UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES	**Ingeniería en Electrónica**	**Examen Final**
	Técnicas Digitales I	Fecha: 03/03/2022

Problema 1 (3 puntos): Diseñar un circuito secuencial sincrónico el cual posee una entrada de datos ***x*** sincrónica con los flancos positivos del reloj. El sistema a diseñar debe detectar la secuencia ***"101"***. Las secuencias detectadas no deben solaparse. El circuito posee dos salidas y_{mo} e y_{me}. Las mismas se deben comportar de la siguiente manera:

Y_{me}: se debe poner en ***'1'*** durante un solo ciclo de reloj cada vez que la entrada haya sido ***"10"*** en los dos ciclos previos de reloj y valga ***'1'*** en el actual.

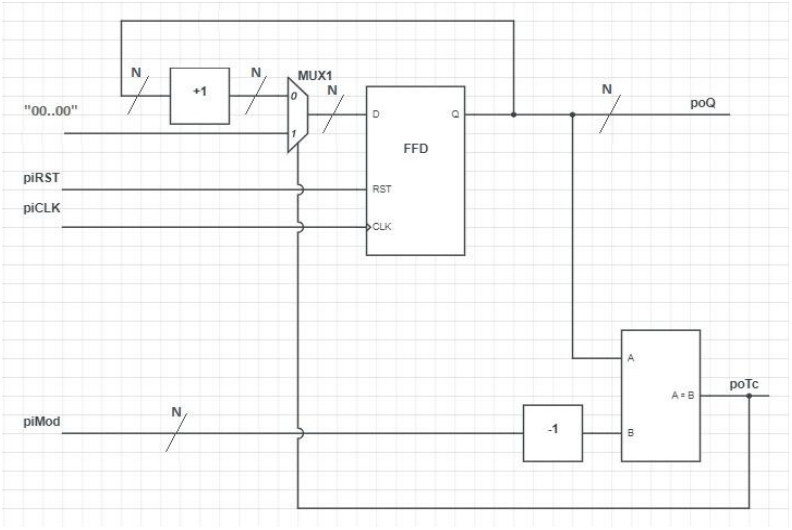
y_{mo}: se debe poner en ***'1'*** durante un solo ciclo de reloj cada vez que la entrada haya sido ***"101"*** en los tres ciclos previos de reloj.

Se pide Diagrama de estados, tablas, simplificación de ecuaciones y el circuito.
Usar las entradas de fuerza asincrónicas de los FFD para qué el circuito tenga reset al encenderse.

NOTA: No se permite el uso de registros de desplazamiento. Basar su diseño en maquinas de estado, realizando el desarrollo COMPLETO para la obtención del circuito.

Problema 2 (3 puntos): Describir en VHDL, utilizando un solo archivo para su descripción y sin usar instanciación de componentes el siguiente circuito. En la descripción debe quedar claramente indicado mediante comentarios que líneas de VHDL sintetizan cada submodulo.

```
Entity ContadorModProg is
Generic(N: NATURAL := 4);
Port(
   piClk : in std_logic;
   piRst : in std_logic;
   piMod : in std_logic_vector(N-1 downto 0);
   poTc : out std_logic;
   poQ : out std_logic_vector(N-1 downto 0)
);
```



Problema 3 (4 puntos): Utilizando bloques de propósito general diseñar un circuito que realice la suma de 4 variables (x_0,x_1,x_2,x_3) de tipo ENTERO (ca2) de 4 bits cada. El circuito debe entregar en su salida ***"y"*** de ***cinco bits*** el resultado de la suma. En caso de que dicho resultado no entre en 5 bits debe saturar el resultado.

Se pueden utilizar
- sumadores de 4 , 5 y 6 bits con entrada y salida de acarreo
- comparadores de magnitud de cualquier cantidad de bits
- multiplexores 2a1, 4a1, 8a1, 16a1, todos de N bits
- compuertas, las que considere necesarias

