Laboratorio 4

Gabriel Baigorri

Gabriel fernández

Fecha de presentación: 13/05/2016

2016

Contenidos

[Índice de Figuras ii](#_Toc451164640)

[Índice de Tablas ii](#_Toc451164641)

[Objetivo 1](#_Toc451164642)

[Desarrollo 1](#_Toc451164643)

[Parte A 1](#_Toc451164644)

[Consigna 1](#_Toc451164645)

[Resolución 1](#_Toc451164646)

[Simulación y Test-Bench 3](#_Toc451164647)

[Parte B 3](#_Toc451164648)

[Consigna 3](#_Toc451164649)

[Resolución 4](#_Toc451164650)

[Simulación y Test-Bench 6](#_Toc451164651)

[Reporte de área 7](#_Toc451164652)

[Parte C 7](#_Toc451164653)

[Consigna 7](#_Toc451164654)

[Resolución 8](#_Toc451164655)

[Simulación y Test-Bench 8](#_Toc451164656)

[Reporte de área 9](#_Toc451164657)

[Parte D 10](#_Toc451164658)

[Consigna 10](#_Toc451164659)

[Resolución 10](#_Toc451164660)

[Simulación y Test-Bench 12](#_Toc451164661)

[Reporte de área 13](#_Toc451164662)

[Problemas y soluciones 15](#_Toc451164663)

[Conclusión 15](#_Toc451164664)

[Apéndices i](#_Toc451164665)

[Apéndice A i](#_Toc451164666)

[Códigos de Componentes i](#_Toc451164667)

[Test-Bench iii](#_Toc451164668)

[Apéndice B iv](#_Toc451164669)

[Códigos de Componentes iv](#_Toc451164670)

[Test-Bench vi](#_Toc451164671)

[Apéndice C viii](#_Toc451164672)

[Códigos de Componentes viii](#_Toc451164673)

[Test-Bench ix](#_Toc451164674)

[Apéndice D xi](#_Toc451164675)

[Códigos de Componentes xi](#_Toc451164676)

[Test-Bench xiv](#_Toc451164677)

[Presupuesto xvi](#_Toc451164678)

# Índice de Figuras

[Figura 1 3](#_Toc451017782)

[Figura 2 4](#_Toc451017783)

[Figura 3 5](#_Toc451017784)

[Figura 4 5](#_Toc451017785)

[Figura 5 7](#_Toc451017786)

[Figura 6 8](#_Toc451017787)

[Figura 7 8](#_Toc451017788)

[Figura 8 9](#_Toc451017789)

[Figura 9 11](#_Toc451017790)

[Figura 10 11](#_Toc451017791)

[Figura 11 13](#_Toc451017792)

[Figura 12 14](#_Toc451017793)

[Figura 13 14](#_Toc451017794)

[Figura 14 15](#_Toc451017795)

# Índice de Tablas

[Tabla 1 3](#_Toc451017796)

[Tabla 2 6](#_Toc451017797)

[Tabla 3 7](#_Toc451017798)

[Tabla 4 10](#_Toc451017799)

[Tabla 5 11](#_Toc451017800)

[Tabla 6 13](#_Toc451017801)

[Tabla 7 14](#_Toc451017802)

# Objetivo

* Interpretación de información especificada en hojas de datos o especificaciones de diseño
* Utilización de las instrucciones secuenciales, concurrentes y paquetes aprendidos en clase.
* Comprensión de la metodología de diseño jerárquico (top-to-down).
* Uso de archivo de restricción para asignar los pines E/S del FPGA vinculados a hardware del board DE2-115.
* Uso de sincronizadores.
* Escritura de restricción de tiempo, frecuencia de trabajo del sistema, en el archivo de restricción.
* Uso y escritura de test bench para verificar funcionalmente el correcto comportamiento de un sistema descrito en VHDL.
* Utilización de la herramienta *MegaWizard* para la generación de un divisor de frecuencia basado en PLL.
* Configuración del FPGA del board DE2-115 con el código VHDL correspondiente.

# Desarrollo

## Parte A

### Consigna

Realizar la descripción en VHDL de un contador tipo LFSR de 4 bits.

* Genere y trate de comprender el circuito *RTL View*.
* Verifique su funcionalidad con un test bench ejecutando su simulación funcional y simulación post-place & route (gate level simulation).

### Resolución

Primeramente se buscó un modelo de contador LFSR de 4 bits, encontrándose el mostrado en la Figura 1, cuya tabla Estado-Salida se observa en la Tabla 1.

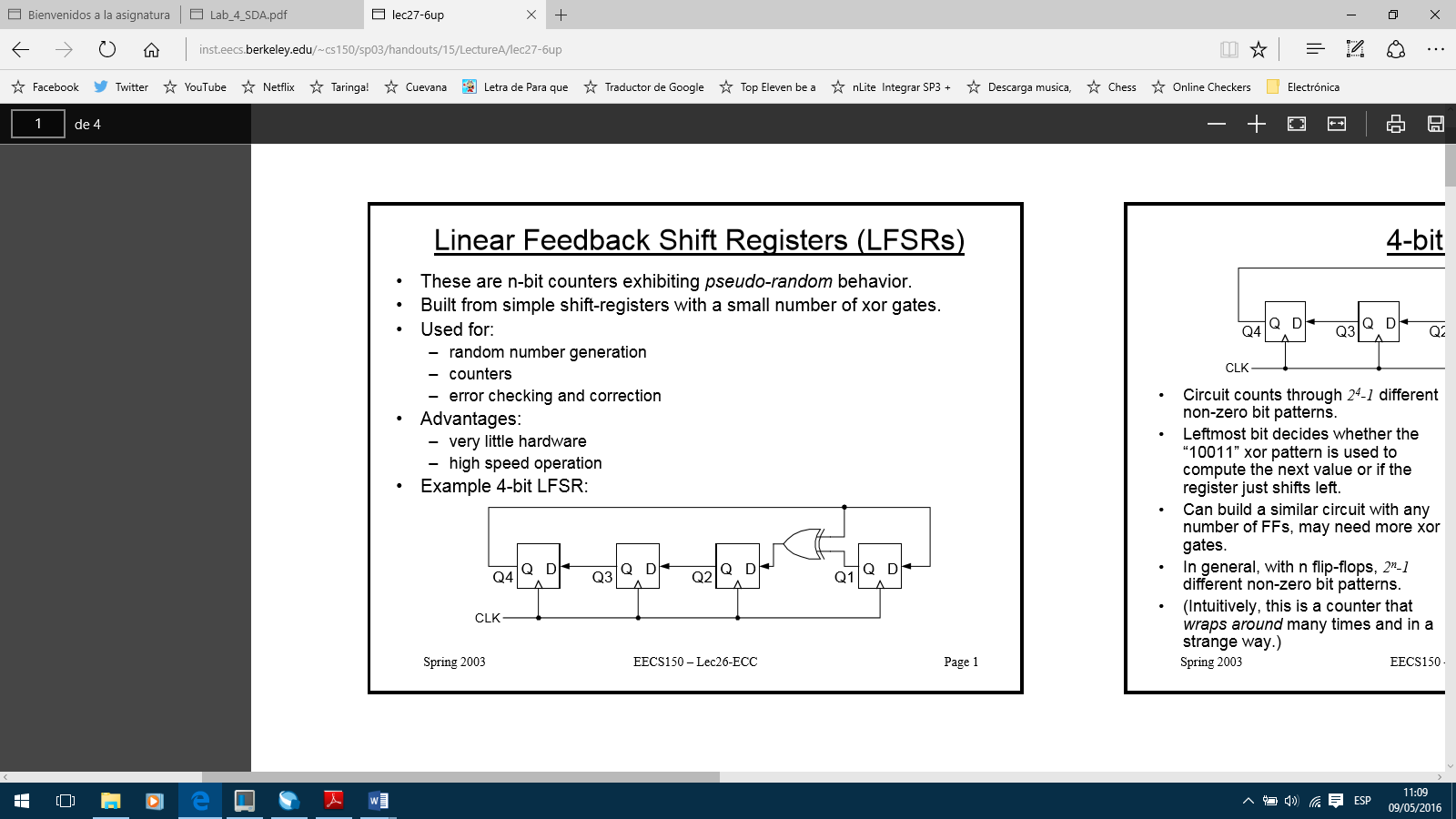


Figura 1

Tabla

|  |  |
| --- | --- |
| Estado | Salida |
| 1 | 1111 |
| 2 | 1101 |
| 3 | 1001 |
| 4 | 0001 |
| 5 | 0010 |
| 6 | 0100 |
| 7 | 1000 |
| 8 | 0011 |
| 9 | 0110 |
| 10 | 1100 |
| 11 | 1011 |
| 12 | 0101 |
| 13 | 1010 |
| 14 | 0111 |
| 15 | 1110 |

Con ello se procedió a realizar el modelo de Flip Flop D, descripto en el Apéndice A. Se debe aclarar que se usó una entrada de seteo, y no de reseteo, debido a que el estado “0000” no pertenece a la secuencia del contador.

Obtenido el modelo del Flip Flop, se sintetizó el contador LFSR, mostrado en la Figura 2.

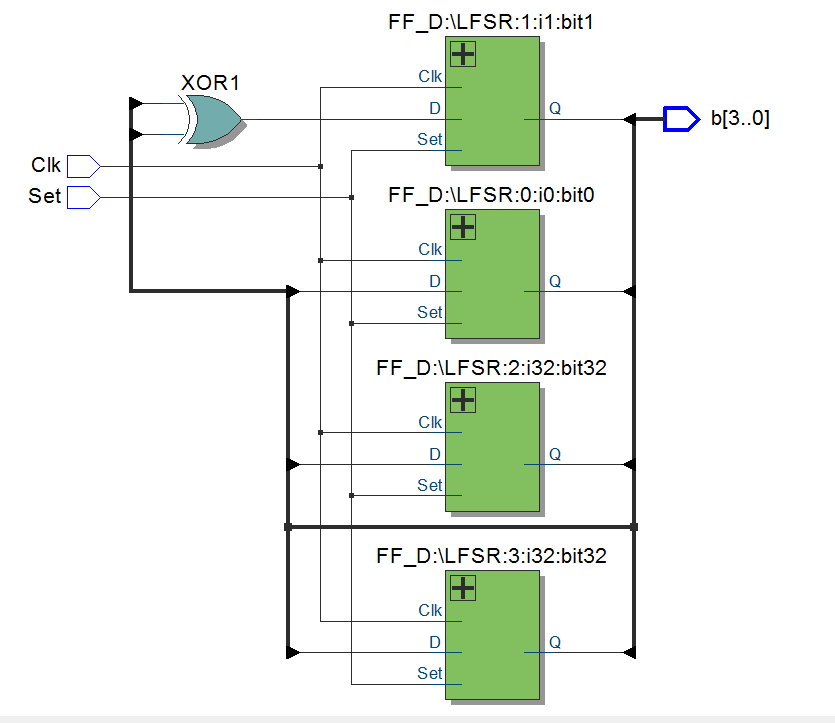


Figura 2

### Simulación y Test-Bench

Se realizó el código Test-Bench, el cual setea todas las salidas y luego espera 16 flancos de reloj. Luego se comprobó que la secuencia generada fuese coincidente con la tabla de estados del contador mostrada anteriormente. El código mencionado se encuentra en el Apéndice A.

