PROYECTO 1 : Juego De Luces

Trabajo Realizado por:
Arnau Mora Gras & Carlos Villena Jiménez
amorgra@teleco.upv.es & cviljim@teleco.upv.es

Índice:

Subtarea 1. Diseño de un Contador Digital Parametrizable.

Se trata de la creación en Verilog HDL de un Diseño de un contador binario síncrono de módulo parametrizable ascendente/descendente.

Cuya función principal del juego de luces será la de incrementar o decrementar el valor binario del sistema dependiendo de una entrada en concreto, la cual será “UP_DOWN”.

El bloque a diseñar tiene la siguiente forma:

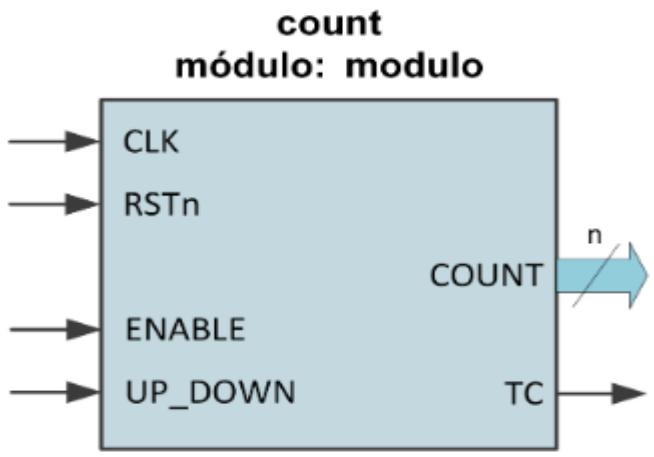

Fig1. Diseño planteado para el contador.

Los **parámetros** que componen este módulo son:

→ módulo: Determina el valor máximo hasta el que se quiere contar.

→ N: Determina el valor de “bits” necesarios para la cuenta del “módulo”.

Las **entradas** que componen este módulo son:

→ CLK: Señal de reloj activo a nivel alto (1).

→ RSTn: Señal de reset asíncrono activo a nivel bajo (0).

→ ENABLE: Señal de funcionamiento activo a nivel alto (1).

→ UP_DOWN: Señal de Incremento ó Decremento del contador (1→UP ; 0→DOWN).

Las **salidas** que compone este módulo son:

→ COUNT: Registro del valor de la cuenta actual con tamaño “N-bits”.

→ TC: Señal de fin de cuenta(Terminal Counter). Si está a “1”,COUNT al máximo.

En cuanto a la **Explicación del Código del Contador** es muy sencilla, a parte de que en el propio código está comentada todas y cada una de las líneas.
Primeramente inicializamos los “parameters”, “inputs” y “outputs” mencionados anteriormente.
Justo después implementamos un bloque procedural “always” con una lista de sensibilidad en donde se encuentran el “CLK” activo a nivel alto y el “RSTn” a nivel bajo, donde se activa este bloque siempre que cambie la señal de reloj o la señal de reset.
Dentro de este bloque, se encuentra todas las posibles acciones a tener en cuenta para un contador que actúa de forma ascendente o descendente, como por ejemplo:
→ Si el reset se activa, ponemos a cero el valor del contador.
→ Si el reset no se activa y el “ENABLE” está activado, entonces tendrá que incrementar el contador en 1 su valor anterior, o, decrementar en 1 su valor anterior, dependiendo si la señal de entrada “UP_DOWN” está en “1” ó “0”, respectivamente.
También habrá que tener en cuenta que si el contador en ese momento ya había llegado a su “FIN de Cuenta” habría que haberlo inicializado previamente.
→ Finalmente y fuera del bloque procedural asignaremos a la señal de salida de Fin de Cuenta “TC” el valor de “1” si y sólo si el valor de “COUNT == (módulo-1)”.

