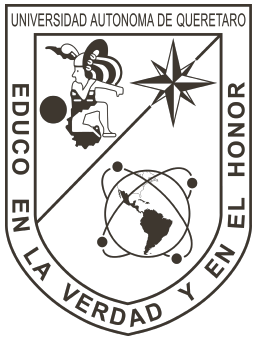
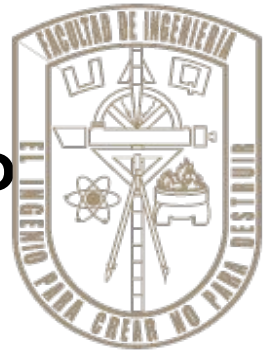


Universidad Autónoma de Querétaro

Facultad de Ingeniería

Ingeniería en Automatización



Sistemas Digitales con Lógica Reconfigurable

M en C. Marcos Romo Avilés

Practica 1: Test bench

A	B	C	$A + (B \cdot C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A	B	C	D	$(C \cdot D) + (A \cdot B)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

A	B	C	D	$A \cdot B \cdot C \cdot D$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Practica 1: Test bench

A	B	C	D	$C \cdot (A+B) \cdot D$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

A	B	C	D	$A+B+C+D$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Practica 1: Test bench

- **Librerías y uso:**

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

Estas líneas indican que el testbench usa la biblioteca IEEE y específicamente el paquete STD_LOGIC_1164, que proporciona definiciones para manejar señales lógicas estándar como '0', '1', 'Z' (alta impedancia), etc.

- **Declaración de la entidad:**

```
entity LogicFunctions_tb is
-- El testbench no tiene puertos.
end LogicFunctions_tb;
```

Aquí se declara la entidad del testbench, LogicFunctions_tb. En los testbenches, normalmente no hay puertos porque sirven para simular el entorno externo de la entidad a probar.

Practica 1: Test bench

- **Declaración de la arquitectura:**

```
architecture Behavioral of LogicFunctions_tb is
```

Esta línea inicia la definición de la arquitectura Behavioral para LogicFunctions_tb.

- **Componente a probar:**

Dentro de la arquitectura, se declara un componente llamado LogicFunctions, que es la entidad que se va a probar.

```
component LogicFunctions
        Port (
            A : in STD_LOGIC;
            B : in STD_LOGIC;
            C : in STD_LOGIC;
            D : in STD_LOGIC;
            F1 : out STD_LOGIC;
            F2 : out STD_LOGIC;
            F3 : out STD_LOGIC;
            F4 : out STD_LOGIC;
            F5 : out STD_LOGIC
        );
end component
```

Practica 1: Test bench

- **Señales para conectar con el componente:**
- Se definen señales A, B, C, D y F1 a F5. Estas señales se usarán para conectar con los puertos del componente LogicFunctions.

```
signal A, B, C, D : STD_LOGIC := '0';
signal F1, F2, F3, F4, F5 : STD_LOGIC;
```

- **Instanciación del componente:**

```
begin
uut: LogicFunctions Port Map (...);
```

- Aquí, se crea una instancia del componente LogicFunctions (a menudo denominada UUT, Unidad Bajo Prueba) y se mapean las señales del testbench a los puertos del componente.

Practica 1: Test bench

- **Proceso de estímulo:**

```
 stimulus: process
    begin
    A <= '0'; B <= '0'; C <= '0'; D <= '0';
        wait for 10 ns;
        A <= '1'; B <= '0'; C <= '0'; D <= '0';
        wait for 10 ns;
        A <= '0'; B <= '1'; C <= '0'; D <= '0';
        wait for 10 ns;
        A <= '1'; B <= '1'; C <= '0'; D <= '0';
        wait for 10 ns;
        -- Termina la simulación
        wait;
    end process stimulus
end Behavioral;
```

Este es un proceso que define cómo cambiarán las señales de entrada (A, B, C, D) con el tiempo. En este caso, se aplican diferentes combinaciones de entradas y se espera un tiempo determinado (wait for 10 ns;) para observar cómo reaccionan las salidas (F1 a F5).

