TACÓMETRO

INTRODUCCIÓN

El tacómetro es un instrumento de medición que nos permite medir las revoluciones por minuto (rpm) a lo que está girando algún objeto que deseamos sensar, en nuestro caso, un motor Diesel de una máquina agrícola.

En su interior está compuesto por 8 bloques llamados componentes (**Fig. 1**), en donde cada uno de ellos cumple con una función específica, y hacen al funcionamiento del conjunto. Estos bloques fueron descriptos de esta manera (como componentes) para tener una mayor prolijidad al momento de describir el código, facilidad de corrección de errores y mejor visualización de la descripción.

Los tres bloques principales son el contador de vueltas, el decodificador de binario a bcd y el multiplexor, el funcionamiento en detalle de cada uno de los mismos serán explicados a continuación. Cada uno de estos bloques esta gobernado por una señal de reset (provenientes de los bloques de reset que se explicaran a lo largo del informe), que los hace trabajar en determinados tiempos, teniendo así un mejor manejo de flujo, en donde cada bloque de reset ajusta la señal de reloj fija, de 100Mhz para nuestro caso, a la señal deseada para cada bloque.

Al tacómetro posee dos entradas, una de ellas es una señal de reloj denominada Ck y hace referencia al clock de la FPGA. Esta señal de entrada al ingresar al tacómetro se desplaza hacia 3 componentes que son Reset cuenta vueltas, Activador decodificador, y Refresco display. La segunda entrada, llamada Sensor, se encuentra conectada al contador de vueltas, ya que es la señal que se va a leer de la rueda dentada, proporcionando 6 pulsos por vueltas.

Las salidas que presenta el tacómetro son dos al igual que las entradas, pero en este caso son de tipo vector. La primera de ellas denominada dígitos, es un vector de 7 bits que se encarga de prender cada uno de los 7 leds del display 7 segmentos de la placa de desarrollo Basys 3, siendo los mismos de ánodo común. La otra salida corresponde a los Anodos de los display, por lo tanto, las salidas Dígitos y Anodos deben estar sincronizadas para encender el número de manera correcta, función que se encarga de realizar el multiplexor.

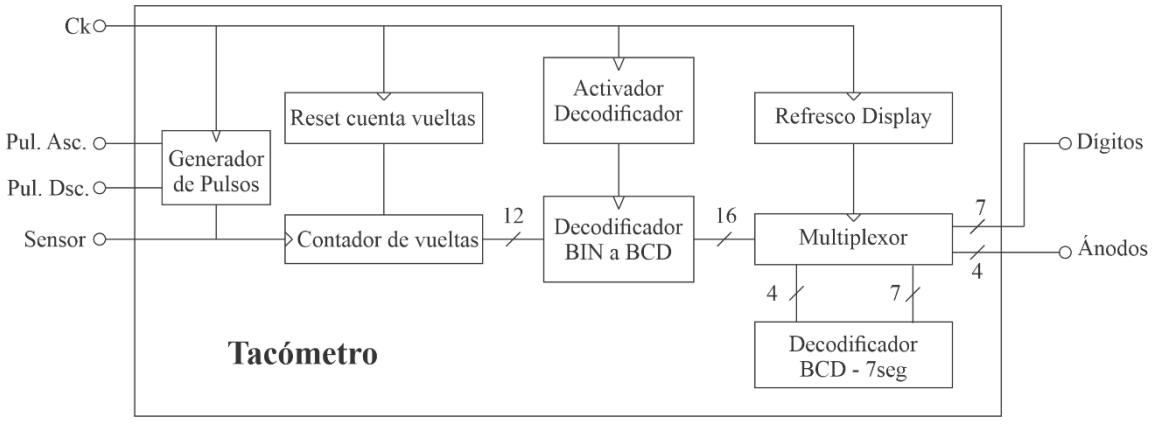

Fig. 1 *– Diagrama de bloques del tacómetro*

DESCRIPCIÓN DE BLOQUES

CONTADOR DE VUELTAS

El contador de vueltas es el circuito que se encarga de leer los pulsos que provienen del sensor, que sensa una rueda dentada de 6 dientes, y como salida entrega una señal de 12 bits que corresponden a las RPM, por lo tanto, puede ir desde 0 a 4095 RPM, por lo tanto, podemos decir que es el bloque más importante del circuito.

