***Laboratorio 1 – Botones y LED***

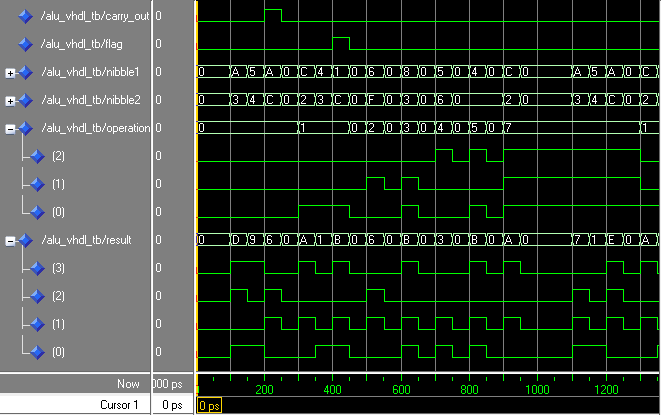
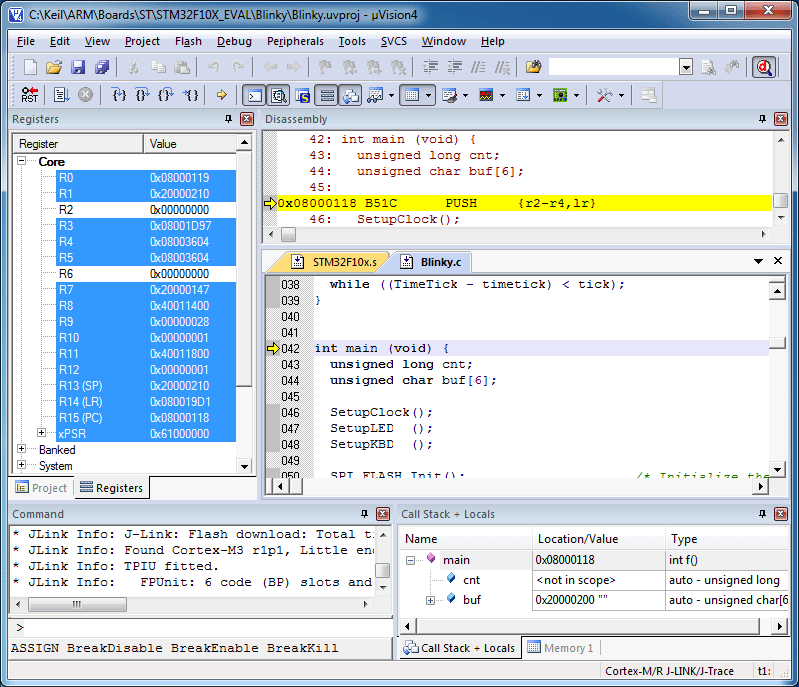
Objetivos del laboratorio son conocer:

* La necesidad de simulación
* El uso de herramientas estándar para simulación y diseño de VHDL
  + Vivado: WebPack
  + Simulador:
* Cómo navegar por el entorno de desarrollo
  + Verificar que la configuración es la correcta
  + Flujo de Desarrollo

*Simulación en VHDL:*

La simulación es necesaria para corroborar el diseño antes de la implementación.

En microcontroladores generalmente se tiene un software (IDE) en el cual podemos integrar breakpoints que son puntos donde el programa es detenido y observar las variables, si bien es cierto esto es muy fácil de realizar en MCUs, en FPGAs y VHDL es prácticamente y virtualmente poco probable de hacer esto.



*Figura 1.1 - Depuración en Microcontroladores Figura 1.2 – Depuración en FPGAs/CPLDs/SPLDs*

El diseño prácticamente se descarga en la plataforma de desarrollo y se verifica si el funcionamiento final es el correcto. Usualmente el diseño sintetiza, el código está bien escrito pero la manera de depurar el código es por medio de la visualización de las formas de onda y la observación de las entradas vs salida todo esto observando la línea de tiempo.

Esto no es del todo malo, esta simulación nos sirve para documentar y por solamente documentando la forma de onda un ingeniero puede entender mejor su diseño.

Para diseños sencillos como este no es necesaria la simulación, pero para diseños más complejos puede determinar cuanto tiempo puede encontrar un error.

Lo necesario para poder simular:

* Haber completado el archivo o archivos VHDL para poder simular
* Completar el “test bench”.
  + El test bench es una prueba de validación de nuestro circuito específicamente para probar lo sintetizado.
  + Un “test bench” no es sintetizable.
  + Se debe integrar el diseño en VHDL con el test bench, usualmente llamada UUT o Unit Under Test.
* Si se trabaja con ModelSim, se debe crear un TCL scrpit.
  + Modelsim se utiliza con Altera Quartus II.
  + Acelera el diseño, simulación y se utiliza para automatizar el diseño pues posee herramientas de línea de comandos

*Ejemplo de diseño – Registro de Corrimiento:*

* Para realizar el siguente ejemplo realizaremos un registro de corrimiento y un test bench para el registro de corrimeinto
* Realizaremos un TCL para nuestro diseño en ModelSim
* Simularemos el registro de corrimiento en Vivado

*Registros de Corrimiento*

El registro de corrimiento que tendremos en este punto es un registro de corrimiento serial a paralelo, esto es, tendrá una entrada de 1-bit y una salida de 4-bits adicionalmente de la entrada de control. Algo parecido a la figura inferior



Figura 1.3. Dibujo en MS Visio para representar el registro de corrimiento

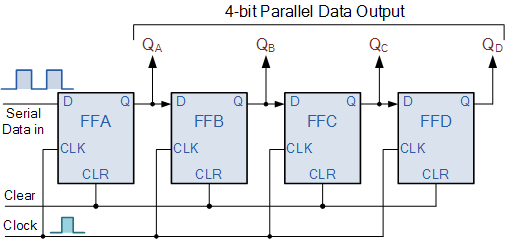


Figura 1.4 – Equivalente en Digital del registro de corrimiento a implementar.

*Creación del Proyecto*

1 - Abrir Vivado en la pantalla principal y presionar en crear proyecto

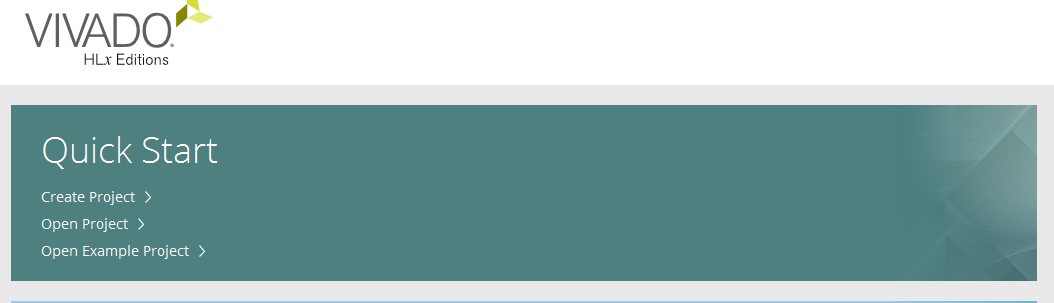


Figura 1.5 – Creación del proyecto en Vivado.