El RTL_viewer del diseño del contador tiene la siguiente forma:

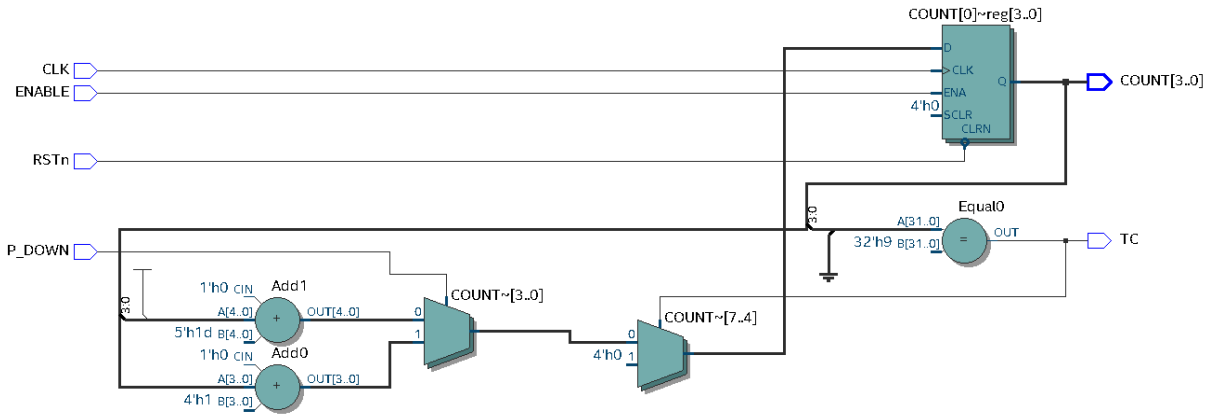

Fig2. Disero RT-viewer final del contador.

Subtarea 1.2. Diseño del TestBench del módulo del Contador Parametrizable.

En este apartado se ha diseñado lo que es el testeo del módulo del contador parametrizable mediante verilog HDL.
Esto ha consistido en la inicialización de parámetros propios del contador sobre este diseño mediante la instancia del propio módulo del contador para que se tomen como referencia dichas señales.

Seguido de esto se genera un “clock” en donde cada medio periodo se invierte el flanco de reloj, siendo el periodo de (T=20 ns).

En el apartado del “Test procedure” con un bloque Initial inicializamos en T=0 todas las señales primaria de forma que el “CLK = 0” , “RSTn = 0” , “ENABLE = 0”, “UP_DOWN = 1”. Así pues, está en flanco de bajada, se activa el reset, para que vuelva al origen del contador, el habilitador está desactivado y la señal de UP_DOWN está en modo “incrementar contador”.

Ya después de medio periodo más se cambia algunos parámetros y vamos haciendo cambios en los valores de las señales para ver si lo que debe de hacer es lo que hace en las “waves” generadas.

Algunas de las waves generadas son las siguientes:

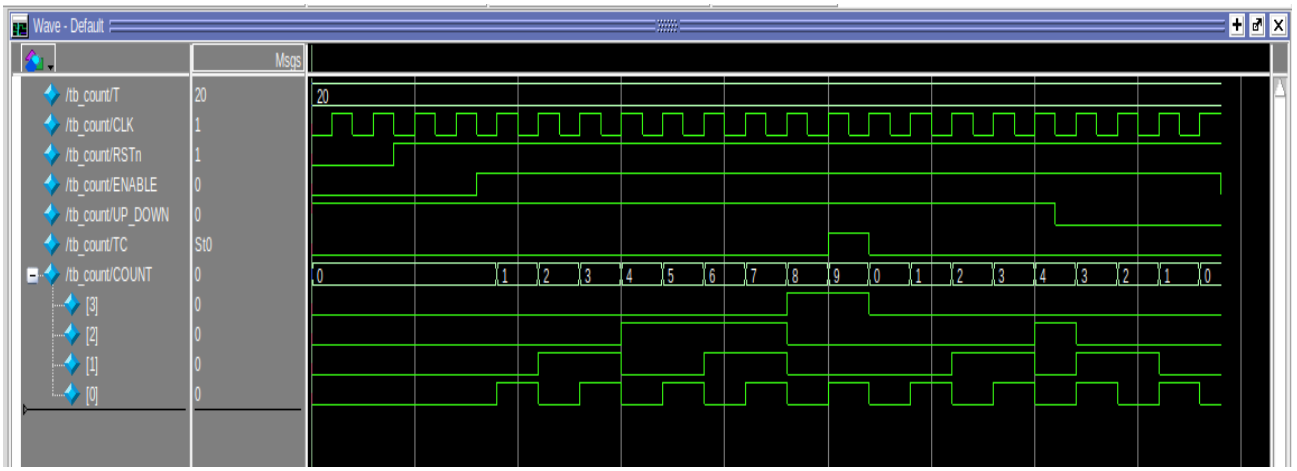

Fig3: Wave del TestBench del contador.

