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 pila is

generic (

pc\_size : integer := 16;

n : integer := 3

);

Port ( PC\_in : in STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

clk, clr, WPC, UP, DW : in STD\_LOGIC;

PC\_out : out STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

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

);

end entity;

architecture Behavioral of pila is

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

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

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

begin

process (clk, clr, UP, DW, WPC)

variable v\_sp : integer := 0;

begin

if (clr = '1') then

v\_sp := 0;

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

elsif (rising\_edge(clk)) then

if (UP = '1') then

v\_sp := v\_sp + 1;

elsif (DW = '1') then

v\_sp := v\_sp - 1;

else

v\_sp := v\_sp;

end if;

if (WPC = '1') then

pointers(v\_sp) <= PC\_in;

else

pointers(v\_sp) <= pointers(v\_sp) + 1;

end if;

aux\_sp <= conv\_std\_logic\_vector(v\_sp, n);

end if;

end process;

PC\_out <= pointers(conv\_integer(aux\_sp));

SP <= aux\_sp;

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\_pila is

-- Port ( );

end entity;

architecture Behavioral of t\_pila is

constant pc\_size : integer := 16;

constant n : integer := 3;

constant clk\_period : time := 10 ns;

component pila is

generic (

pc\_size : integer := pc\_size;

n : integer := n

);

Port ( PC\_in : in STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

clk, clr, WPC, UP, DW : in STD\_LOGIC;

PC\_out : out STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

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

);

end component;

signal PC\_in : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

signal clk, clr, WPC, UP, DW : STD\_LOGIC;

signal PC\_out : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

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

begin

test: pila port map (

PC\_in => PC\_in,

clk => clk,

clr => clr,

WPC => WPC,

UP => UP,

DW => DW,

PC\_out => PC\_out,

SP => SP

);

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\_PC\_in : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

variable v\_clr, v\_WPC, v\_UP, v\_DW : STD\_LOGIC;

variable v\_PC\_out : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

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

begin

file\_open(ARCH\_ENTRADA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica7\entradas.txt", READ\_MODE);

file\_open(ARCH\_SALIDA, "C:\Users\yosaf\Desktop\Sexto\Arquitectura\Practica7\salidas.txt", WRITE\_MODE);

readline(ARCH\_ENTRADA, LINEA\_ENTRADA); -- salta la primera linea

CADENA := " SP";

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

CADENA := " PC";

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 27 LOOP

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

Hread(LINEA\_ENTRADA, v\_PC\_in);

PC\_in <= v\_PC\_in;

read(LINEA\_ENTRADA, v\_clr);

clr <= v\_clr;

read(LINEA\_ENTRADA, v\_WPC);

WPC <= v\_WPC;

read(LINEA\_ENTRADA, v\_UP);

UP <= v\_UP;

read(LINEA\_ENTRADA, v\_DW);

DW <= v\_DW;

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

v\_PC\_out := PC\_out;

v\_SP := SP;

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

Hwrite(LINEA\_SALIDA, v\_PC\_out, right, 5);

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

end loop;

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

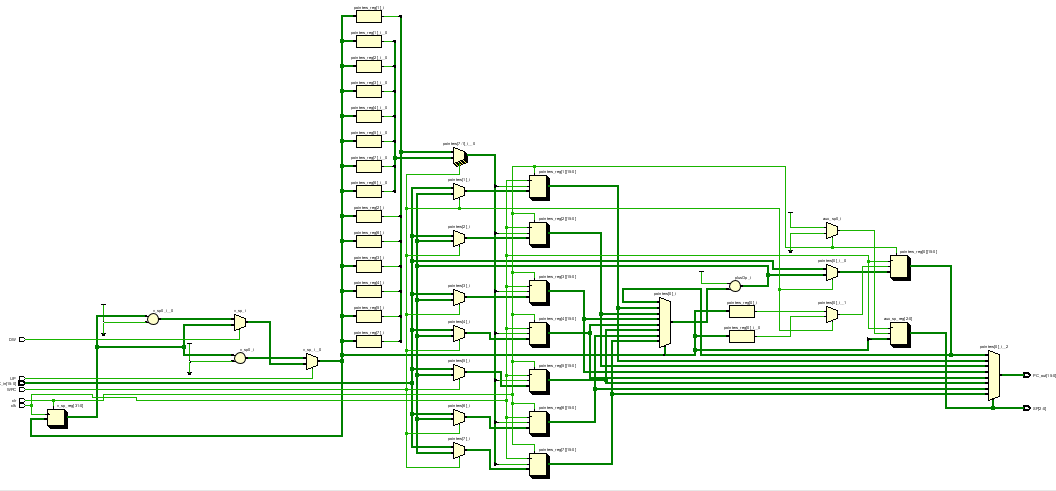
file\_close(ARCH\_SALIDA); -- cierra el archivo

wait;

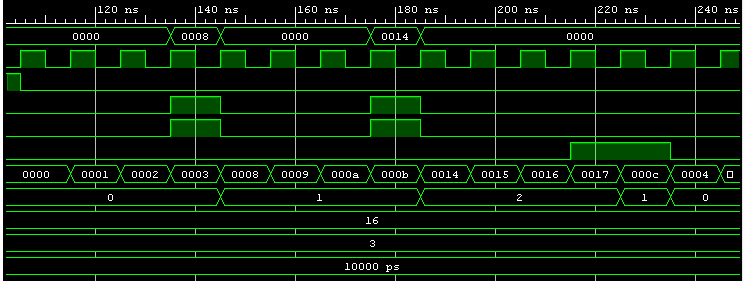
end process;

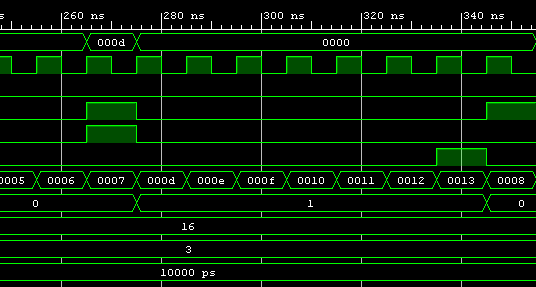
end Behavioral;

# Diagrama RTL



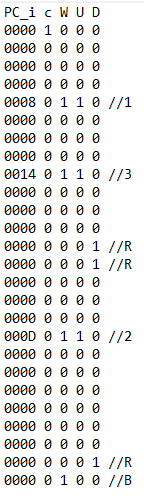
# Forma de onda





# Archivos

## Entrada



## Salida