2 – Hacer click en siguiente

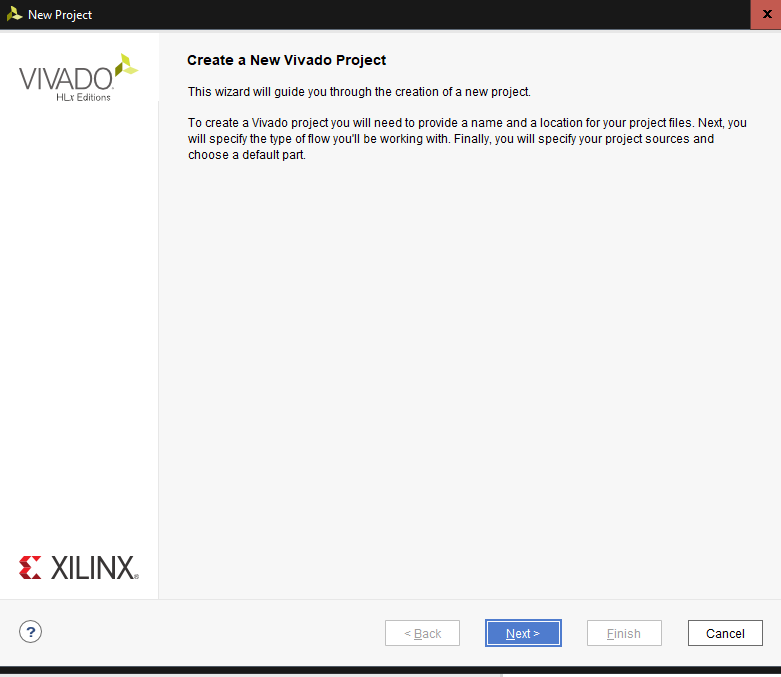


Figura 1.6 – Ventana de nuevo proyecto.

3 – Crear el nuevo proyecto en la carpeta de su conveniencia, asegúrese de tener la marca (checkbox) para que cree el subdirectorio.

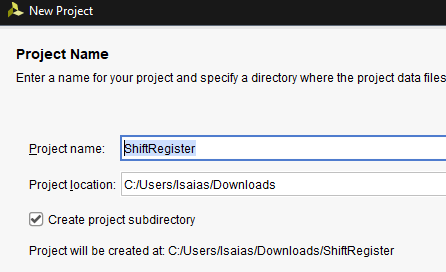
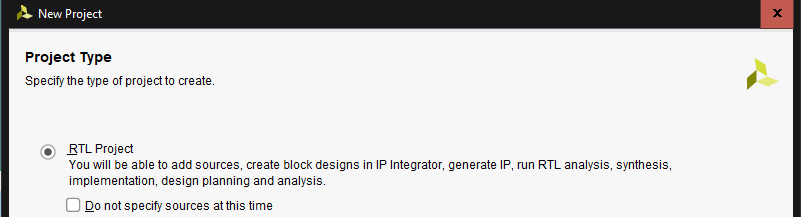


Figura 1.7 – Ventana de Nombre de Proyecto.

4 – Todos los proyectos que crearemos serán de RTL (Register Transfer Logic)

  
Figura 1.8 – Ventana de Tipo de Proyecto.

5 – Seleccione crear un archivo y llamelo ShiftRegister.vhd

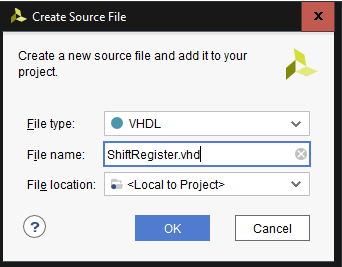


Figura 1.9 – Ventana de Archivo fuente (VHDL)

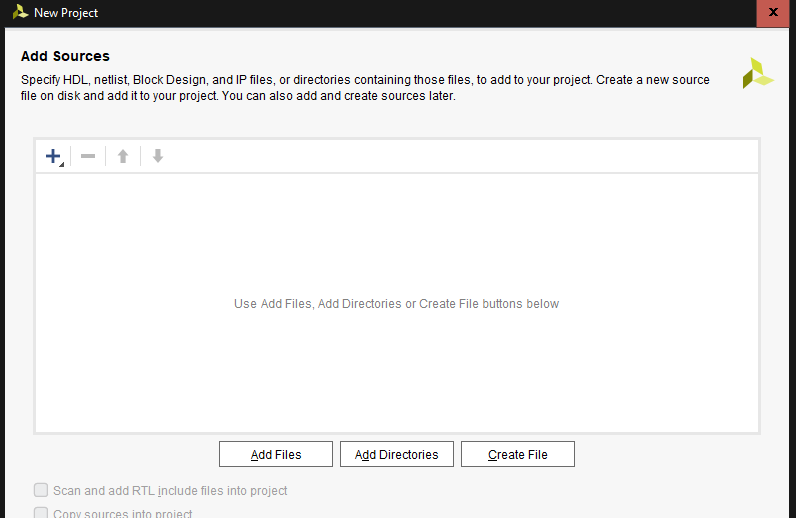


Figura 1.10 – Ventana de adición de fuentes. Aquí se muestra el archiv a añadir.

6 – Creado el archivo navegue al siguiente paso (7)

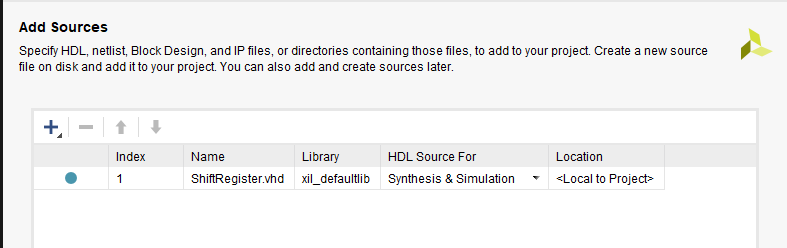


Figura 1.11 – Ventana de adición de fuentes. Con archivo añadido.

7 – No añadiremos ninguna limitente por lo que podemos cambiar a la siguiente pantalla.

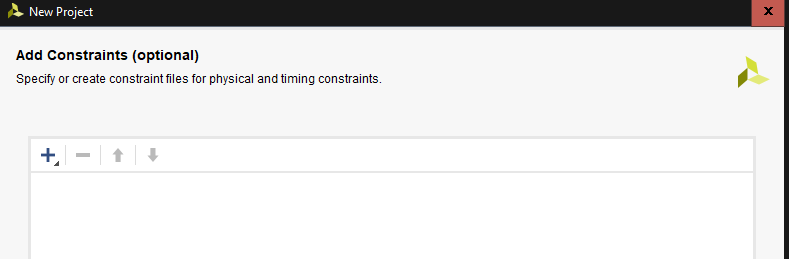


Figura 1.12 – Ventana de limitantes. Se explicará más adelante para que sirve.

