INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Arquitectura de Computadoras

“Práctica 8: Memorias del procesador”

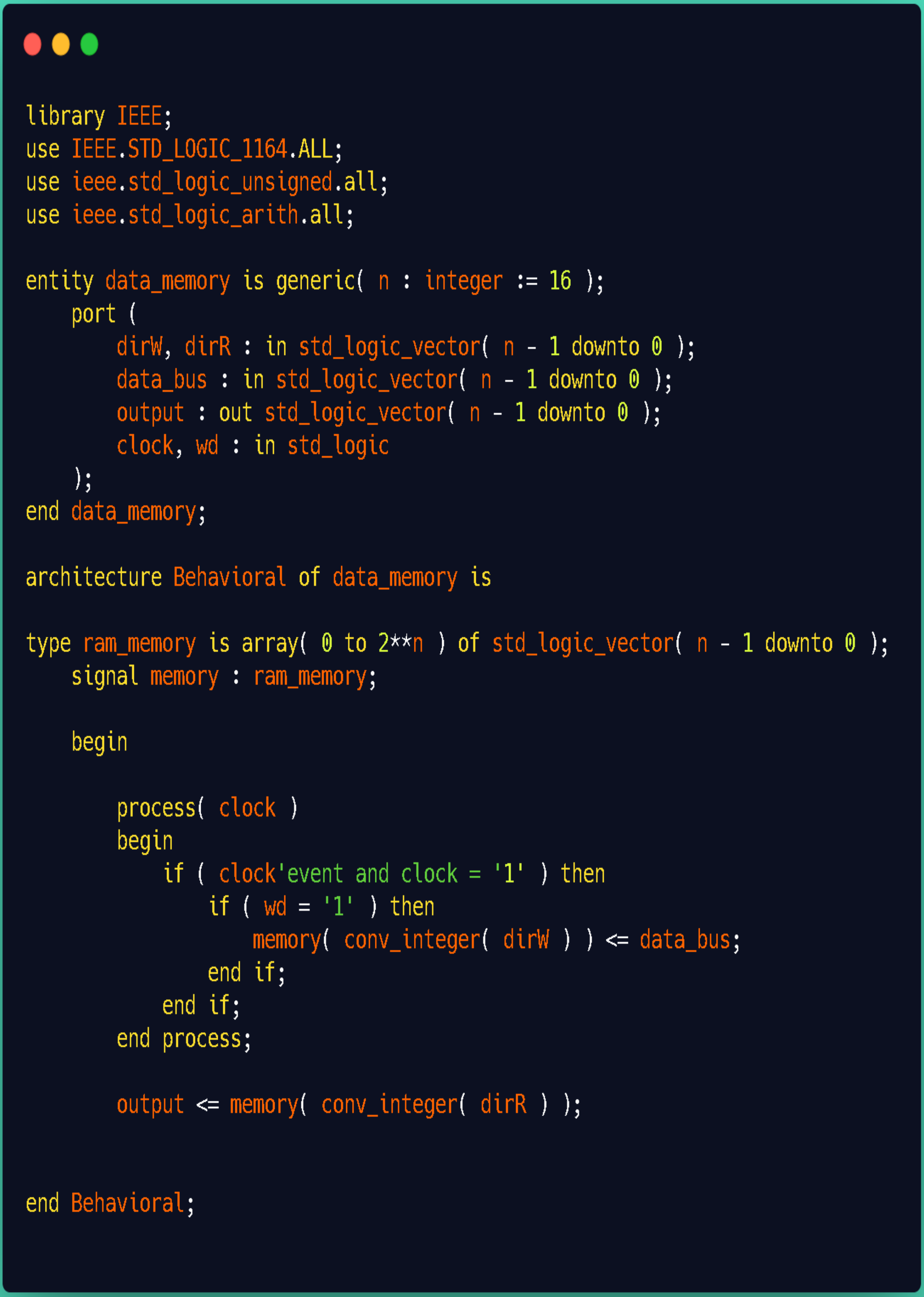
Grupo: 3CV11

Profesor: Vega García Nayeli

Sánchez Becerra Ernesto Daniel

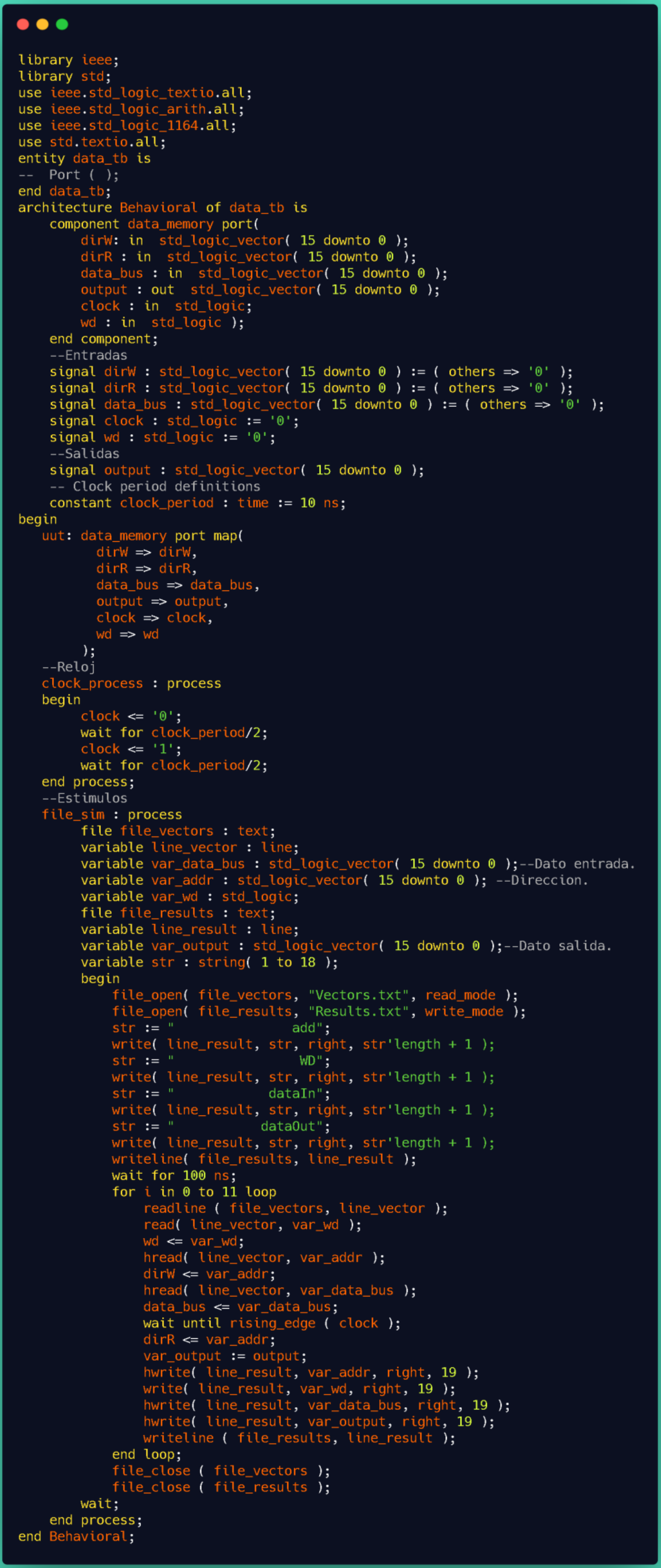
Memoria de Datos

➢ **Código de implementación**



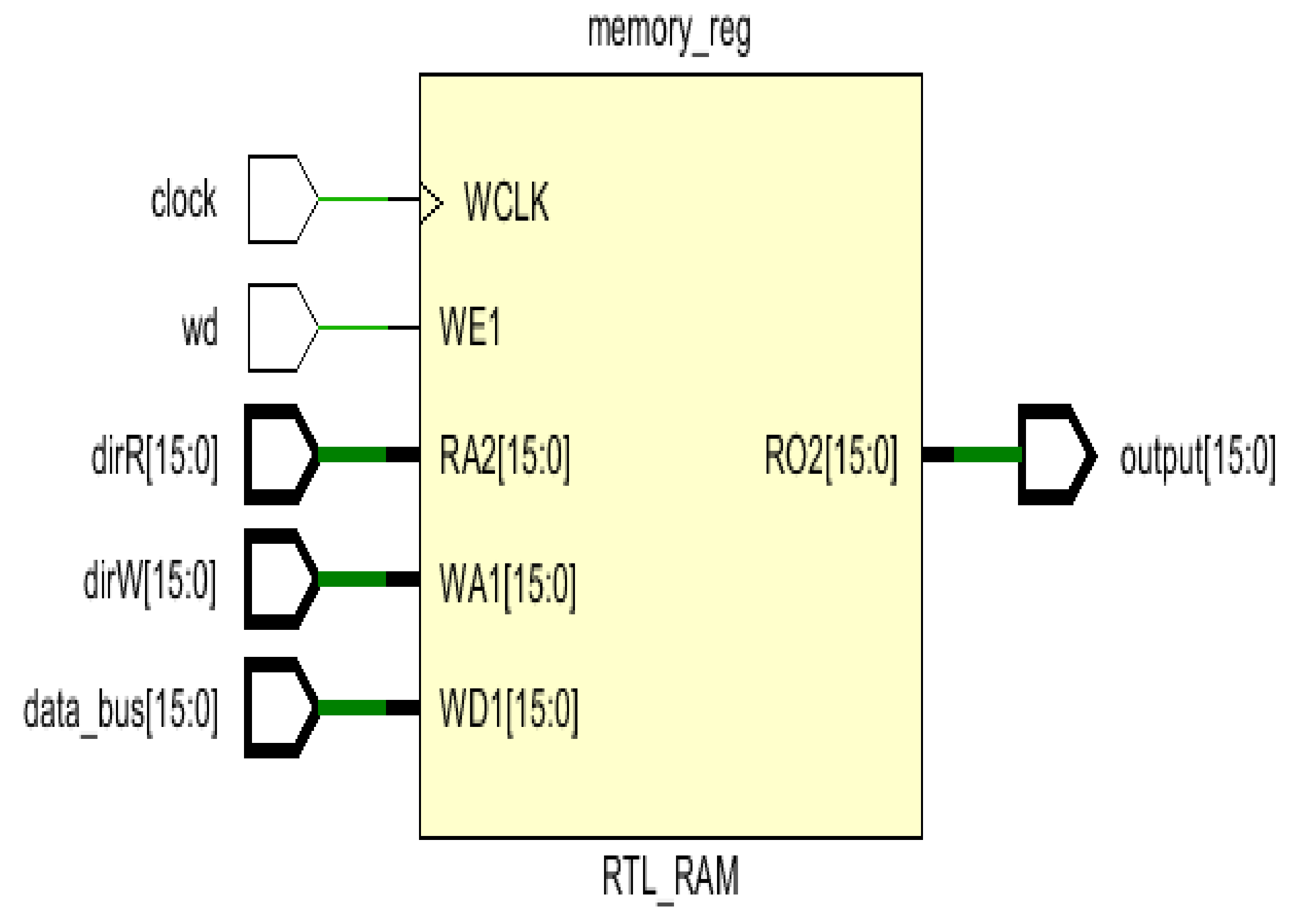
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity data_memory is generic( n : integer := 16 );
    port (
        dirW, dirR : in std_logic_vector( n - 1 downto 0 );
        data_bus : in std_logic_vector( n - 1 downto 0 );
        output : out std_logic_vector( n - 1 downto 0 );