Practica 2: Introducción

Minterminos

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\sum(A * B * C)$$

Maxterminos

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\prod(A + B + C)$$

Practica 2: Introducción

Minterminos

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$000 + 010 + 100 + 110 + 111$$

$$A'B'C' + A'BC' + AB'C' + ABC' + ABC$$

Maxterminos

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$(0 + 0 + 1) * (0 + 1 + 1) * (1 + 0 + 1)$$

$$(A + B + C') * (A + B' + C') * (A' + B + C')$$

Practica 2: Introducción

PROPIEDADES	Conmutativa	$x+y = y+x$	$x \cdot y = y \cdot x$
	Elemento neutro	$0+x = x$	$1 \cdot x = x$
	Distributiva	$x \cdot (y+z) = (x \cdot y)+(y \cdot z)$	$x+(y \cdot z) = (x+y)(x+z)$
	Asociativa	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	$x+(y+z) = (x+y)+z$
	Complementario	$x+ \overline{x} = 1$	$x \cdot \overline{x} = 0$
TEOREMAS	Idempotencia	$x+x = x$	$x \cdot x = x$
	Identidad	$x+1 = 1$	$x \cdot 0 = 0$
	Absorción	$x+x \cdot y = x$	$x \cdot (x+y) = x$
	DeMorgan	$\overline{(x+y)} = \overline{x} \cdot \overline{y}$	$\overline{(x \cdot y)} = \overline{x} + \overline{y}$

Practica 2: Introducción

- **Aplicar la ley de distribución:**

$$\overline{ABC} + \bar{A}B\bar{C} + A\overline{BC} + AB\bar{C} + ABC = (\overline{AB} + \bar{A}B + A\bar{B} + AB)\bar{C} + ABC$$

- **Simplificar usando la ley de complemento y la ley de identidad**

$$\overline{AB} + \bar{A}B + A\bar{B} + AB$$

$$\bar{A}(\bar{B} + B) + A(\bar{B} + B)$$

$$\bar{A}(1) + A(1) = 1$$

$$\bar{C}$$

Incluir el término restante

$$\bar{C} + ABC$$

Practica 2: Introducción

- **Aplicar la ley de distribución:**

$$\overline{ABC} + \bar{A}B\bar{C} + A\overline{BC} + AB\bar{C} + ABC = (\overline{AB} + \bar{A}B + A\bar{B} + AB)\bar{C} + ABC$$

- **Simplificar usando la ley de complemento y la ley de identidad**

$$\overline{AB} + \bar{A}B + A\bar{B} + AB$$

$$\bar{A}(\bar{B} + B) + A(\bar{B} + B)$$

$$\bar{A}(1) + A(1) = 1$$

$$\bar{C}$$

Incluir el término restante

$$\bar{C} + ABC$$

Practica 2

- **Simplifica y simula:**

$$(A + B + C') * (A + B' + C') * (A' + B + C')$$

$$AB + C'$$

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Practica 2

- Supongamos que estás diseñando un sistema de alarma para un coche, que se activa en función de tres factores: si el coche está cerrado (C), si hay movimiento cerca del coche (M), y si el sistema de alarma está activado (A). La alarma debe sonar si:
  - El coche está cerrado, el sistema de alarma está activado y hay movimiento detectado.
  - El coche está cerrado y el sistema de alarma está activado, pero no hay movimiento detectado. Esto es para simular una situación en la que se activa la alarma de manera preventiva.

- Genera la tabla de verdad
- Encuentra la ecuación
- Simula la respuesta

Practica 2

C	M	A	Alarma
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- El coche está cerrado, el sistema de alarma está activado y hay movimiento detectado.
- El coche está cerrado y el sistema de alarma está activado, pero no hay movimiento detectado. Esto es para simular una situación en la que se activa la alarma de manera preventiva.

