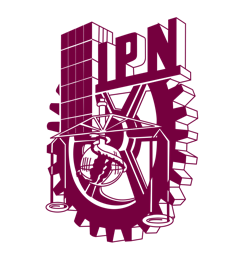
|  |  |  |
| --- | --- | --- |
| Alumno: | García González Aarón Antonio | **11** |
| Grupo: | 3CV2 | |
| Unidad de Aprendizaje: | Arquitectura de computadoras | |
| Profesora: | Vega García Nayeli | |
| Practica #12: | Carta ASM | |
| Fecha: | Domingo 24 de Mayo de 2020 | |

****

# Instrucciones de practica

Implemente la etapa arquitectura para encontrar el número de 1’s en un arreglo de 9 localidades y mostrar el resultado en un display de 7 segmentos, de acuerdo con la siguiente arquitectura y lógica de control, tome en cuenta que la arquitectura debe ser implementada de manera modular.

A close up of a map

Description automatically generated

# Unidad de control

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity unidad\_control is

Port ( ini : in STD\_LOGIC;

clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

a0 : in STD\_LOGIC;

z : in STD\_LOGIC;

la : out STD\_LOGIC;

ea : out STD\_LOGIC;

lb : out STD\_LOGIC;

ec : out STD\_LOGIC;

eb : out STD\_LOGIC);

end unidad\_control;

architecture Behavioral of unidad\_control is

type estados is (e0,e1,e2);

signal actual,siguiente : estados;

begin

process(clk, clr)

begin

if (clr = '1') then

actual <= e0;

elsif (rising\_edge(clk)) then

actual <= siguiente;

end if;

end process;

process(actual,ini,z,a0)

begin

la <= '0';

lb <= '0';

ea <= '0';

eb <= '0';

ec <= '0';

case actual is

when e0 =>

lb <= '1';

if (ini = '1') then

siguiente <= e1;

else

la <= '1';

siguiente <= e0;

end if;

when e1 =>

ea <= '1';

if (z = '1') then *-- arreglo igual a cero*

siguiente <= e2;

else *-- arreglo diferente de cero*

if (a0 = '1') then

eb <= '1';

siguiente <= e1;

else

siguiente <= e1;

end if;

end if;

when e2 =>

ec <= '1';

if (ini = '1') then

siguiente <= e2;

else

siguiente <= e0;

end if;

end case;

end process;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_unidad\_control is

end tb\_unidad\_control;

architecture Behavioral of tb\_unidad\_control is

component unidad\_control

Port ( ini : in STD\_LOGIC;

clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

a0 : in STD\_LOGIC;

z : in STD\_LOGIC;

la : out STD\_LOGIC;

ea : out STD\_LOGIC;

lb : out STD\_LOGIC;

ec : out STD\_LOGIC;

eb : out STD\_LOGIC);

end component;

*-- input signs*

signal ini, clk, clr, a0, z : STD\_LOGIC;

*-- output signs*

signal la, lb, ea, eb, ec : STD\_LOGIC;

*-- Clock period*

constant clk\_period : time := 10ns;

begin

*-- Instanciate*

UC : unidad\_control Port map (

ini => ini,

clr => clr,

clk => clk,

a0 => a0,

z => z,

la => la,

ea => ea,

lb => lb,

ec => ec,

eb => eb

);

*-- Clock process*

CLOCK : process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

*-- process stimulli*

SP : process

begin

clr <= '1';

wait for 30 ns;

clr <= '0';

wait for 60 ns;

ini <= '1';

wait for 10 ns;

ini <= '0';

wait for 50 ns;

a0 <= '1';

wait for 20 ns;

a0 <= '0';

wait for 20 ns;

a0 <= '1';

wait for 10 ns;

a0 <= '0';

wait for 120 ns;

z <= '1';

wait;

end process;

end Behavioral;

