Examen – 23 de Diciembre de 2011

Instrucciones

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado.**
- **Empiece cada ejercicio en una hoja nueva.**
- **Sólo se contestarán dudas de letra. No se aceptarán dudas los últimos 30 minutos del examen.**
- **Duración: 3 horas.**
- **Requisito para Aprobación: responder al menos la mitad del problema teórico (2 de las 4 preguntas) y resolver un problema de práctico de forma correcta.**

Problema 1

Pregunta 1)
Partiendo de la expresión de suma de productos canónicos de una función booleana, explique por qué la compuerta NAND se considera un conjunto logicamente completo. Justifique detalladamente.

Pregunta 2)
Describa los pasos que debe seguir un procesador para atender interrupciones (ciclo de interrupciones).

Pregunta 3)
a) ¿ Cuál es el registro que determina las acciones de la Unidad de Control de un microprocesador? Especifique los pasos necesarios para decodificar y ejecutar una instrucción.
b) Explique que se entiende por control cableado y control microprogramado.

Pregunta 4)
Enumere y describa qué tipos de hazards (obstáculos) pueden aparecer en la aplicación de la técnica de pipelining. Mostrar un ejemplo de cada tipo.

Solución:

ATENCIÓN: LAS SOLUCIONES PUBLICADAS DE LAS PREGUNTAS TEÓRICAS SON LAS MÍNIMAS ACEPTABLES.

Pregunta 1:
Un conjunto lógicamente completo es aquel que me permite expresar cualquier función booleana con los operadores de dicho conjunto.
Como cualquier función booleana puedo expresarla como suma de productos canónicos, basta entonces con que los operadores del conjunto pueda expresar los operadores AND, OR y NOT.
Not:

$$a \# a = \overline{a.a} = \overline{a} + \overline{a} = \overline{a}$$

(Paso 1: Por definición NAND, Paso 2: De morgan, Paso 3: Definición de OR)
And:

$$(a \# b) \# (a \# b) = \overline{(a \# b).(a \# b)} = \overline{(a \# b)}$$

$$\overline{(a \# b)} = \overline{\overline{ab}} = ab$$

Paso 1: Definición de NAND, Paso 2: Definición de AND, Paso 3: Definición de NAND, Paso 4: Complemento de complemento.

Or:

$(a\#a)\#(b\#b)=\overline{(a\#a).(b\#b)}=\overline{\overline{(a.a)}.\overline{(b.b)}}$

$\overline{\overline{(a.a)}.\overline{(b.b)}}=\overline{\overline{a}.\overline{b}}=\overline{\overline{a+b}}=a+b$

Paso 1: Definición de NAND, Paso 2: Definición de NAND, Paso 3: Definición de AND, Paso 4: De morgan, Paso 5: Complemento de Complemento.

Pregunta 2:

1. Se termina de ejecutar la instrucción actual.
2. Se salva en memoria la dirección de la siguiente instrucción a ejecutar.
3. Se identifica el dispositivo que hizo la interrupción.
4. Se obtiene la dirección de la rutina que atiende dicha interrupción.
5. Se enmascaran las interrupciones.
6. Se pasa a ejecutar la rutina de atención para esa interrupción.

Pregunta 3

1. El registro que determina los pasos a realizar por la UC es el INSTRUCTION REGISTER que tiene la instrucción a ejecutar.
El decode consiste en identificar la instrucción a ejecutar mediante el código de operación y los operandos de la misma.
El execute consiste en tomar los valores identificados por los operandos de la instrucción y realizar la operación indicada en el código de operación.

2. Por control cableado entendemos aquella unidad de control implementada como una máquina secuencial donde los pasos a seguir en el ciclo de instrucción están codificados en compuertas lógicas.
Por control microprogramado nos referimos a una unidad de control cuya implementación se realiza mediante una memoria rom en donde se guarda el microprograma que lleva adelante el ciclo de instrucción. Las líneas de control se corresponden con líneas de salida de dicha micro-memoria.

Pregunta 4:

Los tipos de hazard posible son:

- Hazard estructural: Es aquel hazard que sucede cuando las unidades funcionales del CPU no son suficientes para paralelizar las tareas.
Ej: Si solo se tiene un bus de memoria, no es posible realizar el FETCH y la escritura de un dato a memoria simultáneamente.

