



Universidad Politécnica de Madrid

# Grado en Ingeniería Electrónica y Automática

Asignatura

## **Sistemas Electrónicos Digitales**

Cronómetro – Cuenta Atrás





Universidad Politécnica de Madrid

# Grado en Ingeniería Electrónica Y Automática

## Datos del Grupo

*Eduardo Arcones del Álamo*  
(51085)

*Naipí de la Fuente de Pablo*  
(51160)

*Rodrigo Jerez Bedón*  
(51275)

# ÍNDICE

---

1	Objetivo del proyecto.....	6
2	Introducción .....	6
2.1	La tarjeta Spartan-3.....	6
2.2	Control del proceso de síntesis.....	7
2.2.1	Optimización: .....	7
2.2.2	Restricciones: .....	8
2.2.3	Problemas planteados y soluciones propuestas .....	10
3	Realización práctica .....	11
3.1	Pulsadores .....	11
3.2	Interruptores deslizantes .....	11
3.3	El display led de siete segmentos y cuatro dígitos.....	12
3.4	Entidades: .....	14
3.4.1	Global:.....	14
3.4.2	Cronómetro.....	15
3.4.3	Cuenta DHO .....	16
3.4.4	Cuenta BINARIO .....	17
3.4.5	Divisor de frecuencia .....	18
3.4.6	Visualizador.....	20
3.4.7	Sincronizador .....	21
3.4.8	Up_Down .....	21
3.4.9	Salida display.....	22
3.4.10	Reinicio .....	23
3.5	Diagrama de estados.....	24
3.5.1	Cronómetro.....	24
3.5.2	Contador binario (4 bits).....	26
3.5.3	Contador octal.....	26
3.5.4	Contador decimal .....	27
3.5.5	Contador hexadecimal.....	28
3.6	Diagramas de tiempos – Testbench.....	29
3.6.1	Cronómetro.....	29
3.6.2	Cuenta DHO .....	29

3.6.3	Cuenta binario.....	31
3.6.4	Divisor de frecuencia .....	31
3.6.5	Multiplexor.....	32
3.6.6	Decodificador de 7 segmentos.....	32
3.6.7	Up-Down .....	33
3.6.8	Sincronizador .....	33
3.6.9	Salida Display .....	34
3.6.10	Clock 200Hz.....	34

# 1 OBJETIVO DEL PROYECTO

---

El propósito de este trabajo es la implementación en la tarjeta de prácticas Spartan-3 de Xilinx un cronómetro capaz de contar hacia atrás. Para ello se diseñará un cronómetro que ocupe los 4 displays que dispone la tarjeta y que contará hacia adelante y hacia atrás.

A este cronómetro se le añadirán un botón de inicio de cuenta para comenzar el proceso, uno de paro momentáneo que pause la cuenta y un reset maestro que ponga el contador a cero y finalice el proceso.

## 2 INTRODUCCIÓN

---

Con el fin de mejorar el proyecto y aportar ideas originales a la funcionalidad del cronómetro, se añadirá un contador que contará en diferentes sistemas de numeración y nuevas tareas a los botones existentes.

Así, el dispositivo será capaz de contar hacia adelante y hacia atrás en sistema de numeración binario (dígitos 0-1), en octal (dígitos 0-7), en decimal (dígitos 0-9) y en hexadecimal (dígitos 0-F). Puesto que son diferentes sistemas de numeración, el máximo que se alcanzará dependerá del sistema en que se encuentre, siendo 1111 para binario, 7777 para octal, 9999 para decimal y FFFF para hexadecimal, lo que convertido a decimal es 15 para binario, 4095 para octal y 65535 para hexadecimal. Si está en modo cronómetro, como máximo podrá alcanzarse 59:59 en modo decimal ya que serían los minutos y segundos posibles con 4 displays.

Teniendo en cuenta la capacidad del dispositivo de cambiar de un sistema de numeración a otro, se han añadido funcionalidades a los interruptores deslizantes para poder realizar el cambio. Por lo que dependiendo de como estos estén activados o desactivados, se activará la función contador o la función cronómetro. Pero cuando se cambie de un sistema a otro, el contador se reseteará para evitar que se sobrepase el número máximo de un sistema inferior cuando se venga de un sistema superior.

Además se ha añadido un interruptor de vuelta (LAP) que al activarse junto con el interruptor *CE\_N* se pausen los 4 displays, pero no la cuenta interna del contador.

Por último, se ha añadido la posibilidad de que el usuario cargue su propio número para que el contador o el cronómetro empiecen su cuenta a partir de ese número. La introducción del número por el usuario se llevará a cabo a través de los pulsadores de la FPGA y se podrán cambiar cada uno de los cuatro displays por separado.

### 2.1 LA TARJETA SPARTAN-3

Todo esto se implementará y realizará en la tarjeta Spartan-3 de Xilinx, que es la ideal para iniciarse en el uso de las FPGA y tiene un bajo coste para la evaluación del diseño.

Las características más importantes de esta tarjeta son:

- Una FPGA XC3S200FT256 de 200.000 puertas con encapsulado BGA de 256 patillas:
  - Equivalente a 4.230 celdas lógicas.
  - Doce bloques de RAM de 18 kbit (216 kbit en total).
  - Doce multiplicadores hardware de 18 x 18 bits.
  - Cuatro gestores de reloj digitales (DCM).
  - Hasta 173 patillas de entrada salida definidas por el usuario.
- 2 Mbit de memoria flash XCF02S para configuración. Programable en el propio circuito a través del bus JTAG:
  - 1 Mbit está reservado para la configuración de la FPGA.
  - El resto está disponible para almacenar variables no volátiles del usuario.
- 1 Mbyte de memoria RAM estática asíncrona rápida:
  - Dos memorias ISSI IS61LV25616AL-10T seleccionables de forma individual con posibilidad de acceso a bytes individuales.
  - Posibilidad de organizarla en un bloque de 256 kword de 32 bits o dos bloques de 256 kword de 16 bits.
  - Tiempo de acceso de 10 ns.
- Visualizador con cuatro dígitos de siete segmentos.
- Ocho interruptores deslizantes.
- Ocho LED.
- Cuatro pulsadores.
- Un reloj de 50 MHz.
- Un puerto JTAG que nos permitirá conectar la FPGA al ordenador mediante un puerto USB.

## 2.2 CONTROL DEL PROCESO DE SÍNTESIS

### 2.2.1 Optimización:

Hemos seguido la metodología de diseño top-down que consiste en descomponer el problema en partes más pequeñas hasta que la implementación de esas partes sea trivial.

Por eso y con el propósito de sintetizar el código y facilitar su reutilización se ha dividido en módulos y se han empleado funciones. Esto también lleva consigo una mayor optimización del proceso y ayuda al programador a detectar posibles fallos a la hora de simularlo.

Dividir el código en módulos también facilita su simulación y la creación de *testbench* que sinteticen y depuren el código parte por parte, pero no por crear muchos módulos significa que el código esté más sintetizado.

En nuestro caso, teníamos módulos diferentes para cada sistema de numeración (octal, decimal y hexadecimal) que hacían prácticamente lo mismo, por lo que tenerlos por triplicado era inútil. Lo simplificamos todo juntando la programación de los tres sistemas de numeración en el mismo módulo y asignando a cada uno el máximo al que podía llegar, puesto que los dígitos que usan los de base inferior se incluyen dentro de los usados en el de base superior.

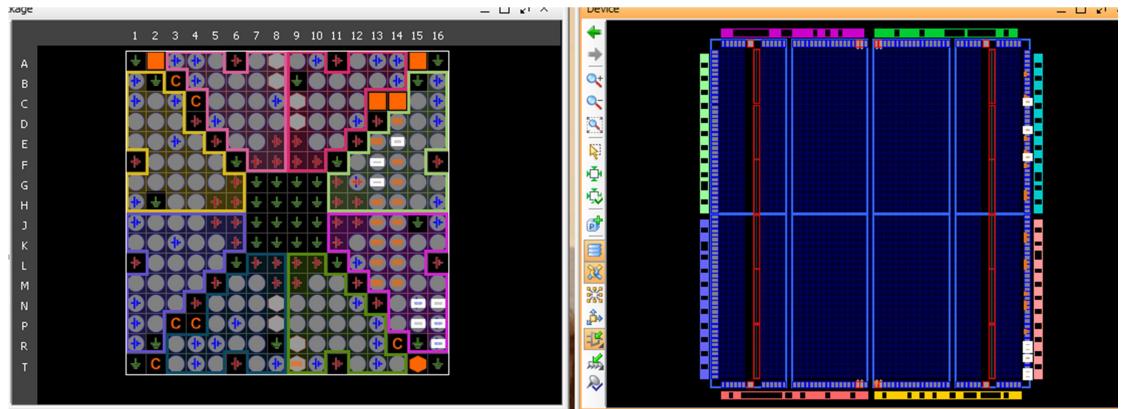
## 2.2.2 Restricciones:

Las restricciones impuestas por el diseñador para el cronómetro-cuenta atrás tienen que ser viables, puesto que si no se pudiesen cumplir la simulación interrumpiría la optimización e informaría del error. Por eso, en este diseño se han impuesto las siguientes restricciones y la siguiente asignación de recursos (*mapping*):

- Asignación de puertos a los multiplexores (uno por cada display):

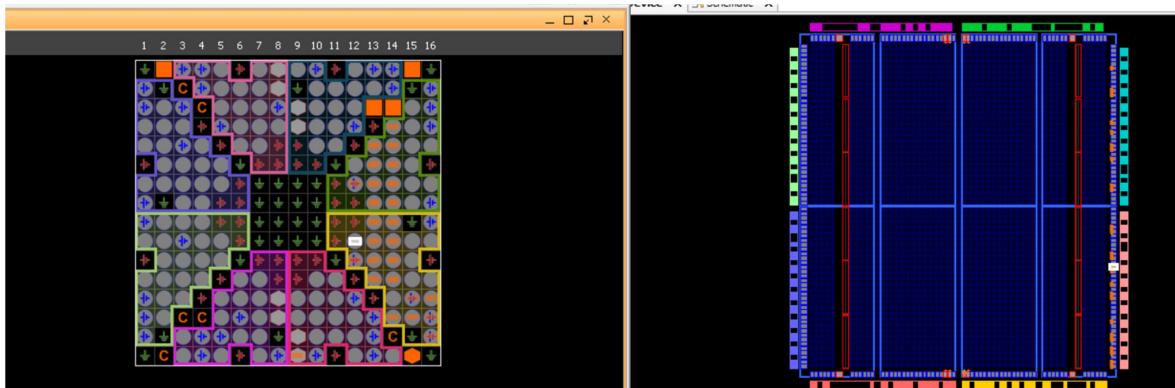


- Asignación de puertos a las salidas (uno por cada segmento del display):