En la figura número 1, podemos observar que inicialmente, como está declarado en el testbench, no debe incrementar y decrementar el contador ya que todas las señales de activación de este están desactivadas.
Una vez el ciclo de reloj ha cambiado al flanco positivo, el reset está desactivado y el Enable activado, es cuando se ve un incremento del contador desde cero hasta 9 (modulo-1).
Cuando llega a “9” vemos también cómo el “TC” ha cambiado de estar a nivel bajo, a nivel alto por un ciclo de reloj, hasta que se ha iniciado de nuevo el contador a cero.
De vuelta sigue con la incrementación, pero cuando llega al valor decimal “4”, la señal “UP_DOWN”, se ha puesto en nivel bajo y empieza a decrementar el contador.

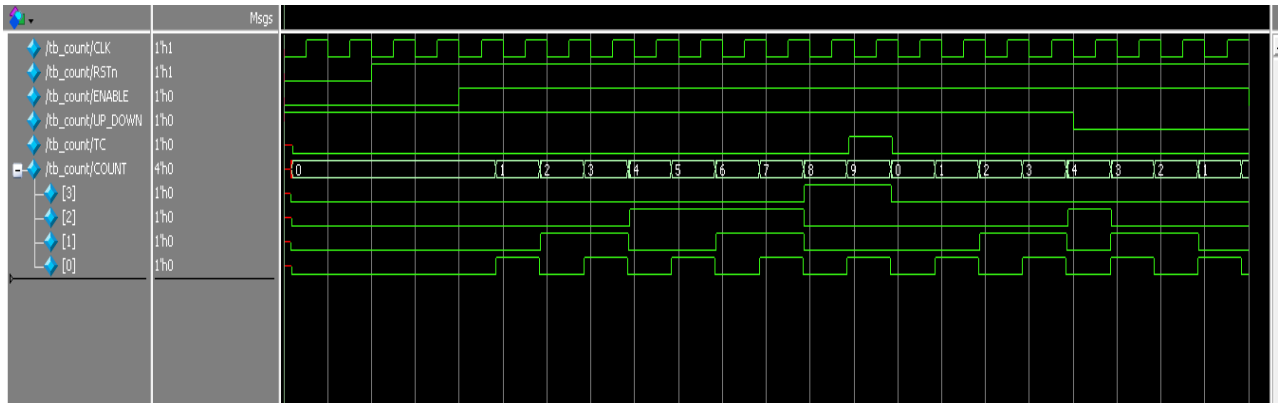

Fig4: Wave del TestBench del contador en gate-level.

En la figura número 2, podemos observar el funcionamiento del gate-level, que básicamente es la simulación de igual forma que la explicada anteriormente,pero con el cambio de que se observan ahora los retardos y pérdidas de conexión, por eso se visualiza las ondas más alargadas.
Como también nos damos cuenta de los latches que hay al principio del temporizador (Se observa de un color rojizo).
La diferencia más rápida de observar, es que cada salida, cambia de forma no directa al cambio de la señal de entrada que le permite la modificación del contador.

Subtarea 2. Diseño de un Registro de Desplazamiento Parametrizable.

Se requiere de un registro de desplazamiento con ancho de bits parametrizable (width), cuya función principal en el proyecto a realizar (Juego de Luces) será la de desplazar síncronamente al reloj los bits de los cuales serán usados para la visualización en la subtarea final, la cual sería la de implementación en placa FPG-A.

El bloque el cual se pretende diseñar es el siguiente:

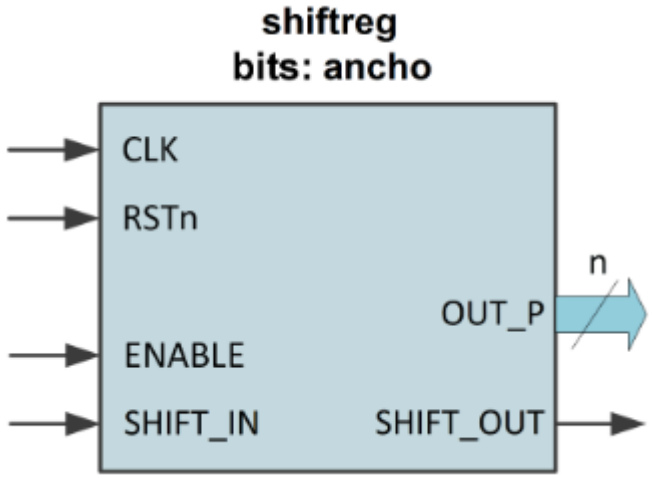

Fig5. Diseño planteado previo al mismo del Registro de desplazamiento Parametrizable.

