Solución del examen de Arquitectura de Computadoras

22 de diciembre del 2012

Instrucciones:

- Indique su nombre, apellido y número de cédula en todas las hojas que entregue.
- Escriba las hojas de un solo lado.
- Las hojas deben estar numeradas y en la primer hoja debe escribirse el total.
- Empiece cada ejercicio en una hoja nueva.
- No puede utilizar material ni calculadora.
- **Apague su celular.**
- La duración del examen es de tres horas. En dicho tiempo debe también completar sus datos.
  Solo se contestarán dudas de letra. No se aceptan preguntas en los últimos 30 minutos del examen.
- Para aprobar debe contestar correctamente un ejercicio entero y dos preguntas.

Pregunta 1

Indique cómo se distribuye en memoria y cuántos bytes ocupa la variable mi_var:

```
// Esto es una declaración de tipo, no de variable. No ocupa memoria.
struct mi_tipo{
    char letras[4];
    short enteros[2];
    char letras_MAYUS[2];
};
// Variable mi_var es de tipo mi_tipo
mi_tipo mi_var[3] ;
```

Pregunta 2

a) Enumere los dos tipos de máquinas de estado vistas en el curso. ¿Cuál es su diferencia fundamental?.

b) Enumere las etapas de la metodología de diseño de circuitos secuenciales como máquinas de estado vista en el curso.

Pregunta 3

A) ¿Qué es la distancia de un sistema de codificación binario? Explique su relación con la posibilidad de detectar y corregir errores.

B) Sea un sistema de codificación binario que codifica en las siguientes palabras:

11011
10110
10101
00011

¿Cuál es la distancia de dicho sistema? ¿Cuántos bits de error se pueden detectar? ¿Cuántos se pueden corregir?

Pregunta 4

Defina los términos *Hit*, *Miss*, *Hit Rate*, *Miss Rate*, *Hit Time*, *Miss Penalty* utilizados al hablar de Jerarquía de Memoria .

Problema 1

Se desea implementar para estas fiestas un sistema que encienda y apague luces de colores en las oficinas en un edificio para que aparente ser un árbol navideño.

Para controlar las luces se utiliza una controladora disponible en el puerto de solo escritura 033h (16 bits). En el byte superior se indica el código de oficina, mientras que en el inferior se indica el color a utilizar para la luz de dicha oficina.

Usted deberá implementar la función `cambiar_luces` que recibe un color y una lista de oficinas y se comunica con la controladora para poner en las luces de todas las oficinas de la lista el color indicado.

Se pide:

Parte A

Implemente en alto nivel la función `void cambiar_luces(unsigned char color, short cantidad_oficinas, unsigned char* lista_oficinas)`. La lista de códigos de oficina se representa como un arreglo de bytes.

Parte B

Ensamble la función `cambiar_luces` para la arquitectura 8086. Recibe en AL el color a utilizar y en el stack la cantidad de elementos en la lista de oficinas y el puntero al inicio de la lista. Debe preservar el valor de los registros y se deben retirar los parámetros del stack. Incluya un ejemplo de invocación.

