2014/2015

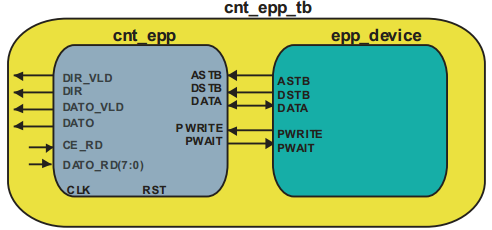
Modelado y síntesis de sistemas electrónicos digitales

PrÁctica Laboratorio

christopher HYDE PEINADO 09046047D

1. **CNT\_EPP**

**1.1.1.-** Código VHDL de la entidad ***cnt\_epp*** y del banco de pruebas utilizado en su simulación.  
También se debe proporcionar el código de la entidad *epp\_device*.



**A)** Código VHDL de la entidad ***cnt\_epp***:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity cnt\_epp is

port (

CLK : in std\_logic;

RST : in std\_logic;

ASTRB : in std\_logic;

DSTRB : in std\_logic;

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : in std\_logic;

PWAIT : out std\_logic;

DATO\_RD : in std\_logic\_vector(7 downto 0);

CE\_RD : out std\_logic;

DIR : out std\_logic\_vector (7 downto 0);

DIR\_VLD : out std\_logic;

DATO : out std\_logic\_vector (7 downto 0);

DATO\_VLD : out std\_logic);

end ;

architecture rtl of cnt\_epp is

signal S1:std\_logic;

signal S11:std\_logic;

signal Q:std\_logic;

signal S2:std\_logic;

signal S22:std\_logic;

signal Q2:std\_logic;

begin

------------------ADDRESS---------------------------------

AddrBiestableD1:process (CLK,RST)

begin

if (RST='1') then

Q <= '0';

elsif (CLK'event and CLK='1')then

Q <= ASTRB;

end if;

end process;

S1<= ASTRB and not Q;

S11<=S1 and not PWRITE;

AddrBiestableD2:process (CLK,RST)

begin

if (RST='1') then

DIR\_VLD <= '0';

elsif (CLK'event and CLK='1')then

DIR\_VLD <= S11;

end if;

end process;

AddrBiestableD3:process (CLK,RST,S11)

begin

if (RST='1') then

DIR <= (others=>'0');

elsif (CLK'event and CLK='1')then

if (S11='1') then

DIR <= DATA;

end if;

end if;

end process;

-----------------DATA------------------------------------

DataBiestableD1:process (CLK,RST)

begin

if (RST='1') then

Q2 <= '0';

elsif (CLK'event and CLK='1')then

Q2 <= DSTRB;

end if;

end process;

S2<= DSTRB and not Q2;

S22<=S2 and not PWRITE;

DataBiestableD2:process (CLK,RST)

begin

if (RST='1') then

DATO\_VLD <= '0';

elsif (CLK'event and CLK='1')then

DATO\_VLD <= S22;

end if;

end process;

DataBiestableD3:process (CLK,RST,S22)

begin

if (RST='1') then

DATO <= (others=>'0');

elsif (CLK'event and CLK='1')then

if (S22='1') then

DATO <= DATA;

end if;

end if;

end process;

--tristate buffer

DATA<= DATO\_RD when ((PWRITE= '1') and (DSTRB='1')) else (others => 'Z');

PWAITbiestableD:process (CLK,RST,ASTRB,DSTRB,PWRITE)

begin

if (RST='1') then

PWAIT <= '0';

elsif (CLK'event and CLK='1')then

if ((PWRITE='0') and (ASTRB='0')) or ((PWRITE='0') and (DSTRB='0')) or ((PWRITE='1') and (DSTRB='0')) then

PWAIT <= '1';

else

PWAIT <= '0';

end if;

end if;

end process;

CE\_RDbiestableD:process (CLK,RST,PWRITE)

begin

if (RST='1') then

CE\_RD <= '1';

elsif (CLK'event and CLK='1')then

if (PWRITE='1') then

CE\_RD <= DSTRB;

end if;

end if;

end process;

end rtl;

**B)** Código VHDL de la entidad ***epp\_device*** modificado para realizar 3 escrituras y una lectura:

Process

procedure epp\_cicle ( address : in std\_logic\_vector(7 downto 0);

data\_io : inout std\_logic\_vector(7 downto 0);

r\_w : in character) is

begin

wait until clk\_epp = '1';

PWRITE <= '0';

wait until clk\_epp = '1';

ASTRB <= '0';

data <= address;

wait for T\_clk\_epp\*EPP\_cicle\_length;

ASTRB <= '1';

wait until clk\_epp = '1';

data <= (others => 'Z');

PWRITE <= '1';

wait until clk\_epp = '1';

wait for T\_clk\_epp\*EPP\_cicle\_length;

--------------------------------------------------------------------------

if r\_w = 'w' then -- write cicle

PWRITE <= '0';

data <= data\_io;

end if;

--------------------------------------------------------------------------

wait until clk\_epp = '1';

DSTRB <= '0';

wait for T\_clk\_epp\*EPP\_cicle\_length;

--------------------------------------------------------------------------

if r\_w = 'r' then -- read cicle

data\_io:= data;

end if;

--------------------------------------------------------------------------

DSTRB <= '1';

wait until clk\_epp = '1';

data <= (others => 'Z');

PWRITE <= '1';

wait until clk\_epp = '1';

end procedure;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use std.textio.all;

use ieee.std\_logic\_textio.all;

entity epp\_device is

port (

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : out std\_logic;

DSTRB : out std\_logic;

ASTRB : out std\_logic;

PWAIT : in std\_logic);

end epp\_device;

architecture sim of epp\_device is

constant T\_clk\_epp : time := 100 ns; -- Internal clock period.

signal clk\_epp : std\_logic := '0'; -- Internal clock signal.

signal read\_value : std\_logic\_vector(7 downto 0) := (others => '0');

constant dir\_frec : std\_logic\_vector( 7 downto 0) := x"F0";

constant dir\_dpram1 : std\_logic\_vector( 7 downto 0) := x"A1";

constant dir\_dpram2 : std\_logic\_vector( 7 downto 0) := x"A2";

constant EPP\_cicle\_length: natural:= 10;

begin

-- internal clock signal generation.

clk\_epp <= not(clk\_epp) after T\_clk\_epp/2;

file arch\_in : text ;

variable bf : line;

variable dato : std\_logic\_vector(7 downto 0);

variable dir : std\_logic\_vector(7 downto 0);

begin

--inicialización

data <= (others => 'Z');

PWRITE <= '1';

DSTRB <= '1';

ASTRB <= '1';

dir := (others => '0');

-----------------first write values----------

wait for 130 ns;

DIR:=dir\_dpram1;

DATO:=X"34";

epp\_cicle ( address => dir,

data\_io => dato,

r\_w => 'w');

-----------------second write values----------

wait for 130 ns;

DIR:=dir\_dpram2;

DATO:=X"44";

epp\_cicle ( address => dir,

data\_io => dato,

r\_w => 'w');

-----------------third write values----------

wait for 130 ns;

DIR:=dir\_frec;

DATO:=X"54";

epp\_cicle ( address => dir,

data\_io => dato,

r\_w => 'w');

-----------------first read values----------

wait for 130 ns;

DIR:=X"FF";

epp\_cicle ( address => dir,

data\_io => dato,

r\_w => 'r');

read\_value<=dato;

wait for 1 us;

report "FIN CICLO R/W" severity failure;

end process;

end sim;

**C)** Código VHDL del **testbench** de simulación ***cnt\_epp\_tb:***

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: cnt\_epp PORT MAP (

CLK => CLK,

RST => RST,

ASTRB => ASTRB\_comm,

DSTRB => DSTRB\_comm,

DATA => DATA\_comm,

PWRITE => PWRITE\_comm,

PWAIT => PWAIT\_comm,

DATO\_RD => DATO\_RD\_comm,

CE\_RD => CE\_RD,

DIR => DIR,

DIR\_VLD => DIR\_VLD,

DATO => DATO,

DATO\_VLD => DATO\_VLD

);

eppDevice:epp\_device

port map(

DATA=>DATA\_comm,

PWRITE=>PWRITE\_comm,

DSTRB=>DSTRB\_comm,

ASTRB=>ASTRB\_comm,

PWAIT=>PWAIT\_comm

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

RST\_process :process

begin

wait for 5 ns;

RST <= '0';

end process;

END rtl;

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY cnt\_epp\_tb IS

END cnt\_epp\_tb;

ARCHITECTURE rtl OF cnt\_epp\_tb IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT cnt\_epp

PORT(

CLK : IN std\_logic;

RST : IN std\_logic;

ASTRB : IN std\_logic;

DSTRB : IN std\_logic;

DATA : INOUT std\_logic\_vector(7 downto 0);

PWRITE : IN std\_logic;

PWAIT : OUT std\_logic;

DATO\_RD : IN std\_logic\_vector(7 downto 0);

CE\_RD : OUT std\_logic;

DIR : OUT std\_logic\_vector(7 downto 0);

DIR\_VLD : OUT std\_logic;

DATO : OUT std\_logic\_vector(7 downto 0);

DATO\_VLD : OUT std\_logic

);

END COMPONENT;

COMPONENT epp\_device

port (

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : out std\_logic;

DSTRB : out std\_logic;

ASTRB : out std\_logic;

PWAIT : in std\_logic);

END COMPONENT;

signal ASTRB\_comm:std\_logic;

signal DSTRB\_comm:std\_logic;

signal PWRITE\_comm:std\_logic;

signal PWAIT\_comm:std\_logic;

signal DATA\_comm:std\_logic\_vector(7 downto 0);

--cnt\_epp

signal DATO\_RD\_comm :std\_logic\_vector(7 downto 0):=x"AA";

signal CE\_RD :std\_logic;

signal DIR :std\_logic\_vector(7 downto 0);

signal DIR\_VLD : std\_logic;

signal DATO :std\_logic\_vector(7 downto 0);

signal DATO\_VLD :std\_logic;

