Martínez Coronel Brayan Yosafat

# Código VHDL

## Memoria de datos

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memoria is

generic (

m : integer := 11;

n : integer := 16

);

Port ( add : in STD\_LOGIC\_VECTOR (m-1 downto 0);

dataIn : in STD\_LOGIC\_VECTOR (n-1 downto 0);

clk, WD : in STD\_LOGIC;

dataOut : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end entity;

architecture Behavioral of memoria is

type arreglo is array (0 to (2\*\*m - 1)) of STD\_LOGIC\_VECTOR (n-1 downto 0);

signal mem : arreglo := (others=>(others=>'0'));

begin

process (clk)

begin

if (rising\_edge(clk)) then

if (WD = '1') then

mem(conv\_integer(add)) <= dataIn;

else

mem <= mem;

end if;

end if;

end process;

dataOut <= mem(conv\_integer(add));

end Behavioral;

## Memoria de programa

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memoria is

generic (

m : integer := 10;

n : integer := 25

);

Port ( pc : in STD\_LOGIC\_VECTOR (m-1 downto 0);

inst : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end entity;

architecture Behavioral of memoria is

type arreglo is array (0 to (2\*\*m - 1)) of STD\_LOGIC\_VECTOR (n-1 downto 0);

signal mem : arreglo := (others=>(others=>'0'));

constant code\_LI : std\_logic\_vector (4 downto 0) := "00001";

constant code\_ADD : std\_logic\_vector (4 downto 0) := "00000";

constant code\_SWI : std\_logic\_vector (4 downto 0) := "00011";

constant code\_ADDI : std\_logic\_vector (4 downto 0) := "00101";

constant code\_BNEI : std\_logic\_vector (4 downto 0) := "01110";

constant code\_NOP : std\_logic\_vector (4 downto 0) := "10110";

begin

mem(0) <= code\_LI & "0000" & "000000000000" & "0000";

mem(1) <= code\_LI & "0001" & "000000000000" & "0001";

mem(2) <= code\_LI & "0010" & "000000000000" & "0000";

mem(3) <= code\_LI & "0011" & "000000000000" & "1100";

mem(4) <= code\_ADD & "0100" & "0000" & "0001" & "00000000";

mem(5) <= code\_SWI & "0100" & "00000000" & "0111" & "0010";

mem(6) <= code\_ADDI & "0000" & "0001" & "000000000000";

mem(7) <= code\_ADDI & "0001" & "0100" & "000000000000";

mem(8) <= code\_ADDI & "0010" & "0010" & "000000000001";

mem(9) <= code\_BNEI & "0011" & "0010" & "111111111011";

mem(10) <= code\_NOP & "00000000000000000000";

inst <= mem(conv\_integer(pc));

end Behavioral;

# Código del TEST-BENCH

## Memoria de datos

library IEEE;

library STD;

use STD.TEXTIO.ALL;

use IEEE.std\_logic\_TEXTIO.ALL;

use IEEE.STD\_LOGIC\_1164.ALL;

entity t\_memoria is

-- Port ( );

end t\_memoria;

architecture Behavioral of t\_memoria is

constant m : integer := 11;

constant n : integer := 16;

constant clk\_period : time := 10 ns;

component memoria is generic (

m : integer := m;

n : integer := n

);

Port ( add : in STD\_LOGIC\_VECTOR (m-1 downto 0);

dataIn : in STD\_LOGIC\_VECTOR (n-1 downto 0);

clk, WD : in STD\_LOGIC;

dataOut : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end component;

signal add : STD\_LOGIC\_VECTOR (m-1 downto 0);

signal dataIn : STD\_LOGIC\_VECTOR (n-1 downto 0);

signal clk, WD : STD\_LOGIC;

signal dataOut : STD\_LOGIC\_VECTOR (n-1 downto 0);

begin

test: memoria port map (

add => add,

dataIn => dataIn,

clk => clk,

WD => WD,

dataOut => dataOut

);

clk\_process : process

begin

CLK <= '0';

wait for clk\_period/2;

CLK <= '1';

wait for clk\_period/2;

end process;

stim\_proc: process

file ARCH\_SALIDA : TEXT;

variable LINEA\_SALIDA : line;

file ARCH\_ENTRADA : TEXT;

variable LINEA\_ENTRADA : line;

variable CADENA : string (1 to 4);

variable v\_add : STD\_LOGIC\_VECTOR (m-1 downto 0);

variable v\_dataIn : STD\_LOGIC\_VECTOR (n-1 downto 0);

variable v\_WD : STD\_LOGIC;

variable v\_dataOut : STD\_LOGIC\_VECTOR (n-1 downto 0);

begin

file\_open(ARCH\_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\entradas1.txt", READ\_MODE);

file\_open(ARCH\_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\salidas1.txt", WRITE\_MODE);

CADENA := " ADD";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " WD";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " DIN";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := "DOUT";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);-- escribe la linea en el archivo

WAIT FOR 100 NS;

FOR I IN 0 TO 11 LOOP

readline(ARCH\_ENTRADA, LINEA\_ENTRADA); -- lee una linea completa

Hread(LINEA\_ENTRADA, v\_add);

add <= v\_add;

Hread(LINEA\_ENTRADA, v\_dataIn);

dataIn <= v\_dataIn;

read(LINEA\_ENTRADA, v\_WD);

WD <= v\_WD;