LAUERNER PEDRO

HOJA N°

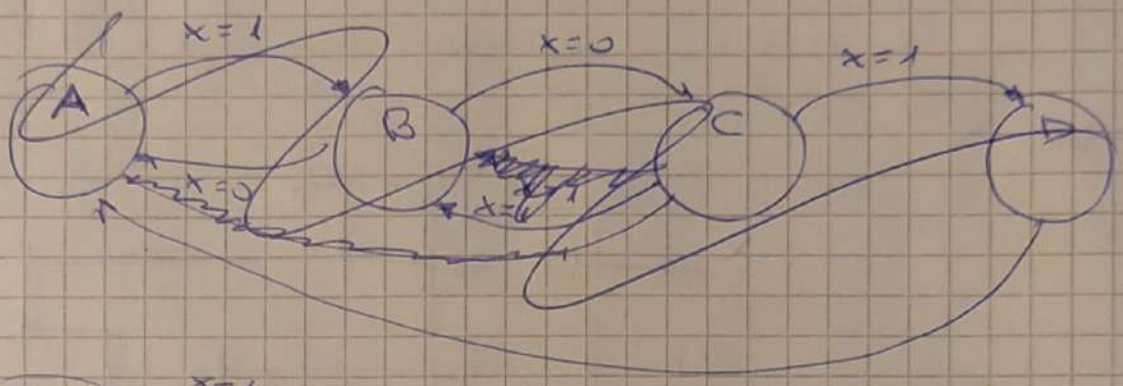
FECHA

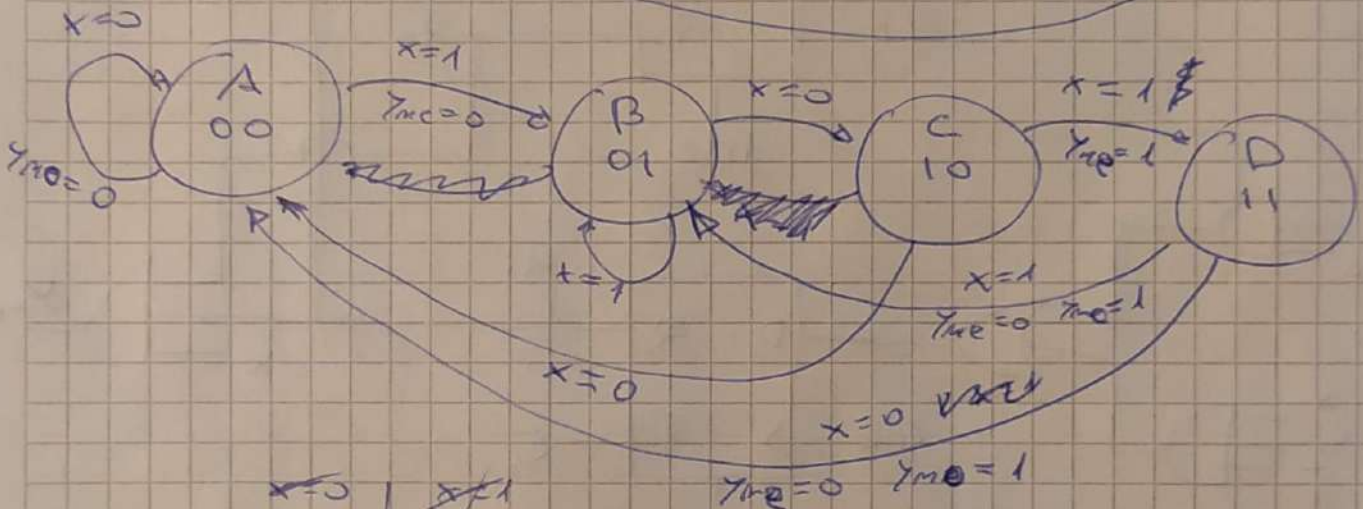
1) entrada x

detectar "101"

Salida Y_{mE} : '1' durante 1 ciclo cuando lee "0" y el actual es '1'

Y_{mO} : '1' durante un ciclo desp de leer "101"





			x=0		x=1	
x	Q_1	Q_0	D_1	D_0	D_1	D_0
0	0	0	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	1
0	1	1	0	0	0	0
1	0	0	0	1		
1	0	1	0	1		
1	1	0	1	1		
1	1	1	0	1		