- Asignación de puertos escalares:

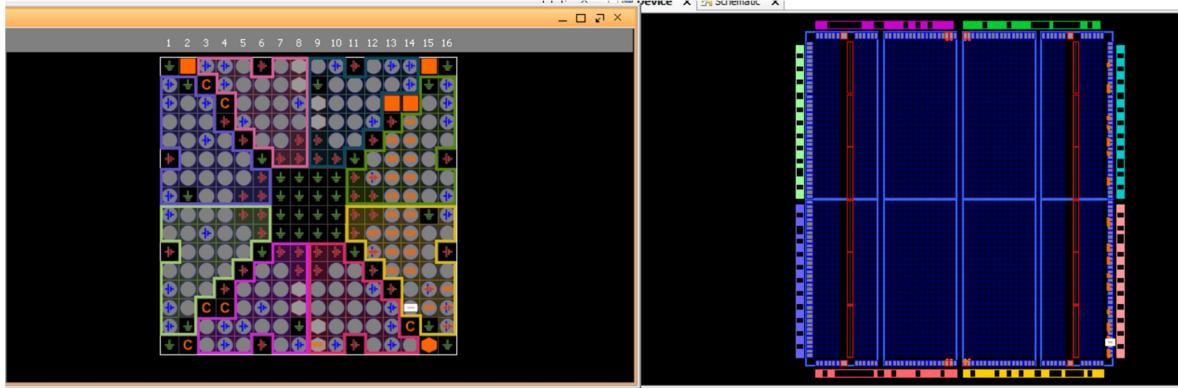
\* Led paro binario



\* Led paro cronómetro



\* Led paro DHO



### 2.2.3 Problemas planteados y soluciones propuestas

Un problema que nos surgió fue al cambiar de sistema de numeración, ya que el máximo de un sistema en base menor es inferior al número más alto que se puede alcanzar en una base mayor. Por tanto al cambiar de una base superior a una inferior el contador infringía los límites máximos en la base inferior dando error. Se solucionó añadiendo un nuevo módulo cuya función era evaluar la posición de cada interruptor de selección y cuando detectase un fallo reiniciar el contador para poder empezar en el nuevo sistema de numeración desde 0.

Otro problema que nos surgió fue con uno de los pulsadores de la FPGA, puesto que no realizaba la función que nosotros lo habíamos asignado. Antes de cambiar el código de programación, ya que eso suponía más trabajo y tiempo, quisimos comprobar que no fuese un fallo de la placa. Cambiamos la FPGA con la que estábamos trabajando y no tuvimos más ese problema, por lo que confirmamos que era un fallo de ese interruptor en concreto y no de nuestra programación.

Con los visualizadores se nos planteó la duda de que si incluyéndolos todos en el mismo módulo de discretización y enviándolos directamente las salidas de los módulos de cuentas no habría interferencias entre unos y otros. Se solucionó implementando un visualizador por módulo y juntando todas las salidas en un multiplexor para enviarlo al display de siete segmentos.

Al incluir el multiplexor anterior, se utilizó el divisor de frecuencia de 1Hz como al resto de módulos, pero al estar este a 1Hz en vez de a 200Hz como un multiplexor que habíamos utilizado anteriormente no se veían bien los números. Como la frecuencia que usaba era la que nosotros le introducíamos directamente, usamos la frecuencia del *clock* (50MHz).

### 3 REALIZACIÓN PRÁCTICA

---

El cronómetro con capacidad de cuenta atrás se mostrará a través de los cuatro displays que incluye la placa y podrá empezar una cuenta de 0 o desde un número introducido por el usuario a través de los pulsadores.

#### 3.1 PULSADORES

Solo funcionan cuando el interruptor *Load\_N* está apagado, ya que está negado y eso significa que cuando está negado está activado.

De derecha a izquierda, los pulsadores realizarán las siguientes funciones:

- Down: disminuye el número de un display de unidad en unidad.
- Up: aumenta el número de un display de unidad en unidad.
- Cambiar display: selecciona uno de los 4 displays en orden según se pulse.
- Reset: resetea el contador y lo pone a 0000 (esta función también estará disponible con el interruptor *Load\_N* activo).

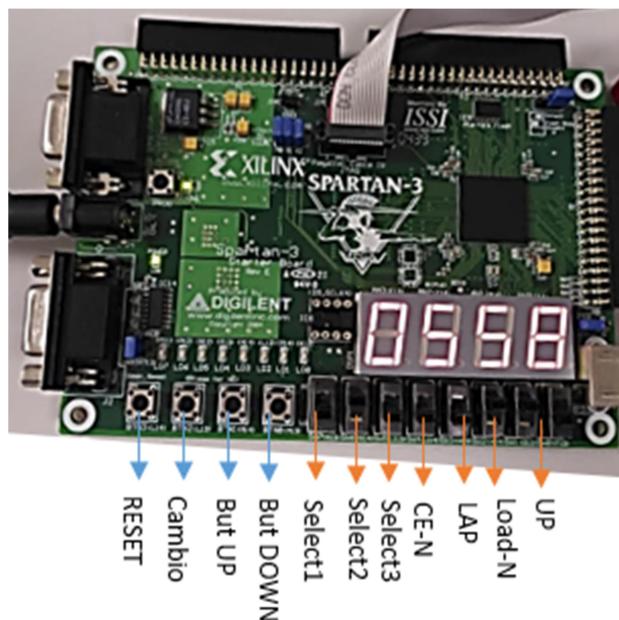
#### 3.2 INTERRUPTORES DESLIZANTES

Como ya se ha explicado anteriormente, a través de los interruptores deslizantes se podrán llevar a cabo diferentes funciones del cronómetro.

De derecha a izquierda de la FPGA, los interruptores deslizantes realizarán las siguientes funciones:

- Up/Down: contar hacia adelante cuando ON y hacia atrás cuando OFF
- Load\_N: carga el número seleccionado por el usuario cuando OFF
- Lap: cuando *CE\_N* se encuentra a 1 (cuenta deshabilitada), se pausa el display pero no la cuenta interna, por lo que cuando se desactive el display saltará al número que corresponda.
- CE\_N: habilita la cuenta cuando OFF y la pausa (pausando el display) cuando ON
- Select1, Select2 y Select3: dependiendo de sus valores seleccionan un modo de trabajo u otro.
  - Cronómetro-temporizador: 010-011
  - Contador binario: 000-001
  - Contador decimal: 100
  - Contador hexadecimal: 101-111
  - Contador octal: 110

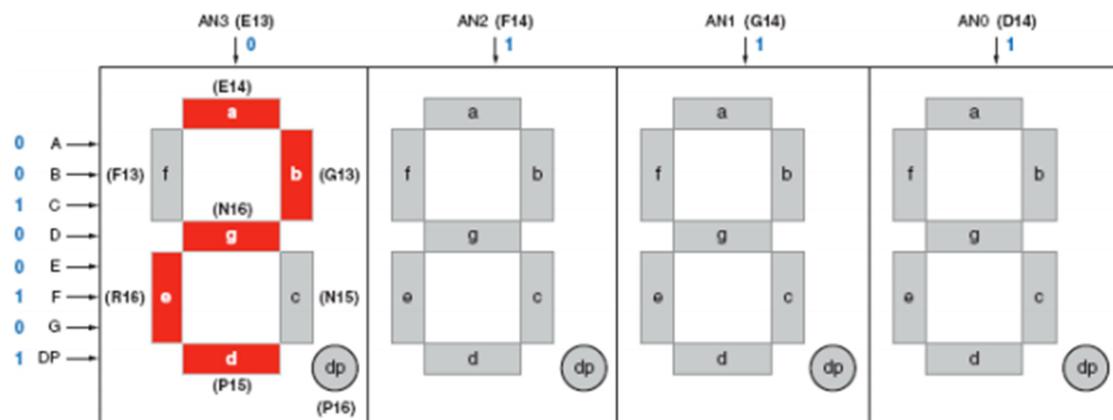
El usuario podrá seleccionar el número que desee modificando los cuatro displays por separado y cargarlo en el programa mediante el interruptor deslizante *Load\_N*. Luego podrá seleccionar si quiere que este número aumente (interruptor off) o disminuya (interruptor on) a través del interruptor *Up/Down*. Si quiere pausar los displays (mientras el tiempo sigue pasando) activará el interruptor *Lap* y el interruptor *CE\_N* a nivel alto, y si quiere resetear la cuenta y ponerla a 0 pulsará el botón *Reset*.



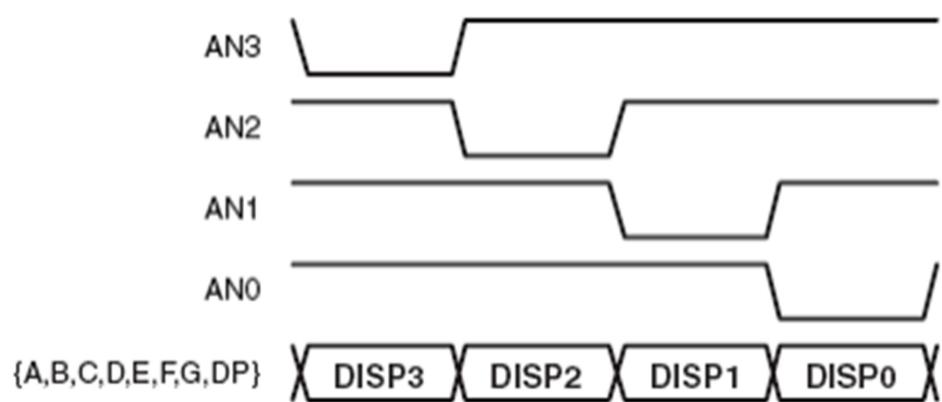
### 3.3 EL DISPLAY LED DE SIETE SEGMENTOS Y CUATRO DÍGITOS

Para poder visualizar los números en la placa se emplearán cuatro display led de siete segmentos, pudiéndose así obtener números de cuatro cifras .

