



SISTEMAS DIGITALES II

DEBER PARA SEGUNDA EVALUACIÓN

I TÉRMINO 2014-2015

PROBLEMA # 1

Diseñe un pequeño Sistema Digital **CIRCUITO IDENTIFICADOR DE NUMEROS MINIMO Y MAXIMO.**

El circuito debe tener una entrada de **DATOS**, representada por un teclado decimal. Los números pueden estar en el rango entre 00 y 99. Para ingresar cada dígito de los números, se debe activar la entrada **ENTER**. El circuito puede recibir **N** números. La salida del circuito debe tener dos displays de 7 segmentos, que inicialmente están en cero. Cada vez que se ingresa un número, su valor se presenta en los displays. Después de ingresar los **N** números, se activa la entrada **START**. El circuito debe analizar los números ingresados y, después, debe mostrar, por medio de los displays, el número **MIN** y el número **MAX**. Si todos los números ingresados han sido iguales, los displays no mostrarán nada, es decir, deben estar apagados.

Presente:

1. Diagrama general de entradas/salidas
2. Diagrama de bloques de su solución
3. Partición Funcional
4. Diagrama ASM del circuito **Controlador** debidamente documentada. (indicar todos las entradas y salidas)

Nota. Asuma que la implementación será hecha usando tecnología FPGA y lenguaje VHDL. Dimensionar adecuadamente cada componente.

PROBLEMA # 2

Se debe diseñar un sistema digital RECEPTOR-ALMACENADOR de datos que formará parte de un sistema de comunicación más complejo.

Un transmisor envía una trama compuesta de tres bytes, la que es recibida en forma SERIAL en el lado del receptor quien debe almacenarlas en una memoria RAM (512 x 8) hasta completar 128 tramas que serán utilizadas por otro bloque quien correrá un algoritmo de encriptación de información usando un algoritmo matemático.

Para almacenar en la memoria RAM cada trama, el receptor debe considerar que debe almacenar los datos en el orden D3, D1, D2, siendo D1 el primer byte de la trama, D2 el segundo byte de la trama y D3 el tercer dato de la trama.

El transmisor envía la señal Inicio_Trama cada vez que le pasa una trama de tres bytes al receptor. Una vez almacenadas las 128 tramas, el receptor se queda esperando por la señal LISTO que le debe enviar el bloque de encriptación de datos y una vez hecho esto el sistema RECEPTOR-ALMACENADOR vuelve a esperar por la señal Inicio_Trama para volver a iniciar un nuevo proceso de recepción y almacenamiento.

Presentar:

1. **Diagrama de bloques** del Sistema Digital.
2. **Partición Funcional** del Sistema Digital.
3. **Diagrama ASM** del circuito **Controlador** debidamente documentado. (indicar todos las entradas y salidas)

Nota. Asuma que la implementación será hecha usando tecnología FPGA y lenguaje VHDL. Dimensionar adecuadamente cada componente.

PROBLEMA # 3

Dada la siguiente descripción en **VHDL** del funcionamiento de un **Sistema Digital**:

Presentar:

4. **Partición Funcional** del Sistema Digital.
5. **Diagrama ASM** del circuito Controlador del Sistema Digital, indicando claramente todas las salidas que deben ser generadas.
6. **Diagramas de Tiempo** del circuito **Controlador** asumiendo las condiciones de entrada dadas. Indique claramente los nombres y la duración de cada estado (**y**).

```

library ieee;
use ieee.std_logic_1164.all;

entity tema2 is
    port(Resetn, Clock : in std_logic;
          Inicio, LdReg, Mostrar : in std_logic;
          DataA, DataB : in std_logic_vector(3 downto 0);
          Done, BitR : out std_logic);
end tema2;

architecture mixta of tema2 is
    type estado is (Ta, Tb, Tc, Td);
    signal y : estado;
    signal EnA, LdA, EnB, LdB, EnR, LdR, EnC, LdC, smux, zero : std_logic;
    signal EnM, MR, AgtB, AmayorB, Cig0, vcc : std_logic;
    signal S : std_logic_vector(1 downto 0);
    signal A, B, R, Cnt, zeros4, tres : std_logic_vector(3 downto 0);

    component registro_d_i is
        port (Resetn, Clock, En, Ld, L : in std_logic;
              Entpar : in std_logic_vector (3 downto 0);
              Q : buffer std_logic_vector (3 downto 0));
    end component;

    component registro_i_d is
        port (Resetn, Clock, En, Ld, R : in std_logic;
              Entpar : in std_logic_vector (3 downto 0);
              Q : buffer std_logic_vector (3 downto 0));
    end component;

    component contador_down is
        port (Resetn, Clock, En, Ld : in std_logic;
              Ent : in std_logic_vector(3 downto 0);
              Q : buffer std_logic_vector(3 downto 0));
    end component;

    component ffd is
        port(Resetn, Clock, En, D : in std_logic;
              Q : out std_logic);
    end component;
begin

```

```

-- Circuito Controlador
MSS_transiciones: process(Resetn, Clock)
begin
    if Resetn = '0' then y <=Ta;
    elsif Clock'event and Clock = '1' then
        case y is
            when Ta=> if Inicio = '0' then y <=Ta; else y <=Tb; end if;
            when Tb=> if Cig0 = '0' then y <=Tb; else y <=Tc; end if;
            when Tc=> if Mostrar = '0' then y <=Tc; else y <=Td; end if;
            when Td=> if Cig0 = '0' then y <=Td; else y <=Ta; end if;
        end case;
    end if;
end process;
MSS_salidas: process(y, LdReg, AmayorB, B(0))
begin
    EnA <='0'; LdA <='0'; EnB <='0'; LdB <='0'; EnR <='0'; LdR <='0';
    EnC <='0'; LdC <='0'; EnM <='0'; S <="00"; Mr <='0'; Done <='0';
    case y is
        when Ta=> EnR<='1'; LdR <='1'; EnC <='1'; LdC <='1';
            if LdReg = '1' then EnA <='1'; LdA <='1'; EnB <='1'; LdB <='1'; end if;
            if Inicio = '1' then EnM <='1'; end if;
        when Tb=> EnA <='1'; EnB <='1'; EnR <='1'; EnC <='1';
            if (A(3) = '1' and AmayorB = '0') or (A(3) = '0' and B(0) = '0')
            then S(0) <='1'; end if;
            if A(3) = '0' then S(1) <='1'; end if;
        when Tc=> Done <='1';
            if Mostrar = '1' then EnC <='1'; LdC <='1'; end if;
        when Td=> EnC <='1'; Mr <='1'; EnR <='1';
    end case;
end process;

```

```
-- Procesador de datos
```

```
zero <='0'; zeros4 <="0000"; vcc <='1';
```

```
tres <="0011";
```

```
regA: registro_d_i port map(Resetn, Clock, EnA, LdA, zero, DataA, A);
```

```
regR: registro_d_i port map(Resetn, Clock, EnR, LdR, smux, zeros4, R);
```

```
regB: registro_i_d port map(Resetn, Clock, EnB, LdB, zero, DataB, B);
```

```
ff: ffd port map(Resetn, Clock, EnM, AgtB, AmayorB);
```

```
cnt_down: contador_down port map(Resetn, Clock, EnC, LdC, tres, Cnt);
```

```
AgtB <='1' when A > B else '0';
```

