

FUNDACIÓN FULGOR
CURSO DE DISEÑO DIGITAL AVANZADO



PROGRAMA DE COMUNICACIONES
Trabajo Práctico N°4
VERILOG

ALUMNO : Martinez, Luciano Micael

DOCENTES : Dr. Ing. Pola, Luis Ariel
Ing. Leguizamón, Santiago

CÓRDOBA, ARGENTINA
18 de Julio de 2024



CONTENIDO

1. Introducción	2
2. Marco Teórico	2
2.1. Descripción a nivel de RTL en Verilog	2
2.2. Modelado Jerárquico	2
2.3. Bloques en Verilog	2
2.4. Tipos de Datos en Verilog	3
3. Enunciado	4
3.1. Ejercicio 1	4
3.1.1. Modo Shift Register	4
3.1.2. Modo Flash	4
3.2. Ejercicio 2	5
4. Actividad Práctica	6
4.1. Módulos de Diseño	6
4.1.1. Módulo counter	7
4.1.2. Módulos de secuencias	8
4.1.3. Módulo top	9
4.2. Simulación en Testbench	10
4.2.1. Simulación	11
4.3. Módulos de VIO e ILA	14
4.4. Pruebas en la FPGA remota	15
4.4.1. Pruebas con la VIO	16
4.4.2. Pruebas con la ILA	19
5. Conclusiones	20



1. Introducción

El presente informe describe la implementación de diversos controladores de LEDs en una FPGA utilizando el lenguaje de descripción de hardware Verilog. El objetivo principal es desarrollar y evaluar la funcionalidad de los sistemas de control de LEDs, aplicando los conceptos fundamentales de Verilog y diseño digital.

Este proyecto tiene como meta aprender a utilizar la herramienta Vivado de Xilinx, con el fin de adquirir experiencia en el trabajo con una FPGA remota a través de los módulos Virtual Input/Output (VIO) y Integrated Logic Analyzer (ILA), que permiten la interacción y el monitoreo de señales en tiempo real.

Los siguientes apartados detallan el proceso de diseño, la implementación en Verilog, la simulación con Testbench, así como la implementación en la FPGA remota.

2. Marco Teórico

2.1. Descripción a nivel de RTL en Verilog

Verilog es un lenguaje de descripción de hardware (HDL) de alto nivel que permite modelar sistemas digitales a nivel de Register Transfer Level (RTL). Facilita la descripción estructural y comportamental de circuitos electrónicos mediante la especificación de interacciones entre registros y su lógica de interconexión.

2.2. Modelado Jerárquico

El modelado jerárquico en Verilog organiza los módulos del sistema como instancias de otros módulos de manera estructurada, facilitando la reutilización y comprensión del diseño. El módulo de nivel superior, denominado 'top', contiene las instancias de módulos inferiores.

2.3. Bloques en Verilog

Verilog utiliza diferentes bloques para describir el comportamiento del hardware:

- **Initial:** El bloque 'initial' se utiliza para inicializar variables o configurar el estado inicial del sistema. Se ejecuta una sola vez al inicio de la simulación.
- **Always:** El bloque 'always' se usa para describir comportamientos secuenciales y combinacionales del sistema.
 1. **Secuencial (always @ (posedge clk)):** Describe comportamientos que ocurren en cada transición de flanco de un clock (posedge clk). Se usa para modelar registros y sincronización.
 2. **Combinacional (always @ (*)):** Describe comportamientos que dependen de cambios en las entradas (*). No está asociado con un clock y permite describir lógica combinacional.



2.4. Tipos de Datos en Verilog

A continuación se presentan los tipos de datos principales en Verilog y cómo se utilizan:

- **Wire:** Representa conexiones físicas entre componentes del circuito. No almacena valores internamente y se usa para interconectar señales.
- **Reg:** Permite el almacenamiento temporal de valores. Puede inferir un dispositivo de almacenamiento físico (como un Flip-Flop) cuando se asocia a una señal de clock ('posedge clk') dentro de un bloque 'always', permitiendo almacenar datos entre ciclos de clock.
- **Integer:** Representa números enteros de tamaño fijo. Se usa para operaciones aritméticas y almacenamiento de valores numéricos enteros.
- **Time:** Representa unidades de tiempo. Se utiliza para modelar retardos y sincronizaciones temporales en el diseño.
- **Constantes:** Valores constantes que no cambian durante la simulación. Se expresan en formatos como decimal, binario, octal o hexadecimal, y se utilizan para definir parámetros fijos.

Estos conceptos son fundamentales para la descripción precisa y efectiva del hardware en Verilog, garantizando la correcta implementación y simulación del diseño digital.



3. Enunciado

3.1. Ejercicio 1

Empleando los conceptos aprendidos de Verilog, implementar un controlador de LEDs en la FPGA tomando como referencia el diseño mostrado en la Figura 1. Los nombres en **ROJO** son puertos.

- `i_reset` es el reset del sistema, el cual pone a cero el contador e inicializa el shift register (SR) con algún valor de salida en 1 y el bloque Flash (FS) con todos los valores de salida en 1. Además, el reset es asíncrono y el pulsador es normal-cerrado.
- `i_sw[0]` controla el enable (1) del contador. En estado (0) todo se detiene sin alterar el estado actual del contador y del SR/FS.
- El pulsador `i_btn[0]` elige el modo de trabajo de los LEDs RGB. Cada vez que se pulsa el modo cambia entre SR o FS.

3.1.1. Modo Shift Register

- Se desplaza únicamente cuando el contador llegó a algún límite R0-R3 seleccionado a través del `i_sw[2:1]`.
- La dirección de desplazamiento se selecciona con `i_sw[3]`.

3.1.2. Modo Flash

- La salida cambia de estado cada vez que se llega al límite el contador. Ejemplo, el contador llega a R0 la salida está en 1, cuando el contador vuelve a llegar a R0 la salida cambia a 0.

La elección del límite se puede realizar en cualquier momento del funcionamiento.

- Los pulsadores `i_btn[3:1]` eligen el color de salida de los RGB. Es decir, cada vez que se pulsa alguno de los botones se pasa a rojo, verde o azul. Si se aprieta más de una vez el mismo pulsador, el LED debe permanecer en el mismo color.
- Además, se debe encender un LED (o LEDs [3:0]) según el pulsador que se haya apretado.

Nota 1: Tener en cuenta que el pulsador `i_btn[3:0]` regresa al estado inicial siempre que se suelta.