        clock, wd : in std_logic
    );
end data_memory;

architecture Behavioral of data_memory is

type ram_memory is array( 0 to 2**n ) of std_logic_vector( n - 1 downto 0 );
    signal memory : ram_memory;

    begin

        process( clock )
        begin
            if ( clock'event and clock = '1' ) then
                if ( wd = '1' ) then
                    memory( conv_integer( dirW ) ) <= data_bus;
                end if;
            end if;
        end process;

        output <= memory( conv_integer( dirR ) );


end Behavioral;
```

➢ **Código de simulación**

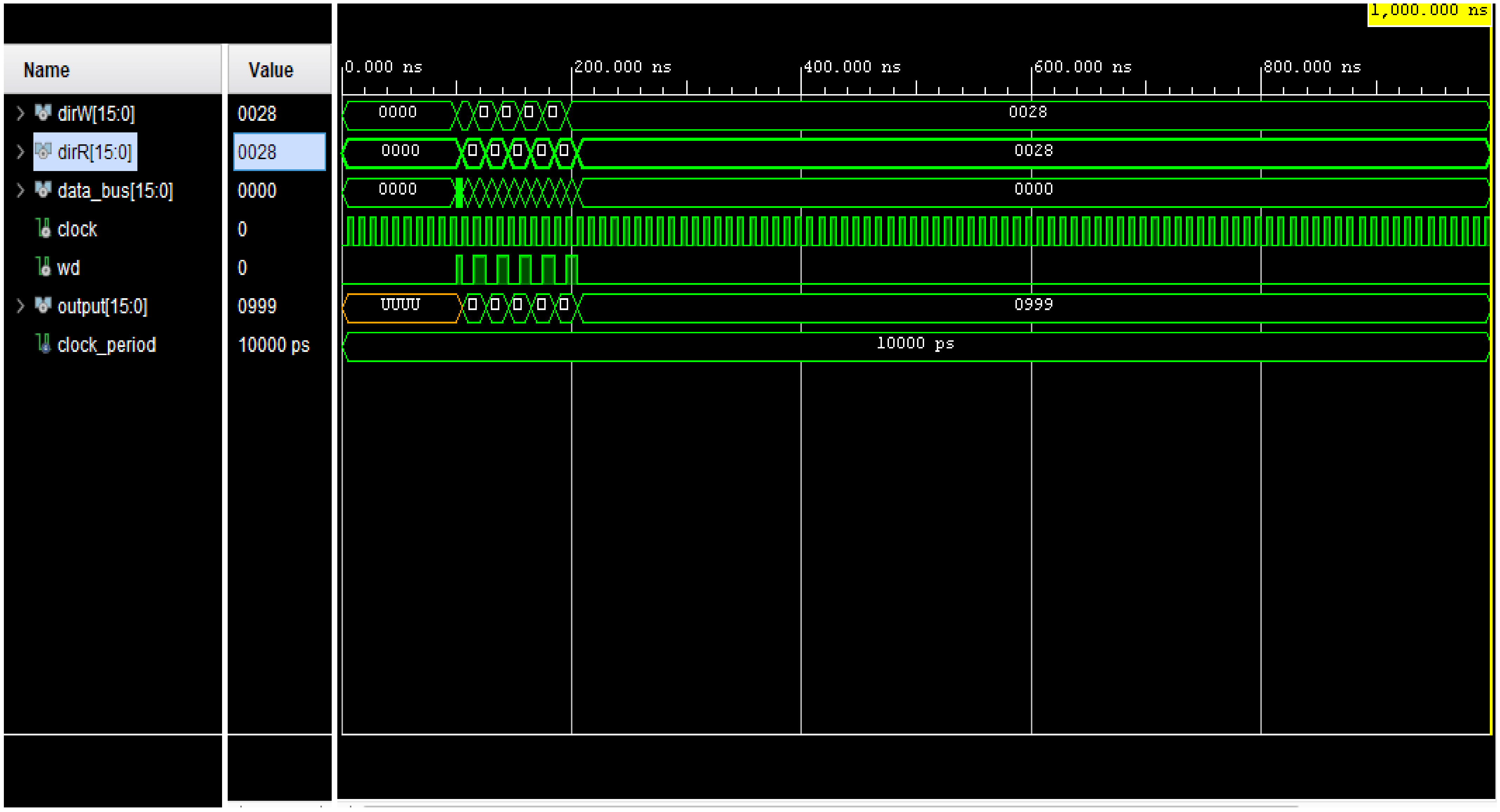