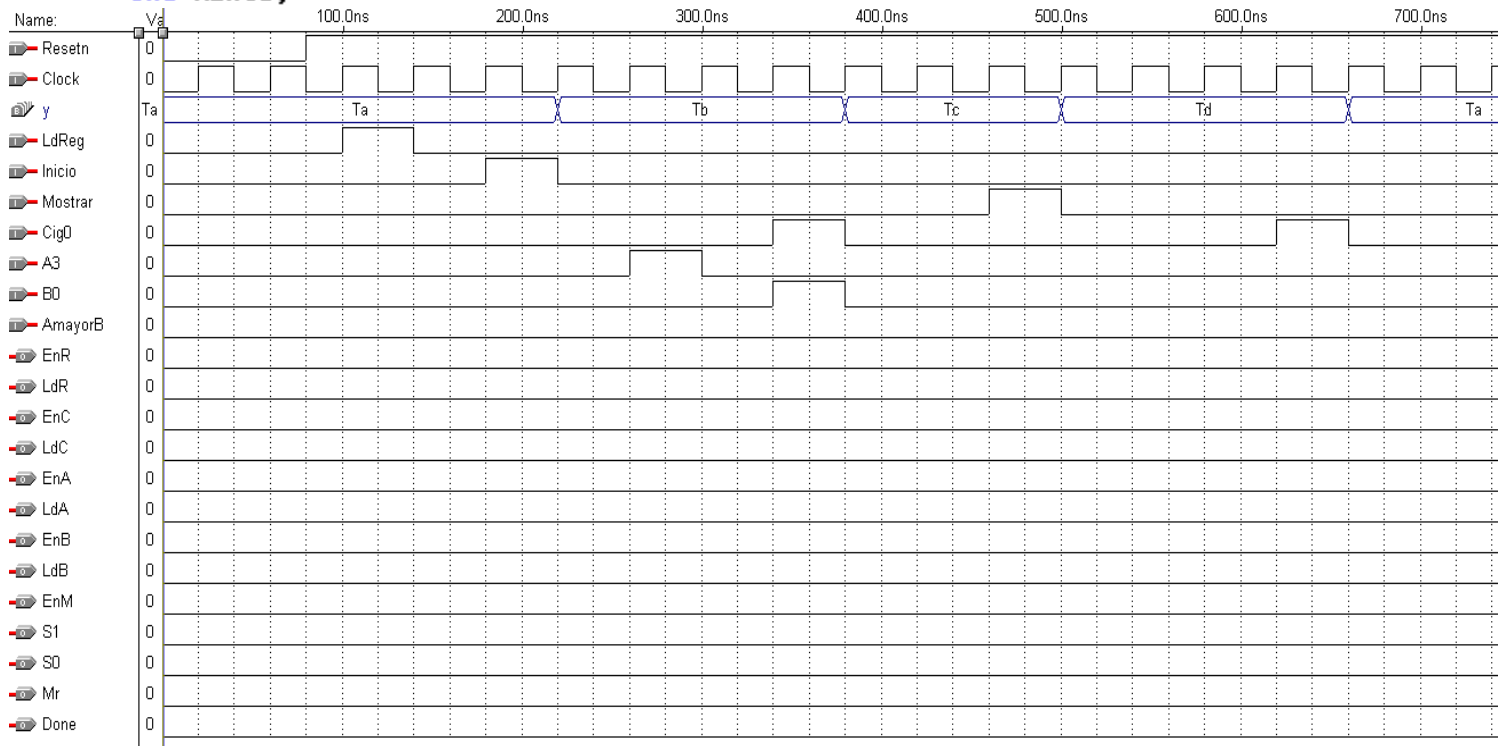
```
Cig0 <='1' when Cnt ="0000" else '0';
```

```
with S select
```

```
    smux <= B(0) when "00",
           not B(0) when "01",
           vcc when "10",
           zero when others;
```

```
BitR <= R(3) when MR ='1' else 'Z';
```

```
end mixta;
```



PROBLEMA # 4

Dada la siguiente descripción en **VHDL** del funcionamiento de un **Sistema Digital**:
Presentar:

1. **Partición Funcional** del Sistema Digital.
2. **Diagrama ASM** del circuito Controlador del Sistema Digital, indicando claramente todas las salidas que deben ser generadas.
3. **Diagramas de Tiempo** del circuito **Controlador** asumiendo las condiciones de entrada dadas. Indique claramente los nombres y la duración de cada estado (**y**).

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

entity problema2_enero2009_new is
    port(Resetn, Clock : in std_logic;
          Start        : in std_logic;
          N1, N2       : in std_logic_vector (3 downto 0);
          Ready, Err, Fin: out std_logic;
          Salida       : out std_logic_vector (3 downto 0));
end problema2_enero2009_new;

architecture mixta of problema2_enero2009_new is

    component registro_sost
        port(Resetn, Clock, En : in std_logic;
              EntPar : in std_logic_vector (3 downto 0);
              Q      : out std_logic_vector (3 downto 0));
    end component;

    type estado is (S1, S2, S3, S4, S5);
    signal y: estado;
    signal En1, En2, Mostrar, Bsel, Cin, Cout : std_logic;
    signal AmayorB, AmenorB : std_logic;
    signal SN1, SN2, MN2 : std_logic_vector (3 downto 0);
    signal SR : std_logic_vector (4 downto 0);
begin

    -- Controlador
    MSS_trans: process(Resetn, Clock)
    begin
        if Resetn = '0' then y <= S1;
        elsif Clock'event and Clock = '1' then
            case y is
                when S1=> if Start = '0' then y <= S1; else y <= S2; end if;
                when S2=> if AmayorB = '1' then y <= S3;
                           elsif AmenorB = '1' then y <= S4; else y <= S1; end if;
                when S3=> y <= S5;
                when S4=> if Cout = '0' then y <= S5; else y <= S1; end if;
                when S5=> if Start = '0' then y <= S1; else y <= S5; end if;
            end case; end if; end process;
```

```

MSS_salidas: process(y, Start, AmenorB, Cout)
begin
Ready <= '0'; Err <= '0'; Fin <= '0'; Mostrar <= '0';
Bsel <= '0'; En1<='0'; En2 <= '0'; Cin <= '0';
    case y is
        when S1=> Ready <= '1';
            if Start = '0' then En1 <= '1'; En2 <= '1'; end if;
        when S2=>
        when S3=> Bsel <= '1'; Cin <= '1'; Mostrar <= '1';
        when S4=> if Cout = '1' then Err <= '1';
                    else Mostrar <= '1'; end if;
        when S5=> Fin <= '1';
    end case; end process;

-- Procesador de datos

reg1: registro_sost port map(Resetn, Clock, En1, N1, SN1);
reg2: registro_sost port map(Resetn, Clock, En2, N2, SN2);
reg3: registro_sost port map(Resetn, Clock, Mostrar, SR(3 downto 0), Salida);

