CAPÍTULO 4
DISEÑO DE SISTEMAS DIGITALES

EJEMPLOS DE DISEÑO DE SISTEMAS DIGITALES

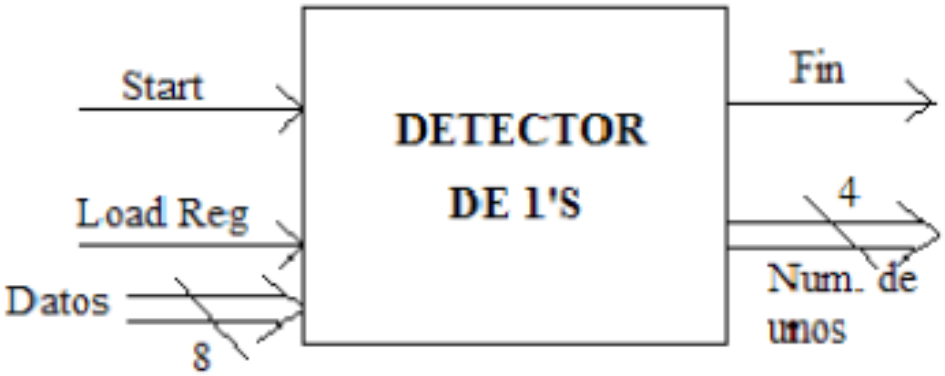
DISEÑO DE UN SISTEMA DIGITAL DETECTOR DE 1's.

Datos:

Se desea detectar cuantos unos (1) hay en la entrada **Datos**, que está conformada por ***n*** bits. Si la señal Load_Reg es verdadera, se asume que el dato es válido y debe ser cargado en el circuito.

La detección debe iniciar si la señal **START** es verdadera.

En la salida **Num_unos** se debe indicar en binario, la cantidad de unos presentes en el dato cargado. La salida **Fin** será verdadera cuando termine la detección.

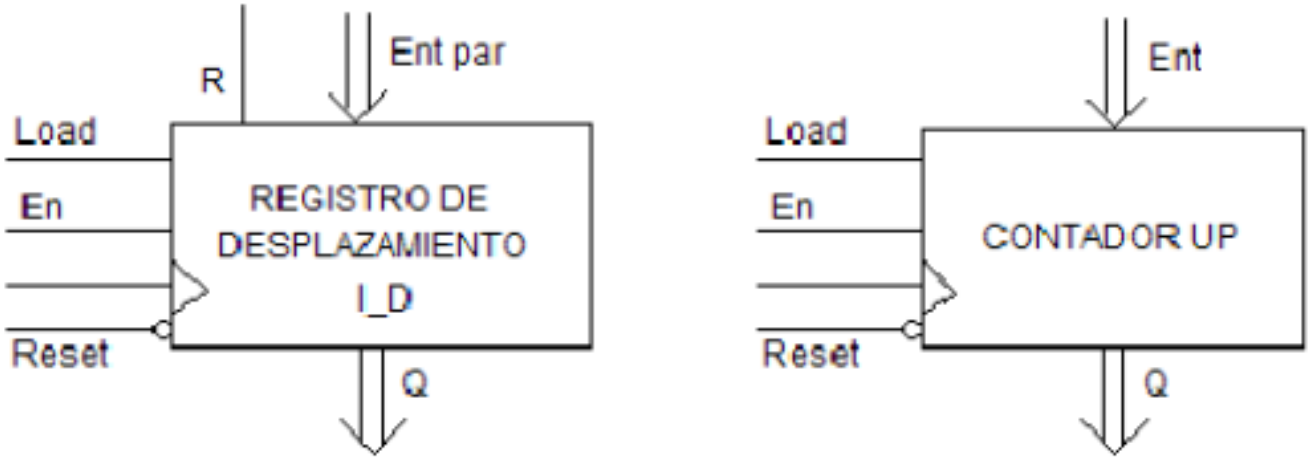


Tenemos que diseñar simultáneamente la **Partición Funcional** y el **Diagrama ASM** del circuito **Controlador**.