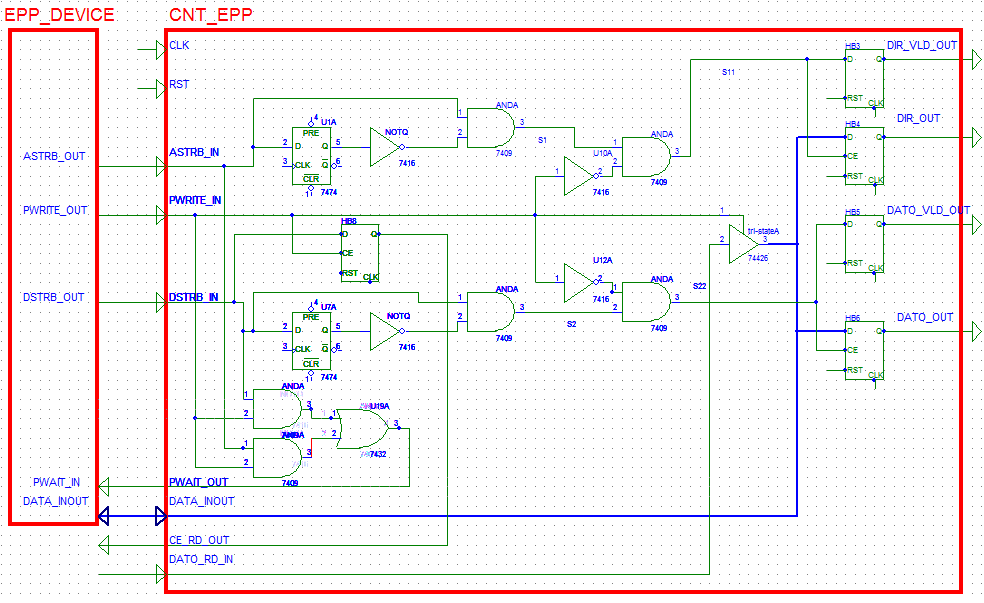
-- Clock period definitions

constant CLK\_period : time := 10 ns;

signal clk : std\_logic;

signal rst : std\_logic:='1';

**1.1.2-** Razonamiento de por qué se ha adoptado la solución presentada.

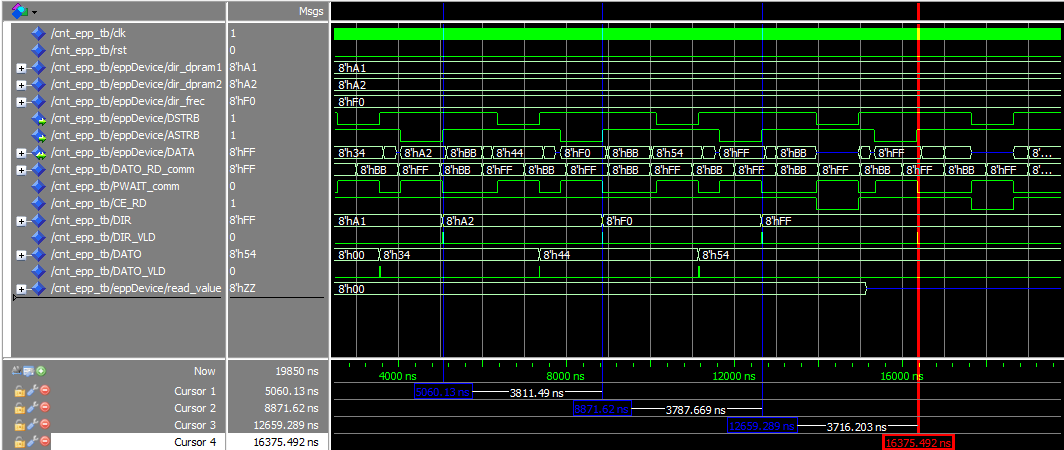
El código presentado anteriormente es la implementación del siguiente esquema:

Gracias a este circuito se ha podido llegar a una solución óptima para el diseño del módulo ***cnt\_epp***.

Éste módulo recibe los datos de temporización del ***epp\_device*** y proporciona las salidas de Dir y Dato y Dir y Dato válidos, junto con señales de pwait y CE\_rd.

El modulo está controlado por el reloj del sistema CLK, luego se trata de un módulo con sincronismo.

**1.1.3.-** Pantallazo de simulación donde se refleje que la entidad ***cnt\_epp*** funciona correctamente. Para ello se realizarán tres ciclos de escritura, uno en cada uno de las direcciones que se van a utilizar en el diseño (*dir\_frec*, *dir\_dpram1* y *dir\_dpram2*) y una operación de lectura en la dirección a elegir por el alumno.



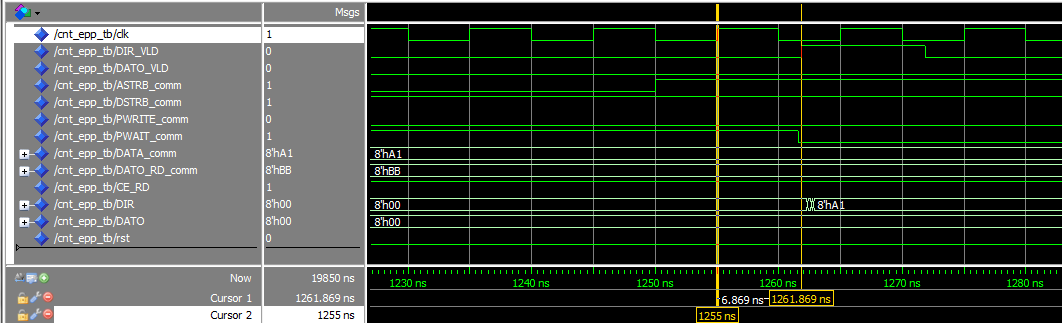
Las tres barras divisoras de color azul de la figura muestra los momentos de de escritura. Se puede ver como el primer caso escribe en la dirección A1h (*dir\_dpram1*) el dato 34h, la segunda división escribe en la posición A2h (*dir\_dpram2*) el dato 44h y el tercer divisor azul escribe en la posición F0h (*dir\_frec*) el dato 54.

Por último el divisor de color rojo, indica un ciclo de lectura, en este caso lee de la posición FFh.

**1.2.1.-** Tabla donde se incluya los recursos utilizados para la implementación del controlador del  
puerto EPP. Los datos anteriores serán extraídos del informe de síntesis.

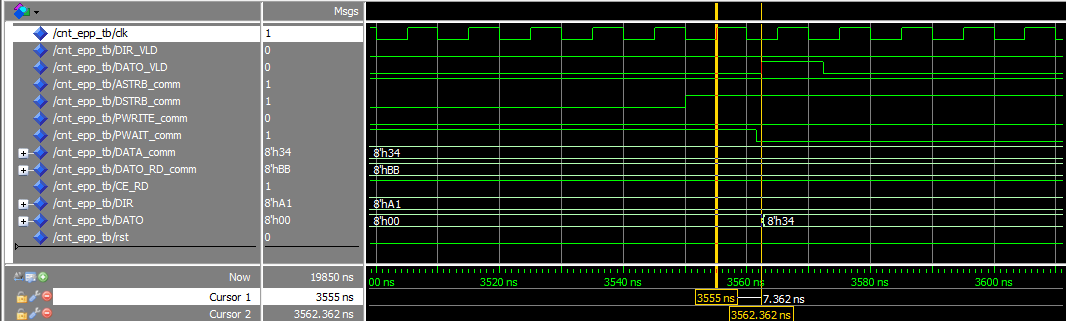
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 21 | 54,576 | 1% |  | |
| Number used as Flip Flops | 21 |  |  |  | |
| Number used as Latches | 0 |  |  |  | |
| Number used as Latch-thrus | 0 |  |  |  | |
| Number used as AND/OR logics | 0 |  |  |  | |
| Number of Slice LUTs | 20 | 27,288 | 1% |  | |
| Number used as logic | 20 | 27,288 | 1% |  | |
| Number using O6 output only | 19 |  |  |  | |
| Number using O5 output only | 0 |  |  |  | |
| Number using O5 and O6 | 1 |  |  |  | |
| Number used as ROM | 0 |  |  |  | |
| Number used as Memory | 0 | 6,408 | 0% |  | |
| Number of occupied Slices | 8 | 6,822 | 1% |  | |
| Number of MUXCYs used | 0 | 13,644 | 0% |  | |
| Number of LUT Flip Flop pairs used | 20 |  |  |  | |
| Number with an unused Flip Flop | 1 | 20 | 5% |  | |
| Number with an unused LUT | 0 | 20 | 0% |  | |
| Number of fully used LUT-FF pairs | 19 | 20 | 95% |  | |
| Number of unique control sets | 1 |  |  |  | |
| Number of slice register sites lost         to control set restrictions | 3 | 54,576 | 1% |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/cnt_epp/cnt_epp_map.xrpt?&DataKey=IOBProperties) | 41 | 218 | 18% |  | |
| IOB Flip Flops | 1 |  |  |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 0 | 232 | 0% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 1 | 376 | 1% |  | |
| Number used as OLOGIC2s | 1 |  |  |  | |
| Number used as OSERDES2s | 0 |  |  |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 3.36 |  |  |  | |

**1.2.2.-** Medida del retardo entre el flanco activo de la señal de reloj y la activación de los puertos DIR\_VLD y DATO\_VLD. Se debe mostrar el resultado a través de pantallazos de simulación utilizando, para ello, dos cursores.

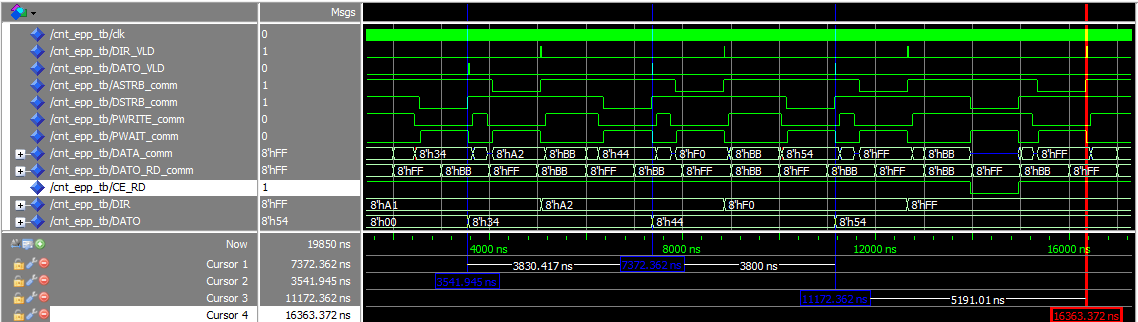


La figura anterior muestra el retardo entre el flanco activo de la señal de reloj y la activación del puero DIR\_VLD, como se aprecia hay un cierto retardo de 6.869 ns.

En el siguiente caso se muestra el retardo entre el flanco activo del reloj y DATO\_VLD el cual es de 7.362 ns, un poco mayor que el caso anterior.



**1.2.3.-** Pantallazo de simulación donde se refleje el que la entidad cnt\_epp función correctamente.



En la simulación temporal se puede apreciar la misma conducta que ocurría en la simulación funcional, luego se deduce que el comportamiento sigue siendo correcto, sólo que en ersta ocasión se incluyen retardos provocados por el hardware.

