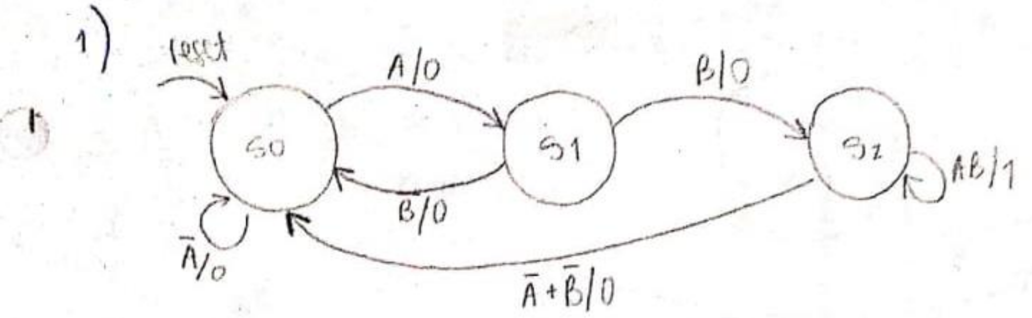
Valerie Valdez Carné: 19659
Universidad del Valle de Guatemala Sección: algo

Electrónica Digital
Laboratorio #6

Parte No. 1

1)



Caja Negra

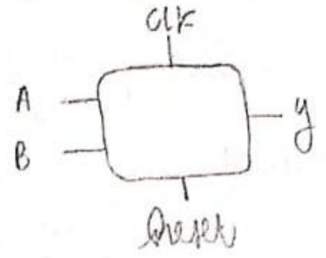


Tabla de transiciones

Presente	input A	input B	futuro	output
S_0	1	x	S_1	0
S_0	0	x	S_0	0
S_1	x	1	S_2	0
S_1	x	0	S_0	0
S_2	0	0	S_0	0
S_2	1	1	S_2	1
S_2	1	0	S_0	0
S_2	0	1	S_0	0

Tabla codificada

00	S_0
01	S_1
10	S_2

Tabla de transiciones de estado codificada

Actual S_1	Actual S_0	input A	input B	futuro S_1'	futuro S_0'	output
0	0	1	x	0	1	0
0	0	0	x	0	0	0
0	1	x	1	1	0	0
0	1	x	0	0	0	0
1	0	0	0	0	0	0
1	0	1	1	1	0	1
1	0	1	0	0	0	0
1	0	0	1	0	0	0

