UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería Electrónica
Técnicas Digitales I

Examen Final
23-12-2020

Apellido, nombre	Legajo	Cantidad de hojas	Nota

Problema 1 (4 puntos).
Diseñe un circuito a nivel RTL que tenga dos entradas (***A*** y ***B***) de N bits una entrada ***C*** (C_1C_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:

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

Problema 2 (2 puntos)

```
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*
t_0	0	1
t_1	1	1
t_2	0	1
t_3	1	0
t_4	0	0
t_5	0	1

Problema 3 (4 puntos).
Dada la secuencia 100-010-001-000-111-000-111

a) Determine el diagrama de estados, las tablas de estado futuro y de salidas.
b) Implemente con flip flops D codificando los estados en one-hot[1]. De las ecuaciones de excitación y las ecuaciones de las salidas.
c) ¿Cuántos estados no utilizados tiene esta implementación?
d) Implemente en VHDL con reset asincrónico. Para este punto puede utilizar la codificación de estados que desee.

[1] Dónde en la codificación del estado sólo puede haber un único flip-flop en uno a la vez.

GIOVANA,
GUIDO A.

LEGAJO: 156.926-7

23/12/20

① a)

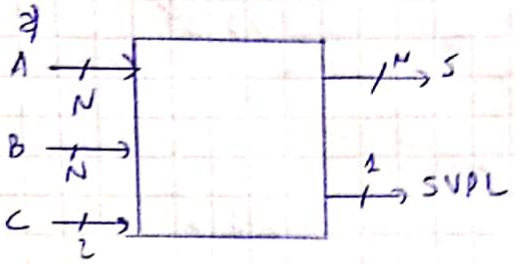
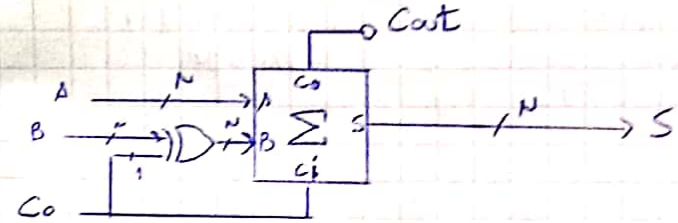
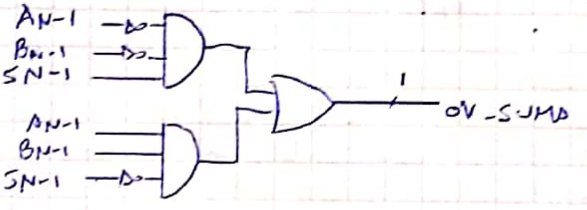
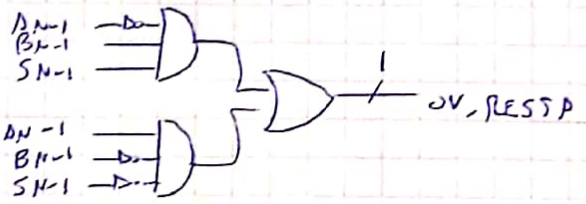


Diagrama RTL

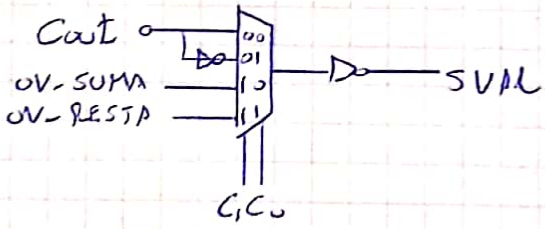




Circuito de OVERFLOW para la suma de dos números en CA2 (si OV_SUMA=1, error)



Circuito de OVERFLOW para la resta de dos números en CA2 (si OV_RESTA=1, error)



Si ④ en magnitudes el carry out "Cout" está en 1, significa que necesito ESE bit para representar correctamente la salida. Como la salida tiene N bits al igual que la entrada, debo descartar ese bit, por lo que entonces tengo error.

Si ⑤ para lo opuesto: al negar el número que resta y sumarle 1 (S = A - B es igual a S = A + $\bar{B}$ + 1), la operación es exitosa cuando Cout es 1, y errónea si Cout = 0 (en ambos casos se descarta ese bit)

Exitosos { Si $C_0 = 0$ y $C_{out} = 1$ → Error (si $C_1 = 0$ en ambos casos
Si $C_0 = 1$ y $C_{out} = 0$ → Error

b) A = "1101" , B = "0110"

CASO C: "00"