La formas de onda de entradas y salidas para la simulación funcional se observan en la Figura 3, mientras que para la simulación Post - Place & Route sólo se observará el retardo clock-out, en la Figura 4, siendo de aproximadamente 6.75ns.

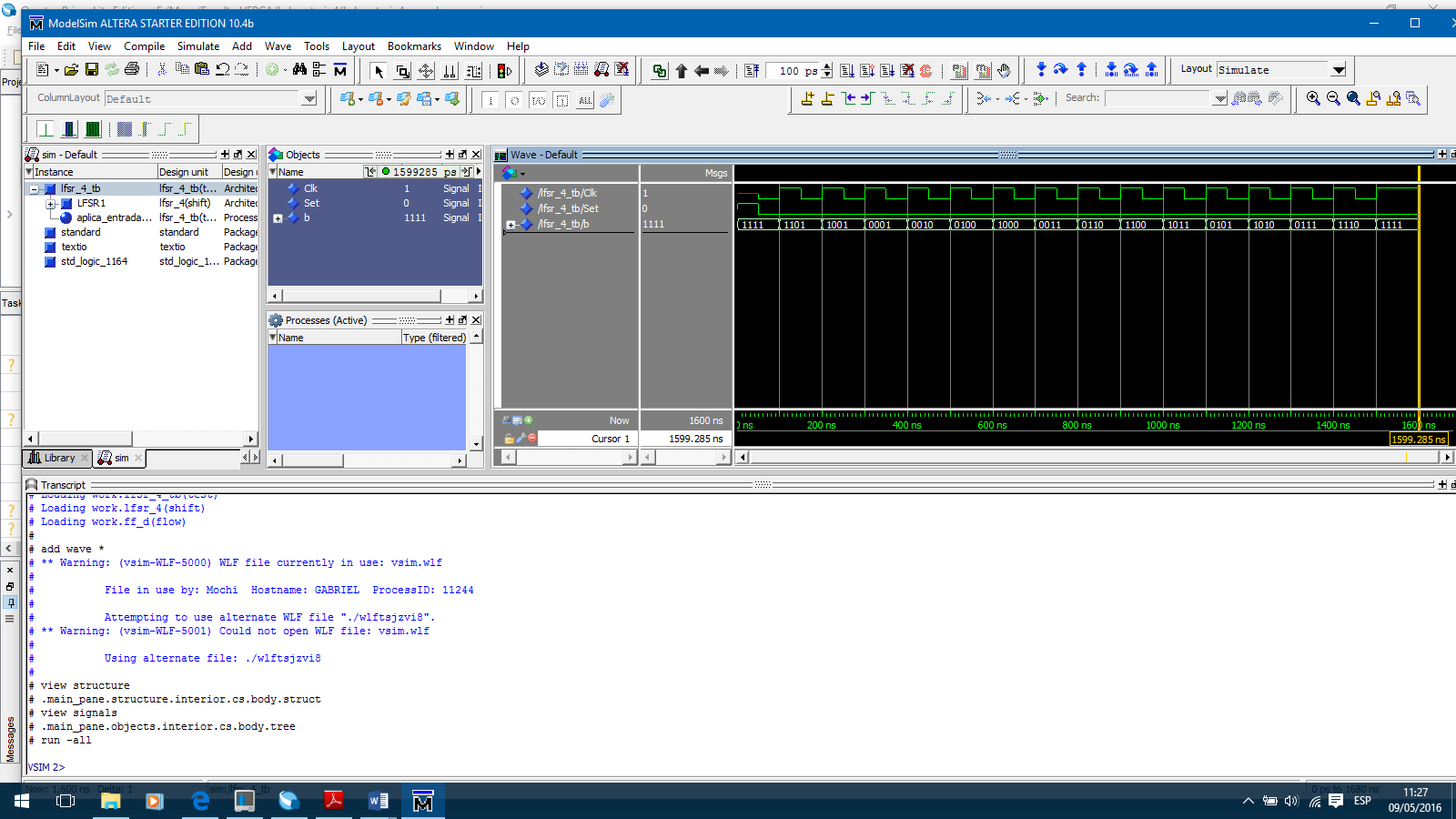


Figura 3

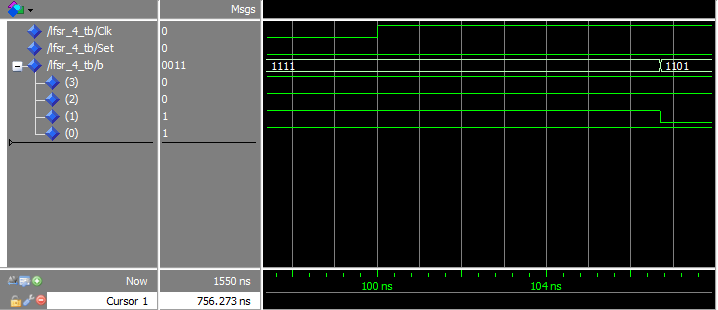


Figura 4

## Parte B

### Consigna

El Kit de desarrollo DE2-115 tiene un oscilador de 50MHz como entrada de reloj al FPGA. Realice un divisor de frecuencias para obtener las siguientes frecuencias: 0.1Hz, 0.5Hz, 1Hz, 2Hz y 5Hz.

* Describa en VHDL el divisor de frecuencia, usando constantes para los diferentes valores máximos del divisor.
* Como selector de frecuencia de salida use las llaves (switches) disponibles en el board.
* Sincronice las entradas asincrónicas con la frecuencia del reloj de entrada al FPGA.
* Utilice el módulo de conversor BCD-7Segmentos del laboratorio anterior para mostrar en los display 7 segmentos la frecuencia seleccionada por la llave respectiva (no hace falta usar el punto; por ejemplo para 0.5 muestre 05).
* Para cada combinación de las llaves (switches), la frecuencia seleccionada deberá excitar el LED\_0 del board
* Compruebe el funcionamiento funcional del código VHDL con un Test Bench. Analice que pasa con el tiempo real y el tiempo de simulación cuando simule su circuito.
* Escriba el correspondiente archivo de restricciones (constraints file). Recuerde detallar en el constraint file la frecuencia de trabajo de su sistema.
* Compruebe la correcta asignación de la señales de entrada/salida con los respectivos I/O pads del FPGA revisando el PAD report.
* Genere el bitstream y configure el FPGA.
* Compruebe el correcto funcionamiento del sistema al cambiar las llaves seleccionadoras de frecuencia como así también la correcta indicación del valor de frecuencia en los display 7 segmentos respectivos.

### Resolución

Se procedió a generar un divisor de frecuencia utilizando un contador que alcanzado cierto valor (prescaler) se reteara y conmutara la salida, este valor tendría que ser coincidente temporalmente con el semiperiodo de salida deseado.

Para ello se definieron las constantes a partir de la siguiente ecuación.

Se optó por dejar la frecuencia de entrada al divisor variable con el deseo de hacer un componente versátil sumado a que en la parte C del laboratorio se trabajaría con una frecuencia de entrada distinta, ello significaría que la cantidad de bits utilizada por el prescaler y el contador debería ser variable, o en su defecto colocar un techo a la frecuencia de entrada; se optó por la primera opción y para el cálculo de bits necesarios se implementó una función logaritmo base 2 y se utilizó la siguiente ecuación.

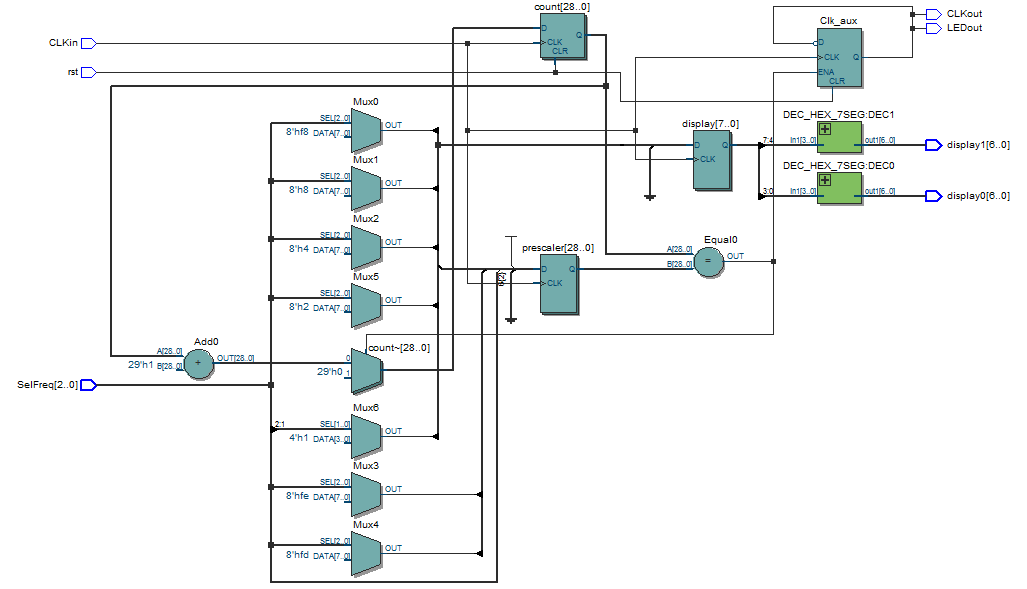
Por otro lado, se agregó un reloj de salida extra para excitar un LED de la placa para poder observar su correcto funcionamiento. La frecuencia seleccionada por los switch es mostrada mediante dos display 7 segmentos, para ello se utilizó el modelo de decodificador hexadecimal a 7 segmentos descriptos en el Laboratorio 3.

Para la selección de la frecuencia de salida, se utilizaron 3 switches de la placa, obteniéndose la Tabla 2

Tabla

|  |  |  |
| --- | --- | --- |
| Switches | Frecuencia | Frecuencias en Test-Bench |
| 000 | 0,1 Hz | 1MHz |
| 001 | 0,5 Hz | 500KHz |
| 010 | 2 Hz | 100KHz |
| 011 | 5 Hz | 5Hz |
| 1XX | 1 Hz | 1Hz |

Mediante la herramienta RTL View se obtuvo un esquema del hardware sintetizado, el cual se observa en la Figura 5.