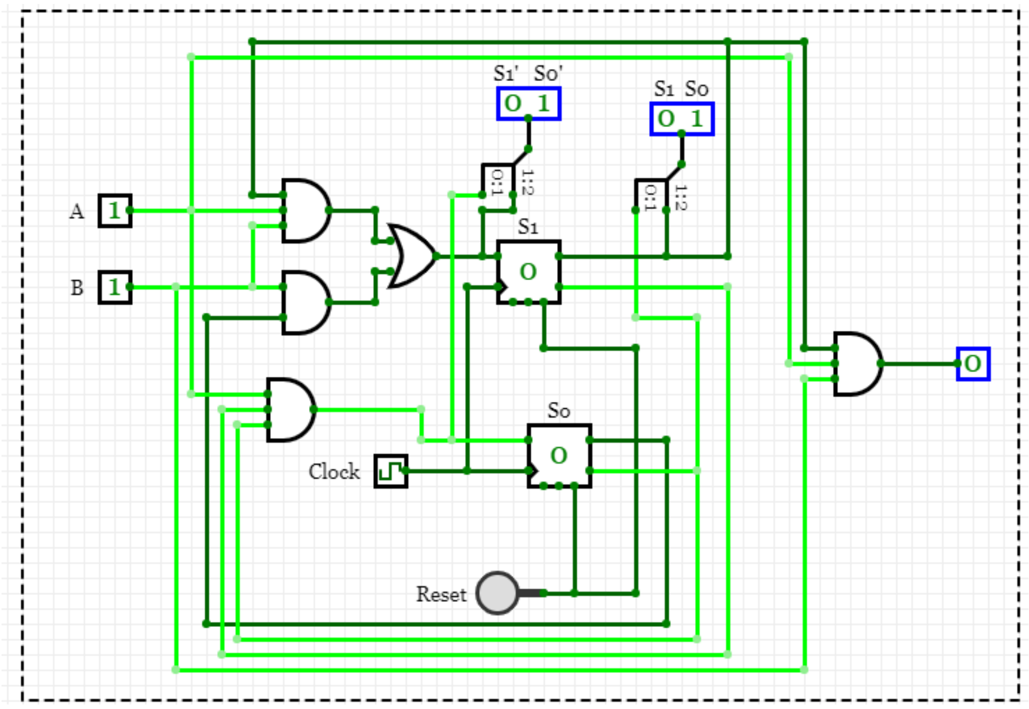
Ecuaciones booleanas

$S_1' = S_0 B + S_1 A B$

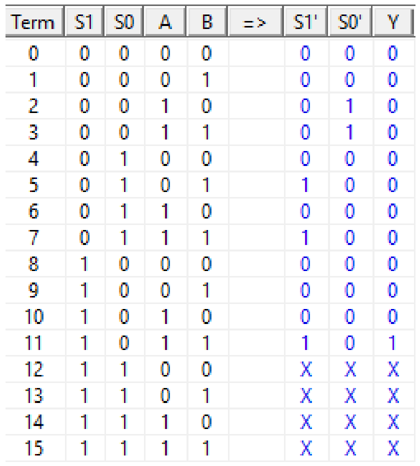
$S_0' = S_1' S_0' A$

$y = S_1 A B$

Implementación en Circuitverse



Implementación en Logic Friday

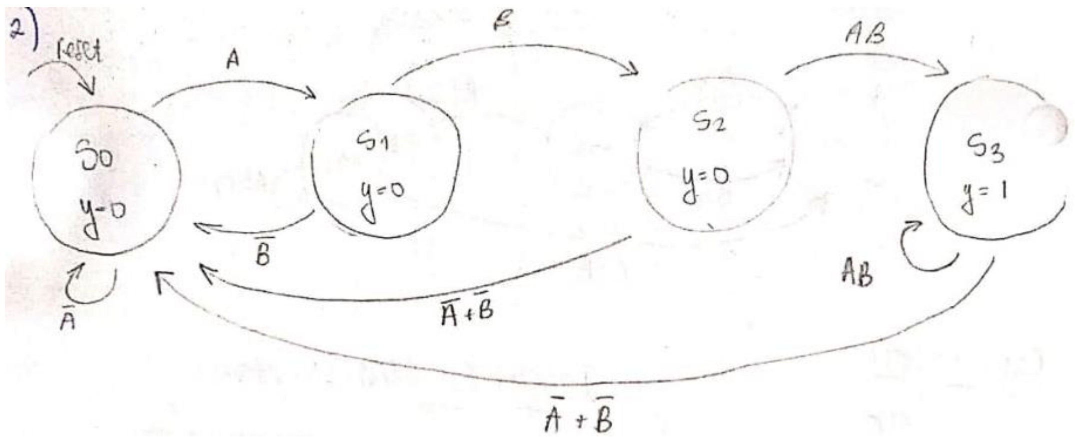


Term	S1	S0	A	B	=>	S1'	S0'	Y
0	0	0	0	0		0	0	0
1	0	0	0	1		0	0	0
2	0	0	1	0		0	1	0
3	0	0	1	1		0	1	0
4	0	1	0	0		0	0	0
5	0	1	0	1		1	0	0
6	0	1	1	0		0	0	0
7	0	1	1	1		1	0	0
8	1	0	0	0		0	0	0
9	1	0	0	1		0	0	0
10	1	0	1	0		0	0	0
11	1	0	1	1		1	0	1
12	1	1	0	0		X	X	X
13	1	1	0	1		X	X	X
14	1	1	1	0		X	X	X
15	1	1	1	1		X	X	X

