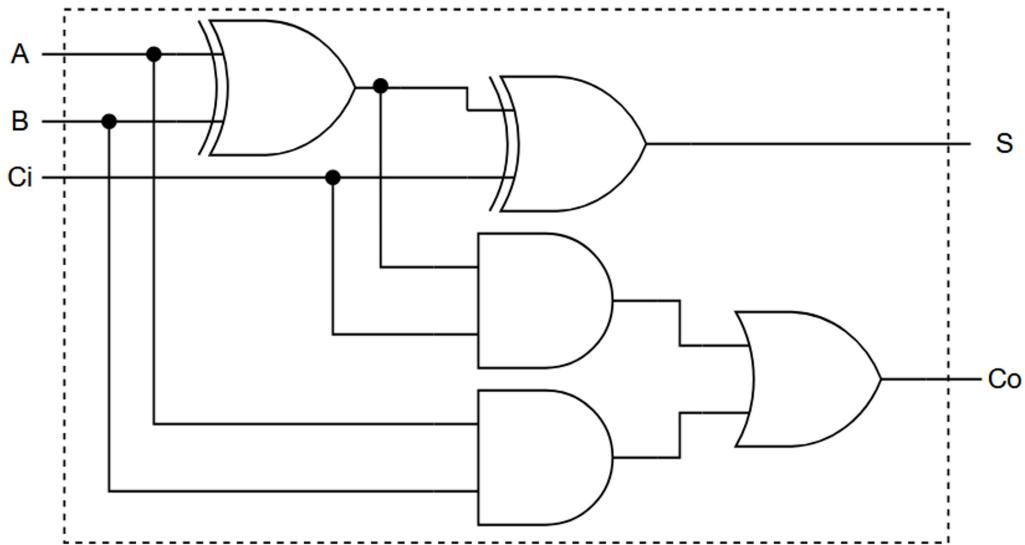
$$S = \overline{B}\overline{A}C_i + \overline{B}AC_i + BAC_i + B\overline{A}C_i$$

$$= \overline{C}_i(\overline{B}A + B\overline{A}) + C_i(BA + \overline{B}\overline{A})$$

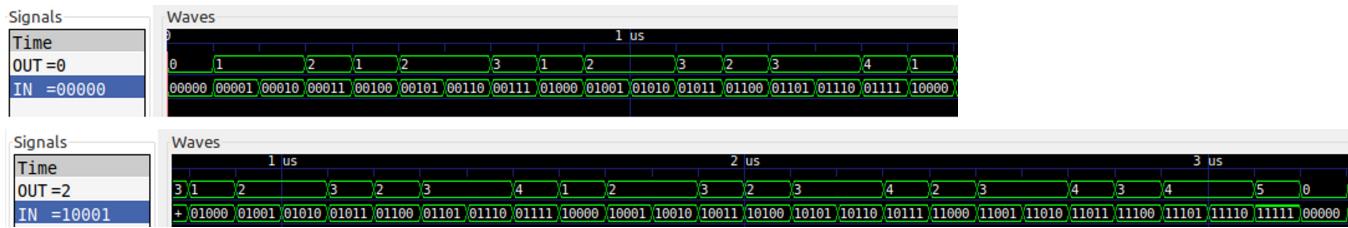
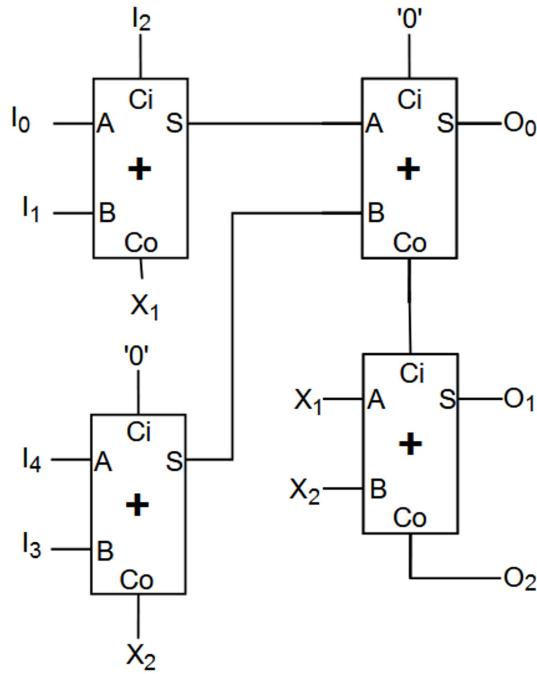
$$= \overline{C}_i(B \oplus A) + C_i(\overline{B} \oplus A) = C_i \oplus (B \oplus A)$$

