Arquitecturas y Organización de Computadoras I

2: Análisis de circuitos digitales

Rafael Ignacio Zurita

Depto. Ingeniería de Computadoras

September 21, 2020

Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

» Señales digitales

* La electrónica interna de un computador actual utiliza componentes analógicos (como los transistores en un circuito cmos).

* De todas maneras se modela de manera digital para operar con dos niveles de voltaje: un voltaje alto y un voltaje bajo

* El resto de los valores de voltaje son temporales y ocurren durante la transición entre los valores alto y bajo o bajo y alto

» Señales digitales

* Esta es una razón clave por la que los computadores utilizan números binarios, ya que un sistema binario se corresponde directamente con la abstracción subyacente a la electrónica
* En las diferentes implementaciones electrónicas los voltajes y sus relaciones difieren,
* Por lo que no se utiliza el valor del voltage sino una indicación de si la señal esta activada o no (verdadera o no, 1 o 0, etc).

» Señales digitales

- Por eso hablamos de señales que son:
  - (lógicamente) ciertas, ó 1, ó afirmadas, asertadas
  - (lógicamente) falsas, ó 0, ó negadas
- Los valores 0 y 1 reciben el nombre de complementarios o inversos el uno del otro

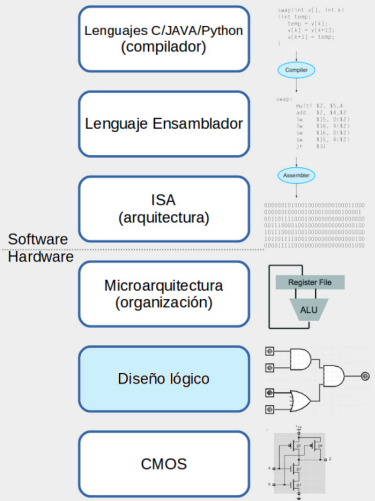
Diseño Lógico - Diseño Digital

- Señales digitales
- **Análisis de Circuitos Digitales**
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

» Diseño lógico o digital

Los bloques lógicos (circuitos digitales o lógicos) se clasifican en dos tipos:

- **circuitos combinacionales**
  - Sus salidas dependen sólo de las entradas
- **circuitos secuenciales**
  - Mantienen un estado interno. Sus salidas pueden depender tanto de las entradas actuales como del valor almacenado en memoria, conocido como estado del bloque
  - Permiten modelar memorias (**registros**) y **máquinas de estados finitos**
  - Las máquinas de estado finito permiten modelar **máquinas algoritmicas** (por ej. una CPU)



Diseño Lógico - Diseño Digital

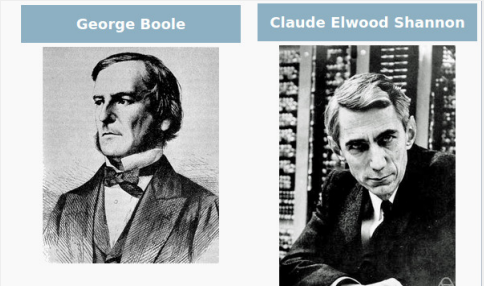
- Señales digitales
- Análisis de Circuitos Digitales
- **Algebra de Boole**
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

» Análisis de Circuitos Digitales

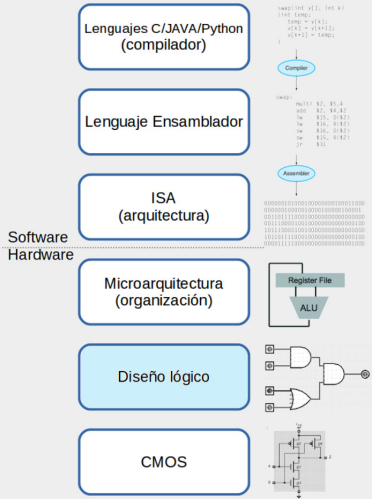
Orígenes de la teoría de análisis y diseño de circuitos lógicos o digitales:

Booleano En 1854 George Boole define un algebra que utiliza sólo dos valores.

Switching En 1938 Claude Shannon demuestra que el algebra de boole puede ser utilizado para analizar y modelar circuitos digitales.



Diseño lógico o digital



» Circuitos Combinacionales

Circuito combinacional: La función lógica (salida) depende únicamente de las señales de entrada.