$$F = CM'A + CMA$$

Practica 2

- Imagina que estás diseñando un sistema de control de iluminación inteligente para una habitación que depende de tres factores: sensor de presencia en la habitación (P), nivel de luz exterior (L) y si es horario nocturno (N). El sistema debe controlar la iluminación bajo las siguientes condiciones:
  - La luz debe encenderse si es de noche (N) y hay alguien en la habitación (P), independientemente del nivel de luz exterior.
  - Si es de día (no N), la luz debe encenderse solo si hay alguien en la habitación (P) y el nivel de luz exterior es bajo (L).
  - Si no hay nadie en la habitación (no P), la luz debe estar apagada, independientemente de las otras condiciones.
- Genera la tabla de verdad
- Encuentra la ecuación
- Simula la respuesta

Practica 2

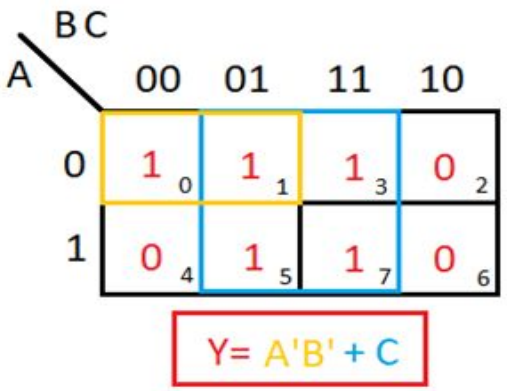
P	L	N	Luz
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- La luz debe encenderse si es de noche (N) y hay alguien en la habitación (P), independientemente del nivel de luz exterior.
- Si es de día (no N), la luz debe encenderse solo si hay alguien en la habitación (P) y el nivel de luz exterior es bajo (L).
- Si no hay nadie en la habitación (no P), la luz debe estar apagada, independientemente de las otras condiciones.

$$F = PL'N + PLN' + PLN$$

Practica 2: mapas de karnaugh

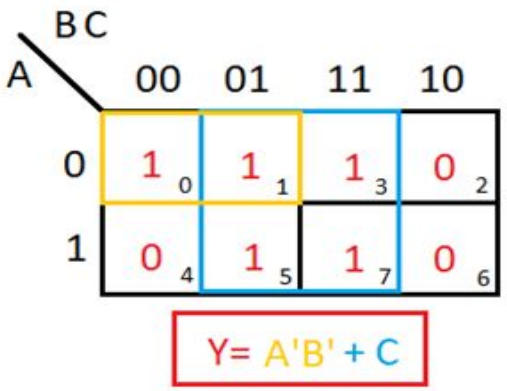
A	B	C	Y	Min
0	0	0	1	A'B'C'
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	1	A'BC
1	0	0	0	
1	0	1	1	AB'C
1	1	0	0	
1	1	1	1	ABC



- **Dibujar el mapa de Karnaugh**
  - Crea una tabla con celdas que representen todas las combinaciones posibles de las variables de entrada. Para dos variables, el mapa es de 2x2, para tres variables, es de 2x4 o 4x2, para cuatro variables, es de 4x4, y así sucesivamente.

Practica 2: mapas de karnaugh

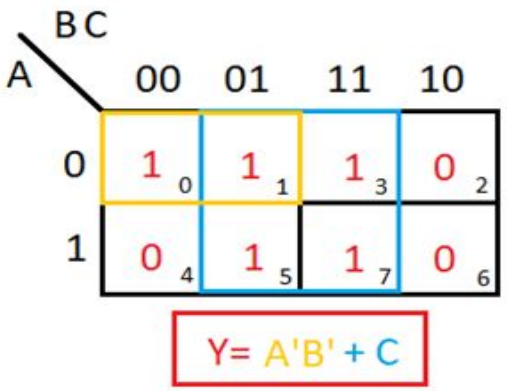
A	B	C	Y	Min
0	0	0	1	A'B'C'
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	1	A'BC
1	0	0	0	
1	0	1	1	AB'C
1	1	0	0	
1	1	1	1	ABC



- **Dibujar el mapa de Karnaugh**
  - Etiqueta las filas y columnas del mapa de manera que cada celda represente una combinación única de las variables de entrada. Asegúrate de que las etiquetas sean adyacentes en términos de una diferencia de un solo bit (siguiendo la secuencia de Gray).

