Universidad de Castilla-La Mancha
Escuela Superior de Informática

Diseño VLSI

Laboratorio 9

Generador de secuencia temporizado

Autor: María Espinosa Astilleros

Fecha: 26 de abril de 2020

Índice

1. INTRODUCCIÓN

El objetivo de este laboratorio es generar una secuencia de 8 valores (de 4 bits cada uno) que se repetirá indefinidamente. Las cifras serán generadas en orden desde la posición izquierda (más significativa) hacia la derecha (menos significativa). Además, el sistema se completará con una señal de reset activo a nivel bajo y una señal de habilitación activa a nivel alto.

2. PLANTEAMIENTO

Primero dibujé el diagrama de estados del circuito:

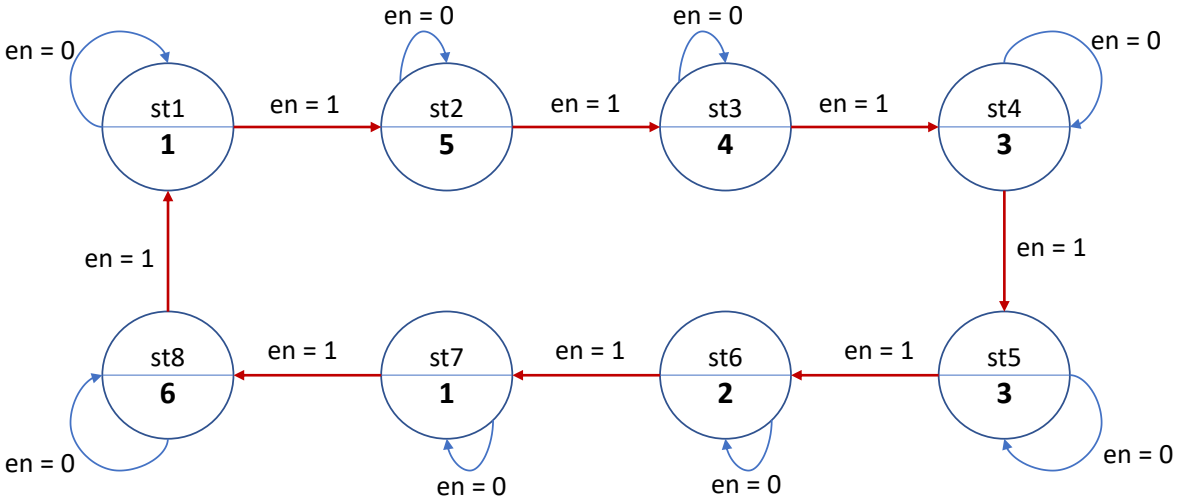

Figura 1: Diagrama de estados

En 1 cada estado será cada uno de los digitos. Por lo tanto tendremos 8 estados y cada uno de ellos solamente dependerán del estado anterior, por lo que nos encontramos con un sistema de Moore.
Una vez que sabemos los estados que forman el diagrama hay que indicar cuándo cambiaremos de estado. En este caso pasaremos al estado siguiente cuando la señal de habilitación (enable) esté activa y permaneceremos en el mismo estado en caso de que la señal de habilitación esté desactivada.

Una vez que ya tenemos el diagrama de estados lo podemos adaptar en una tabla de estados:

Cuadro 1: Tabla de estados

	Estado siguiente		Salida	
Estado inicial	en = 0	en = 1	en = 0	en = 1
st1	st1	st2	0001	0101
st2	st2	st3	0101	0100
st3	st3	st4	0100	0011
st4	st4	st5	0011	0011
st5	st5	st6	0011	0010
st6	st6	st7	0010	0001
st7	st7	st8	0001	0110
st8	st8	st1	0110	0001

Como se puede observar en la tabla 1 si la señal de habilitación está activada mostraremos el estado siguiente y si la señal de habilitación está deshabilitada mostraremos el estado actual.

3. RESOLUCIÓN DEL PROBLEMA

Para programar todo lo explicado anteriormente creamos primero las entradas y las salidas (1). En este caso las entradas serán la señal del reloj, el reset y la señal de habilitación y la salida será el estado dependiendo de la señal de habilitación y del estado anterior.

Listing 1: Declaración de entradas y salidas

```
1 ENTITY secuencia IS
2       PORT (clk, resetn : IN STD_LOGIC;          -- reloj y reset asincrono a nivel bajo
3             en: IN STD_LOGIC;                    -- señal de habilitación
4             z : OUT STD_LOGIC_VECTOR(0 TO 3));   -- salida de la secuencia
5   END ENTITY;
```

Posteriormente (2) pasamos a declarar los estados de nuestro circuito y creamos un signal *estado* para poder utilizarlo posteriormente.

