Registro de Carga Serial y Paralela

Taller de Sistemas Embebidos

Ishai Gun Roffe

Profesor: Oscar Alvarado Nava

Indice

Descripción del funcionamiento del circuito

Se realizó en una Nexys 3 un circuito compuesto de un registro, un decodificador y las entradas y salidas correspondientes para checar el correcto funcionamiento del mismo.

El registro guarda datos de 4 bits, está compuesto por cuatro Flip Flops de tipo D, el registro es capaz de recibir datos tanto de forma paralela como serial (del bit más significativo al menos significativo), la entrada paralela y serial del registro están conectadas a los switches de la tarjeta de desarrolla. La salida del registro es de tipo paralela y se conecta a cuatro leds y al display de siete segmentos, el cual también despliega valores hexadecimales, de esta manera es capaz de mostrar cualquier valor que reciba del registro.

Flip Flop

Descripción

El Flip Flop es el componente del registro que guarda la información.

Entradas y salidas:

clk	*entrada*	Señal de reloj.
reset	*entrada*	Señal de reset, asigna el valor cero a la salida q.
d	*entrada*	Dato de entrada, el dato que guardara el flip flop.
q	*salida*	Dato de salida, el dato que devuelve el flip flop.

Comportamiento:

t		t+1
reset	d	q
1	X	0
0	d	d

Código:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ff is
    Port ( clk   : in   STD_LOGIC;
           reset : in   STD_LOGIC;
           d     : in   STD_LOGIC;
           q     : out  STD_LOGIC);
end ff;

architecture Behavioral of ff is

signal t : STD_LOGIC;

begin

process(clk,reset,d)
begin
       if clk='1' then
              if reset='1' then
                     t <= '0';
              else
                     t <= d;
              end if;
       end if;
end process;

process(clk,d)
begin
       if clk='0' then
              q <= t;
       end if;
end process;

end Behavioral;
```

Multiplexor

Descripción

El multiplexor es el que se encarga de asignar cual será la entrada del flip flop, este determina si el dato que se recibira será la salida del flip flop anterior (serial) o el valor recibido del switch correspondiente, todos los flip flops tienen un multiplexor excepto el primero (el que guarda el bit más significativo) ya que la entrada de este es la entrada paralela de este flip flop asi como la entrada serial del registro en si.

Entradas y Salidas:

a	*entrada*	Salida del Flip Flop anterior
b	*entrada*	Entrada del swich
s	*entrada*	Selecciona la entrada que pasará
o	*salida*	Hacia el siguiente Flip Flop

Comportamiento:

Entrada	Salida
s	o
0	a
1	b

Código:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mltp is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : in  STD_LOGIC;
           o : out  STD_LOGIC);
end mltp;

architecture Behavioral of mltp is

begin
      process(s,a,b)
      begin
      if s='0' then
                  o <= a;
            else
                  o <= b;
      end if;
      end process;
end Behavioral;
```

Reloj

Descripción

El reloj es necesario para la transferencia serial de datos, este tiene una duración de al rededor de un segundo y puede ser habilitado o deshabilitado, esto con la finalidad de hacer más fácil la interacción con el usuario.

Este módulo está encargado de reducir la frecuencia de reloj de la Nexys 3, de 100 Mhz a 1Hz.

Entradas y Salidas:

ci	entrada	Entrada 100 Mhz
co	salida	Salida 1 Hz

Código:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reloj is
    Port ( ci : in  STD_LOGIC;
           co : out  STD_LOGIC);
end reloj;

architecture Behavioral of reloj is
      signal x: STD_LOGIC := '0';
      signal y: integer range 0 to 99999999 := 0;
begin
      fd: process(ci)
      begin
            if rising_edge(ci) then
                  if (y=99999999) then
                        x <= NOT(x);
                        y <= 0;
                  else
                        y <= y + 1;
                  end if;
            end if;
      end process;

co <= x;