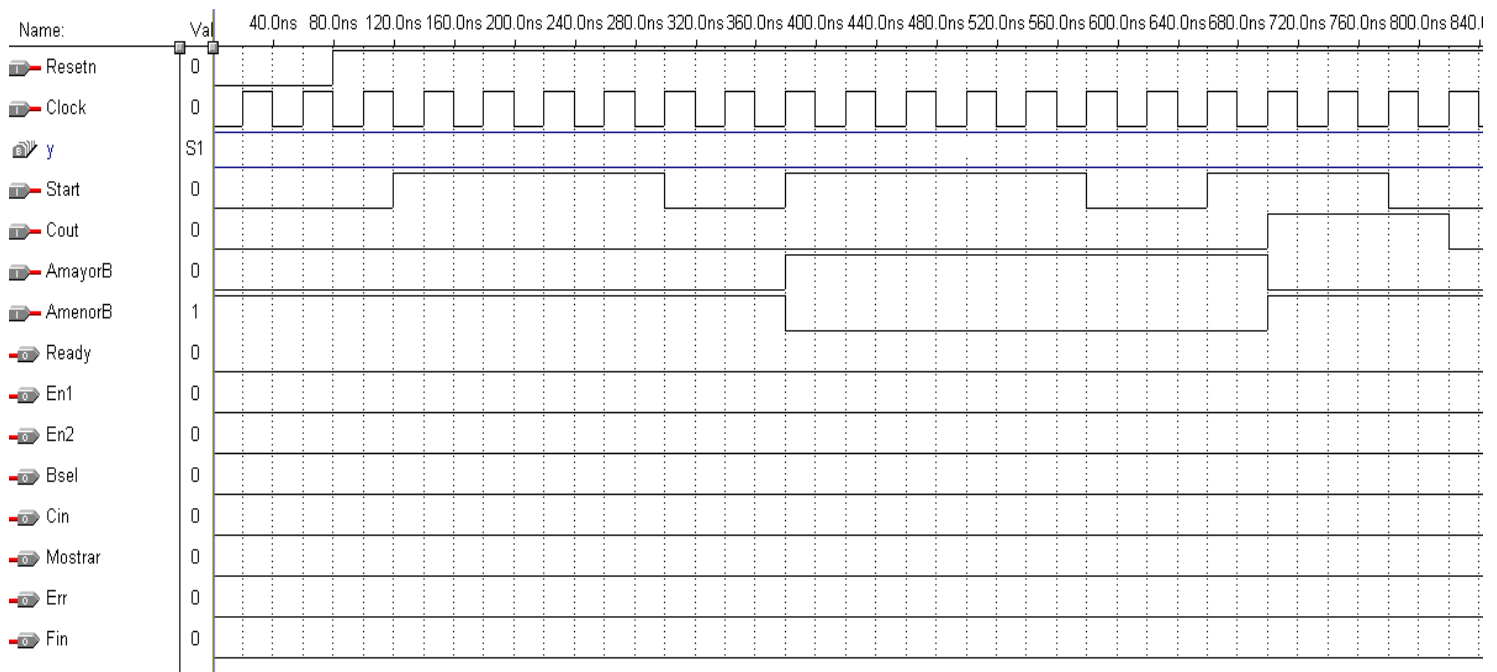
AmayorB <= '1' when SN1 > SN2 else '0';
AmenorB <= '1' when SN1 < SN2 else '0';

SR <= SN1 + MN2 + Cin;
Cout <= SR(4);

MN2 <= SN2 when Bsel = '0' else not SN2;

end mixta;

```



PROBLEMA # 5

Realizar el diseño de un **SISTEMA CALCULO DE DETERMINANTE DE UNA MATRIZ**. El sistema recibe una matriz cuadrada A(3x3) y luego procede a calcular el determinante de dicha matriz.

Ejemplo:

A=[1 5 8; 7 1 3; 0 1 2]
B=det(A)

```
>> A=[1 5 8; 7 1 3; 0 1 2]

A =

     1     5     8
     7     1     3
     0     1     2

>> B=det(A)