Nota 2: Asumir en principio que las entradas de pulsadores y llaves no requieren de sincronización ni circuito anti-rebotes (se consideran síncronas).

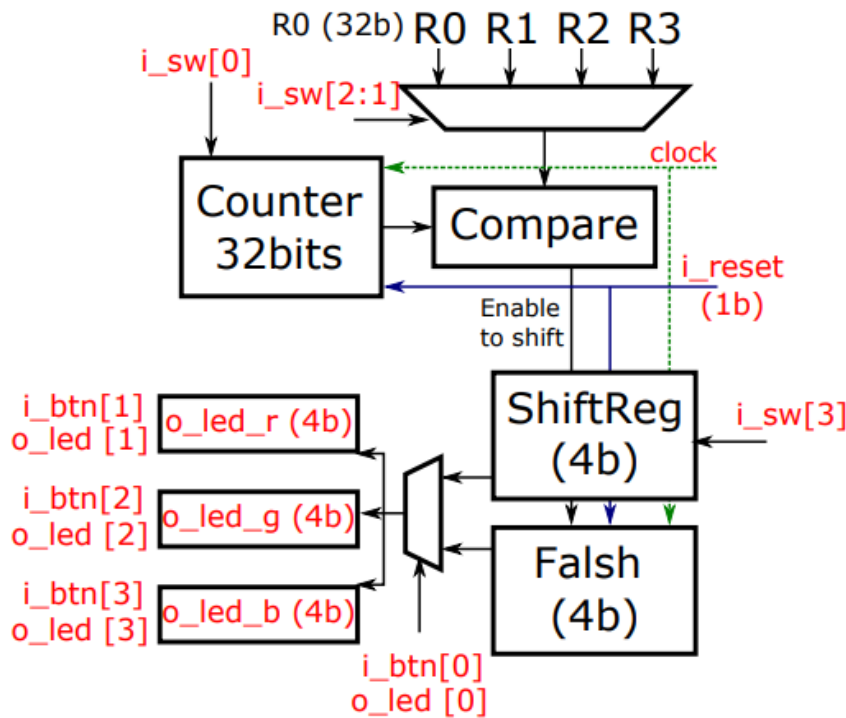


Figura 1: Diagrama en bloques de control de leds.

3.2. Ejercicio 2

Por último, agregar un tercer modo que prende dos leds y los desplaza hacia el centro o del centro hacia afuera.

- En este modo en el estado inicial se prenden en cualquier color los leds 0 y 3 o 1 y 2 (Ejemplo o led r[0] y o led r[3]).
- El puerto i_sw[3] selecciona la dirección de rotación. Ejemplo: (0,3)on → (1,2)on → all off ó (1,2)on → (0,3)on → all off.
- El puerto i_btn[0] debe cambiar entre las tres funciones, es decir por cada vez que se aprieta debe aparecer una secuencia nueva de leds.

En la Figura 2 se muestra el diagrama en bloques modificado.

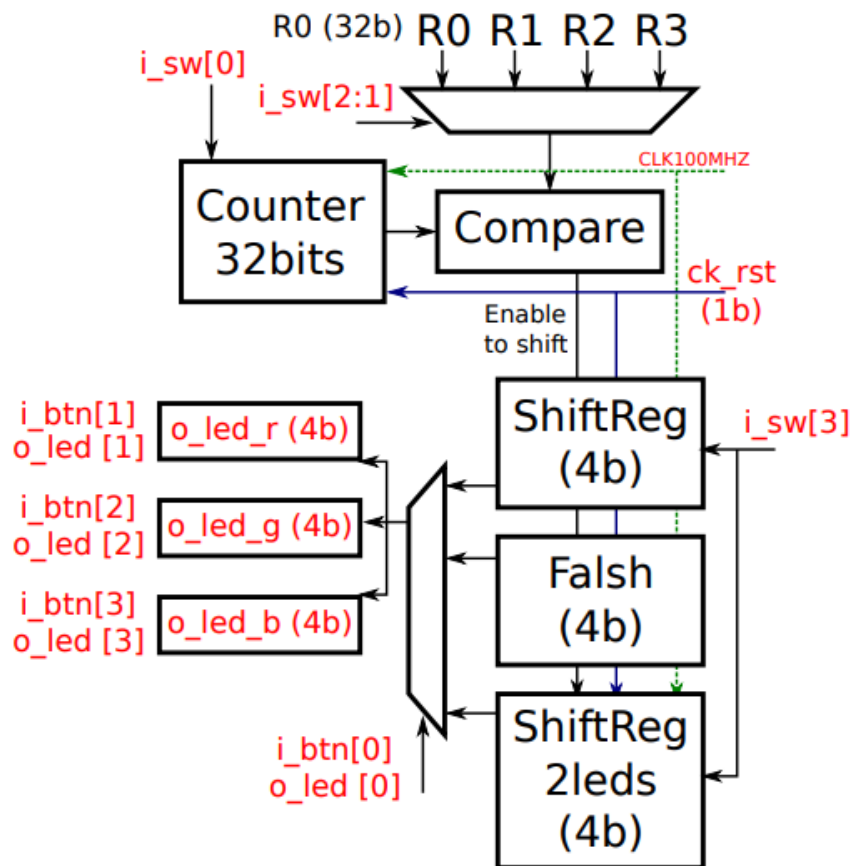


Figura 2: Diagrama en bloques de control de leds modificado.

4. Actividad Práctica

4.1. Módulos de Diseño

La descripción del sistema propuesto incluye la jerarquía que se muestra en la Figura 3. En esta instancia, los distintos bloques funcionales están instanciados en el módulo top, donde se realiza la conexión entre ellos.



Figura 3: Jerarquía de los módulos del diseño en Verilog.



A continuación se realiza una descripción de cada modulo del diseño. Los bloques de VIO e ILA se detallan en la Sección 4.3.

4.1.1. Módulo counter

El módulo counter consiste en un contador parametrizado que controla su estado mediante señales de reloj, reset y selectores de límites de referencia (R0, R1, R2, y R3). A continuación se describen sus componentes principales:

- **Parámetros:**

- n_SW: Cantidad de bits del selector (switches).
- n_COUNT: Cantidad de bits del contador.

- **Entradas:**

- i_clk: Señal de clock.
- i_reset: Señal de reset asíncrona.
- i_sw: Selectores del el límite de referencia y habilitador del contador.

- **Salidas:**

- o_valid: Salida de enable para los módulos de secuencias (shift_register, flash y shift_reg2leds).

- **Registros (reg):**

- counter: Almacena el valor actual del contador.
- enable: Controla la señal de habilitación de salida o_valid.