# Forma de onda de la simulación

A picture containing monitor, screen, sitting, holding

Description automatically generated

# Diagrama RTL



# Registro

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Registro is

Port ( la : in STD\_LOGIC;

ea : in STD\_LOGIC;

clk : in STD\_LOGIC;

clr : in STD\_LOGIC;

da : in STD\_LOGIC\_VECTOR (8 downto 0);

qa : out STD\_LOGIC\_VECTOR (8 downto 0));

end Registro;

architecture Behavioral of Registro is

signal aux\_a : STD\_LOGIC\_VECTOR(8 downto 0); *-- bus*

begin

process(clk,clr,la,ea)

begin

if (clr = '1') then

aux\_a <= "000000000";

elsif(rising\_edge(clk)) then

if (la = '0' and ea = '0') then

aux\_a <= aux\_a;

elsif (la = '1' and ea = '0') then

aux\_a <= da;

elsif (la = '0' and ea = '1') then

aux\_a <= to\_stdlogicvector(to\_bitvector(aux\_a) SRL 1);

end if;

end if;

end process;

qa <= aux\_a;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_Registro is

end tb\_Registro;

architecture Behavioral of tb\_Registro is

component Registro is

Port ( la : in STD\_LOGIC;

ea : in STD\_LOGIC;

clk : in STD\_LOGIC;

clr : in STD\_LOGIC;

da : in STD\_LOGIC\_VECTOR (8 downto 0);

qa : out STD\_LOGIC\_VECTOR (8 downto 0));

end component;

*-- input signs*

signal la, ea, clk, clr : STD\_LOGIC;

signal da : STD\_LOGIC\_VECTOR(8 downto 0);

*-- output signs*

signal qa : STD\_LOGIC\_VECTOR(8 downto 0);

*-- clock period definition*

constant clk\_period : time := 10 ns;

begin

*-- Instaciate component*

REG : Registro Port map(

la => la,

ea => ea,

clk => clk,

clr => clr,

da => da,

qa => qa

);

*-- clock process*

CLOCK : process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

*-- Stimulus process*

SP : process

begin

da <= "110011001";

wait for 150 ns;

clr <= '0';

la <= '1';

ea <= '0';

wait for 150 ns;

la <= '0';

ea <= '1';

wait for 150 ns;

wait;

end process;

end Behavioral;

# Forma de onda de la simulación

A picture containing screenshot, grass, player, baseball

Description automatically generated

# Diagrama RTL



# Contador

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Contador is

Port ( lb : in STD\_LOGIC;

eb : in STD\_LOGIC;

clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

qb : out STD\_LOGIC\_VECTOR (3 downto 0));

end Contador;

architecture Behavioral of Contador is

constant zero: std\_logic\_vector(3 downto 0) := "0000";

begin

process(clk, clr, eb, lb)

variable aux\_qb : STD\_LOGIC\_VECTOR(3 downto 0);

begin

if (clr = '1') then

aux\_qb := zero;

elsif (rising\_edge(clk)) then

if (lb='1' and eb='1') then

aux\_qb := aux\_qb;

elsif (lb='1' and eb='0') then

aux\_qb := zero;

elsif (lb='0' and eb='1') then

aux\_qb := aux\_qb + 1;

end if;

end if;

qb <= aux\_qb;

end process;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_Contador is

end tb\_Contador;

architecture Behavioral of tb\_Contador is

component Contador is

Port ( lb : in STD\_LOGIC;

eb : in STD\_LOGIC;

clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

qb : out STD\_LOGIC\_VECTOR (3 downto 0));

end component;

*-- input signs*

signal lb, eb, clr, clk : STD\_LOGIC;

*-- output signs*

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

begin

*-- Instaciate component for test*

COUNT : Contador Port map(

lb => lb,

eb => eb,

clr => clr,

clk => clk,

qb => qb

);

*-- Clock process*

CLOCK : process

begin

clk <= '0';

wait for 5ns;

clk <= '1';

wait for 5ns;

end process;