8 – Busque su tarjeta relacionada dependiendo del FPGA. Este caso es para la Arty-A7

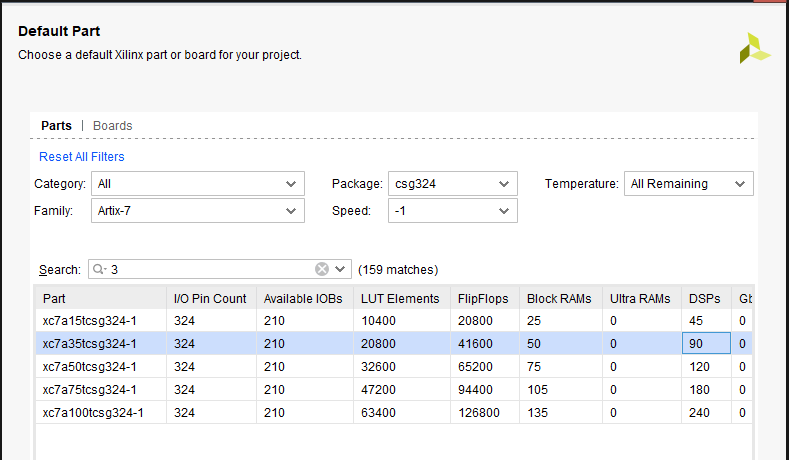


Figura 1.13 – Ventana para seleccionador de IC. Cada IC posee características diferentes en capacidad de LUTs, DSP, RAM, etc.

9 – Haga click en finalizar

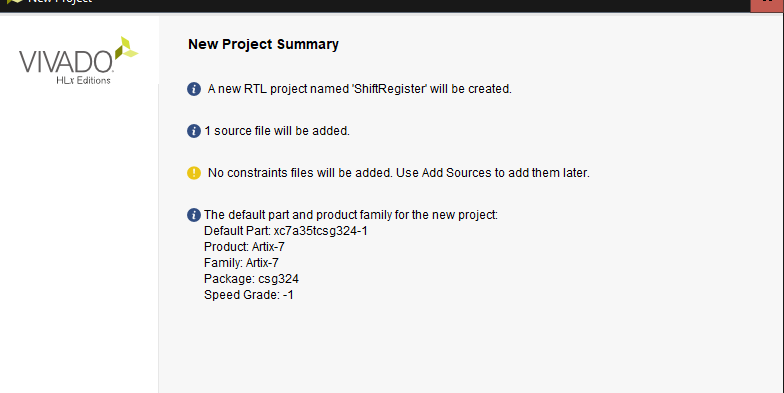


Figura 1.14 – Completado el proyecto luego podemos iniciar a codificar.

10 – Nos pedirá definir las entradas y salidas pero nosotros pasaremso este punto por alto para completar.

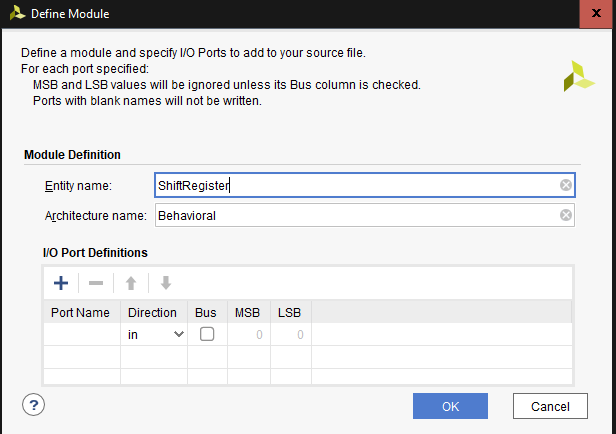


Figura 1.15 – Definición de Módulo. Este paso ayuda al generador automático a realizar la configuración del archivo VHD.

11 – Al ir al menú principal podemos abrir el archivo VHD y poder codificar

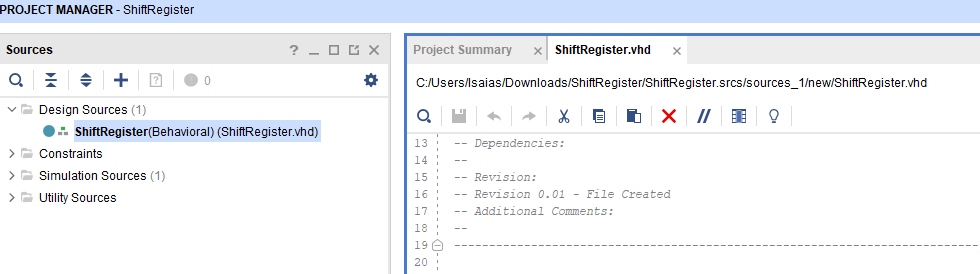
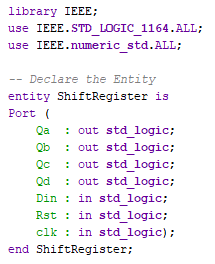
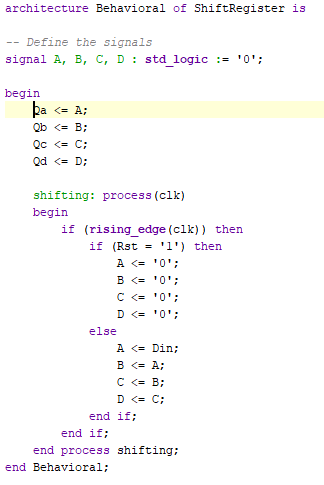


Figura 1.16 – Ventana de Project Manager. Desde aquí se pueden acceder al proyecto o definir generalidades.

12 – Cree el siguiente código en VHDL

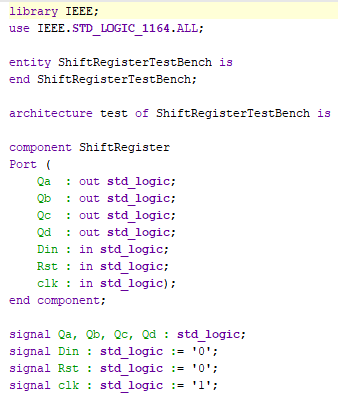
Listado 1.1 – Código en VHDL para registro de corrimiento.

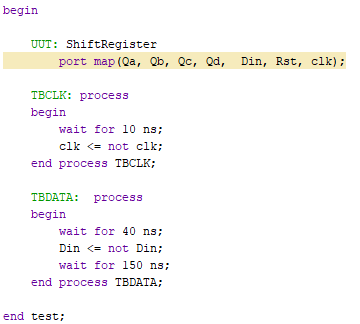




13 – Cree el siguiente archivo VHD que representa el test bench

Listado 1.2 – Listado para Test Bench. El test bench es un archivo para probar la arquitectura a base de la entidad.





14 – El Project manager debe lucir como sigue:

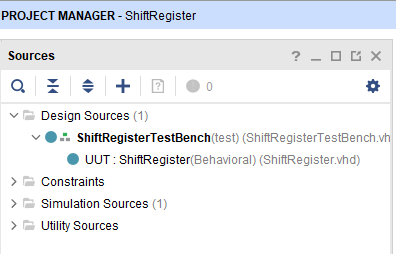


Figura 1.17 – Archivos VHD de Shift Register y Test Bench. Note que el archivo de registro de corrimiento (test bench) contiene la entidad.

