***Laboratorio 2 – Circuitos Combinacionales***

Objetivos del laboratorio son conocer:

* Realizar un contador y divisor de frecuencia [No es un circuito combinacional sino secuencial]
* Realizar un multiplexor
* Para cada caso sintetizar y escribir el test bench
* Realizar estos bloques básicos para tomar y tener proyectos más complejos

*Parte 1 – Divisor de Frecuencia*

*¿Qué es un divisor de frecuencia?*

Este no es parte de un circuito combinacional, pero forma parte de un bloque básico para la creación de proyectos en VHDL y es muy utilizado para visualizar. Debido a que el reloj utilizado de la Arty es de 100MHz, la frecuencia interna de los módulos (algunos) es de 10ns. En esta primera parte del laboratorio utilizaremos un divisor de frecuencia para lograr una frecuencia de 1s a partir del reloj principal.

Si bien es cierto este es un método rudimentario existen otros métodos analógicos más eficientes de generación de reloj como el PLL (Phase Shifted Lock) donde se puede enganchar una frecuencia interna mayor en base a un cristal o algún resonador externo menor físico a su entrada.



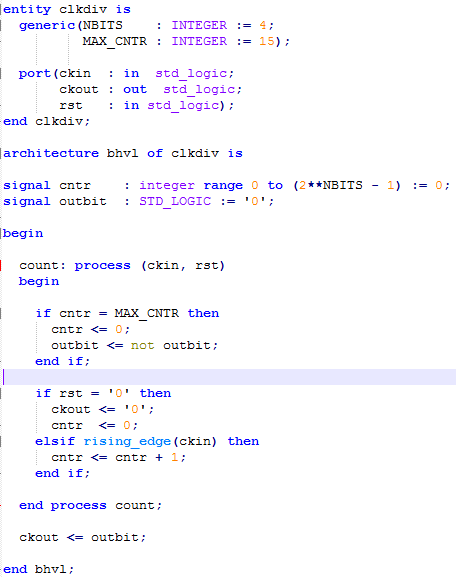
Figura 1. Diagrama de Bloques del Divisor de Frecuencia. Implementaremos el Bloque del Medio.



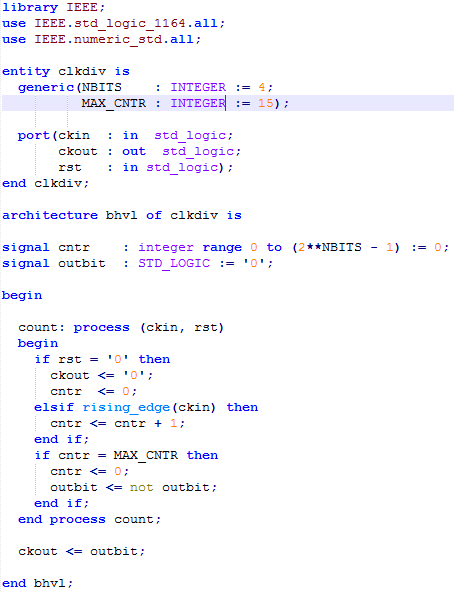
Figura 2. Representación Pictórica del Divisor de Frecuencia. La frecuencia de entrada es dividida proporcionalmente a la cuenta de pulsos deseada.

El siguiente apartado se refiere al código en VHDL para un divisor de frecuencia

Listado 1. Divisor de Frecuencia en VHDL

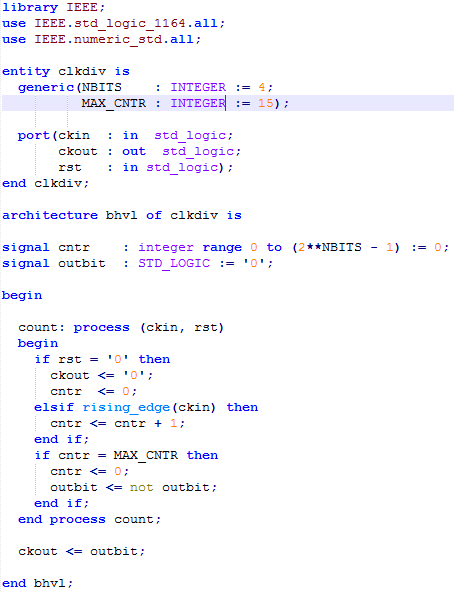


Explicaremos los apartados más importantes en cada caso. Por ejemplo en el caso inferior, la librería para operar valores numéricos es numeric\_std, esta la utilizaremos para el genérico en MAX\_CNTR más adelante y NBITS.