```vhdl
library ieee;
library std;
use ieee.std_logic_textio.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_1164.all;
use std.textio.all;
entity data_tb is
--  Port ( );
end data_tb;
architecture Behavioral of data_tb is
    component data_memory port(
        dirW: in  std_logic_vector( 15 downto 0 );
        dirR : in  std_logic_vector( 15 downto 0 );
        data_bus : in  std_logic_vector( 15 downto 0 );
        output : out  std_logic_vector( 15 downto 0 );
        clock : in  std_logic;
        wd : in  std_logic );
    end component;
    --Entradas
    signal dirW : std_logic_vector( 15 downto 0 ) := ( others => '0' );
    signal dirR : std_logic_vector( 15 downto 0 ) := ( others => '0' );
    signal data_bus : std_logic_vector( 15 downto 0 ) := ( others => '0' );
    signal clock : std_logic := '0';
    signal wd : std_logic := '0';
    --Salidas
    signal output : std_logic_vector( 15 downto 0 );
    -- Clock period definitions
    constant clock_period : time := 10 ns;
begin
   uut: data_memory port map(
          dirW => dirW,
          dirR => dirR,
          data_bus => data_bus,
          output => output,
          clock => clock,
          wd => wd
        );
   --Reloj
   clock_process : process
   begin
        clock <= '0';
        wait for clock_period/2;
        clock <= '1';
        wait for clock_period/2;
   end process;
   --Estimulos
   file_sim : process
        file file_vectors : text;
        variable line_vector : line;
        variable var_data_bus : std_logic_vector( 15 downto 0 );--Dato entrada.
        variable var_addr : std_logic_vector( 15 downto 0 ); --Direccion.
        variable var_wd : std_logic;
        file file_results : text;
        variable line_result : line;
        variable var_output : std_logic_vector( 15 downto 0 );--Dato salida.
        variable str : string( 1 to 18 );
        begin
            file_open( file_vectors, "Vectors.txt", read_mode );
            file_open( file_results, "Results.txt", write_mode );
            str := "               add";
            write( line_result, str, right, str'length + 1 );
            str := "                WD";
            write( line_result, str, right, str'length + 1 );
            str := "            dataIn";
            write( line_result, str, right, str'length + 1 );
            str := "           dataOut";
            write( line_result, str, right, str'length + 1 );
            writeline( file_results, line_result );
            wait for 100 ns;
            for i in 0 to 11 loop
                readline ( file_vectors, line_vector );
                read( line_vector, var_wd );
                wd <= var_wd;
                hread( line_vector, var_addr );
                dirW <= var_addr;
                hread( line_vector, var_data_bus );
                data_bus <= var_data_bus;
                wait until rising_edge ( clock );
                dirR <= var_addr;
                var_output := output;
                hwrite( line_result, var_addr, right, 19 );
                write( line_result, var_wd, right, 19 );
                hwrite( line_result, var_data_bus, right, 19 );
                hwrite( line_result, var_output, right, 19 );
                writeline ( file_results, line_result );
            end loop;
            file_close ( file_vectors );
            file_close ( file_results );
        wait;
    end process;
end Behavioral;
```

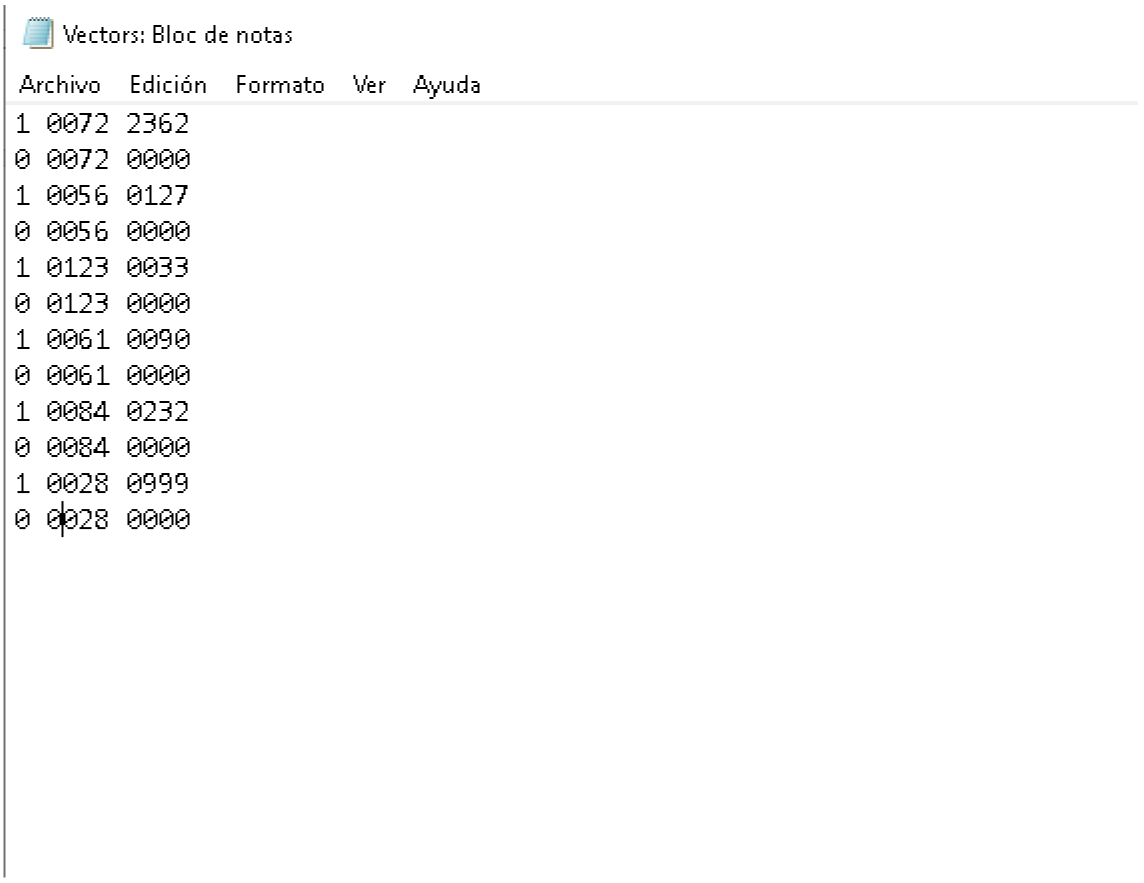
➢ **Diagrama RTL**



- **Forma de onda**

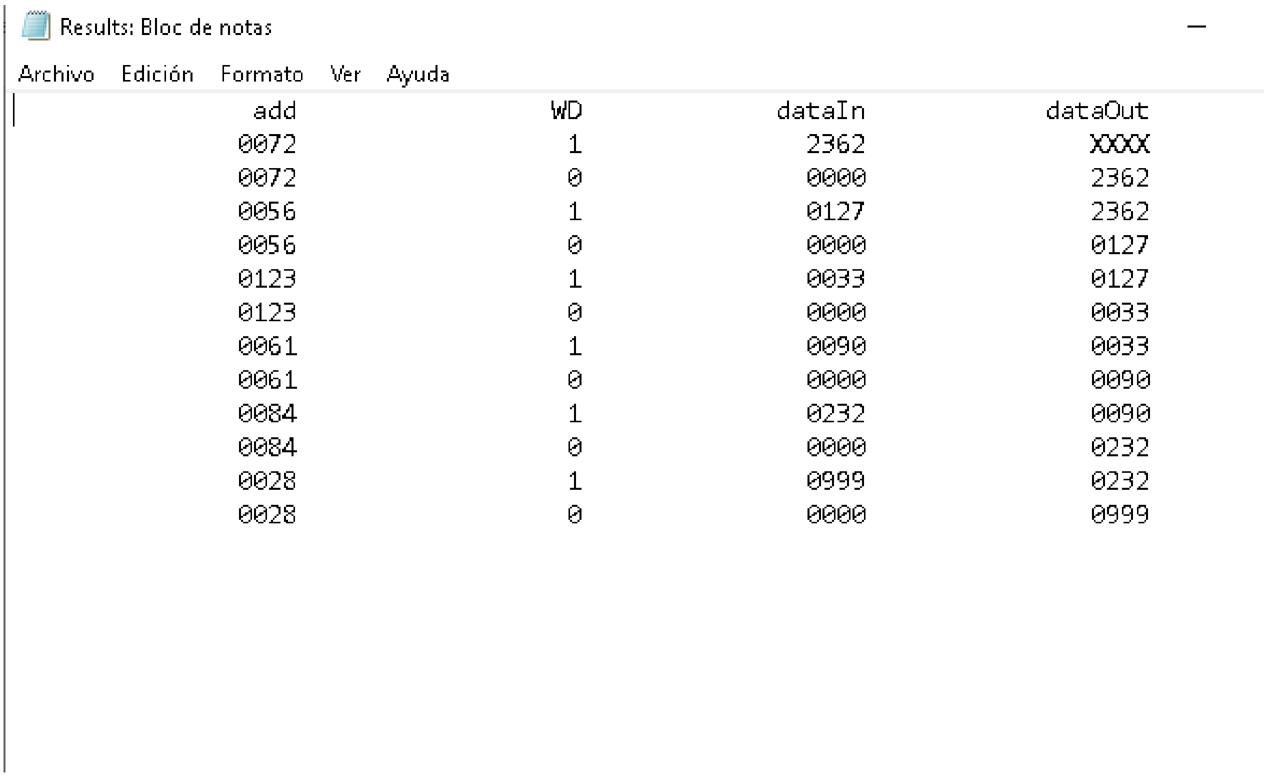


➢ Archivo de estímulos