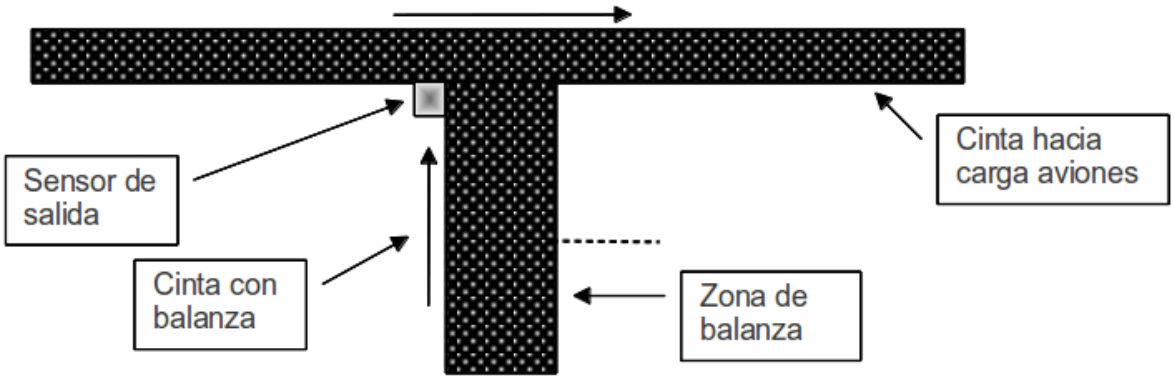
Problema 2

A la empresa **BAGAGGEIN** se le ha encomendado diseñar el sub-sistema de control de las cintas transportadoras utilizadas para realizar el despacho de las maletas en la modalidad de autoservicio en el Aeropuerto Internacional de Carrasco.

El sistema principal le solicita al usuario los datos del vuelo y los datos personales y luego le solicita que presione el botón “Despacho”.

El sub-sistema consta de los siguientes elementos:

- Una cinta con balanza, donde es posible pesar la maleta (la balanza funciona solo si la cinta está detenida) y moverla hacia otra cinta que la llevará hacia el sistema de carga de los aviones. El motor de la cinta con balanza se controla con el bit de salida cinta (1 = encendido).
- Un detector de objetos colocado al final de la cinta con balanza. El estado del sensor se obtiene en el bit de entada sensor (1 = hay un objeto pasando frente al sensor).
- Una balanza que da el peso (en Kg) del objeto colocado sobre la cinta con balanza en cuatro bits de entrada p3 p2 p1 p0 (p0 es el menos significativo).
- Un botón (“Despacho”) para iniciar el proceso de peso y despacho automático de la maleta, accesible como un bit de entrada despacho (1 = botón apretado).
- Un letrero luminoso con el texto “Peso Excesivo” que se enciende con el bit de salida letrero (1 = encendido).



El sistema debe funcionar de la siguiente manera: cuando el usuario coloca la maleta sobre la cinta detenida y pulsa el botón de despacho, en caso que el peso de la maleta sea menor o igual a 10 Kg deberá activar la cinta con balanza hasta expulsar totalmente la maleta de la misma. En caso que el peso sea superior el sistema deberá encender el letrero luminoso hasta que el usuario retire la maleta y por tanto el peso deje de ser mayor a 10 Kg.

Se pide:

Construir el circuito lógico que controla los bits de salida. Se dispone de compuertas lógicas y flip-flops tipo D con reset asincrónico.

Nota: se sugiere comenzar por construir un comparador de 4 bits y luego diseñar un circuito secuencial siguiendo la metodología del curso.

Solución

Pregunta 1

Un short son 2 bytes, mientras que cualquier char es 1 byte. Los arreglos se representan poniendo un elemento a continuación del otro. Dado que en este struct todos los elementos ocupan una cantidad par de bytes, se ponen todos de forma contigua.

Cada elemento de la tabla identifica un byte, las direcciones crecen hacia abajo.

mi_var[0].letras[0]
mi_var[0].letras[1]
mi_var[0].letras[2]
mi_var[0].letras[3]
mi_var[0].enteros[0](parte baja)
mi_var[0].enteros[0](parte alta)
mi_var[0].enteros[1](parte baja)
mi_var[0].enteros[1](parte alta)
mi_var[0].letras_MAYUS[0]
mi_var[0].letras_MAYUS[1]
mi_var[1].letras[0]
....

Se repite análogamente para mi_var[1] y mi_var[2].

Un elemento del tipo mi_var ocupa entonces 10 bytes (4 * 1 + 2 * 2 + 2 * 1 = 10 bytes). La variable mi_var que es un arreglo de 3 elementos de este tipo, ocupa entonces **30 bytes**.

Pregunta 2

A) Las dos máquinas de estado vistas en el curso son:

- Máquina de Mealy.
- Máquina de Moore.

La diferencia fundamental entre ambas es que en la máquina de *Mealy* la salida del circuito depende del estado actual y de la entrada actual, mientras que en la máquina de Moore la salida del circuito depende solamente del estado actual.

