***Laboratorio 4 – Brillo en LEDs***

Objetivos del laboratorio son conocer:

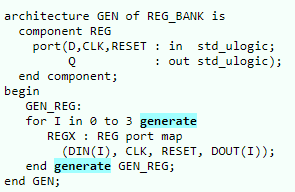
* Como hacer hacer generates
* Realizar instanciamiento de entidades en un archivo de entidad global

¿Cómo generate es utilizado en VHDL?

En esta sección realizaremos un *contador* que finalmente organizaremos en nuestra estructura final, un archivo llamado que controla el brillo de los LEDs de la tarjeta Arty A7.

En VHDL el token “generate” corresponde a es un mecanismo para elaboración de porciones descripciones iterativas. En palabras comunes, iterativamente posiciona un objeto. Se puede utilizar en un lazo “for” o “if”, consta de dos partes, la parte iterativa y la parte generativa.

Podemos entonces realizar instanciamiento de las entidades en un archivo VHDL realizando “copy” y “paste” pero sería mejor realizarlo con la estructura “generate”. En este caso lo utilizaremos para realizar un contador síncrono y asíncrono, pero antes, observaremos la estructura de una sentencia “generate”.



Listado 1.1. Muestra de una sentencia generate. Note que para generate podemos usar “if”, pero en este ejemplo estamos utilizando “for”.

Veamos un mejor uso de esta sentencia. Imaginemos que queremos realizar concatenamiento muchas compuertas AND como se muestra en la figura inferior:

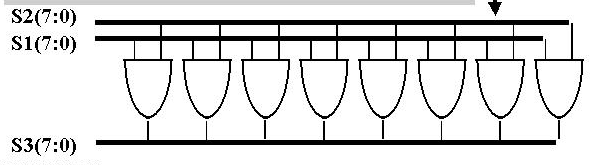
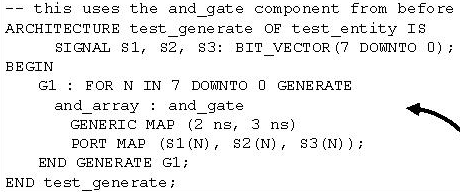


Figura 1.1. Grupo de Compuertas AND.

Para la figura anterior podríamos en VHDL empezar realizando llamado de componente a compontent haciendo port mapping una por una, sin embargo, como vemos en el siguiente código tenemos un resultado más eficiente si escribimos todo utilizando la sentencia generate.



Listado 1.2. Realización de compuertas AND utilizando la sentencia Generate.

Otro ejemplo utilizando if para generate será por ejemplo, si queremos tener dos tipos de compuerta en un solo diseño, por ejemplo un flip flop tipo J-K y un flip flop tipo D.

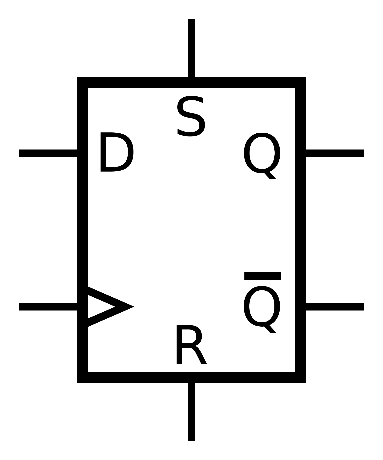
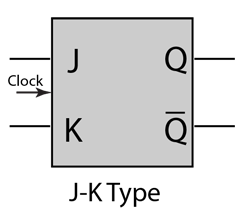


Figura 1.2. Flip flops JK y Flip flop tipo D.