Tablas de Verdad

- Debido a que un bloque de lógica combinatoria no contiene memoria, puede especificarse completamente definiendo los valores de las salidas para cada posible conjunto de valores de entrada
- Dicha descripción se da normalmente en forma de tabla de verdad

Tablas de Verdad

- Para un bloque lógico con n entradas, existen 2 n posiciones en la tabla de verdad, puesto que este es el número de combinaciones posible de los valores de entrada
- Cada posición en la tabla especifica el valor de todas las salidas para una combinación particular de las entrada
- Las tablas de verdad pueden **describir completamente cualquier función lógica** combinatoria
- Sin embargo, su tamaño **crece rápidamente** y puede dificultar su comprensión

Algebra de Boole

- Todas la variable tienen valores 0 ó 1
- Existen tres operadores:
  - **OR** , se escribe $+$, como en A $+$B . El resultado es 1 si alguna de la variables de entrada es 1. También se conoce como suma lógica
  - **AND** , se escribe . , como en A.B . El resultado es 1 sólo si ambas entradas son 1. También se conoce como producto lógico
  - **NOT** , se escribe . El resultado es 1 sólo si la entrada es 0. La aplicación del operador NOT a un valor lógico resulta en una inversión o negación de dicho valor

Álgebra de Boole - Leyes y Teoremas

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Álgebra de Boole - Leyes y Teoremas

- Cualquier función lógica puede ser reescrita como una ecuación, con la salida en la parte izquierda de la igualdad, y una función de las variables de entrada utilizando las operaciones del álgebra a la derecha.
- Ejemplo:

$$E = ((A \cdot B) + (A \cdot C) + (B \cdot C)) \cdot (\overline{A \cdot B \cdot C})$$

We can also derive E by realizing that E is true only if exactly two of the inputs are true. Then we can write E as an OR of the three possible terms that have two true inputs and one false input:

$$E = (A \cdot B \cdot \overline{C}) + (A \cdot C \cdot \overline{B}) + (B \cdot C \cdot \overline{A})$$

Compuertas/Puertas (Gates)

- Los bloques lógicos se construyen a partir de compuertas (puertas) lógicas que realizan las funciones lógicas básicas como AND, OR y NOT
- Una puerta AND o una OR pueden tener múltiples entradas, con la salida igual a la AND o la OR de todas ellas
- La función lógica NOT se realiza mediante un inversor que siempre tiene una entrada

» Circuitos Combinacionales

Compuertas/Puertas (Gates)

AND

ab	c
00	0
01	0
10	0
11	1

OR

ab	c
00	0
01	1
10	1
11	1

NOT

a	b
0	1
1	0

Compuertas/Puertas (Gates)

XOR

ab	c
00	0
01	1
10	1
11	0

NAND

ab	c
00	1
01	1
10	1
11	0

NOR

ab	c
00	1
01	0
10	0
11	0

Diseño Lógico - Diseño Digital

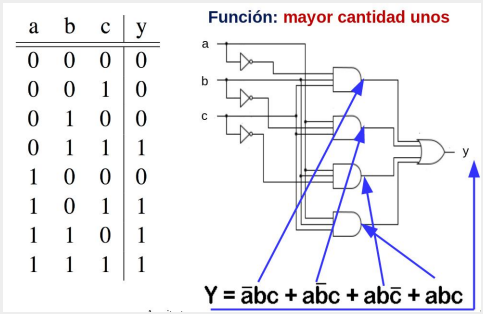
- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- **Metodología para realizar un diseño lógico/digital**
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

» Circuitos Combinacionales

Metodología para realizar un diseño lógico/digital de un circuito combinacional

1. Definir coloquialmente todas las entradas y todas las salidas
2. Confeccionar una tabla de verdad para todas las combinaciones de valores de entrada y todas las salidas
3. Describir para cada salida una función lógica en base a la tabla
4. Confeccionar un diagrama del circuito digital resultante

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

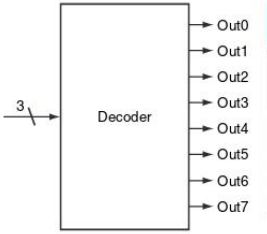


Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- **Circuitos combinacionales comunes**
- Relojes
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

» Circuitos Combinacionales

Construcción de diseño lógico/digital: Decoder

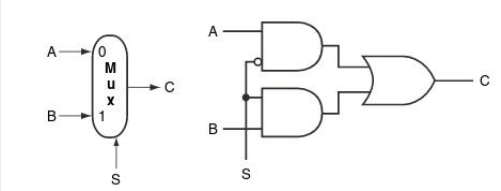


a. A 3-bit decoder

Inputs			Outputs							
12	11	10	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