- Hazard de control: Esto sucede cuando se altera el flujo de ejecución de un programa y en consecuencia las siguientes instrucciones a ejecutar no se corresponden con las que se estaban procesando en el pipeline, teniendo que descartarlas.
Ej:

```
R1 = R2 + R3  IF  DE  EX  ME  WB
JMP TAG           IF  DE  EX  ME  WB
R9 = R8 + R7          IF  DE  EX  XX
...                               XX
...                               XX
TAG:
R4 = R6 + R5                      XX  IF  DE  EX  ME  WB
```

- Hazard de datos: Este hazard surge cuando dos instrucciones cercanas trabajan con un mismo dato y una instrucción debe esperar a que la otra termine para poder seguir adelante, deteniendo el pipeline.
Ej *(marcada la espera con puntos suspensivos):*

```
R1 = R2 + R3          IF    DE    EX    ME    WB
R4 = R1 + R5                IF    ..    ..    ..    DE    EX    ME    WB
```

Problema 2

Se quiere resolver en forma recursiva la función de McCarthy definida para enteros positivos de la siguiente forma:

$$M(n) = \begin{cases} M(M(n+11)) & \text{for } n \leq 100 \\ n - 10 & \text{for } n > 100. \end{cases}$$

Se pide:

(a) Resolver el problema en forma recursiva en alto nivel y compilar el algoritmo en assembler 8086. Para el pasaje de parámetros debe utilizarse el stack.

(b) Dibujar el árbol de llamadas para n=96 y calcular consumo máximo de stack para esta invocación.

Solución:

a)

```
function mcCarthy(Integer n): Integer;
begin
  if (n <= 100)
     mcCarthy :=  mcCarthy( mcCarthy(n+11))
  else
     mcCarthy := n - 10;
end
```

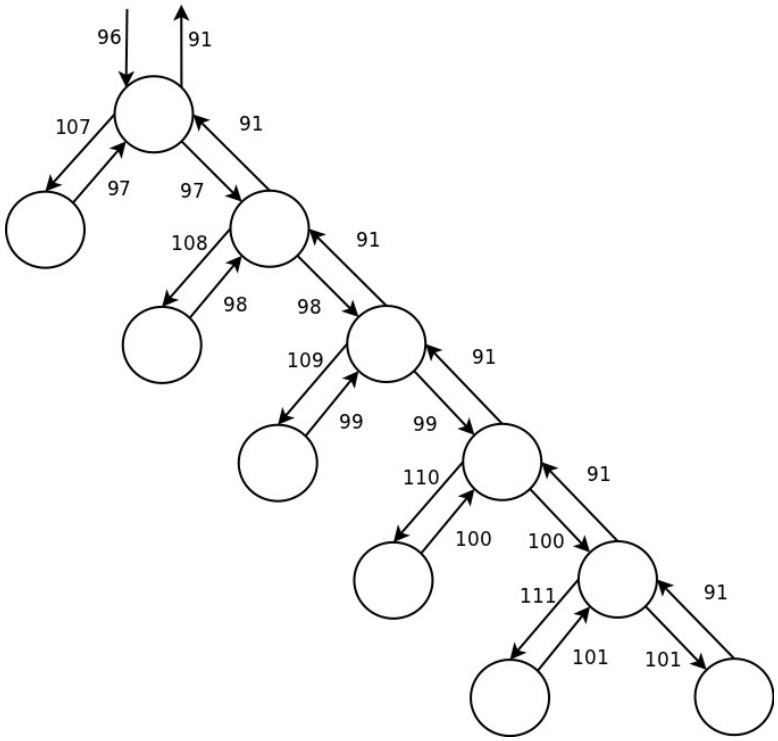
```
mcCarthy proc
  push BP         ; respaldar registros y ajustar BP para acceder a al stack
  mov BP, SP
  push AX

  mov AX, [BP+4]  ; cargar AX con el parametro n
  cmp AX, 100
  ja else
  add AX, 11      ; calcular n+11
  push AX         ; coloco el parametro para la llamada
  call mcCarthy   ; calcular M(n+11) dejando en el
  call mcCarthy   ; calcular M(M(n+11)
  pop AX          ; obtener el resultado del stack
  jmp fin
else:
  sub AX, 10
fin:
  mov [BP+4], AX  ; colocar resultado en el stack en lugar del parametro de entrada
  pop AX          ; restaurar registros
  pop BP
  ret             ; retorno
endp
```

b)

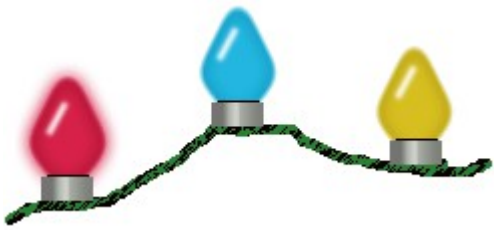
Cada llamada consume ocho bytes (dos bytes parámetro, dos bytes IP, dos bytes registro BP y dos bytes registro AX). En la figura de la derecha se muestra el árbol de llamadas cuando la primera invocación se lleva a cabo con parámetro n=96. En este caso se mantiene simultáneamente la información para seis llamadas.