La generación sería algo parecido a esto para el caso superior (dos entidades en una arquitectura con generate)

library ieee;

use ieee. std\_logic\_1164.all;

use ieee. std\_logic\_arith.all;

use ieee. std\_logic\_unsigned.all;

entity FF is

PORT( D, J, K, CLOCK: in std\_logic := '0';

Q, QBAR: out std\_logic := '0');

end FF;

Architecture behavioral of FF is

begin

-- JK Flip Flop Generate

if JK = true generate

PROCESS(CLOCK)

variable tmp: std\_logic;

begin

if(CLOCK='1' and CLOCK'EVENT) then

if(J='0' and K='0')then

tmp := tmp;

elsif(J='1' and K='1')then

tmp := not tmp;

elsif(J='0' and K='1')then

tmp :='0';

else

tmp :='1';

end if;

end if;

end PROCESS;

end generate;

-- D Flip Flop

if JK = false generate

process(CLOCK)

begin

if(CLOCK='1' and CLOCK'EVENT) then

tmp <= D;

end if;

end process;

end generate

Q <= tmp;

QBAR <= not tmp;

end behavioral;

Listado 1.3. Generación de dos flip flop, a utilizar, usando generate.

Para aplicar lo anterior veremos como generar un contador. Realizaremos un contador pero con ciertos aditivos usaremos entonces dos ejemplos de un contador asíncrono o síncrono y también la sentencia generate para realizar estos cambios.

Cuando utilizamos contadores necesitamos pensar en que tenemos varias entradas, nos centraremos en un contador de cuenta ascendente. Necesitamos entonces:

CLK: Entrada de reloj. Esta entrada a cada flanco de reloj de subida nos dará un incremento de la cuenta.

RST: Entrada de reinicio de de la cuenta. Más adelante se explicará que efecto tiene un contador síncrono vs asíncrono aquí.

Max\_Count: Salida que nos indica que la cuenta máxima fue lograda, es un pulso.

Adicionalmente utilizaremos un genérico que estará prefijado el valor máximo de la cuenta para nuestra facilidad.

*Contadores*

Un contador es un circuito digital que cuenta desde un valor inicial, generalmente 0 hasta su cuenta máxima y luego vuelve a reiniciar. En este caso realizaremos dos contadores en una sola arquitectura. De la siguiente figura inferior ‘Asinc’ simboliza la salida de cuenta con reset asíncrono y así mismo se dará el caso para la salida ‘Sinc’. ‘Clk’ es la entrada de reloj y ‘Rst’ será la entrada de reinicio del sistema.



Figura 1.3 – Reset asíncrono, si ocurre un evento de reinicio, el contador inmediatamente realizará un reinicio y se repondrá el contador a 0, la cuenta empezará cuando se de el siguiente flanco de subida del reloj.

Un reinicio asíncrono se da en cualquier momento, es decir, no importa el flanco de subida o bajada del reloj, necesitamos tener un reinicio de la cuenta cuando ocurra esta señal de reset. En el caso del contador síncrono, solamente se da el reinicio de la cuenta cuando la señal de reset es detectada con un flanco de subida.

*1 - Creación del Proyecto de Counter*

1.1 - Abrir Vivado en la pantalla principal y presionar en abrir proyecto

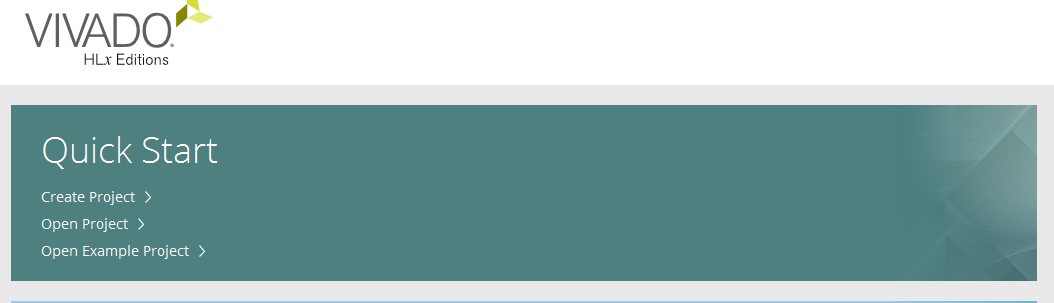
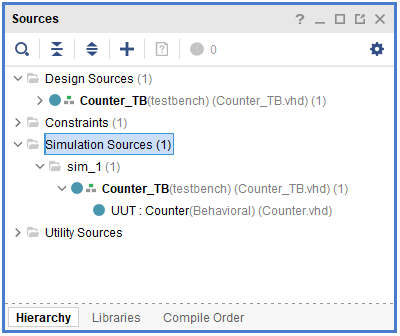


Figura 1.4 – Creación del proyecto en Vivado.