Crear un TCL script:

15 – Cree el siguiente TCL en Vivado



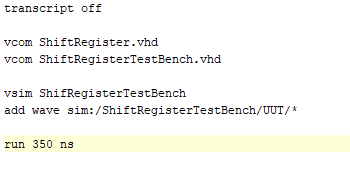


Figura 1.18 – Archivo TCL. El archivo es útil para correr en ModelSim, por el momento no lo utilizaremos pero se muestra como crearlo.

16 - Seguidamente corra la simulación

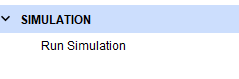


Figura 1.19 – Ejecución de Simulación

17 – Se debe observar la ventana que sigue

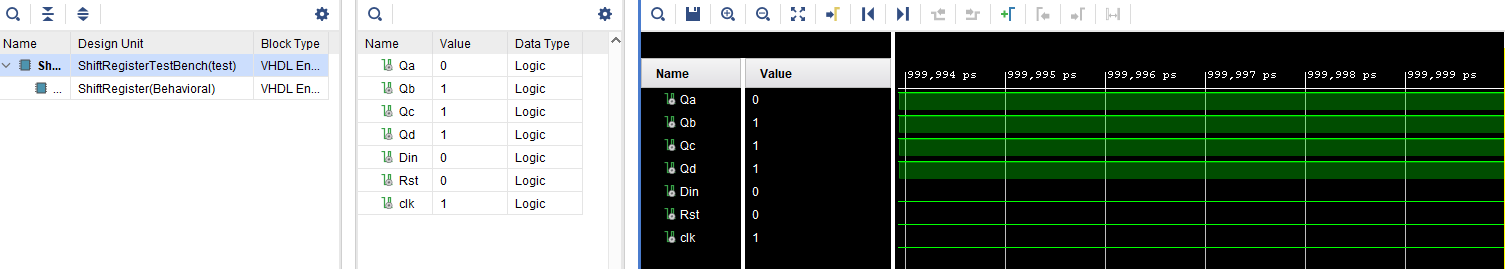


Figura 1.20 – Simulación. Actualmente se muestra solo una parte de la simulación.

18 – Presione Botón derecho sobre la simulación y deje la ventana en Full View o Ctrl+0

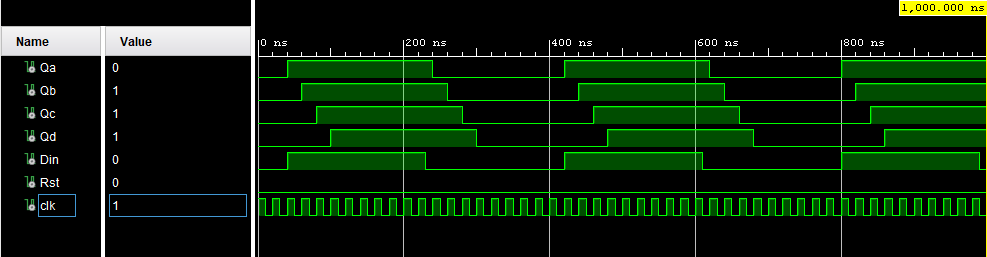


Figura 1.21 – Se muestra luego de hacer “fit” en la ventana toda la corrida de simulación en VHDL.

19 – Utilice los botones de simulación y navegue por toda la pantalla para observar como navegar en Vivado con la herramienta.

Creación de Proyecto de Botones y LED

1 – Cree un proyecto llamado Botones\_LED como se realizó en la sección anterior.

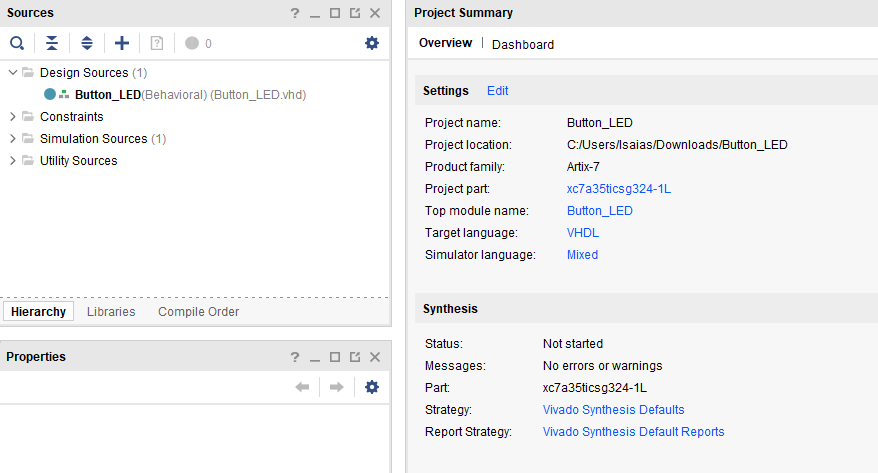
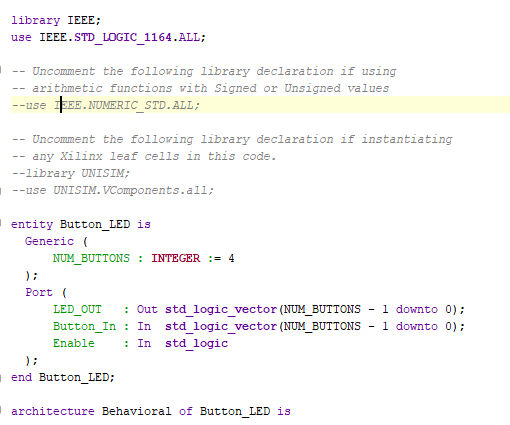
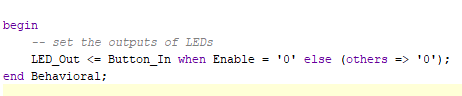


Figura 1.22 – Creación de Archivo VHD LED con botón.

2 – Crear el archivo de proyecto en VHDL para leds y botones.

Listado 1.3 – Creación del archivo de LEDs y Botones.





3 – Especificar los LEDs y Botones a utilizar en la tarjeta Arty A7, en nuestro caso.

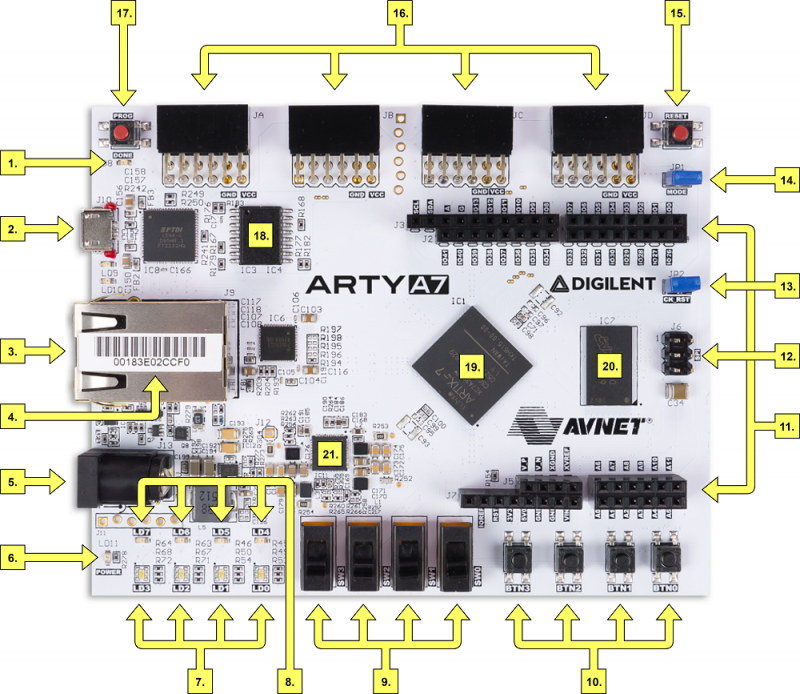
Primeramente mostramos los componentes que posee nuestra tarjeta y donde están ubicados.