Funcionamiento

Como dijimos, el contador recibe las señales del sensor y las transforma en rpm, pero debemos reiniciar el contador cada un periodo de tiempo conocido, que para nuestro caso se opto por una frecuencia de 1Hz ya que no se necesita un refresco muy rápido, por lo tanto, es aceptable el segundo que debemos esperar para su estabilización.

Declaramos "signal cuentavueltas" en forma natural que va desde 0 a 4095, debido a que son los 4096 valores que tenemos con $2^{12} = 4096$, utilizamos 2^{12} debido a que nuestro problema nos especifica un rango de frecuencia de 400 a 2500, por lo que si usáramos $2^{11} = 2048$, por lo tanto no llegamos a obtener la frecuencia máxima necesaria.

La cuenta se lleva a cabo en un proceso, por lo que es un contador sincrónico con un reset asíncrono. En este cabe la observación de que, si Rst está activo, no se reciben las señales provenientes del sensor, esto es cierto, pero el circuito de que se encarga de otorgarle la señal de reinicio se mantiene activo durante solo un periodo, por lo que solo estará activo 10ns, que los podemos despreciar ya que es un tiempo relativamente pequeño para las frecuencias que estamos midiendo. Si "Rst" es igual a "1", lo que significa que nuestro reset del contador de vueltas se activó, lo que hacemos es poner en cero el cuenta vueltas (cuentavueltas = 0).

En caso de que se detecte una señal del sensor, la señal encargada de contar las vueltas se incrementa en 10. Sucede de esta manera ya que cada 1 vuelta de la rueda nuestro sensor arroja 6 pulsos, que son cada diente de la rueda dentada, si llamamos "f" a la frecuencia en que se da una vuelta completa y "p" a la cantidad de pulsos que arroja el sensor, podemos obtener a siguiente relación:

$$f * 6 = p$$

La frecuencia nos indica cuantas vueltas da en un segundo, pero a nosotros nos interesa saber rpm, por lo tanto, solo resta convertir la frecuencia a rpm:

$$RPM = f * \frac{60s}{1min}$$

Ahora si despejamos el valor de la frecuencia y obtenemos las rpm a través de los pulsos obtenidos del sensor tendremos:

$$RPM = \frac{p}{6} * \frac{60s}{1min} = p * 10$$

Siendo p cada pulso que se detecta del sensor, por lo tanto, podemos aplicar propiedad distributiva para obtener lo que utilizamos en la descripción:

$$RPM = p * 10 = (1 + 1 + 1 \ldots + 1) * 10 = 10 + 10 + 10 \ldots + 10$$

Al finalizar el proceso, solo resta convertir el valor natural de cuenta vueltas, en binario de 12 bits, mediante la función proporcionada por el lenguaje, y lo asignamos a RPM.

RESET – CUENTA VUELTAS

Como lo mencionamos anteriormente, este circuito se va a encargar de proporcionar un pulso cada 1 segundo para provocar el reinicio del bloque contador de vueltas, de esta manera poder medir las rpm.

Funcionamiento

Partimos de la utilización de un contador, y una señal de reloj la cual es proporcionada por la placa Basys 3 la cual tiene una frecuencia de 100MHz (100 millones de ciclos por segundo); por lo que mediante "process", cada vez que clock cambie de estado vamos a analizar si el flaco fue "1" o "0", en caso de que el flanco sea "0" no ocurre nada, si es el flanco es "1" analizamos el valor de la señal contador.

La cuenta se incrementará hasta un cierto punto, el cual sale de un calculo realizado por dos constantes que posee la descripción, que son: clockFPGA y clockPulso, ambas en Hz. El clock, tanto el de la FPGA como el del pulso, nos proporciona la cantidad de pulsos que va a tener la señal, por lo tanto, la cantidad de flancos ascendentes. Nos interesa conocer cuantos pulsos tienen que ocurrir de la señal de reloj de la FPGA para poder la frecuencia deseada en la salida, por lo tanto, dividimos ambas señales para vos cuantos pulsos necesitaríamos.

Para nuestro caso, si contador es igual a 100 millones (100MHz dividido 1), entonces volvemos a contador a "1"(cabe aclarar que no volvemos a 0 ya que estaríamos despreciando el pulso que hizo que lo hizo reiniciar) y activamos nuestro reset "Rst=1". En caso de que contador no sea igual a 100MHz simplemente sumamos 1 a contador y mantenemos nuestro reset "Rst=0".