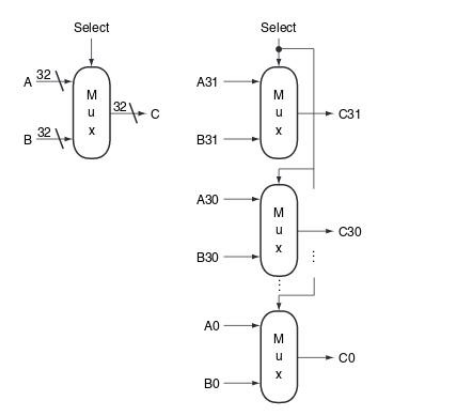
b. The truth table for a 3-bit decoder

Construcción de diseño lógico/digital: Multiplexor



» Circuitos Combinacionales

Construcción de diseño lógico/digital: Multiplexor 32 bits



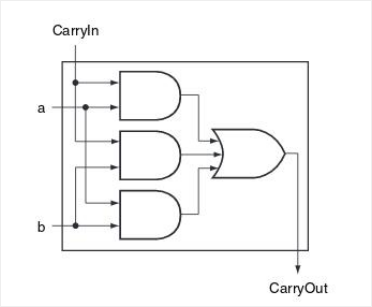
» Circuitos Combinacionales

Diseño de ALU de un bit

Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

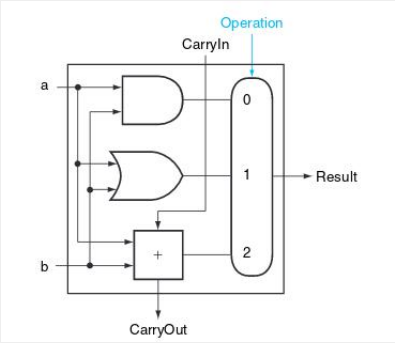
» Circuitos Combinacionales

Diseño de ALU de un bit



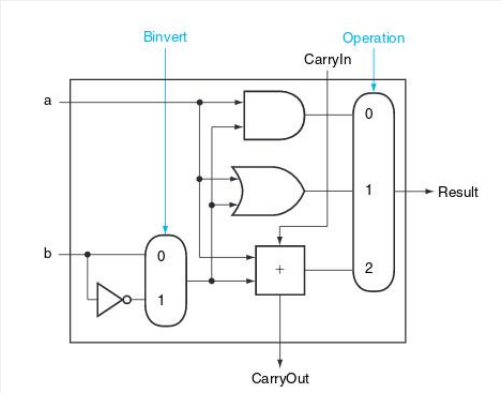
» Circuitos Combinacionales

Diseño de ALU de un bit

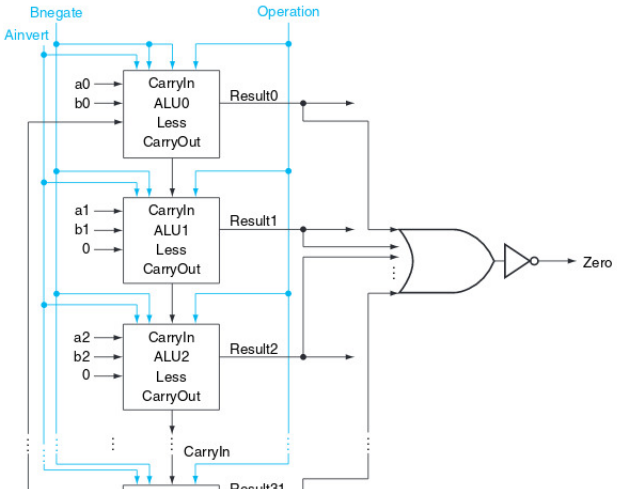


» Circuitos Combinacionales

Diseño de ALU de un bit

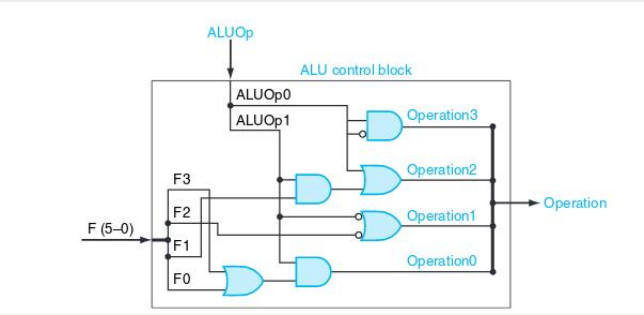


Diseño de ALU de 32 bits



» Circuitos Combinacionales

Diseño de una unidad de control



Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- **Relojes**
- Circuitos secuenciales
- Máquinas de Estado Finito (FSM)

