Memoria MEMORIA_DUAL_PORT

La memoria dual por será la encargada de almacenar un periodo de cada una de las señales a generar. Esta memoria tendrá una organización de 256x8 bits, teniendo un puerto de sólo escritura y otro se sólo lectura. Su finalidad tal y como se ha dicho es almacenar en las direcciones que recibe el contenido que es enviado desde los otros apartados del programa pudiendo posteriormente ser leído dicho dato.

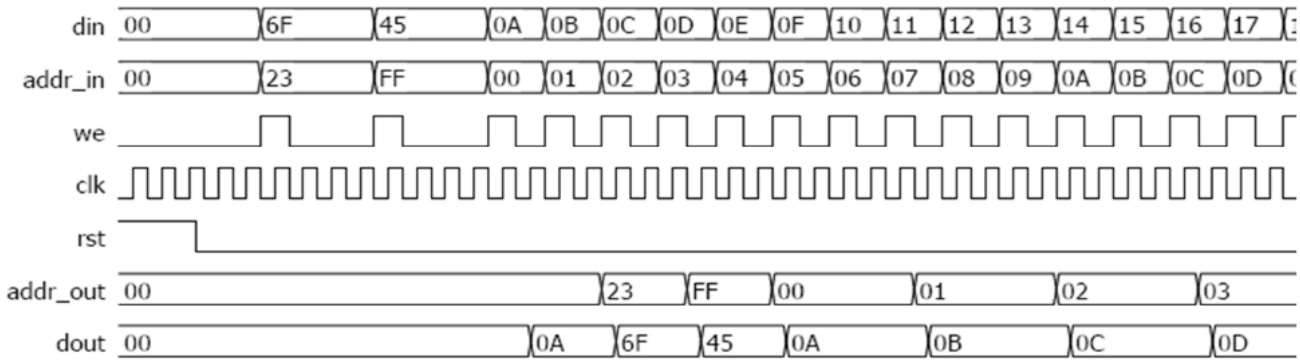
Las entradas y salidas de este apartado son las citadas a continuación ademas se especifica el tipo de dato que son y si son de entrada o de salida:

```
DIN : in std_logic_vector(7 downto 0);
ADDR_IN : in std_logic_vector(7 downto 0);
WE : in std_logic;
CLK : in std_logic;
RST : in std_logic;
ADDR_OUT : in std_logic_vector(7 downto 0);
DOUT : out std_logic_vector(7 downto 0));
```

La finalidad de los puestos citado es:

DIN. Bus de datos correspondiente al puerto de solo escritura.
ADDR_IN. Bus de dirección del puerto de sólo escritura.
WE. Señal de validación de la operación de escritura. Es activa a nivel alto.
DOUT. Bus de datos correspondientes al puerto de solo lectura.
ADDR_OUT. Bus de dirección del puerto de sólo lectura.