El consumo máximo para este caso es entonces 48 bytes (8*6 bytes).



Problema 3

La empresa **Rodolfo el Reno S.A.** especializada en artículos navideños desea desarrollar un nuevo sistema innovador en luces navideñas.



luces navideñas

El mismo utiliza luces de tres colores (**rojo**, **verde y azul**) que se encienden y apagan a intervalos regulares de tiempo, de acuerdo a la posición de un **switch** que indica el modo de funcionamiento del kit como puede verse en la siguiente tabla:

Valor del *switch*	Descripción del funcionamiento
0	Todas las luces apagadas
1	Todas las luces encendidas
2	Se encienden secuencialmente Rojo, Verde y Azul durante 1 segundo cada uno
3	Se encienden secuencialmente Rojo, Verde y Azul durante 2 segundos cada uno

El circuito deberá tener 3 salidas (LuzRoja, LuzVerde y LuzAzul). Las luces se encienden colocando un **1** en la señal correspondiente al color que se desea encender y un **0** para apagarla.

Se pide:
Implementar utilizando la metodología vista en el curso el sistema de luces navideñas utilizando flip flops tipo J-K y compuertas básicas. Se dispone de una señal **clk** (reloj) de frecuencia 1Hz.

Nota: la siguiente tabla muestra la tabla de verdad para el flip flops tipo J-K.

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q'_n

Diagrama de estados

Entradas: S0, S1
Salidas, R, V, A

Codificación switch (S_0S_1)