Relojes

- Los relojes son necesarios en la lógica secuencial, para decidir cuando un elemento que contiene un estado debe ser actualizado.
- Terminología: tiempo del ciclo (período), frecuencia del reloj.
- Metodología : edge-triggered clocking (reloj disparado por flanco)

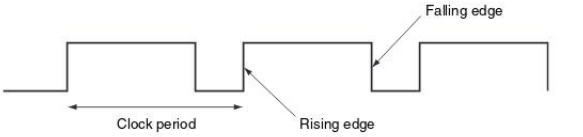


FIGURE C.7.1 A clock signal oscillates between high and low values. The clock period is the time for one full cycle. In an edge-triggered design, either the rising or falling edge of the clock is active and causes state to be changed.

Relojes

- Sistemas síncronos
- edge-triggered clocking posibilita un proceso instantáneo

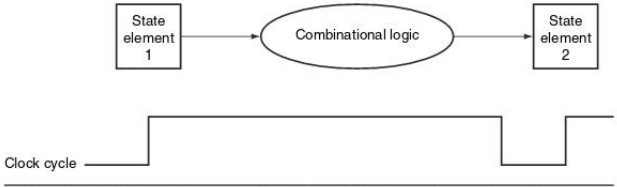


FIGURE C.7.2 The inputs to a combinational logic block come from a state element, and the outputs are written into a state element. The clock edge determines when the contents of the state elements are updated.

Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- **Circuitos secuenciales**
- Máquinas de Estado Finito (FSM)

» Circuitos secuenciales

Elementos de memoria: Set-Reset Latch

* Elemento básico para almacenar un bit. Contiene un Loop en su diseño.

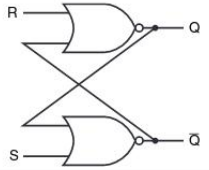


FIGURE C.8.1 A pair of cross-coupled NOR gates can store an internal value. The value stored on the output Q is recycled by inverting it to obtain $\overline{Q}$ and then inverting $\overline{Q}$ to obtain Q. If either R or $\overline{Q}$ is asserted, Q will be deasserted and vice versa.

» Circuitos secuenciales

Elementos de memoria: D Latch

- Elemento básico para almacenar un bit. Contiene un Loop en su diseño
- Sus entradas son el bit a almacenar y la señal de reloj

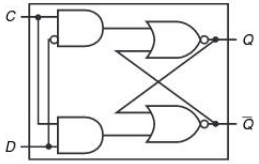


FIGURE C.8.2 A D latch implemented with NOR gates. A NOR gate acts as an inverter if the other input is 0. Thus, the cross-coupled pair of NOR gates acts to store the state value unless the clock input, *C*, is asserted, in which case the value of input *D* replaces the value of *Q* and is stored. The value of input *D* must be stable when the clock signal *C* changes from asserted to deasserted.

» Circuitos secuenciales

Elementos de memoria: D Latch

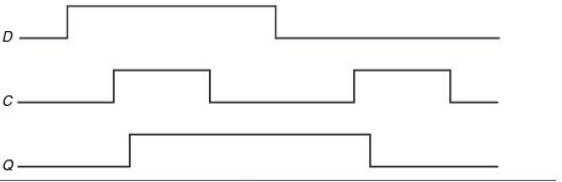


FIGURE C.8.3 Operation of a D latch, assuming the output is initially deasserted. When the clock, *C*, is asserted, the latch is open and the *Q* output immediately assumes the value of the *D* input.

» Circuitos secuenciales

Elementos de memoria: D Flip-Flop

- D Flip-Flop : utilizados en la construcción de REGISTROS
- Elemento básico para almacenar un bit. Contiene un Loop en su diseño
- Sus entradas son el bit a almacenar y la señal de reloj
- Se sincroniza (actualiza) en el flanco de reloj

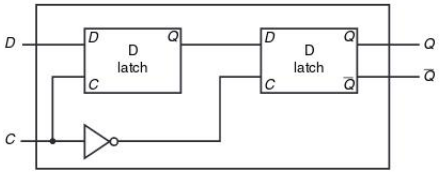


FIGURE C.8.4 A D flip-flop with a falling-edge trigger. The first latch, called the master, is open and follows the input *D* when the clock input, *C*, is asserted. When the clock input, *C*, falls, the first latch is closed, but the second latch, called the slave, is open and gets its input from the output of the master latch.