La solución propuesta consiste en un registro de desplazamiento, donde se cargue el dato original, y con cada pulso de reloj, en la posición menos significativa se muestre cada uno de los bits. Si el bit es uno (1) se manda habilita a un contador binario, mientras que si el bit es cero (0) no se cuenta.

Basándonos en las reglas básicas, podemos darnos cuenta que el registro de desplazamiento y el contador deben ser subcircuitos del **Procesador de Datos**.

La **Partición Funcional** en este caso es sencilla. Necesitamos un registro de desplazamiento de ***n*** bits (por ejemplo, de 8 bits) y un contador de **log2*n*** bits (puede ser un contador de 4 bits).



0110000101110011011000010110111001111010011000011

El registro debe ser de desplazamiento de izquierda a derecha y debe tener capacidad de carga en paralelo para los datos de entrada de **n** bits.

El contador también debe tener capacidad de carga paralela para tener **0's** (encerar el contador) en el estado inicial. Normalmentem no se utiliza la señal **Resetn** para limpiar directamente a contadores en el estado inicial.

En la práctica, la señal de Resetn es usada en los Sistemas Digitales solo para dos propósitos:

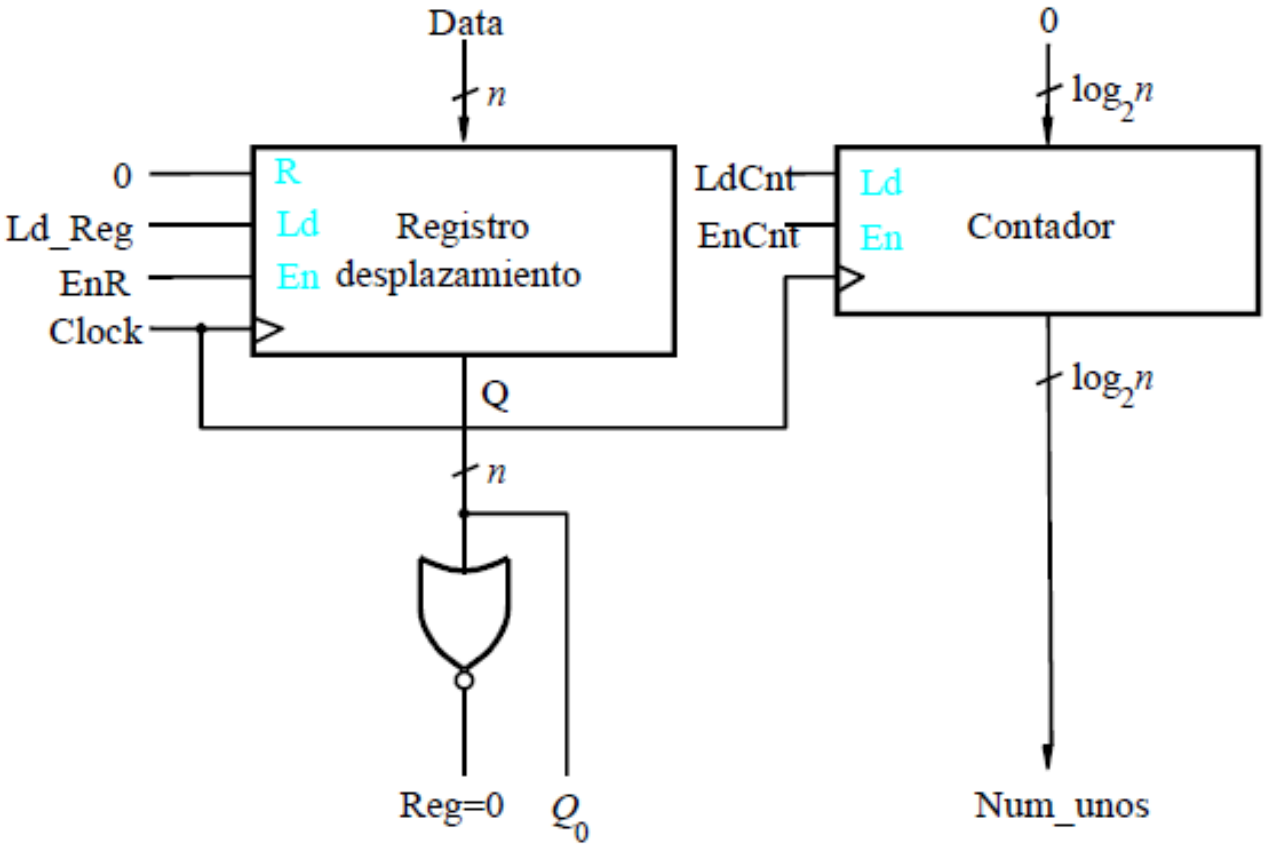
- Para colocar el circuito en el estado inicial luego de prender la fuente de poder.
- Para colocar el circuito en el estado inicial en caso que funcione incorrectamente.