```
1 0072 2362
0 0072 0000
1 0056 0127
0 0056 0000
1 0123 0033
0 0123 0000
1 0061 0090
0 0061 0000
1 0084 0232
0 0084 0000
1 0028 0999
0 0028 0000
```

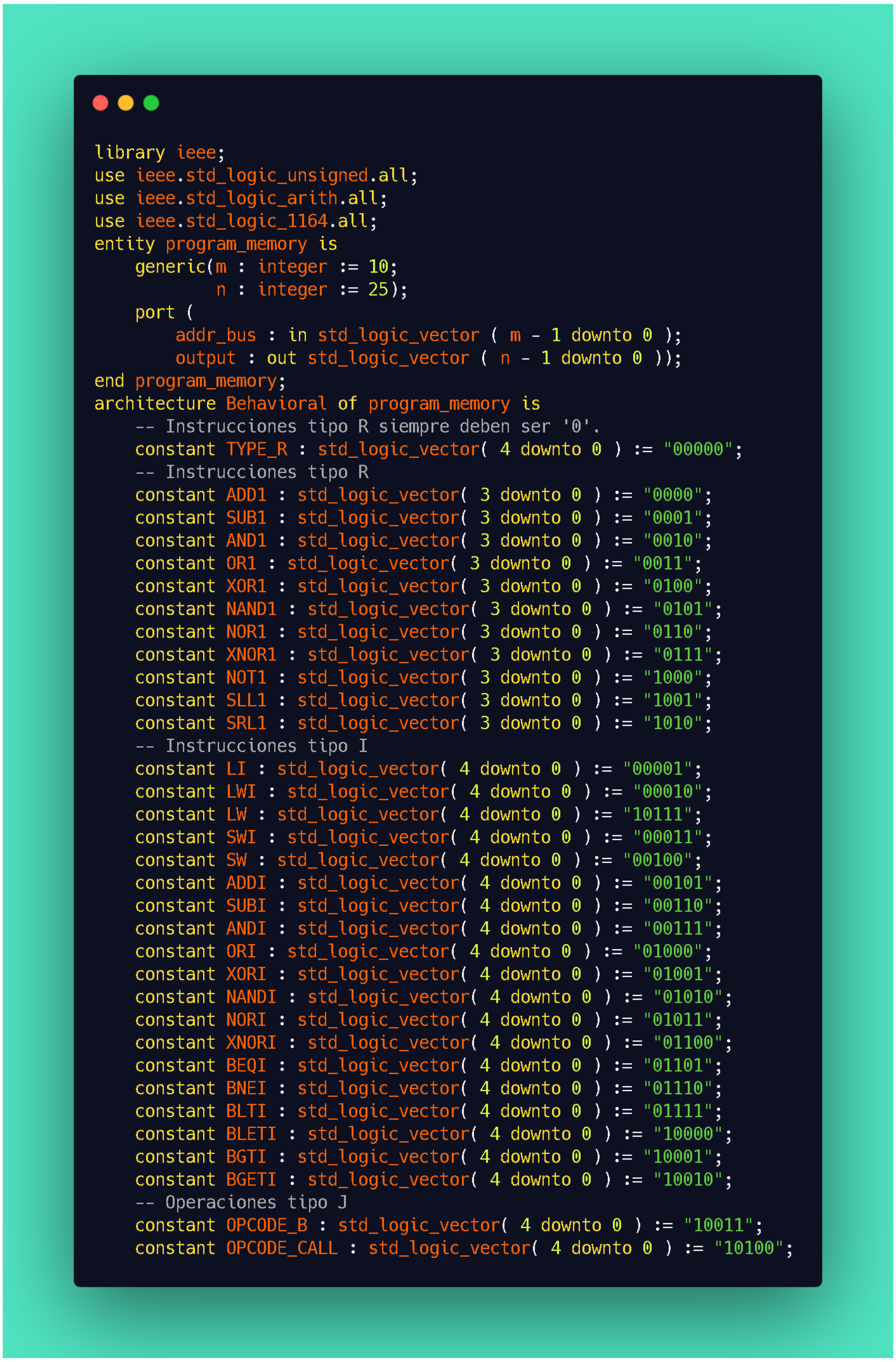
➢ Archivo de resultados



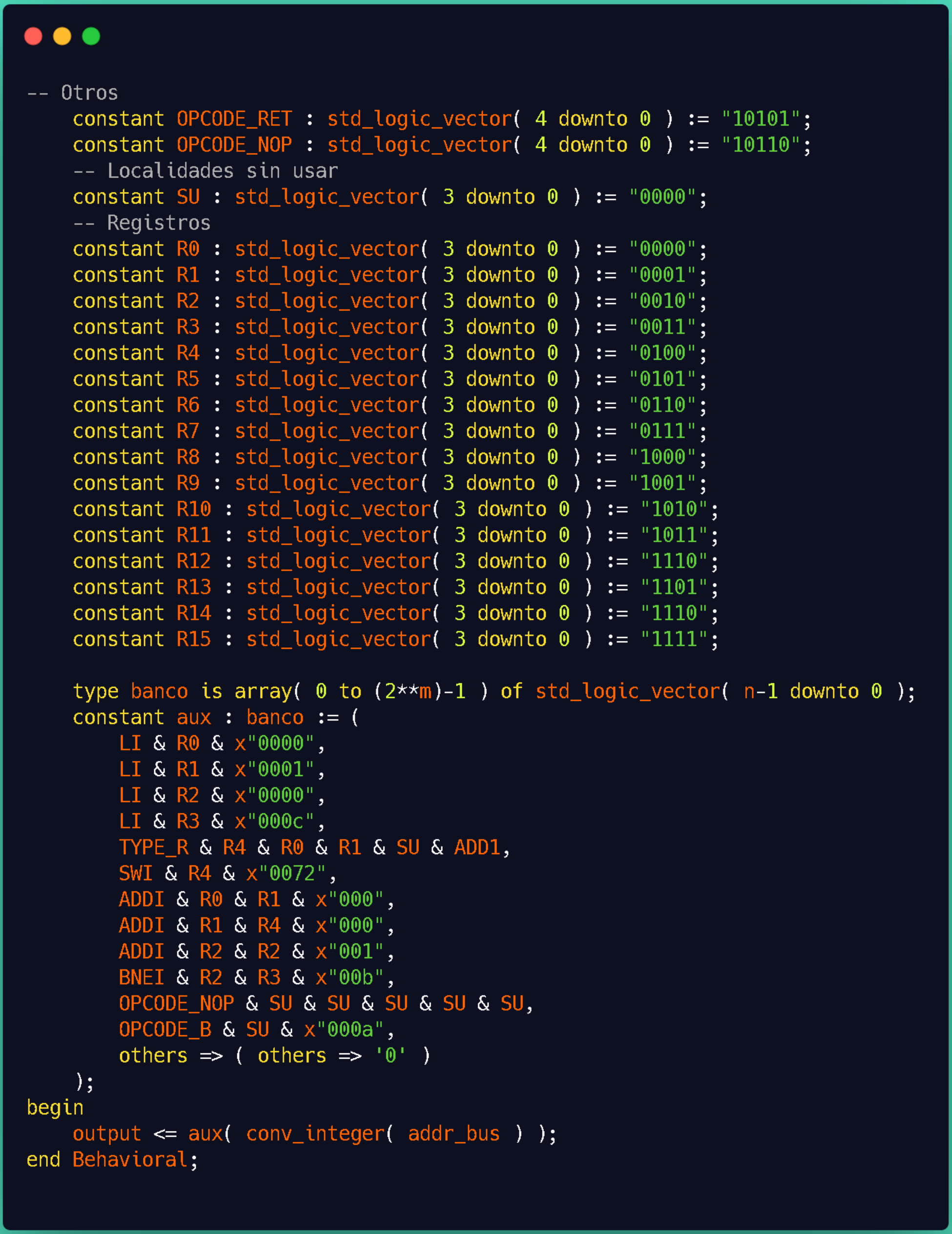
```
             add             WD           dataIn          dataOut
            0072              1             2362             XXXX
            0072              0             0000             2362
            0056              1             0127             2362
            0056              0             0000             0127