1. **TOP\_SYSTEM1**

**2.1.1.**- Especificación en VHDL de los modelos *top\_system1* y el correspondiente banco de  
pruebas *top\_sytem1\_tb.*

**A)** Código VHDL de la entidad ***top\_system1***:

entity top\_system1 is

port (

CLK : in std\_logic;

RST : in std\_logic;

ASTRB : in std\_logic;

DSTRB : in std\_logic;

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : in std\_logic;

PWAIT : out std\_logic;

SWITCHES\_I : in std\_logic\_vector(7 downto 0);

PSH\_BUTTON : in std\_logic;

LEDS\_O : out std\_logic\_vector(7 downto 0)

);

end top\_system1;

architecture Behavioral of top\_system1 is

component cnt\_epp

port (

CLK : in std\_logic;

RST : in std\_logic;

ASTRB : in std\_logic;

DSTRB : in std\_logic;

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : in std\_logic;

PWAIT : out std\_logic;

DATO\_RD : in std\_logic\_vector(7 downto 0);

CE\_RD : out std\_logic;

DIR : out std\_logic\_vector (7 downto 0);

DIR\_VLD : out std\_logic;

DATO : out std\_logic\_vector (7 downto 0);

DATO\_VLD : out std\_logic);

end component ;

signal DATO\_RD\_comm : std\_logic\_vector(7 downto 0);

signal CE\_RD\_comm : std\_logic;

signal DIR\_comm : std\_logic\_vector (7 downto 0);

signal DIR\_VLD\_comm : std\_logic;

signal DATO\_comm : std\_logic\_vector (7 downto 0);

signal DATO\_VLD\_comm : std\_logic;

signal DIR\_REG : std\_logic\_vector (7 downto 0);

signal DATO\_REG : std\_logic\_vector (7 downto 0);

begin

cntEpp:cnt\_epp

port map(

CLK=>CLK ,

RST=>RST ,

ASTRB=>ASTRB ,

DSTRB=>DSTRB ,

DATA=>DATA ,

PWRITE=>PWRITE ,

PWAIT=>PWAIT,

DATO\_RD=>DATO\_RD\_comm,

CE\_RD=>CE\_RD\_comm,

DIR=>DIR\_comm,

DIR\_VLD=>DIR\_VLD\_comm,

DATO=>DATO\_comm,

DATO\_VLD=>DATO\_VLD\_comm

);

AddrBiestableD:process (CLK,RST,DIR\_VLD\_comm)

begin

if (RST='1') then

DIR\_REG <= (others=>'0');

elsif (CLK'event and CLK='1')then

if (DIR\_VLD\_comm='1') then

DIR\_REG <= DIR\_comm;

end if;

end if;

end process;

DataBiestableD:process (CLK,RST,DATO\_VLD\_comm)

begin

if (RST='1') then

DATO\_REG <= (others=>'0');

elsif (CLK'event and CLK='1')then

if (DATO\_VLD\_comm='1') then

DATO\_REG <= DATO\_comm;

end if;

end if;

end process;

LEDS\_O <= DATO\_REG when (PSH\_BUTTON='0') else DIR\_REG;

CE\_RDbiestableD:process (CLK,RST)

begin

if (RST='1') then

DATO\_RD\_comm <= (others=>'0');

elsif (CLK'event and CLK='1')then

if (CE\_RD\_comm='1' and DIR\_comm=x"32") then

DATO\_RD\_comm <= SWITCHES\_I;

end if;

end if;

end process;

end Behavioral;

**B)** Código VHDL del banco de pruebas la entidad ***top\_system1\_tb***:

eppDevice:epp\_device

port map(

DATA=>DATA\_comm,

PWRITE=>PWRITE\_comm,

DSTRB=>DSTRB\_comm,

ASTRB=>ASTRB\_comm,

PWAIT=>PWAIT\_comm

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

RST\_process :process

begin

wait for 5 ns;

RST <= '0';

end process;

PSH\_BUTTON\_process :process

begin

PSH\_BUTTON\_comm <= '0';

wait for 400ns;

PSH\_BUTTON\_comm <= '1';

wait for 400ns;

end process;

END;

ENTITY top\_system1\_tb IS

END top\_system1\_tb;

ARCHITECTURE behavior OF top\_system1\_tb IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT top\_system1

PORT(

CLK : IN std\_logic;

RST : IN std\_logic;

ASTRB : IN std\_logic;

DSTRB : IN std\_logic;

DATA : INOUT std\_logic\_vector(7 downto 0);

PWRITE : IN std\_logic;

PWAIT : OUT std\_logic;

SWITCHES\_I : IN std\_logic\_vector(7 downto 0);

PSH\_BUTTON : IN std\_logic;

LEDS\_O : OUT std\_logic\_vector(7 downto 0)

);

END COMPONENT;

COMPONENT epp\_device

port (

DATA : inout std\_logic\_vector(7 downto 0);

PWRITE : out std\_logic;

DSTRB : out std\_logic;

ASTRB : out std\_logic;

PWAIT : in std\_logic);

END COMPONENT;

signal ASTRB\_comm:std\_logic;

signal DSTRB\_comm:std\_logic;

signal PWRITE\_comm:std\_logic;

signal PWAIT\_comm:std\_logic;

signal DATA\_comm:std\_logic\_vector(7 downto 0);

signal SWITCHES\_I\_comm : std\_logic\_vector(7 downto 0) := x"CA";

signal PSH\_BUTTON\_comm : std\_logic := '0';

signal LEDS\_O\_comm : std\_logic\_vector(7 downto 0);

-- Clock period definitions

constant CLK\_period : time := 10 ns;

signal CLK : std\_logic;

signal RST : std\_logic:='1';

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: top\_system1 PORT MAP (

CLK => CLK,

RST => RST,

ASTRB => ASTRB\_comm,

DSTRB => DSTRB\_comm,

DATA => DATA\_comm,

PWRITE => PWRITE\_comm,

PWAIT => PWAIT\_comm,

SWITCHES\_I => SWITCHES\_I\_comm,

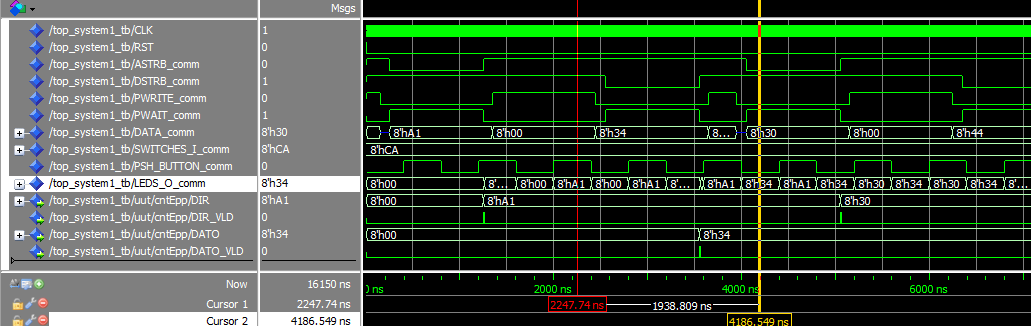
PSH\_BUTTON => PSH\_BUTTON\_comm,

LEDS\_O => LEDS\_O\_comm

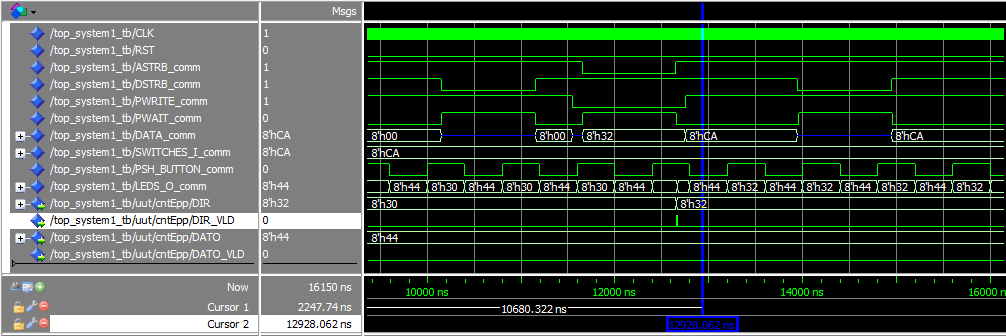
);

**2.1.2.-** Pantallazos de la simulación funcional de los modelos anteriores donde se demuestre que funcionan correctamente.

En el esquema cuando PUSH\_BUTTON esta a nivel alto los leds muestran la dirección registrada (cursor rojo), con un nivel bajo muestra el dato registrado (cursor amarillo).

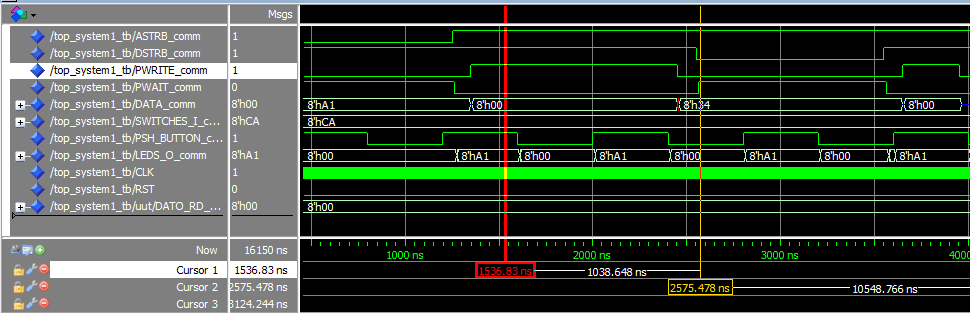


La entrada de los 8 switches programados en el testbench con un valor de CAh pasara a DATO\_RD cuando se realice una lectura de la posición 32h (cursor azul)

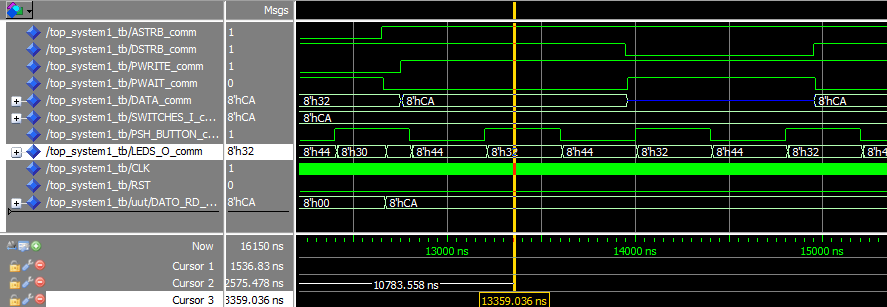


**2.1.3.-** Pantallazos de la simulación temporal de los modelos anteriores donde se demuestre que funcionan correctamente. En este caso se deberá añadir una tabla donde se reflejen los recursos utilizados en su implementación.