*-- Stimulus process*

SP : process

begin

clr <= '1';

lb <= '0';

eb <= '0';

wait for 150 ns;

clr <= '0';

eb <= '1';

wait for 150 ns;

eb <= '0';

lb <= '1';

wait for 150 ns;

wait;

end process;

end Behavioral;

# Forma de onda de la simulación

A picture containing player, ready, ball, baseball

Description automatically generated

# Diagrama RTL



# Decodificador

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decodificador is

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

digito\_out : out STD\_LOGIC\_VECTOR (6 downto 0));

end Decodificador;

architecture Behavioral of Decodificador is *-- "gfedcba"*

constant digito\_0 : STD\_LOGIC\_VECTOR(6 downto 0) := "0111111";

constant digito\_1 : STD\_LOGIC\_VECTOR(6 downto 0) := "0000110";

constant digito\_2 : STD\_LOGIC\_VECTOR(6 downto 0) := "1011011";

constant digito\_3 : STD\_LOGIC\_VECTOR(6 downto 0) := "1001111";

constant digito\_4 : STD\_LOGIC\_VECTOR(6 downto 0) := "1100110";

constant digito\_5 : STD\_LOGIC\_VECTOR(6 downto 0) := "1101101";

constant digito\_6 : STD\_LOGIC\_VECTOR(6 downto 0) := "1111101";

constant digito\_7 : STD\_LOGIC\_VECTOR(6 downto 0) := "0000111";

constant digito\_8 : STD\_LOGIC\_VECTOR(6 downto 0) := "1111111";

constant digito\_9 : STD\_LOGIC\_VECTOR(6 downto 0) := "1101111";

begin

process(qb)

begin

case qb is

when "0000" => digito\_out <= digito\_0;

when "0001" => digito\_out <= digito\_1;

when "0010" => digito\_out <= digito\_2;

when "0011" => digito\_out <= digito\_3;

when "0100" => digito\_out <= digito\_4;

when "0101" => digito\_out <= digito\_5;

when "0110" => digito\_out <= digito\_6;

when "0111" => digito\_out <= digito\_7;

when "1000" => digito\_out <= digito\_8;

when OTHERS => digito\_out <= digito\_9;

end case;

end process;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_Decodificador is

end tb\_Decodificador;

architecture Behavioral of tb\_Decodificador is

component Decodificador is

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

digito\_out : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

*-- input signal*

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

*-- output signal*

signal digito\_out : STD\_LOGIC\_VECTOR (6 downto 0);

begin

*-- Insatnciate compornent*

DECO : Decodificador Port map(

qb => qb,

digito\_out => digito\_out

);

*-- Stimulus process*

SP : process

begin

*-- Vamos a probar con cada numero*

qb <= "0000"; *-- 0*

wait for 10 ns;

qb <= "0001"; *-- 1*

wait for 10 ns;

qb <= "0010"; *-- 2*

wait for 10 ns;

qb <= "0011"; *-- 3*

wait for 10 ns;

qb <= "0100"; *-- 4*

wait for 10 ns;

qb <= "0101"; *-- 5*

wait for 10 ns;

qb <= "0110"; *-- 6*

wait for 10 ns;

qb <= "0111"; *-- 7*

wait for 10 ns;

qb <= "1000"; *-- 8*

wait for 10 ns;

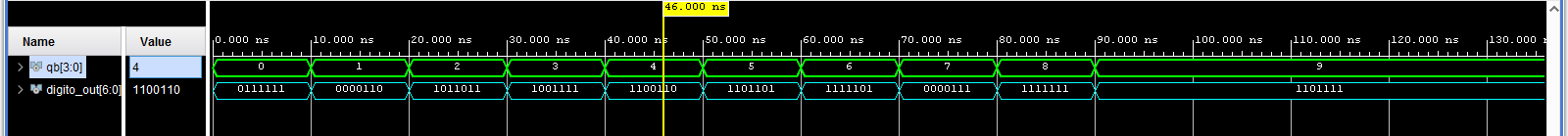
qb <= "1001"; *-- 9*

wait;

end process;

end Behavioral;