- **Conexiones (wire):**

- limit_ref: Establece el límite de referencia (R0, R1, R2, y R3) del contador a partir de un multiplexor, el cual usa como selector a i_sw[2:1].

- **Bloque always:** Actúa con cada flanco ascendente de las señales de clock o reset. Se utiliza para controlar el incremento del contador y la activación de enable según las condiciones de i_reset y i_sw[0].

- **Asignaciones (assign) de o_valid:** Se asigna la señal enable a la salida o_valid para indicar la validez del estado del contador.

Este módulo actúa como un divisor de frecuencia parametrizado, definiendo una señal o_valid que determina la velocidad de las secuencias de los módulos shift_register, flash y shift_reg2leds.



4.1.2. Módulos de secuencias

Los módulos `shift_register`, `flash` y `shift_reg2leds` están diseñados para generar diferentes secuencias de señales para los LEDs RGB de salida. Cada uno de estos módulos posee su propia lógica de generación de señales en registros internos, que luego se asignan a los LEDs por medio de la salida `o_led` de cada módulo.

1. **shift_register:** Implementa un registro de corrimiento tipo anillo, donde solo un único bit se encuentra en estado de alto (1) mientras los demás permanecen en estado de bajo (0). La dirección del desplazamiento se controla mediante una entrada `i_dir`, que está conectada al selector `i_sw[3]`.
2. **flash:** Describe un registro que alterna su valor entre `4'b0000` y `4'b1111` con cada pulso de la señal `valid`.
3. **shift_reg2leds:** A diferencia de los módulos anteriores, este módulo cuenta con dos registros: `ms_reg` y `ls_reg`. La secuencia generada por este módulo consiste en alternar entre los valores `4'b1001`, `4'b0110` y `4'b0000`, lo cual produce un efecto visual de dos LEDs desplazándose hacia el centro o hacia los extremos. Para lograr esto de manera parametrizada, se implementa el corrimiento tipo anillo de ambos registros en direcciones opuestas. La salida `o_led` solo utiliza los bits más significativos de `ms_reg` y los menos significativos de `ls_reg`, ignorando los bits centrales. Al igual que en `shift_register`, la dirección del desplazamiento se controla mediante la entrada `i_dir`.

A continuación se describe la estructura general de los módulos mencionados:

■ **Parámetros:**

- `n_LEDS`: Cantidad de bits para representar los LEDs.

■ **Entradas:**

- `i_clk`: Señal de reloj.
- `i_reset`: Señal de reset asincrónica.
- `i_valid`: Señal de habilitación (proveniente del módulo counter) para pasar al siguiente estado dentro del always.
- `i_dir`: Selector de dirección del corrimiento (1'b0 para izquierda, 1'b1 para derecha).

■ **Salidas:**

- `o_led`: Salida que muestra la secuencia de los LEDs RGB.

■ **Registros (reg):**

- `shift_reg`: Almacena el estado actual del registro de corrimiento en el módulo `shift_register`.
- `flash`: Almacena el estado actual del registro en el módulo `flash`.



- `ms_reg`, `ls_reg`: Almacenan el estado actual de los registros en el modulo `shift_reg2leds`.
- **Bloque** `always`: Lógica secuencial activada por el flanco de subida de `i_clk` o `i_reset` en su lista de control. Dentro de esta, se describe la secuencia de cada módulo.
- **Asignación** (`assign`) `o_led`: Conecta el valor del registro correspondiente (`shift_reg`, `flash`, `ms_reg`, o `ls_reg`) con la salida `o_led`, reflejando el estado actual en los LEDs.

4.1.3. Módulo top

El módulo top se encarga de realizar la interconexión de los módulos anteriores con las entradas y salidas del sistema y entre sí mismos. Además, realiza la lectura de los pulsadores `i_btn` para la selección de los LEDs RGB de salida, así como la secuencia a asignar a los mismos.

- **Entradas:**
 - `i_clk`: Señal de reloj.
 - `i_reset`: Señal de reset.
 - `i_sw`: Selectores o interruptores (switches).
 - `i_btn`: Pulsadores (buttons).
- **Salidas:**
 - `o_led`: LEDs indicadores conectados a los pulsadores.
 - `o_led_r`: LEDs rojos de salida.
 - `o_led_g`: LEDs verdes de salida.
 - `o_led_b`: LEDs azules de salida.
- **Wires:**
 - `valid`: Señal de habilitación para el corrimiento de bits (salida del contador).
 - `led_SR`: Salida del módulo `shift_register` para los LEDs.
 - `led_FS`: Salida del módulo `flash` para los LEDs.
 - `led_SR2L`: Salida del módulo `shift_reg2leds` para los LEDs.
- **Registros (Reg):**
 - `led`: Registro para seleccionar la salida de `shift_register`, `flash` o `shift_reg2leds` hacia los LEDs.
 - `prevBtn`: Registro para almacenar el estado previo de los pulsadores (`i_btn`).
 - `btn`: Registro para almacenar la lectura actual de los pulsadores (`i_btn`) utilizada para la selección del color de los LEDs.



- **count:** Contador utilizado para cambiar entre las funciones de `shift_register`, `flash` y `shift_reg2leds`.
- **Instanciación:** Se instancian los módulos como `u_counter`, `u_shift_reg`, `u_flash` y `u_shift_reg2led`, realizando las interconexiones ya mencionadas.
- **Always Secuencial:** El módulo top posee dos bloques `always` secuenciales que se usan para almacenar las señales de `i_btn` en registros, de forma que se busca hacer una detección de flanco ascendente por medio de la comparación de la entrada actual de `i_btn` con el valor leído en un flanco de reloj anterior, almacenado en un registro `prevBtn`. El valor de `i_btn` se almacena en un registro `btn` cada vez que `i_btn = 1` y `prevBtn = 0`.
- **Always Combinacional:** Si bien la selección de los LEDs RGB se realiza con un pulsador específico, la selección de las secuencias se realiza únicamente con el bit `btn[0]`. Para lograr esto, se implementa un segundo `always` secuencial que incrementa un contador `count` de 0 a 2 para realizar la asignación de la salida de las secuencias a un registro `led` por medio de un multiplexor que realiza la selección según el valor del contador, dentro de un `always` combinacional.
- **Asignación a las salidas:** La asignación a las salidas se realiza por medio de un operador ternario, que permite asignar a los LEDs RGB la señal `led` o asignar un `1'b0` a los mismos.