Figura

La asignación de pines que se usó para implementar el divisor de frecuencia en la placa se observa en la Tabla 3.

Tabla

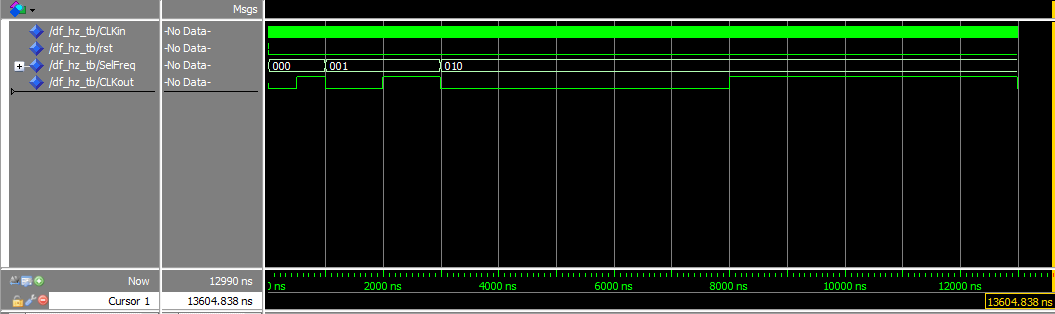
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| To | Direction | Location | I/O Bank | VREF Group |
| CLKin | Input | PIN\_Y2 | 2 | B2\_N0 |
| CLKout | Output | PIN\_AE23 | 4 | B4\_N0 |
| display0[6] | Output | PIN\_H22 | 6 | B6\_N0 |
| display0[5] | Output | PIN\_J22 | 6 | B6\_N0 |
| display0[4] | Output | PIN\_L25 | 6 | B6\_N1 |
| display0[3] | Output | PIN\_L26 | 6 | B6\_N1 |
| display0[2] | Output | PIN\_E17 | 7 | B7\_N2 |
| display0[1] | Output | PIN\_F22 | 7 | B7\_N0 |
| display0[0] | Output | PIN\_G18 | 7 | B7\_N2 |
| display1[6] | Output | PIN\_U24 | 5 | B5\_N0 |
| display1[5] | Output | PIN\_U23 | 5 | B5\_N1 |
| display1[4] | Output | PIN\_W25 | 5 | B5\_N1 |
| display1[3] | Output | PIN\_W22 | 5 | B5\_N0 |
| display1[2] | Output | PIN\_W21 | 5 | B5\_N1 |
| display1[1] | Output | PIN\_Y22 | 5 | B5\_N0 |
| display1[0] | Output | PIN\_M24 | 6 | B6\_N2 |
| LEDout | Output | PIN\_E21 | 7 | B7\_N0 |
| rst | Input | PIN\_AD27 | 5 | B5\_N2 |
| SelFreq[2] | Input | PIN\_AC27 | 5 | B5\_N2 |
| SelFreq[1] | Input | PIN\_AC28 | 5 | B5\_N2 |
| SelFreq[0] | Input | PIN\_AB28 | 5 | B5\_N1 |

El código generado se encuentra en el Apéndice B.

### Simulación y Test-Bench

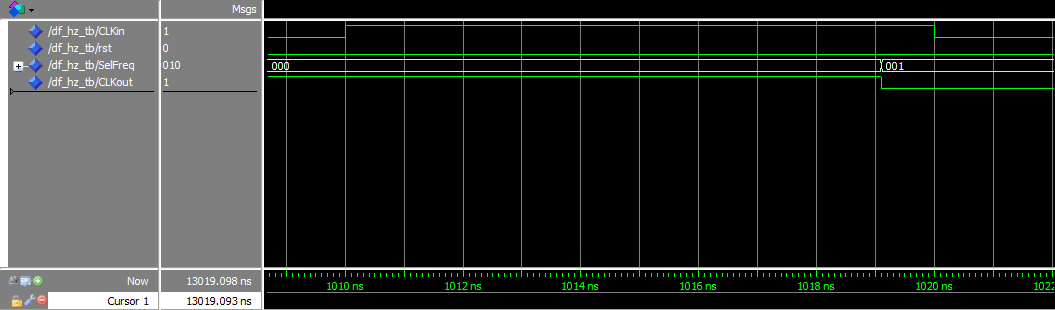
Cabe destacar que se agregaron unas frecuencias más a las pedidas por consigna, de 1 MHz, 500KHz y 100KHz. Esto se realizó ya que a fines de simulación, las demás frecuencias hacían este proceso muy lento.

Se realizó el código Test-Bench, el cual genera distintas combinaciones de los switches para comprobar que las frecuencias de salida fuesen coincidentes con las estipuladas en la Tabla 2, el código puede encontrarse en el Apéndice B. Los resultados de la simulación funcional se observan en la Figura 7.



Figura

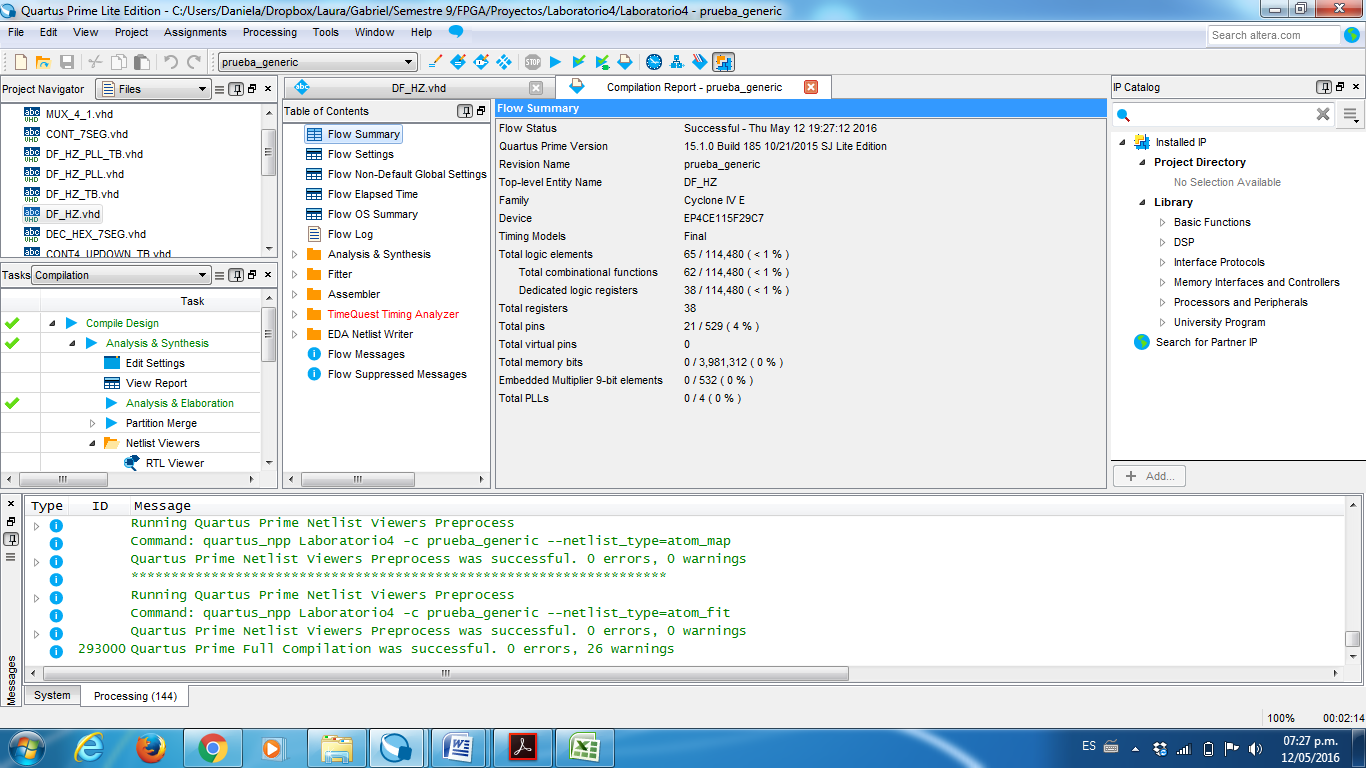
También fue realizada una simulación post-Place&Route para visualizar el retardo en la salida, visualizándose el mismo en la Figura 7. Como era esperable, un retardo de 9ns no afectaría la frecuencia de salida frente al período más corto (200ms para 5Hz).



Figura

### Reporte de área

Desde una de las herramientas del Quartus Se obtuvo el reporte de área (Figura 6) en el cual se puede analizar que no se alcanzó a utilizar ni el 1% de todos los elementos lógicos disponibles en el FPGA y el 4% de los pines.



Figura

## Parte C

### Consigna

Realizar un mismo divisor de frecuencia similar al de la Parte B, pero en este caso el reloj de 50MHz será la entrada de reloj de un PLL.

* Utilice la herramienta MegaWizard para configurar el PLL de modo de obtener a la salida la frecuencia mínima posible.
* Describa en VHDL un divisor de frecuencia para dividir la frecuencia de salida del PLL, de modo de tener a las salidas del divisor de frecuencia, las mismas frecuencias que en la Parte B. Agregue como salida la señal de lock del PLL, la que a su vez deberá excitar un led que indicará cuando el PLL está LOCK.
* Como selector de frecuencia de salida use las llaves (switches) disponibles en el board.
* Sincronice las entradas asincrónicas con la frecuencia del reloj de entrada al FPGA.
* Utilice el módulo de conversor BCD-7Segmentos del laboratorio anterior para mostrar en los display 7 segmentos la frecuencia seleccionada por la llave respectiva (no hace falta usar el punto; para 0.5 muestre 05).
* Para cada combinación de las llaves (switches), la frecuencia seleccionada deberá excitar el LED\_0 del board
* Compruebe el funcionamiento funcional del código VHDL con un Test Bench.
* Escriba el correspondiente archivo de restricciones (constraints file). Recuerde detallar en el constraint file la frecuencia de trabajo de su sistema.
* Compruebe la correcta asignación de la señales de entrada/salida con los respectivos I/O pads del FPGA revisando el PAD report.
* Genere el bitstream y configure el FPGA.
* Compruebe el correcto funcionamiento del sistema al cambiar las llaves seleccionadoras de frecuencia como así también la correcta indicación del valor de frecuencia en los display 7 segmentos respectivos.