Los dígitos de la tarjeta son de tipo LED, cada segmento es un LED y todos tienen un terminal que es común. En nuestro caso se trata de un ánodo común, por lo que para que luzca un segmento debemos aplicar al terminal no común un '0'. Los cuatro dígitos comparten las mismas señales de control de los segmentos y la forma de mostrar números distintos en cada dígito consiste en irlos encendiendo en secuencia haciendo que las líneas de control de los segmentos reflejen el número correspondiente al dígito activo en ese momento. Para iluminar un dígito concreto en esta tarjeta es necesario aplicar un '0' a la linea de control pertinente.



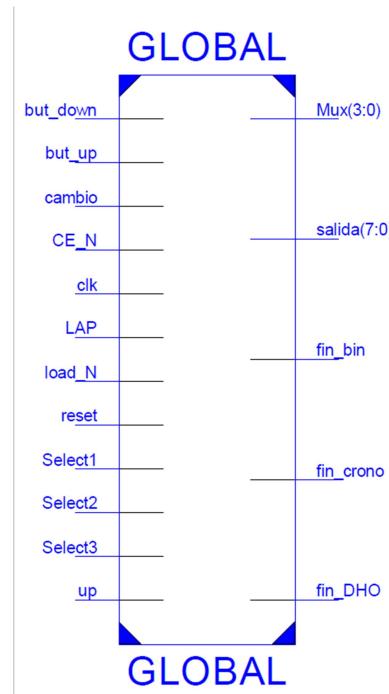
Carácter	a	b	c	d	e	f	G
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0



### 3.4 ENTIDADES:

#### 3.4.1 Global:

El esquema simplificado de la estructura TOP, a la que nosotros hemos llamado GLOBAL, se muestra simplificado en la siguiente figura. El esquema más detallado se encuentra al final del proyecto, en el anexo 1.



La entidad global tendrá los siguientes puertos:

```
24
25 entity GLOBAL is
26   Port ( clk : in STD_LOGIC; --Reloj
27           reset : in STD_LOGIC; --Reset, puesta a 0
28           up : in STD_LOGIC; --Contar hacia adelante cuando ON, hacia atras cuando OFF
29           load_N : in STD_LOGIC; --Cargar el numero seleccionado por el usuario cuando OFF
30           CE_N: in std_logic; --Habilita la cuenta cuando OFF, nivel ON para la cuenta
31           LAP: in std_logic; --Pausar el display pero no la cuenta interna
32           Select1: in std_logic; --Selecciona un sistema de numeracion
33           Select2: in std_logic; --Selecciona un sistema de numeracion
34           Select3: in std_logic; --Selecciona un sistema de numeracion
35           cambio: in std_logic; --Botón que selecciona un display u otro
36           but_up: in std_logic; --Botón para aumentar de uno en uno el número del display
37           but_down: in std_logic; --Botón para disminuir de uno en uno el número del display
38           salida: out std_logic_vector(7 downto 0); --Segmentos del display
39           Mux: out std_logic_vector(3 downto 0); --Displays de la FPGA
40           fin_bin: out std_logic;
41           fin_crono: out std_logic;
42           fin_DHO: out std_logic
43           -- selec_Display: out STD_LOGIC_VECTOR (3 downto 0);
44           -- simbolos_Display: out STD_LOGIC_VECTOR (7 downto 0)
45           );
46
47 end GLOBAL;
```

Y la arquitectura estará formada por 9 componentes (cronometro, cuenta\_DHO, cuenta\_bin, divisor\_frec, visualizador, sincronizador, up\_down, salida\_display y reinicio) y las señales necesarias para realizar las conexiones apropiadas entre los componentes de la arquitectura, los interruptores y botones de la FPGA, los segmentos y los ánodos de los dígitos. También asignará a cada entrada su lugar en la FPGA mediante el mapeo.

### 3.4.2 Cronómetro

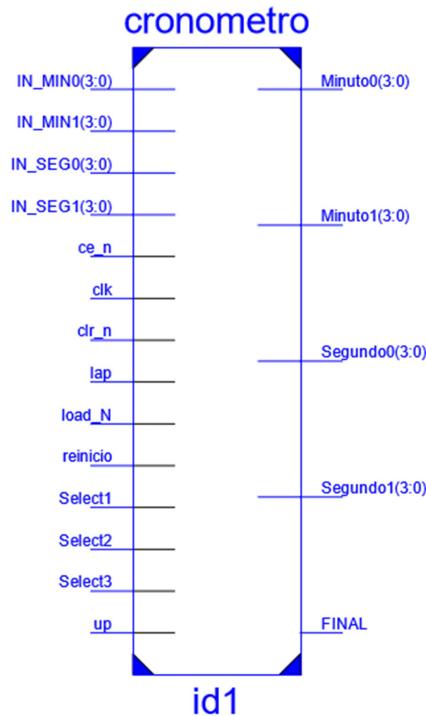
El componente *cronometro* contará en sistema sexagesimal los segundos y minutos hasta llegar a 59:59. Sus entradas y salidas se muestran en el diagrama de bloques a continuación.

Se añade la función de vuelta, que parará el display en un tiempo concreto pero no parará el reloj, por lo que aunque no se muestre este seguirá funcionando. Cuando se desactive el modo *vuelta*, el display mostrará el tiempo que el reloj le comunique, viéndose que este ha avanzado o retrocedido, según esté en modo cronómetro o temporizador respectivamente.

Si se desactiva la cuenta ( $CE\_N = 1$ ) el reloj se parará quedándose fijo el número del display hasta que se vuelva a activar la cuenta, que empezará desde el minuto y segundo en el que estaba.

Para cargar un tiempo determinado deberá estar desactivado el interruptor *Load\_N* y se configurará el tiempo deseado con los pulsadores de la FPGA, pudiéndose elegir los minutos y segundos que el usuario desee. Una vez cargado el nuevo tiempo, se podrá ir hacia adelante (función cronómetro) o hacia atrás (función temporizador).

Cuando el cronómetro llegue a 59:59 o acabe en 00:00 se encenderá un led de aviso de final de cuenta.



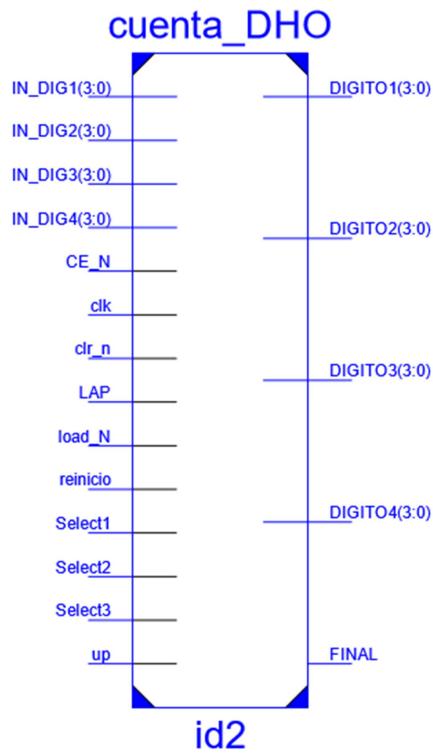
### 3.4.3 Cuenta DHO

El componente *cuenta\_DHO* se encargará de ir aumentando o disminuyendo unidades en los diferentes sistemas de numeración (Digital, Hexadecimal u Octal). Al igual que el cronómetro, se podrá elegir la cifra desde la que se empieza a contar (UP) o a disminuir (DOWN) con los pulsadores de la FPGA.

Adicionalmente, se podrá elegir el sistema en el que se desea realizar la cuenta mediante los interruptores de selección *Select1*, *Select2* y *Select3*. Si el contador está en un sistema y se desea cambiar a otro antes de que este haya finalizado la cuenta (si es hacia atrás) o en cualquier número de la cuenta hacia adelante, el contador se reseteará y empezará la cuenta de nuevo en el sistema elegido.

Como puede observarse en su diagrama de bloques, el módulo de *cuenta\_DHO* tiene las mismas entradas y salidas que el módulo *cronometro*.

Cuando llegue al número máximo de cada sistema o al final de la cuenta (0) se encenderá un led de aviso, que notificará al usuario que la cuenta ha acabado.

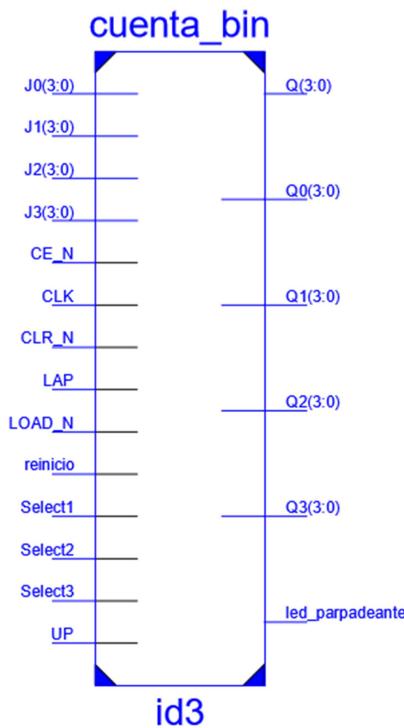


### 3.4.4 Cuenta BINARIO

El componente *cuenta\_bin* se encargará de ir aumentando o disminuyendo el contador en sistema binario, por tanto solo utilizará los dígitos 0 y 1 y como máximo llegará a 1111.

El resto de aspectos serán iguales a los del módulo *cuenta\_DHO*.

Cuando llegue al número máximo (1111) o al final de la cuenta (0) se encenderá un led de aviso, que notificará al usuario que la cuenta ha acabado.



