UNIVERSIDAD NACIONAL DE LA PLATA

E1212 - TÉCNICAS DIGITALES

TRABAJO PRÁCTICO Nº 6

FACULTAD DE INGENIERÍA

Análisis y Síntesis de Sistemas Digitales

Ejercicio 1

Un sistema digital está caracterizado por las siguientes funciones lógicas donde D_i y Q_i son las funciones de entrada y salida a los flip-flops tipo D,

X es la entrada del sistema y Z la salida.

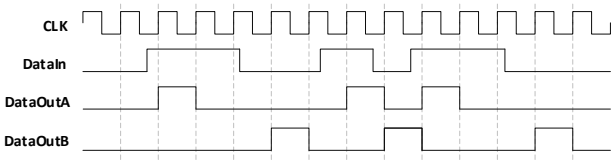
$$D_2 = XQ_2\overline{Q_0} + XQ_1Q_0 \qquad D_0 = X\overline{Q_0} + \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}$$

$$D_1 = X\overline{Q_2}\,\overline{Q_1}\,Q_0 + XQ_1\overline{Q_0} \qquad Z = \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0} + X\,\overline{Q_2}\;\overline{Q_0} + X\overline{Q_2}\,\overline{Q_1}$$

1. ¿El sistema descripto es de tipo Mealy o tipo Moore? Justificar.
2. Obtener la tabla de excitación de los flip-flops y el diagrama de transición de estados. ¿Cuántos estados indefinidos hay? ¿Qué se hace con ellos para evitar errores?
3. ¿Qué función realiza el sistema? Justificar realizando el diagrama de tiempo para el caso en X=1 y X=0.

Ejercicio 2

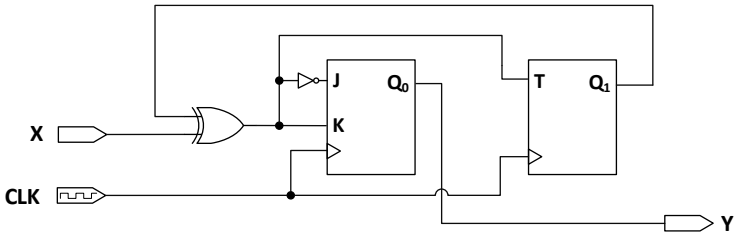
Sintetizar un circuito monoestable basado en el modelo de Moore. El monoestable contará con una entrada de datos *DataIn* y dos salidas. La salida *DataOutA* se pondrá en alto durante un ciclo de reloj cuando *DataIn* pase de bajo a alto mientras que *DataOutB* hará lo mismo, pero cuando *DataIn* pase de alto a bajo como se muestra en el siguiente diagrama de tiempos. Considerar que *DataIn* tiene variaciones más lentas que la señal de reloj.



1. Diseñar la máquina de estados finitos realizando el diagrama de estados, la tabla de excitación de los flip-flops y el circuito esquemático.
2. Comparar los resultados con los obtenidos en el ejercicio 4 del TP Nº 3.
3. Repetir el diseño utilizando el modelo de Mealy.

Ejercicio 3

El siguiente circuito implementa una máquina de estados que es un detector de repetición en la entrada de datos X. Es decir, la salida Y se pondrá en alto durante un ciclo de reloj cuando la entrada X sea dos veces consecutivas “0” o dos veces consecutivas “1”.



1. Determinar si el sistema es tipo Mealy o tipo Moore.
2. Obtener el diagrama y la tabla de excitación de los flip-flops a partir del diagrama esquemático.
3. Si la respuesta al inciso 1 fue de tipo Mealy implementar el sistema mediante Moore. Caso contrario implementar mediante Mealy. Realizar la tabla de excitación y diagrama de estados junto con el circuito esquemático.
4. Comparar ambas implementaciones. ¿Por qué una de ellas requiere menos estados?
5. Implementar ambas máquinas de estado en VHDL, simular y verificar los resultados.