B)

- Modelado del sistema a implementar mediante un Diagrama de Estados.
- Deducir la "Tabla de Estados" a partir del Diagrama de Estados.
- Sistema de codificación para estados, entrada(s) y salida(s).
- Determinar el número de flip-flops necesarios.
- Usar la codificación de los estados, entradas y salidas para pasar de la "Tabla de Estados" a la "Tabla de Transiciones y Salidas".
- Con la ecuacion de los flip-flops pasar de la "Tabla de Transiciones y Salidas" a las tablas de verdad de las funciones lógicas que permitirán determinar el valor de la(s) salida(s), y el valor a presentar en las entradas de los flip-flops para almacenar el nuevo estado en ellos.
- Minimizar las expresiones lógicas en dos niveles mediante Diagramas de Karnaugh.
- Dibujar el circuito.

Pregunta 3

A) La distancia de un sistema de codificación binario es el mínimo natural N tal que dos palabras cualquiera del mismo, difieren en al menos N bits.

Para poder detectar un número m de errores, la distancia del sistema de codificación debe ser al menos m + 1. Para poder corregirlos, la misma debe ser de al menos 2m + 1.
B) Por comparación de las palabras una a una, se ve que todas difieren al menos en 2 bits. Como además la distancia entre 10101 y 10110 es 2, la distancia del sistema es 2.
Por la parte anterior, con este sistema se pueden detectar errores de un bit. No se pueden corregir errores con este sistema.

Pregunta 4

Hit : Se dice que ocurre un hit cuando un objeto de información se encuentra en el lugar de la jerarquía donde se lo está buscando.
Miss : Se dice que ocurre un miss cuando el elemento de información no es encontrado en el lugar de la jerarquía donde se lo está buscando. En este caso es necesario ir a buscar el objeto a un nivel de jerarquía inferior.
Hit Rate : Es la tasa de acierto de encontrar un elemento de información en el lugar de la jerarquía en que se lo busca.
Miss Rate : Es la tasa de fallos en encontrar un elemento de información en el lugar buscado (y coincide con 1 – Hit Rate).
Hit Time : Tiempo de acceso promedio en el nivel de jerarquía considerado (donde se da el hit).
Miss Penalty : Tiempo de acceso promedio adicional requerido para acceder al elemento de información en el nivel de jerarquía inferior. Típicamente ocurre que Hit Time << Miss Penalty.

Problema 1

Parte A:

```
void cambiar_luces(unsigned char color, short cantidad_oficinas, unsigned char* lista_oficinas){
      for(short i = 0; i< cantidad_oficinas; i++){
            out(033h, (lista_oficinas[i] << 8 | color));
      }
}
```

Parte B:

```
Ejemplo invocación:
      mov AL, color
      push color
      push puntero
      call cambiar_luces
      ----------------
      ----------------


proc cambiar_luces
      push BP
      mov  BP, SP
      push AX
      push BX
      push CX
      push SI
      mov BX, [BP+4]
      mov CX, [BP+6]
      ; Se ubica el ip para el retorno
      mov SI, [BP+2]
      mov [BP+6], SI

      mov SI, 0

for:
      cmp CX, SI
      je fin

      mov AH, [BX+SI]
      ; El color ya está en AL
```

```
      out 033h, AX

      add SI, 1
      jmp for

fin:

      pop SI
      pop CX
      pop BX
      pop AX
      pop BP
      ; Se descarta el parámetro adicional y se deja a SP apuntando a IP para el retorno
      add SP, 4

      ret
```