### 3.4.5 Divisor de frecuencia

El componente *divisor\_frec* es un “prescaler” que genera frecuencias más bajas a partir de la señal de reloj de la placa, que es de 50MHz. En nuestro caso nos servirá con una frecuencia de 1Hz ya que corresponde a 1 segundo, que es la frecuencia necesaria para nuestro cronómetro. Los cálculos para dividir la frecuencia de entrada de 50MHz a 1Hz son los siguientes:

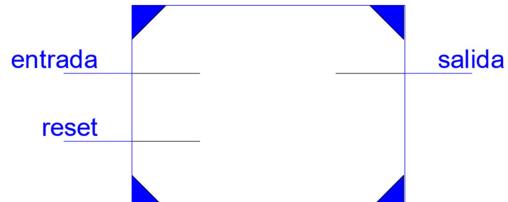
$$contador_{1Hz} = \frac{50MHz}{1Hz} = 50000000$$

El tiempo en alto y en bajo del reloj será el mismo:

$$t_{alto} = t_{bajo} = \frac{50000000}{2} = 25000000$$

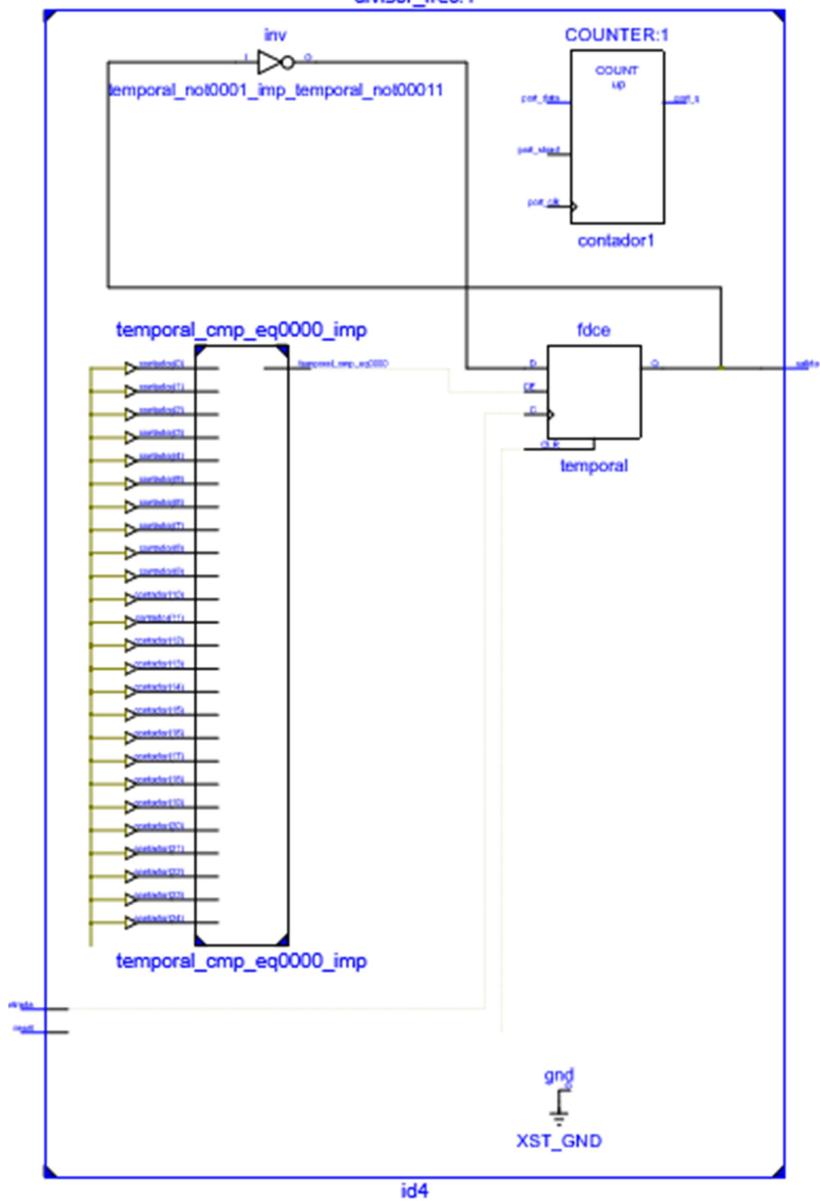
Y dado que empezamos la cuenta desde cero, la constante para el reloj a la frecuencia de 1 Hz será 24999999.

## divisor\_frec



**id4**

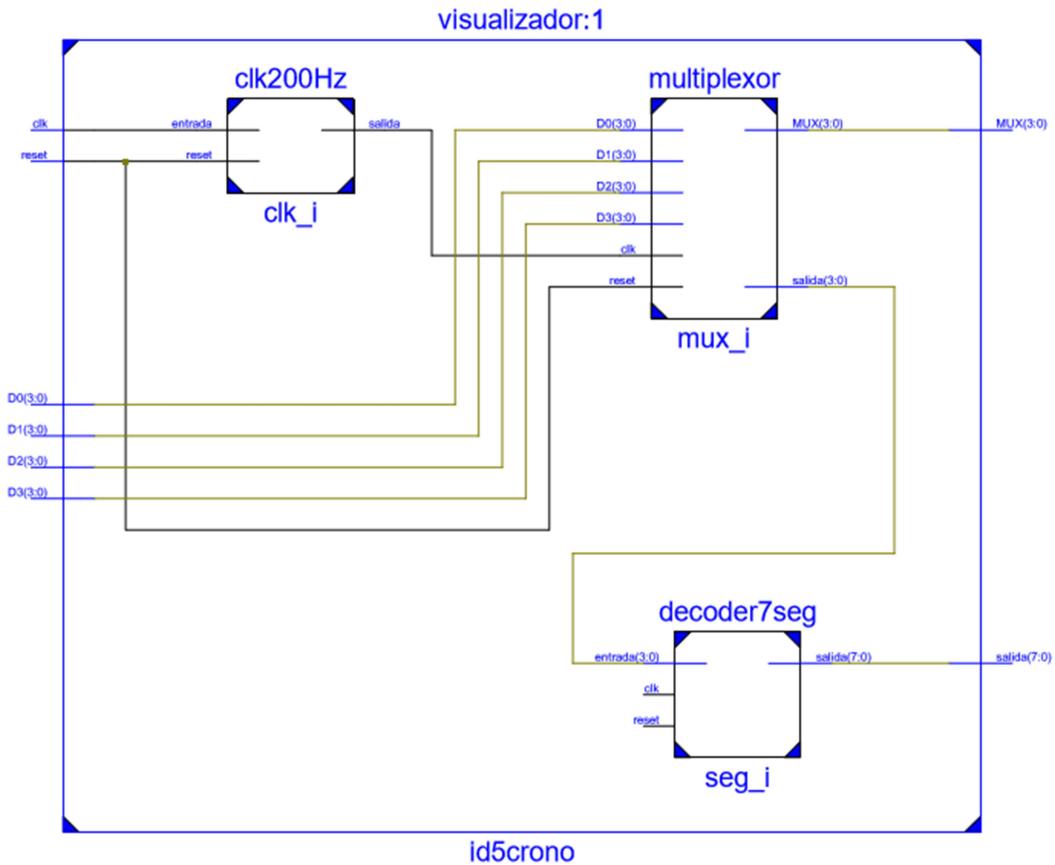
**divisor\_frec:1**

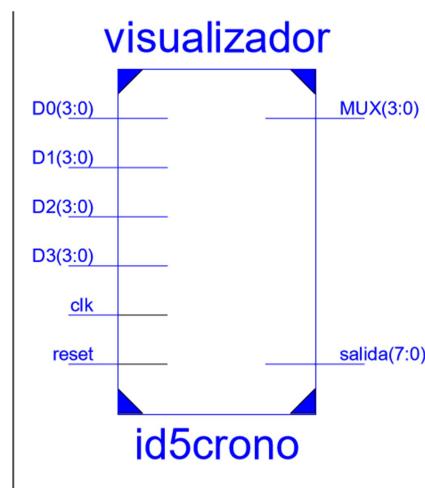


### 3.4.6 Visualizador

El componente *visualizador* coge los números que han salido de los sistemas de numeración y los incluye como entradas para mostrarlos por el display. Como disponemos de cuatro displays, el visualizador debe saber cuál es el número correcto en cada display por lo que añadimos un multiplexor.

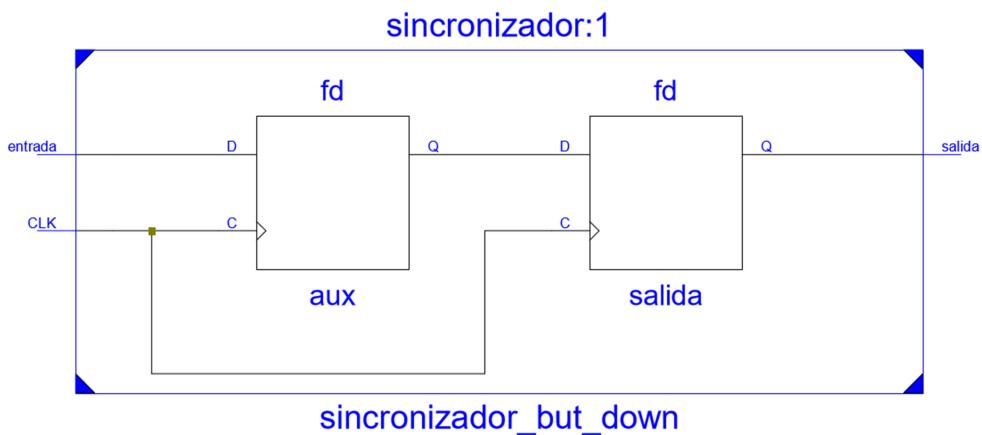
El multiplexor se encarga de mostrar un valor diferente en cada uno de los visualizadores de siete segmentos activando solamente un visualizador a la vez.