Figura 1.23 – Componentes de Tarjeta Arty A7

Tabla 1.1 – Documentación de Componentes de Arty-A7

| **Callout** | **Description** | **Callout** | **Description** | **Callout** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | FPGA programming DONE LED | 8 | User RGB LEDs | 15 | chipKIT processor reset |
| 2 | Shared USB JTAG / UART port | 9 | User slide switches | 16 | Pmod connectors |
| 3 | Ethernet connector | 10 | User push buttons | 17 | FPGA programming reset button |
| 4 | MAC address sticker | 11 | Arduino/chipKIT shield connectors | 18 | SPI flash memory |
| 5 | Power jack for optional external supply | 12 | Arduino/chipKIT shield SPI connector | 19 | Artix FPGA |
| 6 | Power good LED | 13 | chipKIT processor reset jumper | 20 | Micron DDR3 memory |
| 7 | User LEDs | 14 | FPGA programming mode | 21 | Dialog Semiconductor DA9062 power supply |

4 - Dados los componentes de la tarjeta ahora notaremos en referencia cuales son las entradas y salidas.

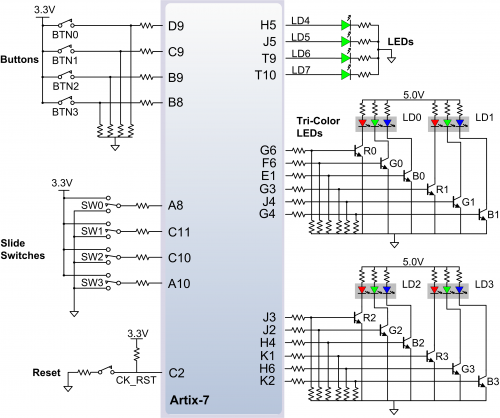


Figura 1.24 – Muestra de ubicación de los botones y LEDs en la Arty A7

5 – Crear el archivo de “constraints” o limintantes, que son necesarios para síntesis.

Situarse sobre Constraints y presionar add sources.

Agregar el archivo de constraints

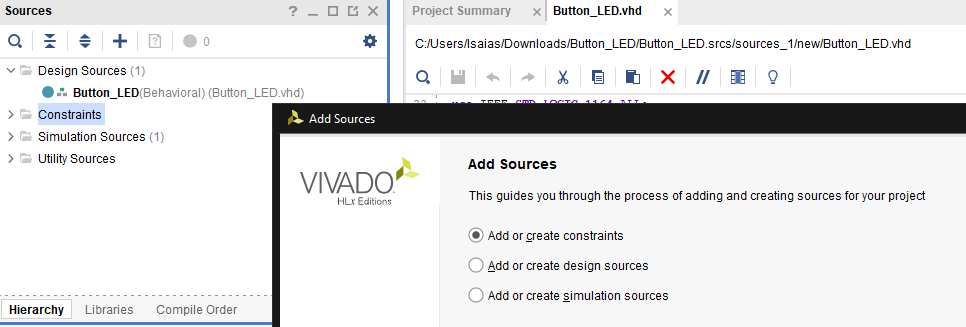


Figura 1.25 – Creación del archivo de “constraints”.

6 - Agregar el archivo de Arty para constraints

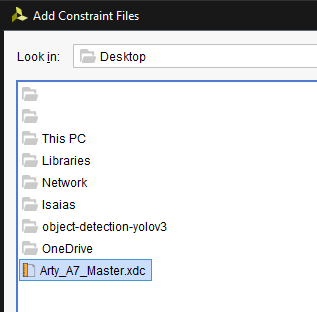


Figura 1.26 – Copia del archivo de definiciones.

7 - Asegurese de tener el click seleccionado para la copia del archivo maestro de constraints

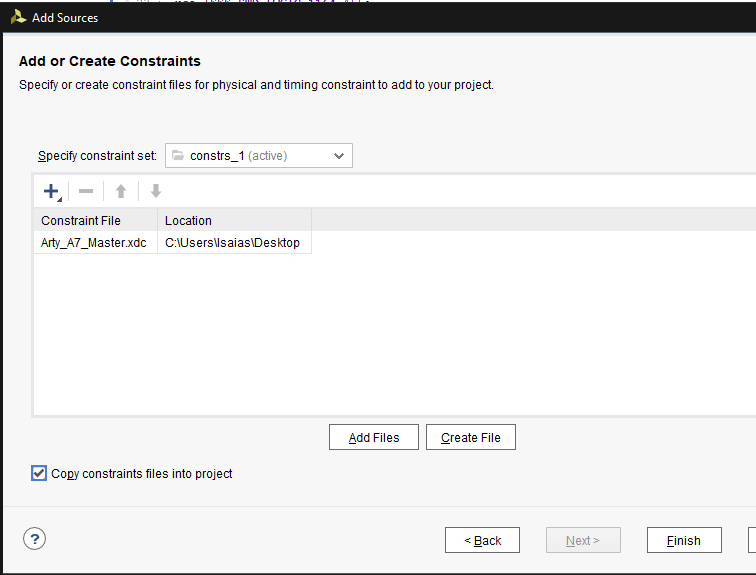


Figura 1.27 – Archivo de definiciones a copiarse a la carpeta de proyecto.

8 - Verifique la ventana de Constraints y observe que se añadió al proyecto

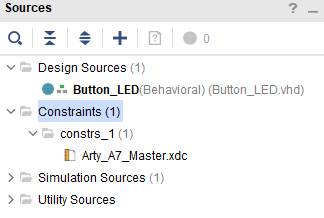


Figura 1.28 – Verificación del archivo de definiciones/limitantes (constraints).

9 - Abrir el archivo XDC y habilitar los LEDs y botones como sigue:

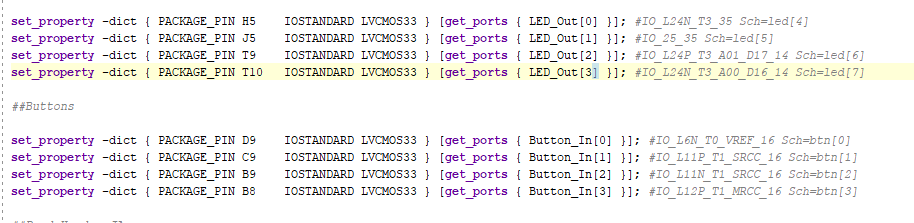


Figura 1.29 – Archivo de definiciones de Arty. Para la Arty A7 y este proyecto, este será el archivo de definiciones.