end Behavioral;
```

Decodificador

Descripción

El decodificador convierte el valor hexadecimal que recibe del registro a un valor que se envía al display de siete segmentos que pueda ser fácilmente legible por una persona.

Este módulo puede representar valores de *0000* a *1111* en formato hexadecimal (0,1,2,....,C,D,E,F).

Entradas y salidas:

i	entrada	Entrada de 4 bits
o	salida	Salida a Display

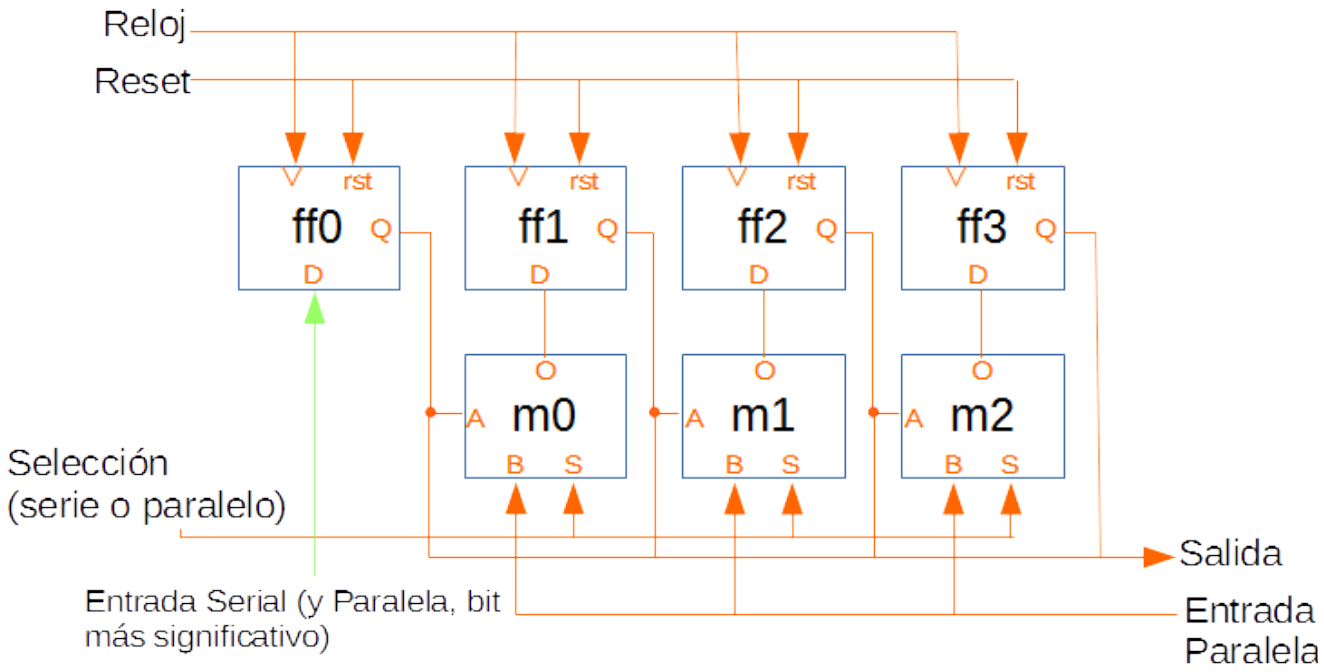
Código

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decod is
    Port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
           o : out  STD_LOGIC_VECTOR (11 downto 0));
end decod;

architecture Behavioral of decod is

begin
process(i)
begin
      case i is
            when "0000" => o <= "000000111110"; --0
            when "1000" => o <= "100111111110"; --1
            when "0100" => o <= "001001011110"; --2
            when "1100" => o <= "000011011110"; --3
            when "0010" => o <= "100110011110"; --4
            when "1010" => o <= "010010011110"; --5
            when "0110" => o <= "010000011110"; --6
            when "1110" => o <= "000111111110"; --7
            when "0001" => o <= "000000011110"; --8
            when "1001" => o <= "000010011110"; --9
            when "0101" => o <= "000100011110"; --a
            when "1101" => o <= "110000011110"; --b
            when "0011" => o <= "011000111110"; --c
            when "1011" => o <= "100001011110"; --d
            when "0111" => o <= "011000011110"; --e
            when others => o <= "011100011110"; --f
      end case;
end process;
end Behavioral;
```