1. Pantallazos simulación temporal



Esta tabla muestra la simulación temporal y como se siguen cumpliendo los eventos que se cumplían en la simulación funcional anteriormente explicada. De igual manera sucede para la simulación temporal donde se busca la escritura en la dirección 32h.



**B)** Tabla donde se incluya los recursos utilizados para la implementación del controlador del  
puerto EPP. Los datos anteriores serán extraídos del informe de síntesis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 45 | 54,576 | 1% |  | |
| Number used as Flip Flops | 45 |  |  |  | |
| Number used as Latches | 0 |  |  |  | |
| Number used as Latch-thrus | 0 |  |  |  | |
| Number used as AND/OR logics | 0 |  |  |  | |
| Number of Slice LUTs | 13 | 27,288 | 1% |  | |
| Number used as logic | 9 | 27,288 | 1% |  | |
| Number using O6 output only | 3 |  |  |  | |
| Number using O5 output only | 0 |  |  |  | |
| Number using O5 and O6 | 6 |  |  |  | |
| Number used as ROM | 0 |  |  |  | |
| Number used as Memory | 0 | 6,408 | 0% |  | |
| Number used exclusively as route-thrus | 4 |  |  |  | |
| Number with same-slice register load | 4 |  |  |  | |
| Number with same-slice carry load | 0 |  |  |  | |
| Number with other load | 0 |  |  |  | |
| Number of occupied Slices | 14 | 6,822 | 1% |  | |
| Number of MUXCYs used | 0 | 13,644 | 0% |  | |
| Number of LUT Flip Flop pairs used | 43 |  |  |  | |
| Number with an unused Flip Flop | 2 | 43 | 4% |  | |
| Number with an unused LUT | 30 | 43 | 69% |  | |
| Number of fully used LUT-FF pairs | 11 | 43 | 25% |  | |
| Number of unique control sets | 6 |  |  |  | |
| Number of slice register sites lost         to control set restrictions | 3 | 54,576 | 1% |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/top_system1/top_system1_map.xrpt?&DataKey=IOBProperties) | 31 | 218 | 14% |  | |
| Number of LOCed IOBs | 31 | 31 | 100% |  | |
| IOB Flip Flops | 1 |  |  |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 0 | 232 | 0% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 1 | 376 | 1% |  | |
| Number used as OLOGIC2s | 1 |  |  |  | |
| Number used as OSERDES2s | 0 |  |  |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 2.39 |  |  |  | |

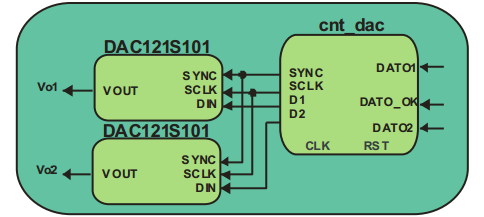
**2.2.1.-** Pantallazos como el de la Figura 28 donde se refleje que la entidad top\_system1  
correctamente para las distintas operaciones.

CAPTURA DIGILENT 1

CAPTURA DIGILENT 2

1. **CNT\_DAC**

**3.1.1.**- Código VHDL del módulo de control de los DACs y del banco de pruebas utilizado en su  
simulación.



**A)** Código VHDL de la entidad ***cnt\_epp***:

SCLKgenerator:process (sclkPulseCounter)

begin

SCLKaux<='0';

if (sclkPulseCounter= sclkPulseValue) then

SCLKaux <= '1';

else

SCLKaux <= '0';

end if;

end process;

SCLK<=SCLKaux;

counter:process(rst,CEcounter,SCLKaux)

begin

if(rst = '1') then

muxSelect <= 0;

elsif (SCLKaux'event and SCLKaux='1') then

if (CEcounter='1') then

if (muxSelect = 15 ) then

muxSelect <= 0;

else

muxSelect <= muxSelect+1;

end if;

end if;

end if;

end process;

muxSelectTX<=to\_unsigned(muxSelect, muxSelectTX'length);

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity cnt\_dac is

port (

CLK : in std\_logic;

RST : in std\_logic;

DATO1 : in std\_logic\_vector(7 downto 0);

DATO2 : in std\_logic\_vector(7 downto 0);

DATO\_OK : in std\_logic;

SYNC : out std\_logic;

SCLK : out std\_logic;

D1 : out std\_logic;

D2 : out std\_logic);

end cnt\_dac;

architecture RTL of cnt\_dac is

type state\_type is (

s0, --HOLDING STATE

s1, --TRANSMISSION STATE

s2 --WAITING STATE

);

signal state: state\_type; --current and next state declaration.

signal SCLKaux: std\_logic;

signal muxSelect: integer range 0 to 15;

signal muxSelectTX: unsigned (3 downto 0);

signal CEcounter: std\_logic;

signal DATO\_1\_16bits: std\_logic\_vector(15 downto 0);

signal DATO\_2\_16bits: std\_logic\_vector(15 downto 0);

signal endTx: std\_logic;

constant sclkPulseValue : unsigned:="1";

signal sclkPulseCounter: integer range 0 to 1;

BEGIN

DATO\_1\_16bits<="0000"&DATO1&"0000";

DATO\_2\_16bits<="0000"&DATO2&"0000";

clkPulseCounter:process (clk,rst)

begin

if (rst='1') then sclkPulseCounter <= 0;

elsif (clk'event and clk='1')then

if (sclkPulseCounter = sclkPulseValue)then sclkPulseCounter <= 0; else sclkPulseCounter <= sclkPulseCounter+ 1; end if;

end if;

end process;

fsm\_proc:process(clk, rst,endTx)

begin

if(rst = '1') then

state <= S0;

SYNC<='1';

CEcounter<='0';

elsif (clk'event and clk = '1') then

case state is

when s0 =>

if (DATO\_OK='0') then

SYNC<='1';

CEcounter<='0'; else --if DATO\_OK=1

state <= s1;

SYNC<='0';

end if;

when s1 =>

CEcounter<='1'; if(endTx = '1') then

SYNC<='0';

state <= s2; end if;

when s2 =>

SYNC<='1';

state <= s0;

end case;

end if;

end process;

end RTL;

mux:process (muxSelectTX,DATO\_1\_16bits,DATO\_2\_16bits)

begin

endTx <= '0';

case muxSelectTX is

when x"0"=> D1 <= DATO\_1\_16bits(15);

D2 <= DATO\_2\_16bits(15);

when x"1"=> D1 <= DATO\_1\_16bits(14);

D2 <= DATO\_2\_16bits(14);

when x"2" => D1 <= DATO\_1\_16bits(13);

D2 <= DATO\_2\_16bits(13);

when x"3" => D1 <= DATO\_1\_16bits(12);

D2 <= DATO\_2\_16bits(12);

when x"4" => D1 <= DATO\_1\_16bits(11);

D2 <= DATO\_2\_16bits(11);

when x"5" => D1 <= DATO\_1\_16bits(10);

D2 <= DATO\_2\_16bits(10);

when x"6" => D1 <= DATO\_1\_16bits(9);

D2 <= DATO\_2\_16bits(9);

when x"7" => D1 <= DATO\_1\_16bits(8);

D2 <= DATO\_2\_16bits(8);

when x"8" => D1 <= DATO\_1\_16bits(7);

D2 <= DATO\_2\_16bits(7);

when x"9" => D1 <= DATO\_1\_16bits(6);

D2 <= DATO\_2\_16bits(6);

when x"A"=> D1 <= DATO\_1\_16bits(5);

D2 <= DATO\_2\_16bits(5);

when x"B"=> D1 <= DATO\_1\_16bits(4);

D2 <= DATO\_2\_16bits(4);

when x"C"=> D1 <= DATO\_1\_16bits(3);

D2 <= DATO\_2\_16bits(3);

when x"D"=> D1 <= DATO\_1\_16bits(2);

D2 <= DATO\_2\_16bits(2);

when x"E"=> D1 <= DATO\_1\_16bits(1);

D2 <= DATO\_2\_16bits(1);

when x"F"=> D1 <= DATO\_1\_16bits(0);

D2 <= DATO\_2\_16bits(0);

endTx <= '1';

when others => D1 <= '0';

D2 <= '0';

end case;

end process;

**B)** Código VHDL del testbench de la entidad ***cnt\_epp\_tb***:

ENTITY cnt\_dac\_tb IS

END cnt\_dac\_tb;

ARCHITECTURE behavior OF cnt\_dac\_tb IS

component cnt\_dac

port (

CLK : in std\_logic;

RST : in std\_logic;

DATO1 : in std\_logic\_vector(7 downto 0);

DATO2 : in std\_logic\_vector(7 downto 0);

DATO\_OK : in std\_logic;

SYNC : out std\_logic;

SCLK : out std\_logic;

D1 : out std\_logic;

D2 : out std\_logic);

end component ;

component DAC121S101

port (

VOUT : out real range 0.0 to 3.5;

SYNC : in std\_logic;

SCLK : in std\_logic;

DIN : in std\_logic);

end component ;

--Inputs

signal CLK : std\_logic := '0';

signal RST : std\_logic := '1';

signal DATO1 : std\_logic\_vector(7 downto 0):=x"00";

signal DATO2 : std\_logic\_vector(7 downto 0):=x"00";

signal DATO\_OK : std\_logic := '0';

--Outputs

signal VOUT1 : real;

signal VOUT2 : real;

signal SYNC\_comm : std\_logic;

signal SCLK\_comm : std\_logic;

signal D1\_comm : std\_logic;

signal D2\_comm : std\_logic;

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

cntdac:cnt\_dac

port map(

CLK=>CLK ,

RST=>RST ,

DATO1=>DATO1 ,

DATO2=>DATO2 ,

DATO\_OK=>DATO\_OK,

SYNC=>SYNC\_comm ,

SCLK=>SCLK\_comm,

D1=>D1\_comm,

D2=>D2\_comm

);

DAC1: DAC121S101

port map (

VOUT =>VOUT1,

SYNC =>SYNC\_comm,

SCLK =>SCLK\_comm,

DIN => D1\_comm);

DAC2: DAC121S101

port map (

VOUT =>VOUT2,

SYNC =>SYNC\_comm,

SCLK =>SCLK\_comm,

DIN=> D2\_comm

);

clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

rst\_process :process

begin

wait for 100 ns;

rst <= '0';

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 800 ns;

DATO1 <=std\_logic\_vector(to\_unsigned(5, 8));

DATO2 <=std\_logic\_vector(to\_unsigned(10, 8));

DATO\_OK<='1';

wait for 20 ns;