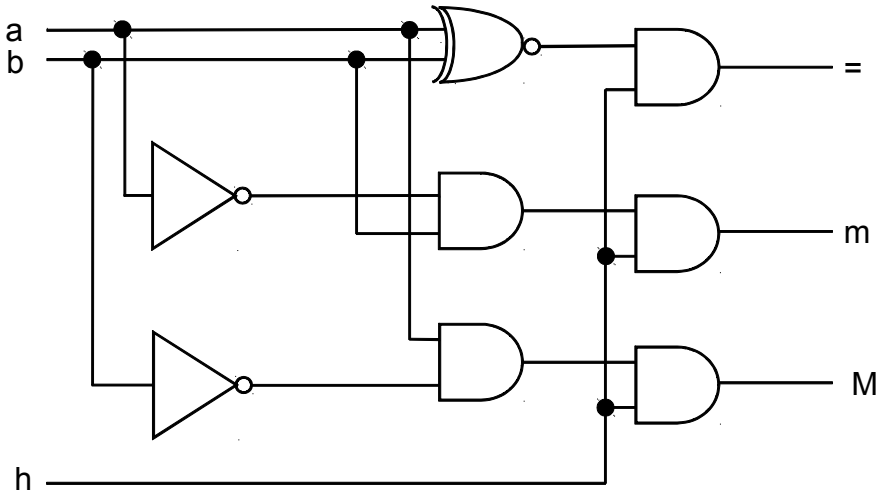
Problema 2

Solución:

Para construir el comparador de 4 bits comenzaremos por uno de 1 bit (compara **a** con **b**) con entrada de habilitación (pensando en ponerlos luego en cascada):

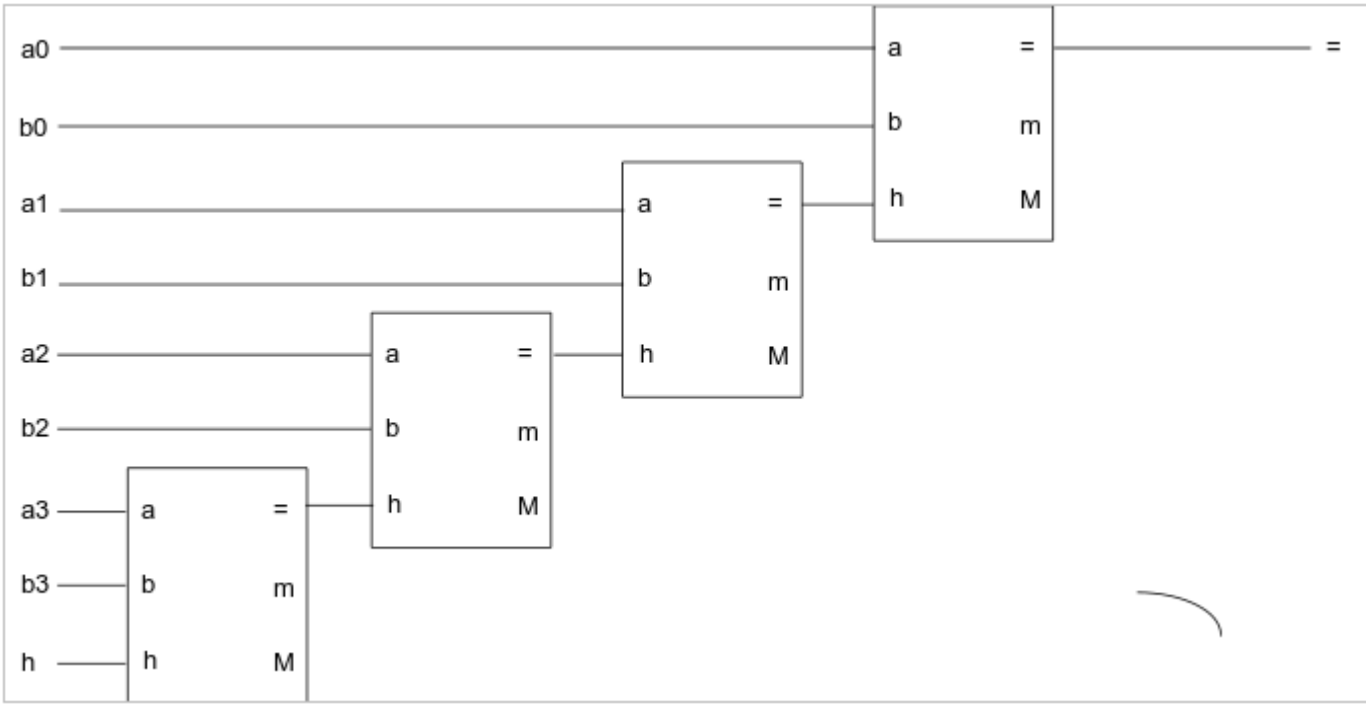
a	b	M	m	=
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