Utilizamos "constant natural" para lograr un mejor entendimiento visual, ya que con números binarios puede resultar más complejo. Luego cada "constant natural" es transformado a binario por el compilador del entorno, para un correcto funcionamiento.

DECODIFICADOR

El decodificador que se implementó utiliza el algoritmo llamado doubble dabble, el cual es utilizado para convertir números binarios en notación decimal codificada en binario (BCD).

Funcionamiento

El algoritmo itera *n* veces, de acuerdo con la cantidad de bits que tenga en mensaje original. En cada iteración, cualquier dígito BCD que sea mayor que 4 se incrementa en 3; luego, todo el espacio temporal se desplaza un bit hacia la izquierda.

El decodificador es activado mediante el bloque activador display, el cual activa la conversión 1 periodo de reloj antes de que se reinicie el contador, para nuestro caso, como tenemos un reloj de 100MHz, la conversión se realiza 10 ns antes del reinicio, justamente para que el contador no se nos reinicie antes de decodificarlo y posteriormente, mostrarlo al usuario mediante un display de 7 segmentos (**Fig. 4**).

Es cierto que si entre el instante en que se realiza la conversión y el instante posterior, que es cuando se reinicia el contador, llega un pulso de la rueda dentada este no es tomado en cuenta en la salida ya que la señal se ha decodificado y en el instante posterior se

reinicia. Sin embargo, este comportamiento se puede despreciar ya que el instante de tiempo corresponde a un periodo de la señal de reloj, que como mencionamos anteriormente, es de 10ns, por lo que es un tiempo muy pequeño respecto a las frecuencias que deseamos medir por lo tanto se desprecia este efecto y se puede considerar que el reinicio del contador y la ejecución del decodificador se realizan al mismo tiempo, pero que se visualizan de manera correcta los rpm.

En nuestro caso los números en rojo indican los que superan al 4, por ende, se les suma 3. En la primera iteración se puede visualizar como el desplazamiento se lleva a cabo de a tres bits en lugar de a uno, esto se debe a que, si durante las primeras dos iteraciones desplazamos de a un bit, en ningún momento vamos a obtener un número que sea mayor a 4, por lo que lo hacemos de esta manera, para simplificar la expresión.

En nuestra descripción el mensaje se denomina entrada y consiste en un vector de 12 bits, el cual proviene del contador de vueltas, y la salida es un vector de 16 bits, en el que

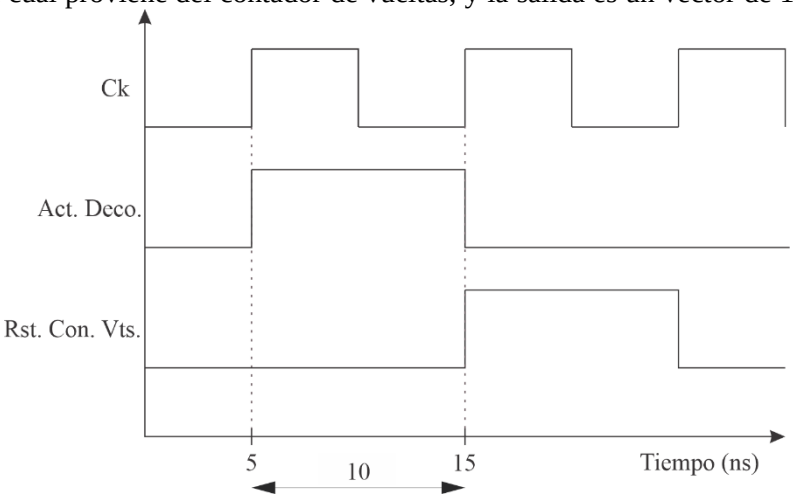

Fig. 4 *– Diagrama de tiempos de activación del decodificador y el reset del cuenta vueltas.*

se almacenan los 4 dígitos a mostrar en el display en BCD. En primer lugar, lo que se hace igualar la entrada a la variable mensaje completo, y luego se desplazan los 3 bits más significativos del mensaje completo a los tres bits menos significativos de la variable conversión parcial que esta igualada a salida, debido a que necesitamos por lo menos 3 dígitos binarios para que se supere el 4. Una vez hecho el desplazamiento se pregunta si en los primeros 4 bits de la conversión parcial el número obtenido es mayor a 4, lo mismo sucede con los segundos 4 bits, y con los terceros 4 bits. Cabe aclarar que, en los últimos 4 bits, es decir en los bits más significativos de la conversión parcial no se realiza la pregunta, ya que recién en la última iteración puede ser mayor a 4, por lo tanto, siguiendo con el algoritmo, no se reajusta. Una vez hecha la pregunta, y llevado a cabo la suma en caso de que sea mayor, se realiza el desplazamiento de a un bit. En este caso este proceso se va a realizar 9 veces.