Listing 2: Creación de los estados

```
1 TYPE estado_Type IS (st1, st2, st3, st4, st5, st6, st7, st8);
2 SIGNAL estado : estado_Type;
```

Ahora tocaría realizar la funcionalidad de nuestro diagrama de estados (3). Primero comprobamos si el reset se ha habilitado, ya que en ese caso el circuito se reiniciará al primer estado; si el reset no está activado es cuando indicamos el estado siguiente de nuestro circuito.

Listing 3: Funcionalidad del diagrama de estados

```
PROCESS (clk, resetn)
      BEGIN
          IF resetn = '0' THEN
              estado <= st1; -- Iniciamos otra vez el contador a su estado inicial
          ELSIF (clk'EVENT AND clk = '1') THEN
              -- El siguiente estado se determina dependiendo del estado actual y la señal de habilitacion
              CASE estado IS
                  WHEN st1=>
                      IF en = '1' THEN
                          estado <= st2;
                      ELSE
                          estado <= st1;
                        END IF;
                    WHEN st2=>
                      IF en = '1' THEN
                          estado <= st3;
                      ELSE
                          estado <= st2;
                        END IF;
                    WHEN st3=>
                      IF en = '1' THEN
                          estado <= st4;
                      ELSE
                          estado <= st3;
                        END IF;
                    WHEN st4=>
                      IF en = '1' THEN
                          estado <= st5;
                      ELSE
                          estado <= st4;
                        END IF;
                    WHEN st5=>
                      IF en = '1' THEN
                          estado <= st6;
                      ELSE
                          estado <= st5;
                        END IF;
                    WHEN st6=>
                      IF en = '1' THEN
                          estado <= st7;
                      ELSE
                          estado <= st6;
                        END IF;
                    WHEN st7=>
                      IF en = '1' THEN
                          estado <= st8;
                      ELSE
                          estado <= st7;
                        END IF;
                    WHEN st8=>
                      IF en = '1' THEN
                          estado <= st1;
                      ELSE
                          estado <= st8;
                        END IF;
              END CASE;
          END IF;
      END PROCESS;
```

Finalmente (4), dependiendo del estado que hayamos obtenido en las líneas de código anteriores indicamos cuál es la salida que se va a mostrar.

Listing 4: Mostramos la salida dependiendo del estado

```
PROCESS (estado)
        BEGIN
            CASE estado IS
                WHEN st1 =>
                        z <= "0001";
                    WHEN st2 =>
                        z <= "0101";
                    WHEN st3 =>
                        z <= "0100";
                    WHEN st4 =>
                        z <= "0011";
                    WHEN st5 =>
                        z <= "0011";
                    WHEN st6 =>
                        z <= "0010";
                    WHEN st7 =>
                        z <= "0001";
                    WHEN st8 =>
                      z <= "0110";
            END CASE;
        END PROCESS;
```

4. SIMULACIÓN

Una vez realizado el proyecto comprobamos que funciona mediante la simulación del *Quartus Prime*.

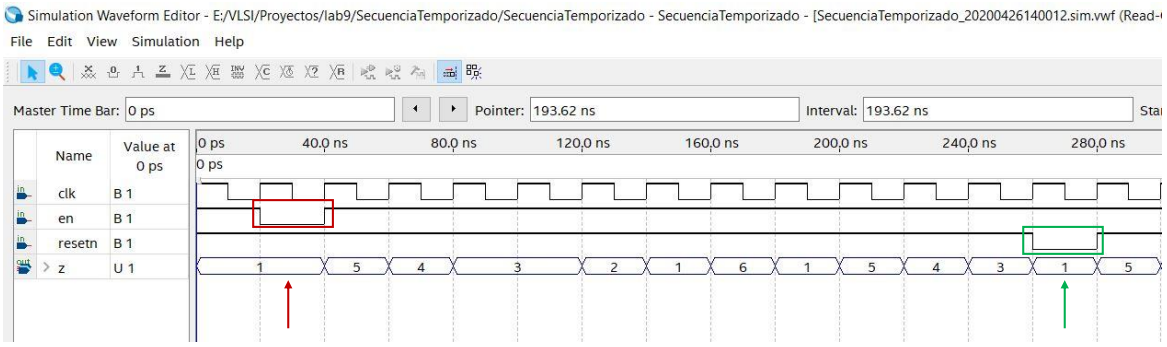

Figura 2: Simulación del proyecto

Como se puede observar en la figura 2 aparece la secuencia de los digitos. Cuando el enable está desactivado (recuadro rojo) el estado permanece en el mismo y cuando vuelve a activarse pasa al siguiente. En el caso del reset cuando está activado (recuadro verde) la secuencia se reinicia para volver a mostrar el primer digito.