DATO\_OK<='0';

wait for 800 ns;

DATO1 <=std\_logic\_vector(to\_unsigned(30, 8));

DATO2 <=std\_logic\_vector(to\_unsigned(40, 8));

DATO\_OK<='1';

wait for 20 ns;

DATO\_OK<='0';

wait for 800 ns;

DATO1 <=std\_logic\_vector(to\_unsigned(85, 8));

DATO2 <=std\_logic\_vector(to\_unsigned(95, 8));

DATO\_OK<='1';

wait for 20 ns;

DATO\_OK<='0';

wait;

end process;

END;

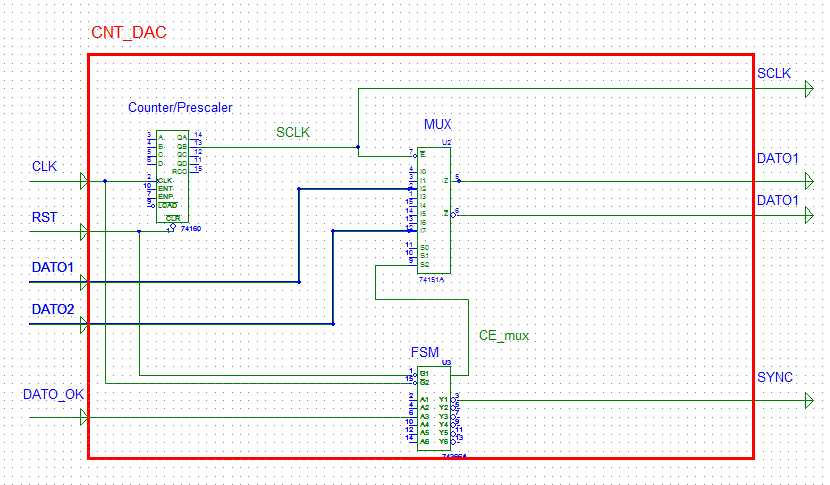
**3.1.2**- Razonamiento de por qué se ha adoptado la solución presentada.

En el diseño de este módulo se ha dos contadores, un modificador de frecuencia de reloj un multiplexor y una maquina de estados.

El primer contador se usa para contar flancos de subida de reloj. La salida del contador es usada por el modificador de frecuencia para crear la señal SCLK.

Los impulsos generados por SCLK son usados por el segundo contador para incrementar la cuenta de selección del multiplexor.

El multiplexor es el encargado de serializar en la salida el valor del dato de entrada.



FSM:

Dato\_ok=1

End\_tx=1

Espera/

SYNC=0

TX/

SYNC=0

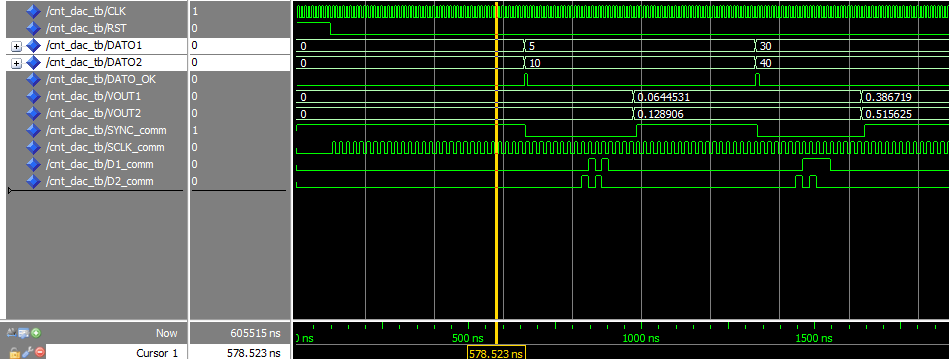
Reposo/

SYNC=1

End\_tx=0

Dato\_ok=0

**3.1.3.**- Pantallazo de simulación funcional donde se refleje que le mencionado circuito funciona  
correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para  
comprobar que para todas las operaciones el módulo funciona correctamente.

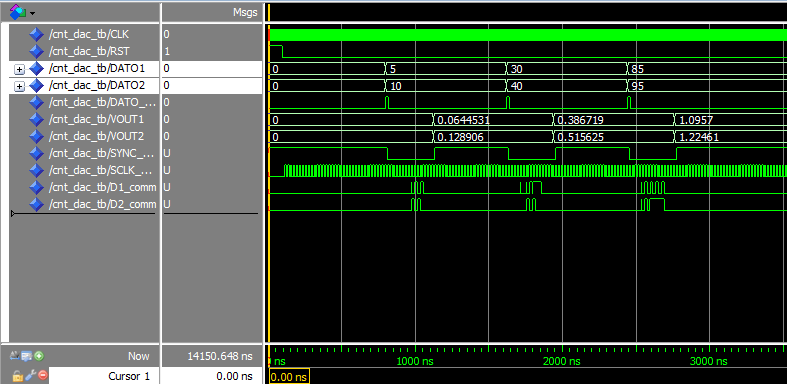


En el esquema anterior se ve como los datos de entrada DATO1 y DATO2 son serializados en D1\_comm y D2\_comm y tras pasar por los DAC121S101 la salida en decimal es aproximado a los valores de entrada.

**3.2.1.**- Tabla donde se incluyan los recursos utilizados para la implementación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 9 | 54,576 | 1% |  | |
| Number used as Flip Flops | 9 |  |  |  | |
| Number used as Latches | 0 |  |  |  | |
| Number used as Latch-thrus | 0 |  |  |  | |
| Number used as AND/OR logics | 0 |  |  |  | |
| Number of Slice LUTs | 15 | 27,288 | 1% |  | |
| Number used as logic | 15 | 27,288 | 1% |  | |
| Number using O6 output only | 13 |  |  |  | |
| Number using O5 output only | 0 |  |  |  | |
| Number using O5 and O6 | 2 |  |  |  | |
| Number used as ROM | 0 |  |  |  | |
| Number used as Memory | 0 | 6,408 | 0% |  | |
| Number of occupied Slices | 7 | 6,822 | 1% |  | |
| Number of MUXCYs used | 0 | 13,644 | 0% |  | |
| Number of LUT Flip Flop pairs used | 15 |  |  |  | |
| Number with an unused Flip Flop | 6 | 15 | 40% |  | |
| Number with an unused LUT | 0 | 15 | 0% |  | |
| Number of fully used LUT-FF pairs | 9 | 15 | 60% |  | |
| Number of unique control sets | 2 |  |  |  | |
| Number of slice register sites lost         to control set restrictions | 7 | 54,576 | 1% |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/cnt_dac/cnt_dac_map.xrpt?&DataKey=IOBProperties) | 23 | 218 | 10% |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 0 | 232 | 0% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 0 | 376 | 0% |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 2.29 |  |  |  | |

**3.2.2.-** Pantallazo de simulación donde se refleje que el controlador de los DACs funciona  
correctamente.



La anterior grafica muestra un comportamiento similar a la simulación funcional, pero se ha tenido que modificar el tiempo del pulso de DATO\_OK ya que en esta simulación con 10 ns no tenia tiempo para modificar valores.

1. **DPRAM\_MEM**

**4.1.1.**- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.

**A)** Código VHDL de la entidad ***dpram\_mem* B)** Código VHDL testbench ***dpram\_mem\_tb***

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity dpram\_mem is

port (

DIN : in std\_logic\_vector(7 downto 0);

ADDR\_IN : in std\_logic\_vector(7 downto 0);

ADDR\_OUT : in std\_logic\_vector(7 downto 0);

DOUT : out std\_logic\_vector(7 downto 0);

WE : in std\_logic;

CLK : in std\_logic;

RST : in std\_logic);

end entity;

architecture rtl of dpram\_mem is

type memory is array (256 downto 0)

of std\_logic\_vector(7 downto 0);

signal mem\_pos: memory;

begin

process(clk)

begin

if (clk'event and clk = '1') then

if (WE = '1') then

mem\_pos(to\_integer(unsigned(ADDR\_IN))) <= DIN;

end if;

end if;

end process;

process(clk)

begin

if (clk'event and clk = '1') then

DOUT <= mem\_pos(to\_integer(unsigned(ADDR\_OUT)));

end if;

end process;

end rtl;

LIBRARY ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

ENTITY dpram\_mem\_tb IS

END dpram\_mem\_tb;

ARCHITECTURE behavior OF dpram\_mem\_tb IS

COMPONENT dpram\_mem

PORT(

DIN : IN std\_logic\_vector(7 downto 0);

ADDR\_IN : IN std\_logic\_vector(7 downto 0);

ADDR\_OUT : IN std\_logic\_vector(7 downto 0);

DOUT : OUT std\_logic\_vector(7 downto 0);

WE : IN std\_logic;

CLK : IN std\_logic;

RST : IN std\_logic

);

END COMPONENT;

--Inputs

signal DIN : std\_logic\_vector(7 downto 0) := (others => '0');

signal ADDR\_IN : std\_logic\_vector(7 downto 0) := (others => '0');

signal ADDR\_OUT : std\_logic\_vector(7 downto 0) := (others => '0');

signal WE : std\_logic := '0';

signal CLK : std\_logic := '0';

signal RST : std\_logic := '1';

signal pulseCounter: integer range 1 to 5;

signal DOUT : std\_logic\_vector(7 downto 0);

constant CLK\_period : time := 10 ns;

BEGIN

uut: dpram\_mem PORT MAP (

DIN => DIN,

ADDR\_IN => ADDR\_IN,

ADDR\_OUT => ADDR\_OUT,

DOUT => DOUT,

WE => WE,

CLK => CLK,

RST => RST

); clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

rst\_process :process

begin

wait for 100 ns;

rst <= '0';

end process;

we\_process :process

begin

we<='1';

wait for 10 ns;

we<='0';

wait for 10 ns;

end process;

**B)**Continuacion testbench ***dpram\_mem\_tb***

din\_process:process (we,rst)

begin

if (rst='1') then

DIN <= x"09";

elsif (we'event and we='1')then DIN<=std\_logic\_vector(unsigned(DIN) + x"01");

end if;

end process;

addressIn\_process:process (we,rst)

begin

if (rst='1') then

ADDR\_IN <= x"00";

elsif (we'event and we='1')then ADDR\_IN<=std\_logic\_vector(unsigned(ADDR\_IN) + x"01");

end if; end process;

addressOut\_process:process (clk,rst)

begin

if (rst='1') then

ADDR\_OUT <= x"00";

elsif (clk'event and clk='0')then

if(pulseCounter = 5)then

pulseCounter <= 1;

ADDR\_OUT<=std\_logic\_vector(unsigned(ADDR\_OUT) + x"01");

else

pulseCounter<=pulseCounter+1;

end if;