4.2. Simulación en Testbench

El módulo `tb_top` se encarga de verificar el correcto funcionamiento del módulo `top`. Este testbench incluye la instanciación del módulo `top`, la generación de la señal de reloj (`clock`), la generación de las señales de entradas para la observación de las salidas en la simulación.

- **Instanciación del Módulo Top:**
 - El módulo `top` es instanciado con parámetros que definen el número de bits de los selectores (`n_SW`), pulsadores (`n_BTN`), LEDs (`n_LEDS`), y el contador (`n_COUNT`), el cual se lo define en 13 bits para facilitar la visualización en la simulación.
 - Las señales de entrada (`i_clk`, `i_reset`, `i_sw`, `i_btn`) y las señales de salida (`o_led`, `o_led_r`, `o_led_g`, `o_led_b`) son conectadas entre el testbench y el módulo `top`.
- **Etapas de Simulación:**
 - **Inicialización:** Las señales de entrada (`i_clk`, `i_reset`, `i_sw`, `i_btn`) son inicializadas a valores conocidos
 - **Pruebas de Secuencia:** Se configuran los selectores y pulsadores para probar diferentes secuencias de funcionamiento del módulo `top`, incluyendo el cambio de secuencias y la selección de colores de los LEDs.



- **Pruebas de Habilitación del Contador:** Se deshabilita y habilita el contador para verificar su funcionamiento.
 - **Pruebas de Límites de Referencia:** Se prueban los diferentes límites de referencia del contador para asegurar que se comporta según lo esperado.
 - **Pruebas de Dirección:** Se cambian las direcciones de corrimiento del registro de desplazamiento (SR y SR2L) y se verifica el comportamiento.
 - **Pruebas de Color:** Se prueban las diferentes combinaciones de colores de los LEDs (R, G, B) para confirmar que el módulo responde correctamente a las entradas.
- **Generación de la Señal de Reloj:**
- Se utiliza un bloque always para generar la señal de reloj (i_clk), la cual invierte su valor cada 5 unidades de tiempo, simulando un reloj con un periodo de 10 unidades de tiempo.

4.2.1. Simulación

A continuación se presentan distintas imágenes de la simulación, donde se busca mostrar las distintas funcionalidades del sistema.

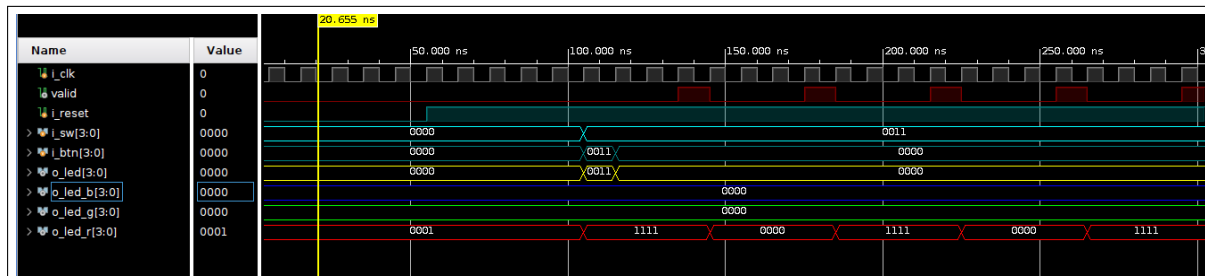


Figura 4: Salida de led rojo en modo FS.

- En la Figura 4 se observa cómo la entrada en el pulsador i_btn se ve reflejada en la salida o_led.
- Se observa el cambio a la secuencia FS (flash) con salida de los leds en rojo.

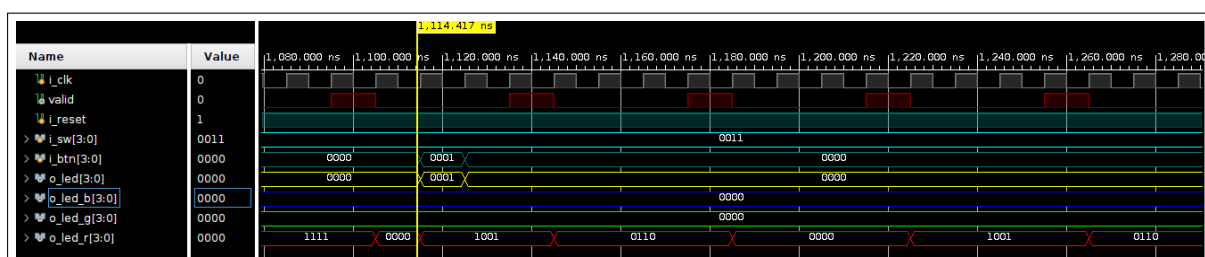


Figura 5: Salida de led rojo en modo SR2L.

- En la Figura 5 se observa el cambio de la secuencia FS a SR2L (shift_reg2leds).

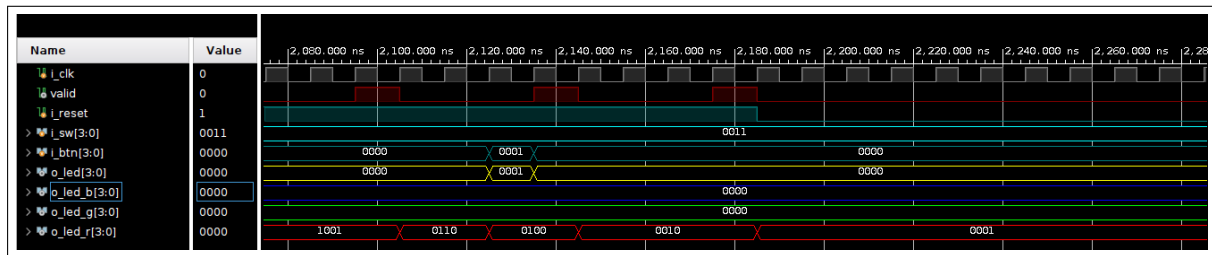


Figura 6: Demostración de acción de señal de reset.

- En la Figura 6 se observa el cambio de la secuencia SR2L a SR (shift_register).
- También se observa cómo actúa la señal de reset, y la inicialización de la secuencia de salida en 1'b0001, dado que se encuentra en la secuencia SR.