            0123              1             0033             0127
            0123              0             0000             0033
            0061              1             0090             0033
            0061              0             0000             0090
            0084              1             0232             0090
            0084              0             0000             0232
            0028              1             0999             0232
            0028              0             0000             0999
```

Memoria de Programa

➢ Código de implementación

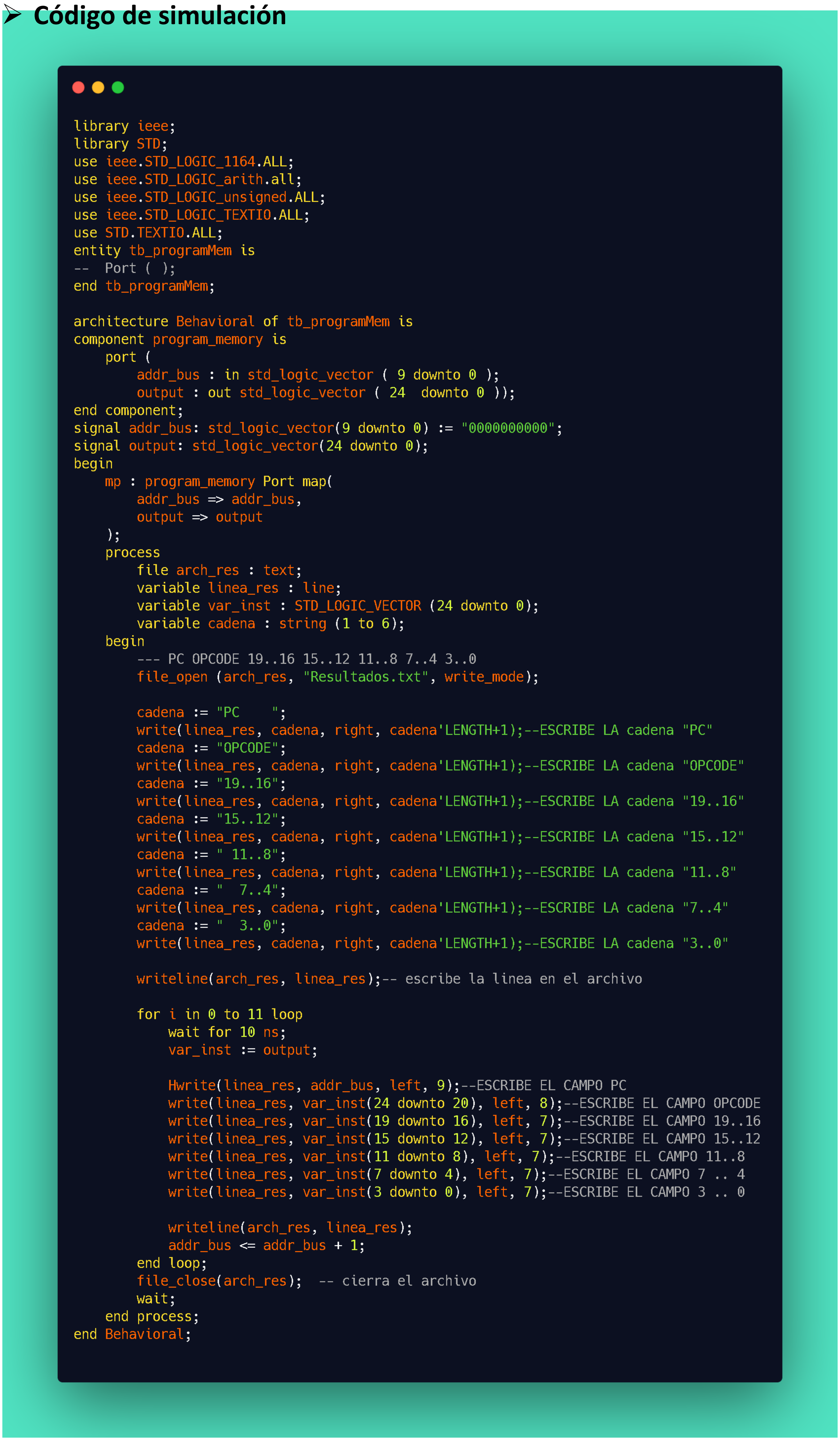