Ecuaciones booleanas

```
Entered by truthtable:
S1' = S1' S0 A' B + S1' S0 A B + S1 S0' A B;
S0' = S1' S0' A B' + S1' S0' A B;
Y = S1 S0' A B;

Minimized:
S1' = S0 B + S1 A B;
S0' = S1' S0' A ;
Y = S1 A B;
```

Parte No. 2



Parte No. 3

3)

Número	código Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

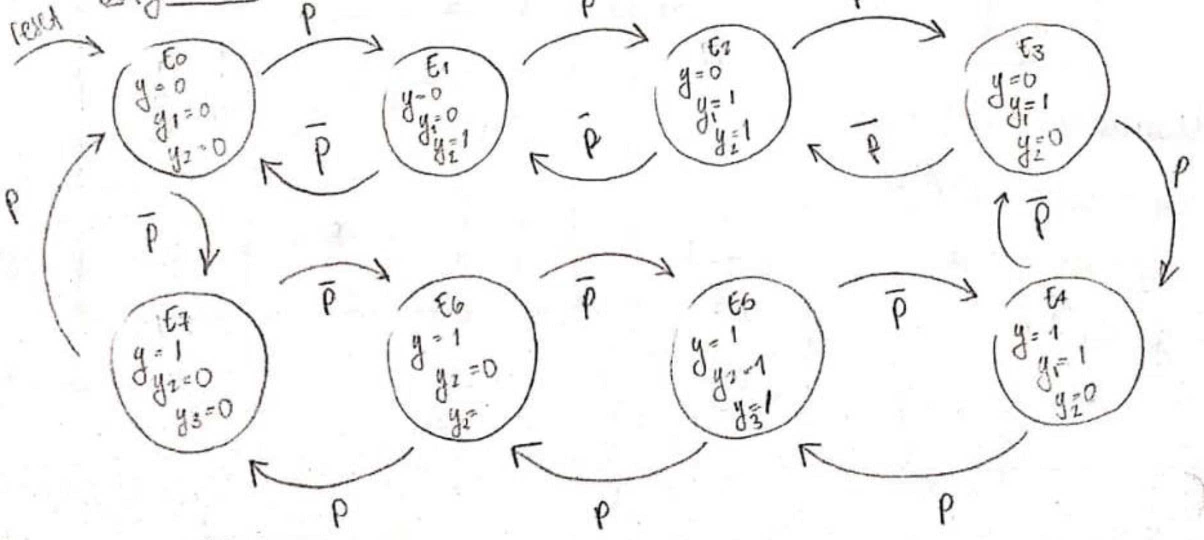
cambia en el flanco de reloj

(0, N-1)

Entrada = 1 ↑

Entrada = 0 ↓

Diagrama


reset
E_0 $y=0$ $y_1=0$ $y_2=0$
E_1 $y=0$ $y_1=0$ $y_2=1$
E_2 $y=0$ $y_1=1$ $y_2=1$
E_3 $y=0$ $y_1=1$ $y_2=0$
E_4 $y=1$ $y_1=1$ $y_2=0$
E_5 $y=1$ $y_2=1$ $y_3=1$
E_6 $y=1$ $y_2=0$ $y_2=$
E_7 $y=1$ $y_2=0$ $y_3=0$
p
$\bar{p}$

donde: $\boxed{\begin{matrix} p = 1 \\ \bar{p} = 0 \end{matrix}}$

Cajas Negra

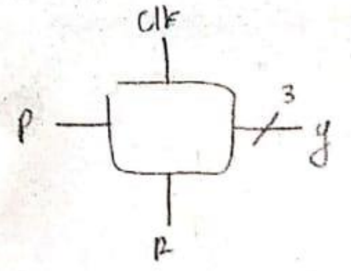


Tabla de salidas

Presente			output			
S_2	S_1	S_0	y_1	y_2	y_3	
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	1	2
0	1	1	0	1	0	4
1	0	0	1	1	0	3
1	0	1	1	1	1	5
1	1	0	1	0	1	6
1	1	1	1	0	0	7