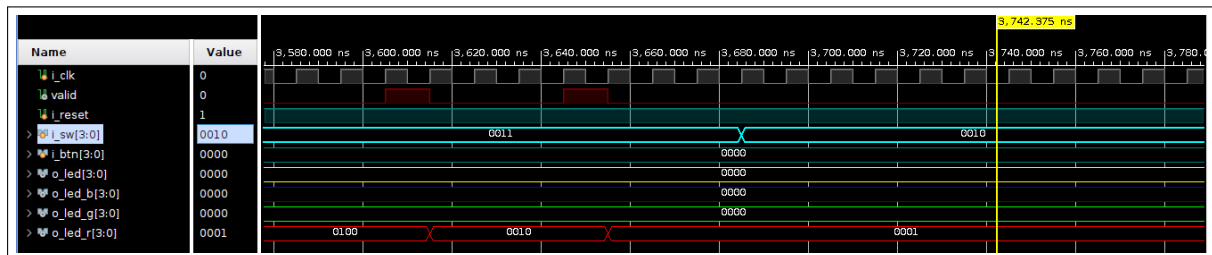


Figura 7: Demostración de deshabilitación del contador.

- En la Figura 7 se observa cómo deshabilitar el contador detiene la secuencia actual de la salida de los leds, es decir, mantiene el estado actual hasta no recibir el próximo pulso de valid.

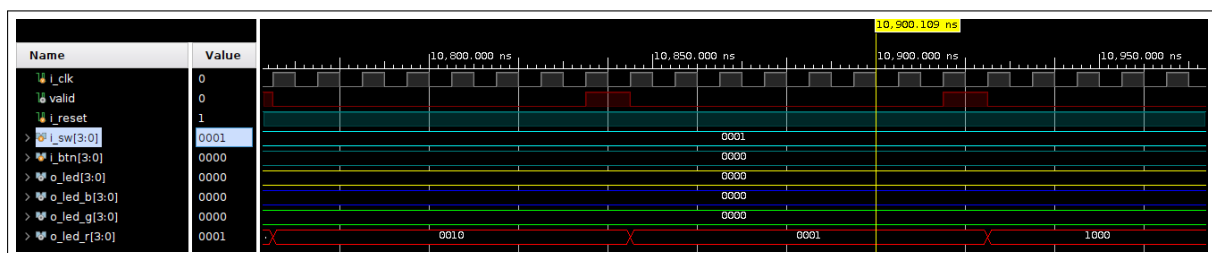


Figura 8: Cambio de límite de referencia del contador.

- En la Figura 8 se observa un cambio en el límite de referencia del contador. En este caso, al elegir un límite más grande, baja la velocidad (aumenta el periodo) de la señal.



- Figura 10: Cambio de color de salida de rojo a verde.

- Figura 11: Cambio de color, salida simultánea en rojo y azul.

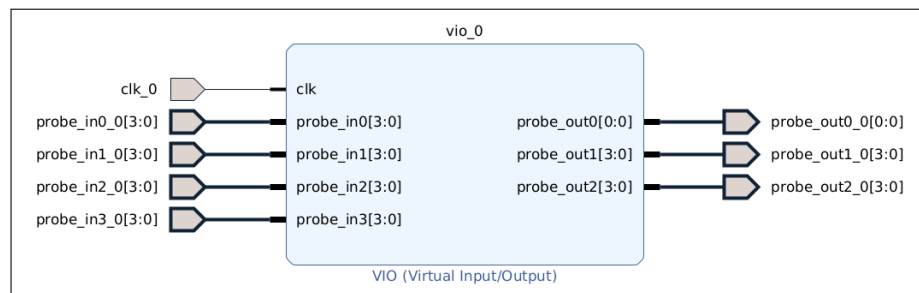
- Figura 12: Cambio de secuencia de FS a SR2L.



- | Name | Value |
|----------------|-------|
| i_clk | 1 |
| valid | 0 |
| i_reset | 1 |
| > i_sw[3:0] | 0011 |
| > i_btn[3:0] | 0000 |
| > o_led[3:0] | 0000 |
| > o_led_b[3:0] | 0000 |
| > o_led_g[3:0] | 0000 |
| > o_led_r[3:0] | 1000 |
-
- The timing diagram displays the following signals and their values over time:
- i_clk**: A periodic clock signal.
 - valid**: A signal that is 0 throughout the shown time range.
 - i_reset**: A signal that is 1 throughout the shown time range.
 - i_sw[3:0]**: A 4-bit input signal with a value of 0011.
 - i_btn[3:0]**: A 4-bit input signal with a value of 0000.
 - o_led[3:0]**: A 4-bit output signal with a value of 0000.
 - o_led_b[3:0]**: A 4-bit output signal with a value of 0000.
 - o_led_g[3:0]**: A 4-bit output signal with a value of 0000.
 - o_led_r[3:0]**: A 4-bit output signal with a value of 1000.
- The time axis is marked at 19,500,000 ns, 19,550,000 ns, 19,600,000 ns, and 19,650,000 ns.

- En la Figura 13 se observan las tres salidas de los leds RGB funcionando en simultáneo.

Para realizar la conexión del módulo de la VIO y el módulo de la ILA con el top, se utiliza la instrucción `'define USE_VIO_ILA`. Cuando esta línea de código está comentada, se tiene la descripción para realizar la síntesis y simulación con testbench, y cuando está descomentada, se tiene la descripción adecuada para usar la VIO y la ILA en la implementación en la FPGA.



En la Figura 14 se observa el esquema de la VIO, la cual posee cuatro entradas correspondientes a las salidas de los LEDs del módulo top, junto con tres salidas correspondientes a las entradas de `i_reset`, `i_sw` y `i_btn`. Este módulo se usa para establecer las entradas del sistema y observar las salidas del módulo implementado en la FPGA de forma virtual.

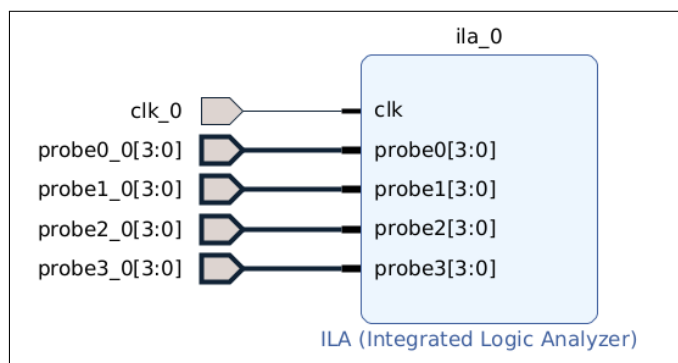


Figura 15: Esquema de la ILA.

En la Figura 15 se observa el esquema de la ILA, la cual posee cuatro puntas de prueba correspondientes a las cuatro salidas de los LEDs del módulo top.

