Martínez Coronel Brayan Yosafat

# Código VHDL

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memoria is

Port ( readReg1, readReg2, writeReg : in STD\_LOGIC\_VECTOR (3 downto 0);

shamt : in STD\_LOGIC\_VECTOR (3 downto 0);

writeData : in STD\_LOGIC\_VECTOR (15 downto 0);

clk, clear, dir, SHE, WR : in STD\_LOGIC;

readData1, readData2 : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end memoria;

architecture Behavioral of memoria is

type arreglo is array (0 to 15) of STD\_LOGIC\_VECTOR (15 downto 0);

signal mem : arreglo;

begin

process (clk, clear)

begin

if (clear = '1') then

for i in 0 to 15 loop

mem(i) <= "0000000000000000";

end loop;

elsif (rising\_edge(clk)) then

if (WR = '1') then

if (SHE = '1') then

if (dir = '1') then

mem(conv\_integer(writeReg)) <= to\_stdlogicvector(to\_bitvector(mem(conv\_integer(readReg1))) sll (conv\_integer(shamt)));

else

mem(conv\_integer(writeReg)) <= to\_stdlogicvector(to\_bitvector(mem(conv\_integer(readReg1))) srl (conv\_integer(shamt)));

end if;

else

mem(conv\_integer(writeReg)) <= writeData;

end if;

else

mem <= mem;

end if;

end if;

end process;

readData1 <= mem(conv\_integer(readReg1));

readData2 <= mem(conv\_integer(readReg2));

end Behavioral;

# Código del TEST-BENCH

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

component memoria is Port ( readReg1, readReg2, writeReg : in STD\_LOGIC\_VECTOR (3 downto 0);

shamt : in STD\_LOGIC\_VECTOR (3 downto 0);

writeData : in STD\_LOGIC\_VECTOR (15 downto 0);

clk, clear, dir, SHE, WR : in STD\_LOGIC;

readData1, readData2 : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end component;

signal readReg1, readReg2, writeReg : STD\_LOGIC\_VECTOR (3 downto 0);

signal shamt : STD\_LOGIC\_VECTOR (3 downto 0);

signal writeData : STD\_LOGIC\_VECTOR (15 downto 0);

signal clk, clear, dir, SHE, WR : STD\_LOGIC;

signal readData1, readData2 : STD\_LOGIC\_VECTOR(15 downto 0);

constant clk\_period : time := 10 ns;

begin

test: memoria port map (

readReg1 => readReg1,

readReg2 => readReg2,

writeReg => writeReg,

shamt => shamt,

writeData => writeData,

clk => clk,

clear => clear,

dir => dir,

SHE => SHE,

WR => WR,

readData1 => readData1,

readData2 => readData2

);

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;

variable var\_result\_1 : STD\_LOGIC\_VECTOR(15 DOWNTO 0);

variable var\_result\_2 : STD\_LOGIC\_VECTOR(15 DOWNTO 0);

file ARCH\_ENTRADA : TEXT;

variable LINEA\_ENTRADA : line;

variable v\_readReg1, v\_readReg2, v\_writeReg : STD\_LOGIC\_VECTOR (3 downto 0);

variable v\_shamt : STD\_LOGIC\_VECTOR (3 downto 0);

variable v\_writeData : STD\_LOGIC\_VECTOR (15 downto 0);

variable v\_clear, v\_dir, v\_SHE, v\_WR : STD\_LOGIC;

variable CADENA : STRING(1 TO 6);

begin

file\_open(ARCH\_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica5\entradas.txt", READ\_MODE);

file\_open(ARCH\_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica5\salidas.txt", WRITE\_MODE);

CADENA := " RR1";

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

CADENA := " RR2";

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

CADENA := " SHAMT";

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

CADENA := " WREG";

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

CADENA := " WD";

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

CADENA := " WR";

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

CADENA := " SHE";

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

CADENA := " DIR";

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

CADENA := " RD1";

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

CADENA := " RD2";

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

read(LINEA\_ENTRADA, v\_clear);

clear <= v\_clear;

read(LINEA\_ENTRADA, v\_WR);

WR <= v\_WR;

read(LINEA\_ENTRADA, v\_SHE);

SHE <= v\_SHE;

read(LINEA\_ENTRADA, v\_dir);

dir <= v\_dir;

Hread(LINEA\_ENTRADA, v\_writeReg);

writeReg <= v\_writeReg;

Hread(LINEA\_ENTRADA, v\_writeData);

writeData <= v\_writeData;

Hread(LINEA\_ENTRADA, v\_readReg1);

readReg1 <= v\_readReg1;

Hread(LINEA\_ENTRADA, v\_readReg2);

readReg2 <= v\_readReg2;

Hread(LINEA\_ENTRADA, v\_shamt);

shamt <= v\_shamt;

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

var\_result\_1 := readData1;

var\_result\_2 := readData2;

Hwrite(LINEA\_SALIDA, v\_readReg1, right, 6);

Hwrite(LINEA\_SALIDA, v\_readReg2, right, 7);

write(LINEA\_SALIDA, v\_shamt, right, 8);

Hwrite(LINEA\_SALIDA, v\_writeReg, right, 6);

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

write(LINEA\_SALIDA, v\_WR, right, 6);

write(LINEA\_SALIDA, v\_SHE, right, 6);

write(LINEA\_SALIDA, v\_dir, right, 7);

Hwrite(LINEA\_SALIDA, readData1, right, 8);

Hwrite(LINEA\_SALIDA, readData2, right, 7);

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

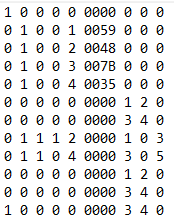
wait;

end process;

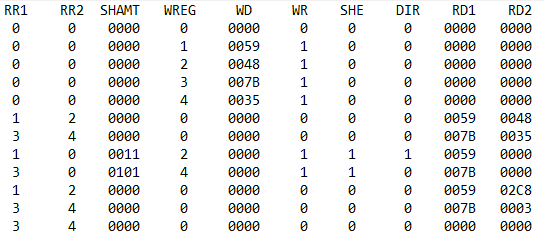
end Behavioral;

# Archivos

## Entrada



## Salida



# Diagramas RTL