### 3.4.7 Sincronizador

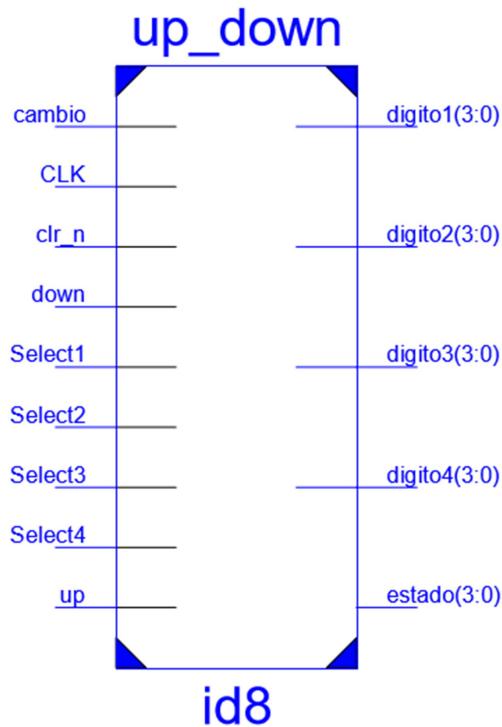
El componente *sincronizador* sirve para eliminar los problemas de rebotes. Estos problemas surgen cuando los botones o los interruptores deslizantes no cambian directamente de “0” a “1”, sino que entran en una situación de metaestabilidad en las que su valor no es fijo.



### 3.4.8 Up\_Down

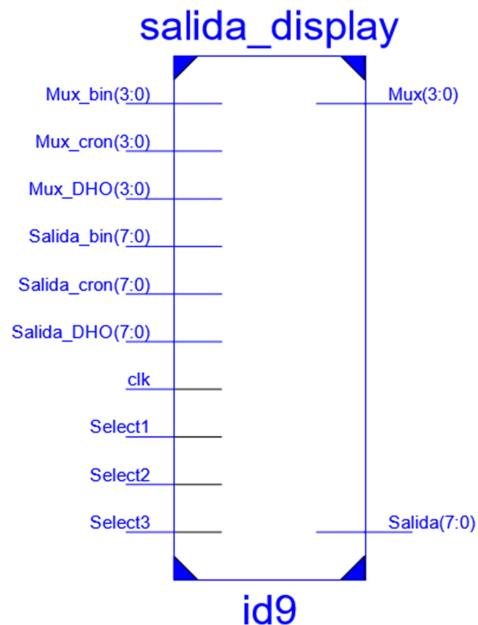
El programa accederá a este módulo cuando el interruptor *Load\_N* este desactivado y servirá para fijar el valor en el que se inicializará la carga. Evaluando las entradas de selección se establecerá el máximo al que se puede cargar, ya que cada base puede llegar a un número máximo diferente.

El pulsador *cambio* elegirá uno de los displays de 7 segmentos mientras que a través de *UP* y *DOWN* se establecerá el valor que desea introducir el usuario.



### 3.4.9 Salida display

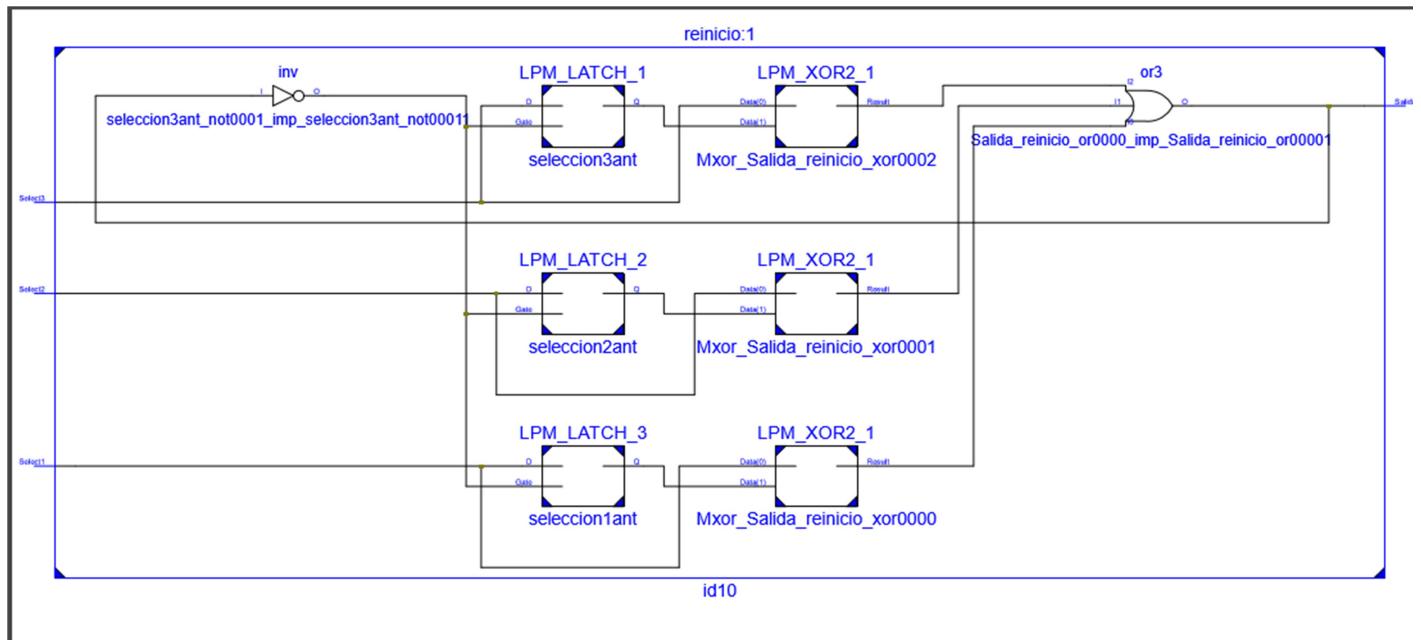
El componente *salida\_display* muestra en el display la salida seleccionada.



### 3.4.10 Reinicio

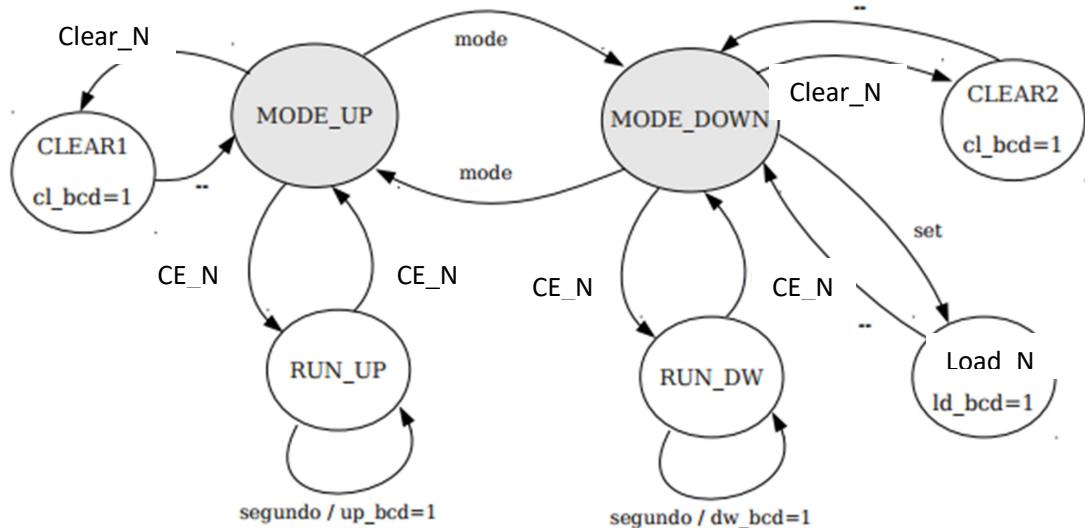
El módulo reinicio evalúa la posición de los interruptores deslizantes *Select1*, *Select2* y *Select3* y si cambia manda una orden a los módulos de cuenta para que se reinic peace.

Con este módulo se soluciona el problema de los cambios de base, ya que el número más alto que se puede llegar en una base menor es inferior al número más alto de una base mayor. Así, si estamos en base 16 y el contador muestra en pantalla ‘AAAA’ y cambiamos a base 8 evitamos el problema que surgiría, ya que en sistema octal equivaldría a ‘177777’ que como puede observarse ocuparía más de 4 displays.