B = -15
```

SEÑALES

MatrizA.- esta señal de 8 bits permite el ingreso de cada uno de los números de las matrices cuadrada A(i, i)=MatrizA, de preferencia el número ingresado debe estar entre 0-9.

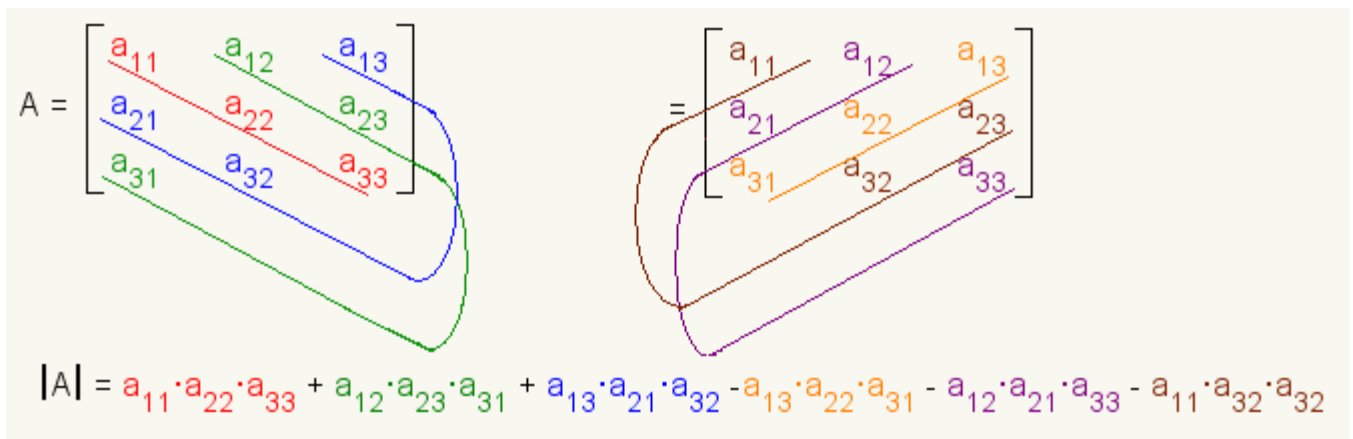
IngresoA.- Esta señal indica cada vez que hay un número a ser ingresado en la matriz A.

Start.- Da inicio al proceso de multiplicación de las matrices luego de ser ingresadas.

LedOK.- Led indicador de salida que da a conocer que la multiplicación fue realizada con éxito.

DetAns.- Señal de salida de 8 bits que muestra el resultado del determinante.

Teclado: permite el ingreso del número entre 0-9 y además permite el ingreso de la; (0X3B)


$$|A| = a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{13} \cdot a_{21} \cdot a_{32} - a_{13} \cdot a_{22} \cdot a_{31} - a_{12} \cdot a_{21} \cdot a_{33} - a_{11} \cdot a_{32} \cdot a_{32}$$

NOTA: se considerará puntos extras si las matrices pueden ser de las siguientes dimensiones: A(nxn).

Se pide presentar:

- Partición Funcional
- Diagrama ASM
- Código VHDL del Controlador

PROBLEMA # 6

Un circuito **Data Logger** es un circuito que almacena información que luego puede ser descargada a una computadora para analizar los datos.

Una empresa que cultiva **ROSAS** para **San Valentín** pide a los estudiantes de Sistemas Digitales 2 diseñar un **Sistema para administrar su Data Logger**.

El Data Logger tiene una memoria de **4Gx32** y se le conecta un sensor de temperatura y uno de humedad.

El sistema debe leer la temperatura y la humedad en intervalos de 10 minutos y almacenar en la memoria de la siguiente forma:

En la primera casilla de memoria, el día y la hora (palabra de 32 bits), siguiente casilla la temperatura y en la tercera el valor de la humedad relativa del suelo (palabras de 32 bits cada una), hasta que se llene la memoria o se presione la tecla **INFORME**.

El sistema estará grabando por semanas incluso, hasta que se presione la tecla **INFORME**. Una vez hecho esto se visualizará en displays de siete segmentos los eventos en que la temperatura estuvo más alta en cada día, es decir mostrará la terna (día/hora, temperatura, humedad relativa) en secuencia pero solo de un valor por día (aquel cuya temperatura sea la más alta del día). Para pasar de un valor a otro de la terna debe utilizarse una tecla **MOSTAR**, asimismo de una terna a otra. Si la memoria se llena antes de que la tecla **INFORME** sea presionada el circuito debe enviar una señal de **ALARMA** y dejar de grabar. Una vez mostrado todas las ternas el circuito debe enviar una señal **SIN DATOS**, la que también se podría encender si se presiona la tecla **INFORME** que se haya guardado el primer dato.

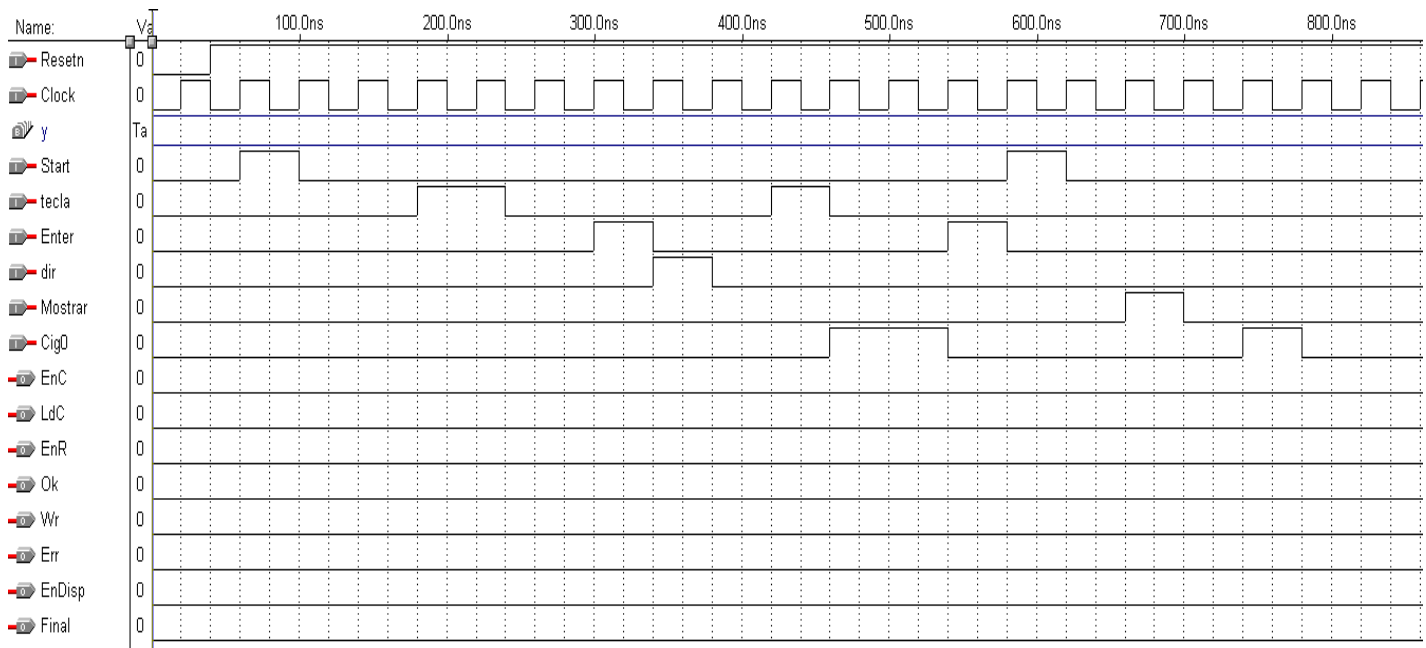
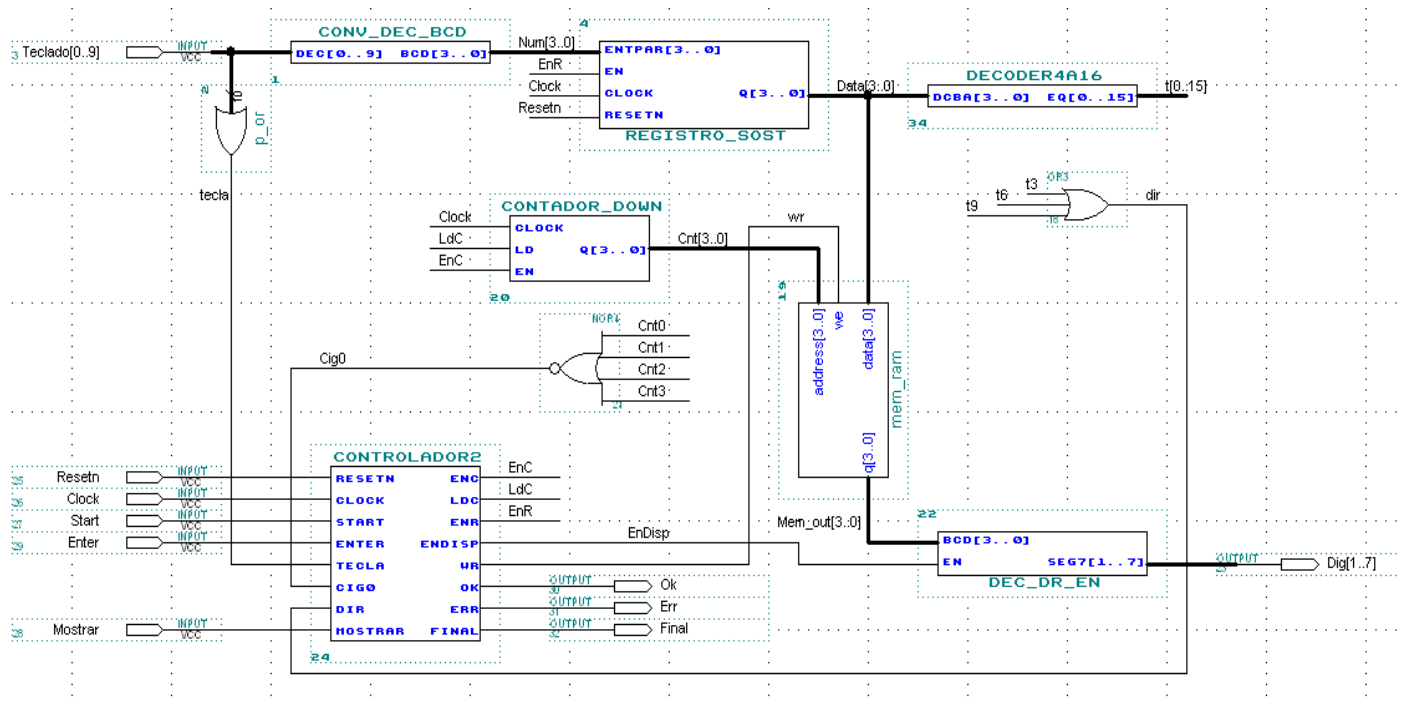
Presente:

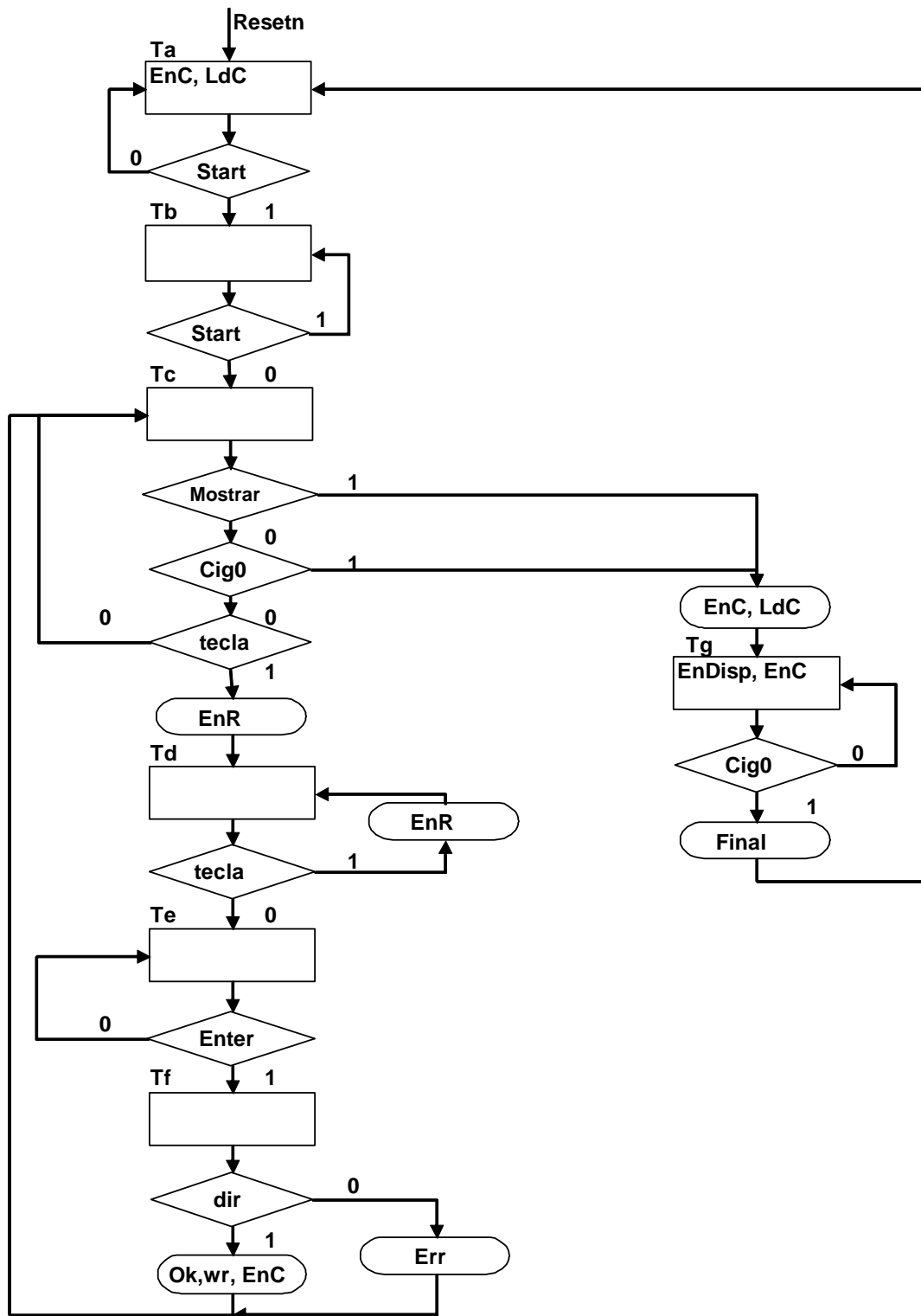
1. **Diagrama general de entradas/salidas (2/30)**
2. **Diagrama de bloques del sistema digital (8/30)**
3. **Partición Funcional (10/30)**
4. **Diagrama ASM** del circuito **Controlador** debidamente documentada. (indicar todos las entradas y salidas) **(10/30)**

Nota. Asuma que la implementación será hecha usando tecnología FPGA y lenguaje VHDL. Dimensionar adecuadamente cada componente.

PROBLEMA # 7

Para el siguiente Sistema Digital, se muestran la **Partición Funcional** y el **Diagrama ASM** del circuito Controlador.





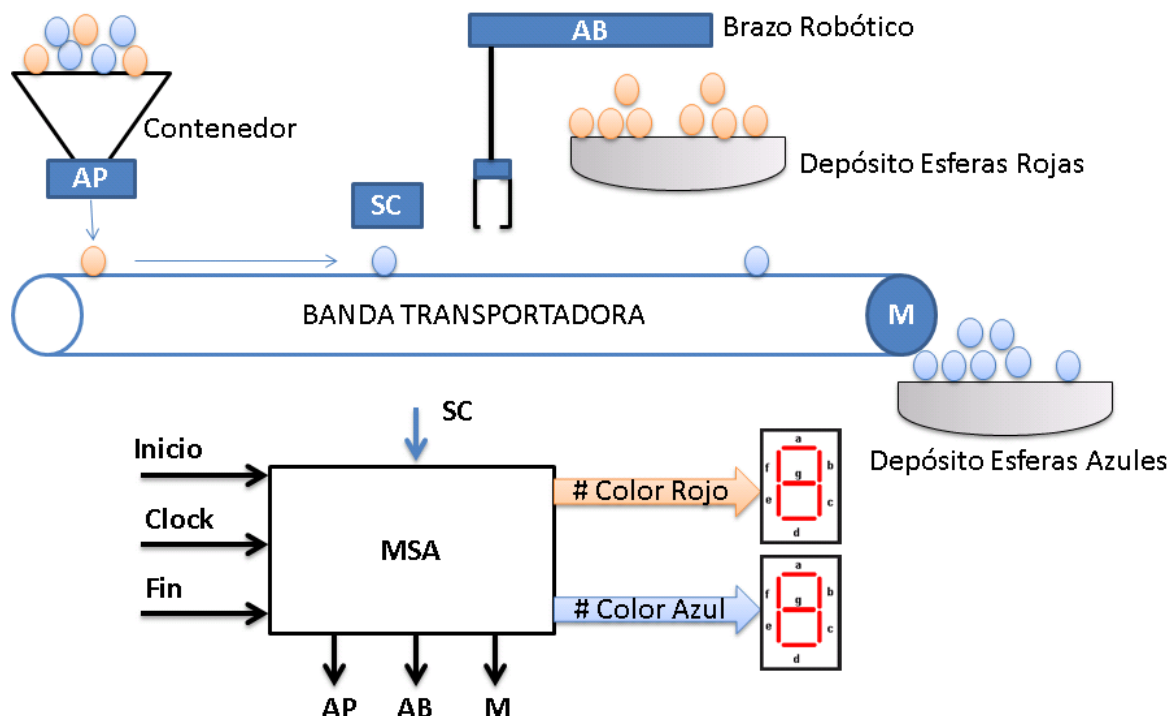
Presente:

1. Descripción del Sistema en un solo programa en **VHDL** usando las declaraciones ***process – case*** para describir las Transiciones de Estados y las Salidas del **Controlador**, y la **descripción estructural** para el **Procesador de Datos**.
Asuma que dispone de archivos **.vhd** en la misma carpeta de Trabajo para todos los sub-circuitos que forman parte del Sistema Digital **excepto para el Controlador y para las puertas lógicas**.
Así mismo suponga que el orden de las entradas en la declaración ***port*** de los sub-circuitos es similar (de izquierda a derecha y de arriba abajo) al del Diagrama Esquemático presentado.
2. Grafique los **Diagramas de Tiempo** del circuito **Controlador** asumiendo las condiciones de entrada dadas. Indique claramente los nombres de los estados (**y**) que corresponden a cada periodo de reloj.

PROBLEMA # 8

Se tienen un proceso de clasificación de esferas por color, esferas que inicialmente se encuentran almacenadas en un **contenedor** que en la salida de las esferas tiene acoplado un **Actuador de Paso (AP)** que permite pasar una sola esfera por vez. Luego cada esfera cae a la **banda transportadora** hasta llegar al **sensor de color (SC)** y si es la esfera roja un **actuador de brazo robótico (AB)** que levanta la lleva al **depósito de esferas rojas**, caso contrario avanza por la banda transportadora hasta llegar al **depósito de esferas azules**.

Se quiere realizar un sistema Digital que luego de presionar y soltar el botón **INICIO**, accione el **Motor (M)** de la **banda transportadora**, luego activará el **actuador de paso** para permitir que caiga una esfera por vez a la banda (la válvula deberá ser activada por 5 segundos por su respuesta mecánica). Una vez que una esfera está en la banda transportadora ésta pasará por el **sensor de color** el cual determinará el destino final de la esfera. Si la esfera es azul, la banda sigue trabajando hasta llevar a ésta al contenedor de esferas azules que queda al final de la banda transportadora. Si el color de la esfera es rojo, la banda transportadora deberá detenerse por 15 segundos que es el tiempo estimado que el brazo robótico demora en **bajar - tomar a esfera - llevar la esfera - dejar la esfera - regresar a su posición original**. El **actuador de paso** se activará automáticamente si el sensor de color detecta que es una esfera azul, pero si es una esfera roja el actuador de paso se activará luego de que el brazo robótico llegue a su posición inicial. Adicionalmente el sistema permitirá **visualizar en dos Display** la cantidad de esferas rojas y azules en tiempo real. El sistema regresará a su estado inicial cuando el operador presiona el botón de **FIN** o cuando uno de los dos depósitos alcance la cantidad de nueve esferas.



Presentar:

1. **Diagrama de bloques** del Sistema Digital.
2. **Partición Funcional** del Sistema Digital.

3. Diagrama ASM del controlador.

PROBLEMA # 9

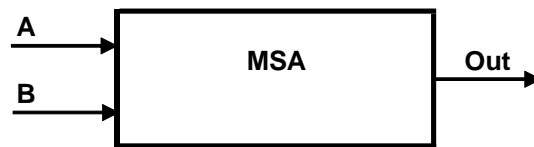
Diseñar en modo Fundamental una **MSA** (Maquina Secuencial Asincrónica) que funciona de la siguiente manera.

La **MSA** tiene dos entradas **A** y **B** y una salida **Out**.

Inicialmente las entradas **A** y **B** y la salida **Out** son igual **0**.

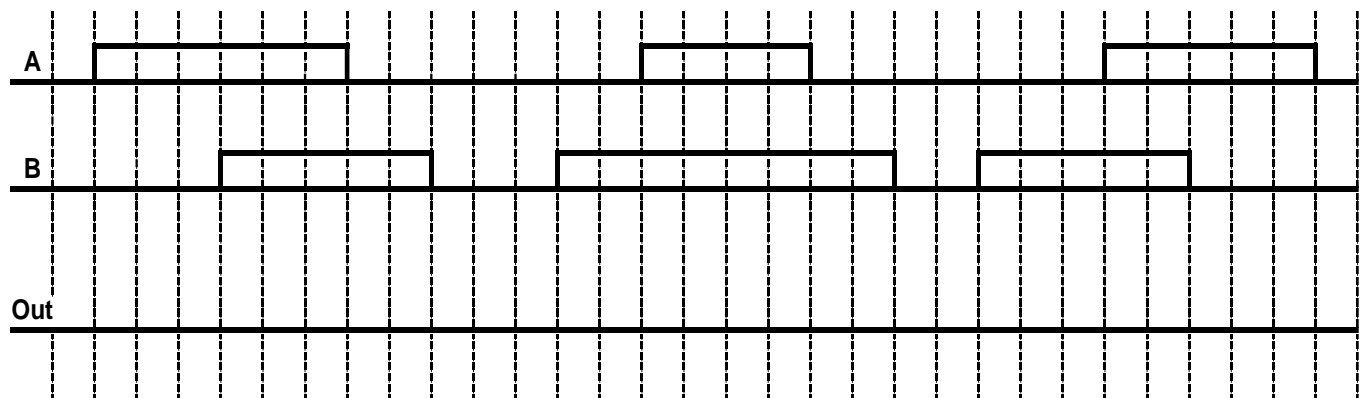
Cuando una de las entradas (**A** o **B**) se hace igual **1**, la salida **Out** también se hace igual a **1**. Si, mientras una de las entradas se mantiene igual a **1**, la otra se hace igual a **1** también, la salida **Out** se mantiene igual a **1**.