0: 00
1: 01
2: 10
3: 11

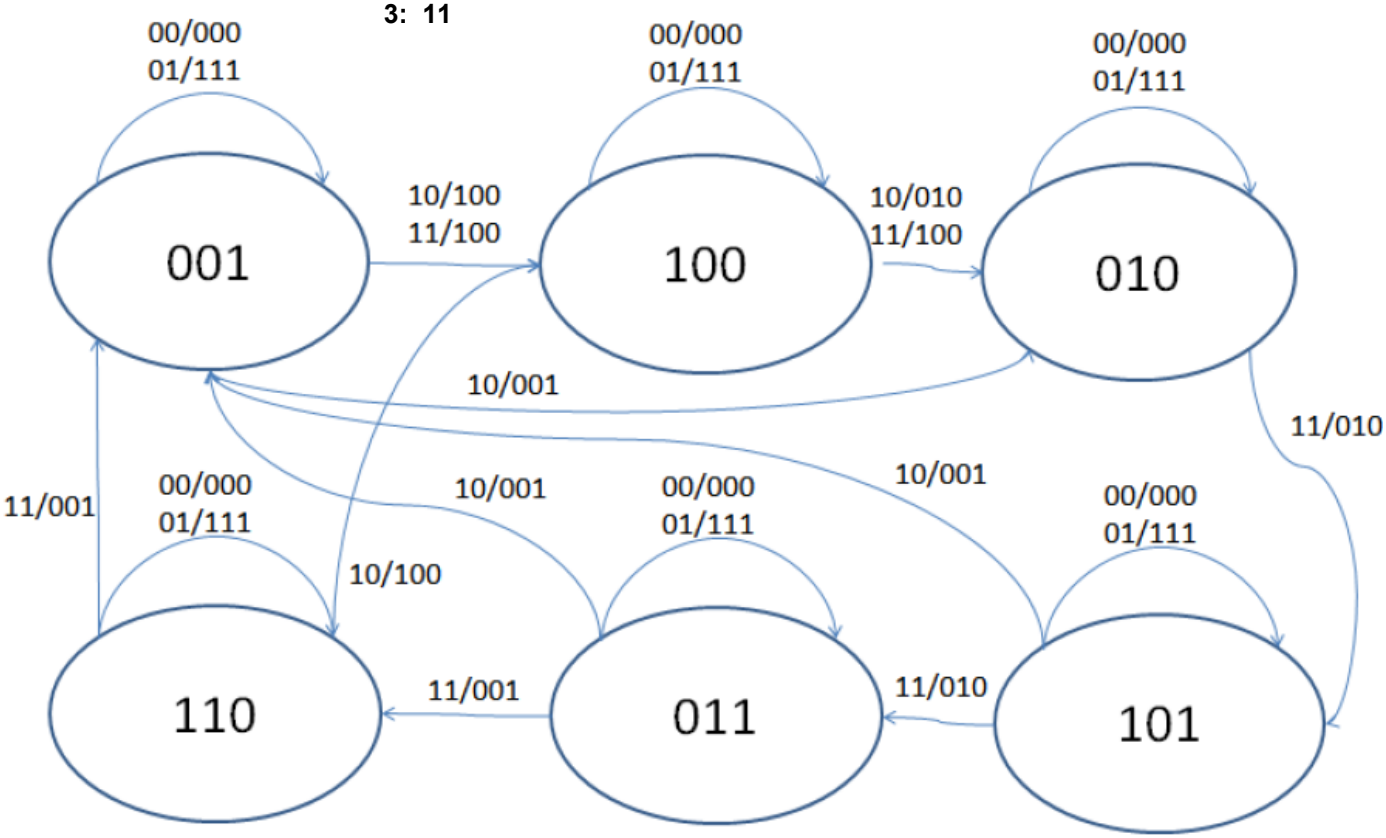


Tabla de estados y transiciones

Estado actual (Qn) Q2Q1Q0	Entradas S0S1	Próximo estado (Qn+1) D2D1D0	J2K2	J1K1	J0K0	Salidas RVA
000	XX	XXX	XX	XX	XX	XXX
001	00	001	0X	0X	1X	000
001	01	001	0X	0X	1X	111
001	10	100	1X	0X	X1	100
001	11	100	1X	0X	X1	100
010	00	010	0X	1X	0X	000
010	01	010	0X	1X	0X	111
010	10	001	0X	X1	1X	001
010	11	101	1X	X1	1X	010
011	00	011	0X	1X	1X	000
011	01	011	0X	1X	1X	111
011	10	001	0X	X1	1X	001
011	11	110	0X	1X	X1	001
100	00	100	1X	0X	0X	000
100	01	100	1X	0X	0X	111
100	10	010	X1	1X	0X	010
100	11	010	X1	1X	0X	100
101	00	101	1X	0X	1X	000
101	01	101	1X	0X	1X	111
101	10	001	X1	0X	1X	001
101	11	011	X1	1X	1X	010
110	00	110	1X	1X	0X	000
110	01	110	1X	1X	0X	111
110	10	100	1X	X1	0X	001
110	11	001	X1	X1	1X	001
111	XX	XXX	XX	XX	XX	XXX

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	0
01	0	0	0	0
11	1	1	X	X
10	1	X	X	1

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	0
01	0	1	0	0
11	1	X	X	X
10	1	X	X	1

J2 = Q2 + Q1'.S1 + Q0'.S1.S0

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	X	1	1	X

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	1	X	X
10	X	1	1	X

K2 = 1

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	0
01	0	0	0	0
11	1	X	X	X
10	0	1	0	0

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	0
01	0	1	0	0
11	1	X	X	X
10	0	1	1	0

J1 = Q2.Q1 + Q2.S1.S0 + Q0'.S1.S0 + Q2.Q0'.S1

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	1	X	X
10	X	X	X	X

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	1	X	X
10	X	X	X	X

K1 = 1

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	1
01	0	1	1	1
11	0	0	X	X
10	0	0	1	1

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	X	1
01	0	1	X	1
11	0	1	X	X
10	0	0	1	1

J0 = Q0 + Q2'.S1

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	X
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	X
01	X	X	1	X
11	X	X	X	X
10	X	X	X	X

K0 = 1

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	0
01	0	0	0	0
11	0	0	X	X
10	0	0	0	0

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	1	1
01	1	0	0	1
11	1	0	X	X
10	1	1	0	1

R = Q1'.S0 + Q2'.Q1'.S1 + Q1'.Q0'.S0

S_0=0

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	0
01	0	0	0	0
11	0	0	X	X
10	0	1	0	0

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	1
01	1	1	0	1
11	1	0	X	X
10	1	0	1	1

V = S1'.S0 + Q2'.Q0'.S0 + Q2.Q0.S0 + Q1'.Q0'.S1.S0'