Los **parámetros** que componen este módulo son:

→ width: Determina el ancho de bits en los cuales se realiza el desplazamiento.

→ n: Determina el número de “bits” necesarios para el ancho “width”.

Las **entradas** que componen este módulo son:

→ CLK: Señal de reloj activo a nivel alto (1).

→ RSTn: Señal de reset asíncrono activo a nivel bajo (0).

→ ENABLE: Señal de funcionamiento activo a nivel alto (1).

→ SHIFT_IN: Señal de DATO de entrada en “serie” (entran de uno en uno).

Las **salidas** que compone este módulo son:

→ SHIFT_OUT: Registro del valor de la salida actual del desplazamiento en “serie”.

→ OUT_P: Registro del valor de la salida actual del desplazamiento en “paralelo”.

Los **registros** que componen este módulo son:

→ mem: Registro de la memoria del módulo con tamaño “n-1 bits”.

En cuanto a la **Explicación del Código del Registro de Desplazamiento** es muy sencillo, a parte de que en el propio código está comentada todas y cada una de las líneas.
Primeramente inicializamos los “parameters”, “inputs” , “outputs” y “registros” mencionados anteriormente.
Justo después implementamos un bloque procedural “always” con una lista de sensibilidad en donde se encuentran el “CLK” activo a nivel alto y el “RSTn” a nivel bajo, donde se activa este bloque siempre que cambie la señal de reloj o la señal de reset.
Dentro de este bloque, se encuentra todas las posibles acciones a tener en cuenta para un contador que actúa de forma ascendente o descendente, como por ejemplo:
→ Si el reset se activa, reiniciamos la memoria y ponemos las salidas a “cero”.
→ Si el reset no se activa y el “ENABLE” está activado, entonces declaramos la salida en “serie” con el bit que se va a perder por el propio desplazamiento, luego desplazamos un bit de la memoria a “izquierdas”, entonces añadimos el bit de entrada en “serie” en esa posición exacta de la memoria (mem) y actualizaremos por último la salida en paralelo con el valor completo de la memoria.

El RTL_viewer del diseño del registro tiene la siguiente forma:

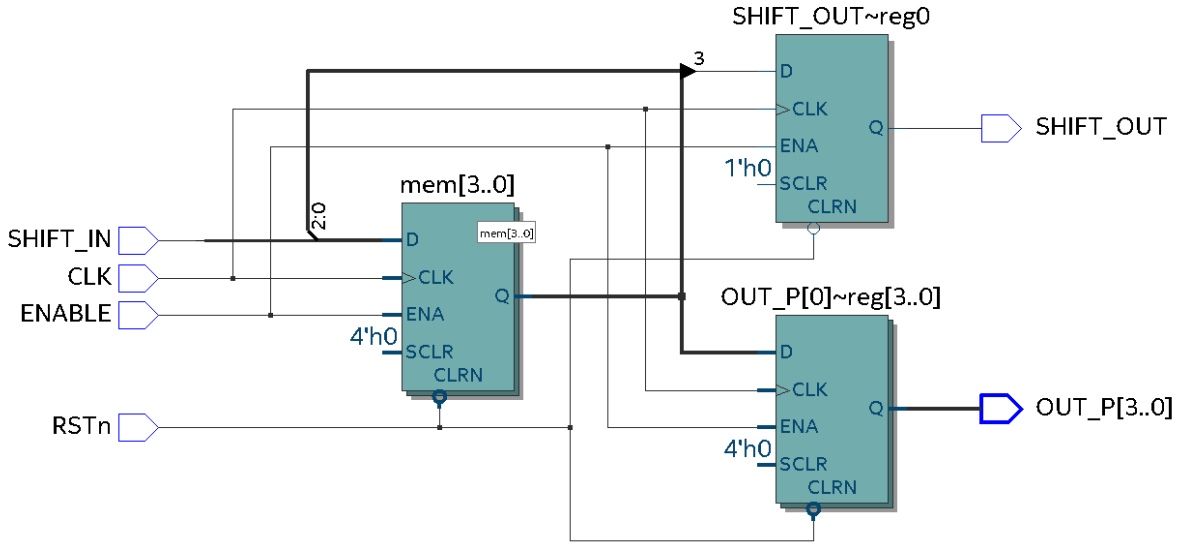

Fig6. Diseño RTL-viewer final del diseño del shifter.

