INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

DISEÑO DE SISTEMAS DIGITALES

EXAMEN SEGUNDO PARCIAL

PROFESOR:
Testa Nava Alexis

Alumno:
Ramírez Cotonieto Luis Fernando

GRUPO:
2CV18

FECHA:
11/Mayo/2021

Examen del Segundo Parcial

Aplicando la metodología de diseño y el análisis necesario. Diseñar un contador síncrono ascendente-descendente arbitrario, utilizando flip-flop T, D y JK en el orden que usted lo deseé.
El programa por diseñar mostrará su numero de boleta en un Display omitiendo aquellos números que se repitan.

Tipo de automata: Moore

Matricula:

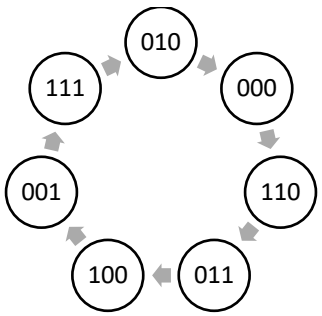
2 0 2 0 6 3 0 4 1 7

Mostrará:

2 0 6 3 4 1 7

Decimal	Binario
0	000
1	001
2	010
3	011
4	100
6	110
7	111

Flujo de estados



Tablas de verdad de los Flip-Flop a utilizar

Q	Q+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q	Q+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Q	Q+1	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabla de transición

Q			Q+1			D2	T1	J0	K0
A	B	C	A'	B'	C'				
0	0	0	1	1	0	1	1	0	0
0	0	1	1	1	1	1	1	X	0
0	1	0	0	0	0	0	1	0	X
0	1	1	1	0	0	1	1	X	1
1	0	0	0	0	1	0	0	1	X
1	0	1	X	X	X	x	X	X	X
1	1	0	0	1	1	0	0	1	X
1	1	1	0	1	0	0	0	X	1

Ecuaciones – Mapas de Karnaugh

Flip Flop D

Q1Q0 / Q2 \	00	01	11	10
0	1	1	1	
1		X		

Ec.D= Q2'Q1' + Q2'Q0
Ec.D = Q2'(Q1'+Q0)

Flip Flop T

Q1Q0 / Q2 \	00	01	11	10
0	1	1	1	1
1		X		

Ec.T= Q2'

Flip Flop JK

Flip Flop J

Q1Q0 / Q2 \	00	01	11	10
0		X	X	
1	1	X	X	1

Ec.J = Q2

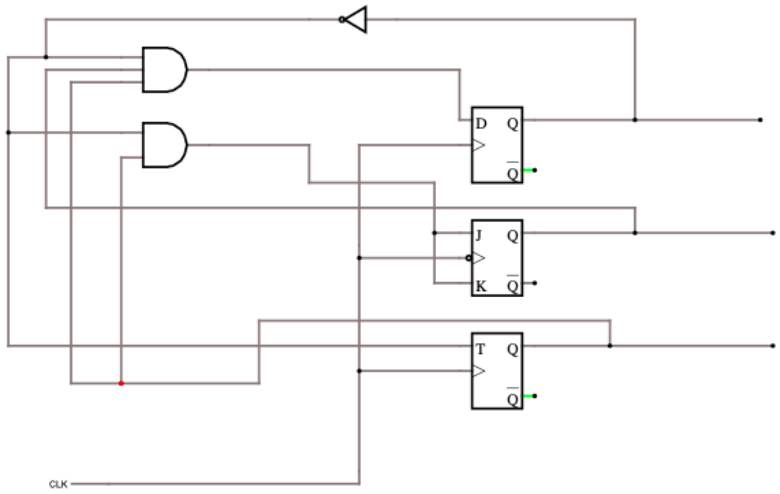
Flip Flop K

Q1Q0 / Q2 \	00	01	11	10
0	X		1	X
1	X	X	1	X

Ec.K= Q1

Circuito

https://tinyurl.com/yjy22vot



t = 0 s
intervalo tiempo = 5 µs
1 Conexión incorrecta

Código VHDL

```
library ieee;
use ieee.std_logic_1164.all;

entity ExamenSegundoLFRC is
port (clock, clear, enable: in std_logic;
      display: out std_logic_vector(6 downto 0) );

attribute pin_numbers of ExamenSegundoLFRC: entity is
"clock:1 clear:13 enable:2"
& " display(0):15 display(1):16 display(2):17 display(3):18 display(4):19 display(5):20 display(6):21";
--& " display(0):21 display(1):20 display(2):19 display(3):18 display(4):17 display(5):16 display(6):15";

end ExamenSegundoLFRC;

architecture A_ExamenSegundoLFRC of ExamenSegundoLFRC is

-- Etiqueta
constant Etiqueta0: std_logic_vector(1 downto 0) := "00";
constant Etiqueta1: std_logic_vector(1 downto 0) := "01";
constant Etiqueta2: std_logic_vector(1 downto 0) := "10";

-- Matricula: 2020630417
-- Numero a mostrar: 2063417
--Simbolos (numeros)
constant Numero2 : std_logic_vector(6 downto 0) := "0010010";
```

```
constant Numero0 : std_logic_vector(6 downto 0) := "0000001";
constant Numero6 : std_logic_vector(6 downto 0) := "0100000";
constant Numero3 : std_logic_vector(6 downto 0) := "0000110";
constant Numero4 : std_logic_vector(6 downto 0) := "1001100";
constant Numero1 : std_logic_vector(6 downto 0) := "1001111";
constant Numero7 : std_logic_vector(6 downto 0) := "0001111";


--Estados
--Usando un codigo definido por el usuario

constant Estado0 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero2; --"000010010"; --Etiqueta0 & Numero2;
constant Estado1 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero0; --"000000001"; --Etiqueta0 & Numero0;
constant Estado2 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero6; --"000100000"; --Etiqueta0 & Numero6;
constant Estado3 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero3; --"000000110"; --Etiqueta0 & Numero3;
constant Estado4 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero4; --"001001100"; --Etiqueta0 & Numero4;
constant Estado5 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero1; --"001001111"; --Etiqueta0 & Numero1;
constant Estado6 : std_logic_vector(7 downto 0) := Etiqueta0 & Numero7; --"000001111"; --Etiqueta0 & Numero7;


--señal
--"salida" hace referencia al cambio de estado y la salida que tiene el estado

signal salida: std_logic_vector(7 downto 0);

begin
   process(clock, clear)
   begin
      if(clear = '1') then
         salida <= Estado0;

      elsif(clock'event and clock = '1') then
         if(enable = '1') then
            case (salida) is
               when Estado0 => salida <= Estado1;
               when Estado1 => salida <= Estado2;
               when Estado2 => salida <= Estado3;
               when Estado3 => salida <= Estado4;
               when Estado4 => salida <= Estado5;
               when Estado5 => salida <= Estado6;
               when Estado6 => salida <= Estado0;
```

```
                when others => salida <= Estado0;
            end case;
        end if;
    end if;
end process;

display <= salida(6 downto 0);

end A_ExamenSegundoLFRC;
```