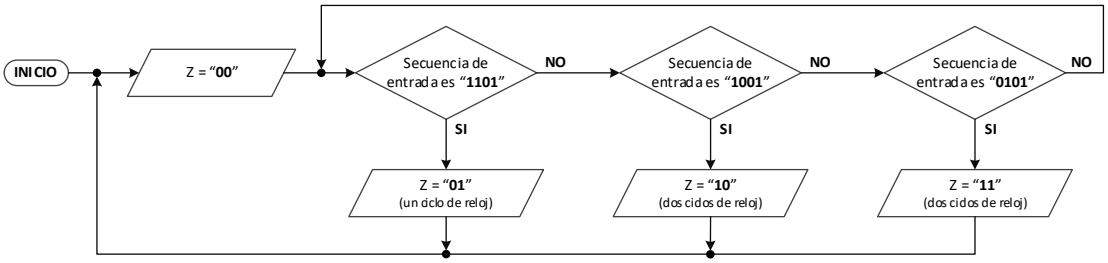
Ejercicio 4

Se requiere implementar un detector de secuencias con la menor cantidad de recursos lógicos posible que responda a los siguientes algoritmos, donde Z es la salida del sistema. Analizar el problema realizando la tabla o el diagrama de transición de estados en cada caso.

1. El algoritmo responde al siguiente pseudocódigo:

```
Z en nivel bajo
Inicio bucle infinito
        Si secuencia de entrada es “1101” entonces
                        Z en nivel alto un período de reloj
        SiNo Si secuencia de entrada es “1001” entonces
                        Z en nivel alto un período de reloj
        SiNo Si secuencia de entrada es “0101” entonces
                        Z en nivel alto un período de reloj
        Fin Si
Fin bucle infinito
```

2. El algoritmo responde al siguiente diagrama de flujo:



Ejercicio 5

En el código que se puede encontrar al final de esta práctica está codificado un generador de paridad serie. Los datos entran en serie por el puerto DATAIN a la misma tasa que la frecuencia de reloj CLK. Una máquina de estados releva los datos en palabras de 3 bits y genera el bit de paridad adecuado. La palabra de 3 bits se convierte posteriormente a formato paralelo que se presenta en el puerto DATAOUT donde los bits 2 a 0 son los bits serie recibidos (el bit 2 corresponde al primer bit de la palabra) y el bit 3 es el bit de paridad generado. La señal de salida DATAREADY se pone en alto para indicar que una nueva palabra está lista para leerse en DATAOUT.

1. Identificar si el tipo de implementación es Mealy o Moore.
2. Realizar el diagrama de transición de estados en base al código VHDL. Comparar con el diagrama generado por el Quartus II.
3. Analizar el funcionamiento del sistema e identificar el tipo de paridad generado (par o impar).
4. Simular el sistema y verificar el inciso anterior.
5. En base al diagrama generar la tabla de transición de estados y compararla con la de Quartus II. Obtener las funciones lógicas de excitación de los flip-flops considerándolos como tipo D.
6. ¿Cómo modificaría el código y la máquina de estados para cambiar el tipo de paridad?

Ejercicio 6

Se requiere diseñar un sistema digital detector de secuencia de 3 bits basado en el modelo de Moore. El sistema cuenta con dos entradas, una por donde ingresan los datos en serie *bitsIn* y otra entrada arbitraria denominada *dataIn.* Además, cuenta con una salida *dataOut* que deberá permanecer en bajo hasta que se detecte la secuencia "101" en *bitsIn.* Cuando esto ocurra *dataOut* deberá ser igual a *dataIn* y deberá permanecer de esa manera hasta que se detecte la secuencia "001" en *bitsIn*. Cuando esto ocurra *dataOut* deberá ser igual a */dataIn* y deberá permanecer en ese estado a la esperar de detectar nuevamente la primera de las secuencias repitiendo el proceso.

1. Diseñar el sistema partiendo del diagrama de transición de estados
2. En base al diagrama realizar la tabla de excitación utilizando flip-flops tipo D disparados por flanco descendente.
3. Obtener las funciones lógicas involucradas en la máquina de estados y realizar el diagrama esquemático del circuito.
4. Generar el código en VHDL de la implementación y simular para verificar el funcionamiento.