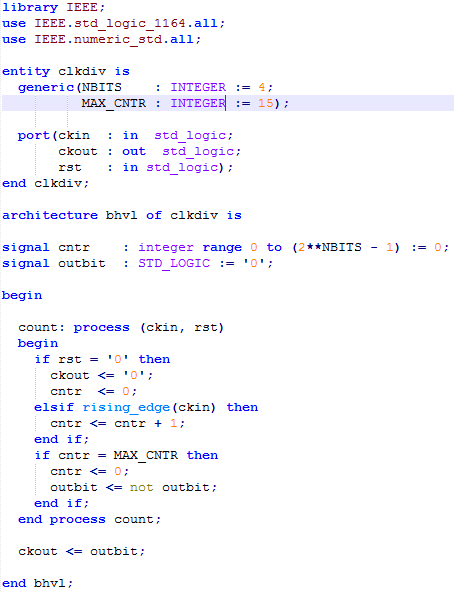


Dentro de la declaración original del componente tenemos los genéricos.

Los genéricos permiten que se describa una entidad de diseño para que, para cada uso de ese componente, su estructura y comportamiento puedan modificarse mediante valores genéricos. En general, se utilizan para construir componentes de hardware parametrizados. Los genéricos pueden ser de cualquier tipo. En resumen, estos permiten ampliar las capacidades de la entidad, p.e., una and de 4 entradas solamente cambiando un genérico.



Seguido tenemos las señales. Como se mencionó en el laboratorio anterior las señales son conexiones internas dentro del componente de VHDL, estas sirven para conectar más componentes lógicos entre si o para llevar señales intermedias a la salida. También otro uso de las señales es para llevar registro de variables internas, no necesariamente asignadas a la salida (como el contador ‘cntr’). Observamos también que tenemos el rango máximo dependiente del genérico, el cual ayuda mucho a parametrizar la cuenta máxima del divisor de frecuencia.

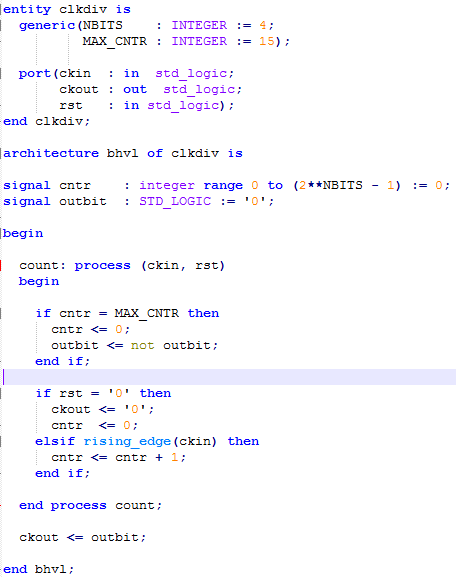


Ya finalmente tenemos la declaración de la arquitectura. Observemos que existe un proceso de etiqueta ‘count’ que tiene una lista de señales sensitivas ‘ckin’ y ‘rst’. La lista sensitiva sirve para la detección de cambios, por ejemplo. Si el reloj cambia de flanco, en este caso se activa el proceso inmediatamente y nos lleva a la ejecución interior del mismo.