"**=**" es $(a \oplus b)'$
"**m**" es a' . b
"**M**" es a . b'

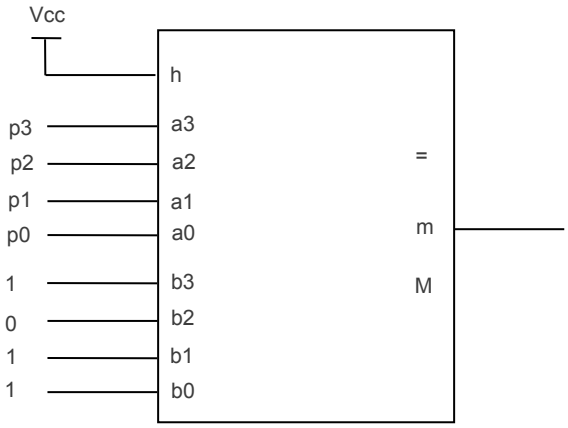


Para conseguir el de 4 bits ponemos en cascada del bit mas significativo al menos significativo, encadenando la salida de igual ("=") a la entrada de habilitación. La ultima salida de "=" es la salida del comparador completo y las salidas de "m" y "M" se logran con el OR de las de cada bit.

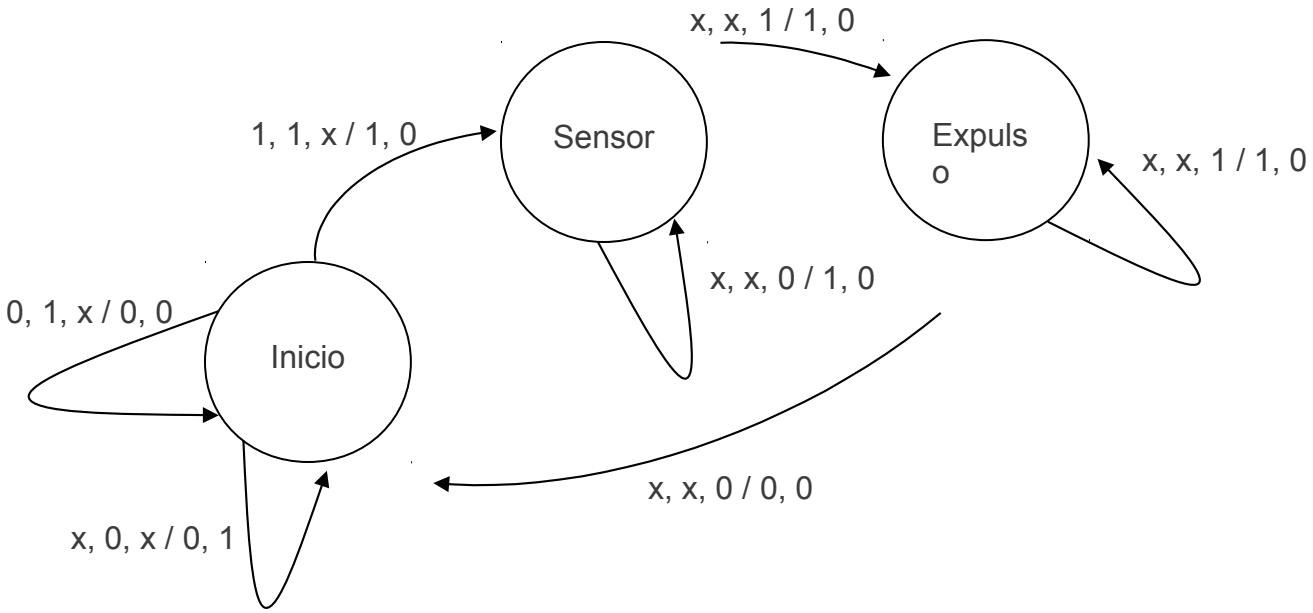
El circuito del comparador queda como sigue:



Con el comparador realizo la comparación del peso (p3 p2 p1 p0) con el número once (1011) y utilizo la salida de “menor” (m).