Si, luego, una de las entradas se hace igual a **0**, la salida **Out** se hace igual a **0** también y se mantiene igual a **0** aun cuando esta entrada se hace igual a **1** de nuevo, hasta que **MSA** regresa al estado inicial.



Presente:

1. Diagrama de Estados Primitivo (Formato: **A B / Out**). Mapa de Estados Primitivo. Tabla de Implicantes.
2. Diagrama de Estados Reducido. Mapa de asignación de Código de Estados.
3. Mapa de Excitación. Mapas para las salidas **Y₁** y **Y₀** y para la salida **Out**.
4. Diagrama de tiempo para la salida **Out** asumiendo valores de las entradas **A** y **B** dados.
Indica claramente los periodos de tiempo correspondiente a cada estado de su Diagrama de Estados Reducido.
5. Indique si su circuito corre riesgo de tener los **Hazard Estáticos** o no. Como se puede evitar.



PROBLEMA # 10

Diseñar en modo Fundamental una **MSA** (Maquina Secuencial Asincrónica) que tiene dos entradas **S** y **D** y una salida **LD**.

Siempre que la entrada **S** sea **0**, la salida **LD** es **0**.

Cuando la entrada **S** se hace **1**, la salida **LD** se hace igual a **D**.

La salida **LD** permanece sin cambios hasta que la entrada **S** se hace **0** de nuevo.



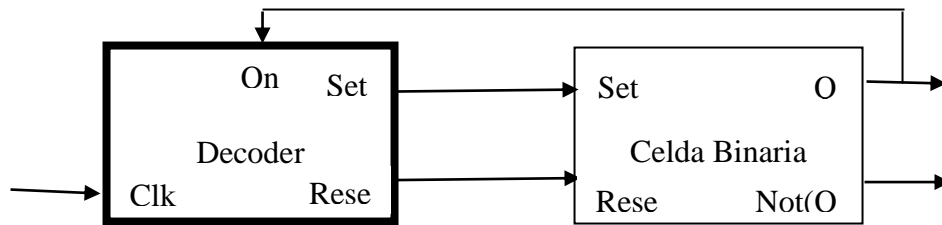
Presentar:

1. Diagrama de Estados Primitivo (Formato: **S D / LD**). Mapa de Estados Primitivo
2. Tabla de Implicantes. Diagrama de Estados Reducido
3. Mapa de asignación de Código de Estados. Mapa de Excitación. Mapas para las salidas **Y₁** y **Y₀** y para la salida **LD**
4. Indique si su circuito corre riesgo de tener los **Hazard Estáticos** o no. Como se puede evitar.

PROBLEMA # 11

Diseñar un Maquina Secuencial Asíncrona (MSA) que hace el trabajo de un decodificador que en conjunto con una celda Binaria trabaja como un **Flip-Flop Inv**.

Flip-Flop Inv				Celda binaria				
Clk	Qn	Qn+1		Set	Reset	Qn	Qn+1	
↑	0	1	Inv	0	0	0	0	Hold
↑	1	0	Inv	0	0	1	1	Hold
↑	0	0	Hold	0	1	0	0	Rst
↑	1	1	Hold	0	1	1	0	Rst
				1	0	0	1	Set
				1	0	1	1	Set
				1	1	0	⊘	⊘
				1	1	1	⊘	⊘



Presentar:

- Diagrama de estados primitivo.
- Mapa de estados primitivo.
- Diagrama de equivalencia máxima.
- Diagrama de estados reducido.
- Asignación de código de estados.
- Mapa de excitación.
- Decodificador de estado siguiente y decodificador de salida.
- Implementar el circuito completo.

PROBLEMA # 12

Realizar el diseño de un **SISTEMA EFICIENTE DE DETECCIÓN DE DIGITOS**. El sistema recibe una trama de dígitos entre 0 -9 de forma aleatoria y permite detectar el número que más veces se repite en toda la trama.

SEÑALES

Keypad.- son 10 teclas que permiten el ingreso de los dígitos entre 0-9.

Ingreso Datos.- Esta señal le indica al sistema que el usuario desea ingresar la secuencia y se debe validar que el ingreso de los dígitos no sea mayor a 256.

Fin Ingreso Datos.- si el número de dígitos ingresados no es mayor a 256 el usuario puede dejar de ingresar los números en cualquier momento.

LedSec.- Led indicador que el sistema está solicitando el ingreso de la trama de dígitos.

Start.- Luego de ingresar correctamente los datos el sistema valida la tecla start para empezar a trabajar.

Ejemplo.

Secuencia (30 dígitos entre 0 - 9)	24563129563245298061230689725
Número que más se repite	2
Repeticiones del Número	6

Display Patron.- indicará el mejor patrón detectado.

Display Repeticiones.- Indica la cantidad de veces que se repite la mejor trama detectada.

NOTA: el pseudocódigo para detectar las veces que se repite cada dígito común es.