4.4. Pruebas en la FPGA remota

Una vez instanciados los bloques de la VIO y la ILA en el módulo top, se realiza la implementación y generación del bitstream para poder probar el sistema en la FPGA.

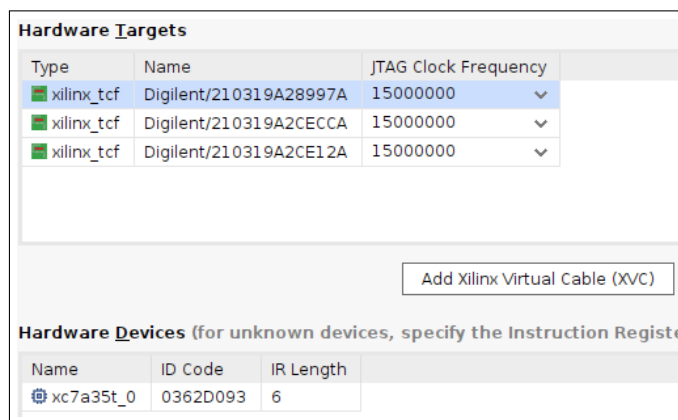


Figura 16: Conexión con la FPGA remota.

En la Figura 16 se muestra la selección de la placa, accedida por medio del programa PuTTY. Una vez programada con el archivo de bitstream, los módulos de VIO e ILA permiten realizar diversas pruebas.



4.4.1. Pruebas con la VIO

La VIO permite configurar valores en sus salidas (entradas del sistema descrito) y visualizar sus correspondientes entradas (salidas del sistema descrito). A continuación se presentan algunas pruebas, basadas en la simulación del testbench.

Puertos de la VIO:

- `clk_0 = (i_clk)`
- `probe_in0_0 = (o_led)`
- `probe_in1_0 = (o_led_r)`
- `probe_in2_0 = (o_led_g)`
- `probe_in3_0 = (o_led_b)`
- `probe_out0_0 = (i_reset)`
- `probe_out1_0 = (i_sw)`
- `probe_out2_0 = (i_btn)`






















hw_vio_1				
    				
Name	Value	Activity	Direction	VIO
 u_vio/vio_0_probe_out0	[B] 1		Output	hw_vio_1
✓  u_vio/vio_0_probe_out1[3:0]	[H] 9		Output	hw_vio_1
 u_vio/vio_0_probe_out1[3]	1		Output	hw_vio_1
 u_vio/vio_0_probe_out1[2]	0		Output	hw_vio_1
 u_vio/vio_0_probe_out1[1]	0		Output	hw_vio_1
 u_vio/vio_0_probe_out1[0]	1		Output	hw_vio_1
✓  u_vio/vio_0_probe_out2[3:0]	[H] B		Output	hw_vio_1
 u_vio/vio_0_probe_out2[3]	1		Output	hw_vio_1
 u_vio/vio_0_probe_out2[2]	0		Output	hw_vio_1
 u_vio/vio_0_probe_out2[1]	1		Output	hw_vio_1
 u_vio/vio_0_probe_out2[0]	1		Output	hw_vio_1
>  u_vio/probe_in2_0_1[3:0]	[H] 0		Input	hw_vio_1
>  u_vio/probe_in3_0_1[3:0]	[H] 0		Input	hw_vio_1
>  u_vio/probe_in1_0_1[3:0]	[H] 8		Input	hw_vio_1
>  u_vio/probe_in0_0_1[3:0]	[H] B		Input	hw_vio_1

Figura 17: Prueba de la secuencia SR con la salida de los LEDs rojos.



Secuencia SR: 4'b0001 (1), 4'b0010 (2), 4'b0100 (4), 4'b1000 (8).

hw_vio_1					
Q + -					
Name	Value	Activity	Direction	VIO	
u_vio/vio_0_probe_out0	[B] 1		Output	hw_vio_1	
▼ u_vio/vio_0_probe_out1[3:0]	[H] 1		Output	hw_vio_1	
u_vio/vio_0_probe_out1[3]	0		Output	hw_vio_1	
u_vio/vio_0_probe_out1[2]	0		Output	hw_vio_1	
u_vio/vio_0_probe_out1[1]	0		Output	hw_vio_1	
u_vio/vio_0_probe_out1[0]	1		Output	hw_vio_1	
▼ u_vio/vio_0_probe_out2[3:0]	[H] D		Output	hw_vio_1	
u_vio/vio_0_probe_out2[3]	1		Output	hw_vio_1	
u_vio/vio_0_probe_out2[2]	1		Output	hw_vio_1	
u_vio/vio_0_probe_out2[1]	0		Output	hw_vio_1	
u_vio/vio_0_probe_out2[0]	1		Output	hw_vio_1	
> u_vio/probe_in2_0_1[3:0]	[H] F		Input	hw_vio_1	
> u_vio/probe_in3_0_1[3:0]	[H] F		Input	hw_vio_1	
> u_vio/probe_in1_0_1[3:0]	[H] 0		Input	hw_vio_1	
> u_vio/probe_in0_0_1[3:0]	[H] D		Input	hw_vio_1	

Figura 18: Prueba de la secuencia FS con la salida de los LEDs verdes y azules.

Secuencia FS: 4'b0000 (0), 4'b1111 (F).



hw_vio_1				
Name	Value	Activity	Direction	VIO
u_vio/vio_0_probe_out0	[B] 1		Output	hw_vio_1
✓ u_vio/vio_0_probe_out1[3:0]	[H] D		Output	hw_vio_1
u_vio/vio_0_probe_out1[3]	1		Output	hw_vio_1
u_vio/vio_0_probe_out1[2]	1		Output	hw_vio_1
u_vio/vio_0_probe_out1[1]	0		Output	hw_vio_1
u_vio/vio_0_probe_out1[0]	1		Output	hw_vio_1
✓ u_vio/vio_0_probe_out2[3:0]	[H] E		Output	hw_vio_1
u_vio/vio_0_probe_out2[3]	1		Output	hw_vio_1
u_vio/vio_0_probe_out2[2]	1		Output	hw_vio_1
u_vio/vio_0_probe_out2[1]	1		Output	hw_vio_1
u_vio/vio_0_probe_out2[0]	0		Output	hw_vio_1
✓ u_vio/probe_in2_0_1[3:0]	[H] 6		Input	hw_vio_1
u_vio/probe_in2_0_1[3]	0		Input	hw_vio_1
u_vio/probe_in2_0_1[2]	1		Input	hw_vio_1
u_vio/probe_in2_0_1[1]	1		Input	hw_vio_1
u_vio/probe_in2_0_1[0]	0		Input	hw_vio_1
> u_vio/probe_in3_0_1[3:0]	[H] 6		Input	hw_vio_1
> u_vio/probe_in1_0_1[3:0]	[H] 6		Input	hw_vio_1
> u_vio/probe_in0_0_1[3:0]	[H] E		Input	hw_vio_1

Figura 19: Prueba de la secuencia SR2L con todas las salidas de los LEDs RGB.