WAIT UNTIL RISING\_EDGE(clk); --ESPERO AL FLANCO DE SUBIDA

v\_dataOut := dataOut;

Hwrite(LINEA\_SALIDA, v\_add, right, 5);

write(LINEA\_SALIDA, v\_WD, right, 5);

Hwrite(LINEA\_SALIDA, v\_dataIn, right, 5);

Hwrite(LINEA\_SALIDA, v\_dataOut, right, 5);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);-- escribe la linea en el archivo

end loop;

file\_close(ARCH\_ENTRADA); -- cierra el archivo

file\_close(ARCH\_SALIDA); -- cierra el archivo

wait;

end process;

end Behavioral;

## Memoria de programa

library IEEE;

library STD;

use STD.TEXTIO.ALL;

use IEEE.std\_logic\_TEXTIO.ALL;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

entity t\_memoria is

-- Port ( );

end t\_memoria;

architecture Behavioral of t\_memoria is

constant m : integer := 10;

constant n : integer := 25;

constant period : time := 20 ns;

component memoria is generic (

m : integer := m;

n : integer := n

);

Port (

pc : in STD\_LOGIC\_VECTOR (m-1 downto 0);

inst : out STD\_LOGIC\_VECTOR (n-1 downto 0)

);

end component;

signal pc : STD\_LOGIC\_VECTOR (m-1 downto 0) := (others => '0');

signal inst : STD\_LOGIC\_VECTOR (n-1 downto 0);

begin

test: memoria port map (

pc => pc,

inst => inst

);

stim\_proc: process

file ARCH\_SALIDA : TEXT;

variable LINEA\_SALIDA : line;

variable v\_inst : STD\_LOGIC\_VECTOR (n-1 downto 0);

file ARCH\_ENTRADA : TEXT;

variable LINEA\_ENTRADA : line;

variable CADENA : string (1 to 7);

variable v\_pc : STD\_LOGIC\_VECTOR (m-1 downto 0);

begin

file\_open(ARCH\_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\entradas2.txt", READ\_MODE);

file\_open(ARCH\_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica6\salidas2.txt", WRITE\_MODE);

CADENA := " PC";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " OPCODE";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " 19..16";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " 15..12";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " 11..8";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " 7..4";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

CADENA := " 3..0";

write(LINEA\_SALIDA, CADENA, right, CADENA'LENGTH + 1);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);-- escribe la linea en el archivo

WAIT FOR 100 NS;

FOR I IN 0 TO 9 LOOP

readline(ARCH\_ENTRADA, LINEA\_ENTRADA); -- lee una linea completa

Hread(LINEA\_ENTRADA, v\_pc);

pc <= v\_pc;

wait for 20 ns;

v\_inst := inst;

Hwrite(LINEA\_SALIDA, v\_pc, right, 9);

write(LINEA\_SALIDA, v\_inst(24 downto 20), right, 7);

write(LINEA\_SALIDA, v\_inst(19 downto 16), right, 7);

write(LINEA\_SALIDA, v\_inst(15 downto 12), right, 8);

write(LINEA\_SALIDA, v\_inst(11 downto 8), right, 9);

write(LINEA\_SALIDA, v\_inst(7 downto 4), right, 8);

write(LINEA\_SALIDA, v\_inst(3 downto 0), right, 8);

writeline(ARCH\_SALIDA, LINEA\_SALIDA);-- escribe la linea en el archivo

end loop;

file\_close(ARCH\_ENTRADA); -- cierra el archivo

file\_close(ARCH\_SALIDA); -- cierra el archivo

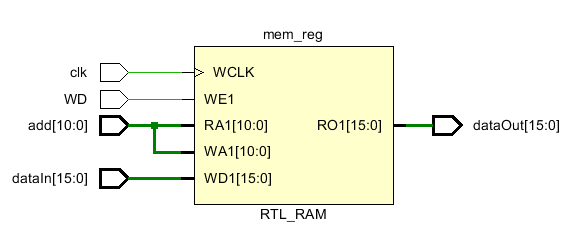
wait;

end process;

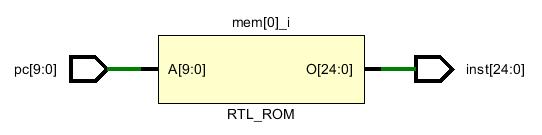
end Behavioral;

# Diagramas RTL

## Memoria de datos

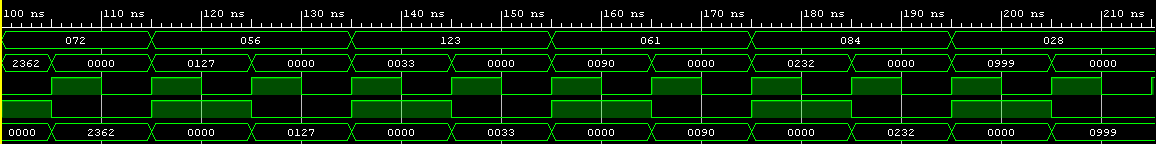


## Memoria de programa



# Forma de onda

## Memoria de datos



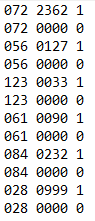
## Memoria de programa



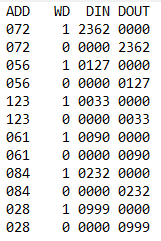


# Archivos

## Entrada – Memoria de Datos



## Salida – Memoria de datos



## Entrada – Memoria de programa



## Salida - Memoria de programa