$$C_o = BA + C_i\overline{B}A + C_iB\overline{A}$$

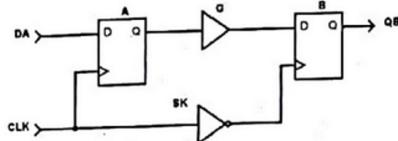
$$= BA + C_i(B \oplus A)$$



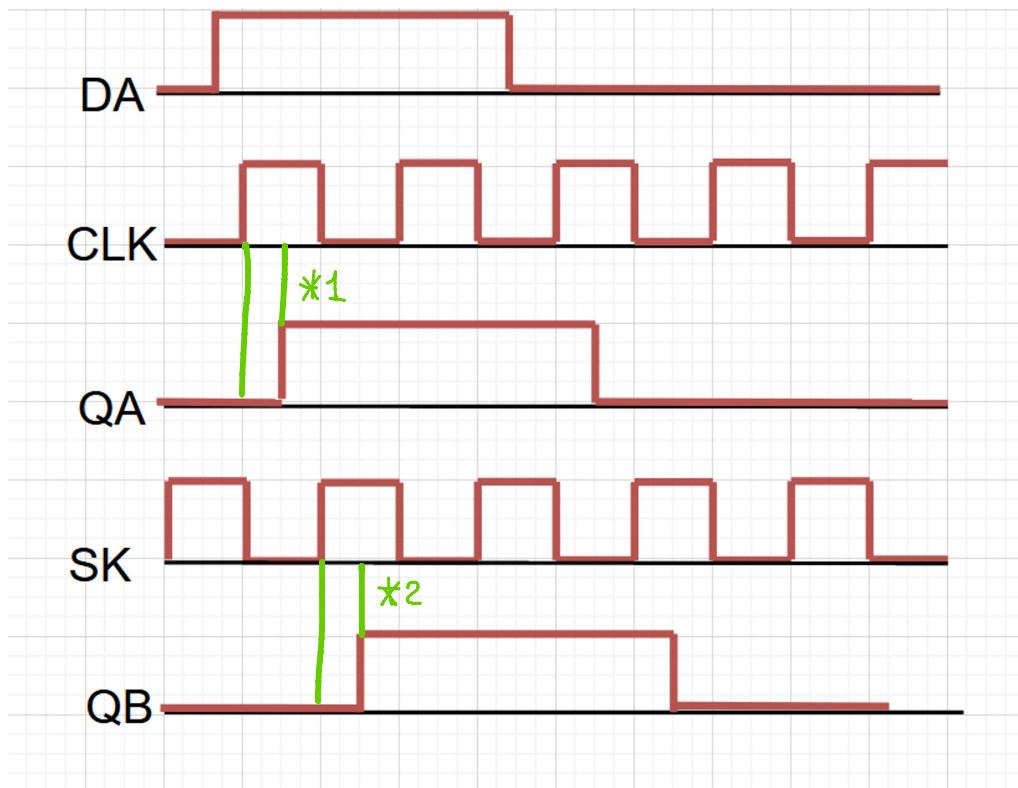
Podemos hacer uso de este bloque combinacional y obtener la sumatoria de todos los bits en '1' presentes en la entrada



- 2. Dado el siguiente circuito



- a. Dibuje el diagrama temporal correspondiente indicando todos los tiempos intervinientes para determinar la frecuencia máxima de clock.
- b. Escriba la expresión correspondiente a la frecuencia máxima del circuito.
- Tenga en cuenta que SK es un negador y su tiempo de propagación es cero.



3. Implemente en VHDL un circuito con la siguiente entidad que calcule la paridad par de un dato de entradas de N bits.

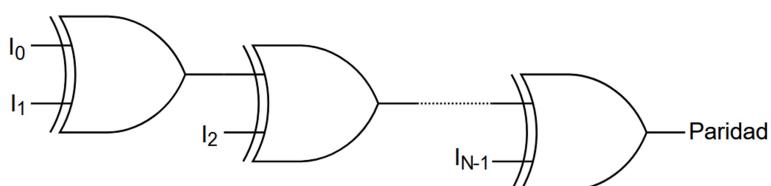
```
entity paridadPar is
    generic (N : integer := 8);
    Port (entrada : in std_logic_vector (N-1 downto 0);
          paridad: out std_logic);
end paridadPar;
```

Dónde:

- entrada: Entrada de datos de los que se calcula la paridad.
- paridad: Salida indicando la paridad par del dato.

Paridad par: Se cuenta la cantidad de unos. Si el total es impar, el bit de paridad es uno.

La paridad de una trama de bits se puede obtener mediante la operación XOR entre todos los componentes de la misma, es decir:



```
signal subRes: STD_LOGIC_VECTOR(N-2 downto 0);
begin
    todo: for i in 0 to N-2 generate
        primero: if i = 0 generate
            subRes(0) <= entrada(0) xor entrada(1);
        end generate primero;
        resto: if(i > 0 and i <= N-2) generate
            subRes(i) <= subRes(i-1) xor entrada(i+1);
        end generate resto;
    end generate todo;
    paridad <= subRes(N-2);
```

4. Explique la diferencia entre una maquina de estados Moore y Mealy

- Dibuje el diagrama de bloques de cada una de ellas.
- Dibuje un diagrama de estados Mealy de un detector de la secuencia B"10"
- Dibuje un diagrama de estados Moore de un detector de la secuencia B"10"

Para los diagramas de estado indique claramente cuál es el estado, la entrada y la salida.

- Moore: Su salida depende solo del estado en el que se encuentra
- Mealy: Su salida depende del estado en el que se encuentra y de una o mas entradas.