Secuencia SR2L: 4'b1001 (9), 4'b0110 (6), 4'b0000 (0).



4.4.2. Pruebas con la ILA

La ILA permite visualizar las salidas del módulo, es decir, los LEDs, bajo diversas condiciones. Para realizar las pruebas se establece un valor de disparo (trigger) para capturar los valores de las salidas en el analizador de ondas, como se muestra en las siguientes imágenes.

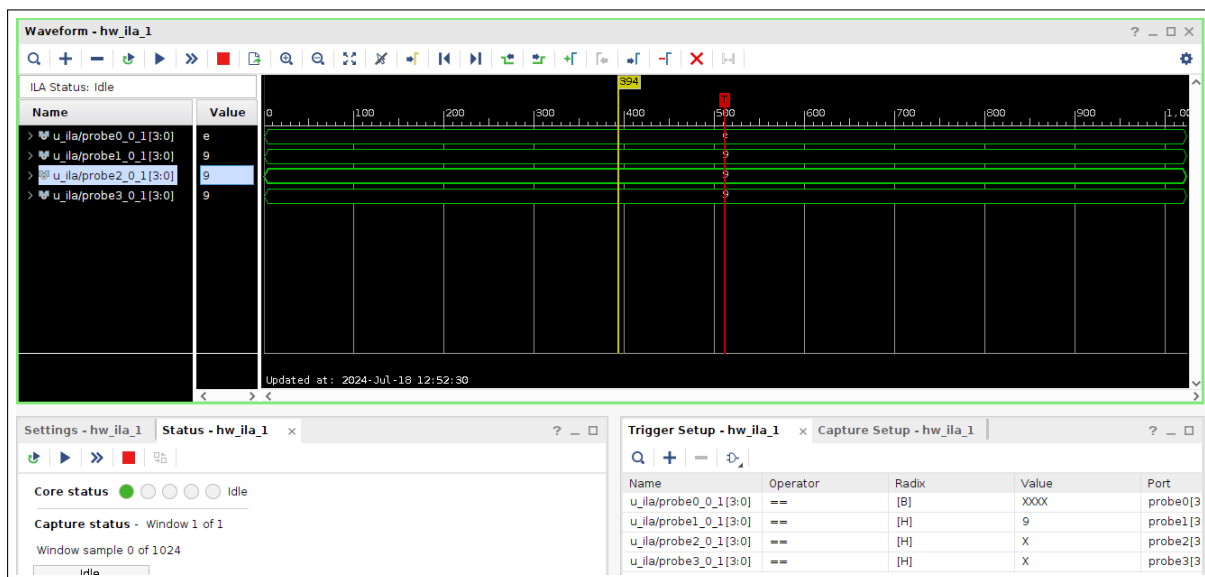


Figura 20: Prueba de la secuencia SR2L con la ILA.

Secuencia SR2L: 4'b1001 (9), 4'b0110 (6), 4'b0000 (0).

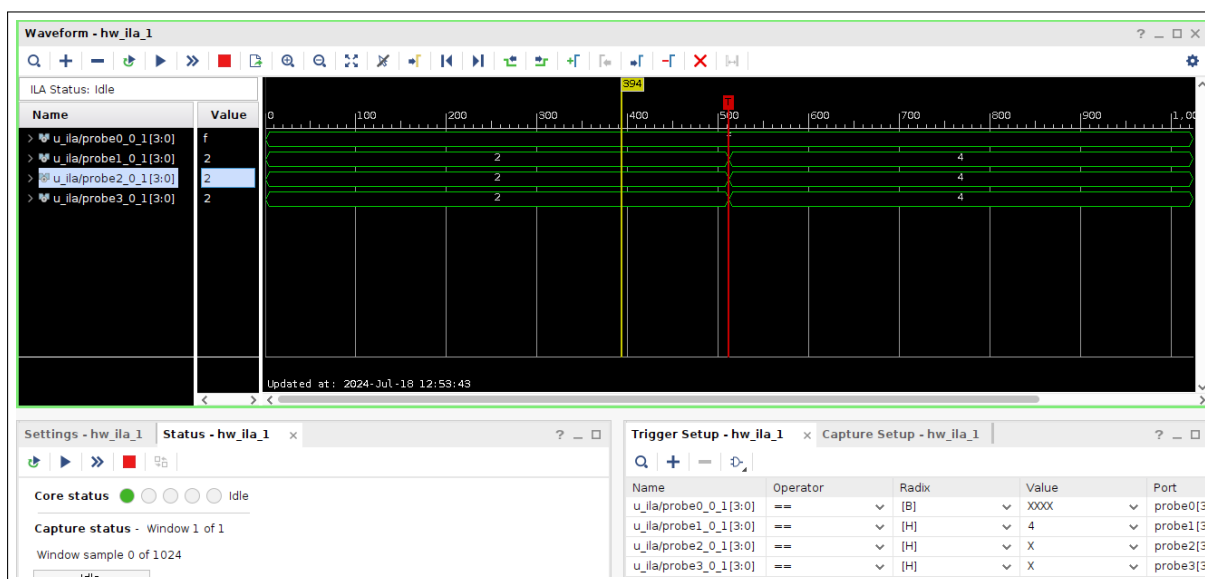


Figura 21: Prueba de la secuencia SR con la ILA.

Secuencia SR: 4'b0001 (1), 4'b0010 (2), 4'b0100 (4), 4'b1000 (8).