10 – Genere el archivo de salida para programación del FPGA

En este punto estamos listos para genera el archivo que en Xilinx es un bit file, necesario para que nuestro FPGA opere.

Para esto necesitamos un archivo en el nivel superior. En este caso el archivo de botones y LED.

Adicionalmente un archivo de constraints que relaciona los pines con su proyecto.

Seguidamente realizar la corrida de síntesis. Lo que hace la síntesis es verificar que la sintaxis y asignación es la correcta, sería el equivalente a compilación en microcontroladores.

Cuando la síntesis completa enteramente significa que el sintetizador genero el diseño sin inconvenientes.

Si su diseño no cabe (por cantidad lógica de compuertas o LUT) entonces la síntesis fallará. Para este diseño sencillo quedará emplazado en el componente pues es de pocos recursos.

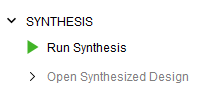


Figura 1.30 – Ejecución de Síntesis. Este es uno de los procesos más importantes en el diseño digital con FPGAs.

11 – La síntesis en ejecución puede verse en la pantalla Project Summary, y tenemos entonces que nos puede dar advertencias las cuales debemos atender si son muy importantes, de lo contrario, la ventana se muestra como sigue:

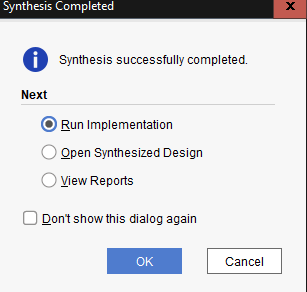


Figura 1.31 – Finalización de síntesis satisfactoria. Puede dar advertencias.

12 – Como se mencionó anteriormente la síntesis puede pasar pero puede ser que al correr la implementación no quepa en el FPGA. Seguidamente se muestra la pantalla de información de la corrida de implementación.

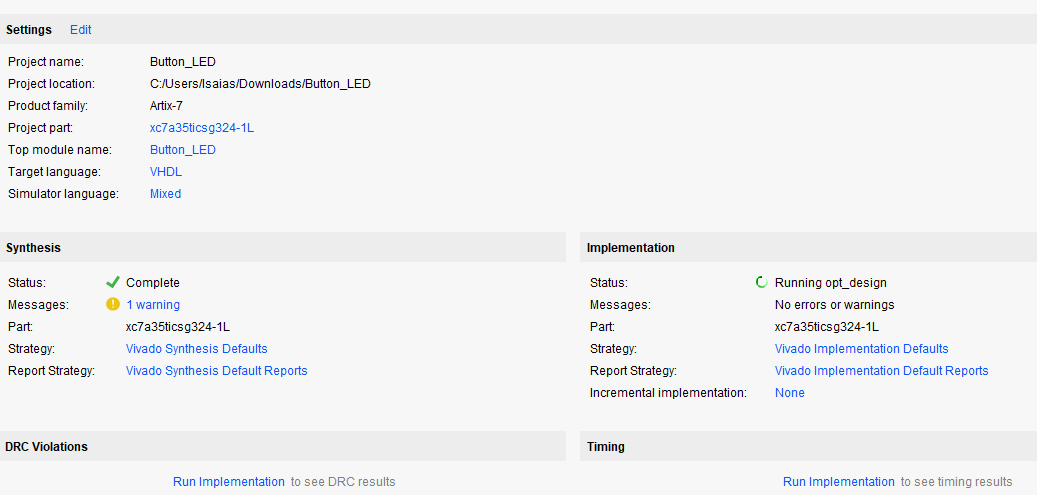


Figura 1.32 – Pantalla de sumarización de proyecto. Actualmente con ejecución de implementación.

13 – Debería mostrarse la siguiente pantalla si completa exitosamente

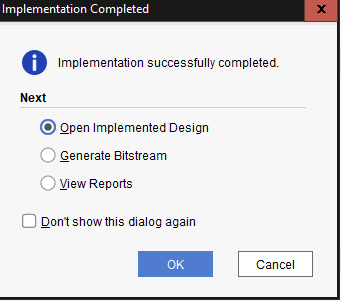


Figura 1.32 – Implementación Exitosa. Luego de este paso se puede ir a generar el bitstream.

14 – Para generar el archivo de salida en el FPGA tenemos que generar el archivo bitstream seleccionando su implementación. Nuevamente, en la ventana de Project summary se puede ver el resultado de esto.

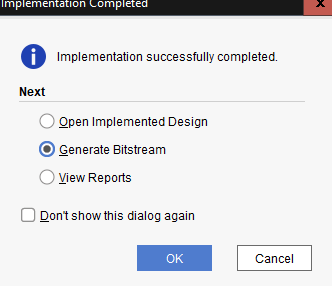


Figura 1.33 – Generación del Bitstream. El bitstream es el último paso del flujo de diseño continuo para programación de la tarjeta.

15 – Seguidamente se abrirá la siguiente ventana cuando complete y nos preguntará la siguiente situación, en la cual eligiremos abrir el hardware manager.

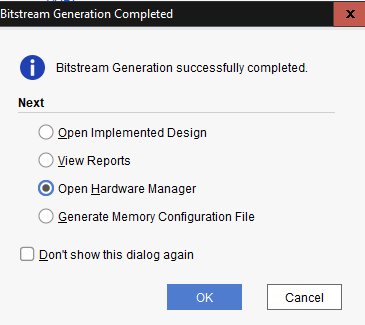


Figura 1.34 – Completada la fase de Bitstream. Ahora procedemos a realizar otra acción, p.e., la acción final sería implementar en la tarjeta.

16 – El hardware manager es el proceso final de la implementación del diseño en la tarjeta real y se observa en la ventana principal si se posee una tarjeta de desarrollo de Xilinx.

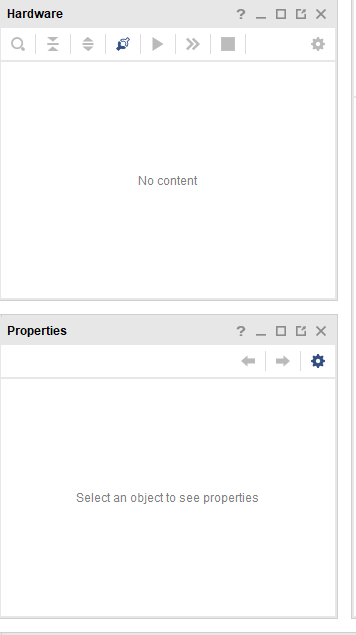


Figura 1.35 – Ventana de Hardware Manager. Actualemente no se tiene conectada la tarjeta, se verificará más adelante su implementación.

Por el momento cerraremos esta ventana y observaremos que advertencias nos dio el proyecto para ver que tan importantes son.

17 – Una de las advertencias es como sigue:

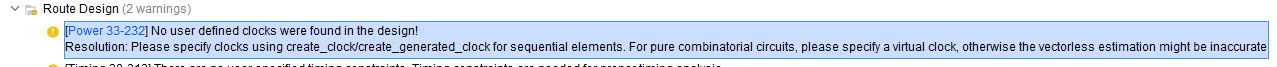


Figura 1.36 – Ventana de Mensajes. Esta ventana está en la parte inferior de su proyecto. Muestra advertencias, errores, logs y demás situaciones relevantes al diseño.