x \ Q_1Q_0	00	01	11	10
0	0	0	0	0
1	1	1	1	1

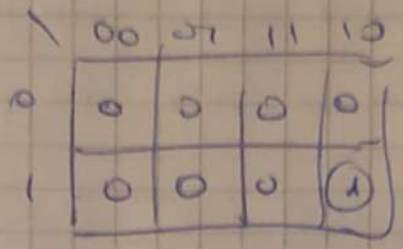
$D_0 = x$

x \ Q_1Q_0	00	01	11	10
0	0	1	0	0
1	0	0	0	1

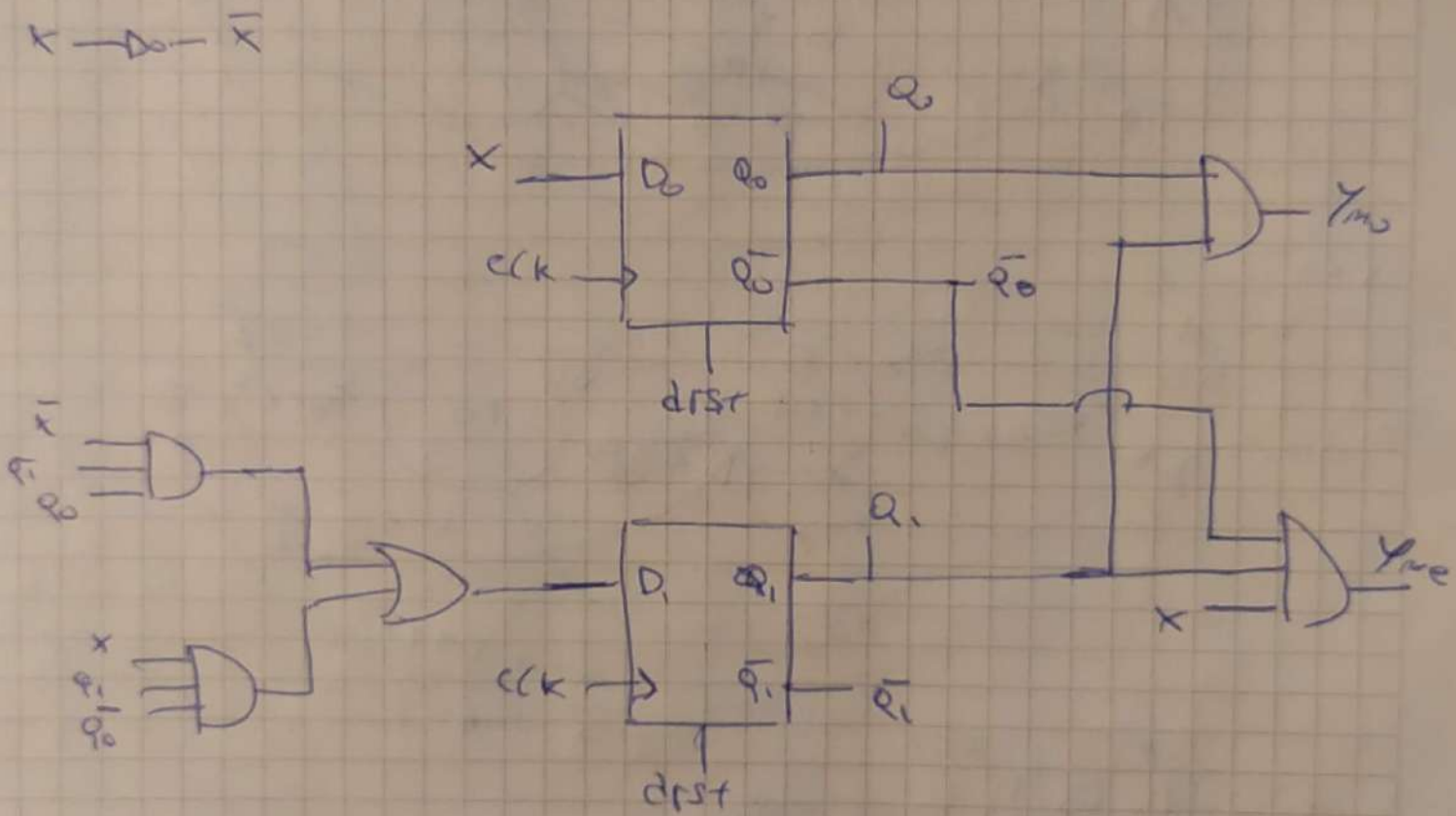
$D_1 = \bar{x} \cdot \bar{Q_1} \cdot Q_0 + x \cdot Q_1 \cdot \bar{Q_0}$

x	Q_1	Q_0	Y_{m0}	Y_{me}
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

$Y_{m0} = Q_1 Q_0$



$Y_{me} = x \cdot Q_1 \cdot \bar{Q_0}$



Zuckner Pedro

HOJA N°

FECHA

3) Sumar 4 variables de 4 bits cada una en Ca2. Salida y de 5 bits (saturar si no entra)