```
  1 1 0 1  (13)   ⇒  S: "0 0 1 1"
+ 0 1 1 0  (6)      SVAL = 0
(1) 0 0 1 1  (3)
```

cout ↗

CASO C: "01"

④
```
  1 1 0 1  (13)   ⇒  S: "0 1 1 1"
- 0 1 1 0  (6)      SVAL = 1
  0 1 1 1  (7)
```

cout

CASO C: "10"

```
  1 1 0 1  (-3)   ⇒  S: "0 0 1 1"
+ 0 1 1 0  (6)      SVAL = 1
  0 0 1 1  (3)
```

CASO C: "11"

```
  1 1 0 1  (-3)   →  S: "0 1 1 1"
- 0 1 1 0  (6)      SVAL = 0
  0 1 1 1  (7)
```

②

#	a	b	Ci	S	Co
t_0	0	1	0	1	0
t_1	1	1	0	0	1
t_2	0	1	1	0	1
t_3	1	0	1	0	1
t_4	0	0	1	1	0
t_5	0	1	0	1	0

$S = a \oplus b \oplus Ci$

$Co = ab + aCi + bCi$

RISING_EDGE(CLK) then
Ci <= Co

⇒ Para $t_0 \Rightarrow S = 1$
$t_1 \Rightarrow S = 0$
$t_2 \Rightarrow S = 0$
$t_3 \Rightarrow S = 0$
$t_4 \Rightarrow S = 1$
$t_5 \Rightarrow S = 1$

---

④
```
  1 1 0 1
- 0 1 1 0   ⇒   1 1 0 1
             + 1 0 1 0
              (1) 0 1 1 1
```
⇒ S: "0 111"
SVAL: 1

Cout: ↗

↳ Se ve que, aunque Cout SE DESCARTA, el 1 indica operación exitosa

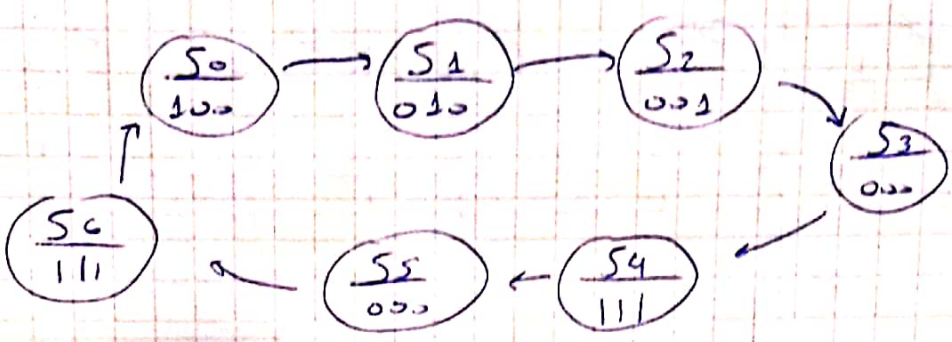
GIOVANA, GUIDO A.

LEGAJO: 156.986-7

23/12/20

③ 100 - 010 - 001 - 000 - 111 - 000 - 111

a)



S	S*	$Y_2 Y_1 Y_0$
S_0	S_1	1 0 0
S_1	S_2	0 1 0
S_2	S_3	0 0 1
S_3	S_4	0 0 0
S_4	S_5	1 1 1
S_5	S_6	0 0 0
S_6	S_7	1 1 1

b) Codificación ONE HOT