Construyamos una máquina de estados cuyas entradas sean **d** (despacho), **m** (salida del comparador de peso) y **s** (sensor) y cuyas salidas sean **c** (cinta) y **l** (letrero).



En base a este diagrama de estados construimos la tabla de estados:

	Entradas			Prox. Estado	Salidas	
Estado	d	m	s		c	l
Inicio	0	0	0	Inicio	0	1
	0	0	1	Inicio	0	1
	0	1	0	Inicio	0	0
	0	1	1	Inicio	0	0
	1	0	0	Inicio	0	1
	1	0	1	Inicio	0	1
	1	1	0	Sensor	1	0
	1	1	1	Sensor	1	0
Sensor	0	0	0	Sensor	1	0
	0	0	1	Expulso	1	0
	0	1	0	Sensor	1	0
	0	1	1	Expulso	1	0
	1	0	0	Sensor	1	0
	1	0	1	Expulso	1	0
	1	1	0	Sensor	1	0
	1	1	1	Expulso	1	0
Expulso	0	0	0	Inicio	0	0
	0	0	1	Expulso	1	0
	0	1	0	Inicio	0	0
	0	1	1	Expulso	1	0
	1	0	0	Inicio	0	0
	1	0	1	Expulso	1	0
	1	1	0	Inicio	0	0
	1	1	1	Expulso	1	0

Codificando los estados como:

Inicio	00
Sensor	01
Expulso	10

Y usando la ecuación de los FF tipo D, pasamos a la tabla de transiciones y salidas:

Estado		Entradas			Prox. Est.		Salidas	
e1	e0	d	m	s	d1	d0	c	l
0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	1	0	0	0	0	0	1
0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	1	1	0
0	1	0	0	0	0	1	1	0
0	1	0	0	1	1	0	1	0
0	1	0	1	0	0	1	1	0
0	1	0	1	1	1	0	1	0
0	1	1	0	0	0	1	1	0
0	1	1	0	1	1	0	1	0
0	1	1	1	0	0	1	1	0
0	1	1	1	1	1	0	1	0
1	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1	0
1	0	0	1	0	0	0	0	0
1	0	0	1	1	1	0	1	0
1	0	1	0	0	0	0	0	0
1	0	1	0	1	1	0	1	0
1	0	1	1	0	0	0	0	0
1	0	1	1	1	1	0	1	0
1	1	0	0	0	X	X	X	X
1	1	0	0	1	X	X	X	X
1	1	0	1	0	X	X	X	X
1	1	0	1	1	X	X	X	X
1	1	1	0	0	X	X	X	X
1	1	1	0	1	X	X	X	X
1	1	1	1	0	X	X	X	X
1	1	1	1	1	X	X	X	X

Los diagramas de Karnaugh correspondientes resultan:

