









library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity Practica11 is port(

clk, clr : in std\_logic;

ini : in std\_logic;

D : in std\_logic\_vector (5 downto 0);

LB, EB, EC : out std\_logic;

A : inout std\_logic\_vector (5 downto 0)

);

end Practica11;

architecture aPractica11 of Practica11 is

type estados is (e0, e1, e2);

signal edo\_act, edo\_sig : estados;

signal Z, LA, EA, a0 : std\_logic;

begin

--proceso que determina el estado actual y el siguiente

process(clk, clr)

begin

if(clr= '1') then

edo\_act <= e0;

elsif(rising\_edge(clk)) then

edo\_act <= edo\_sig;

end if;

end process;

--proceso que determina quien es el estado siguiente

process(edo\_act, ini, z, a0)

begin

LA <= '0';

EA <= '0';

LB <= '0';

EB <= '0';

EC <= '0';

case edo\_act is

when e0 =>

LB <= '1';

if(ini = '1') then

edo\_sig <= e1;

else

LA <= '1';

edo\_sig <= e0;

end if;

when e1 =>

EA <= '1';

if(Z = '1') then

edo\_sig <= e2;

elsif(z = '0') then

if(a0 = '1') then

EB <= '1';

elsif(a0 = '0') then

edo\_sig <= e1;

end if;

end if;

when e2 =>

EC <= '1';

if(ini = '1') then

edo\_sig <= e2;

else

edo\_sig <= e0;

end if;

end case;

end process;

--registro

process(clk, clr)

begin

if(clr = '1') then

A <= "000000";

elsif(rising\_edge(clk)) then

if(EA='0' and LA = '1') then

A <= D;

elsif(EA = '1' and LA = '0') then

A(5) <= '0';

for i in 0 to 4 loop

A(i) <= A(i+1);

end loop;

--A <= to\_stdlogicvector(to\_bitvector(A) ror 1); --falta corregir corrimiento a la derecha

end if;

end if;

end process;

Z <= not(A(5) or A(4) or A(3) or A(2) or A(1) or A(0));

a0 <= A(0);

end architecture;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity prac11 is port(

LB, EB :in std\_logic;

clk, clr: in std\_logic;

disp: out std\_logic\_vector(6 downto 0);

EC: inout std\_logic

);

end prac11;

architecture aprac11 of prac11 is

--signal conv: std\_logic\_vector(6 downto 0);

signal B: std\_logic\_vector(2 downto 0);

begin

process(clk, clr)

begin

if(clr = '1') then

B<="000";

elsif(rising\_edge(clk)) then

if(LB = '1')then

B<="000";

elsif(EB = '1') then

B<=B+1;

elsif(LB = '0' and EB = '0') then

B <= B;

end if;

end if;

end process;

process(EC, B)

begin

if(EC = '0') then

disp <= "0000001";

else

case B is

when "000" =>disp<="1111110";

when "001" =>disp<="0110000";

when "010" =>disp<="1101101";

when "011" =>disp<="1111001";

when "100" =>disp<="0110011";

when "101" =>disp<="1011011";

when "110" =>disp<="1011111";

when others =>disp<="0000001";

end case;

end if;

end process;

end architecture;

1. ¿Cuántos dispositivos PLD 22V10 son necesarios para el desarrollo de

esta práctica?

**2 dispositivos, uno para la carta ASM, y otro para el contador y salida a display**

2. ¿Cuántos dispositivos de la serie 74xx (TTL) ó 40xx (CMOS) hubieras

necesitado para el desarrollo de esta práctica?

**Más de 80**

3. ¿Cuántos pines de entrada/salida de cada PLD 22V10 se usan en el

diseño?

**GAL1: 9 de entrada, 9 de salida**

**GAL2: 5 de entrada, 7 de salida**

4. ¿Cuántos términos producto ocupan las ecuaciones para cada señal de

salida y que porcentaje se usa en total de los PLD 22V10?

**GAL1: 34, 28%**

**GAL2: 22, 18%**

5. ¿Cuántos FF’s ocupa el autómata de control de la microarquitectura?

6. ¿Qué puedes concluir de esta práctica?

**Fue la primera práctica en la que implementamos el diseño de una carta ASM, la cual es una manera diferente de representar un autómata de Mealy o Moore de manera mas intuitiva ya que se presentan los pasos a seguir como si fuera un algoritmo, la implementación es muy similar a lo que ya hemos trabajado durante el curso por lo que fue una práctica sencilla de implementar, personalmente tuve dificultades para concluir la práctica ya que mi entorno de simulación tuvo errores que no sabía como resolver, aunque mi lógica era correcta, el resultado no era el esperado, pero con ayuda de la profesora y compañeros pude concluirla.**