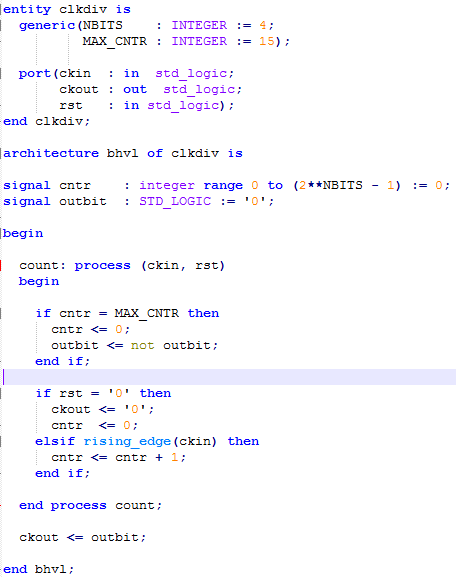
Como observamos detectamos el cambio de la señal de reloj con ‘rising\_edge’ que es una función equivalente a lo que han acostumbrado a ver como ‘ckin’event and ckin=1’, tiene la misma equivalencia.

Si el botón de reset es presionado en cualquier momento la cuenta es reiniciada y el reloj va a bajo.

Adicionalmente se realiza la cuenta cada vez que se detecta cambio en el reloj de entrada y finalmente si la cuenta llega a máximo esta es reiniciada a 0 y la señal ‘outbit’ cambia de nivel (toggle).

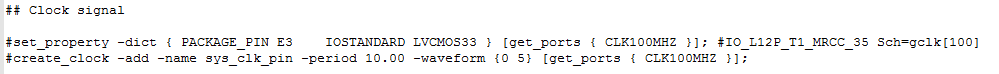


Finalmente y no menos importante asignamos el cambio de la señal a la salida



NOTA: Al momento de crear el bitstream del contador cambie las entradas y salidas del sistema en el archivo XDC.

Para esto recuerde cambiar la entrada a la del reloj del sistema, para el caso de la Arty el archivo XDC tiene la señal asignada a un porting de CLK100MHz, en este caso esta señal será llamada ckin. Recuerde que este archivo es sensitivo a mayúsculas y minúsculas en el nombre de las señales.



*Parte 2 – Multiplexor*

En esta sección generaremos un multiplexor. El multiplexor es un elemento digital, del cual podemos derivar ecuaciones lógicas. La función principal es derivar de un grupo de entradas la entrada seleccionada a la salida. Comunmente son de más entradas que salidas. Por ejemplo, un multiplexor puede ser de 8 entradas de 4 bits, pero con un selector de 3 entradas para dar una salida de 4 bits.

Lo siguiente se observa mejor en un ejemplo:



Figura 3. Multiplexor de 8 entradas, 3 entradas de selección y una salida de 4 bits

Por ejemplo estas son las combinaciones posibles:

C[0:2] = “000” entonces D[0:3] = S1[0:3]

C[0:2] = “001” entonces D[0:3] = S2[0:3]

C[0:2] = “010” entonces D[0:3] = S3[0:3]

C[0:2] = “011” entonces D[0:3] = S4[0:3]

C[0:2] = “100” entonces D[0:3] = S5[0:3]

C[0:2] = “101” entonces D[0:3] = S6[0:3]

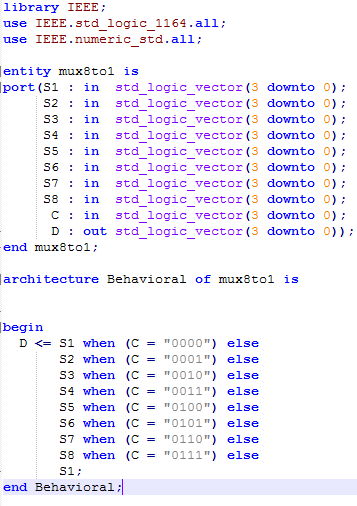
C[0:2] = “110” entonces D[0:3] = S7[0:3]

C[0:2] = “111” entonces D[0:3] = S8[0:3]

Podemos también tener una entrada de selección ENB, pero esta es fácil de habilitar y no es importante ahora mismo, concentrémonos en la funcionalidad.