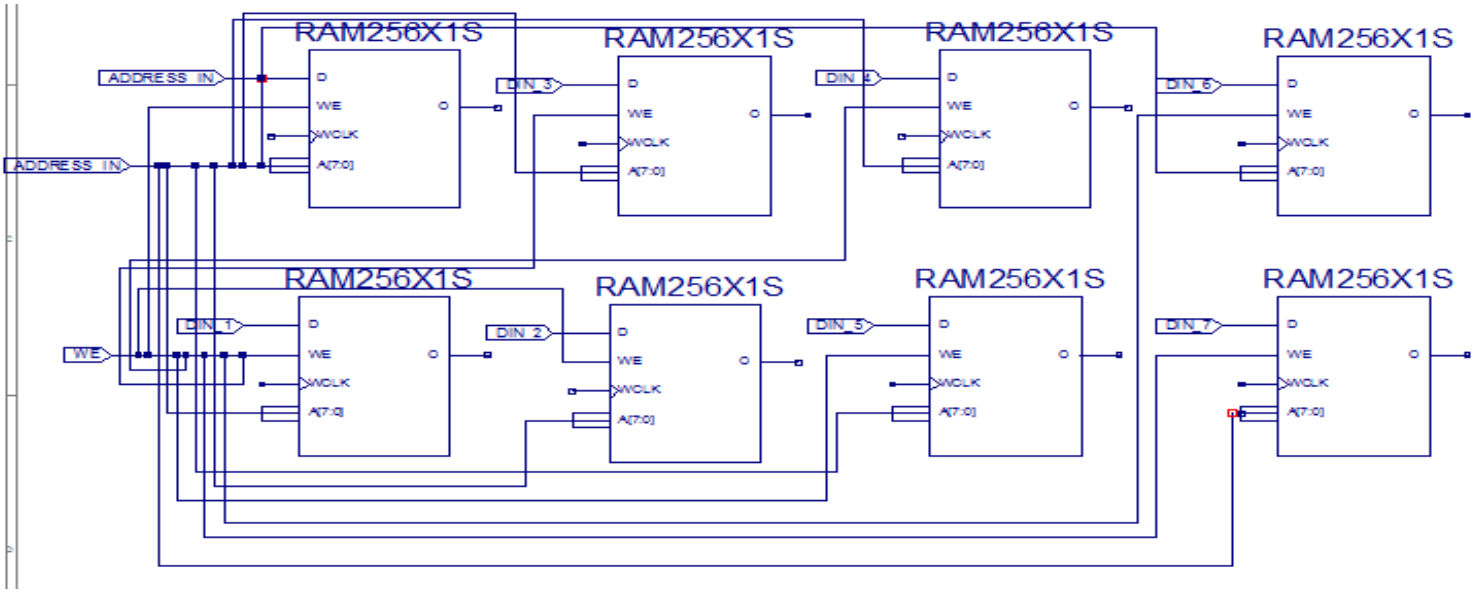
El esquema de funcionamiento de la memoria es suministrador por el profesor por lo que solo es necesario comprender su funcionamiento y como ha de utilizarse correctamente y obteniendo finalmente un resultado parecido al descrito en la imagen.



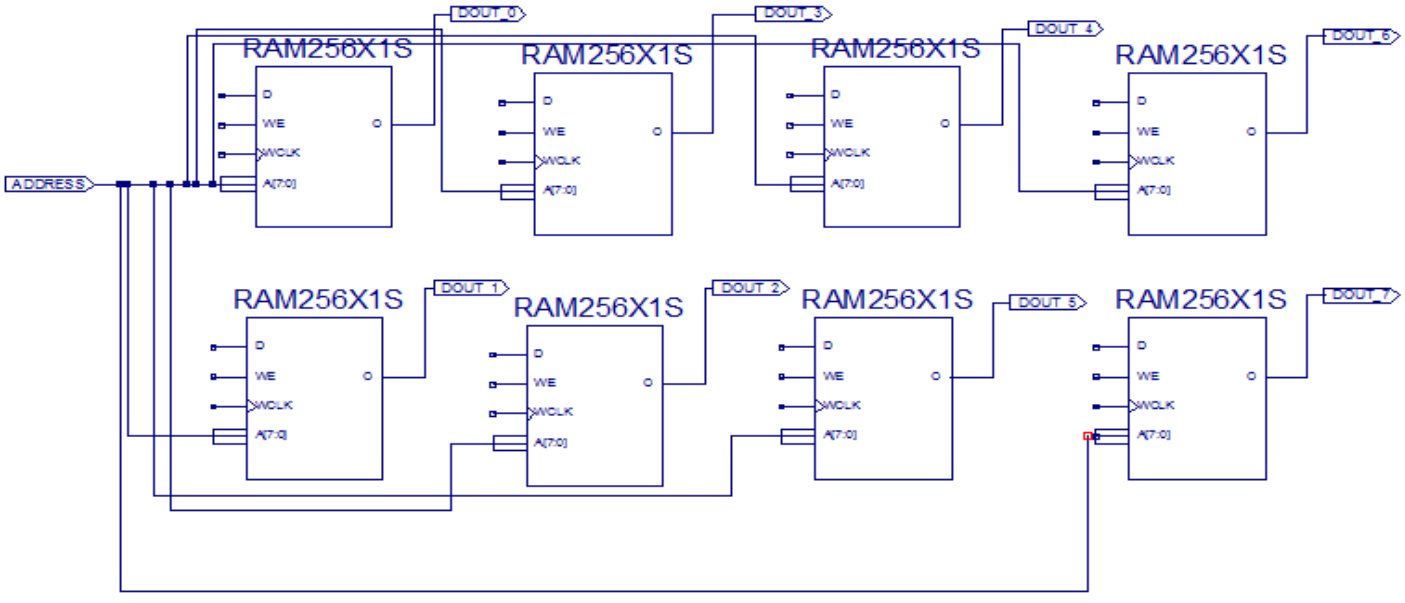
Este código a sido suministrado por el profesor por lo que solo se ha de entender si funcionamiento y verificar si funciona correctamente en los casos posibles.