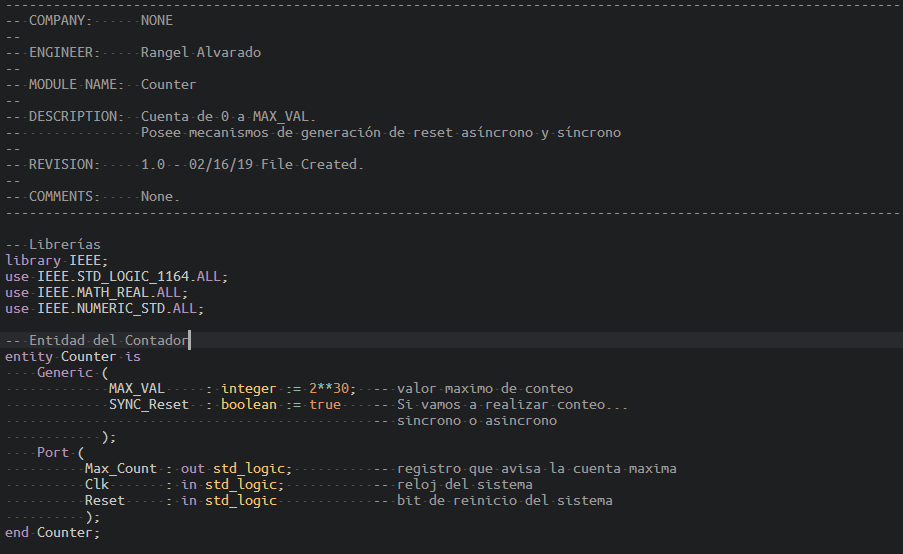
1.2 – Del proyecto suministrado abra el Project manager, deberá haber terminado en una pantalla similar a esta:

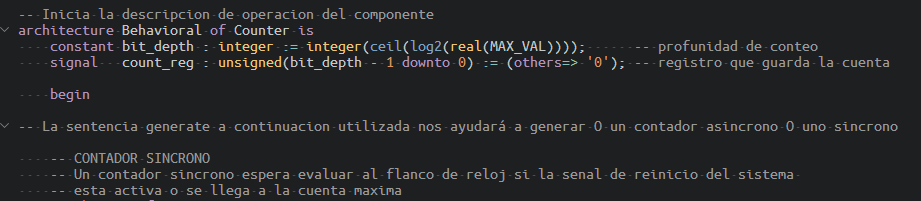


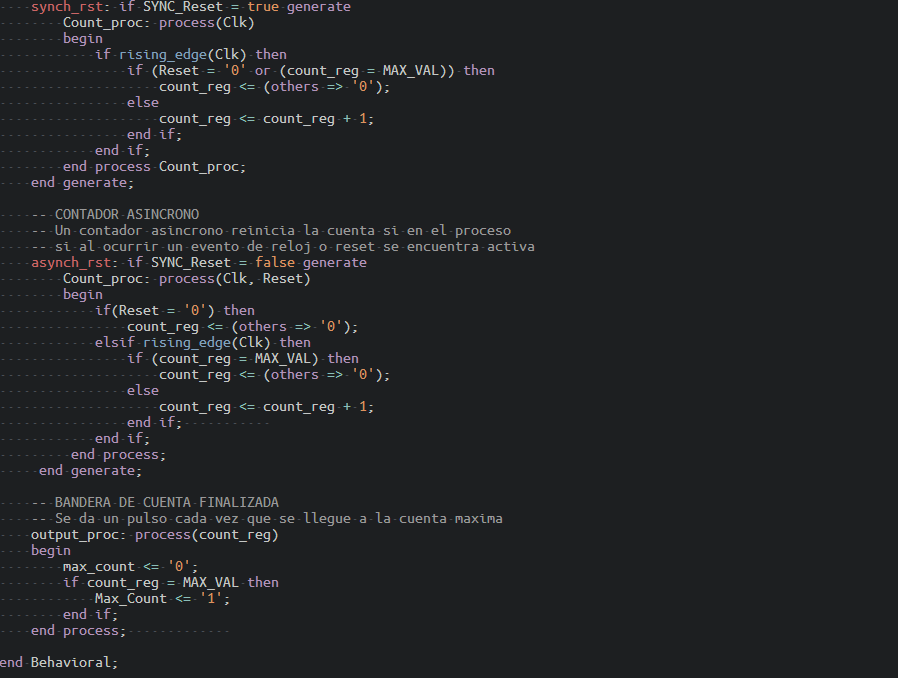
*Figura 1.5 – Archivos de trabajo de proyecto*