Ecuaciones booleanas

$S_2' = S_2'S_1S_0P + S_2'S_1'S_0'P' + S_2S_0P' + S_2S_1P' + S_2S_1S_0'$

$S_1' = S_1S_0P' + S_1'S_0P + S_1'S_0'P' + S_1S_0'P$

$S_0' = S_0'$

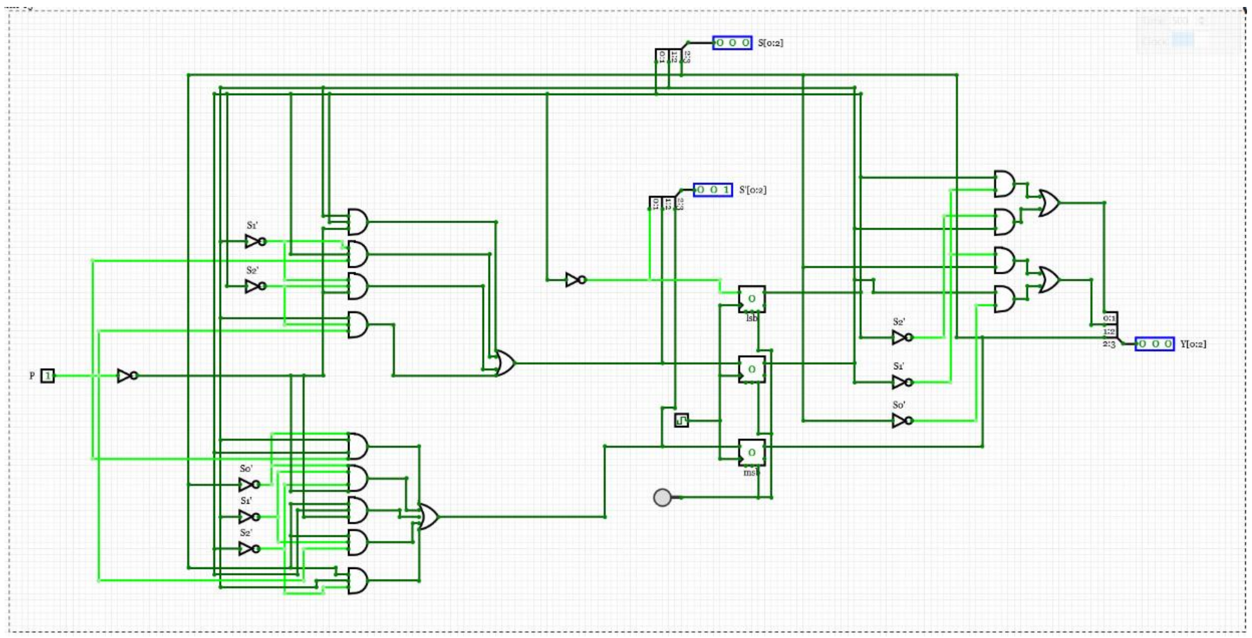
Tabla de transiciones

Presente	input	futuro
E0	0	E7
E0	1	E1
E1	0	E0
E1	1	E2
E2	0	E1
E2	1	E3
E3	0	E2
E3	1	E4
E4	0	E3
E4	1	E5
E5	0	E4
E5	1	E6
E6	0	E5
E6	1	E7
E7	0	E6
E7	1	E0

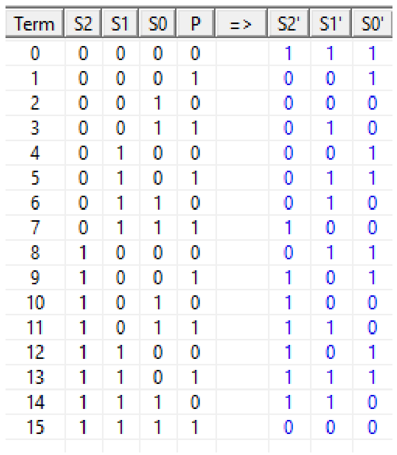
Tabla de transiciones codificada

Presente			input	futuro		
S_2	S_1	S_0	up/down	S_2'	S_1'	S_0'
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	0	0	0