A continuación, se adjunta una tabla realizando un ejemplo de cómo trabaja el algoritmo con una señal de 12 bits de entrada (**Fig. 5**):

Unidad de mil	Centena	Decena	Unidad	Mensaje
0000	0000	0000	0000	111111111111
0000	0000	0000	0111	111111111000
0000	0000	0000	1010	111111111000
0000	0000	0001	0101	111111110000
0000	0000	0001	1000	111111110000
0000	0000	0011	0001	111111100000

0000	0000	0110	0011	111111000000
0000	0000	1001	0011	111111000000
0000	0001	0010	0111	111110000000
0000	0001	0010	1010	111110000000
0000	0010	0101	0101	111100000000
0000	0010	1000	1000	111100000000
0000	0101	0001	0001	111000000000
0000	1000	0001	0001	111000000000
0001	0000	0010	0011	110000000000
0010	0000	0100	0111	100000000000
0010	0000	0100	1010	100000000000
0100	0000	1001	0101	000000000000
4	**0**	**9**	**5**	

Fig. 5 *– Tabla de conversión de una señal de 11 bits en binario a código BCD.*

ACTIVADOR DECODIFICADOR

El activador decodificador se encarga justamente de sincronizar el cuenta vueltas y el decodificador, para que la señal proveniente del cuenta vueltas se decodifique a BCD en el momento justo, de manera de tener en su salida un valor constante durante 1 segundo.

Funcionamiento

El funcionamiento es muy similar al Reset del cuenta vueltas, con la excepción de que, como podemos apreciar en la descripción, de que el contador va a contar hasta el pulso anterior que el contador de vueltas, es decir, el contador de vueltas se incrementa hasta 100.000.000, en cambio el activador del decodificador se incrementara hasta 99.999.999, con el fin de que reaccione un periodo antes de que se reinicie la cuenta, ya que Al ser el mismo reloj interno el de la FPGA no podemos sincronizar 2 reset en los mismos instantes, como se detalló anteriormente, y para no perder la fase, el contador vuelve a 0 y no a 1, como era en el caso anterior..

Para el caso en que “Ck sea 1” y no se llegó al final de la cuenta, simplemente lo que hacemos es incrementar contador en +1, y poner reset en 0.

MULTIPLEXOR

El multiplexor se encarga de ir prendiendo cada digito del display 7 segmentos, ya que internamente la conexión en la placa Basys 3 es la que se encuentra en la **Fig. 2**. Como entrada recibe del decodificador, los rpm a mostrar en display en código BCD, por lo tanto, este deberá convertir esa señal a una salida para un diplay de 7 segmentos e ir seleccionando los ánodos según corresponda.

Funcionamiento

Para sincronizar el display e ir prendiendo los dígitos regularmente, necesitamos de una señal de reloj proporcionada por un divisor de frecuencia, que se encarga de disminuir la frecuencia de reloj de la placa ya que la misma es demasiado alta. A medida que lleguen al circuito pulsos de reloj, este va a seleccionar el digito que desea prender.

En el vector de entrada tenemos el numero completo en BCD, pero debemos desglosarlo para prender de a 1 digito a la vez, por lo tanto, utilizaremos una señal denominada cont, la cual se encarga de guardar el valor de posición del vector donde se encuentra cada digito (**Fig. 3**). Comenzando cont 15, que es el valor más alto y por lo tanto donde se encuentra la unidad de mil, almacenamos el primer digito en una variable denominada digitoBCD que se encargara de almacenar el valor del digito a encender y se la pasa al decodificador de BCD a 7 segmentos, el cual escribe la salida del multiplexor. El with select se encarga de seleccionar el ánodo correcto dependiendo del valor de cont, por ende, del digito que estamos prendiendo.