Circuitos digitales y funcionamiento

Escritura: Este circuito seria algo aproximado a lo mostrado en la imagen adjunta a continuación, su finalidad es que una vez que la dirección donde se quiere almacenar la información y que la información esta preparada se produce la señal de WE la cual permite guardar el contenido de DIN en la dirección que queremos.



Lectura: De la misma manera que en la escritura en este acaso cuando queremos ver el contenido de la memoria solo hemos de insertar la direccion donde queremos hacer la lectura y se volcara el contenido de esta en la salida.



Tanto en la escritura como en la lectura las acciones se realizan durante CLK'event and CLK = '1' pero no se ha implementado para simplificar el dibujo.

Código Memoria_dual

```
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);
    WE       : in  std_logic;
    CLK      : in  std_logic;
    RST      : in  std_logic;
    ADDR_OUT : in  std_logic_vector(7 downto 0);
    DOUT     : out std_logic_vector(7 downto 0));
end entity;

architecture rtl of dpram_mem is
  type mem_type is array (255 downto 0) of std_logic_vector(7 downto 0);
  signal mem : mem_type;
  attribute ram_style : string;
  attribute ram_style of mem : signal is "block";

begin

  process (CLK) is
  begin
    if CLK'event and CLK = '1' then
      if WE='1' then
        mem(to_integer(unsigned(ADDR_IN))) <= DIN;
      end if;
    end if;
  end process;

  process (CLK, RST) is
  begin
    if RST = '0' then
      DOUT <= (others => '0');
    elsif CLK'event and CLK = '1' then
      DOUT <= mem(to_integer(unsigned(ADDR_OUT)));
    end if;
  end process;


end rtl;
```

Código TestBench Memoria_dual

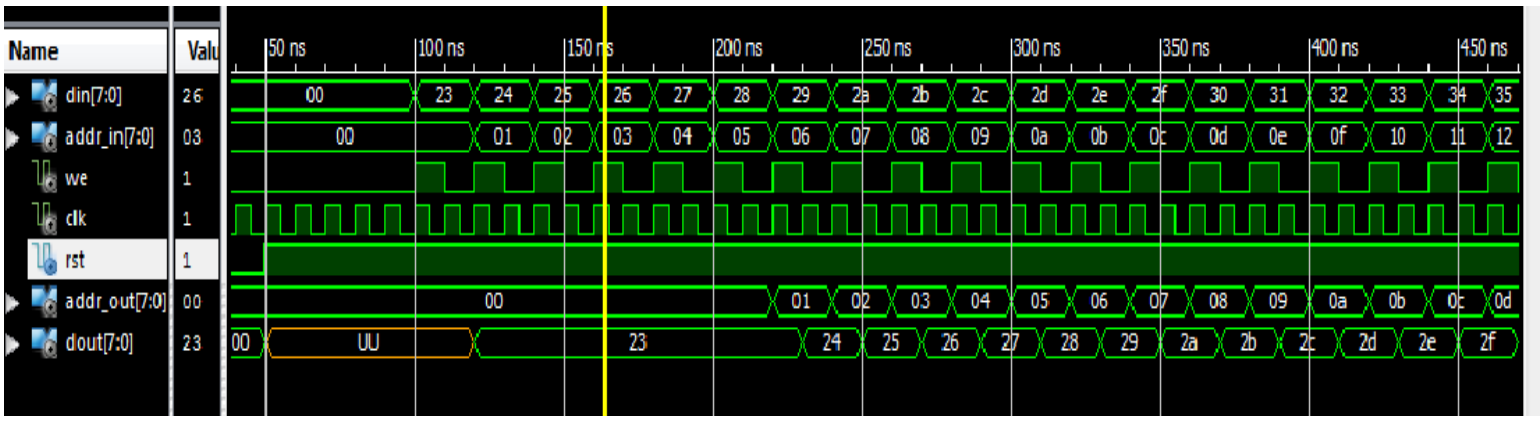
Con este código se busca probar que todos los casos posibles tanto de lectura como de escritura funcionan correctamente.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