# Forma de onda de la simulación



# Diagrama RTL



# Multiplexor

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Multiplexor is

Port ( digito\_in : in STD\_LOGIC\_VECTOR (6 downto 0);

ec : in STD\_LOGIC;

digito\_final : out STD\_LOGIC\_VECTOR (6 downto 0));

end Multiplexor;

architecture Behavioral of Multiplexor is

constant guion : STD\_LOGIC\_VECTOR(6 downto 0) := "1000000"; *-- gfedcba*

begin

with ec select

digito\_final <= digito\_in when '1',

guion when others;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_Multiplexor is

end tb\_Multiplexor;

architecture Behavioral of tb\_Multiplexor is

component Multiplexor

Port ( digito\_in : in STD\_LOGIC\_VECTOR (6 downto 0);

ec : in STD\_LOGIC;

digito\_final : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

*-- input signs*

signal digito\_in : STD\_LOGIC\_VECTOR(6 downto 0);

signal ec : STD\_LOGIC;

*-- output sign*

signal digito\_final : STD\_LOGIC\_VECTOR(6 downto 0);

begin

*-- Instanciate component*

MUX : Multiplexor Port map(

digito\_in => digito\_in,

ec => ec,

digito\_final => digito\_final

);

*-- Stimulus process*

SP : process

begin

ec <= '0';

wait for 20 ns;

ec <= '1';

digito\_in <= "0111111"; *-- 0*

wait for 20 ns;

digito\_in <= "0000110"; *-- 1*

wait for 20 ns;

digito\_in <= "1011011"; *-- 2*

wait for 20 ns;

digito\_in <= "1001111"; *-- 3*

wait for 20 ns;

digito\_in <= "1100110"; *-- 4*

wait for 20 ns;

ec <= '0';

wait for 20 ns;

ec <= '1';

wait for 20 ns;

digito\_in <= "1101101"; *-- 5*

wait for 20 ns;

digito\_in <= "1111101"; *-- 6*

wait for 20 ns;

digito\_in <= "0000111"; *-- 7*

wait for 20 ns;

digito\_in <= "1111111"; *-- 8*

wait for 20 ns;

digito\_in <= "1100111"; *-- 9*

wait for 20 ns;

ec <= '0';

wait for 20 ns;

wait;

end process;

end Behavioral;

# Forma de onda de la simulación

# Diagrama RTL

# Carta ASM (arquitectura completa)

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity cartaASM is

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

ini : in STD\_LOGIC;

data\_in : in STD\_LOGIC\_VECTOR (8 downto 0);

data\_out : out STD\_LOGIC\_VECTOR (8 downto 0);

digit\_out : out STD\_LOGIC\_VECTOR (6 downto 0));

end cartaASM;

architecture Behavioral of cartaASM is

*-- unidad de control*

component unidad\_control is

Port ( ini, clr, clk, a0, z : in STD\_LOGIC;

la, ea, lb, ec, eb : out STD\_LOGIC);

end component;

*-- registros*

component Registro is

Port ( la, ea, clk, clr : in STD\_LOGIC;

da : in STD\_LOGIC\_VECTOR (8 downto 0);

qa : out STD\_LOGIC\_VECTOR (8 downto 0));

end component;

*-- contador*

component Contador is

Port ( lb, eb, clr, clk : in STD\_LOGIC;

qb : out STD\_LOGIC\_VECTOR (3 downto 0));

end component;

*-- decodificador*

component Decodificador is

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

digito\_out : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

*-- multiplexor*

component Multiplexor is

Port ( digito\_in : in STD\_LOGIC\_VECTOR (6 downto 0);

ec : in STD\_LOGIC;

digito\_final : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

*-- se�ales o buses de alambrado*

signal sgnLA, sgnLB, sgnEA, sgnEB, sgnEC, sgnZ : STD\_LOGIC;

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

signal sgnOutDeco : STD\_LOGIC\_VECTOR(6 downto 0);

signal sgnData : STD\_LOGIC\_VECTOR(8 downto 0);

begin

sgnZ <= '1' when sgnData = "000000000" else '0';