» Circuitos secuenciales

Elementos de memoria: D Flip-Flop

* Si la señal D cambia cuando la señal de reloj está baja (desacertada) el estado interno del flip-flop no cambia. Eso lo distingue de un latch.

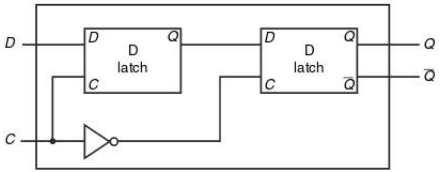


FIGURE C.8.4 A D flip-flop with a falling-edge trigger. The first latch, called the master, is open and follows the input *D* when the clock input, *C*, is asserted. When the clock input, *C*, falls, the first latch is closed, but the second latch, called the slave, is open and gets its input from the output of the master latch.

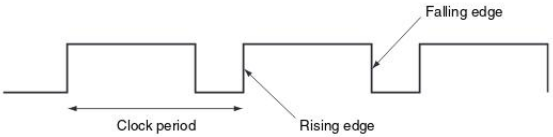
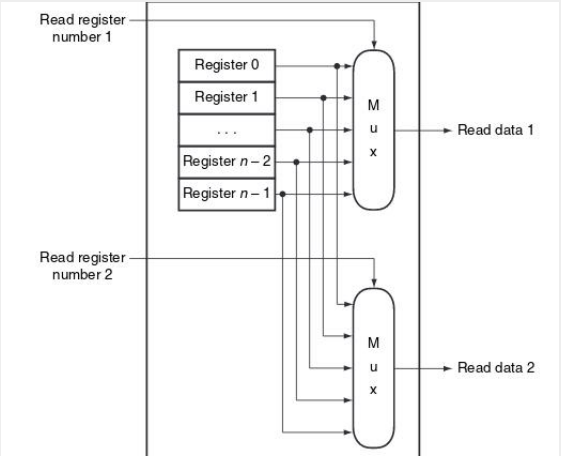


FIGURE C.7.1 A clock signal oscillates between high and low values. The clock period is the

» Diseño digital

Diseño de ALU y Registros

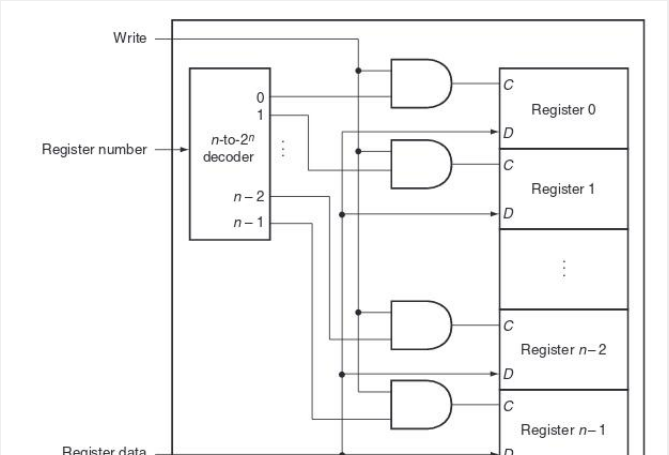
* Elementos de estado, multiplexores, decodificadores, ALU



» Diseño digital

Diseño de ALU y Registros

* Elementos de estado, multiplexores, decodificadores, ALU



Diseño Lógico - Diseño Digital

- Señales digitales
- Análisis de Circuitos Digitales
- Algebra de Boole
- Metodología para realizar un diseño lógico/digital
- Circuitos combinacionales comunes
- Relojes
- Circuitos secuenciales
- **Máquinas de Estado Finito (FSM)**

» Diseño digital

Máquinas de Estado Finito (FSM)

- Permiten especificar circuitos secuenciales
- Ejemplo: control de la calefacción de un coche

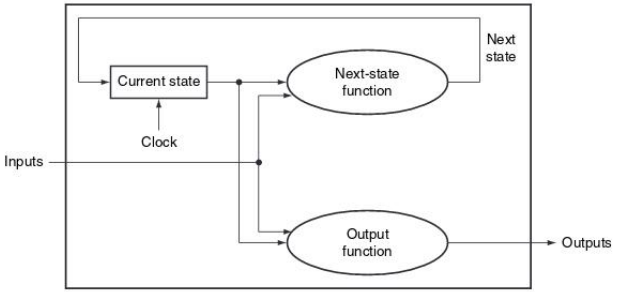


FIGURE C.10.1 A state machine consists of internal storage that contains the state and two combinational functions: the next-state function and the output function. Often, the output function is restricted to take only the current state as its input; this does not change the capability of a sequential machine, but does affect its internals.