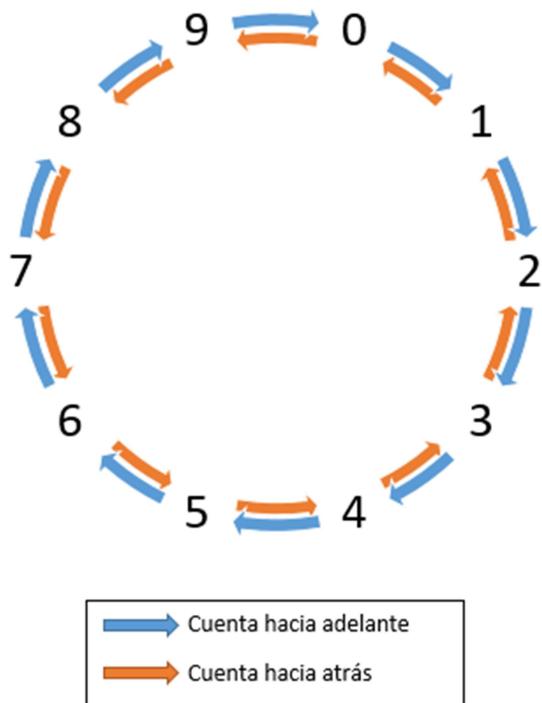


## 3.5 DIAGRAMA DE ESTADOS

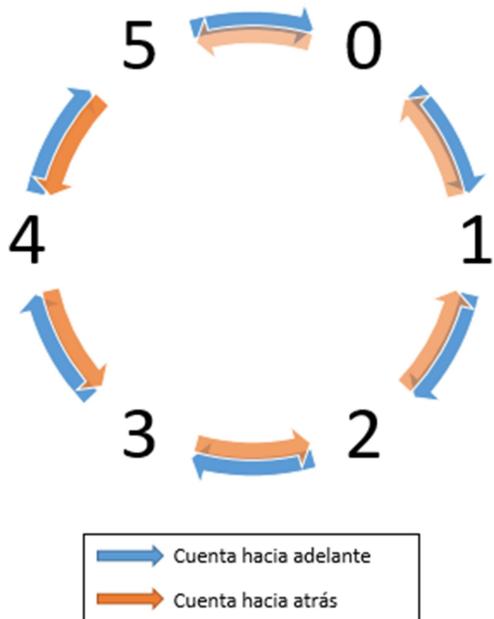
### 3.5.1 Cronómetro



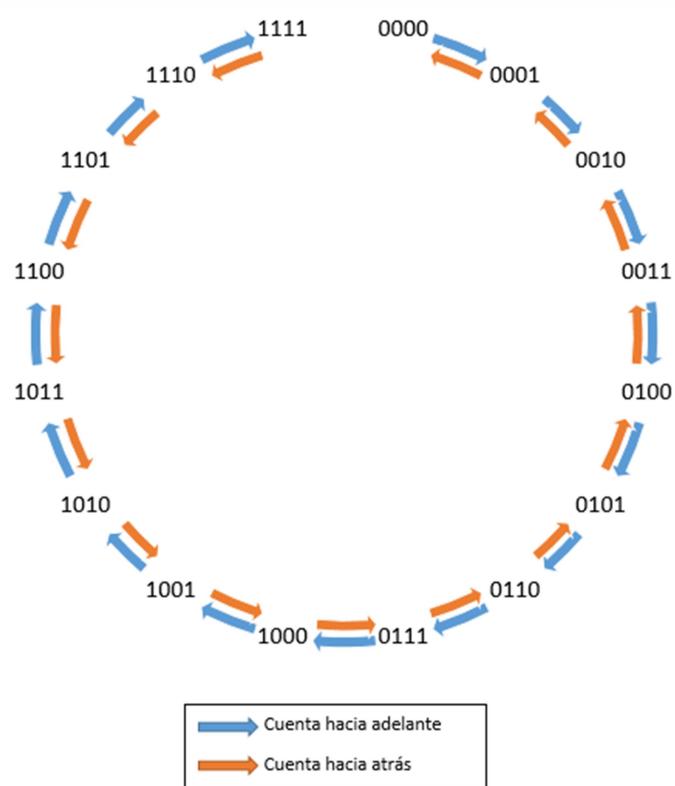
3.5.1.1 Displays 1 y 3 (unidad segundos, unidad minutos):



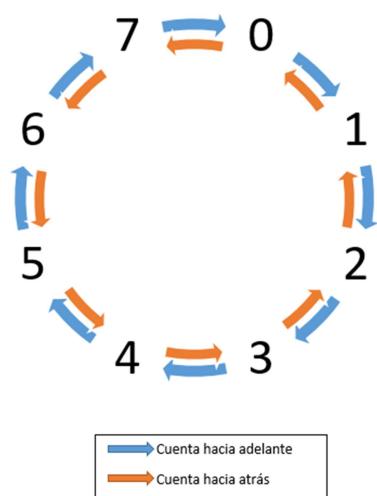
3.5.1.2 Displays 2 y 4 (decena segundos, decena minutos):



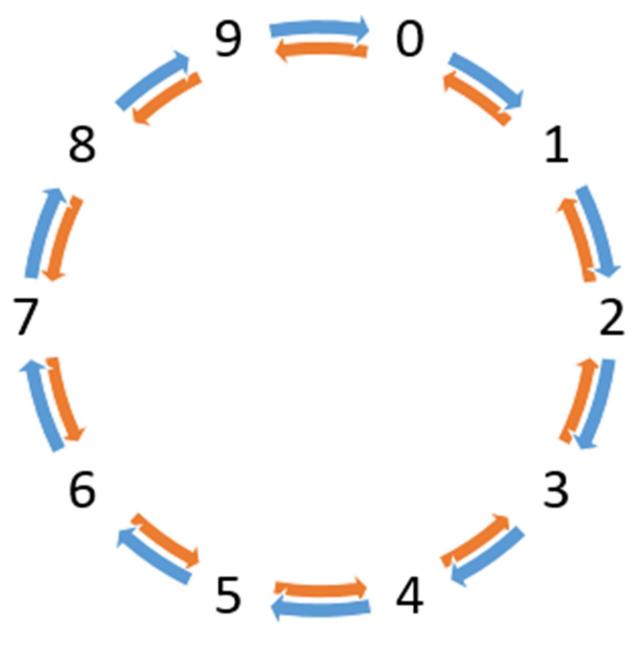
### 3.5.2 Contador binario (4 bits)