Practica 2: mapas de karnaugh

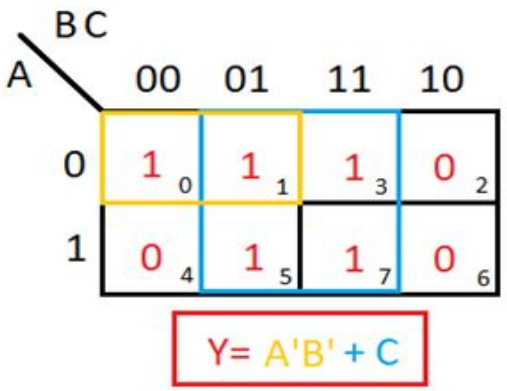
A	B	C	Y	Min
0	0	0	1	A'B'C'
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	1	A'BC
1	0	0	0	
1	0	1	1	AB'C
1	1	0	0	
1	1	1	1	ABC



- **Llenar el mapa de Karnaugh**
  - Rellena el mapa con los valores de la función de salida de la tabla de verdad. Coloca un '1' en las celdas donde la función es verdadera y un '0' donde es falsa.

Practica 2: mapas de karnaugh

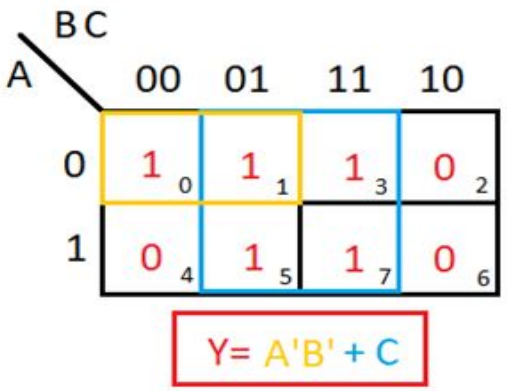
A	B	C	Y	Min
0	0	0	1	A'B'C'
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	1	A'BC
1	0	0	0	
1	0	1	1	AB'C
1	1	0	0	
1	1	1	1	ABC



- **Identificar grupos**
  - Busca grupos de 1s en el mapa. Estos grupos deben ser rectángulos o cuadrados y pueden contener 1, 2, 4, 8 (o cualquier potencia de 2) celdas. Pueden envolver los bordes del mapa.
  - El objetivo es cubrir todos los 1s con el menor número de grupos posibles.

Practica 2: mapas de karnaugh

A	B	C	Y	Min
0	0	0	1	A'B'C'
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	1	A'BC
1	0	0	0	
1	0	1	1	AB'C
1	1	0	0	
1	1	1	1	ABC



- **Escribir la expresión simplificada**
  - Para cada grupo, escribe un término de producto (AND) que represente las variables comunes a todas las celdas del grupo. Si una variable cambia dentro de un grupo, no se incluye. Si una variable es 0 en todas las celdas del grupo, se incluye en su forma complementada.
  - La expresión simplificada de la función es la suma (OR) de todos estos términos de producto.

Practica 2: mapas de karnaugh

- Imagina que estás diseñando un sistema para monitorizar las condiciones ambientales en un data center, utilizando cuatro variables para determinar si se debe activar un sistema de respaldo. Las variables son:
  - Temperatura (T): Normal (0) o Alta (1).
  - Humedad (H): Normal (0) o Alta (1).
  - Fluctuación de energía (F): Sin fluctuación (0) o Con fluctuación (1).
  - Estado del servidor (S): Operativo (0) o Error (1).
- El sistema de respaldo debe activarse bajo las siguientes condiciones complejas:
- Si hay una fluctuación de energía y la temperatura es alta, independientemente de las otras variables.
- Si el servidor presenta un error y hay alta humedad, independientemente de las otras variables.
- Si hay una fluctuación de energía y el servidor presenta un error, pero la temperatura es normal.
- Si la temperatura y la humedad son altas, pero no hay fluctuación de energía y el servidor está operativo.

Practica 2: mapas de karnaugh

T	H	F	S	Resultado
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TH/FS	00	01	11	10
00	0	0	1	0
01	0	1	1	0
11	1	1	1	1
10	0	1	0	1

$$TH + FS + HS + TS$$