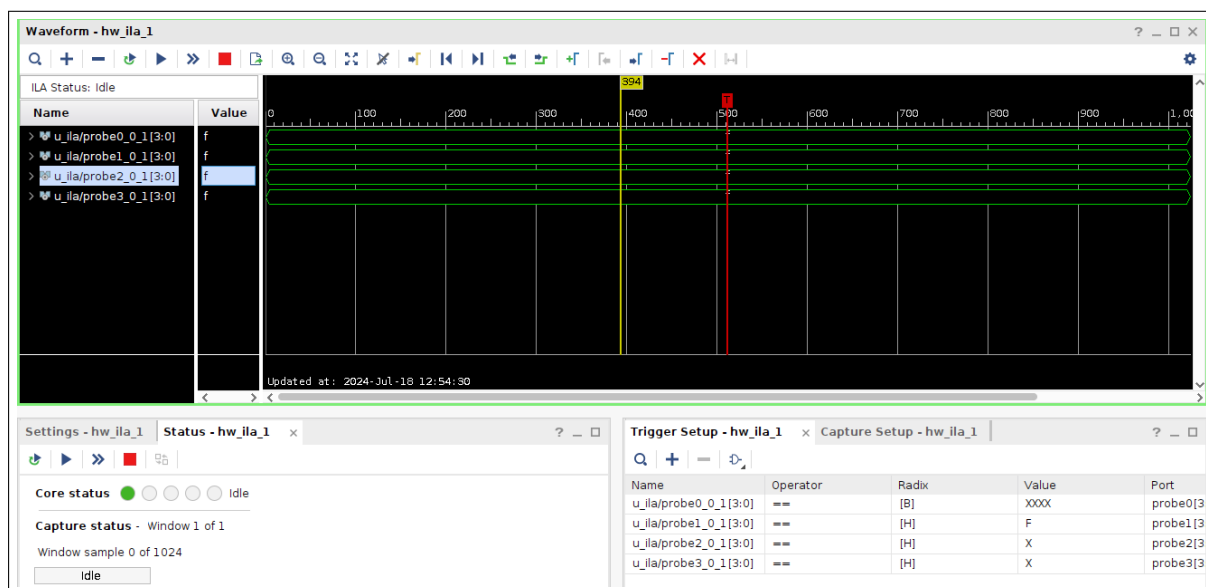


Figura 22: Prueba de la secuencia FS con la ILA.

Secuencia FS: 4'b0000 (0), 4'b1111 (F). s

5. Conclusiones

A partir del presente trabajo se pudieron aplicar los distintos conceptos de descripción en Verilog de sistemas secuenciales y combinacionales, junto con temas de instanciación, uso de distintos tipos de datos y descripción jerárquica.

Además, se aprendió cómo realizar diversas pruebas en una FPGA remota, haciendo uso de los bloques de la VIO y la ILA.

En la Figura 23 se observa el esquema del sistema sintetizado con los bloques de la VIO y la ILA. Se pueden observar estos dos bloques en azul, junto con los módulos de secuencia en amarillo, el módulo del contador en violeta, y los multiplexores y bloques secuenciales, de contador y detector de las señales de los pulsadores en verde.

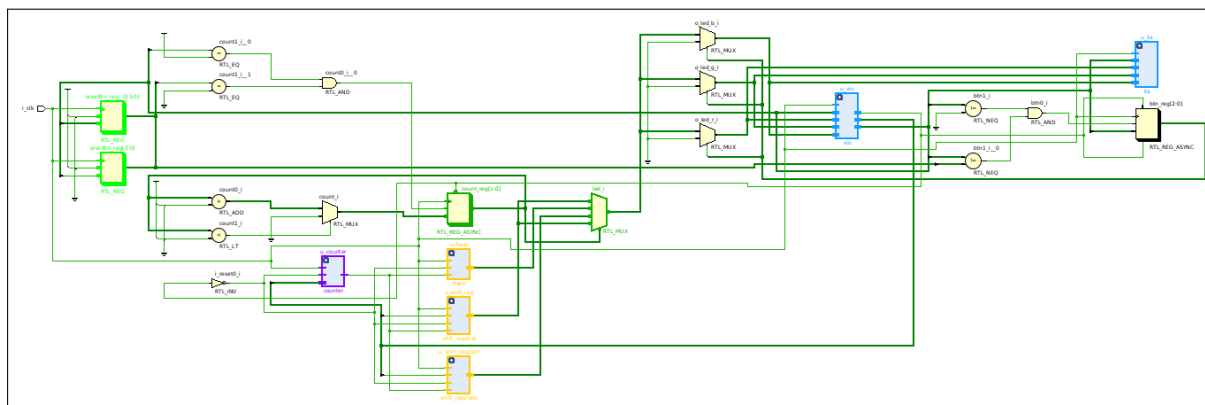


Figura 23: Esquema de RTL con módulos de VIO e ILA.



En la Figura 24 se observa el mismo diagrama, pero con las salidas y entradas del módulo top, sin la instanciación de la VIO e ILA.

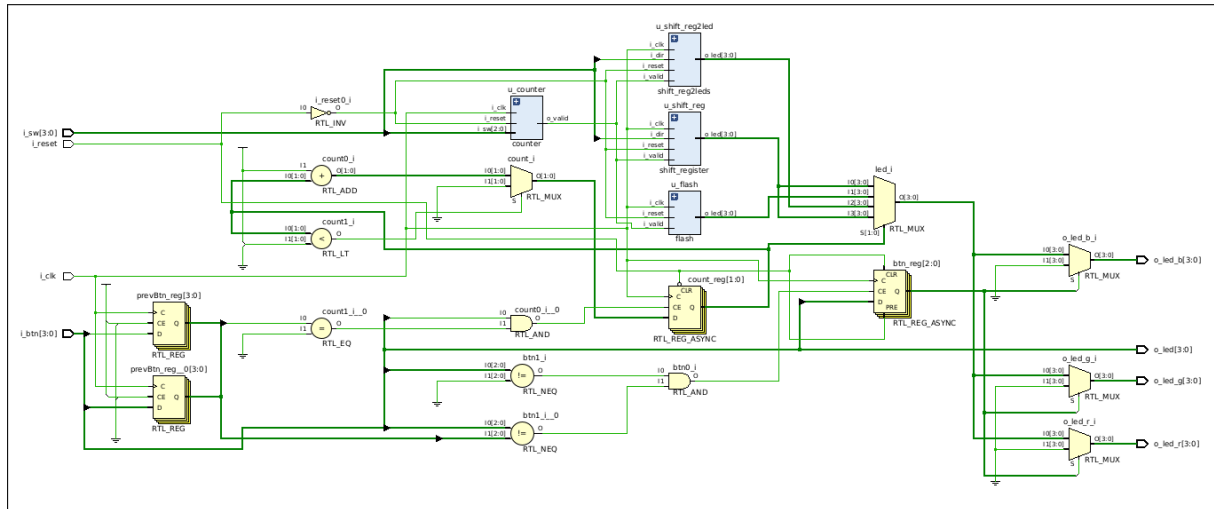


Figura 24: Esquema de RTL del proyecto general.