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

entity dpram_mem_tb is

end entity dpram_mem_tb;

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

architecture dpram_mem of dpram_mem_tb is

  -- component ports
  signal DIN      : std_logic_vector(7 downto 0) :=(others =>'0');
  signal ADDR_IN  : std_logic_vector(7 downto 0) :=(others =>'0');
  signal WE       : std_logic:='0';
  signal CLK      : std_logic:='1' ;
  signal RST      : std_logic;
  signal ADDR_OUT : std_logic_vector(7 downto 0) :=(others =>'0');
  signal DOUT     : std_logic_vector(7 downto 0);


begin  -- architecture dpram_mem

  -- component instantiation
  DUT: entity work.dpram_mem
    port map (
      DIN      => DIN,
      ADDR_IN  => ADDR_IN,
      WE       => WE,
      CLK      => CLK,
      RST      => RST,
      ADDR_OUT => ADDR_OUT,
      DOUT     => DOUT);

  -- clock generation
  CLK <= not CLK after 5 ns;
  RST <= '0', '1' after 50 ns;

  process is
  begin  -- process escritura donde se realizan las inserciones de los datos
    wait for 100 ns;
    for i in 0 to 255 loop
      DIN <=  std_logic_vector(to_unsigned(35+i, 8));
```

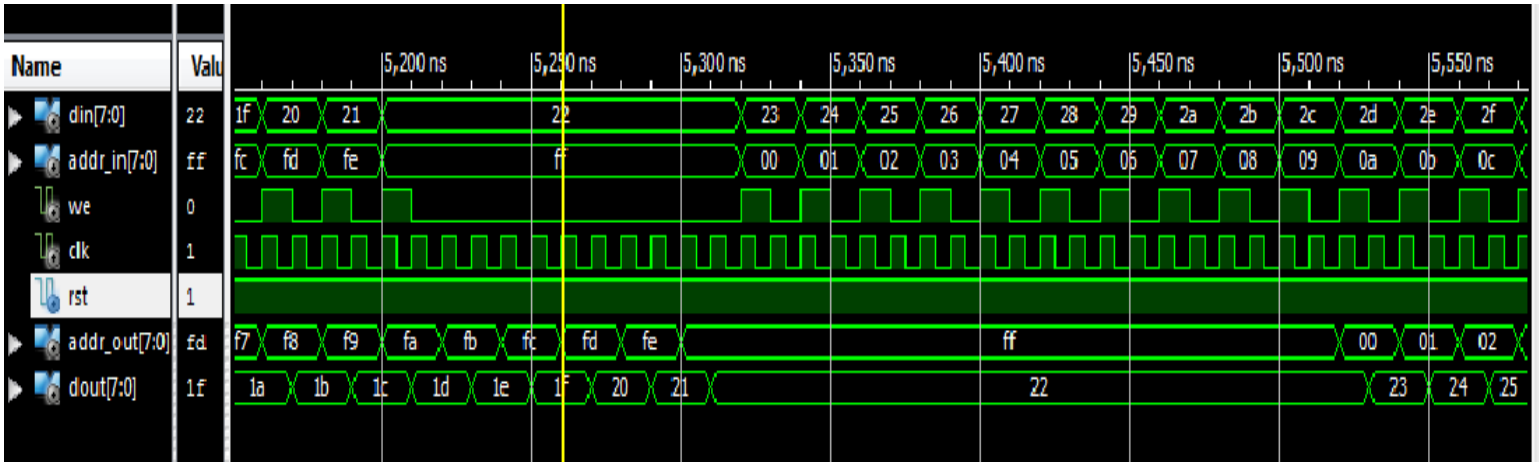
```
    ADDR_IN <=  std_logic_vector(to_unsigned(i, 8));
    WE <= '1';
    wait for 10 ns;
    WE <= '0';
    wait for 10 ns;
  end loop;  -- i
end process;