```vhdl
library ieee;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_1164.all;
entity program_memory is
    generic(m : integer := 10;
            n : integer := 25);
    port (
        addr_bus : in std_logic_vector ( m - 1 downto 0 );
        output : out std_logic_vector ( n - 1 downto 0 ));
end program_memory;
architecture Behavioral of program_memory is
    -- Instrucciones tipo R siempre deben ser '0'.
    constant TYPE_R : std_logic_vector( 4 downto 0 ) := "00000";
    -- Instrucciones tipo R
    constant ADD1 : std_logic_vector( 3 downto 0 ) := "0000";
    constant SUB1 : std_logic_vector( 3 downto 0 ) := "0001";
    constant AND1 : std_logic_vector( 3 downto 0 ) := "0010";
    constant OR1 : std_logic_vector( 3 downto 0 ) := "0011";
    constant XOR1 : std_logic_vector( 3 downto 0 ) := "0100";
    constant NAND1 : std_logic_vector( 3 downto 0 ) := "0101";
    constant NOR1 : std_logic_vector( 3 downto 0 ) := "0110";
    constant XNOR1 : std_logic_vector( 3 downto 0 ) := "0111";
    constant NOT1 : std_logic_vector( 3 downto 0 ) := "1000";
    constant SLL1 : std_logic_vector( 3 downto 0 ) := "1001";
    constant SRL1 : std_logic_vector( 3 downto 0 ) := "1010";
    -- Instrucciones tipo I
    constant LI : std_logic_vector( 4 downto 0 ) := "00001";
    constant LWI : std_logic_vector( 4 downto 0 ) := "00010";
    constant LW : std_logic_vector( 4 downto 0 ) := "10111";
    constant SWI : std_logic_vector( 4 downto 0 ) := "00011";
    constant SW : std_logic_vector( 4 downto 0 ) := "00100";
    constant ADDI : std_logic_vector( 4 downto 0 ) := "00101";
    constant SUBI : std_logic_vector( 4 downto 0 ) := "00110";
    constant ANDI : std_logic_vector( 4 downto 0 ) := "00111";
    constant ORI : std_logic_vector( 4 downto 0 ) := "01000";
    constant XORI : std_logic_vector( 4 downto 0 ) := "01001";
    constant NANDI : std_logic_vector( 4 downto 0 ) := "01010";
    constant NORI : std_logic_vector( 4 downto 0 ) := "01011";
    constant XNORI : std_logic_vector( 4 downto 0 ) := "01100";
    constant BEQI : std_logic_vector( 4 downto 0 ) := "01101";
    constant BNEI : std_logic_vector( 4 downto 0 ) := "01110";
    constant BLTI : std_logic_vector( 4 downto 0 ) := "01111";
    constant BLETI : std_logic_vector( 4 downto 0 ) := "10000";
    constant BGTI : std_logic_vector( 4 downto 0 ) := "10001";
    constant BGETI : std_logic_vector( 4 downto 0 ) := "10010";
    -- Operaciones tipo J
    constant OPCODE_B : std_logic_vector( 4 downto 0 ) := "10011";
    constant OPCODE_CALL : std_logic_vector( 4 downto 0 ) := "10100";
```



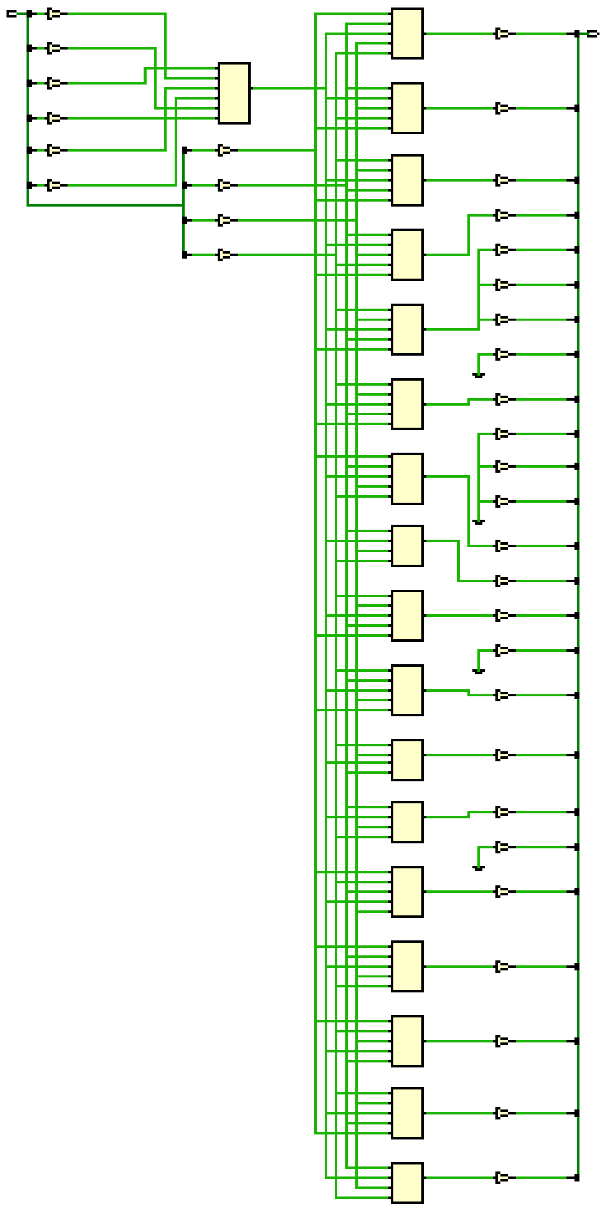
```
-- Otros
    constant OPCODE_RET : std_logic_vector( 4 downto 0 ) := "10101";
    constant OPCODE_NOP : std_logic_vector( 4 downto 0 ) := "10110";
    -- Localidades sin usar
    constant SU : std_logic_vector( 3 downto 0 ) := "0000";
    -- Registros
    constant R0 : std_logic_vector( 3 downto 0 ) := "0000";
    constant R1 : std_logic_vector( 3 downto 0 ) := "0001";
    constant R2 : std_logic_vector( 3 downto 0 ) := "0010";
    constant R3 : std_logic_vector( 3 downto 0 ) := "0011";
    constant R4 : std_logic_vector( 3 downto 0 ) := "0100";
    constant R5 : std_logic_vector( 3 downto 0 ) := "0101";
    constant R6 : std_logic_vector( 3 downto 0 ) := "0110";
    constant R7 : std_logic_vector( 3 downto 0 ) := "0111";
    constant R8 : std_logic_vector( 3 downto 0 ) := "1000";
    constant R9 : std_logic_vector( 3 downto 0 ) := "1001";
    constant R10 : std_logic_vector( 3 downto 0 ) := "1010";
    constant R11 : std_logic_vector( 3 downto 0 ) := "1011";
    constant R12 : std_logic_vector( 3 downto 0 ) := "1110";
    constant R13 : std_logic_vector( 3 downto 0 ) := "1101";
    constant R14 : std_logic_vector( 3 downto 0 ) := "1110";
    constant R15 : std_logic_vector( 3 downto 0 ) := "1111";