1.3 – Comience y escriba en VHDL el siguiente código. Note que utilizaremos una entrada SYNC\_Reset que nos servirá para realizar el generate y conseguir un contador síncrono vs asíncrono.

*Listado 1.4 – Creación del archivo Counter.vhd.*

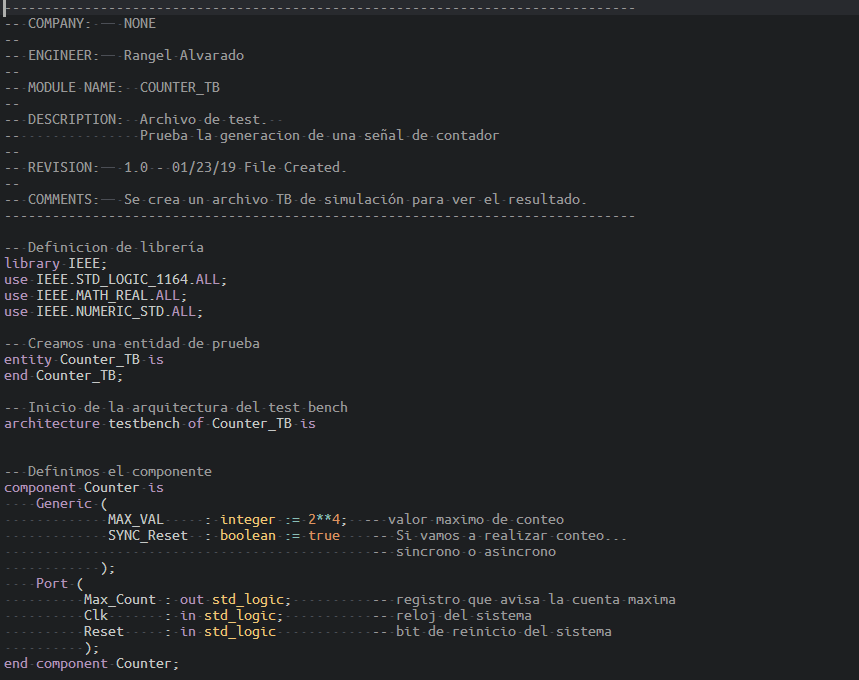


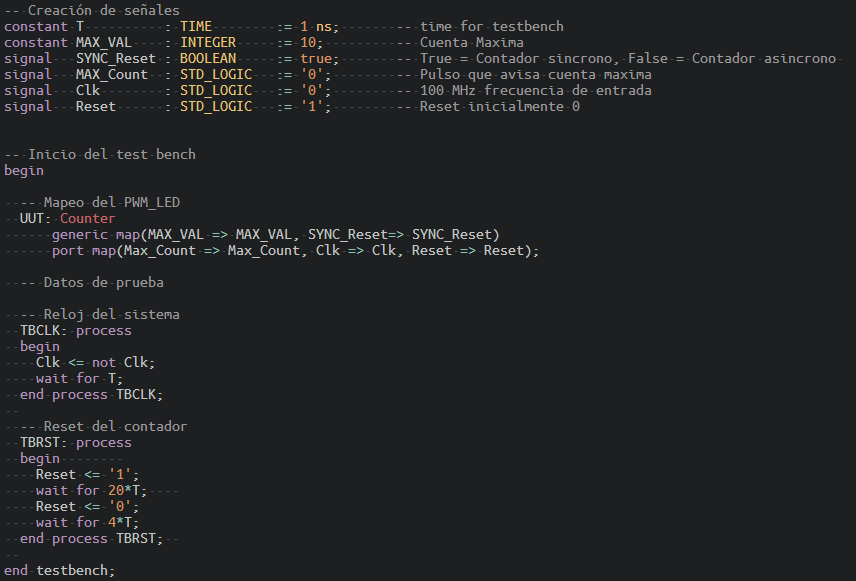




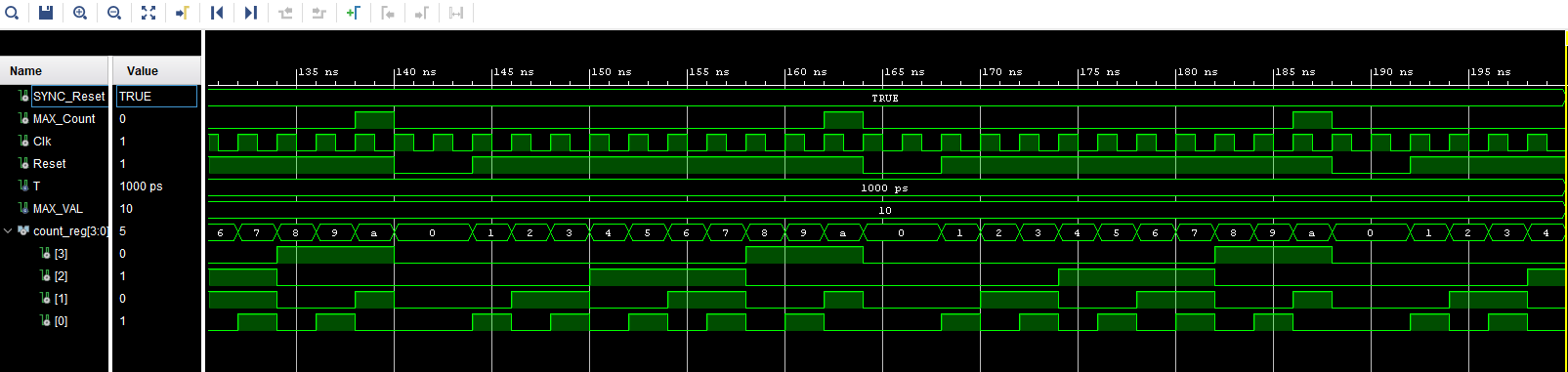
1.4 – Ahora comience a realizar el siguiente archivo de prueba de la entidad.

*Listado 1.5 – Creación del archivo Counter\_TB.vhd.*





1.5 – Simule. Debería poseer una salida parecida a esta



*Figura 1.6 – Resultado de la simulación.*

Observe que la señal count\_reg cuenta de manera binaria 000 001 010… 0101 0000. Note que pasa cuando se genera la señal de reset, en este caso generamos un contador asíncrono.

También a destacar la bandera MAX\_Count que se eleva al registrar la cuenta máxima.

*2 – Intanciamiento de partes (componentes o arquitecturas).*