Subtarea 2.2. Diseño del TestBench del módulo del Registro Parametrizable.
En este apartado se ha diseñado lo que es el testeo del módulo del registro de desplazamiento con ancho de bit parametrizable mediante verilog HDL.
Esto ha consistido en la inicialización de parámetros propios del contador sobre este diseño mediante la instancia del propio módulo del contador para que se tomen como referencia dichas señales.

Seguido de esto se genera un “clock” en donde cada medio periodo se invierte el flanco de reloj, siendo el periodo de (T=20 ns).

En el apartado del “Test procedure” con un bloque Initial inicializamos en T=0 todas las señales primaria de forma que el “CLK = 0” , “RSTn = 0” , “ENABLE = 0”, “SHIFT_IN = 0”.

Así pues, está en flanco de bajada, se activa el reset, para que vuelva al origen de memoria, el habilitador está desactivado y los datos de entrada son nulos aún, hasta que otro medio ciclo de reloj después le introducimos un bit “1” y así le vamos añadiendo valores a las entradas de diferentes formas.
Recordar que en el propio código se ve todo muy claro.

A continuación se adjuntan un par de imágenes de las simulaciones del testbench del módulo del registro de desplazamiento parametrizable.

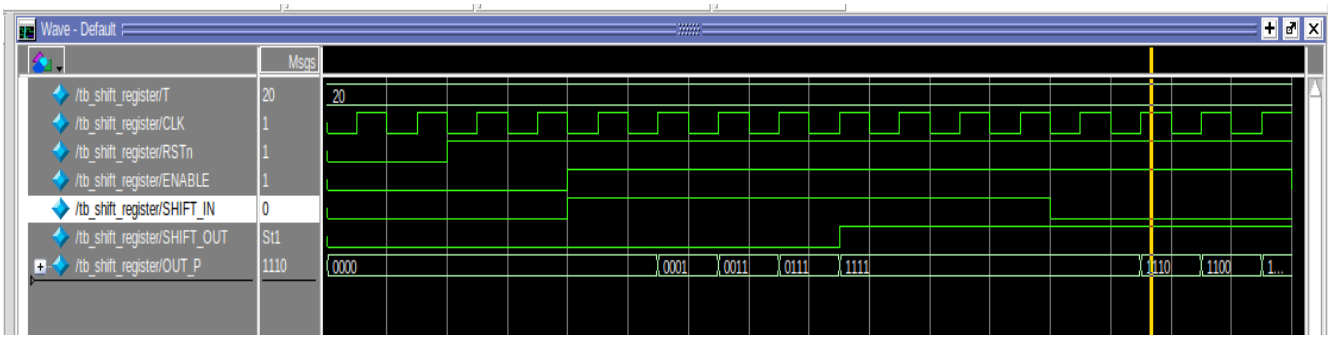

Fig7. Simulación del TestBench del Registro de Desplazamiento Parametrizable.

Subtarea 3. Diseño del TOP (Juego de Luces).

En esta subtarea se deben implementar tanto el “contador” como el “registro de desplazamiento” comentados anteriormente de tal forma que el contador actúe como un “Divisor de Frecuencia” ya que el reloj interno de la placa funciona a “50” millones de Hercios y lo tenemos que minimizar para que el desplazamiento de LEDS se observe cada “0.25” segundos. De este modo el registro también ha de ser modificado de tal forma que este actúe como un “Contador Johnson” generando así 14 ciclos.