De esta manera decimos que de forma general, los subcomponentes para El procesador de datos, tendrán dos tipos de señales:

En : Enable (Habilitación del dispositivo) y Ld = Load (carga de datos)

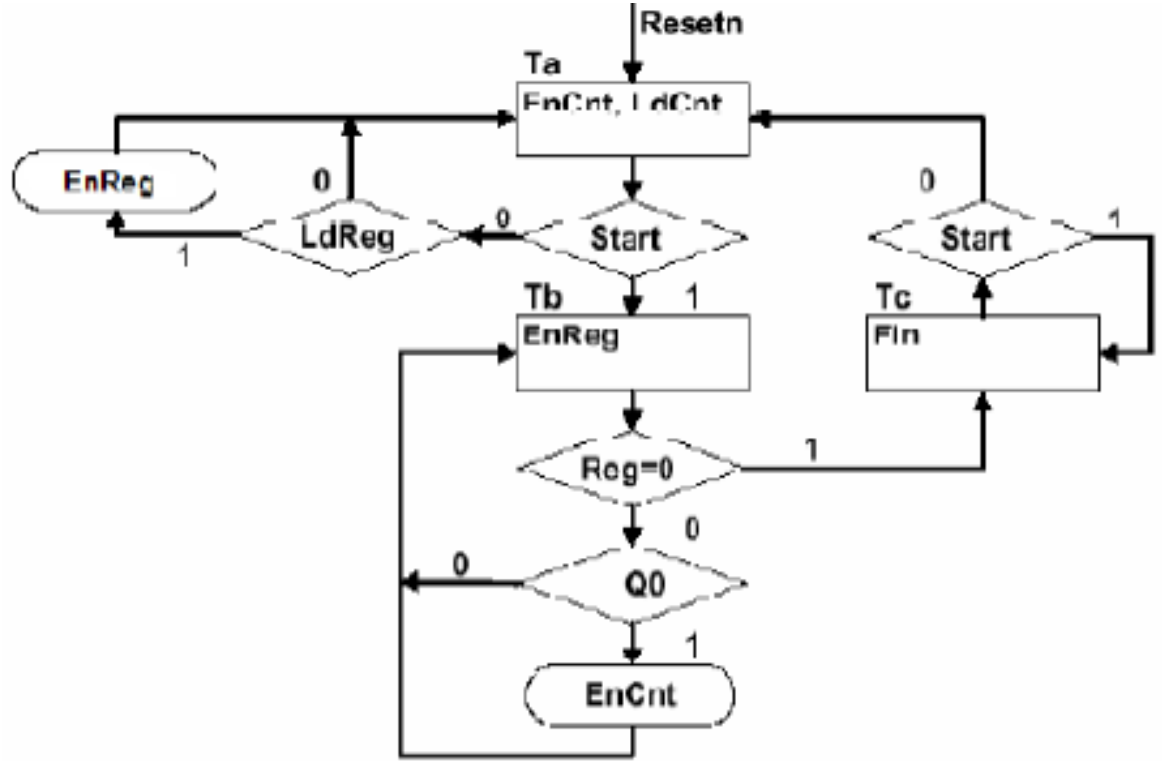
Si En=0 : El dispositivo mantiene el último valor en la salida
Si En=1 y Clock=flanco :
Si Ld=1 : Se carga el dato de la entrada paralela
Si Ld=0 : El dispositivo opera según su diseño (desplaza, cuenta, etc).