    type banco is array( 0 to (2**m)-1 ) of std_logic_vector( n-1 downto 0 );
    constant aux : banco := (
        LI & R0 & x"0000",
        LI & R1 & x"0001",
        LI & R2 & x"0000",
        LI & R3 & x"000c",
        TYPE_R & R4 & R0 & R1 & SU & ADD1,
        SWI & R4 & x"0072",
        ADDI & R0 & R1 & x"000",
        ADDI & R1 & R4 & x"000",
        ADDI & R2 & R2 & x"001",
        BNEI & R2 & R3 & x"00b",
        OPCODE_NOP & SU & SU & SU & SU & SU,
        OPCODE_B & SU & x"000a",
        others => ( others => '0' )
    );
begin
    output <= aux( conv_integer( addr_bus ) );
end Behavioral;
```

➢ Código de simulación

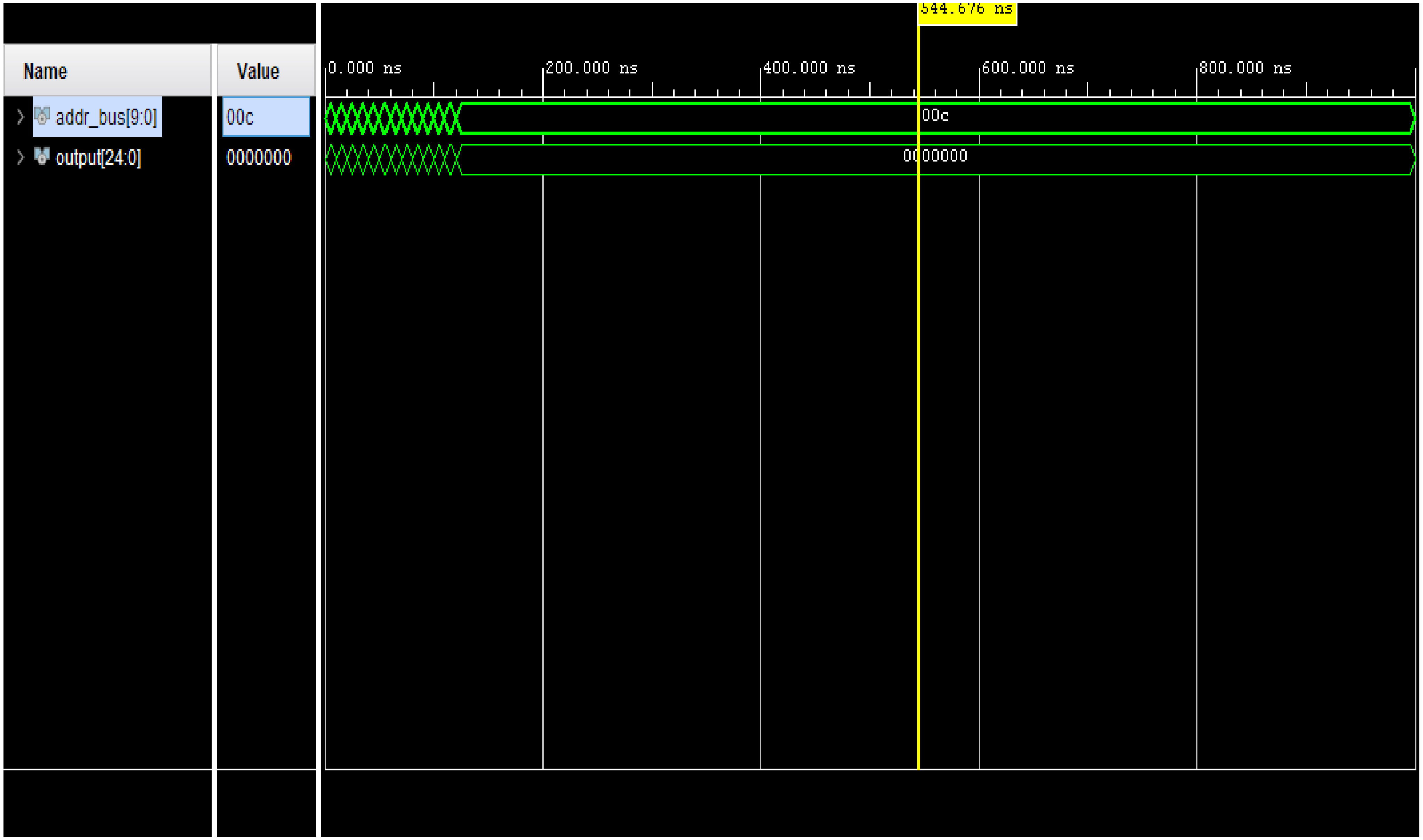