Permite añadir más archivos VHDL en el proyecto. Si se tienen múltiples diseños entonces siempre habrá un diseño superior de entidad y se mapearán las otras entidades dentro de este diseño. Los únicos pines que saldrán de esta arquitectura superior (o entrarán) serán la arquitectura inicial y final, los otros componentes serán señales de la misma entidad principal.

En resumen, tendremos un envolvente de sub arquitecturas en esta entidad principal.

Tome en consideración que cuando se hace instanciamiento se utilizan más recursos de la tarjeta (más componentes lógicos).

El siguiente diagrama muestra más a detalle lo anterior descrito en palabras.



*Figura 2.1 – Resultado de la simulación.*

2.1 – Abrir el archivo de “LED\_Brightness – Start Here”. Al final de concluir esta sección ud debe terminar como sigue:

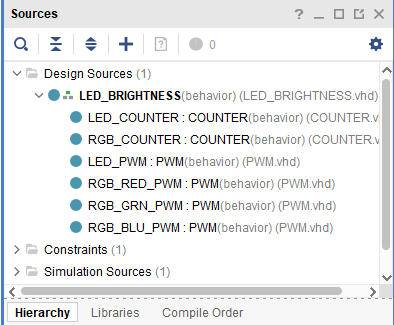


Figura 2.2 – Archivo final de LED\_Brightness. Todos los components intanciados.