Partición Funcional



Una vez elegidos los subcircuitos del **Procesador de Datos**, en el **Diagrama ASM** se debe indicar exactamente que señales va a recibir y debe generar el circuito **Controlador**.

Diagrama ASM del Controlador



En estado **Ta** el **Controlador** debe generar las señales incondicionales de habilitación **EnCnt** y de carga **LdCnt** para encerar el contador_up.

Al finalizar el estado **Ta** el **Controlador** pregunta por la entrada externa **Star**t. Si **Start** no ha sido activado pregunta por la otra entrada externa **LdReg** que es la señal de carga del registro_i_d. Si **LdReg** ha sido activada el **Controlador** genera la salida condicional de habilitación **EnReg**.

En el estado **Tb** el **Controlador** genera la salida incondicional **EnReg** que permitirá el desplazamiento a la derecha del registro_i_d.

Puesto que para manejar el registro_i_d y el contador_up, se emplea la misma señal de ***Clock*** que utiliza el **Controlador,** el desplazamiento del registro_i_d se realizará al finalizar el estado **Tb** con el siguiente flanco de subida del **Clock**.

En el mismo instante de tiempo el **Controlador** preguntara si el ***Dato*** de 8 bits cargado en el registro_i_d tiene **1´s**.

El **Controlador** debe recibir esta entrada interna de algún circuito que analiza el ***Dato*** cargado en la salida del registro_i_d y puede generar la señal **Num = 0 (Reg = 0**). En este caso puede ser una simple puerta **NOR** de 8 entradas.

Para evaluar si el Dato contiene **1´s** es necesario preguntar bit por bit si cada uno de ellos es 0 o 1.

La señal **Q0** debe llegar al circuito desde la salida del **registro_i_d**, desde el bit menos significativo (más a la derecha).

Si este bit es un **1** el **Controlador** debe generar la salida condicional de EnCnt que habilitara el conteo del **contador_up**. Ya que al contador_up le llega la misma señal de **Clock**, entonces, el **contador_up** se incrementa al finalizar el estado **Tb**.

Si el **Dato** no tiene ni un **1**, entonces, el **Controlador** va al estado **Tc**, genera la salida **Fin** y espera que la entrada **Start** se desactive. Este paso es necesario porque los **Sistemas Digitales** trabajan con una frecuencia de **Clock** muy grande, incomparable con el tiempo de reacción de un ser humano.

IMPLEMENTACION CON VHDL

Se puede ahora diseñar el circuito **Controlador** y los subcircuitos del **Procesador de Datos** en **VHDL** y luego interconectar todo utilizando el **Editor Grafico**.

Código VHDL para el contador_up.

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

entity contador_up is
    generic (m  : integer:= 4);
    port (  Resetn, Clock, En, Ld   : in std_logic;
            Ent             : in std_logic_vector(m-1 downto 0);
            Q               : buffer std_logic_vector(m-1 downto 0));
end contador_up;

architecture comportamiento of contador_up is
begin
    process (Resetn, Clock)
    begin
        if Resetn ='0' then Q <= (others=>'0');
        elsif (Clock'event and Clock ='1') then
            if En ='1' then
                if Ld ='1' then Q <= Ent;
                    else Q <= Q+'1';
                end if;
            end if;
        end if;
    end process;
end comportamiento;
```

Código VHDL para el registro de desplazamiento i_d.

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

entity registro_i_d is
    generic (   n           : integer:= 8);
    port     (  Entpar      : in std_logic_vector (n-1 downto 0);
                Ld, En, R   : in std_logic;
                Resetn      : in std_logic;
                Clock       : in std_logic;
                Q           : buffer std_logic_vector (n-1 downto 0));
end registro_i_d;

architecture comportamiento of registro_i_d is
begin
    process (Resetn, Clock)
    begin
        if Resetn ='0' then Q <= (others =>'0');
        elsif (Clock'event and Clock ='1') then
        if En ='1' then
            if Ld ='1' then Q <= Entpar;
            else desplazamiento: for i in 0 to n-2 loop
                Q(i) <= Q(i+1);
            end loop;
            Q(n-1) <= R;
            end if;
        end if;
        end if;
    end process;
end comportaniento;
```