### Resolución

Para realizar esta parte, se comenzó con la configuración del PLL, mediante el uso de la herramienta MegaWizard. En ésta, se configura la frecuencia de entrada al PLL en 50 MHz y se obtiene la menor frecuencia de salida mediante prueba y error en 1,2 KHz. A su vez, esta frecuencia será la entrada a un divisor de frecuencia (el mismo usado en la Parte B) para poder llegar a los valores de frecuencias deseados. El código del hardware descripto se encuentra en el Apéndice C.

La asignación de pines que se usó para implementar el divisor de frecuencia en la placa se observa en la Tabla 3.

Tabla

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| To | Direction | Location | I/O Bank | VREF Group |
| CLKin | Input | PIN\_Y2 | 2 | B2\_N0 |
| CLKout | Output | PIN\_AE23 | 4 | B4\_N0 |
| display0[6] | Output | PIN\_H22 | 6 | B6\_N0 |
| display0[5] | Output | PIN\_J22 | 6 | B6\_N0 |
| display0[4] | Output | PIN\_L25 | 6 | B6\_N1 |
| display0[3] | Output | PIN\_L26 | 6 | B6\_N1 |
| display0[2] | Output | PIN\_E17 | 7 | B7\_N2 |
| display0[1] | Output | PIN\_F22 | 7 | B7\_N0 |
| display0[0] | Output | PIN\_G18 | 7 | B7\_N2 |
| display1[6] | Output | PIN\_U24 | 5 | B5\_N0 |
| display1[5] | Output | PIN\_U23 | 5 | B5\_N1 |
| display1[4] | Output | PIN\_W25 | 5 | B5\_N1 |
| display1[3] | Output | PIN\_W22 | 5 | B5\_N0 |
| display1[2] | Output | PIN\_W21 | 5 | B5\_N1 |
| display1[1] | Output | PIN\_Y22 | 5 | B5\_N0 |
| display1[0] | Output | PIN\_M24 | 6 | B6\_N2 |
| LEDout | Output | PIN\_E21 | 7 | B7\_N0 |
| rst | Input | PIN\_AD27 | 5 | B5\_N2 |
| SelFreq[2] | Input | PIN\_AC27 | 5 | B5\_N2 |
| SelFreq[1] | Input | PIN\_AC28 | 5 | B5\_N2 |
| SelFreq[0] | Input | PIN\_AB28 | 5 | B5\_N1 |

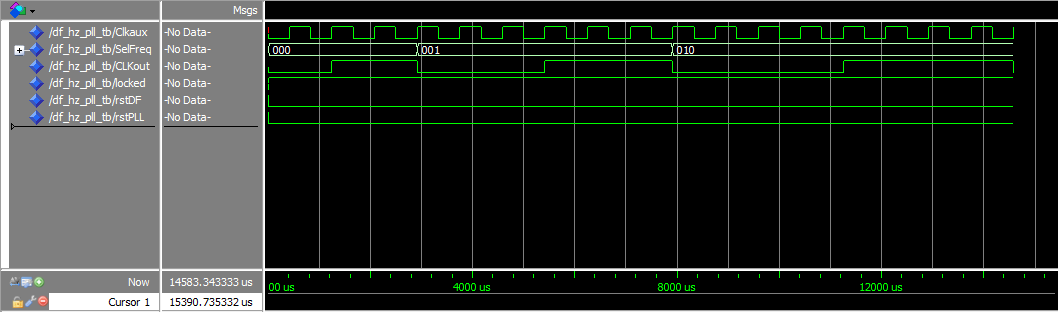
### Simulación y Test-Bench

Cabe destacar que se agregaron unas frecuencias más a las pedidas por consigna, de 600Hz, 400Hz y 300KHz. Esto se realizó ya que a fines de simulación, las demás frecuencias hacían este proceso muy lento.

Tabla

|  |  |
| --- | --- |
| Switches | Frecuencias en Test-Bench |
| 000 | 600Hz |
| 001 | 400Hz |
| 010 | 300Hz |

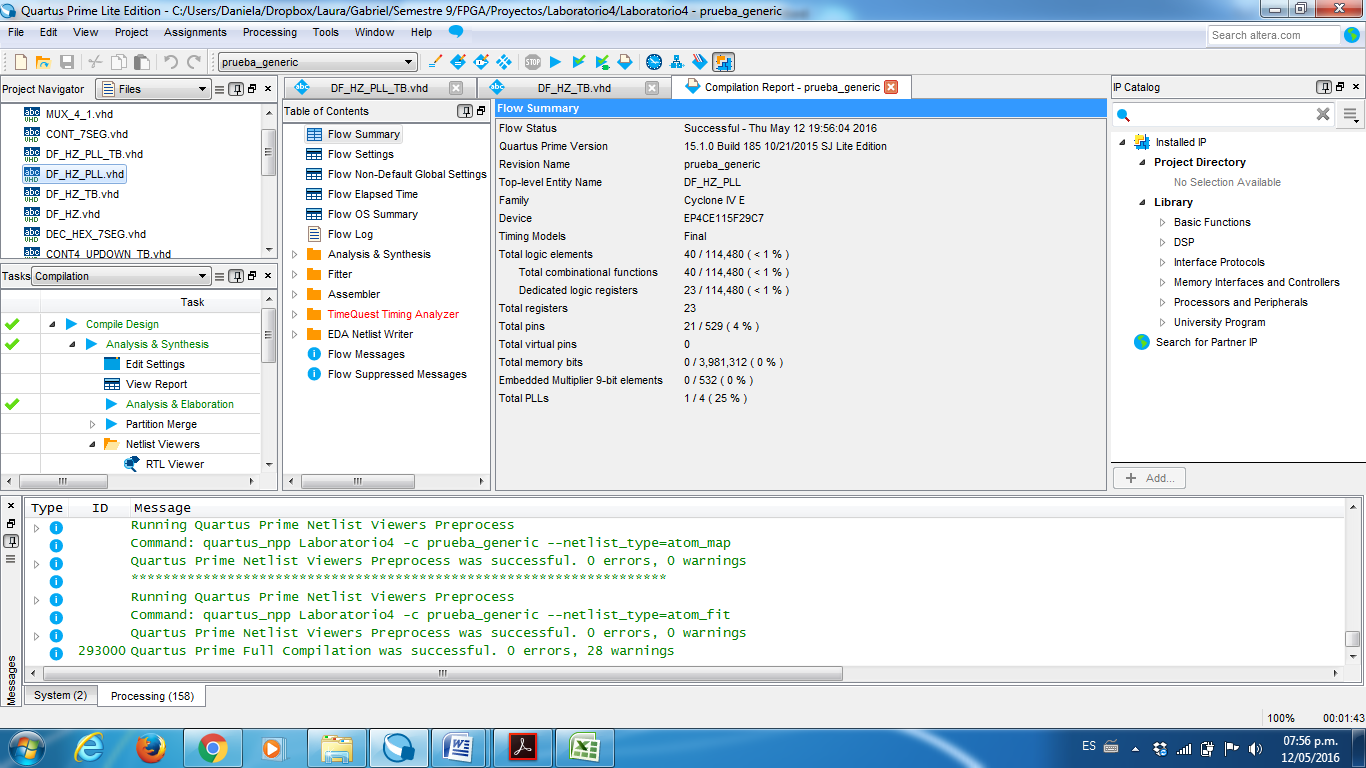
Se realizó el código Test-Bench, el cual genera distintas combinaciones de los switches para comprobar que las frecuencias de salida fuesen coincidentes con las estipuladas en la Tabla 5. Los resultados de la simulación funcional se observan en la Figura 9.



Figura

### Reporte de área

Desde una de las herramientas del Quartus Se obtuvo el reporte de área (Figura 10) en el cual se puede analizar que no se alcanzó a utilizar ni el 1% de todos los elementos lógicos disponibles en el FPGA y el 4% de los pines. Además, se observa que se utilizó 1 de los 4 PLLs disponibles.



Figura

## Parte D

### Consigna

Ahora está en condiciones de completar la Parte B del Laboratorio 3. Reemplace las ‘cajas negras’ de cada uno de los contadores por:

* Un contador LFSR (descripto en la Parte A del presente laboratorio)
* Un contador UP
* Un contador DOWN
* Un contador a su elección
* Describa el Top Level modulo con todos los componentes necesarios.
* Como señal de reloj para los contadores utilice la salida de 1Hz (un segundo) del divisor de frecuencias realizado en la Parte B de este laboratorio
* Utilice una señal de reset asincrónica.
* Sincronice las entradas asincrónicas con la frecuencia del reloj de entrada al FPGA.
* Dibuje un detallado diagrama en bloques del diseño que implemente.
* Compruebe el correcto funcionamiento del sistema descripto mediante la adecuada simulación del mismo.
* Investigue el timing report, y descubra cual es el camino más crítico de este diseño. Realice un dibujo esquemático del mismo, el que puede ser obtenido usando la herramienta correspondiente.
* Escriba un archivo de restricción (constraint file) donde especifique la frecuencia de funcionamiento de su diseño.
* Asigne los I/O pads respectivos.
* Compruebe la correcta asignación de la señales de entrada/salida con los respectivos I/O pads del FPGA revisando el reporte de entradas y salidas (Inputs and Outputs reports).
* Genere el modelo de simulación post-place and route (proceso Generate Post-Place & Route Simulation Model). Abrir el archivo \*.vho en el directorio del proyecto y ver su estructura. Opcional: abrir el archivo \*.sdf en el directorio del proyecto y comente con respecto a los tres valores que cada componente tiene en su instancia.
* Una vez comprobado el correcto funcionamiento en simulación, genere el archivo de configuración del FPGA.
* Configurar el Cyclone IV en el laboratorio.
* Compruebe al correcto funcionamiento del sistema implementado en el kit DE2-115.

### Resolución

Para realizar la práctica se debían visualizar la cuenta de 4 contadores distintos y tener la posibilidad, mediante los switches, de elegir cuál de ellos observar mediante los 7 segmentos. El primer contador es el LFSR implementado en la parte A, el segundo y tercero son contadores UP y DOWN respectivamente mientras que el último es un contador ascendente de paso 2.