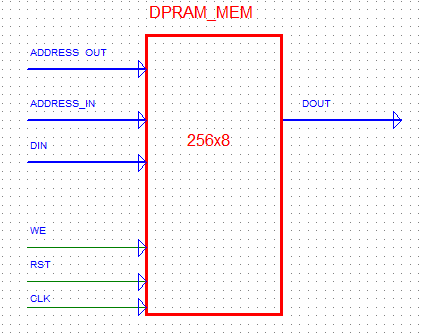
end if; end process;

END;

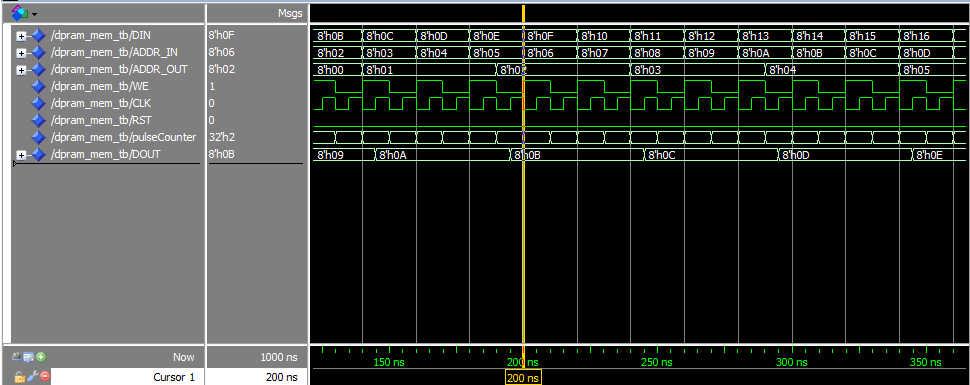
**4.1.2**- Estudio detallado del funcionamiento de la memoria dual port.

La memoria dual port para el ciclo de escritura recibe datos por el puerto DIN y direcciones por el puerto ADDR\_IN. Cuando se activa la señal WE el dato recibido se almacena en un array bidimensional de 256 direcciones de 8 bits cada dirección, en la dirección especifica por ADDR\_IN.

Para el ciclo de lectura recibe la dirección a través del puerto ADDR\_OUT y DOUT envía el dato almacenado en el array correspondiente a esa dirección.



**4.1.3.**- Pantallazo de simulación donde se refleje que el mencionado circuito funciona  
correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para  
comprobar que el módulo funciona correctamente.

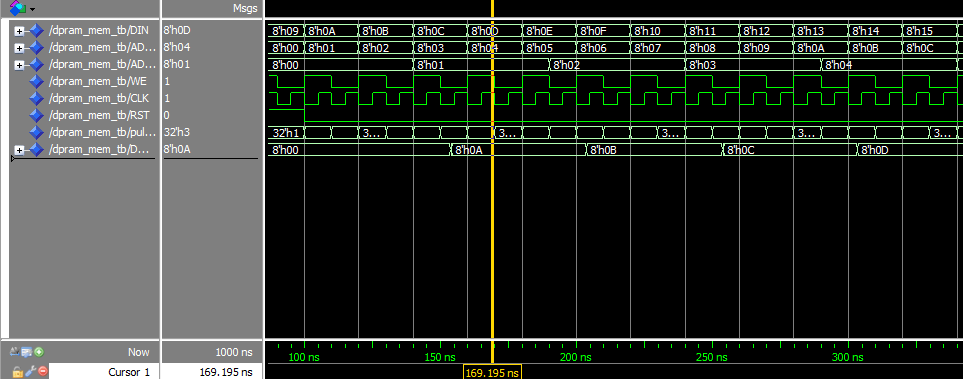


La gráfica anterior muestra la simulación funcional de la memoria ram ***dpram\_mem.*** En el se puede ver como a la salida DOUT tiene el dato que se ha introducido en ADDR\_IN pero la salida se produce con un desplazamiento en el tiempo respecto a la entrada.

**4.2.1.**- Tabla donde se incluyan los recursos utilizados para la implementación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 0 | 54,576 | 0% |  | |
| Number of Slice LUTs | 0 | 27,288 | 0% |  | |
| Number of occupied Slices | 0 | 6,822 | 0% |  | |
| Number of MUXCYs used | 0 | 13,644 | 0% |  | |
| Number of LUT Flip Flop pairs used | 0 |  |  |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/dpram_mem/dpram_mem_map.xrpt?&DataKey=IOBProperties) | 34 | 218 | 15% |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 1 | 232 | 1% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 0 | 376 | 0% |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 1.03 |  |  |  | |

**4.2.2.-** Pantallazo de simulación donde se refleje que el módulo funciona correctamente.



Como se puede ver en la simulación temporal, el comportamiento de la memoria es igual a la simulación funcional pero con algunos retardos en la inicialización y en la transición de las señales provocadas por el hardware.

1. **CNT\_DPRAM**

**5.1.1.**- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.

**A)** Código VHDL de la entidad ***cnt\_dpram***

ADDRESS<=counter;

fsm\_proc:process(clk, rst,DIR\_VLD,DIR,DATO\_VLD,DATO)

begin

if(rst = '1') then

state <= s0;

CEcounter<='0';

we\_dp1<='0';

we\_dp2<='0';

dirPrev<=(others=>'0');

dirActual<=(others=>'0');

clearCounter<='0';

elsif (clk'event and clk = '1') then

case state is

WHEN s0 =>

we\_dp1<='0';

we\_dp2<='0';

clearCounter<='0';

dirActual<=DIR;

if(DIR\_VLD= '1' and (DIR=dir\_dpram1 or DIR=dir\_dpram2))then

state<=s1;

end if;

WHEN s1 =>

if (DATO\_VLD= '1' and dirActual=dirPrev)then

state<=s2;

elsif(DATO\_VLD= '1' and dirActual/=dirPrev) then

state<=s3;

end if;

WHEN s2 =>

CEcounter<='1';

state<=s4;

WHEN s3 =>

clearCounter<='1';

state<=s4;

WHEN s4 =>

if(DIR=dir\_dpram1)then

we\_dp1<='1';

elsif (DIR=dir\_dpram2) then

we\_dp2<='1';

end if; dirPrev<=DIR; CEcounter<='0'; clearCounter<='0'; state<=s0;

end case;

end if;

end process;

end RTL;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity cnt\_dpram is

port (

CLK : in std\_logic;

RST : in std\_logic;

DIR : in std\_logic\_vector (7 downto 0);

DIR\_VLD : in std\_logic;

DATO : in std\_logic\_vector (7 downto 0);

DATO\_VLD : in std\_logic;

ADDRESS : out std\_logic\_vector(7 downto 0);

DATA : out std\_logic\_vector(7 downto 0);

WE\_DP1 : out std\_logic;

WE\_DP2 : out std\_logic);

end cnt\_dpram;

architecture RTL of cnt\_dpram is

constant dir\_dpram1 : std\_logic\_vector( 7 downto 0) := x"A1";

constant dir\_dpram2 : std\_logic\_vector( 7 downto 0) := x"A2";

type state\_type is (

s0, --HOLDING STATE

s1, --WAIT DATA STATE

s2, --CLOCK ENABLE STATE

s3, --CLEAR STATE

s4 --WE\_DP ENABLE STATE

);

signal state: state\_type

signal dirActual: std\_logic\_vector(7 downto 0);

signal dirPrev: std\_logic\_vector(7 downto 0);

signal CEcounter: std\_logic;

signal clearCounter: std\_logic;

signal addressTX: std\_logic\_vector(7 downto 0);

signal counter: std\_logic\_vector(7 downto 0);

begin -- RTL

DATA<=DATO;

counter\_proc:process(clk,rst,CEcounter,clearCounter)

begin

if (rst='1')then

counter<=(others=>'0');

elsif (clk'event and clk = '1') then

if(CEcounter='1') then

if(counter=x"FF")then

counter<=x"00";

else

counter<=std\_logic\_vector(unsigned(counter) + 1);

end if; elsif(clearCounter='1') then counter<=(others=>'0');

end if;

end if;

end process;

**B)** Código VHDL del testbench entidad ***cnt\_dpram\_tb***

DATO\_process :process

begin

if(rst='1')then

DATO<=(others=>'0');

wait for 500 ns;

end if;

DATO <= x"55";

DATO\_VLD<='1';

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"69";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"85";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"42";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"65";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"00";

wait for 10 ns;

wait;

end process;

begin

if(rst='1')then

DIR<=(others=>'0');

wait for 250 ns;

end if;

DIR <= x"a2";

DIR\_VLD<='1';

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a2";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

wait;

end process;

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY cnt\_dpram\_tb IS

END cnt\_dpram\_tb;

ARCHITECTURE behavior OF cnt\_dpram\_tb IS

COMPONENT cnt\_dpram

PORT(

CLK : IN std\_logic;

RST : IN std\_logic;

DIR : IN std\_logic\_vector(7 downto 0);

DIR\_VLD : IN std\_logic;

DATO : IN std\_logic\_vector(7 downto 0);

DATO\_VLD : IN std\_logic;

ADDRESS : OUT std\_logic\_vector(7 downto 0);

DATA : OUT std\_logic\_vector(7 downto 0);

WE\_DP1 : OUT std\_logic;

WE\_DP2 : OUT std\_logic

);

END COMPONENT;

--Inputs

signal CLK : std\_logic := '0';

signal RST : std\_logic := '1';

signal DIR : std\_logic\_vector(7 downto 0) := (others => '0');

signal DIR\_VLD : std\_logic := '0';

signal DATO : std\_logic\_vector(7 downto 0) := (others => '0');

signal DATO\_VLD : std\_logic := '0';

--Outputs

signal ADDRESS : std\_logic\_vector(7 downto 0);

signal DATA : std\_logic\_vector(7 downto 0);

signal WE\_DP1 : std\_logic;

signal WE\_DP2 : std\_logic;

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

uut: cnt\_dpram PORT MAP (

CLK => CLK,

RST => RST,

DIR => DIR,

DIR\_VLD => DIR\_VLD,

DATO => DATO,

DATO\_VLD => DATO\_VLD,

ADDRESS => ADDRESS,

DATA => DATA,

WE\_DP1 => WE\_DP1,

WE\_DP2 => WE\_DP2

);

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

RST\_process :process

begin

wait for 20 ns;

RST <= '0';

end process;

**5.1.2**- Razonamiento de por qué se ha adoptado la solución presentada.

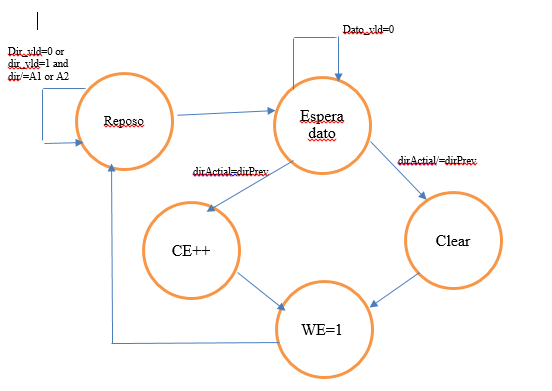
Este módulo es el controlador de las memorias dual port. Su misión es indicar que dirección de la memoria debe ser accedida dada el modulo de memoria a donde se quiera acceder.