La implementación la cogemos del guión de la propia práctica:

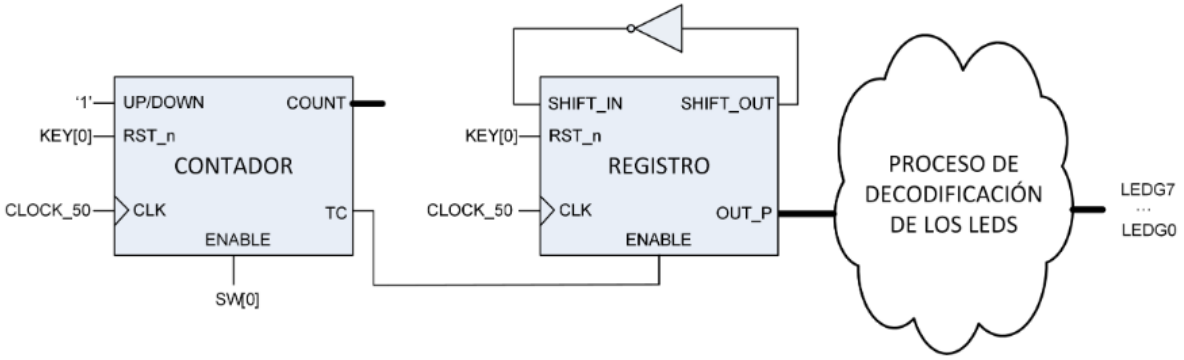

Fig8. Diagrama de bloques del TOP.

Y la estructura de LEDs que se quiere lograr es la siguiente:

Ciclos	LEDG7	LEDG6	LEDG5	LEDG4	LEDG3	LEDG2	LEDG1	LEDG0
1	■							
2		■						
3			■					
4				■				
5					■			
6						■		
7							■	
8								■
9							■	
10						■		
11					■			
12				■				
13			■					
14		■						

Fig10. Onda de activación y desactivación de LEDs.

Para ello realizamos entonces unos nuevos módulos de contador y registro de desplazamiento para así no perder como tal los archivos ya realizados.

Para el nuevo módulo del contador, como divisor de frecuencia, cogeremos el creado en la práctica “0” que es la misma idea.

Lo que buscamos aquí es instanciar dos veces el contador realizado para la subtarea #1, en donde la primera instancia actúa como divisor de frecuencia y la segunda instancia actuaría como final de cuenta.

El módulo declarado para que vaya a 0.25 segundos, es de 12.5 millones, ya que **(12.5 / 50(Hz)) = 0.25(s)** y en el módulo de la segunda instancia la ponemos como el valor que queramos contar como máximo.

Todo esto es lo mismo que instanciar directamente en el TOP y poner dicho módulo.

Para el nuevo módulo del registro de desplazamiento(johnson), haremos finalmente lo último comentado para el contador, es decir, hacemos lo cambios justos y oportunos para que actúe de esta forma en la instanciación del “shift_register” en el TOP.

En cuanto al desarrollo del decodificador, lo realizamos en el propio diseño del TOP , el cual lo haremos mediante un bloque procedural “always”, cuya lista de sensibilidad será la señal de salida “OUTPUT” de tamaño “7-bits” (14-ciclos).

Aquí se incluirá una estructura de control de flujo del tipo “case/default”, en donde a cada uno de los desplazamiento en cuanto a bit se le asigna una salida de tamaño “8-bits”, para la representación de los propios LEDs.

El RTL-viewer del diseño del “TOP” quedaría tal que así:

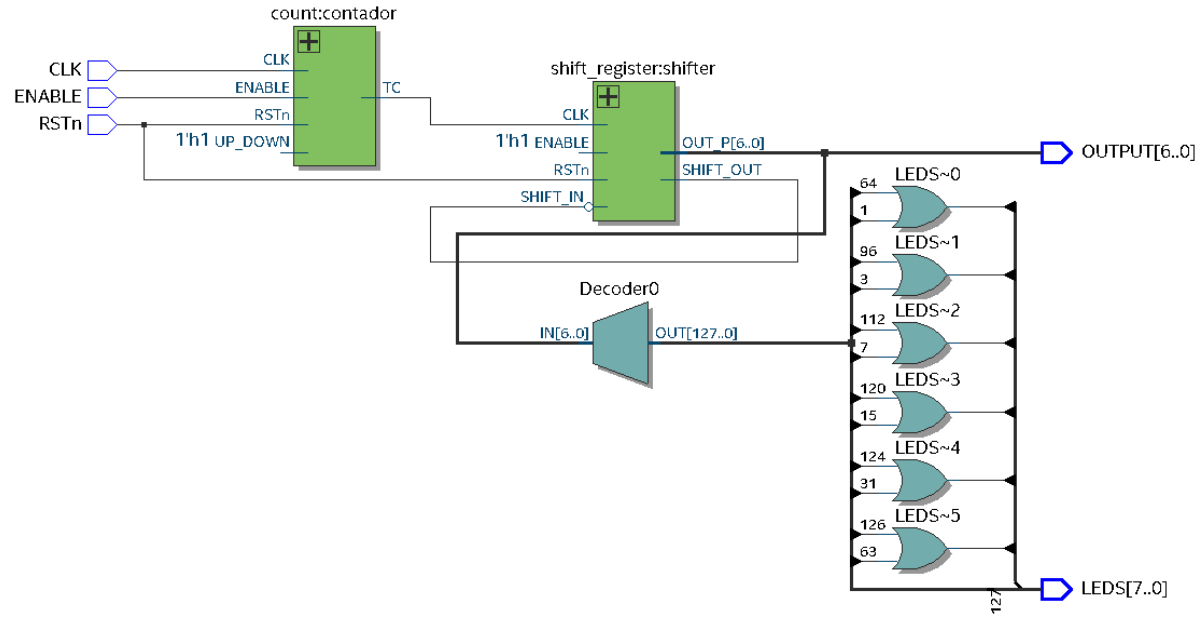

Fig11. Diseño RTL-viewer final del TOP.

Subtarea 3.2. Diseño del TestBench del TOP (Juego de Luces).

Como todos los demás testbench realizados y comentados en este primer proyecto, instanciamos lo que es el módulo del “TOP”, declaramos la entradas mínimas y necesarias, como lo son, el reloj(CLK), enable(ENABLE), reset(RSTn), salida(OUTPUT).

Luego generamos el propio reloj, de igual forma que los anteriores y seguidamente iniciamos el “Test Procedure” con un “initial”.

Las imágenes de la simulación del “waveform” es la siguiente:

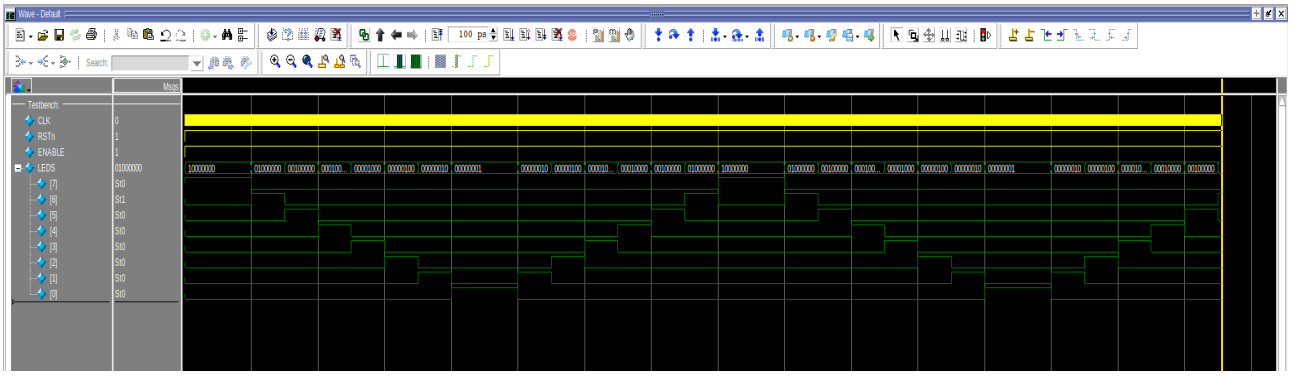

Fig12. Simulación del TestBench del Juego de Luces.

Subtarea 3.3. Verificando en Placa el TOP (Juego de Luces).

En esta subtarea verificaremos el diseño del juego de luces sobre la placa del laboratorio, la cual es la placa DE2-115.