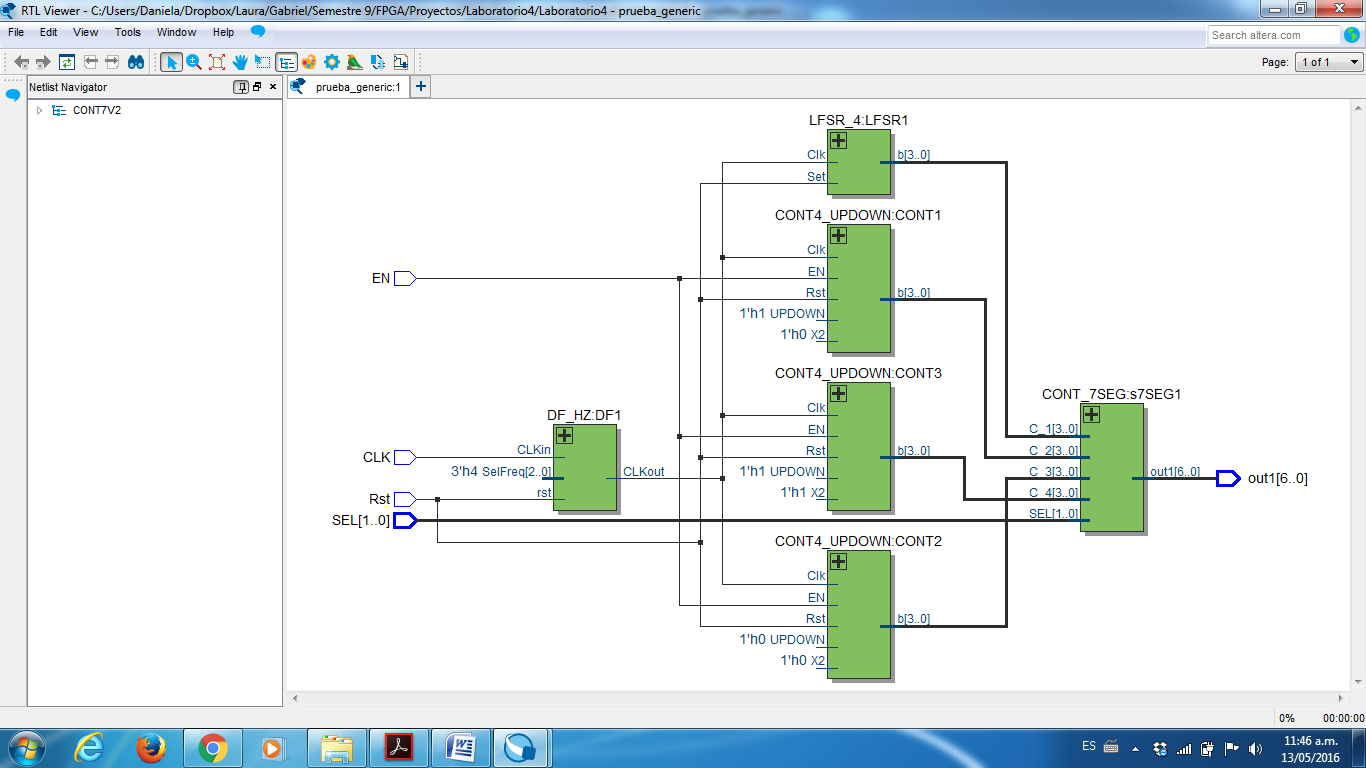
Para la realización de los contadores, se utilizó una variable auxiliar a la cual se le suma una unidad más el valor del X2 (activación del contador de paso 2) en caso de los contadores ascendentes, o resta una unidad en caso del contador descendente. Esta variable es la que luego va a ser mostrada por los 7 segmentos.

La asignación de pines que se usó para implementar el divisor de frecuencia en la placa se observa en la Tabla 6.

Tabla

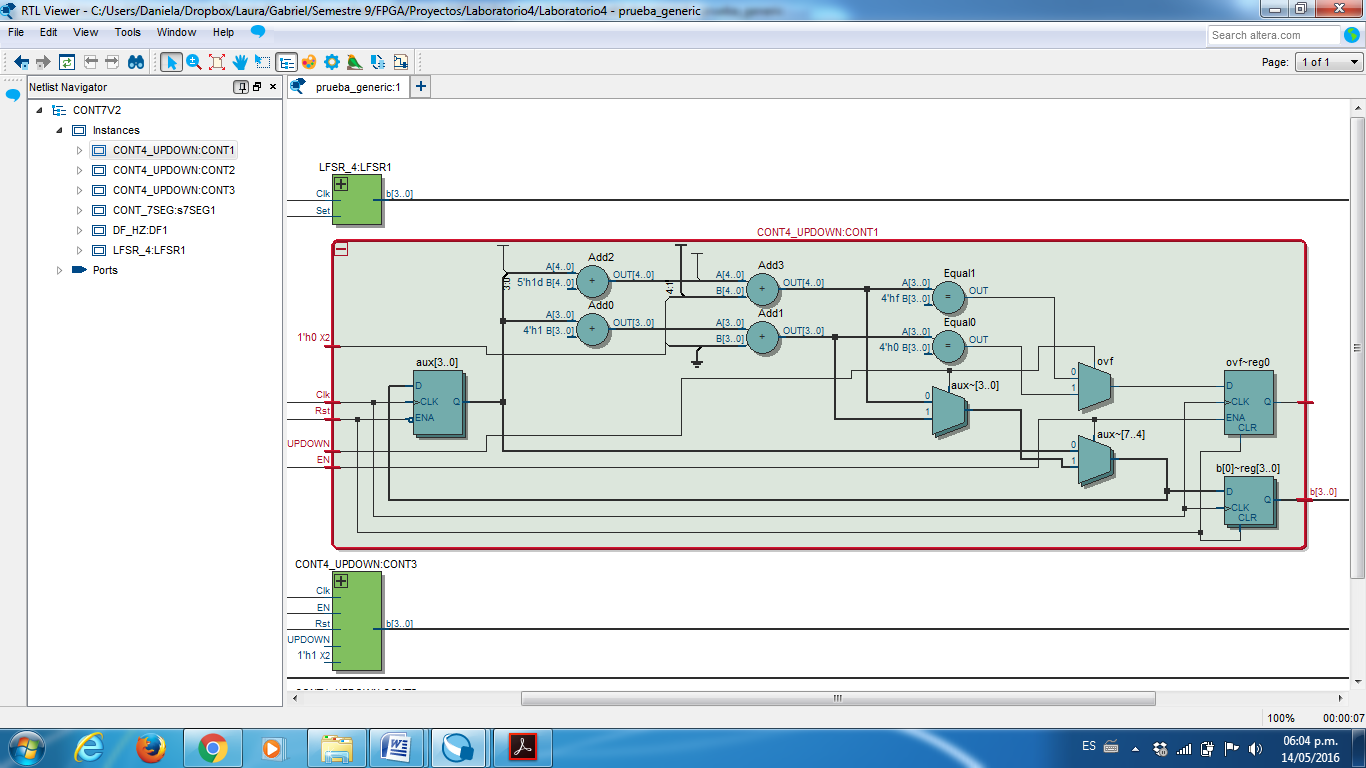
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| To | Direction | Location | I/O Bank | VREF Group |
| CLK | Input | PIN\_Y2 | 2 | B2\_N0 |
| EN | Input | PIN\_AC27 | 5 | B5\_N2 |
| out1[6] | Output | PIN\_G18 | 7 | B7\_N2 |
| out1[5] | Output | PIN\_F22 | 7 | B7\_N0 |
| out1[4] | Output | PIN\_E17 | 7 | B7\_N2 |
| out1[3] | Output | PIN\_L26 | 6 | B6\_N1 |
| out1[2] | Output | PIN\_L25 | 6 | B6\_N1 |
| out1[1] | Output | PIN\_J22 | 6 | B6\_N0 |
| out1[0] | Output | PIN\_H22 | 6 | B6\_N0 |
| Rst | Input | PIN\_AD27 | 5 | B5\_N2 |
| SEL[1] | Input | PIN\_AC28 | 5 | B5\_N2 |
| SEL[0] | Input | PIN\_AB28 | 5 | B5\_N1 |

El modelo de los 4 contadores sintetizados, junto al divisor de frecuencia y el conversor de 7 segmentos se observa en la Figura 11.



Figura

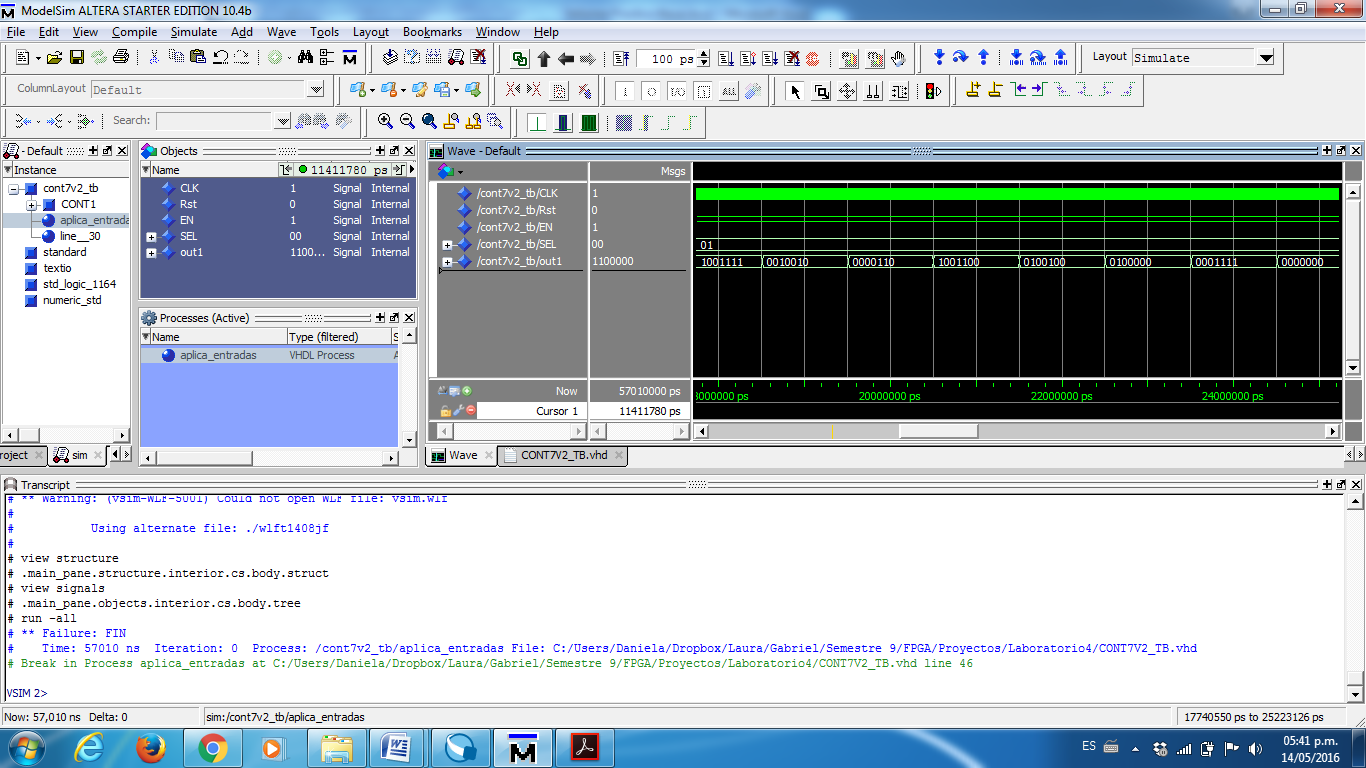
El funcionamiento del contador UPDOWN, se detalla en la Figura 12. La entrada UPDOWN va a determinar si se desea que el contador sea ascendente o descendente.