Ejercicio 7

Se pretende diseñar un generador de onda cuadrada programable basado en el modelo de Moore que permita configurar el ciclo de trabajo de la forma de onda de salida con los siguientes requerimientos:

- El generador deberá contar con una señal *ctrl* para configurar el ciclo de trabajo según la siguiente tabla donde a mayor ciclo de trabajo mayor será la proporción de tiempo en alto en la forma de onda de salida.

CTRL	Ciclo de trabajo
00	66.6%
01	50%
10	40%
11	33.3%

- El nivel alto de la forma de onda de salida debe corresponder siempre a dos períodos de reloj. Por ejemplo, si el ciclo de trabajo fuese 25% la forma de onda de salida deberá estar 2 ciclos de reloj en alto y otros 6 en bajo. De igual manera, si fuese 10% la relación alto/bajo en ciclos de reloj deberá ser 2/18.

1. Realizar el diagrama de estados, obtener las funciones lógicas de excitación de los flip-flops y dibujar el diagrama esquemático del circuito para una implementación con flip-flops tipo D.
2. Implementar el diseño en VHDL y verificar su funcionamiento.

Código Ejercicio 5

```
library ieee;
use ieee.std_logic_1164.all;

entity EJ5 is
port(
    -- In Control
    CLK : IN std_logic;
    RSTn : IN std_logic;

    -- In Data
    DATAIN : IN std_logic;
    -- Out Data
    DATAOUT : OUT std_logic_vector(3 downto 0);
    DATAREADY : OUT std_logic
);

end EJ5;

architecture Arq of EJ5 is
    type State_type is (S0,S1a,S1b,S2a,S2b,S3a,S3b);
    signal fsmState : State_type;
    signal parityBit : std_logic;
    signal dataOutAux : std_logic_vector(3 downto 0);
    begin

-- Control de estados
    ctrlPr: process(RSTn,CLK)

begin
    if RSTn = '0' then
    fsmState <= S0;
elsif rising_edge(CLK) then

case fsmState is
    when S0 =>
        if DATAIN = '1' then
            fsmState <= S1a;
        else
            fsmState <= S1b;
        end if;
    when S1a =>
        if DATAIN = '1' then
            fsmState <= S2b;
```

```
-- Control de salidas
outPr: process (fsmState,DATAIN)
begin
    DATAREADY <= '0';
    case fsmState is
        when S0 =>
            if DATAIN = '1' then
                parityBit <= '1';
            else
                parityBit <= '0';
            end if;
        when S1a =>
            if DATAIN = '1' then
                parityBit <= '0';
            else
                parityBit <= '1';
            end if;
        when S1b =>
            if DATAIN = '1' then
                parityBit <= '1';
            else
                parityBit <= '0';
            end if;
        when S2a =>
            if DATAIN = '1' then
                parityBit <= '0';
            else
                parityBit <= '1';
            end if;
        when S2b =>
            if DATAIN = '1' then
                parityBit <= '1';
            else
                parityBit <= '0';
            end if;
        when S3a =>
                parityBit <= '1';
                DATAREADY <= '1';
        when S3b =>
                parityBit <= '0';
                DATAREADY <= '1';
        when others =>
                parityBit <= '0';
```

```
        else
            fsmState <= S2a;
        end if;
    when S1b =>
        if DATAIN = '1' then
            fsmState <= S2a;
        else
            fsmState <= S2b;
        end if;
    when S2a =>
        if DATAIN = '1' then
            fsmState <= S3b;
        else
            fsmState <= S3a;
        end if;
    when S2b =>
        if DATAIN = '1' then
            fsmState <= S3a;
        else
            fsmState <= S3b;
        end if;
    when S3a =>
        fsmState <= S0;
    when S3b =>
        fsmState <= S0;
    when others =>
        fsmState <= S0;
end case;
end if;
end process;
-- Continúa ->>
```

```
        end case;
end process;

-- Conversión serie / paralelo
dataOutAux <= (others=>'0') when RSTn = '0' else parityBit &
dataOutAux(1 downto 0) & DATAIN when rising_edge(CLK);
DATAOUT <= dataOutAux;

end Arq;
```