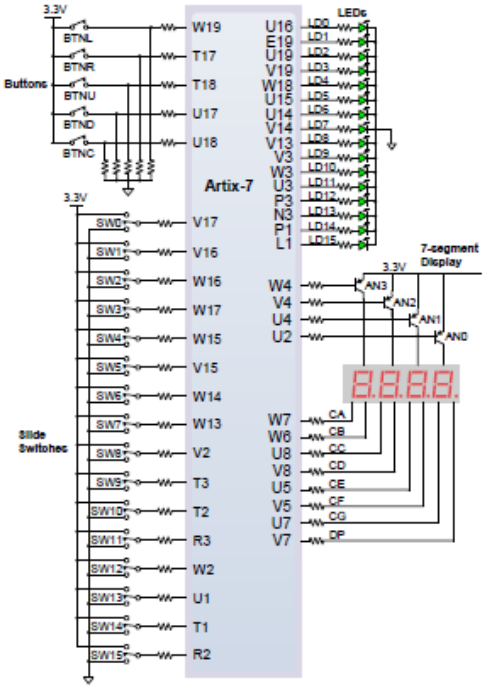

Fig. 2 *– Disposición de pines de display 7 segmentos en Basys 3*

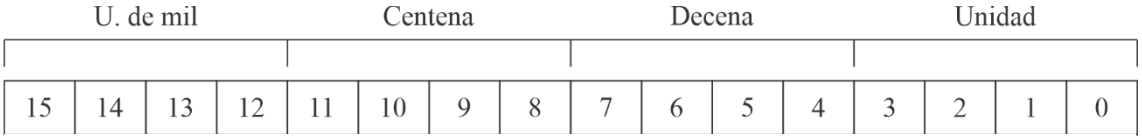

Fig. 3 *– Posicionamiento en vector Entrada*

Con la llegada de un pulso de reloj la señal cont disminuye en 4, posicionándose en el siguiente número, es decir la centena, repitiendo el mismo procedimiento detallado anteriormente, (pasarlo al decodificador bcd-7 segmentos, escribiendo la salida y seleccionado el catodo). Un caso limite es cuando la señal cont se encuentre en 3, por lo que en lugar de disminuir 4 con la llegada de un pulso de reloj nos volvemos a posicionar en la unidad de mil. Este ciclo se repite a una frecuencia lo suficientemente rápida para “engañar al ojo”, y notar que los dígitos permanezcan siempre encendidos, por lo tanto, se debe seleccionar una frecuencia que funcione de esta manera, en nuestro caso se optó por 2400Hz.

REFRESCO DISPLAY

El refresco de display se encarga de cambiar el digito que se enciende en un determinado tiempo, haciéndolo lo suficientemente rápido como para que el ojo lo perciba siempre encendido al display en su totalidad, y dejarlo el suficiente tiempo para poder visualizar el digito.

Funcionamiento

Debemos implementar un divisor de frecuencia para poder apreciar los valor obtenidos en nuestro display, ya que 100MHz del reloj de la FPGA es un valor muy grande y no lograríamos visualizar nada. En forma práctica variando la frecuencia de "clockSaliente" obtuvimos que 2400 Hz es un valor en el cual el funcionamiento es óptimo ya que no se nota el parpadeo y se tiene una sensación de que el display siempre permanece encendido.

Para ello simplemente implementamos un contador de flancos ascendentes, el cual esta gobernado por el clock de la placa. Cuando el contador llega al limite de ciclos deseados, simplemente invierte su estado, por lo que se declara como inout, es decir, "Ck_out = 0", al negarlo "Ck_out = 1", y viceversa. Con esto logramos generar un tren de pulsos con la frecuencia desea.

Como en este caso, deseamos obtener una onda completa como salida y no solo un pulso temporáneo, debemos hacer un cambio en el limite del contador, ya que si lo dejáramos como lo hicimos anteriormente en los demás, cada vez que se llogre la cantidad de ciclos deseada, en la salida veríamos mitad semi ciclo solamente, por lo tanto dividimos el conjunto en dos, o multiplicamos el divisor por dos que es análogo, de esta manera obtenemos la cantidad de pulsos que se necesita para realizar media onda, por lo tanto en la salida veremos una onda completa.