2.2 – Iniciaremos entonces con un archivo como sigue:

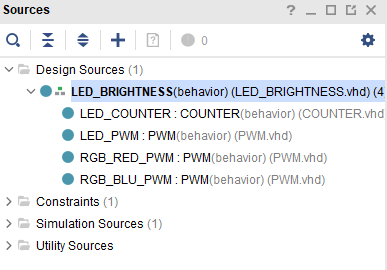


Figura 2.3 – Archivo inicial de LED\_Brightness. Faltan instanciar componentes

2.3 – Doble click en el archivo de VHDL de LED\_BRIGHTNESS.VHD, observe su contenido. Note que tenemos varias entradas, varias salidas y que usamos algunos switches y leds para señalizar.

2.4 – Navegue hasta la línea 117 y observe que faltan componentes a instanciar.

2.5 – Hacer doble click en la instancia LED\_COUNTER. Note que al hacer click nos lleva al archivo original de contador.

2.6 – Hacer doble click en la instancia RGB\_RED\_PWM y Repita lo mismo para RGB\_BLU\_PWM. Note que en este caso abre el mismo archivo de VHDL. Esta es la ventaja de intanciar, no tenemos que escribir el mismo código una y otra vez, solo hacer una llamado. *A notar que para instanciar debemos tener los archivos VHDL ingresados al proyecto.*

2.7 - Ahora realizaremos instanciamiento de los componentes faltantes. Utilice las siguientes líneas de código para instanciar cada parte.

*Listado 2.1 – Componentes a instanciar en arquitectura superior LED\_Brightness.vhd.*

*-- RGB Counter*

*RGB\_COUNTER: COUNTER*

*generic map(RGB\_MAX\_COUNT, SYN\_RESET)*

*port map(Max\_Count => rgb\_max\_cnt, Clk => Clk, Reset => rgb\_counter\_rst);*

*-- RGB LED PWM Signal Generator [GREEN] (4 bit, 50Hz)*

*RGB\_GRN\_PWM: PWM*

*generic map(4, INPUT\_CLK, 50)*

*port map(grn\_pwm\_reg, std\_logic\_vector(rgb\_duty\_cycle(7 downto 4)), Clk, RGB\_Enable);*

2.8 – Salve y observe que se agregaron los componentes, ya instanciados.

2.9 – Abrir el archivo XDC y observar la asignación ya puesta de entradas y salidas.

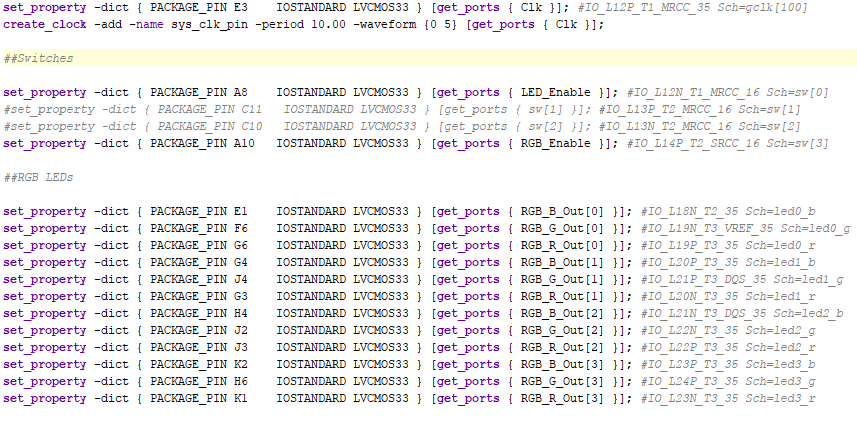
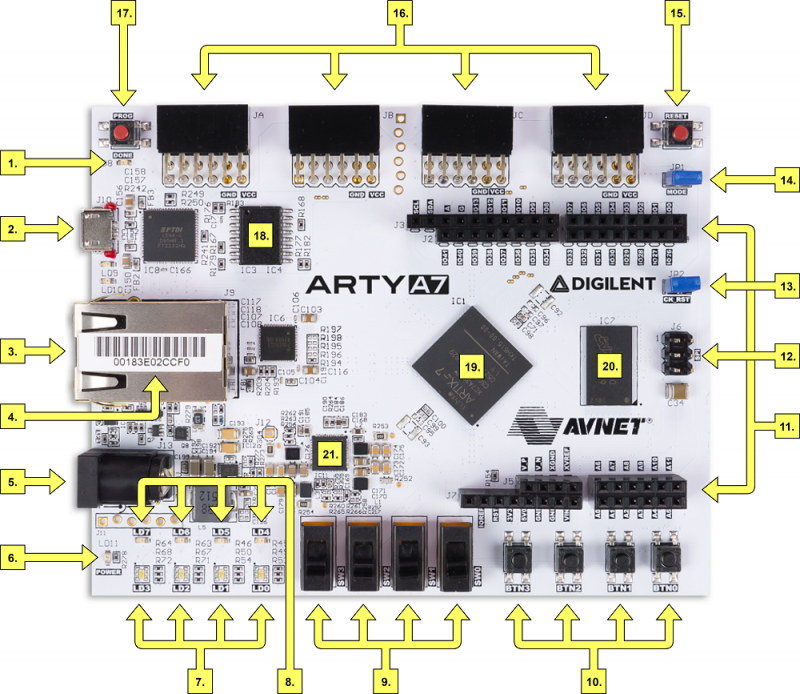