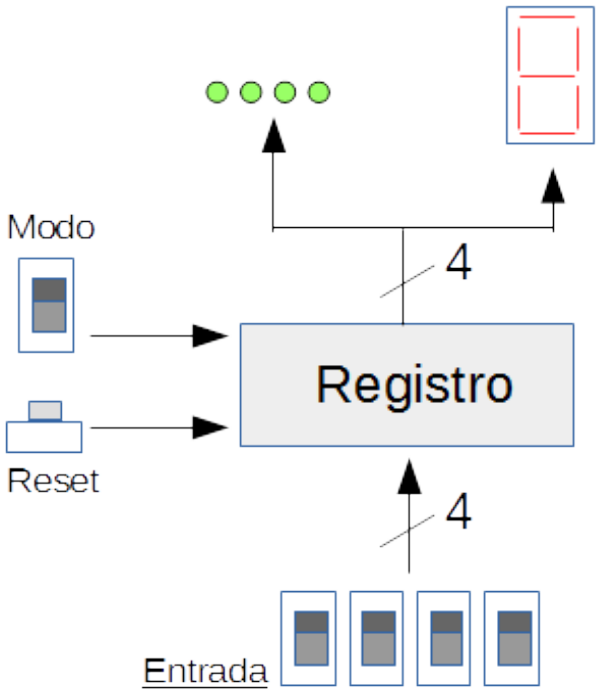
Modulo Principal

El módulo principal es donde se hacen las instancias de todos los módulos y se interconectan entre sí.

El registro está compuesto de cuatro flip flops y tres multiplexores, los cuales están conectados de la siguiente manera.



El circuito en general se puede representar de la siguiente manera:



Código:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clk : in STD_LOGIC;
           sw : in  STD_LOGIC_VECTOR (5 downto 0);
           pb : in  STD_LOGIC;
           led : out  STD_LOGIC_VECTOR (7 downto 0);
           disp : out  STD_LOGIC_VECTOR (11 downto 0));
end main;

architecture Behavioral of main is

component decod
      Port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
             o : out  STD_LOGIC_VECTOR (11 downto 0));
end component;

component ff
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           d : in  STD_LOGIC;
           q : out  STD_LOGIC);
end component;

component mltp
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : in  STD_LOGIC;
           o : out  STD_LOGIC);
end component;

component reloj
      Port ( ci : in STD_LOGIC;
             co : out STD_LOGIC) ;
end component;

signal s: STD_LOGIC_VECTOR (3 downto 0);
signal r: STD_LOGIC_VECTOR (2 downto 0);
signal c,d: STD_LOGIC;

begin

ff0:ff
port map(
      clk => d,
      reset => pb,
      d => sw(3),
      q => s(0)
);

led(3) <= s(0);
```

```
mltp0:mltp
port map(
      a => s(0),
      b => sw(2),
      s => sw(4),
      o => r(0)
);

ff1:ff
port map(
      clk => d,
      reset => pb,
      d => r(0),
      q => s(1)
);

led(2) <= s(1);

mltp1:mltp
port map(
      a => s(1),
      b => sw(1),
      s => sw(4),
      o => r(1)
);

ff2:ff
port map(
      clk => d,
      reset => pb,
      d => r(1),
      q => s(2)
);

led(1) <= s(2);

mltp2:mltp
port map(
      a => s(2),
      b => sw(0),
      s => sw(4),
      o => r(2)
);

ff3:ff
port map(
      clk => d,
      reset => pb,
      d => r(2),
      q => s(3)
);

led(0) <= s(3);
```

```
decod0:decod
port map(
      i => s,
      o => disp
);

reloj0: reloj
port map(
      ci => clk,
      co => c
);

process(sw(5),c)
begin
      d <= c AND sw(5);
end process;

led(7) <= c;
led(6) <= d;
led(5) <= '0';
led(4) <= '0';

end Behavioral;
```