Para implementar un multiplexor en VHDL tenemos el siguiente código:

Listado 2. Multiplexor de 8 entradas, 3 entradas de selección y una salida de 4 bits

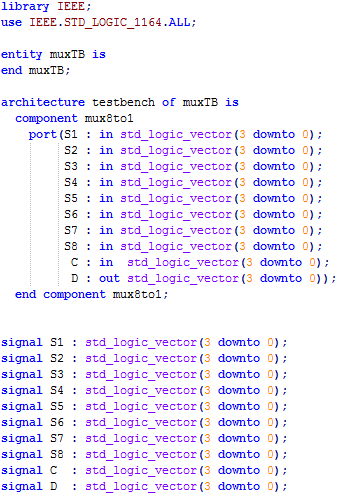


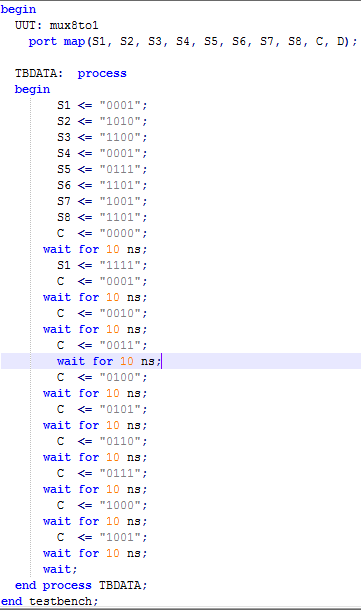
Del listado anterior lo únicoco que hay que destacar es la sección que existe debajo de begin. Luego de construir la arquitectura obtenemos un multiplexor fácilmente con este estilo, la salida ‘D’ es asignada a cada valor al cambio de la entrada de selección ‘C’, cuando ‘C’ cambia de valor, por ejemplo a ‘0010’ la dalida ‘D’ tendrá el valor D = S3. ¿Qué sucederá a combinaciones como por ejemplo “1101” (11)? Pues la última línea resuelve que para cualquier valor no declarado en la tabla de prioridades este se convertirá en D = S1.

También existen otras maneras de escribir este mismo multiplexor en VHDL. Las tres maneras posibles son por medios secuenciales, concurrentes y con arreglos de bits; la actualmente utilizada es la más sencilla que es la concurrente. Obervar otras implementaciones de: <https://bit.ly/2Inic3e>

El testbench para comprobar o validar el multiplexor es el siguiente:

Listado 3. Test Bench para el Mux8to1 en VHDL.





Sin mucho que aclarar en este apartado podemos observar solamente que dictaminamos valores iniciales para S1, S2, S3, …, S8 y que luego de la primera iteracción de señales fijamos otro valor para S1. Note también que las señales de S1 a S8 tienen valores arbitrarios, sin ningún significado, solo para validar.

Finalmente ud. debería ver la forma de onda de salida a algo parecido a continuación en Vivado.

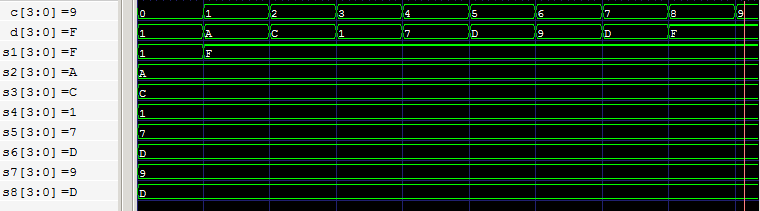


Figura 4. Resultado de Evaluación del Test Bench.

*Parte 3 – Decodificador de BCD a 7 segmentos*

Un decodificador es un circuito combinacional, cuya función es convertir un código binario de entrada (natural, BCD, etc.) de N bits de entrada y M líneas de salida (N puede ser cualquier entero y M es un entero menor o igual a 2N), tales que cada línea de salida será activada para una sola de las combinaciones posibles de entrada. Normalmente, estos circuitos suelen encontrarse como decodificador / demultiplexor. Esto es debido a que un demultiplexor puede comportarse como un decodificador.

El uso más común de los decodificadores y más útil es el decodificador de BCD a 7 Segmentos. Para un decodificador de BCD a 7 Segmentos la entrada viene codificada en BCD y este es traducido a cada uno de los LEDs de salida del dispositivo (7 segmentos) para visualizar el número.