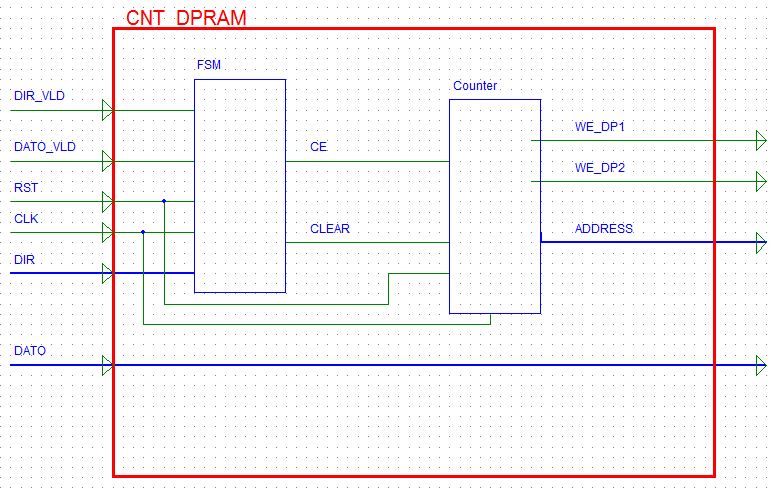
Por ejemplo; si se desea acceder el modulo de memoria A1h repetidas veces, este modulo se encarga de escribir indicar posiciones distintas de escritura dentro del mismo modulod de memoria A1h

El modulo esta compuesto por un contador y un una maquina de estados.

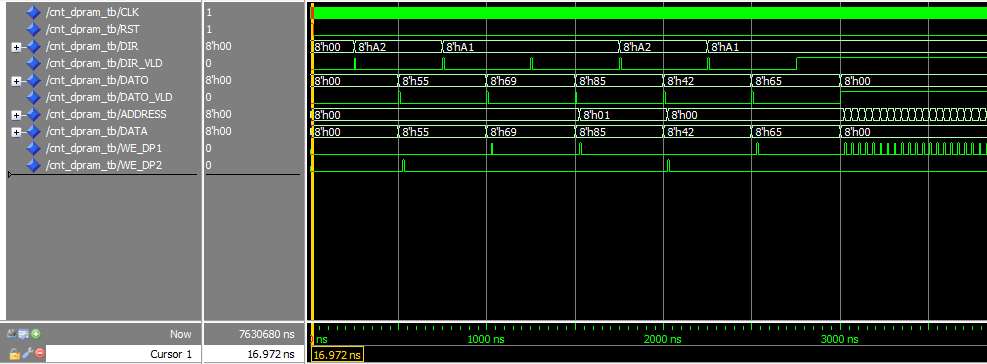
El funcionamiento de la maquina de estados regula el funcionamiento general del sistema.

La maquina de estados se encarga de estudiar cuando se ha recibido una nueva dirección y comprueba si es una dirección para el mismo modulo o para el modulo asignado la ejecución anterior.





**5.1.3.**- Pantallazo de simulación donde se refleje que el mencionado circuito funciona  
correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para  
comprobar que el módulo funciona correctamente.



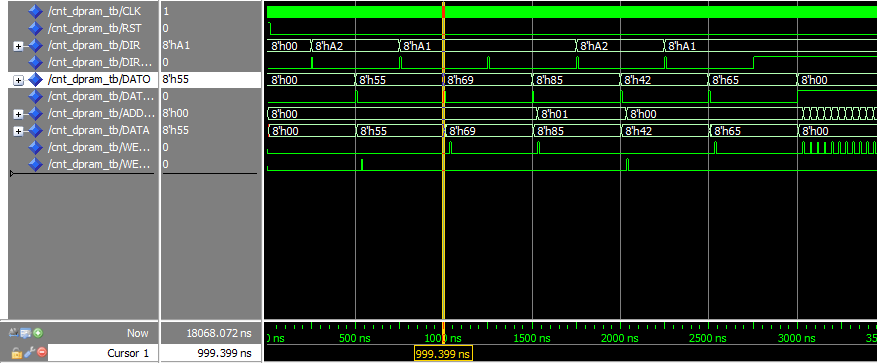
Esta grafica muestra el funcionamiento del modulo.

El proceso se inicializa con DIR=A2 y una vez que se recibe DATO\_VLD la salida ADDRESS escribe un 0 ya que es la primera vez que se escribe en la memoria A2. El siguiente paso de DIR es A1, y se aprecia como ADDRESS mantiene el valor 0 ya que también es la primera vez que se accede la memoria A1. El siguiente ciclo vuelve a ser DIR=A1, como esta es la segunda referencia consecutiva a esta memoria ADDRESS es ahora 01h.

**5.2.1.**- Tabla donde se incluyan los recursos utilizados para la implementación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 31 | 54,576 | 1% |  | |
| Number used as Flip Flops | 31 |  |  |  | |
| Number used as Latches | 0 |  |  |  | |
| Number used as Latch-thrus | 0 |  |  |  | |
| Number used as AND/OR logics | 0 |  |  |  | |
| Number of Slice LUTs | 21 | 27,288 | 1% |  | |
| Number used as logic | 21 | 27,288 | 1% |  | |
| Number using O6 output only | 15 |  |  |  | |
| Number using O5 output only | 0 |  |  |  | |
| Number using O5 and O6 | 6 |  |  |  | |
| Number used as ROM | 0 |  |  |  | |
| Number used as Memory | 0 | 6,408 | 0% |  | |
| Number of occupied Slices | 10 | 6,822 | 1% |  | |
| Number of MUXCYs used | 0 | 13,644 | 0% |  | |
| Number of LUT Flip Flop pairs used | 34 |  |  |  | |
| Number with an unused Flip Flop | 4 | 34 | 11% |  | |
| Number with an unused LUT | 13 | 34 | 38% |  | |
| Number of fully used LUT-FF pairs | 17 | 34 | 50% |  | |
| Number of unique control sets | 4 |  |  |  | |
| Number of slice register sites lost         to control set restrictions | 1 | 54,576 | 1% |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/cnt_dpram/cnt_dpram_map.xrpt?&DataKey=IOBProperties) | 38 | 218 | 17% |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 0 | 232 | 0% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 0 | 376 | 0% |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 3.13 |  |  |  | |

**5.2.2.-** Pantallazo de simulación donde se refleje que el módulo de representación funciona  
correctamente.



El comportamiento de la simulación temporal es similar a la simulación funcional. El funcionamiento sigue siendo correcto.

1. **GEN\_DIR**

**6.1.1.**- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.

**A)** Código VHDL de la entidad ***gen\_dir***

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity gen\_dir is

generic ( N : integer := 68 );

port (

CLK : in std\_logic;

RST : in std\_logic;

DIR : in std\_logic\_vector (7 downto 0);

DIR\_VLD : in std\_logic;

DATO : in std\_logic\_vector (7 downto 0);

DATO\_VLD : in std\_logic;

ADDR\_OUT : out std\_logic\_vector(7 downto 0);

DATO\_OK : out std\_logic);

end gen\_dir;

architecture rtl of gen\_dir is

constant dir\_frec : std\_logic\_vector( 7 downto 0) := x"F0";

type state\_type is (

s0, --HOLDING STATE

s1 --WAIT DATA STATE

);

signal state: state\_type; --current and next state declaration.

signal valor\_freq: std\_logic\_vector( 7 downto 0);

signal cnt: std\_logic\_vector( 15 downto 0);

signal valor\_freq\_comm: std\_logic\_vector( 15 downto 0);

signal CE: std\_logic;

signal prescalerCounter: unsigned(6 downto 0); --7 bits to cover generic 68

signal dato\_ok\_trigger: std\_logic;

begin

prescaler\_proc:process (CLK,RST)

begin

if (RST='1') then

CE <= '0';

prescalerCounter<=(others=>'0');

elsif (CLK'event and CLK='1')then

CE <= '0';

prescalerCounter<=prescalerCounter+1;

if (prescalerCounter = N)then

CE <= '1';

prescalerCounter<=(others=>'0');

end if;

end if;

end process;

fsm\_proc:process(clk, rst,valor\_freq,DIR\_VLD,DIR,DATO\_VLD)

begin

if(rst = '1') then

state <= s0;

valor\_freq<=(others=>'0');

elsif (clk'event and clk = '1') then

case state is

WHEN s0 =>

if(DIR\_VLD= '1' and DIR=dir\_frec)then state<=s1; end if;

WHEN s1 =>

if (DATO\_VLD= '1')then

valor\_freq<=DATO;

state<=s0;

end if;

end case;

end if;

end process;

valor\_freq\_comm<=std\_logic\_vector(unsigned(valor\_freq) + unsigned(cnt));

registro:process (clk,rst,valor\_freq\_comm,CE)

begin

if (rst='1') then

cnt <= (others=>'0');

elsif (clk'event and clk='1')then

if (CE = '1')then

cnt<=valor\_freq\_comm;

end if;

end if;

end process;

ADDR\_OUT<=cnt(15 downto 8);

dato\_ok\_trigger<='1' when (cnt /= x"0000") else ('0');

data\_ok\_proc:process (clk,rst,prescalerCounter)

begin

if (rst='1') then

DATO\_OK<='0';

elsif (clk'event and clk='1')then

if (prescalerCounter = 1)then

DATO\_OK<='1';

else

DATO\_OK<='0';

end if;

end if;

end process;

end rtl;

**B)** Código VHDL del testbench de la entidad ***gen\_dir\_tb***

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY gen\_dir\_tb IS

END gen\_dir\_tb;

ARCHITECTURE behavior OF gen\_dir\_tb IS

COMPONENT gen\_dir

PORT(

CLK : IN std\_logic;

RST : IN std\_logic;

DIR : IN std\_logic\_vector(7 downto 0);

DIR\_VLD : IN std\_logic;

DATO : IN std\_logic\_vector(7 downto 0);

DATO\_VLD : IN std\_logic;

ADDR\_OUT : OUT std\_logic\_vector(7 downto 0);

DATO\_OK : OUT std\_logic

);

END COMPONENT;