Implementación en Circuitverse



Implementación en Logic Friday

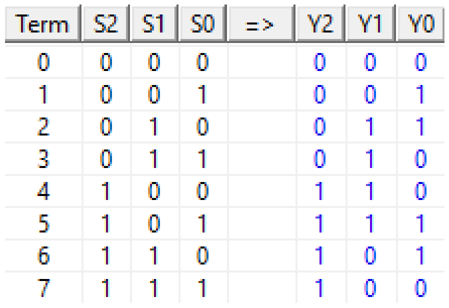


Term	S2	S1	S0	P	=>	S2'	S1'	S0'
0	0	0	0	0		1	1	1
1	0	0	0	1		0	0	1
2	0	0	1	0		0	0	0
3	0	0	1	1		0	1	0
4	0	1	0	0		0	0	1
5	0	1	0	1		0	1	1
6	0	1	1	0		0	1	0
7	0	1	1	1		1	0	0
8	1	0	0	0		0	1	1
9	1	0	0	1		1	0	1
10	1	0	1	0		1	0	0
11	1	0	1	1		1	1	0
12	1	1	0	0		1	0	1
13	1	1	0	1		1	1	1
14	1	1	1	0		1	1	0
15	1	1	1	1		0	0	0

Ecuaciones booleanas

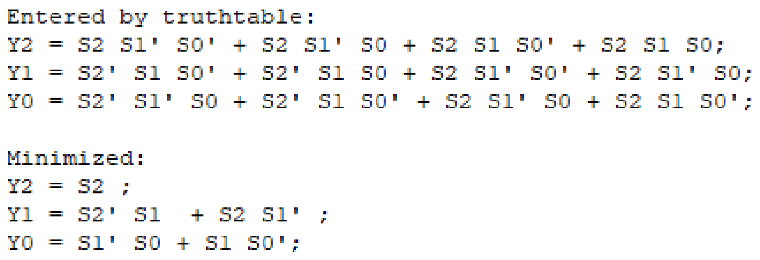
```
Entered by truthtable:
S2' = S2' S1' S0' P' + S2' S1 S0 P + S2 S1' S0' P + S2 S1' S0 P' + S2
S1' S0 P + S2 S1 S0' P' + S2 S1 S0' P + S2 S1 S0 P';
S1' = S2' S1' S0' P' + S2' S1' S0 P + S2' S1 S0' P + S2' S1 S0 P' +
S2 S1' S0' P' + S2 S1' S0 P + S2 S1 S0' P + S2 S1 S0 P';
S0' = S2' S1' S0' P' + S2' S1' S0' P + S2' S1 S0' P' + S2' S1 S0' P +
S2 S1' S0' P' + S2 S1' S0' P + S2 S1 S0' P' + S2 S1 S0' P;

Minimized:
S2' = S2' S1 S0 P + S2' S1' S0' P' + S2 S1' S0  + S2 S0' P + S2 S1
P';
S1' = S1' S0 P + S1 S0' P + S1 S0 P' + S1' S0' P';
S0' = S0' ;
```

Tabla de salidas



Term	S2	S1	S0	=>	Y2	Y1	Y0
0	0	0	0		0	0	0
1	0	0	1		0	0	1
2	0	1	0		0	1	1
3	0	1	1		0	1	0
4	1	0	0		1	1	0
5	1	0	1		1	1	1
6	1	1	0		1	0	1
7	1	1	1		1	0	0