01101010010010100110000101101110

Código VHDL para el controlador

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

entity controlador is
port (  Resetn, Clock   : in std_logic;
        Start, LdReg    : in std_logic;
        Regig0, Q0  : in std_logic;
        EnCnt,LdCnt     : out std_logic;
        EnReg,Fin               : out std_logic);
end controlador;

architecture comportamiento of controlador is
    type state is (Ta, Tb, Tc);
    signal y    : state;

begin
process (Resetn, Clock)
    begin
        if Resetn ='0' then y <= Ta;
            elsif (Clock'event and Clock ='1') then
            case y is
                when Ta =>  if  Start ='0' then y <= Ta; else y <= Tb;  end if;
                when Tb =>  if  Regig0 ='0' then y <= Tb; else y <= Tc; end if;
                when Tc =>  if  Start ='1' then y <= Tc; else y <= Ta;  end if;
            end case;
        end if;
end process;
```

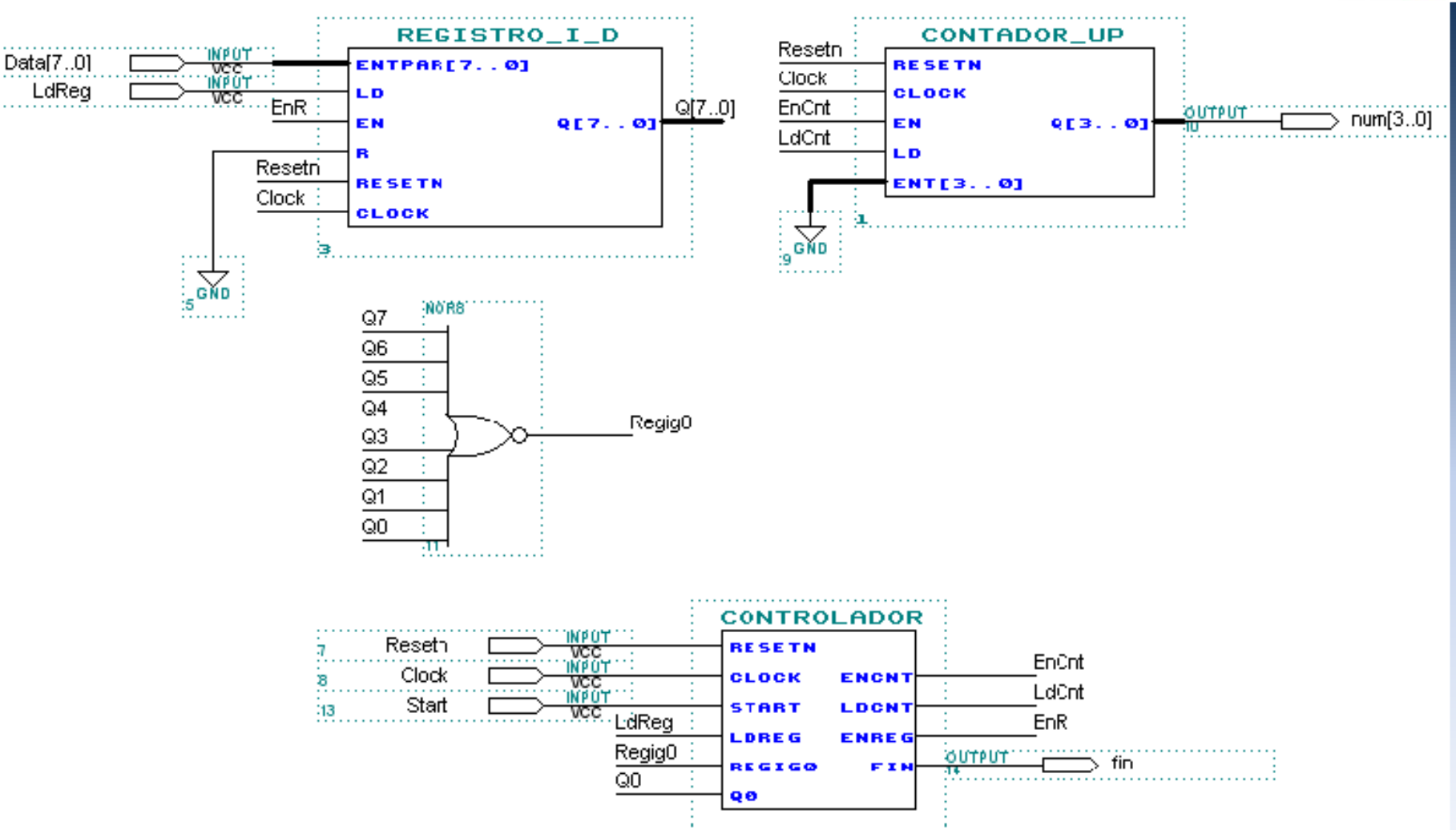

01100001011100110110000101101110011110100110000 1

Código VHDL para el controlador

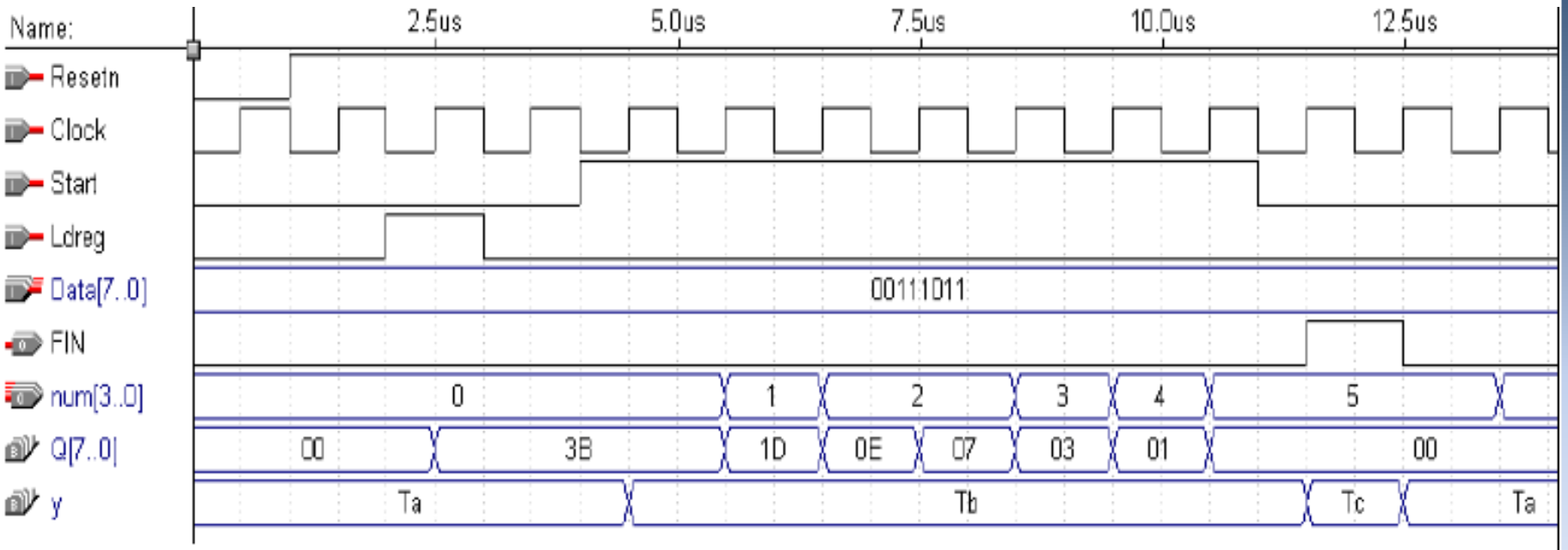
```
process (y, start, LdReg, Regig0, Q0)
    begin
        EnReg <= '0'; LdCnt <='0'; EnCnt <='0'; Fin <='0';
        case y is
            when Ta => EnCnt <='1'; LdCnt <='1';
                    if (start='0' and LdReg ='1') then EnReg <='1'; end if;
            when Tb => EnReg <='1';
                    if (Regig0='0' and Q0 ='1') then EnCnt <='1'; end if;
            when Tc => Fin <='1';
        end case;
end process;