» Diseño digital

Máquinas de Estado Finito (FSM)

Existen dos clasificaciones teóricas:

* Autómata de Mealy
* Autómata de Moore

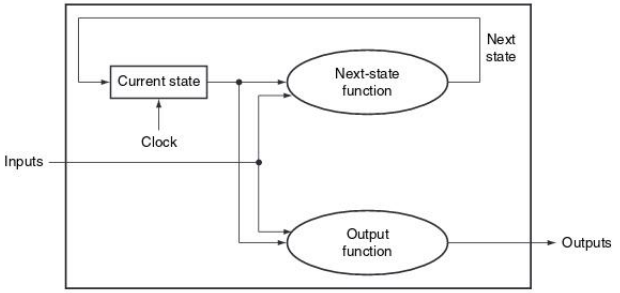


FIGURE C.10.1 A state machine consists of internal storage that contains the state and two combinational functions: the next-state function and the output function. Often, the output function is restricted to take only the current state as its input; this does not change the capability of a sequential machine, but does affect its internals.

» Diseño digital

Máquinas de Estado Finito (FSM)

* Ejemplo: Luces de giro del Ford Thunderbird

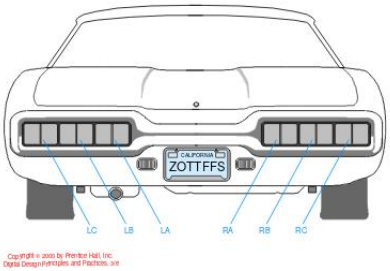
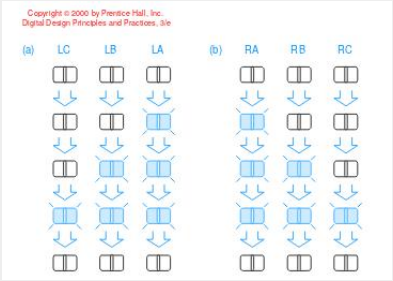


Figure 1: Thunderbird Tail Lights

» Diseño digital

Máquinas de Estado Finito (FSM)

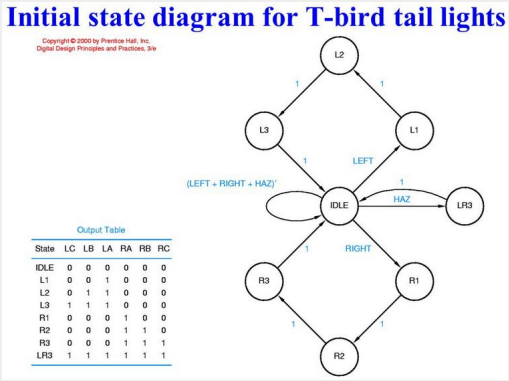
* Ejemplo: Luces de giro del Ford Thunderbird



» Diseño digital

Máquinas de Estado Finito (FSM)

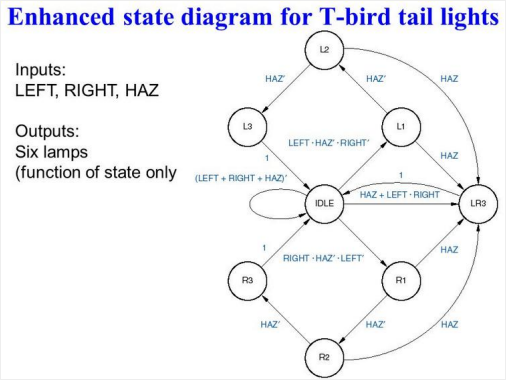
* Ejemplo: Luces de giro del Ford Thunderbird
* Diagrama de estados



» Diseño digital

Máquinas de Estado Finito (FSM)

* Ejemplo: Luces de giro del Ford Thunderbird
* Diagrama de estados (segunda versión)



» Consejos y preguntas

* ¿Preguntas?

» Bibliografía

Libros

* David. Patterson John L. Hennessy (1995), ORGANIZACIÓN Y DISEÑO DE COMPUTADORES La interfaz hardware/software, McGraw-Hill (8 copias en biblioteca).