Ecuaciones de salida

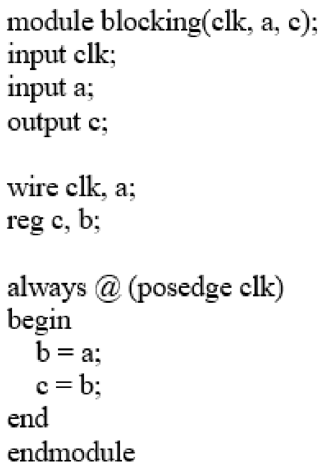


```
Entered by truthtable:
Y2 = S2 S1' S0' + S2 S1' S0 + S2 S1 S0' + S2 S1 S0;
Y1 = S2' S1 S0' + S2' S1 S0 + S2 S1' S0' + S2 S1' S0;
Y0 = S2' S1' S0 + S2' S1 S0' + S2 S1' S0 + S2 S1 S0';

Minimized:
Y2 = S2 ;
Y1 = S2' S1  + S2 S1' ;
Y0 = S1' S0 + S1 S0';
```

Parte 4

Un *blocking assigment* trabaja "en serie" debido a que tiene que ejecutarse antes de que se ejecuten los estados que le siguen en un bloque secuencial. Sin embargo, no evitará la ejecución de declaraciones que se ejecutan en un bloque paralelo. Por otro lado, para asignarlas se utiliza el signo igual (=).

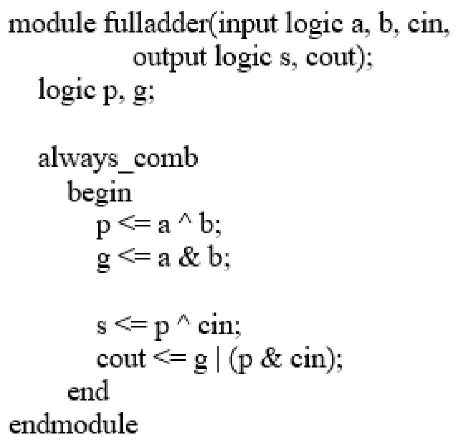
Ejemplo



```
module blocking(clk, a, c);
input clk;
input a;
output c;

wire clk, a;
reg c, b;

always @ (posedge clk)
begin
   b = a;
   c = b;
end
endmodule
```

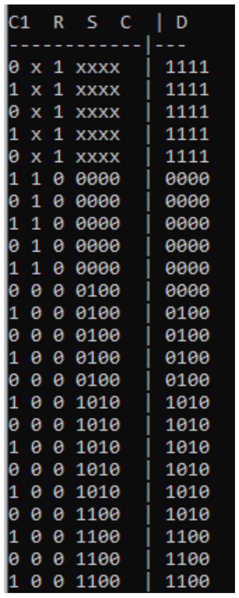
Un *nonblocking statement* permite programar asignaciones sin bloquear el flujo del procedimiento. Por lo que, se puede utilizar al momento de realizar varias asignaciones de registros dentro del mismo paso del tiempo sin preocuparse por el orden o la dependencia de cada uno y para asignarla se utiliza <= Por esta razón se dice que se parece más al hardware real que las asignaciones de bloqueo.

Ejemplo



```
module fulladder(input logic a, b, cin,
                 output logic s, cout);
  logic p, g;

  always_comb
    begin
      p <= a ^ b;
      g <= a & b;

      s <= p ^ cin;
      cout <= g | (p & cin);
    end
endmodule
```

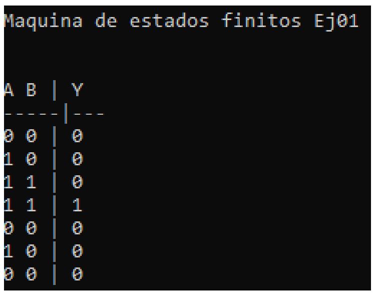
Parte No. 5



```
C1  R  S  C   | D
------------|---
0 x 1 xxxx  | 1111
1 x 1 xxxx  | 1111
0 x 1 xxxx  | 1111
1 x 1 xxxx  | 1111
0 x 1 xxxx  | 1111
1 1 0 0000  | 0000
0 1 0 0000  | 0000
1 1 0 0000  | 0000
0 1 0 0000  | 0000
1 1 0 0000  | 0000
0 0 0 0100  | 0000
1 0 0 0100  | 0100
0 0 0 0100  | 0100
1 0 0 0100  | 0100
0 0 0 0100  | 0100
1 0 0 1010  | 1010
0 0 0 1010  | 1010
1 0 0 1010  | 1010
0 0 0 1010  | 1010
1 0 0 1010  | 1010
0 0 0 1100  | 1010
1 0 0 1100  | 1100
0 0 0 1100  | 1100
1 0 0 1100  | 1100
```