process is
begin  -- process lectura de donde se leen los datos ya escritos
  wait for 200 ns;
  for o in 0 to 255 loop
    ADDR_OUT <= std_logic_vector(to_unsigned(o, 8));
    wait for 20 ns;
  end loop;  -- o
end process;



end architecture dpram_mem;
```

Simulacion Funcional

En el diagrama funcional del circuito se busca implementar todos los posibles casos que hay para así verificar el correcto funcionamiento de circuito y así poder testearlo con el código de pruebas y verificar que se cumplen los requisitos.



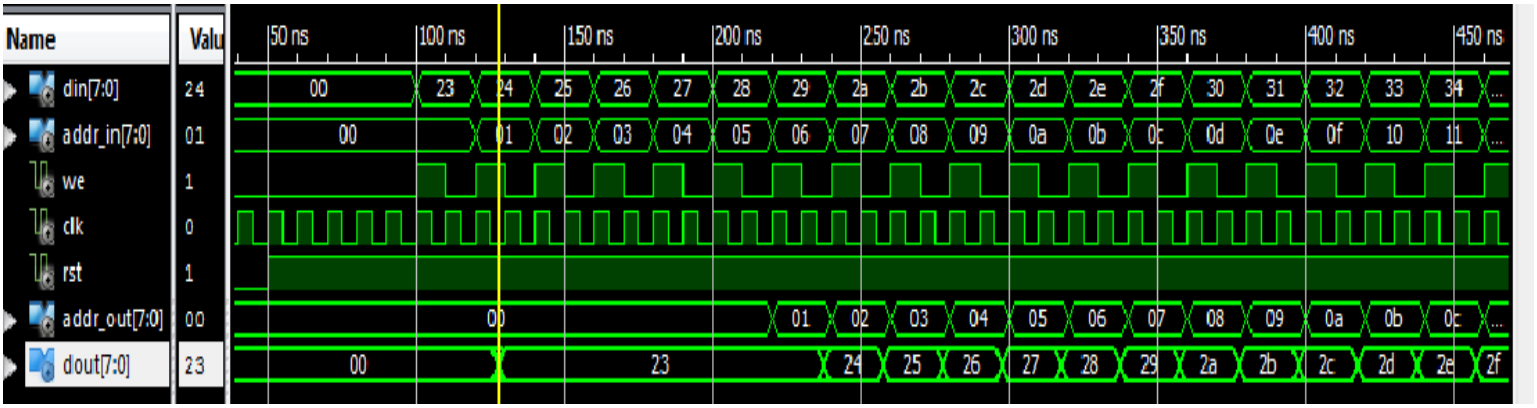
Por el gran tamaño de los casos a comprobar solo se muestra una captura genérica con todo en conjunto, En el siguiente diagrama funcional se muestra como cambia al superar el limite de las 255 posiciones.



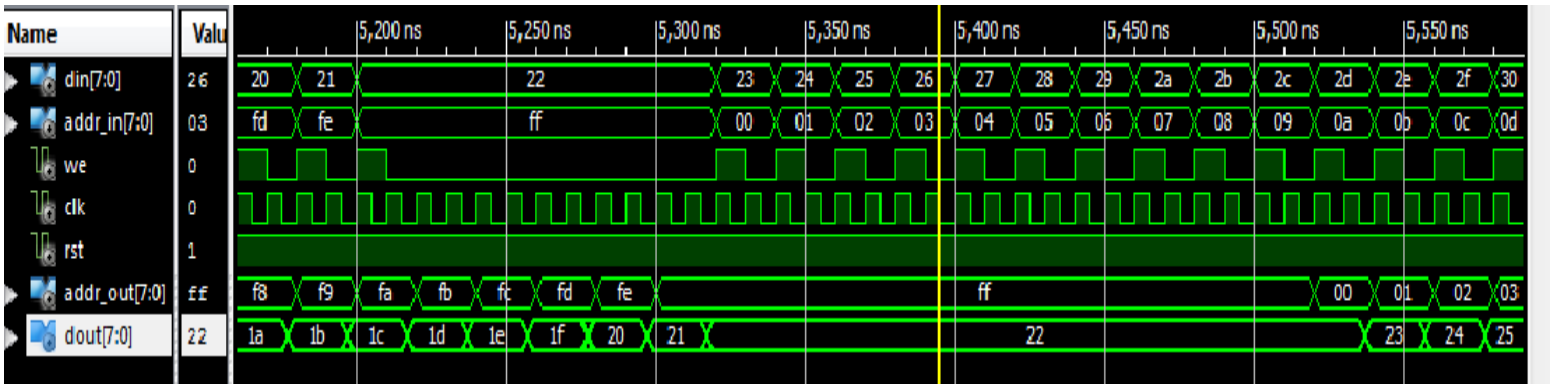
Una vez verificado este apartado podemos realizar el post-place&route para comprobar que el circuito que hemos diseñado cumple con las características en el apartado temporal.

Simulación Temporal

En el diagrama temporal del circuito se busca implementar todos los posibles casos que hay para así verificar el correcto funcionamiento de circuito y así poder testearlo con el código de pruebas y verificar que se cumplen los requisitos este esquema es mas propenso al fallo ya que se tienen en cuenta los retardos de las puerta lo cual puede hacer que el circuito que en el funcional iba perfectamente necesite cambios para poder llegar a funcionar bien.



Por el gran tamaño de los casos a comprobar solo se muestra una captura genérica con todo en conjunto, En el siguiente diagrama funcional se muestra como cambia al superar el limite de las 255 posiciones.



Una vez comprobado que todos los casos se resuelven correctamente podemos dar por concluido este apartado del trabajo.

