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;
```

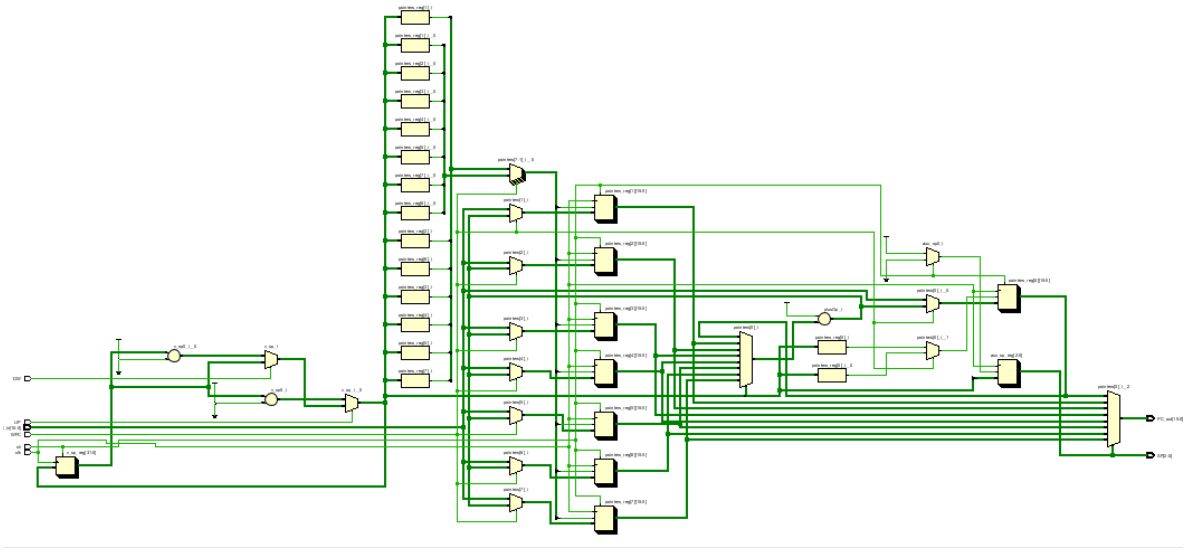
CÓDIGO DEL TEST-BENCH

```
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);
```

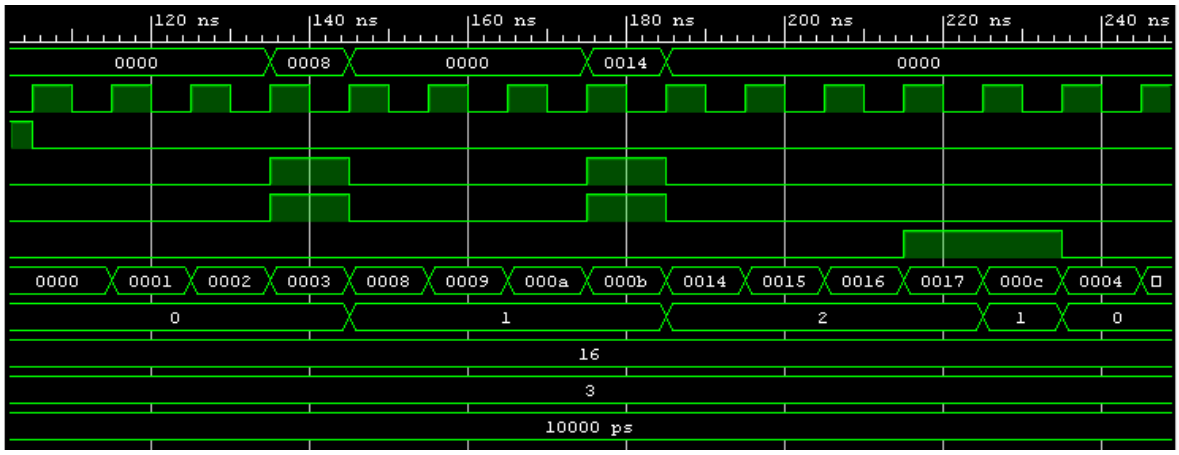
```
signal SP : STD_LOGIC_VECTOR (n-1 downto 0);
begin
    test: pila port map (
        PC_in => PC_in,
        clk => clk,
        clr => clr,
        WPC => WPC,
        UP => UP,
        DW => DW,
        PC_out => PC_out,
        SP => SP
    );

   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";
```

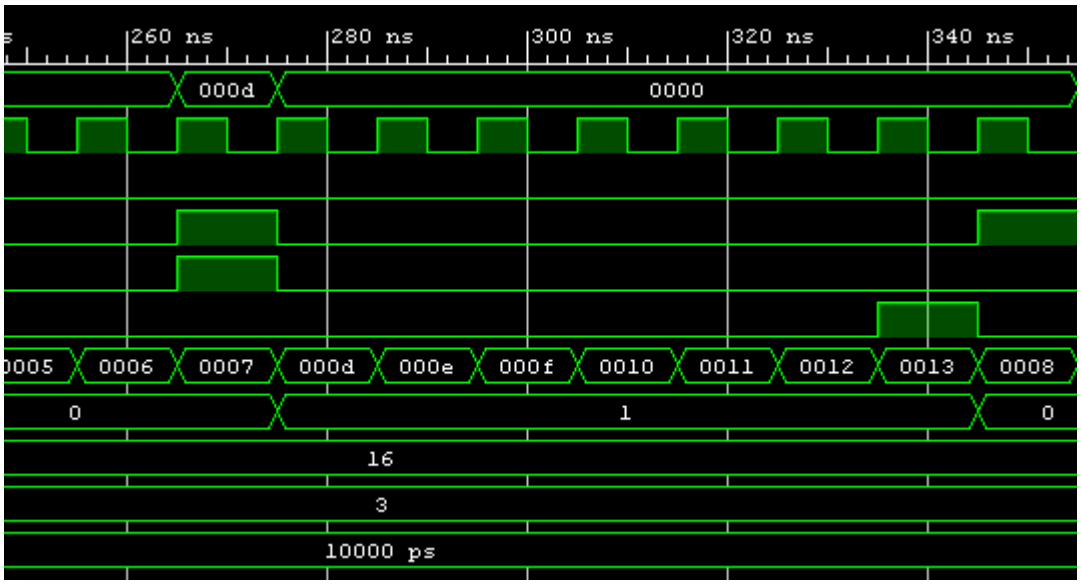
```
        write(LINEA_SALIDA, CADENA, right, CADENA'LENGTH + 1);

        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

```
PC_i c W U D
0000 1 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0008 0 1 1 0 //1
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0014 0 1 1 0 //3
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 1 //R
0000 0 0 0 1 //R
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
000D 0 1 1 0 //2
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 1 //R
0000 0 1 0 0 //B
```

SALIDA

```
SP   PC
 0 0000
 0 0000
 0 0001
 0 0002
 0 0003
 1 0008
 1 0009
 1 000A
 1 000B
 2 0014
 2 0015
 2 0016
 2 0017
 1 000C
 0 0004
 0 0005
 0 0006
 0 0007
 1 000D
 1 000E
 1 000F
 1 0010
 1 0011
 1 0012
 1 0013
 0 0008
 0 0000
 0 0001
```