S_0 = 0000001
S_1 = 0000010
S_2 = 0000100
S_3 = 0001000
S_4 = 0010000
S_5 = 0100000
S_6 = 1000000

S_6 S_5 S_4 S_3 S_2 S_1 S_0	S_6^+ S_5^+ S_4^+ S_3^+ S_2^+ S_1^+ S_0^+	$Y_2 Y_1 Y_0$
0 0 0 0 0 0 1	0 0 0 0 0 1 0	1 0 0
0 0 0 0 0 1 0	0 0 0 0 1 0 0	0 1 0
0 0 0 0 1 0 0	0 0 0 1 0 0 0	0 0 1
0 0 0 1 0 0 0	0 0 1 0 0 0 0	1 1 1
0 0 1 0 0 0 0	0 1 0 0 0 0 0	0 0 0
0 1 0 0 0 0 0	1 0 0 0 0 0 0	1 1 1
1 0 0 0 0 0 0	0 0 0 0 0 0 1	

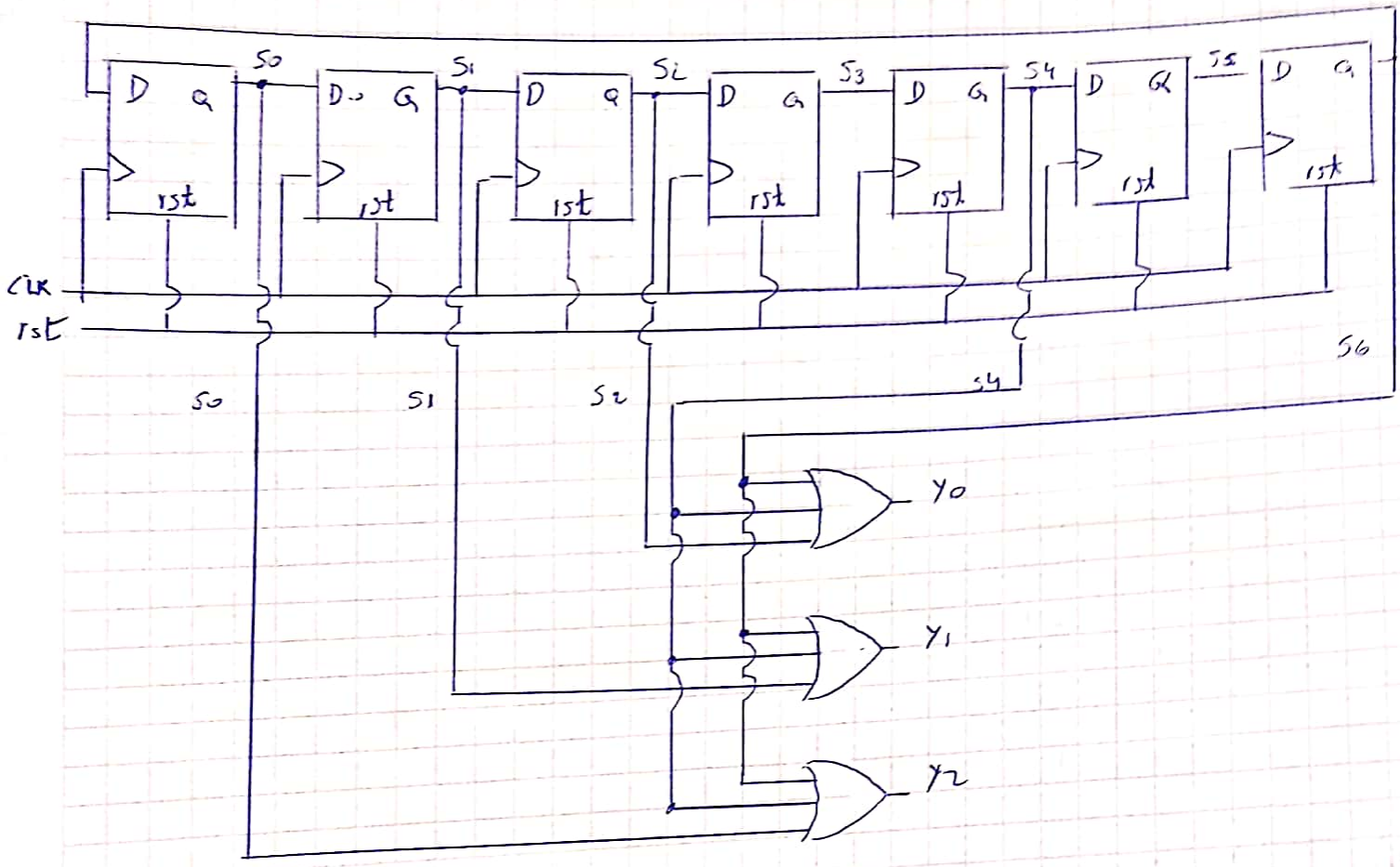
Todos los demás estados no representados son DON'T CARE.

$S_0^+ = S_6$
$S_1^+ = S_0$
$S_2^+ = S_1$
$S_3^+ = S_2$
$S_4^+ = S_3$
$S_5^+ = S_4$
$S_6^+ = S_5$

$Y_0 = S_2 + S_4 + S_6$

$Y_1 = S_1 + S_4 + S_6$

$Y_2 = S_0 + S_4 + S_6$



c) Si los flip-flop D No fueran ONE-HOT, podría codificarse de la siguiente manera:

$S_0: 000$
$S_1: 001$
$S_2: 010$
$S_3: 011$
$S_4: 100$
$S_5: 101$
$S_6: 110$

⇒ Con 3 FFD y la lógica combinacional correspondiente podría resolverse el contador

Es decir que, habiendo usado 7 FFD, TENEMOS 4 FFD que podrían haberse ahorrado.

Ahora, con 7 FFD, tengo 2^7 combinaciones para los estados (128). Como solo uno 6, nos sobran 122 combinaciones para posibles estados.

GIOVANA, GUIDO A.

LEGAJO 156.980-7

23/12/20

1)