Recursos utilizados

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	8	54,576	1,00%
Number used as Flip Flops	8		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	58	27,288	1,00%
Number used as logic	10	27,288	1,00%
Number using O6 output only	8		
Number using O5 output only	0		
Number using O5 and O6	2		
Number used as ROM	0		
Number used as Memory	48	6,408	1,00%
Number used as Dual Port RAM	48		
Number using O6 output only	48		
Number using O5 output only	0		
Number using O5 and O6	0		
Number used as Single Port RAM	0		
Number used as Shift Register	0		
Number of occupied Slices	16	6,822	1,00%
Number of MUXCYs used	0	13,644	0,00%
Number of LUT Flip Flop pairs used	58		
Number with an unused Flip Flop	50	58	86,00%
Number with an unused LUT	0	58	0,00%
Number of fully used LUT-FF pairs	8	58	13,00%
Number of unique control sets	5		
Number of slice register sites lost to control set restrictions	16	54,576	1,00%
Number of bonded IOBs	35	218	16,00%
Number of RAMB16BWERs	0	116	0,00%
Number of RAMB8BWERs	0	232	0,00%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0,00%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0,00%
Number of BUFG/BUFGMUXs	1	16	6,00%
Number used as BUFGs	1		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	8	0,00%
Number of ILOGIC2/ISERDES2s	0	376	0,00%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0,00%
Number of OLOGIC2/OSERDES2s	0	376	0,00%
Number of BSCANs	0	4	0,00%
Number of BUFHs	0	256	0,00%
Number of BUFPLLs	0	8	0,00%
Number of BUFPLL_MCBs	0	4	0,00%
Number of DSP48A1s	0	58	0,00%
Number of ICAPs	0	1	0,00%
Number of MCBs	0	2	0,00%
Number of PCILOGICSEs	0	2	0,00%
Number of PLL_ADVs	0	4	0,00%
Number of PMVs	0	1	0,00%
Number of STARTUPs	0	1	0,00%
Number of SUSPEND_SYNCs	0	1	0,00%
Average Fanout of Non-Clock Nets	5		