10000 (+ negativo) 01111 (+ positivo)

- Hago $X_0 + X_1$ con MSB negado (binario desp) (5 bits extendido antes MSB)
- Hago $X_2 + X_3$ con MSB negado (5 bits extiendo) antes MSB)
- Suma $\overbrace{(X_0+X_1)}^{A} + \overbrace{(X_2+X_3)}^{B}$ con MSB negado (5 bits) (binario desp).

5+5=10

5+7=12

5-7=-2

-5+7=2

-5-7=-12

$$\begin{array}{r} -4 \;\; 100 \\ -4 \;\; 100 \\ \hline 1000 \end{array}$$

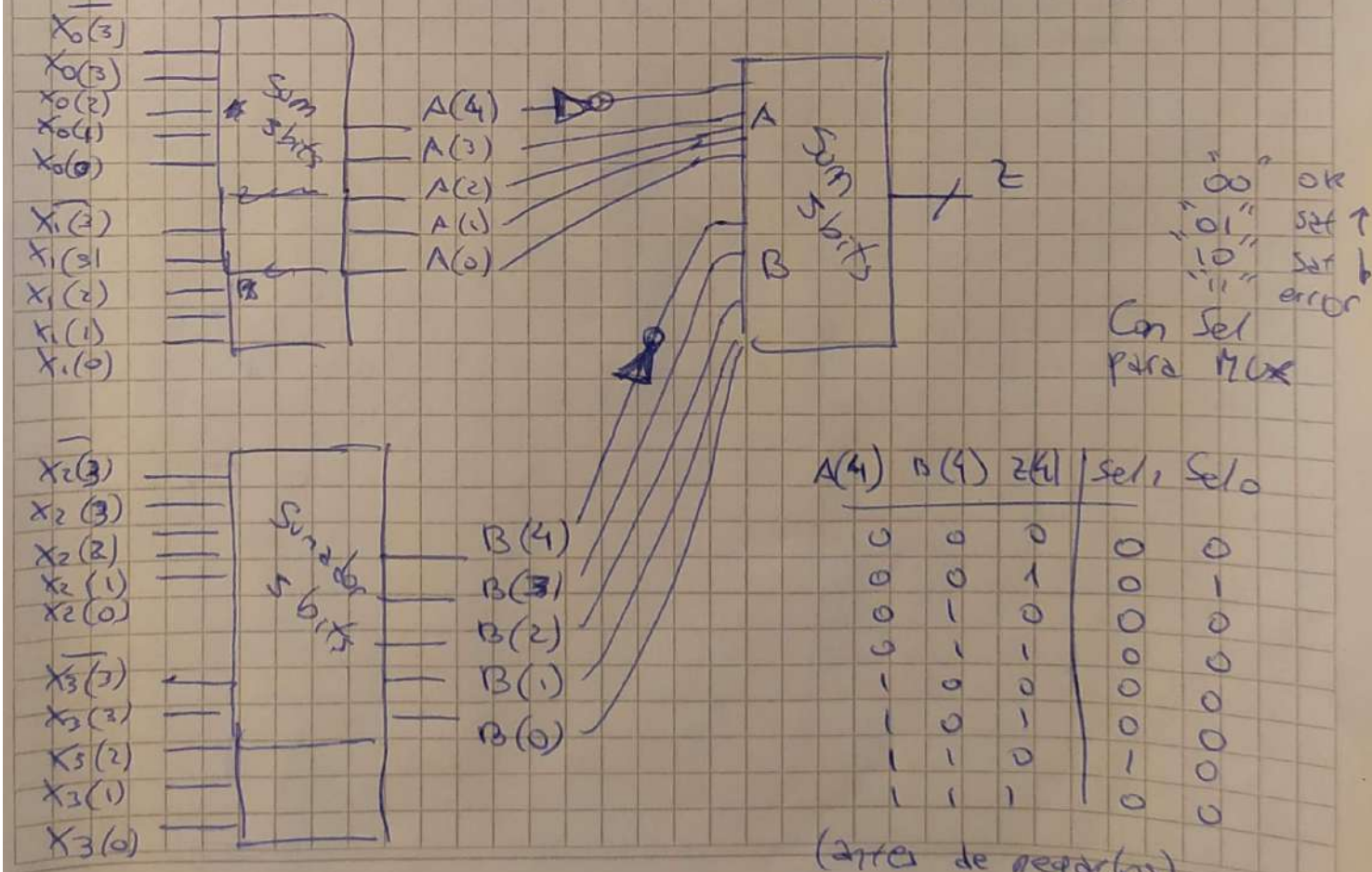
chequeo

A neg y B neg : resul neg

A pos y B pos : resul pos

MSB = 1 ambos : resul MSB = 1

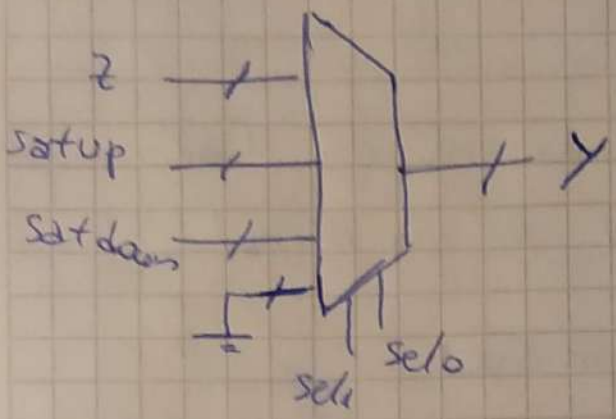
MSB = 0 ambos : resul



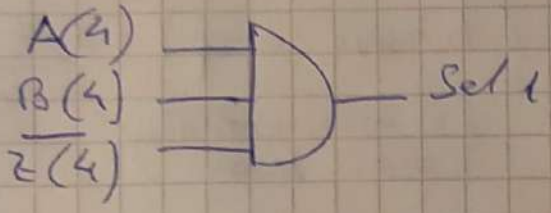
A(4)	B(4)	Z(4)	Sel1	Sel0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	0

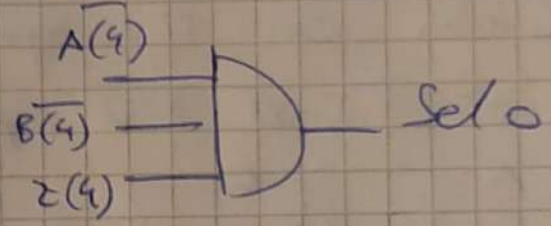
(antes de negar(os))

NOTA



setup = "01111"
Set down = "10000"





2) Ver archivo VHD.

```vhdl

-- EJERCICO 2 PARCIAL LAUCKNER PEDRO

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Ejer2 is

    generic(N: NATURAL :=4);

    port(
        piClk : in std_logic;
        piRst : in std_logic;
        piMod : in std_logic_vector(N-1 downto 0);