```
LIBRARY IEEE
USE IEEE.std_logic_1164.ALL

entity PROBLEMA3 is
PORT( CLK, RST : in std_logic;
      Y : out std_logic_vector(3-1 downto 0);
    );
end entity PROBLEMA3;

architecture arch_PROBLEMA3 of PROBLEMA3 is
 type tstate (A,B,C,D,E,F,G)
 signal qNOW, qNEXT : tstate

 begin
 P1: process (CLK)                    -- LÓGICA DEL CLOCK
 begin
     if (rst='1' then
        qNOW <= A
     elseif RISING_EDGE(CLK) then
        qNOW <= qNEXT
     end if
 end process, P1;

 P2: process (qNOW)                   -- LÓGICA DE ESTADO FUTURO
 begin

     case qNOW is
        when A =>
          qNEXT <= B;

        when B =>
          qNEXT <= C;

        when C =>
          qNEXT <= D;

        when D =>
          qNEXT <= E;

        when E =>
          qNEXT <= F;

        when F =>
          qNEXT <= G;

        when G =>
          qNEXT <= A

        when other =>
```

```
            qNext <= A;
        end case;
    end process P2;
    P3: process (qNow)                    --LÓGICA DE SALIDA
    begin
        case qNow is
            when A =>
                Y <= "100";
            when B =>
                Y <= "010";
            when C =>
                Y <= "001";
            when D =>
                Y <= "000";
            when E =>
                Y <= "111";
            when F =>
                Y <= "000";
            when G =>
                Y <= "111";
            when others =>
                Y <= "100";
        end case;
    end process P3;
end architecture arch_PROBLEMA3;
```

Correcciones:

- En el Punto 3, inciso C, se usan **7** combinaciones, y no 6 como indiqué en el examen, por lo que en realidad **sobran 121 combinaciones** (y no 122).