```vhdl
library ieee;
library STD;
use ieee.STD_LOGIC_1164.ALL;
use ieee.STD_LOGIC_arith.all;
use ieee.STD_LOGIC_unsigned.ALL;
use ieee.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;
entity tb_programMem is
--  Port ( );
end tb_programMem;

architecture Behavioral of tb_programMem is
component program_memory is
    port (
        addr_bus : in std_logic_vector ( 9 downto 0 );
        output : out std_logic_vector ( 24  downto 0 ));
end component;
signal addr_bus: std_logic_vector(9 downto 0) := "0000000000";
signal output: std_logic_vector(24 downto 0);
begin
    mp : program_memory Port map(
        addr_bus => addr_bus,
        output => output
    );
    process
        file arch_res : text;
        variable linea_res : line;
        variable var_inst : STD_LOGIC_VECTOR (24 downto 0);
        variable cadena : string (1 to 6);
    begin
        --- PC OPCODE 19..16 15..12 11..8 7..4 3..0
        file_open (arch_res, "Resultados.txt", write_mode);

        cadena := "PC    ";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "PC"
        cadena := "OPCODE";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "OPCODE"
        cadena := "19..16";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "19..16"
        cadena := "15..12";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "15..12"
        cadena := " 11..8";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "11..8"
        cadena := "  7..4";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "7..4"
        cadena := "  3..0";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena "3..0"

        writeline(arch_res, linea_res);-- escribe la linea en el archivo

        for i in 0 to 11 loop
            wait for 10 ns;
            var_inst := output;

            Hwrite(linea_res, addr_bus, left, 9);--ESCRIBE EL CAMPO PC
            write(linea_res, var_inst(24 downto 20), left, 8);--ESCRIBE EL CAMPO OPCODE
            write(linea_res, var_inst(19 downto 16), left, 7);--ESCRIBE EL CAMPO 19..16
            write(linea_res, var_inst(15 downto 12), left, 7);--ESCRIBE EL CAMPO 15..12
            write(linea_res, var_inst(11 downto 8), left, 7);--ESCRIBE EL CAMPO 11..8
            write(linea_res, var_inst(7 downto 4), left, 7);--ESCRIBE EL CAMPO 7 .. 4
            write(linea_res, var_inst(3 downto 0), left, 7);--ESCRIBE EL CAMPO 3 .. 0

            writeline(arch_res, linea_res);
            addr_bus <= addr_bus + 1;
        end loop;
        file_close(arch_res);  -- cierra el archivo
        wait;
    end process;
end Behavioral;
```

- **Diagrama RTL**

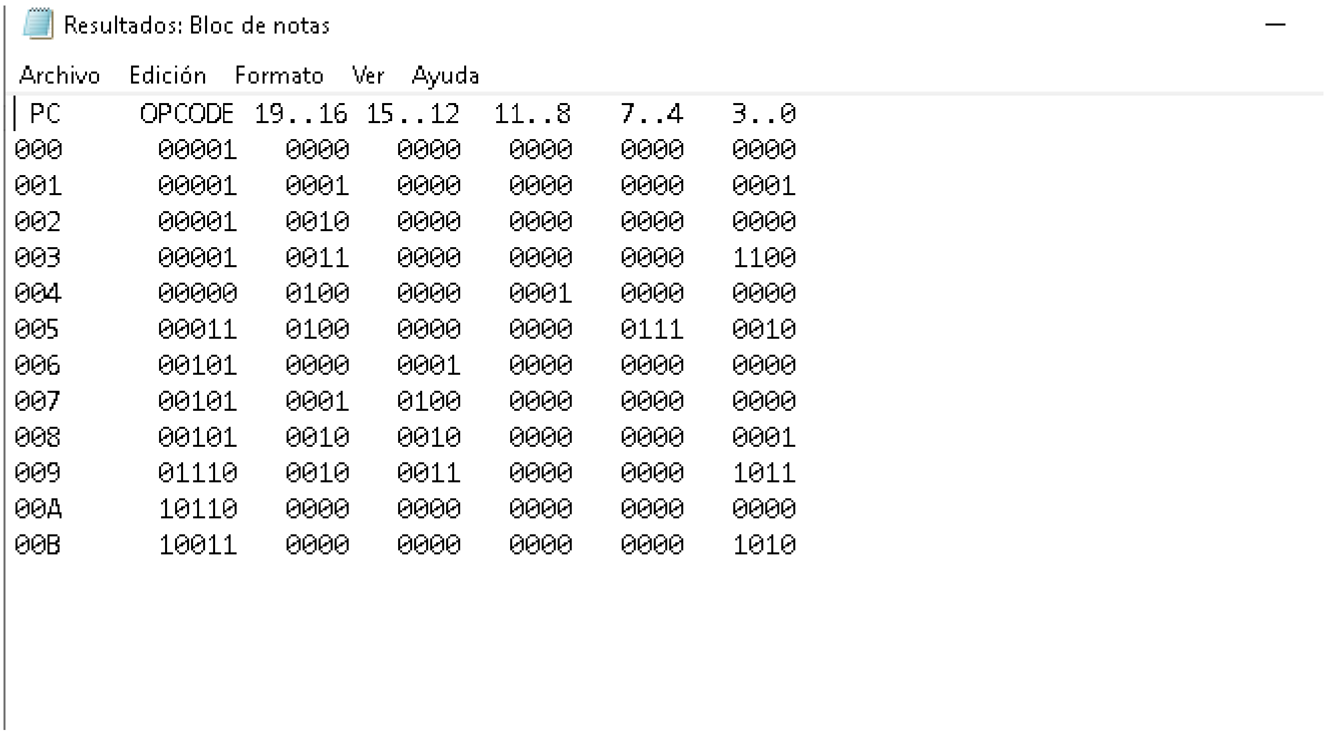


➢ Forma de onda



Name	Value
addr_bus[9:0]	00c
output[24:0]	0000000

➢ Archivo de resultados



```
PC      OPCODE 19..16 15..12  11..8   7..4   3..0
000      00001   0000   0000   0000   0000   0000
001      00001   0001   0000   0000   0000   0001
002      00001   0010   0000   0000   0000   0000
003      00001   0011   0000   0000   0000   1100
004      00000   0100   0000   0001   0000   0000
005      00011   0100   0000   0000   0111   0010
006      00101   0000   0001   0000   0000   0000
007      00101   0001   0100   0000   0000   0000
008      00101   0010   0010   0000   0000   0001
009      01110   0010   0011   0000   0000   1011
00A      10110   0000   0000   0000   0000   0000
00B      10011   0000   0000   0000   0000   1010
```