### 3.5.3 Contador octal

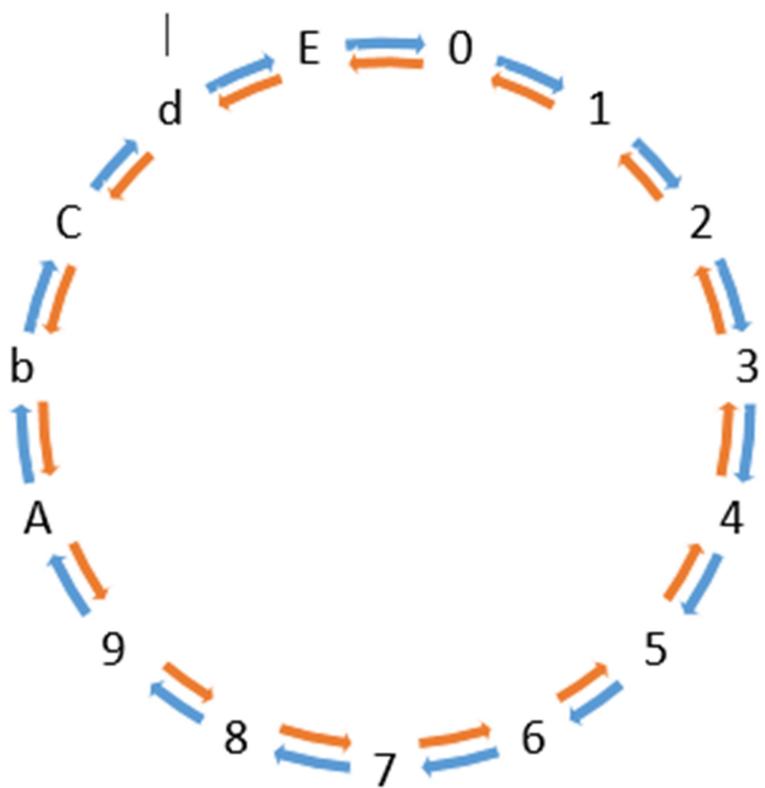


### 3.5.4 Contador decimal



Cuenta hacia adelante
Cuenta hacia atrás

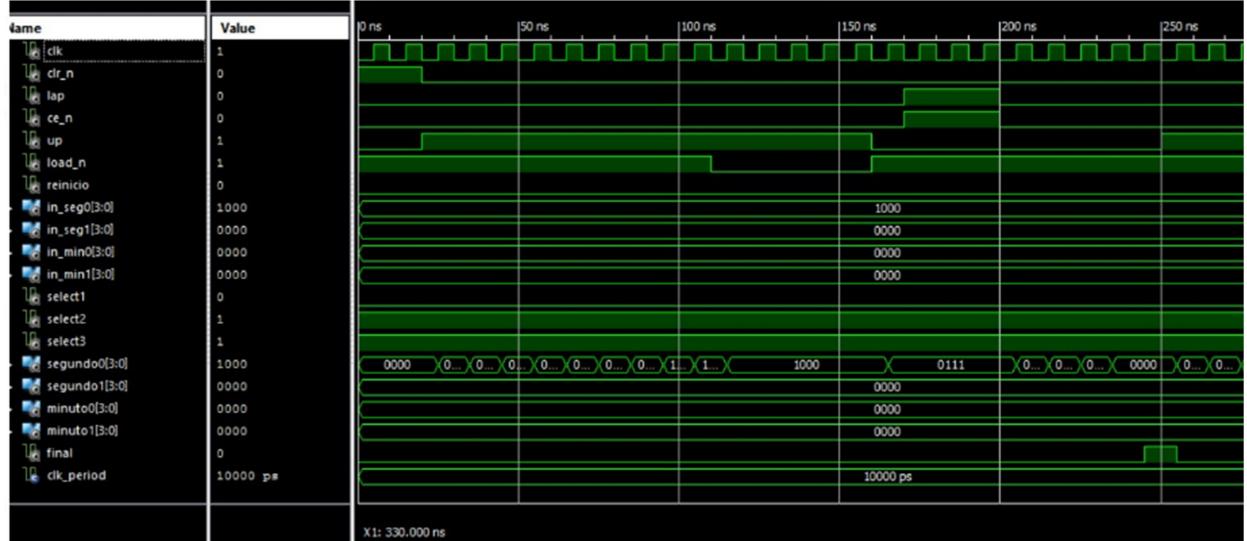
### 3.5.5 Contador hexadecimal



	Cuenta hacia adelante
	Cuenta hacia atrás

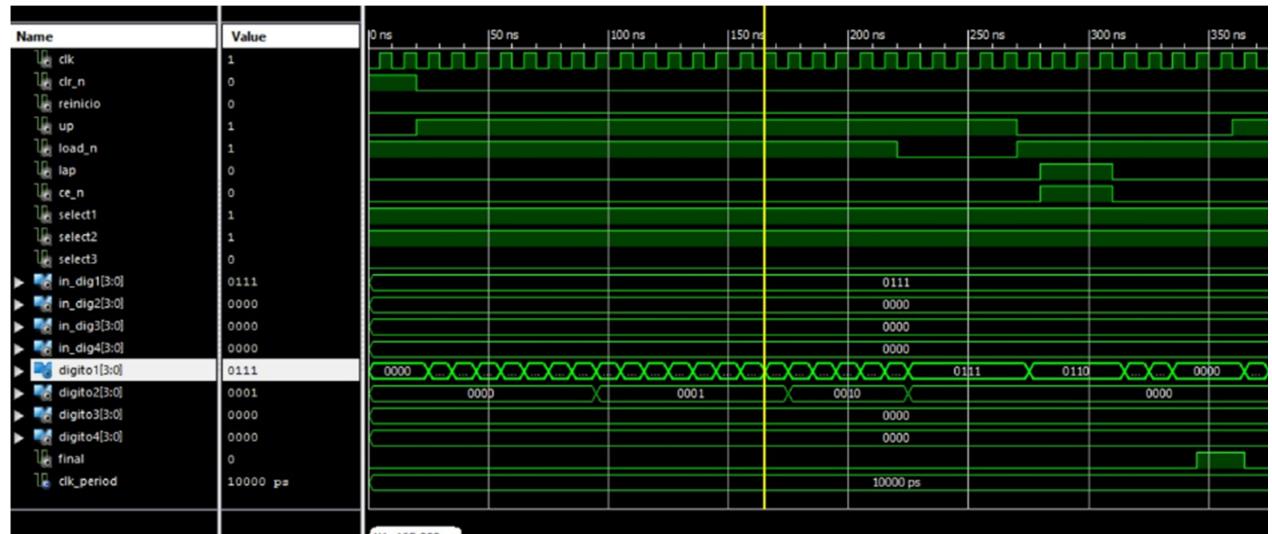
## 3.6 DIAGRAMAS DE TIEMPOS – TESTBENCH

### 3.6.1 Cronómetro

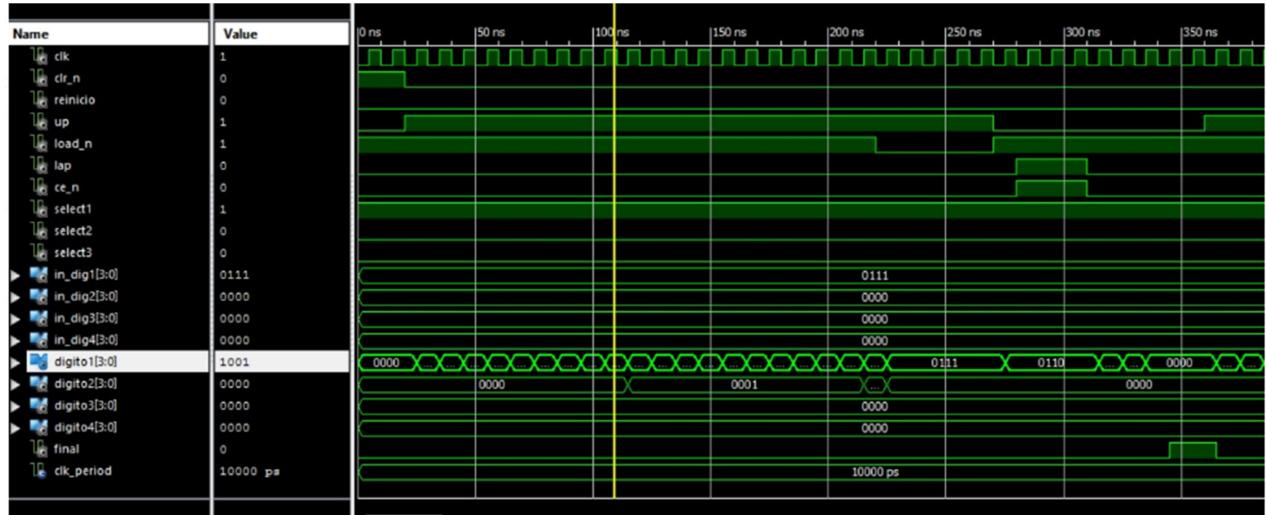


### 3.6.2 Cuenta DHO

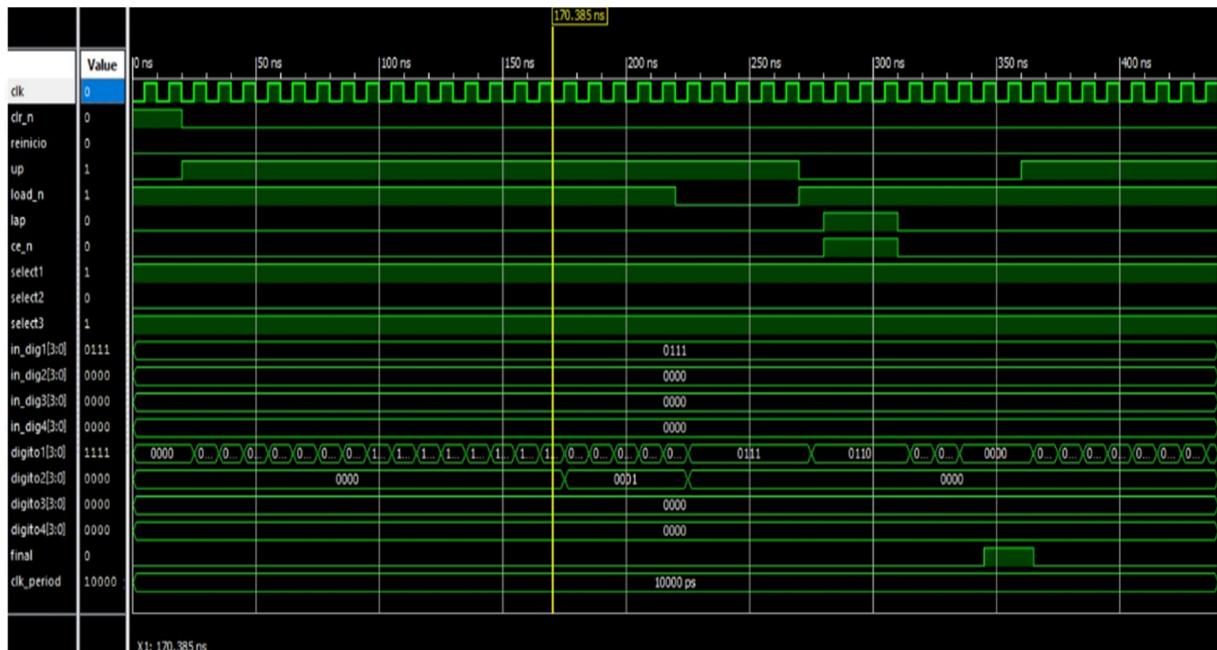
#### 3.6.2.1 Octal



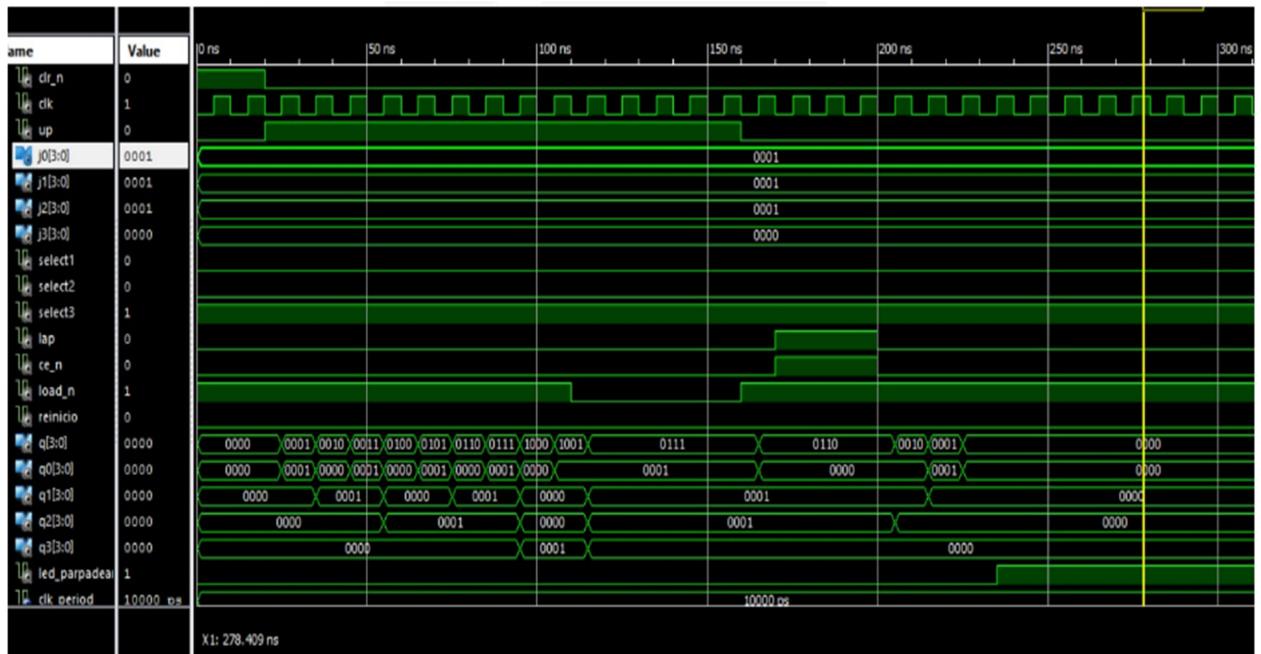