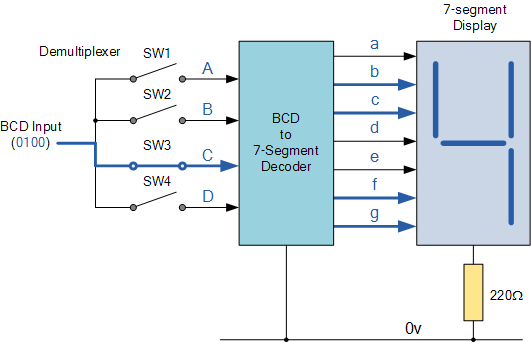
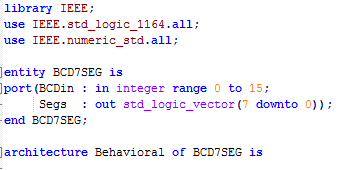
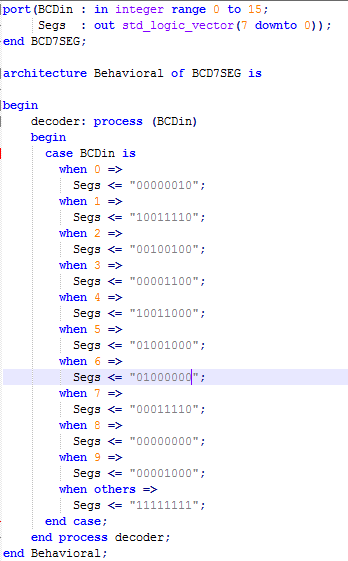


Figura 5. Decodificador (Demux) de BCD a 7 segmentos. Dependiendo del caso tendremos 7 segmentos de ánodo común y de cátodo común.

El siguiente listado en VHDL es para un decodificador de BCD a 7 Segmentos:

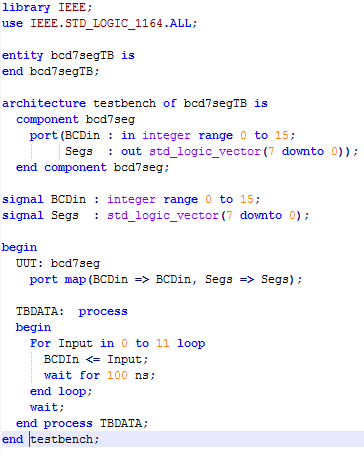
Listado 3. Decodificador de BCD a 7 segmentos





Adicionalmente se le suministra el Testbech:

Listado 4. Testbench de decodificador de BCD a 7 segmentos



Finalmente se muestra el resultado de la gráfica de evaluación del test bench

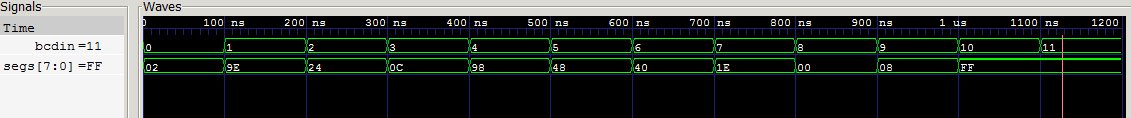


Figura 6. Diagrama de tiempo del bloque de pruebas.

Evaluación Laboratorio 2:

25% - Para el divisor de frecuencia:

* Genere un archivo de simulación para el divisor de frecuencia
* Calcule NBITS y MAX\_CNTR para lograr una frecuencia de salida de 60 Hz. De ser necesario cambie el tipo de variables o el archivo de VHDL
* Compruebe con un osciloscopio y el FPGA.

25% - Para el multiplexor

* Generar un multiplexor de 3 entradas de 2 bits y una salida de 2 bits. Nota: El selector es de menor cantidad al del laboratorio
* Teniendo una salida de 3 leds, generar una frecuencia de 180 Hz con el divisor de frecuencia y encender cada led a 60 Hz
  + Ejemplo: in1 = “001”, in2 =”010”, in3 = “100” y pasar estas salidas cada 60 Hz
* Generar el archivo de simulación en VHDL
* Probar en la tarjeta y con un osciloscopio

50 % - Realizar un decodificador de BCD a 7 segmentos

* Seguir la guía superior y probar en el FPGA
* Expandir la tabla para conseguir además de números las vocales de la ‘a’ a la ‘u’
* Conseguir también lo anterior pero con ecuaciones lógicas (solamente números, no letras)
* Probar ambas partes en el FPGA