d1

e1e0\dm	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	0	0

s=0

e1e0\dm	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	1	1	1	1

s=1

d1 = s.e0 + s.e1

d0

e1e0\dm	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	X	X	X	X
10	0	0	0	0

e1e0\dm	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	X	X	X	X
10	0	0	0	0

s=0 s=0

d0 = e0. !s + !e1.!e0.d.m

c

e1e0\dm	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	X	X	X	X
10	0	0	0	0

s=0

e1e0\dm	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	1	1

s=1

c = e0 + s.e1 + !e1.d.m

l

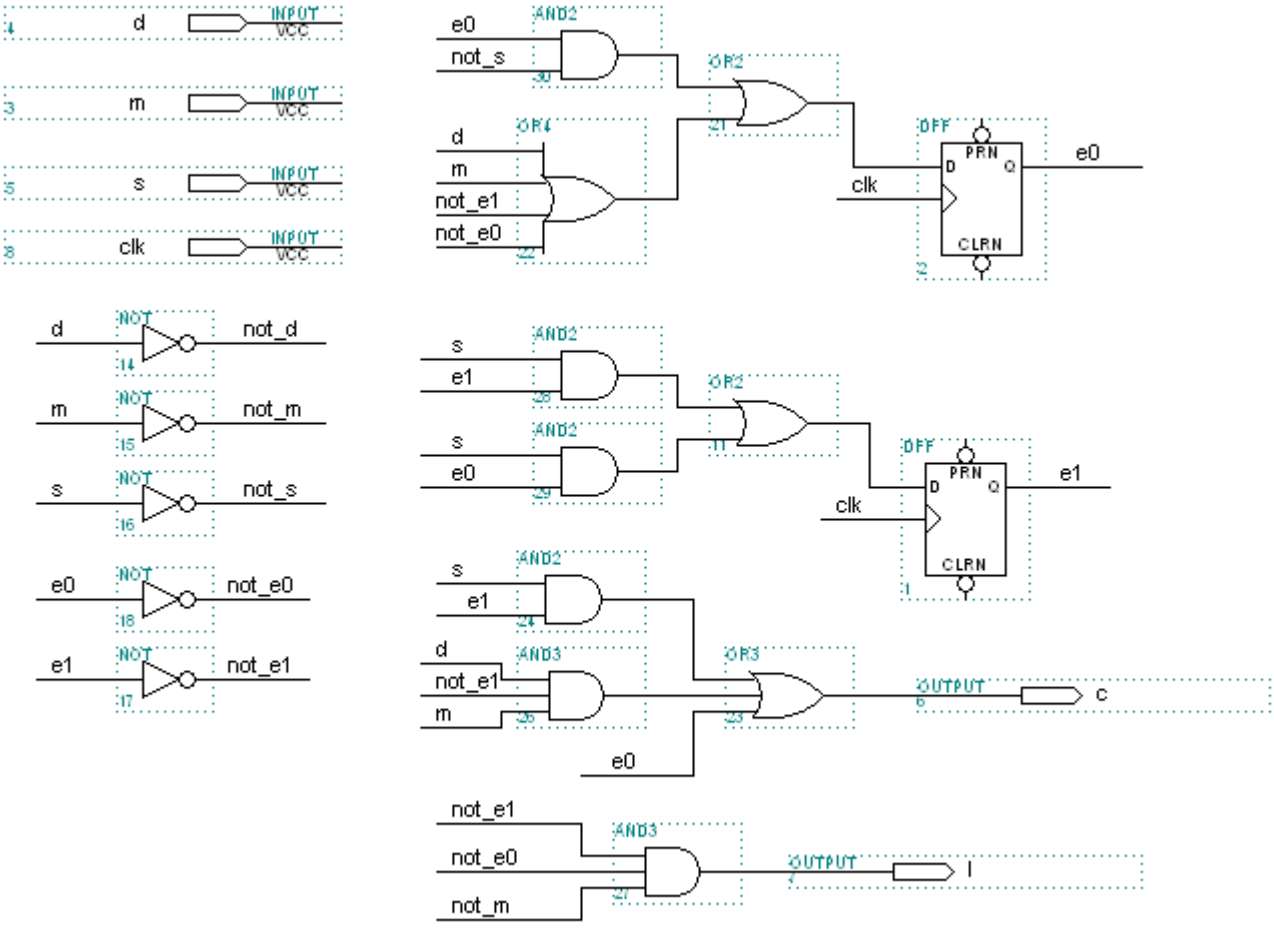
e1e0\dm	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	X	X	X	X
10	0	0	0	0

s=0

e1e0\dm	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	X	X	X	X
10	0	0	0	0

s=1

l = !e1. !e0. !m


d
m
s
clk
INPUT
VCC
NOT
not_d
not_m
not_s
not_e0
not_e1
e0
not_s
AND2
OR2
d
m
not_e1
not_e0
AND4
DFF
PRN
CLRN
clk
e0
s
e1
e1
AND3
OR3
OUTPUT
c
not_m
l