GENERADOR DE PULSOS

El generador de pulsos es un circuito de prueba, el cual se encarga de proporcionarle distintas frecuencias de prueba al tacómetro para poder verificar su funcionamiento. Cuenta de dos pulsadores, con los cuales es posible regular la frecuencia de salida, pudiendo seleccionar hasta 7 valores de frecuencias diferentes: 40Hz, 70Hz, 100Hz, 130Hz, 160Hz, 190Hz, 220Hz, 250Hz; dichos valores nos permiten tener un rango desde 400RPM hasta 2500RPM, ya que la relación que hay entre la frecuencia de los pulsos de entradas y los rpm de salida es un factor de 10.

Funcionamiento

La descripción en general esta conformada por dos procesos, más un monoestable antirebotes para cada pulsador.

El monoestable antirebotes es un circuito el cual se encarga de eliminar el ruido producido por un rebote de un pulsador mecánico, su funcionamiento básicamente se trata de esperar un determinador tiempo cuando se produce un cambio en la señal de entrada, para verificar de este modo su estabilidad. Eso se logra con un contador y un reloj, con una frecuencia fija. Cuando el circuito detecta un cambio entre el estado del pulsador que tenía almacenado en la señal estadoAnterior y el estado actual del pulsador (función realizada con una compuerta XOR), reinicia el contador en cero, almacena el nuevo valor de la entrada en la señal, y comienza a contar cuantos pulsos de reloj se mantiene estable, y como cada pulso de reloj tiene un periodo fijo, podemos determinar el tiempo que demora. Si se cumple la cuenta, significa que los valores se mantuvieron estables por lo cual asignamos a la salida el valor de la entrada, en caso contrario de que el pulsador cambie de estado antes de terminar la cuenta, esta se reinicia, comenzando el ciclo nuevamente.

La cantidad de pulsos que necesitamos para cumplir el tiempo del antirebotes lo podemos calcular con la siguiente formula:

$$Pulsos\ necesarios = tiempoAntirebotes * FrecuenciaReloj * 1000$$

En donde el tiempo de antirebotes está en milisegundos y la frecuencia de reloj en Mhz, los 1000 sirve para ajustar unidades ya que la frecuencia de reloj esta en ms, es decir $1x10^{-3}$s y la frecuencia de reloj en MHz, es decir $100x10^{6}$HZ. Si no pondríamos el factor la potencia tendría que estar expresada en KHz, ya que los exponentes se cancelan entre sí, pero como en la práctica es más fácil describir un reloj en términos de MHz debemos corregirlo ya que 1Mhz = 1000KHz.

El primer proceso, el cuál posee la etiqueta DDF, es un divisor de frecuencia variable, el cual transforma una señal proveniente de un circuito oscilador, en nuestro caso de 100Mhz, a una determinada frecuencia que puede ser modificada por el usuario. Funciona de manera similar a los demás divisores de frecuencia del resto del circuito, pero con la diferencia de que es variable, ya que la señal pulsos_clockSaliente puede variar, por lo tanto, cambia la cantidad de pulsos de reloj que se necesitan para lograr la frecuencia deseada, cambiando así la frecuencia de salida.

El segundo proceso, marcado con la etiqueta ajuste, sirve justamente para ajustar la frecuencia de salida, el mismo tiene en su lista de sensibilidad la señal detectaPulso, que surge de hacer la operación XOR entre la salida de los monoestables, debido a que el proceso se lleva a cabo cuando alguna de las dos señales tenga un valor diferente, protegiendo al proceso de la situación de pulsar ambos simultáneamente y además, un proceso no puede estar gobernado por dos señales de reloj, entendiéndose por señales de reloj a situaciones donde pregunta, mediante una sentencia if-then, por flancos ascendentes, y tampoco puede una señal estar gobernada por dos relojes, por lo tanto se optó por la opción que se detalló anteriormente y dentro de la sentencia if-then se encuentra un case, el cual se encarga de sumar, en caso de presionar el pulsador ascendente, y de restar en caso contrario, ambas operaciones realizadas sobre la señal selectorFrecuencia (cabe aclarar que ambas operaciones se pueden realizar si no se llega a sus límites) ya que las únicas posibles combinaciones que hace que la sentencia if sea verdadera son pAntiRebotes “10” o “01”. Finalmente contamos con un with-select-when, que se encarga de asignar a la señal pulsos_clockSaliente los pulsos necesarios para determinadas frecuencias, dependiendo del selectorFrecuencia.