end comportamiento;
```

Ahora podemos interconectar el bloque controlador con el procesador de datos.
Para esto usaremos el editor gráfico de QUARTUS II, conectando cada uno de los componentes previamente diseñados.

Diseño completo interconectado



Diagramas de tiempo del diseño completo



También se puede unir el Controlador con el Procesador de Datos usando la descripción estructural de VHDL.

Para esto, preparamos un package llamado componentes que incluya a los circuitos del Procesador para poder usarlos en este diseño o en cualquier otro

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

package componentes is
-- 1. contador que cuenta desde 0 hasta modulo-1 con habilitacion y carga
component contador_up
    generic (m  : integer:= 4);
    port (  Resetn, Clock, En, Ld   : in std_logic;
            Ent                     : in std_logic_vector(m-1 downto 0);
            Q                       : buffer std_logic_vector(m-1 downto 0));
end component;

-- 2. registro de desplazamiento a la derecha de n bits
-- con entradas de habilitación, resetn y carga
component registro_i_d
    generic (   n           : integer:= 8);
    port    (   Entpar      : in std_logic_vector (n-1 downto 0);
                Ld, En, R   : in std_logic;
                Resetn      : in std_logic;
                Clock       : in std_logic;
                Q           : buffer std_logic_vector (n-1 downto 0));
end component;

end componentes;
```

Código VHDL para el circuito Detector de 1's con descripción estructural

```
library ieee;
use ieee.std_logic_1164.all;
-- Se declara el paquete creado anteriormente
use work.componentes.all;

-- La entidad usa señales de entrada y salida de todo el sistema
entity detector_1_b is
port (  Resetn, Clock    : in std_logic;
        Start, LdReg     : in std_logic;
        Data             : in std_logic_vector (7 downto 0);
        Num              : buffer std_logic_vector (3 downto 0);
        Fin              : out std_logic);
end detector_1_b;

architecture comportamiento of detector_1_b is
    type estado is (Ta, Tb, Tc);
    signal y      : estado;
    -- Las señales intermedias se declaran con SIGNAL
    signal Q      : std_logic_vector (7 downto 0);
    signal Regig0, EnReg, EnCnt, LdCnt, R   : std_logic;
    signal Cero : std_logic_vector (3 downto 0);
```

Código VHDL para el circuito Detector de 1's con descripción estructural

```
-- Descripción de la MSS del controlador
begin
    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  Start ='0' then y <= Ta; else y <= Tb;  end if;
                when Tb =>  if  Regig0 ='0' then y <= Tb; else y <= Tc; end if;
                when Tc =>  if  Start ='1' then y <= Tc; else y <= Ta;  end if;
            end case;
        end if;
end process;

MSS_salidas: process (y, start, LdReg, Regig0, Q(0))
    begin
        EnReg <= '0'; LdCnt <='0'; EnCnt <='0'; Fin <='0';
        case y is
            when Ta => EnCnt <='1'; LdCnt <='1';
                    if (start='0' and LdReg ='1') then EnReg <='1'; end if;
            when Tb => EnReg <='1';
                    if (Regig0='0' and Q(0) ='1') then EnCnt <='1'; end if;
            when Tc => Fin <='1';
        end case;
end process;
```


01100001011100110110000101101110011110100110000 1

El primer process describe los cambios de estado del Controlador (Decodificador de Estado Siguiente y Memoria de Estados) y el segundo describe cuando el Controlador genera las salidas (Decodificador de Salida).

```
-- Conexiones externas al contador y al registro
Cero <="0000";
R <='0';

-- Creación de la señal Regig0 (puerta NOR)
Regig0 <='1' when Q = "00000000" else '0';

-- Instancias del registro y del contador

reg_desplazamiento: registro_i_d generic map (n => 8)
                    port map (Data, LdReg, EnReg, R, Resetn, Clock, Q);
contador_up1: contador_up   generic map (m => 4)
            port map ( Resetn, Clock, EnCnt, LdCnt, Cero, Num);

end comportamiento;
```

Diagramas de tiempo del diseño estructural completo