Para ello se realizará la asignación de PINES en el pinplanner de Quartus de la forma que se detalla en la siguiente imagen:

Nombre Puerto	Pin FPGA	Nombre Puerto	Pin FPGA
CLOCK_50	Y2	LEDG[0]	E21
		LEDG[1]	E22
KEY[0]	M23	LEDG[2]	E25
KEY[1]	M21	LEDG[3]	E24
		LEDG[4]	H21
SW[0]	AB28	LEDG[5]	G20
SW[1]	AC28	LEDG[6]	G22
		LEDG[7]	G21

Fig13. Imagen de la asignación de PINES a realizar sobre el pinPlanner.

Se adjuntará en la entrega un vídeo grabado de la placa en el laboratorio, mostrando el funcionamiento del sistema de juego de luces.

Subtarea 3.4. Verificando casos EXTREMOS.

En esta última parte de la tarea, verificaremos uno de los casos extremos propuestos, que es: ¿Qué ocurre con la verificación del TOP en el caso de haber llegado a la cuenta final (TC=1 y el enable está en ENABLE=0)?

Concluimos, de que no ocurre nada realmente raro, ya que al estar el enable desactivado, el programa no puede seguir dando pasos(contar, descontar), entonce se observará en este caso, el caso “default” del decoder propuesto, es decir: todos los LEDs desactivados.

En la siguiente imagen se mostrará la simulación del mismo testbech:

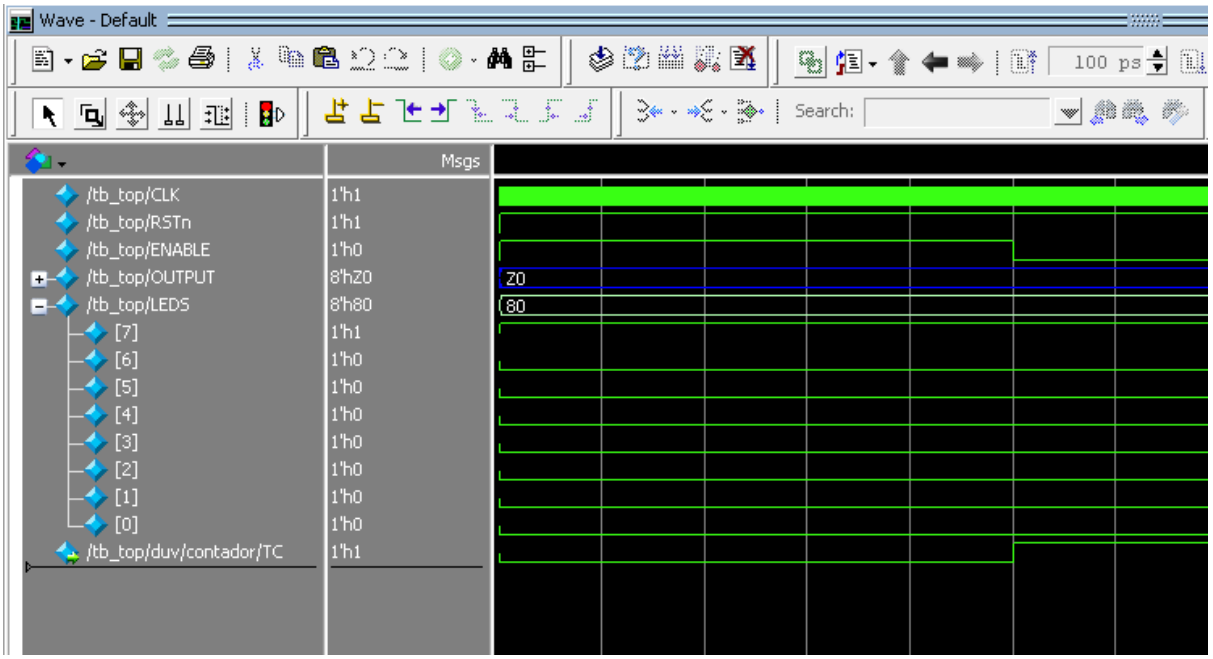

Fig14. Simulación de un caso EXTREMO (TC=1 ; ENABLE=0).

El cambio realizado para este caso EXTREMO en cuanto a diseño del testbench ha sido muy simple. Quedaría tal que así, como se muestra en la siguiente imagen:

```
#(T*2)

ENABLE = 1;

@(posedge duv.contador.TC);
ENABLE = 0;

#(T*50000000);

#(T*4)

ENABLE = 0 ;

$display("Test finished") ;
$stop ;


e
```

La parte del “duv” es la que ha sido añadida.