```
Inicializar RAMCnt=0
For i=0 to TamSecuencia
  CNT=0
  For j=0 to TamSecuencia
    R1=RAM(i)
    R2=RAM(j)
    If Ri=Rj
      Inc(CNT)
    Endif
  endFor
  if RAMCnt(RAM(i))<CNT
    RAM(RAM(i))=CNT
  EndFor
```

Se pide presentar:

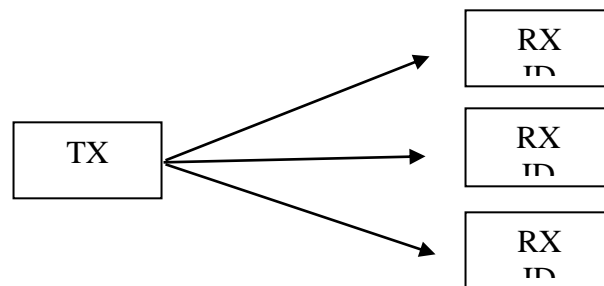
- Partición Funcional
- Diagrama ASM
- Código VHDL del Controlador

PROBLEMA # 13

Realizar el diseño de un **SISTEMA DE COMUNICACIÓN CON VALIDACIÓN CHECKSUM**. El sistema está conformado por un transmisor que envía datos bajo un protocolo establecido hacia módulos receptores que validarán la trama recibida.

Protocolo de comunicación:

Byte de Inicio	Byte ID de Equipo	Byte de Data	Byte Checksum
0X24	0X01 ó 0X0A ó XA1	0X00 – 0XFF	XOR



SEÑALES TX

PaqueteOut.- esta señal de 8 bits envía cada uno de los paquetes de la trama byte a byte, es decir envía de forma ordenada primero Byte de Inicio, Byte de ID, Byte de data y finalmente Byte de CheckSum.

SincTx.- Esta señal sirve de sincronía para indicar al equipo receptor que existe un dato listo para ser leído.

StartTx.- Señal para empezar a enviar los datos.

StopTx.- Señal para detener el envío de datos.

IDRx.- El Byte de ID será leído constantemente antes de enviar los datos, de esa forma puede enviar la trama a diferentes equipos.

SEÑALES RX

PaqueteIn.- esta señal de 8 bits recibe cada uno de los paquetes de la trama byte a byte, es decir recibe de forma ordenada primero Byte de Inicio, Byte de ID, Byte de data y finalmente Byte de CheckSum.

SincRx.- Esta señal sirve de sincronía para indicar al equipo receptor que existe un dato listo para ser leído.

StartRx.- Señal que empezar a recibir los datos.

StopTx.- Señal para detener la recepción de datos.

IDRx.- El byte de ID es cargado inicialmente antes de presionar Start de tal forma que el ID del equipo de recepción se compara con el ID recibido.

Se pide presentar:

- Partición Funcional
- Diagrama ASM
- Código VHDL del Controlador

PROBLEMA # 14

Realizar el diseño de un sistema de alarma comunitaria para una ciudad, el cual notifica al UPC más cercana lo siguiente: el número cuadra, número de casa y la emergencia que tienen los habitantes de esa casa.

El sistema cuenta con un equipo transmisor en cada casa (máximo 99 casas por cuadra), la ciudad en total tiene máximo 90 cuadras, cada alarma de casa debe ser capaz de enviar al UPC una trama de comunicación de 5 bytes de datos, en el siguiente formato:

1: Byte de Inicio	2: Byte ID Cuadra	3: Byte ID Casa	4: Emergencia	5: Byte de Fin
0xF0	0x00 – 0x5A	0x00 – 0x63	0x01 – 0x03	0xF1

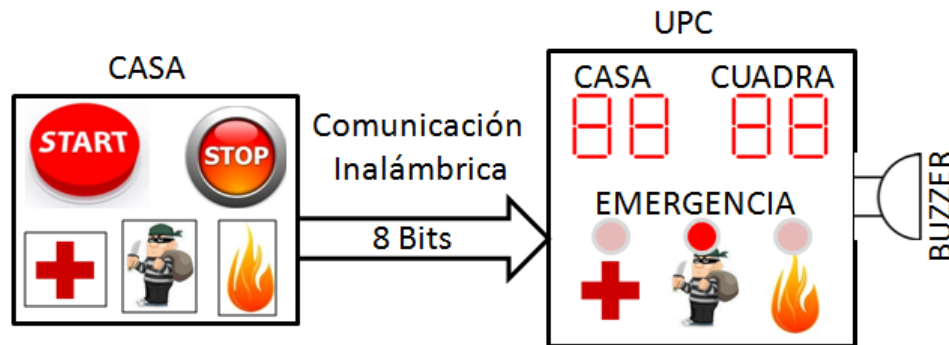
Los tipos de emergencia del sistema de alarma son:

- 1: Emergencia Médica
- 2: Robo
- 3: Incendio

El Sistema de monitoreo que está en la UPC tiene 4 displays de 7 segmentos, 3 leds para indicar la emergencia y una salida digital para un Buzzer como indicador acústico. Este sistema deberá validar los Bytes de inicio-Fin para luego mostrar la información a los policías permanentemente durante 30 segundos. En ese tiempo el Buzzer deberá sonar a una

Frecuencia de 5kHz (Señal Cuadrada con un ciclo de trabajo del 50% generada por el sistema del UPC).

Ambos sistemas por separado tienen un pulsador de **START** y un botón de **RESET** para reiniciar cada sistema.



NOTA: Asuma que la comunicación es inalámbrica con modulación QPSK y HalfDuplex, de Punto - Multipunto. Imagine que siempre hay línea de vista.

Se pide presentar:

- Partición Funcional
- Diagrama ASM
- Código VHDL del Controlador

PROBLEMA # 15

Realizar el diseño de un sistema Digital de **CONTROL DE PERSONAL**. El sistema controla el ingreso y salida de forma automática del personal de trabajo (máximo 9). Cuando un trabajador entra o sale debe:

- Con 4 botones indicará: **entrada trabajo, pausa inicio, pausa fin y salida de trabajo.**
- Con un teclado decimal ingresará su número de **ID de trabajador** (1-9)
- Usando el teclado decimal se autenticará con un **password** único por persona de 4 dígitos.

Dos botones para indicar que está saliendo o ingresando, un pulsador que le sirve para iniciar el sistema (**START**), un botón de **RESET** para inicializarlo todo.

Al finalizar la jornada de trabajo (máximo 12 horas) el trabajador se autenticará y el sistema debe indicar en dos displays de 7 segmentos las horas de trabajo de cada persona (redondear hacia abajo) y además el ID del trabajador en otro display, información que el departamento de Talento Humano usará para hacer el pago por hora.

IMPORTANTE: el sistema debe permitir **REGISTRAR** los empleados inicialmente y cada uno con su clave de registro.



NOTA: Asuma que el registro se lo hace una sola vez. Los trabajadores podrán pausar su trabajo para comer.

Se pide presentar:

- Partición Funcional
- Diagrama ASM
- Código VHDL del Controlador

PROBLEMA # 16