Figura 2.2 – Archivo inicial de LED\_Brightness. Faltan instanciar componentes

*3. Arty A7*

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

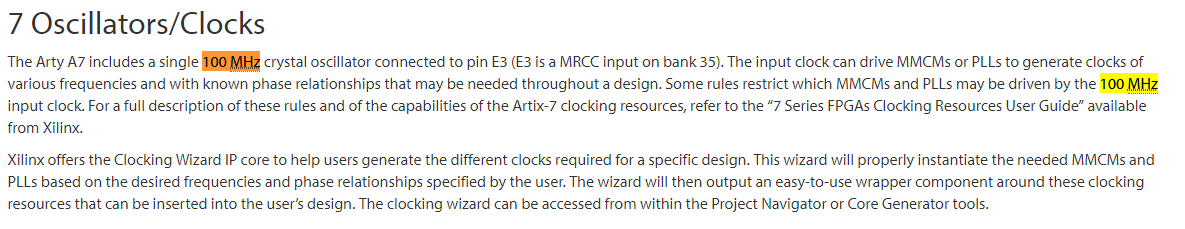
*Figura 1.5 – Componentes de Tarjeta Arty A7*

Tabla 3.1 – 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 |

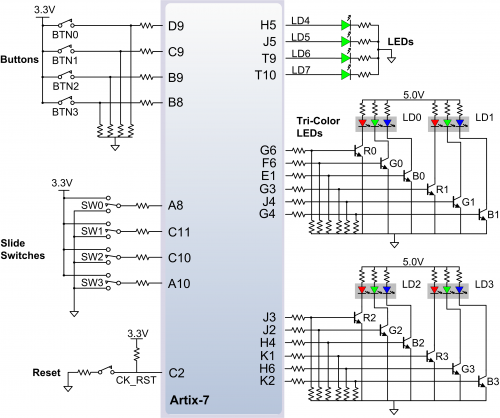
*Reloj / Oscilador:*

Según la documentación de la Arty, posee un cristal que corre alrededor de 100MHz y esto es de considerar al momento de realizar el ingreso de MAX\_VALUE



*Figura 3.2. En la documentación de la Arty A7 detalla que el reloj es de 100 MHz.*

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



*Figura 1.16 – Muestra de ubicación de los botones y LEDs en la Arty A7*

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

3.3 - Abrir el archivo XDC y observe solamente como están declaradas las entradas y las salidas a utilizar. Piense porqué se habilita esta entrada o esta salida, igualmente puede verificarlo al momento de que la tarjeta tiene descargado el código.

4 - Evaluación

50% - Completar todos los pasos anteriores:

* 10% Ejecución del programa de PWM de LED en un FPGA.
* 40% Simulación y su TestBench

50% - Realizar lo siguiente

* Demostrar el proyecto andando en la tarjeta de FPGA