Figura

### Simulación y Test-Bench

Se realizó el código Test-Bench, en el cual se espera un tiempo necesario en cada combinación de los switches con el fin de comprobar que los contadores funcionaran correctamente. En la Figura 13 se puede observar la salida del contador descendente hacia el display 7 segmentos. Gracias a la Tabla 7, se observa que la cuenta numérica desciende en forma correcta.



Figura

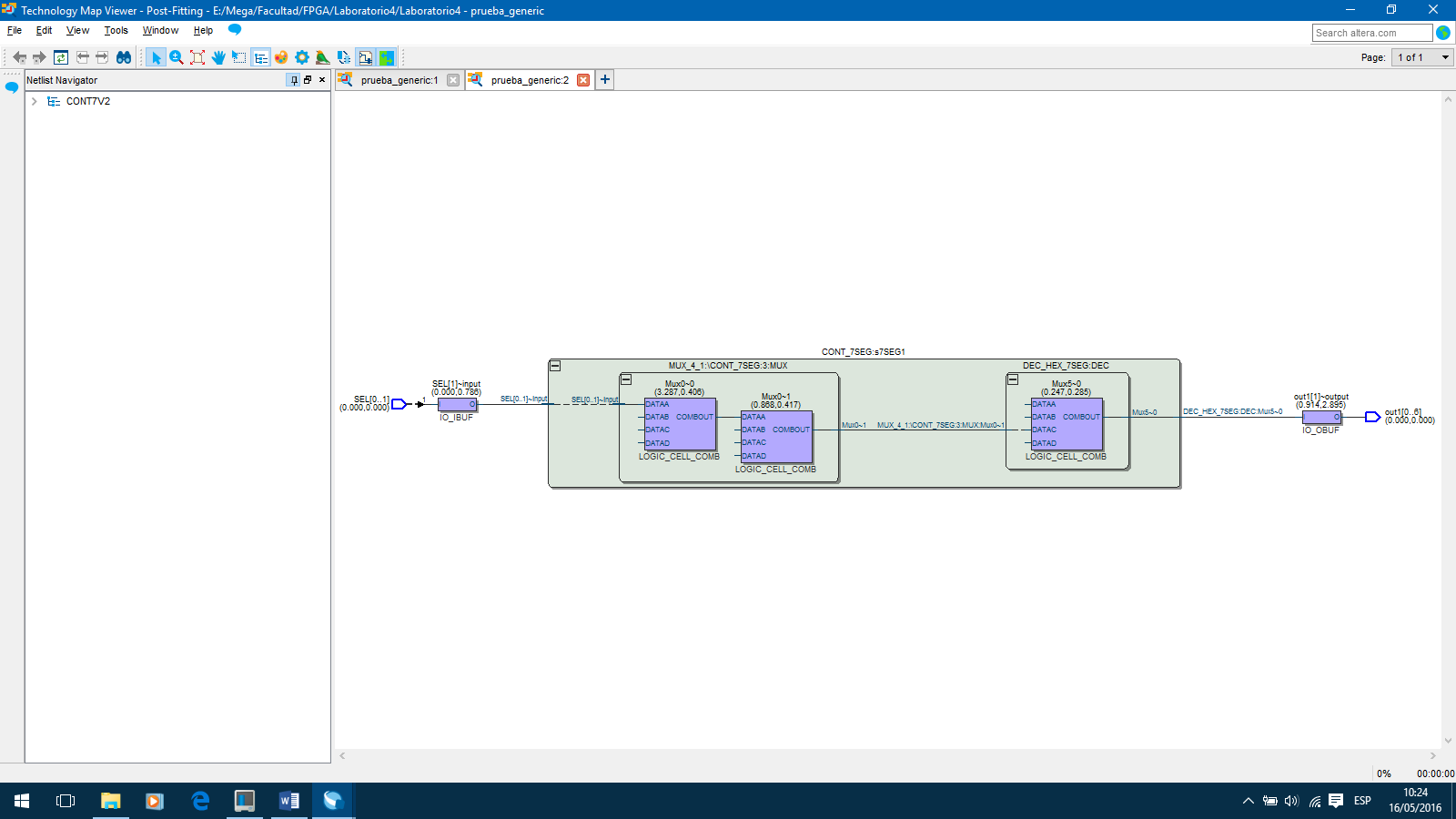
Tabla

|  |  |
| --- | --- |
| Código 7 Segmentos | Valor numérico |
| 0000001 | 0 |
| 1001111 | 1 |
| 0010010 | 2 |
| 0000110 | 3 |
| 1001100 | 4 |
| 0100100 | 5 |
| 0100000 | 6 |
| 0001111 | 7 |
| 0000000 | 8 |
| 0001100 | 9 |
| 0001000 | A |
| 1100000 | B |
| 0110001 | C |
| 1000010 | D |
| 0110000 | E |
| 0111000 | F |

Se utilizó la herramienta “Report Path” del TimeQuest Timing Analyzer para la obtención del retardo de propagación más largo. Este se encontraba entre la entrada SEL[1] y la salida out1[1] y su duración es de 10.105 nanosegundos. En la Tabla 8 se muestra como se compone este camino de forma detallada. Además, se obtuvo la localización del camino mediante la herramienta Technology Map Viewer de Quartus, que se puede observar en la Figura 14.

Tabla

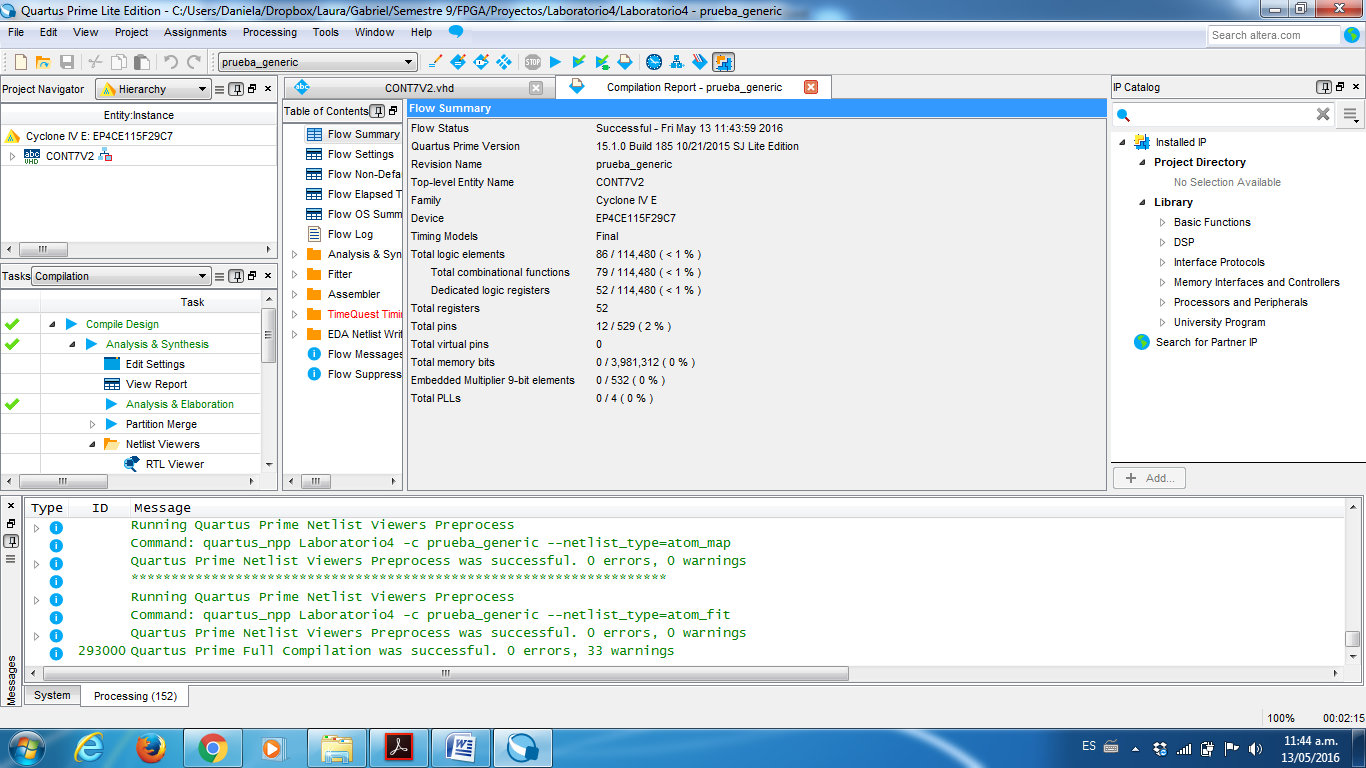
|  |  |  |
| --- | --- | --- |
| Delay | From Node | To Node |
| 10.105 | SEL[1] | out1[1] |
| 9.939 | SEL[1] | out1[0] |
| 9.930 | SEL[1] | out1[3] |
| 9.913 | SEL[1] | out1[2] |
| 9.910 | SEL[1] | out1[4] |
| 9.897 | SEL[1] | out1[1] |
| 9.871 | SEL[1] | out1[5] |
| 9.736 | SEL[0] | out1[1] |
| 9.733 | SEL[1] | out1[6] |
| 9.704 | SEL[1] | out1[3] |



Figura

### Reporte de área

Desde una de las herramientas del Quartus Se obtuvo el reporte de área (Figura 14) en el cual se puede analizar que no se alcanzó a utilizar ni el 1% de todos los elementos lógicos disponibles en el FPGA y el 2% de los pines.