### 3.6.2.2 Decimal



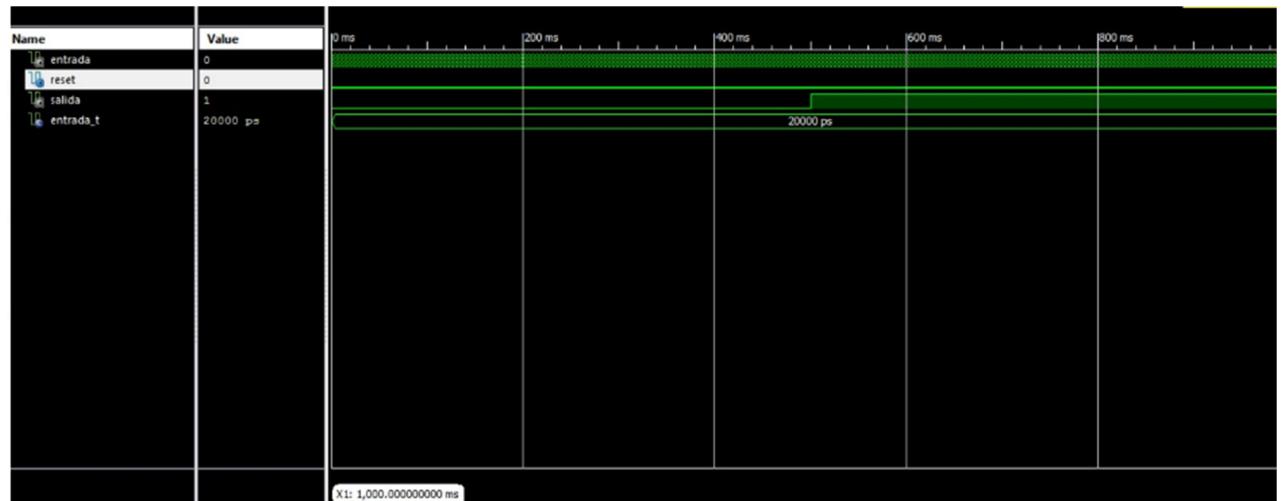
### 3.6.2.3 Hexadecimal



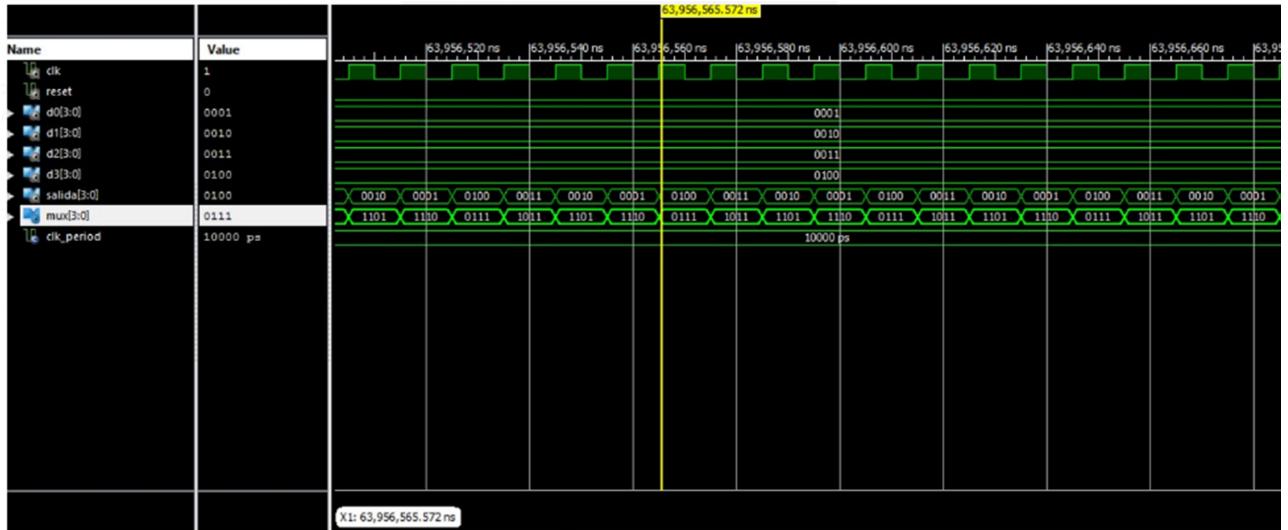
### 3.6.3 Cuenta binario



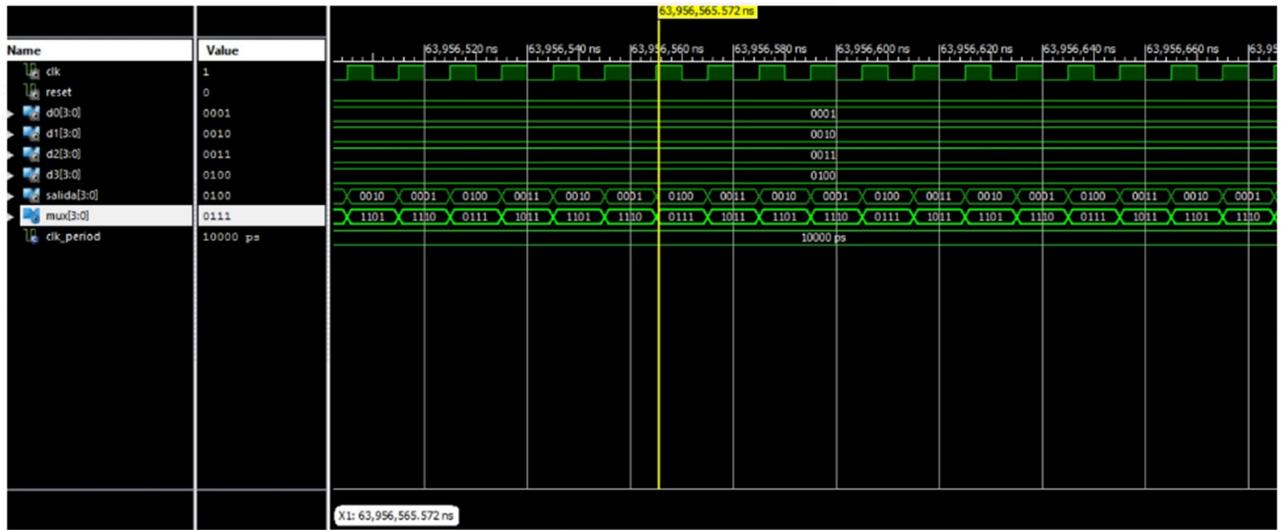
### 3.6.4 Divisor de frecuencia



### 3.6.5 Multiplexor



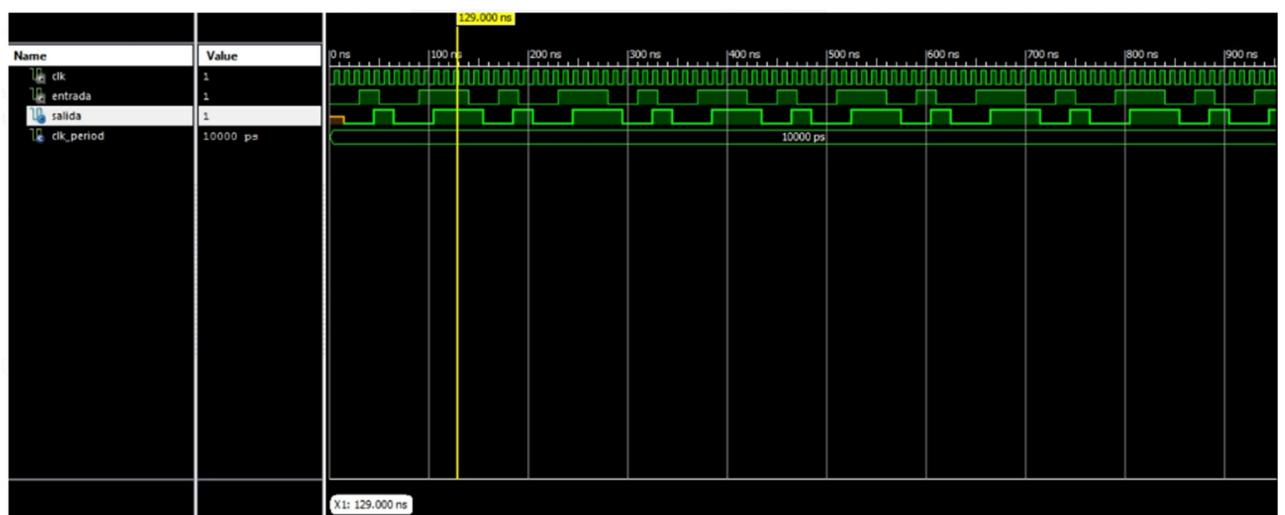
### 3.6.6 Decodificador de 7 segmentos



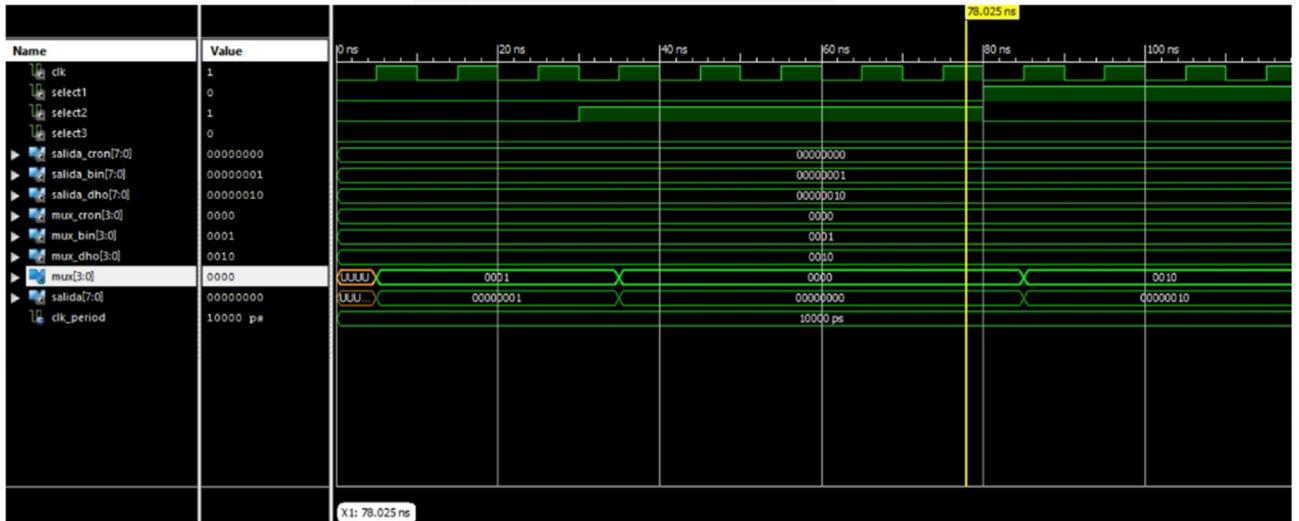
### 3.6.7 Up-Down



### 3.6.8 Sincronizador



### 3.6.9 Salida Display



### 3.6.10 Clock 200Hz