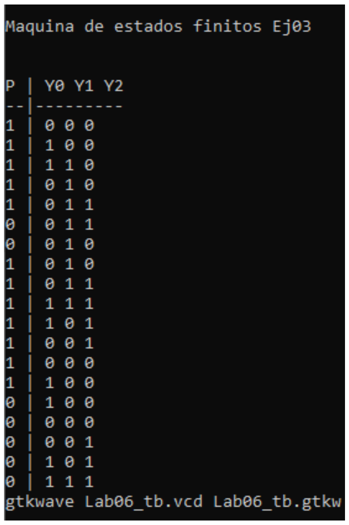
Parte No. 6

Tabla de la máquina de estados finitos Ejercicio 01



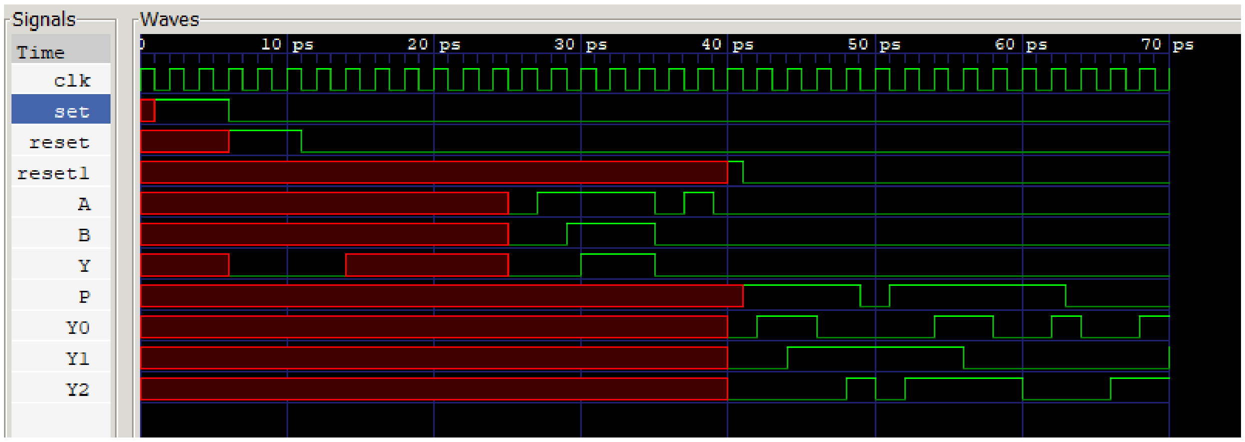
```
Maquina de estados finitos Ej01


A B | Y
-----|---
0 0 | 0
1 0 | 0
1 1 | 0
1 1 | 1
0 0 | 0
1 0 | 0
0 0 | 0
```

Tabla de la máquina de estados finitos Ejercicio 03



```
Maquina de estados finitos Ej03


P | Y0 Y1 Y2
--|---------
1 | 0 0 0
1 | 1 0 0
1 | 1 1 0
1 | 0 1 0
1 | 0 1 1
0 | 0 1 1
0 | 0 1 0
1 | 0 1 0
1 | 0 1 1
1 | 1 1 1
1 | 1 0 1
1 | 0 0 1
1 | 0 0 0
1 | 1 0 0
0 | 1 0 0
0 | 0 0 0
0 | 0 0 1
0 | 1 0 1
0 | 1 1 1
gtkwave Lab06_tb.vcd Lab06_tb.gtkw
```

Diagrama de timing para ambos ejercicios



Link del repositorio:

https://github.com/valeelorraine/Laboratorios_Digital