--Inputs

signal CLK : std\_logic := '0';

signal RST : std\_logic := '1';

signal DIR : std\_logic\_vector(7 downto 0) := (others => '0');

signal DIR\_VLD : std\_logic := '0';

signal DATO : std\_logic\_vector(7 downto 0) := (others => '0');

signal DATO\_VLD : std\_logic := '0';

--Outputs

signal ADDR\_OUT : std\_logic\_vector(7 downto 0);

signal DATO\_OK : std\_logic;

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: gen\_dir PORT MAP (

CLK => CLK,

RST => RST,

DIR => DIR,

DIR\_VLD => DIR\_VLD,

DATO => DATO,

DATO\_VLD => DATO\_VLD,

ADDR\_OUT => ADDR\_OUT,

DATO\_OK => DATO\_OK

);

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

RST\_process :process

begin

wait for 10 ns;

RST <= '0';

end process;

DIR\_VLD\_process :process

begin

if(rst='1')then

DIR<=(others=>'0');

wait for 250 ns;

end if;

DIR <= x"a2";

DIR\_VLD<='1';

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"F0";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a2";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"a1";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='1';

DIR <= x"F0";

wait for 10 ns;

DIR\_VLD<='0';

wait for 490 ns;

DIR\_VLD<='0';

DIR <= x"00";

wait for 10 ns; wait;

end process;

END;

DATO\_process :process

begin

if(rst='1')then

DATO<=(others=>'0');

wait for 500 ns;

end if;

DATO <= x"55";

DATO\_VLD<='1';

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"69";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"85";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"42";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"65";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='1';

DATO <= x"00";

wait for 10 ns;

DATO\_VLD<='0';

wait for 490 ns;

DATO\_VLD<='0';

DATO <= x"00";

wait for 10 ns;

wait;

end process;

**6.1.2**- Razonamiento de por qué se ha adoptado la solución presentada.

Siguiendo el esquema proporcionado, se han creado los siguientes componentes;

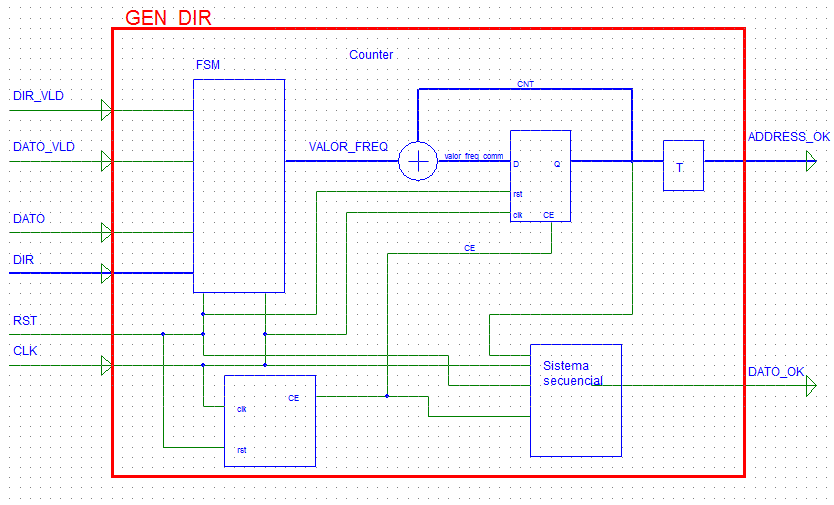
Un prescaler con un factor de división de 68 pulsos de CLK.

Una maquina de estados que espera a una dirección valida y un dato valido. Un vez recibido ambos valores validos, comprueba a su vez que la DIR introducida corresponde con la dirección F0h. Si es asi el valor del puerto DATO de entrada se suma al valor que sale del biestable.

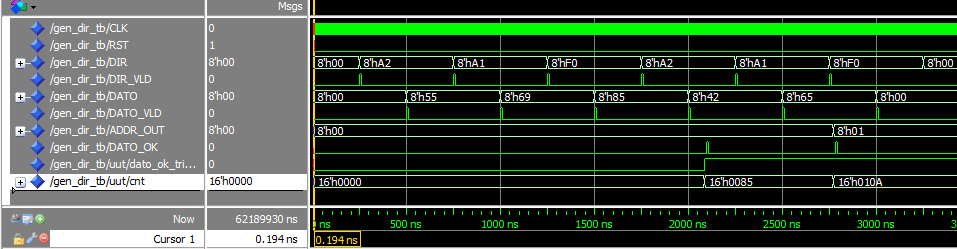
El biestable mantiene el valor del DATO durante un ciclo del prescaler y retorna el valor al sumador para sumarlo con un nuevo valor de entrada.

Por ultimo se ha diseñado un sistema secuencial en el cual, cuando el contador del prescaler alcanza el valor 1 y la salida de cnt es distinto al inicio (“0000”h) se activa la salida DATO\_OK.

Se ha tomado el inicio en 0h de cnt porque es un valor que tras ser sumado repetidas veces dos cualesquiera nunca originara el valor 0h.



**6.1.3.**- Pantallazo de simulación donde se refleje que el mencionado circuito funciona  
correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para  
comprobar que el módulo funciona correctamente.

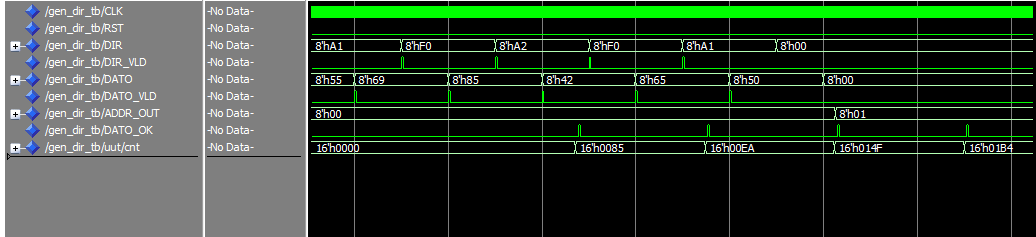


En la gráfica anterior cuando llega una dirección en DIR igual a F0h y se detecta un dato valido (DATO\_VLD=1) y la salida para el primer caso de ADDR\_OUT es igual a 00h.

**6.2.1.**- Tabla donde se incluyan los recursos utilizados para la implementación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device Utilization Summary** | | | | | [**[-]**](?&ExpandedTable=DeviceUtilizationSummary) |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** | |
| Number of Slice Registers | 34 | 54,576 | 1% |  | |
| Number used as Flip Flops | 34 |  |  |  | |
| Number used as Latches | 0 |  |  |  | |
| Number used as Latch-thrus | 0 |  |  |  | |
| Number used as AND/OR logics | 0 |  |  |  | |
| Number of Slice LUTs | 32 | 27,288 | 1% |  | |
| Number used as logic | 31 | 27,288 | 1% |  | |
| Number using O6 output only | 22 |  |  |  | |
| Number using O5 output only | 7 |  |  |  | |
| Number using O5 and O6 | 2 |  |  |  | |
| Number used as ROM | 0 |  |  |  | |
| Number used as Memory | 0 | 6,408 | 0% |  | |
| Number used exclusively as route-thrus | 1 |  |  |  | |
| Number with same-slice register load | 0 |  |  |  | |
| Number with same-slice carry load | 1 |  |  |  | |
| Number with other load | 0 |  |  |  | |
| Number of occupied Slices | 14 | 6,822 | 1% |  | |
| Number of MUXCYs used | 16 | 13,644 | 1% |  | |
| Number of LUT Flip Flop pairs used | 40 |  |  |  | |
| Number with an unused Flip Flop | 8 | 40 | 20% |  | |
| Number with an unused LUT | 8 | 40 | 20% |  | |
| Number of fully used LUT-FF pairs | 24 | 40 | 60% |  | |
| Number of unique control sets | 3 |  |  |  | |
| Number of slice register sites lost         to control set restrictions | 6 | 54,576 | 1% |  | |
| Number of bonded [IOBs](C://Users/christopher/Dropbox/workspace/modelado14/gen_dir/gen_dir_map.xrpt?&DataKey=IOBProperties) | 29 | 218 | 13% |  | |
| Number of RAMB16BWERs | 0 | 116 | 0% |  | |
| Number of RAMB8BWERs | 0 | 232 | 0% |  | |
| Number of BUFIO2/BUFIO2\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFIO2FB/BUFIO2FB\_2CLKs | 0 | 32 | 0% |  | |
| Number of BUFG/BUFGMUXs | 1 | 16 | 6% |  | |
| Number used as BUFGs | 1 |  |  |  | |
| Number used as BUFGMUX | 0 |  |  |  | |
| Number of DCM/DCM\_CLKGENs | 0 | 8 | 0% |  | |
| Number of ILOGIC2/ISERDES2s | 0 | 376 | 0% |  | |
| Number of IODELAY2/IODRP2/IODRP2\_MCBs | 0 | 376 | 0% |  | |
| Number of OLOGIC2/OSERDES2s | 0 | 376 | 0% |  | |
| Number of BSCANs | 0 | 4 | 0% |  | |
| Number of BUFHs | 0 | 256 | 0% |  | |
| Number of BUFPLLs | 0 | 8 | 0% |  | |
| Number of BUFPLL\_MCBs | 0 | 4 | 0% |  | |
| Number of DSP48A1s | 0 | 58 | 0% |  | |
| Number of ICAPs | 0 | 1 | 0% |  | |
| Number of MCBs | 0 | 2 | 0% |  | |
| Number of PCILOGICSEs | 0 | 2 | 0% |  | |
| Number of PLL\_ADVs | 0 | 4 | 0% |  | |
| Number of PMVs | 0 | 1 | 0% |  | |
| Number of STARTUPs | 0 | 1 | 0% |  | |
| Number of SUSPEND\_SYNCs | 0 | 1 | 0% |  | |
| Average Fanout of Non-Clock Nets | 2.97 |  |  |  | |

**6.2.2.-** Pantallazo de simulación temporal donde se refleje que el módulo de representación funciona  
correctamente.



Analizando la simulación temporal se veque con la primera lectura de DRI=F0 se obtine el dato 85h, los bits de mayor peso pasan a ADDR\_OUT, es decir el valor será de 0h y se activa el DATO\_OK. El segundo valor de lectura para F= es 65h sumando ambos se obtiene EAh. En ADDR\_OUT de nuevo solo pasan los de mayor peso luego sigue siendo 0h. Una vez pasado el periodo del prescaler, como no hay una entrada valida nueva se mantiene el valor anterior (65h) que sumado al valor anterior EAh resulta en 014F. Este nuevo valor provoca en la salida ADDR\_OUT el valor de 01h ya que solo coge los de mayor peso del bus CNT.