*-- Instaciar modulo de unidad de control*

UC : unidad\_control

Port map(

ini => ini,

clr => clr,

clk => clk,

a0 => sgnData(0),

z => sgnZ,

la => sgnLA,

ea => sgnEA,

lb => sgnLB,

ec => sgnEC,

eb => sgnEB

);

*-- Instanciar modulo de registro*

REG : Registro

Port map(

la => sgnLA,

ea => sgnEA,

clk => clk,

clr => clr,

da => data\_in,

qa => sgnData

);

*-- Instanciar modulo de contador*

CONT : Contador

Port map(

lb => sgnLB,

eb => sgnEB,

clr => clr,

clk => clk,

qb => sgnOutCounter

);

*-- Instanciar modulo de decodificacion*

DECO : Decodificador

Port map(

qb => sgnOutCounter,

digito\_out => sgnOutDeco

);

*-- Instanciar modulo de multiplexor*

MUX : Multiplexor

Port map(

digito\_in => sgnOutDeco,

ec => sgnEC,

digito\_final => digit\_out

);

data\_out <= sgnData;

end Behavioral;

# Código de simulación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_asm is

end tb\_asm;

architecture Behavioral of tb\_asm is

component cartaASM

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

ini : in STD\_LOGIC;

data\_in : in STD\_LOGIC\_VECTOR (8 downto 0);

data\_out : out STD\_LOGIC\_VECTOR (8 downto 0);

digit\_out : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

*-- input signs*

signal ini : STD\_LOGIC;

signal clk : STD\_LOGIC;

signal clr : STD\_LOGIC;

signal data\_in : STD\_LOGIC\_VECTOR (8 downto 0);

*-- outpuu signs*

signal data\_out : STD\_LOGIC\_VECTOR (8 downto 0);

signal digit\_out : STD\_LOGIC\_VECTOR (6 downto 0);

constant clk\_period : time := 10 ns;

begin

*-- clock*

CLOCK\_P : process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

uut : cartaASM Port map(

clr => clr,

clk => clk,

ini => ini,

data\_in => data\_in,

data\_out => data\_out,

digit\_out => digit\_out

);

*-- Process stimuli*

process\_stimuli : process

begin

clr <= '1';

clr <= '0';

wait for 20ns;

data\_in <= "101101011";

wait for 20 ns;

ini <= '1';

wait for 150 ns;

ini <= '0';

clr <= '1';

wait for 20 ns;

clr <= '0';

data\_in <= "000011101";

wait for 20 ns;

ini <= '1';

wait for 150 ns;

ini <= '0';

clr <= '1';

wait for 20 ns;

clr <= '0';

data\_in <= "000010000";

wait for 20 ns;

ini <= '1';

wait for 150 ns;

ini <= '0';

clr <= '1';

wait for 20 ns;

clr <= '0';

data\_in <= "100001000";

wait for 20 ns;

ini <= '1';

wait for 150 ns;

ini <= '0';

clr <= '1';

wait for 20 ns;

clr <= '0';

data\_in <= "000000000";

wait for 20 ns;

ini <= '1';

wait for 150 ns;

wait;

end process;

end Behavioral;

# Forma de onda de la simulación

A picture containdcdsccdccscing monitor, grass, screen, television

Description automatically generatedMuestra el cambio que se hace en el data out

Muestra el corrimiento

A picture containing screenshot

Description automatically generated

Muestra el ejemplo de simulación a

A picture containing screen, riding, player, light

Description automatically generated

Muestra el ejemplo de simulación b

A picture containing grass, screenshot, monitor, television

Description automatically generated

Muestra el ejemplo de simulación cA picture containing grass, monitor, television, green

Description automatically generated

Muestra el ejemplo de simulación d

A picture containing monitor, television, grass, screen

Description automatically generatedMuestra el ejemplo de simulación e

A picture containing riding, large, green, screen

Description automatically generated

# Diagrama RTL