Figura

# Problemas y soluciones

* Para el cálculo de bits necesarios para un determinado valor de prescaler se debía usar la función logaritmo, al no tener acceso a una función más directa se debió realizar un bloque function para implementarla.
* Cuando se simulaba el comportamiento del componente de la parte C, siendo las simulaciones bastante prolongadas, se pausaba la simulación y se corroboraba que sucediera adecuadamente. Sin embargo se pensó que el componente estaba mal descrito debido a que el periodo de la onda de salida no era el correcto. Finalmente se decidió por generar una salida ClockAux a fines de simulación para corroborar los cambios de la salida del PLL (reloj de 1200Hz) descubriéndose que el mismo, al ser reseteado el PLL, era inicializado en ‘1’. Se consideró que en la práctica esto no sería un problema debido a que solo afectaría al primer período luego del reset. Aunque a fines de simulación se optó por invertir esta señal para que se visualizara el correcto funcionamiento.
* En el componente CONT4\_UPDOWN, para implementar la función de la entrada X2 se decidió sumar dicha entrada al valor de la cuenta, sin embargo para ello se debía transformar el valor de la entrada al tipo unsigned. Al no haber una conversión directa del tipo std\_logic a unsigned se debió enmascarar el dato como std\_logic\_vector concatenándolo con una cadena vacía y así poder realizar el casteo correspondiente.
* Se agregaron constantes para frecuencias más altas y con ellas se realizaron los Test-Bench debido a los largos tiempos de simulación que suponía generar frecuencias bajas.

# Conclusión

Con la realización de esta práctica se aprendió a utilizar la herramienta MegaWizard para la generación de un sintetizador de frecuencia basado en PLL.

Además se puso en práctica la reutilización de componentes generados anteriormente. Esto implica comprender la importancia de realizar código que puede ser reutilizado en un futuro.

Por otro lado se aprendió a realizar la creación de componentes genéricos. Esto es muy útil ya que en más de una ocasión se debió usar componentes que cumplían las mismas funciones pero tenían características diferentes.

# Apéndices

## Apéndice A

### Códigos de Componentes

#### FF\_D.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Modelo de Flip Flop D

entity FF\_D is

port

(

-- Input ports

D : in std\_logic;

Clk : in std\_logic; --Reloj

Set : in std\_logic; --Seteo asincrónico

-- Output ports

Q : out std\_logic

);

end FF\_D;

architecture flow of FF\_D is

begin

ff\_d\_clk: process (Clk, Set)

begin

if (Set='1') then

Q<='1';

elsif (rising\_edge(Clk)) then

Q<=D;

end if;

end process;

end flow;

#### LFSR\_4.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--CONTADOR LINEAR FEEDBACK SHIFT REGISTER.

--Cuenta una secuencia pseudoaleatoria de 4 bits cíclica de 15 estados. Posee una entrada de seteo.

entity LFSR\_4 is

port

(

-- Input ports

Clk : in std\_logic; --Entrada de reloj

Set : in std\_logic; --Entrada de seteo (Se setea la salida en "1111" debido a que "0000" no es una salida válida en este contador.

-- Output ports

b : out std\_logic\_vector(3 downto 0)

);

end LFSR\_4;

architecture shift of LFSR\_4 is

component FF\_D is

port

(

-- Input ports

D : in std\_logic;

Clk : in std\_logic;

Set : in std\_logic;

-- Output ports

Q : out std\_logic

);

end component;

signal XOR1: std\_logic; --Salida de XOR

signal int: std\_logic\_vector(3 downto 0); --Conexiones internas del contador, sirven como salidas también.

begin

XOR1<=int(0) XOR int(3);

LFSR: for i in 3 downto 0 generate

i32: if (i<4 and i>1) generate

bit32: FF\_D port map(D=>int(i-1), Clk=>Clk, Set=>Set,Q=>int(i)); --Cuarto y tercer bit de salida

end generate i32;

i1: if(i=1) generate

bit1: FF\_D portmap(D=>XOR1,Clk=>Clk,Set=>Set,Q=>int(i)); --Segundo bit

end generate i1;

i0: if(i=0) generate

bit0: FF\_D portmap(D=>int(3),Clk=>Clk,Set=>Set,Q=>int(i)); --Primer bit

end generate i0;

end generate LFSR;

b<=int;

end shift;

### Test-Bench

#### LFSR\_4\_TB.vhd

library ieee;

use ieee.std\_logic\_1164.all;

entity LFSR\_4\_TB is

end LFSR\_4\_TB;

architecture test of LFSR\_4\_TB is

component LFSR\_4 is

port

(

-- Input ports

Clk : in std\_logic;

Set : in std\_logic;

-- Output ports

b : out std\_logic\_vector(3 downto 0)

);

end component;

signal Clk, Set : std\_logic;

signal b : std\_logic\_vector(3 downto 0);

begin

LFSR1: LFSR\_4 port map(

Clk=>Clk,

Set=>Set,

b=>b

);

aplica\_entradas: process

begin

Set <= '1';

wait for 50ns;

Set <= '0';

for i in 0 to 14 loop

Clk<='0';

wait for 50ns;

Clk<='1';

wait for 50ns;

end loop;

wait for 50ns;

wait;

end process;

end test;

## Apéndice B

### Códigos de Componentes

#### DF\_HZ.vhd

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

--Divisor de frecuencias, permite salidas de (0.1Hz, 0.5Hz, 1Hz, 2Hz y 5Hz), con una entrada de reloj por defecto de 50MHz.

--Es posible especificar la frecuencia de entrada en caso contrario.

entity DF\_HZ is

generic

(

freq: integer := 50000000

);

port(

CLKin: in std\_logic; --Reloj de entrada

rst: in std\_logic; --Reset asincrónico

SelFreq: in std\_logic\_vector(2 downto 0); --Selector de frecuencia ("000" => 0.1Hz, "001" => 0.5Hz, "010" => 2Hz, "011" => 5Hz, "1xx" => 1Hz)

CLKout: out std\_logic; --Reloj de salida

LEDout: out std\_logic; --Salida para LED (puede ser utilizada como salida de reloj auxiliar)

display1: out std\_logic\_vector(6 downto 0); --Salida 7 segmentos. Indicador de frecuencia de salida. Número más significativo

display0: out std\_logic\_vector(6 downto 0) --Salida 7 segmentos. Indicador de frecuencia de salida. Número menos significativo

);

end DF\_HZ;

architecture beh of DF\_HZ is

function Log2( input:integer ) return integer is --Función logaritmo base 2. Utilizada para determinar la cantidad mínima de bits para sintetizar todas las frecuencias.

variable temp,log:integer;

begin

temp:=input;

log:=0;

while (temp /= 0) loop

temp:=temp/2;

log:=log+1;

end loop;

return log;

end function log2;

constant PS\_01HZ: integer := freq\*5-1; --Constante prescaler de 0.1Hz

constant PS\_05HZ: integer := freq-1; --Constante prescaler de 0.5Hz

constant PS\_1HZ: integer := freq/2-1; --Constante prescaler de 1Hz

constant PS\_2HZ: integer := freq/4-1; --Constante prescaler de 2Hz

constant PS\_5HZ: integer := freq/10-1; --Constante prescaler de 5Hz

constant PS\_1MHZ: integer := freq/2000000-1;--Constante de 1MHz a fines de simulación

constant PS\_500KHZ: integer := freq/1000000-1;--Constante de 500KHz a fines de simulación

constant PS\_100KHZ: integer := freq/200000-1; --Constante de 100KHz a fines de simulación

constant PS\_600HZ: integer := freq/600-1; --Constante de 1KHz a fines de simulación

constant PS\_400HZ: integer := freq/400-1; --Constante de 1KHz a fines de simulación

constant PS\_300HZ: integer := freq/300-1; --Constante de 1KHz a fines de simulación

constant nbits: integer := Log2(PS\_01HZ);

signal prescaler,count: unsigned(nbits downto 0);

signal Clk\_aux: std\_logic;

component DEC\_HEX\_7SEG is

port(

in1 : in std\_logic\_vector(3 downto 0);

out1 : out std\_logic\_vector(6 downto 0)

);

end component;

signal display: std\_logic\_vector(7 downto 0);

begin

freq\_div: process (clkin,rst)

--Proceso de síntesis de clock de salida. Reset asincrónico y toggleo de salida.

begin

if(rst='1') then

Clk\_aux <= '0';

count <= (others => '0');

elsif rising\_edge(Clkin) then

if (count = prescaler) then

count <= (others=>'0');

Clk\_aux <= not Clk\_aux;

else

count <= count+"1";

end if;

end if;

end process freq\_div;

sinc\_entradas: process (SelFreq,clkin)

--Proceso de seteo de prescaler y actualización de 7 segmentos antes cambios de entrada.

begin

if rising\_edge(clkin) then

case SelFreq is

when "000" =>

prescaler <= to\_unsigned(PS\_01HZ,nbits+1);

display <= "00000001";

when "001" =>

prescaler <= to\_unsigned(PS\_05HZ,nbits+1);

display <= "00000101";

when "010" =>

prescaler <= to\_unsigned(PS\_1HZ,nbits+1);

display <= "00100000";

when "011" =>

prescaler <= to\_unsigned(PS\_2HZ,nbits+1);

display <= "01010000";

when others =>

prescaler <= to\_unsigned(PS\_5HZ,nbits+1);

display <= "00010000"; end case;

end if;

end process sinc\_entradas;

clkout<=clk\_aux;

LEDout<=clk\_aux;

DEC1:DEC\_HEX\_7SEG port map(in1=>display(7 downto 4),out1=>display1);

DEC0:DEC\_HEX\_7SEG port map(in1=>display(3 downto 0),out1=>display0);

end beh;

### Test-Bench

#### DF\_HZ\_TB.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Codigo Test-Bench para el componente DF\_HZ

entity DF\_HZ\_TB is

end DF\_HZ\_TB;

architecture test of DF\_HZ\_TB is

signal CLKin: std\_logic:='0';

signal rst: std\_logic;

signal SelFreq:std\_logic\_vector(2 downto 0);

signal CLKout: std\_logic;

component DF\_HZ is

generic

(

freq: integer := 50000000

);

port(

CLKin: in std\_logic;

rst: in std\_logic;

SelFreq: in std\_logic\_vector(2 downto 0);

CLKout: out std\_logic;

LEDout: out std\_logic;

display1: out std\_logic\_vector(6 downto 0);

display0: out std\_logic\_vector(6 downto 0)

);

end component;

begin

DF1: DF\_HZ generic map(freq=>50000000) port map(CLKin=>CLKin,rst=>rst,SelFreq=>SelFreq,CLKout=>CLKout,LEDout=>open,display1=>open,display0=>open);

CLKin<= not(CLKin) after 10ns;

aplica\_entradas: process

begin

rst <= '1';

SELFreq<="000";

wait for 10ns;

rst <= '0';

wait until falling\_edge(ClkOut); --Se espera un ciclo de la salida

SELFreq<="001";

wait until falling\_edge(Clkout);

SELFreq<="010";

wait until falling\_edge(CLKout);

assert(false) report "FIN" severity failure;

end process aplica\_entradas;

end test;

## Apéndice C

### Códigos de Componentes

#### DF\_HZ\_PLL.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Divisor de frecuencias, permite salidas de (0.1Hz, 0.5Hz, 1Hz, 2Hz y 5Hz), con una entrada de reloj por defecto de 50MHz.