        poTc : out std_logic;
        poQ : out std_logic_vector(N-1 downto 0)

    );

end entity;


architecture A_Ejer2 of Ejer2 is

    signal sTc : std_logic;
    signal sQ : std_logic_vector(N-1 downto 0);
    signal sD : std_logic_vector(N-1 downto 0);
    signal sMuxIn : std_logic_vector(N-1 downto 0);
    signal sB : std_logic_vector(N-1 downto 0);


begin

    sMuxIn <= std_logic_vector( unsigned(sQ) + 1 );     -- Tomo sQ le sumo uno y lo pongo en la primer entrada del MUX

    sD <= sMuxIn when sTc = '0' else (others => '0');  -- MUX selecciona entrada con sTc y pone en la salida sMuxIn o "00...00"

    sB <= std_logic_vector( unsigned(piMod) - 1 );      -- Resto 1 a la entrada piMod y lo pongo en B

    sTc <= '1' when sB = sQ else '0';       -- comparo sQ y sB, si son iguales sTc es 1

    poQ <= sQ;          -- asigno las signals a las salidas
    poTc <= sTc;


    process(piClk, piRst)

    begin

        if(piRst = '1') then

            sQ <= (others => '0');

        elsif(rising_edge(piClk)) then

            sQ <= sD;

        end if;

    end process;

end architecture;
```