Esto no significa que no funcionará el diseño sino es una señal que nos avisa (del logger de Vivado) que no hemos implementado una señal de reloj, lo cual es muy común para VHDL

18 – Para programar la Arty debe asegurarse de energizarla por el cable USB. Verificar que al conectar se encienda el LED DONE (LD11) que es el de alimentación de la tarjeta.

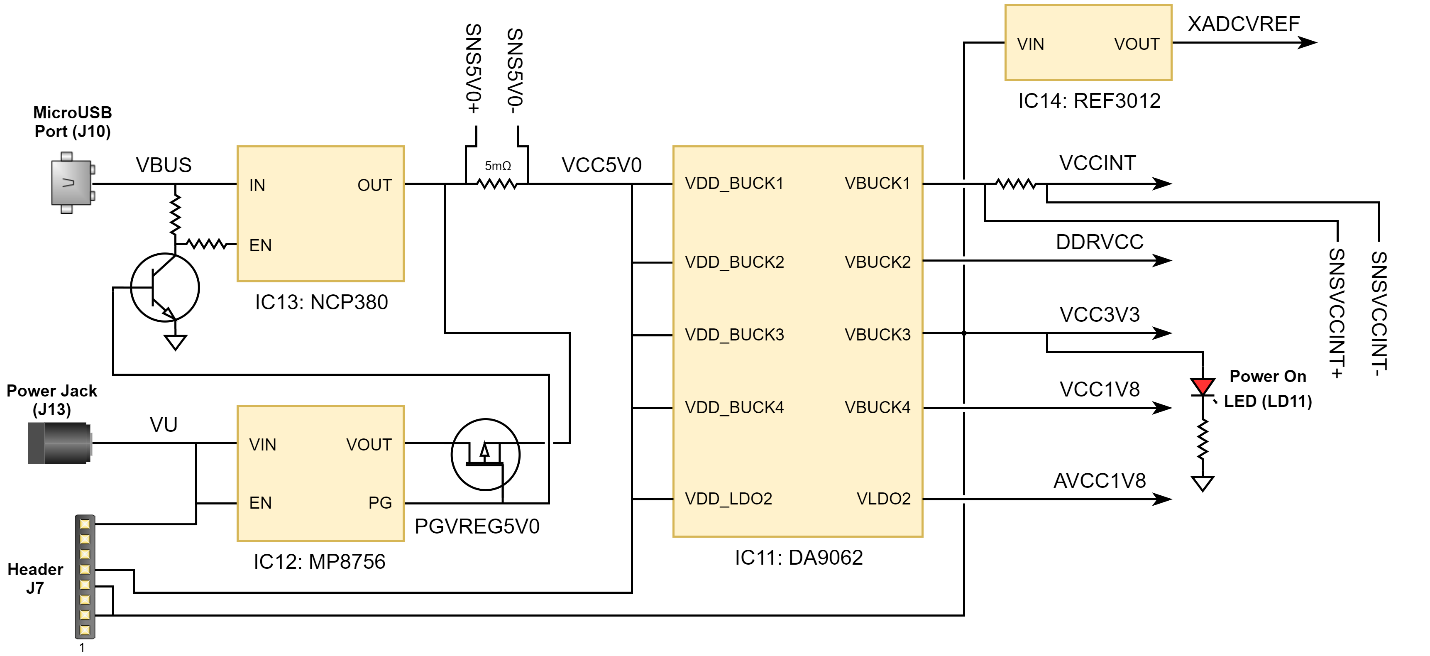


Figura 1.37 – Esquemático de circuito de fuente de poder de Arty A7.

Programación del BitStream en el FPGA:

19 – Conecte primeramente su tarjeta por medio del cable micro USB a la PC

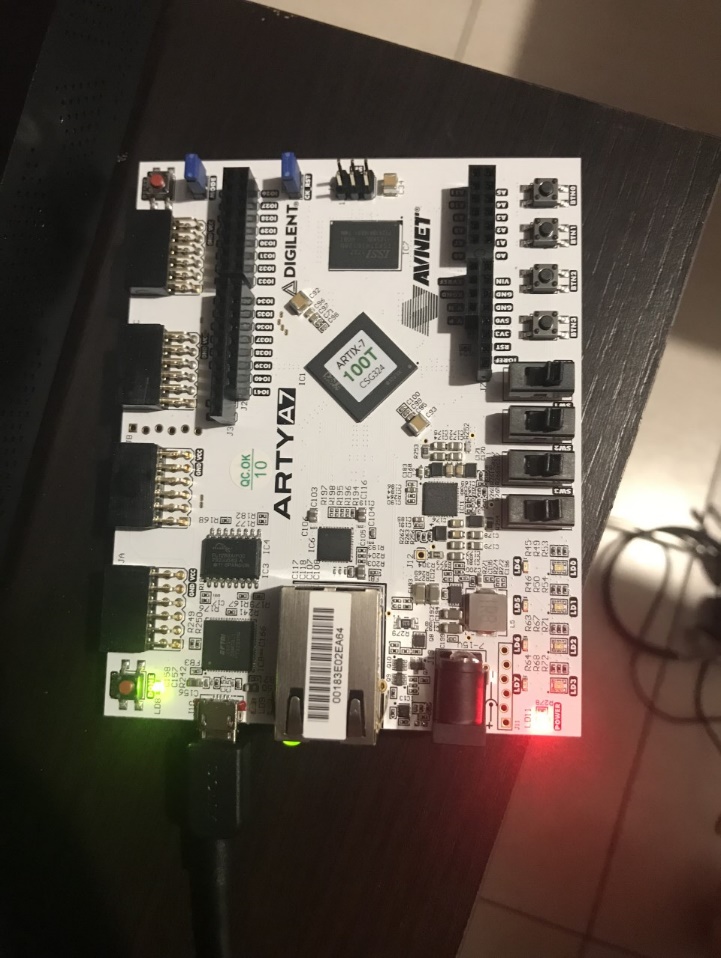


Figura 1.38 – Muestra de nuestra tarjeta Arty A7. Cable de USB conectado.

Debe observar los LEDs rojo (PWR) verde (no programado) y el de Ethernet (no importante aún).

20 – Abrir la ventana de Hardware Manager

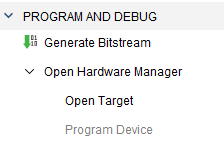


Figura 1.39 – Fase final, apertura del Programador.

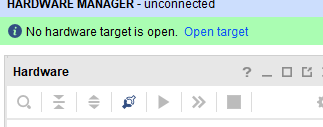


Figura 1.40 – Ventana de detección del dispositivo.

21 – Hacer click en Open Target y luego en Auto-Connect. Esta función solo funcionará en Vivado con tarjetas de Xilinx. Cualquier otro fabricante, aunque el proceso es el mismo, los archivos generados y las herramientas son diferentes sin embargo, el flujo de diseño en FPGAs es coincidente.