S_0=0

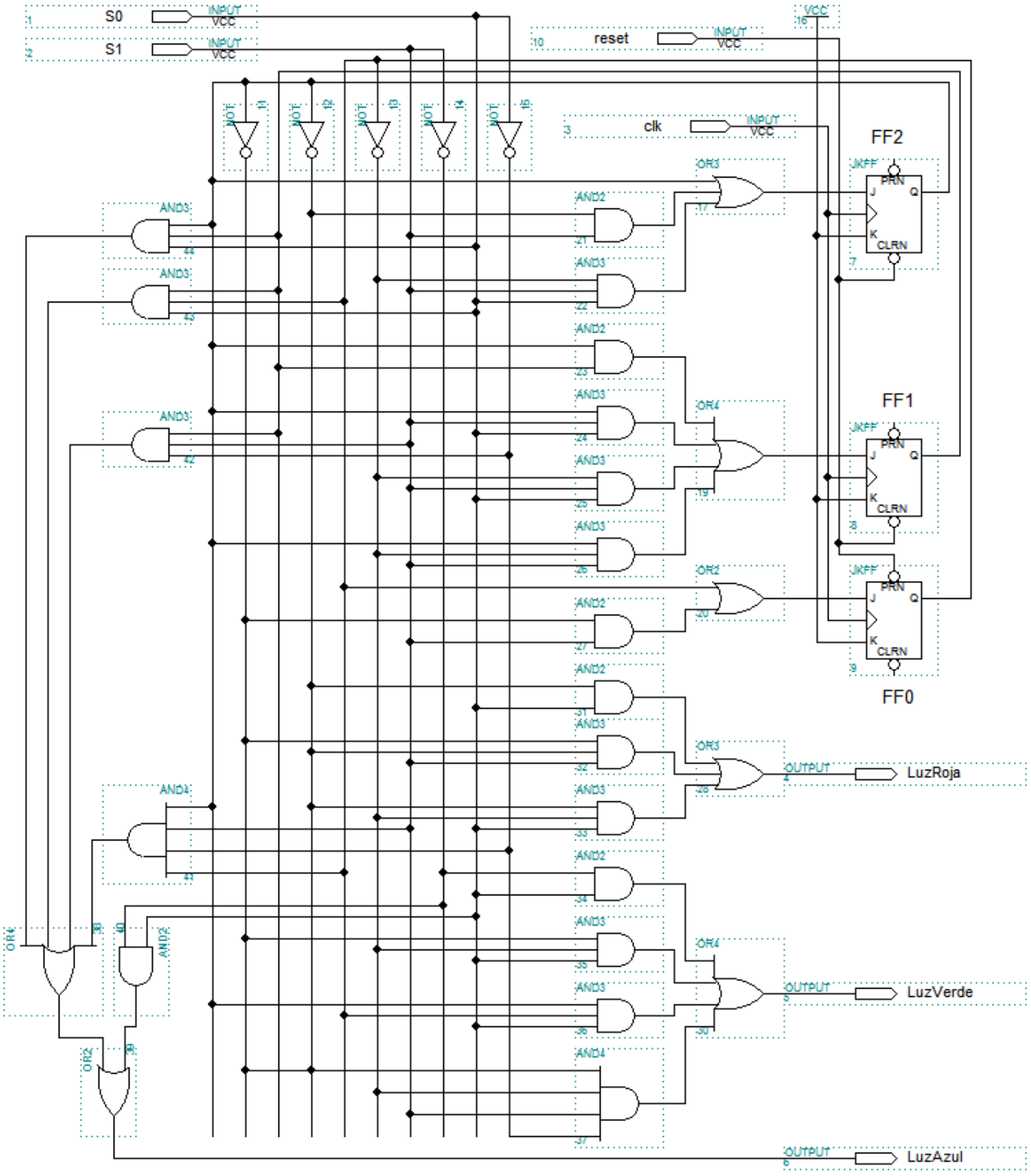
Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	0
01	0	1	1	0
11	0	1	X	X
10	0	0	1	0

S_0=1

Q_2Q_1/Q_0S_1	00	01	11	10
00	X	X	0	1
01	1	0	1	1
11	1	1	X	X
10	1	0	0	1

A = S1'.S0 + Q2.Q1.S0 + Q1.Q0.S0 + Q1.S1.S0' + Q2.Q0.S1.S0'

Circuito:


S0
S1
reset
clk
VCC
INPUT
NOT
AND3
AND2
AND4
OR3
OR4
OR2
JKFF
PRN
CLRN
FF2
FF1
FF0
OUTPUT
LuzRoja
LuzVerde
LuzAzul