Diseñar un Sistema Digital que resuelve diagramas de estados que dada una RAM de 8 direcciones donde se escriben operadores identificados por un número ID como se indica en la siguiente tabla:

ID Operador	Operador
0x01	and(y2,y1,A)
0x02	nand(y2,y1,A)
0x03	and(nand(y2,y1),A)
0x04	and(y2,nand(y1,A))
0x00	Termino sin Operador

El sistema deberá evaluar la RAM de 8 direcciones, donde cada 2 direcciones se representan dos bits del decodificador de estado siguiente y el decodificador de salida, como si indica a continuación:

Decoder	Dir RAM	Dato Ram
Y1	0x00	0x01
	0x01	0x03
Y2	0x02	0x02
	0x03	0x04
Out	0x04	0x03
	0x05	0x01
-	0x06	0x00
-	0x07	0x00

El sistema deberá retornar una memoria RAM que represente la tabla de estados presentes y siguientes tal como se indica en la siguiente tabla:

Dir RAM	Dato RAM (y2,y1,A,Y1,Y2,Out,x,x)
0x00	%000 010 --
0x01	%001 110 --
0x02	%010 010 --
0x03	%011 110 --
0x04	%100 011 --
0x05	%101 111 --
0x06	%110 011 --
0x07	%111 101 --

NOTA: la memoria RAM de salida deberá ser inicialmente puesta en cero.

Se pide:

- Hacer la partición funcional del sistema completo: Mss, Ram, Msi, etc.
- Diagrama ASM del controlador.
- Código VHDL de la MSS.

PROBLEMA # 17

Diseñar un Sistema Digital que simplifica ecuaciones booleanas, para lo cual deberá recibir una RAM de 8 direcciones donde se escriben operadores identificados por un número ID como se indica en la siguiente tabla:

ID Operador	Operador
0x00	Termino sin Operador

0x01	and(not(y2),not(y1),y0)
0x02	and(not(y2),y1,not(y0))
0x03	Termino sin Operador
0x04	Termino sin Operador
0x05	and(y2,not(y1),y0)
0x06	and(y2,y1,not(y0))
0x07	and(y2,y1,y0)

El sistema deberá evaluar la RAM de 8 direcciones, donde todas cada dirección tendrá el valor de una de las operaciones (0x01 – 0x07) las que tengan el valor (0x00) no deberán ser evaluadas, además NO deberán repetirse operaciones en la RAM, como si indica a continuación:

Variable	Dir RAM	Dato RAM (término de la ecuación)
Salida	0x00	0x01
	0x01	0x05
	0x02	0x06
	0x03	0x02
	0x04	0x07
	0x05	0x00
	0x06	0x00
	0x07	0x00

El sistema deberá retornar una memoria RAM que represente los terminos de la ecuación simplificada:

ID Operador	Operador
0x00	Término sin Operador
0x01	and(y1,not(y0))
0x02	and(y2,y1)
0x03	and(y2,y0)
0x04	and(not(y1),y0)
0x05	and(not(y2),y1,not(y0))
0x06	and(not(y2),not(y1),y0)
0x07	and(y2,y1,y0)
0x08	and(y2,y1,not(y0))
0x09	and(y2,not(y1,y0))

En las direcciones de memoria donde no hay terminos se los llena con 0x00.

Dir RAM	Dato RAM (y2,y1,y0,Salida,x,x,x,x)
---------	------------------------------------

0x00	0x01
0x01	0x02
0x02	0x03
0x03	0x04
0x04	0x00
0x05	0x00
0x06	0x00
0x07	0x00

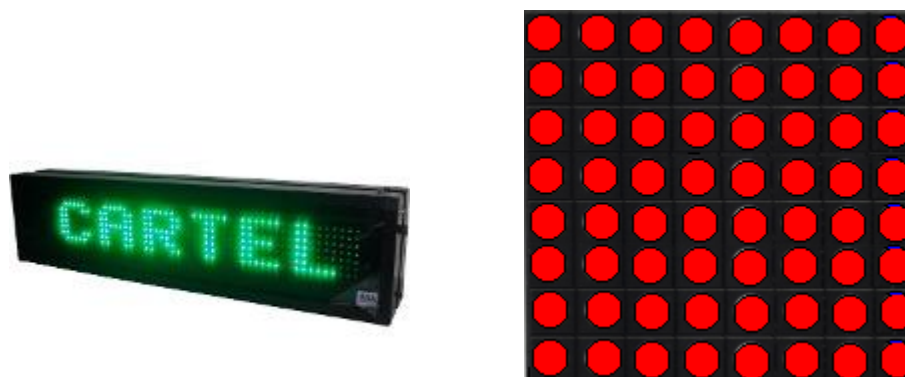
NOTA: la memoria RAM de salida deberá ser inicialmente puesta en cero.

Se pide:

- Hacer la partición funcional del sistema completo: Mss, Ram, Msi, etc.
- Diagrama ASM del controlador.
- Código VHDL de la MSS.

Ejercicio 18:

Realizar el diseño de un Sistema Digital MARQUESINA CON MATRIZ DE LEDS. El sistema tiene como interfaz de salida cuatro matrices de leds de 8x8 que permite visualizar letras que estarán pregrabadas en una memoria RAM de 8 direcciones, las letras deberán visualizarse mientras se desplazan de izquierda a derecha. El ingreso de las letras a la memoria RAM se lo hará por medio de un teclado alfabético, para lo cual se deberá presionar la tecla Grabar letra por letra y luego de presionar la tecla Start. Si luego de grabar el mensaje el usuario presiona la tecla Mostar, el sistema empezará a hacer aparecer las letras en la marquesina, letra a letras empezará a aparecer hasta que se vea unas cuatro letras en la marquesina a la vez. Una vez que se pierde la última letra del mensaje de 8 caracteres, el sistema mostrará nuevamente el mensaje, todo esto hasta que el usuario presione la tecla salir.



SEÑALES

Tecla Start.- el sistema permitirá grabar mensaje o mostrar mensaje siempre y cuando se presione Start primero.

Tecla Reset: Esta tecla solo deberá resetear a la MSS, es decir la MSS será quien resetee los bloques MSI que sean necesarios.

Tecla Grabar. - Permitirá ingresar letra a letra el mensaje a ser mostrado, el mensaje tiene máximo 8 letras ya que serán almacenadas en una ram de 8bits x 8 direcciones.

Tecla Mostrar. - Permitirá al sistema empezar a mostrar el mensaje mientras se desplaza.

Tecla Stop. - Esta tecla permite detener la marquesina y si luego se presiona Grabar, se podrá ingresar un nuevo mensaje, caso contrario si se presiona mostrar se vuelve a mostrar el mensaje almacenado anteriormente.

Matriz de leds. - matriz de 8x8 leds que permiten formar cualquier letra, pero para encender los puntos para las letras se deberán hacer multiplexaciones.

Teclo alfabético. - Teclado con la letra del alfabeto para ingresar cualquier mensaje.

NOTA: la velocidad de desplazamiento del mensaje es 2puntos / segundo.

Se pide:

- a) Hacer la partición funcional del sistema completo: Mss, Ram, Msi, etc.
- b) Diagrama ASM del controlador.