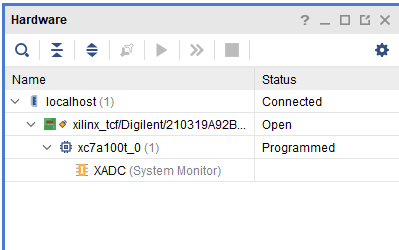


Figura 1.41 – Ventana de Auto Conexión. Observamos que ha detectado el componente.

22 – Encontrada la tarjeta observamos de la figura anterior que el FPGA viene programado de fábrica con una configuración. Nosotros la sobreescribiremos. Hacer click en Program Device.

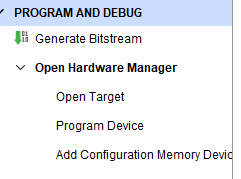


Figura 1.42. Ventana de Programación. Al realizar Program device realizaremos la copia del archivo en la tarjeta.

23 – Seleccione program Device. Busque el archivo de configuración en la carpeta de proyecto en runs/impl\_1. El archivo \*.bit estará dentro de esta carpeta.

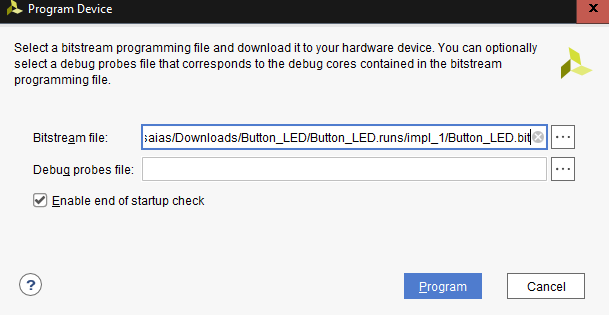


Figura 1.43 – Ventana de Selección del BitStream. Como observa las impleemtaciones están en la carpeta \*.runs/impl\_\*/\*.bit.

24 – En caso personal, se tuvo que realizar nuevamente la sintetización, implementación y síntesis debido a que la pieza escogida no era la adecuada referente a la que se debía programar

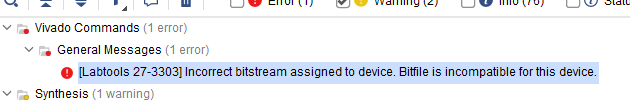


Figura 1.44 – Error en escritura del bitstream. Como el componente es diferente al programar, por consiguiente el bitstream generado no encaja en el FPGA, se debe de volver a realizar los procesos.

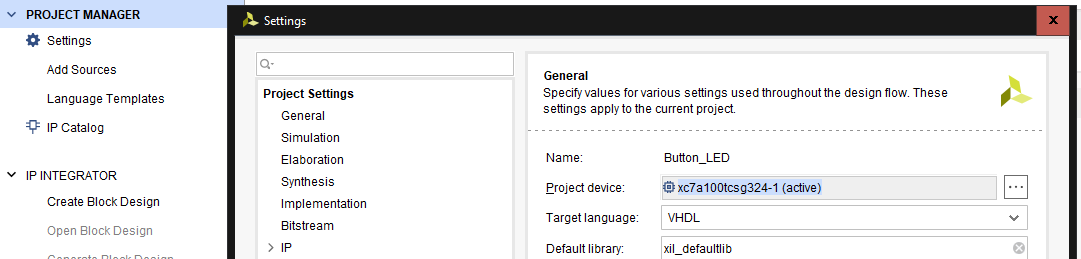


Figura 1.45 – Cambio de pieza. Esto se puede realizar fácilmente en los ajustes del proyecto, opción del dispositivo de proyecto.

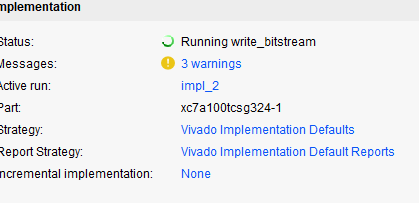


Figura 1.46 – Resultado de la ejecución de implementación.

25 – El resultado de la implementación ha sido exitoso y ha sido programado el FPGA

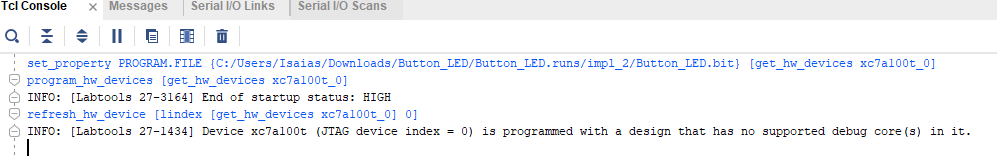


Figura 1.47 – Resultado final de la programación del dispositivo.

26 – Pruebe la tarjeta:

* SW0 en su posición normal es el enable
* Los botones BTN0 a BTN3 controlan los LEDs LED0 a LED3
* Observe la función del Enable y que se puede realizar paralelamente el encendido de LEDs.



Figura 1.48 – Test del programa de botones y LEDs.

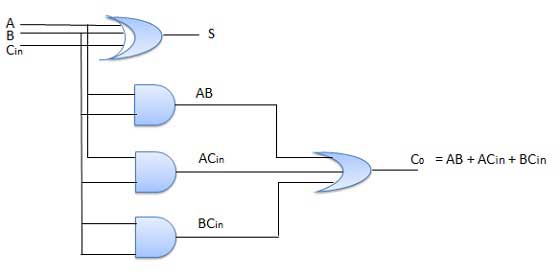
Evaluación:

50% - Completar todos los pasos anteriores:

* Simulación de registro de corrimiento.
* Ejecución del programa de botones y LED en un FPGA.

50% - Realizar los siguientes cambios

* Para el siguiente circuito
  + Escriba el Código en VHDL
  + Simule
  + Implemente en el FPGA



* Cambie su proyecto de Botón y LED para que sea por medio de interruptores, Enable = BTN0:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | SW0 | SW1 | SW2 | SW3 | Enable (BTN) |
| 1 | LED | 1 | 2 | 0 | 2 | 0 |
| 2 | LED | 3 | 2 | 1 | 0 | 1 |
| 3 | LED | 0 | 1 | 3 | 0 | 0 |
| 4 | LED | 0 | 2 | 2 | 1 | 1 |
| 5 | LED | 0 | 3 | 0 | 2 | 2 |
| 6 | LED | 3 | 1 | 2 | 2 | 2 |
| 7 | LED | 0 | 1 | 2 | 1 | 3 |
| 8 | LED | 1 | 0 | 3 | 2 | 2 |
| 9 | LED | 0 | 2 | 3 | 3 | 0 |
| 10 | LED | 1 | 2 | 3 | 2 | 2 |
| 11 | LED | 0 | 2 | 1 | 1 | 3 |
| 12 | LED | 2 | 1 | 2 | 3 | 1 |
| 13 | LED | 3 | 1 | 1 | 1 | 1 |
| 14 | LED | 2 | 3 | 3 | 2 | 2 |
| 15 | LED | 3 | 2 | 2 | 2 | 2 |
| 16 | LED | 3 | 2 | 3 | 2 | 1 |
| 17 | LED | 3 | 2 | 2 | 0 | 0 |