--Utiliza un PLL para reducir la frecuencia de entrada de 50MHz a 1.2KHz

entity DF\_HZ\_PLL is

port(

clkin : in std\_LOGIC;

SelFreq : in std\_logic\_vector(2 downto 0);

rstPLL : in std\_LOGIC; --Reset del PLL

rstDF : in std\_LOGIC; --Reset del divisor de frecuencias

CLKout : out std\_logic; --Reloj de salida del divisor de frecuencias

CLKaux : out std\_logic; --Reloj de salida del PLL

LEDout : out std\_logic;

locked : out std\_LOGIC;

display1: out std\_logic\_vector(6 downto 0);

display0: out std\_logic\_vector(6 downto 0)

);

end DF\_HZ\_PLL;

architecture beh of DF\_HZ\_PLL is

component PLL is

port

(

areset : in STD\_LOGIC;

inclk0 : in STD\_LOGIC;

c0 : out STD\_LOGIC ;

locked : out STD\_LOGIC

);

end component;

component DF\_HZ is

generic

(

freq: integer := 50000000

);

port(

CLKin: in std\_logic;

rst: in std\_logic;

SelFreq: in std\_logic\_vector(2 downto 0);

CLKout: out std\_logic;

LEDout: out std\_logic;

display1: out std\_logic\_vector(6 downto 0);

display0: out std\_logic\_vector(6 downto 0)

);end component;

signal clkaux1,nclkaux1 : std\_logic;

begin

PLL1 : PLL port map (

areset => rstPLL,

inclk0 => clkin,

c0 => clkaux1,

locked => locked

);

DF1 : DF\_HZ generic map(freq=>1200) port map (

CLKin => nclkaux1,

rst => rstDF,

SelFreq => SelFreq,

CLKout => clkout,

LEDout => ledout,

display1 => display1,

display0 => display0

);

nclkaux1<=not(clkaux1);

Clkaux<=nClkaux1;

end beh;

### Test-Bench

#### DF\_HZ\_PLL\_TB.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Test-Bench de DF\_HZ\_PLL

entity DF\_HZ\_PLL\_TB is

end DF\_HZ\_PLL\_TB;

architecture test of DF\_HZ\_PLL\_TB is

signal CLKin: std\_logic:='0';

signal SelFreq:std\_logic\_vector(2 downto 0);

signal CLKout,Clkaux: std\_logic;

signal locked: std\_LOGIC;

signal rstDF,rstPLL: std\_logic:='1';

component DF\_HZ\_PLL is

generic

(

freq: integer := 50000000

);

port(

clkin : in std\_LOGIC;

SelFreq : in std\_logic\_vector(2 downto 0);

rstPLL : in std\_LOGIC;

rstDF : in std\_LOGIC;

CLKaux : out std\_logic;

CLKout : out std\_logic;

LEDout : out std\_logic;

locked : out std\_LOGIC;

display1: out std\_logic\_vector(6 downto 0);

display0: out std\_logic\_vector(6 downto 0)

);

end component;

begin

DFPLL1: DF\_HZ\_PLL port map(clkin=>clkin,SelFreq=>SelFreq,CLKout=>CLKout,locked=>locked,

LEDout=>open,display0=>open,display1=>open,rstDF=>rstDF,

rstPLL=>rstPLL,Clkaux=>Clkaux);

CLKin<= not(CLKin) after 10ns;

aplica\_entradas: process

begin

SELFreq<="000";

rstPLL<='0';

rstDF<='0';

wait until falling\_edge(Clkout); --Se espera un período de salida

SELFreq<="001";

wait until falling\_edge(Clkout);

SELFreq<="010";

wait until falling\_edge(Clkout);

report "FIN" severity failure;

end process aplica\_entradas;

end test;

## Apéndice D

### Códigos de Componentes

#### CONT7V2.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Contador LFSR/UP/DOWN/UP (X2) con entrada selectora y frecuencia de salida de 1Hz, se debe ingresar con 50MHz.

--Posee entrada de habilitación y reseteo.

entity CONT7V2 is

port

(

-- Input ports

CLK : in std\_logic; --Reloj de entrada

Rst : in std\_logic; --Reset asincrónico

EN : in std\_logic; --Entrada de habilitación (no afecta a LFSR)

SEL : in std\_logic\_vector(1 downto 0); --Entrada selectora

-- Output ports

out1 : out std\_logic\_vector(6 downto 0)

);

end CONT7V2;

architecture beh of CONT7V2 is

component CONT\_7SEG is

port

(

-- Input ports

C\_1 : in std\_logic\_vector(3 downto 0);

C\_2 : in std\_logic\_vector(3 downto 0);

C\_3 : in std\_logic\_vector(3 downto 0);

C\_4 : in std\_logic\_vector(3 downto 0);

SEL : in std\_logic\_vector(1 downto 0);

-- Output ports

out1 : out std\_logic\_vector(6 downto 0)

);

end component;

component LFSR\_4 is

port(

Clk : in std\_logic;

Set : in std\_logic;

b : out std\_logic\_vector(3 downto 0)

);

end component;

component CONT4\_UPDOWN is

port

(

-- Input ports

Clk : in std\_logic;

UPDOWN: in std\_logic;

EN : in std\_logic;

X2 : in std\_logic;

Rst : in std\_logic;

-- Output ports

b : out std\_logic\_vector(3 downto 0);

ovf: out std\_logic

);

end component;

component DF\_HZ is

generic

(

freq: integer := 50000000

);

port(

CLKin: in std\_logic;

rst: in std\_logic;

SelFreq: in std\_logic\_vector(2 downto 0);

CLKout: out std\_logic;

LEDout: out std\_logic;

display1: out std\_logic\_vector(6 downto 0);

display0: out std\_logic\_vector(6 downto 0)

);

end component;

signal lfsraux: std\_logic\_vector(3 downto 0);

signal cont1aux,cont2aux,cont3aux: std\_logic\_vector(3 downto 0);

signal Clkaux: std\_logic;

begin

s7SEG1 : CONT\_7SEG port map(

C\_1 => lfsraux,

C\_2 => cont1aux,

C\_3 => cont2aux,

C\_4 => cont3aux,

SEL => SEL,

out1 => out1

);

LFSR1 : LFSR\_4 port map(

Clk => Clkaux,

Set => Rst,

b => lfsraux

);

CONT1 : CONT4\_UPDOWN port map( --Contador UP

Clk => Clkaux,

UPDOWN => '1',

EN => EN,

X2 => '0',

Rst => Rst,

b => cont1aux,

ovf => open

);

CONT2 : CONT4\_UPDOWN port map( --Contador DOWN

Clk => Clkaux,

UPDOWN => '0',

EN => EN,

X2 => '0',

Rst => Rst,

b => cont2aux,

ovf => open

);

CONT3 : CONT4\_UPDOWN port map( --Contador UP 2X

Clk => Clkaux,

UPDOWN => '1',

EN => EN,

X2 => '1',

Rst => Rst,

b => cont3aux,

ovf => open

);

DF1 : DF\_HZ port map(

CLKin => CLK,

rst => Rst,

SelFreq => "100",

CLKout => Clkaux,

LEDout => open,

display1 => open,

display0 => open

);

end beh;

### Test-Bench

#### CONT7V2\_TB.vhd

library ieee;

use ieee.std\_logic\_1164.all;

--Test-Bench de CONT7V2

entity CONT7V2\_TB is

end CONT7V2\_TB;

architecture test of CONT7V2\_TB is

component CONT7V2 is

port

(

-- Input ports

CLK : in std\_logic;

Rst : in std\_logic;

EN : in std\_logic;

SEL : in std\_logic\_vector(1 downto 0);

-- Output ports

out1 : out std\_logic\_vector(6 downto 0)

);

end component;

signal CLK,Rst,EN : std\_logic:='0';

signal SEL : std\_logic\_vector(1 downto 0);

signal out1 : std\_logic\_vector(6 downto 0);

begin

CONT1: CONT7V2 port map(CLK=>CLK,Rst=>Rst,EN=>EN,SEL=>SEL,out1=>out1);

CLK<=not(CLK) after 10ns;

aplica\_entradas: process

begin

Rst<='1';

SEL<="00";

wait for 1.01us;

Rst<='0';

EN<='1';

wait for 16us;

SEL<="01";

wait for 16us;

SEL<="10";

wait for 16us;

SEL<="11";

wait for 8us;

assert(false) report "FIN" severity failure;

end process aplica\_entradas;

end test;

# Presupuesto

Detalle de horas trabajadas

|  |  |
| --- | --- |
| Tarea | Tiempo [Horas] |
| Diseño y desarrollo de Parte A | 1 |
| Diseño y desarrollo de Parte B | 2 |
| Diseño y desarrollo de Parte C | 1 |
| Diseño y desarrollo de Parte D | 0.5 |
| Diseño y desarrollo de Test-Bench de Parte A | 1 |
| Diseño y desarrollo de Test-Bench de Parte B | 2 |
| Diseño y desarrollo de Test-Bench de Parte C | 3 |
| Diseño y desarrollo de Test-Bench de Parte D | 0.5 |
| Programación y testeo de Parte B | 0.5 |
| Programación y testeo de Parte C | 1 |
| Programación y testeo de Parte D | 0.5 |
| Informe | 24 |
|  | 35 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ítem** | **Cantidad** | **Costo unitario [U$D]** | **Subtotal[U$D]** | **Costo total [U$D]** |
| Costo de diseño |  |  |  | 350,00 |
| Personal | 35 | 10,00 | 350,00 |  |
| Costo de materiales |  |  |  | 118,00 |
| Placa de desarrollo DE2-115 amortizada | 1 | 68,00 | 68,00 |  |
| Equipo amortizado | 1 | 50,00 | 50,00 |  |
|  |  |  |  |  |
|  |  |  | Total (S/Imp): | **468,00** |
|  |  |  | Total (C/Imp): | **567,00** |

Para la realización del proyecto se debe abonar el 35% (treinta y cinco) del costo total antes de dar comienzo con el mismo, mientras que el saldo restante se podrá cancelar mediante un plan de pago a convenir.

El pago puede ser realizado en dólares estadounidenses o en pesos argentinos tomando la cotización del dólar del día en que se efectúe el pago.

El presente presupuesto tiene un plazo de validez de 15 (quince) días desde que se realiza la entrega al cliente.