Contents

Chapter 1

Compuertas

Las compuertas son circuitos que se emplean para generar niveles lógicos digi- tales (o sea unos y ceros, distinto a la electrónica lineal, que se preocupa por el valor en la curva) en formas específicas.

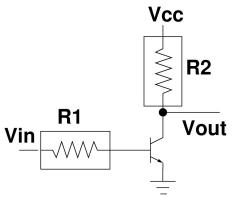
A continuación veremos algunas de las compuertas más importantes y se plantearón ejercicios prácticos.

1.1 Inversor

Le inversor, o NOT, es una compuerta que dada una entrada entrega una salida con el valor de entrada invertido. La notación que ocuparemos entonces para una entrada A negada será $\overline{A}$. Un circuito integrado que contiene compuertas NOT es el 7404; se ocupará en los laboratorios.

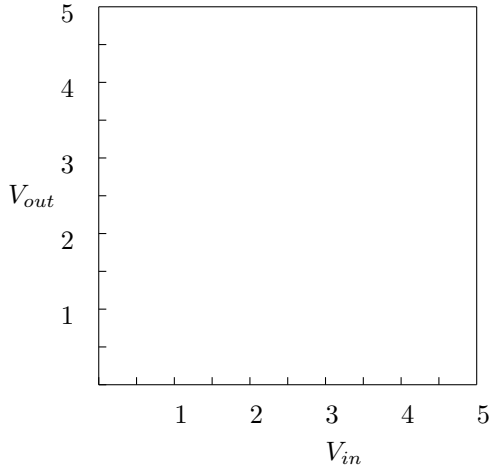
A	$\overline{A}$
0	1
1	0

El valor de la salida es el inverso de la entrada en este inversor simple:



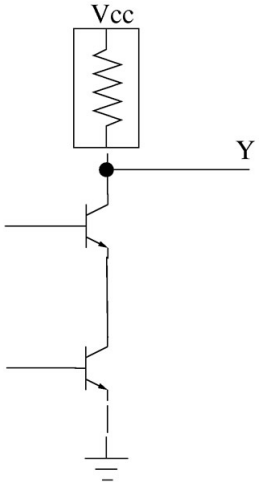
Si la unión entre la base y el emisor de este transistor bipolar es equivalente a un diódo, se puede esquematizar la corriente I_{be} como función del voltaje V_{be}. Cuando V_{be} es inferior a 0,6V

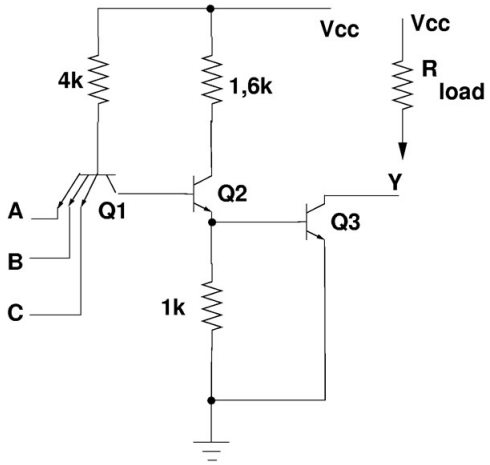
la corriente a través de la base es aproximadamente cero. En esa condición el transistor no va a conducir corriente desde el colector al emisor, y aparece como un interruptor abierto. V_{out} es alto en este caso, porque está conectado a V_{cc} a través de resistencia R2. Si se aumenta V_{be}, y supera a 0,6V, entonces la corriente I_{be} va a empezar a fluir, la base del transistor va a contener electrones libres, y la corriente I_{ce} también puede fluir. Así, V_{out} baja hasta llegar a cero.



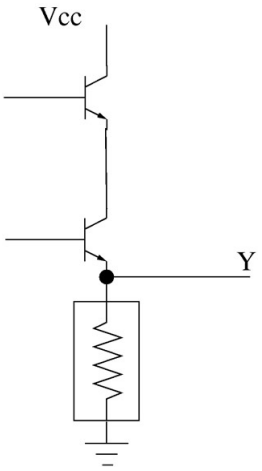
Dibujar la relación entre V_{in} y V_{out}. Que pasa con los voltajes de entrada intermedios?

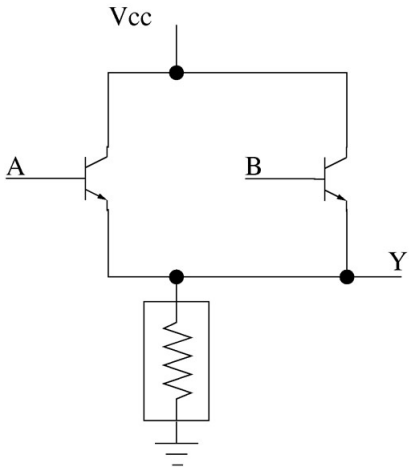
1.2 NANDs



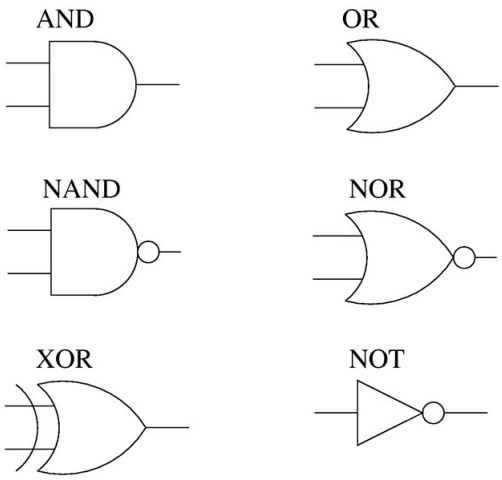


AND / OR





Símbolos lógicos



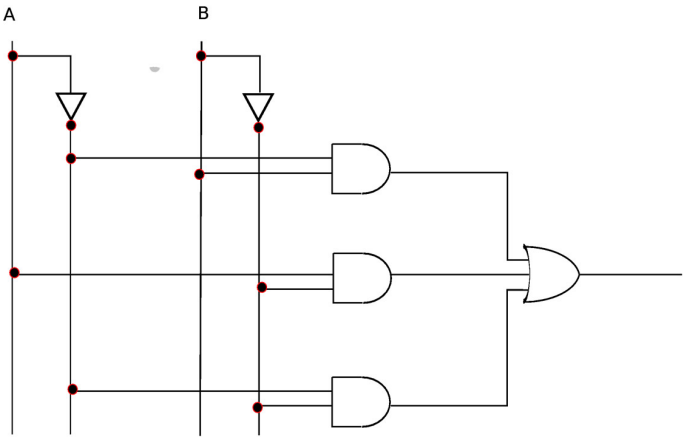
Exercises

1. Diseñar un circuito NOR simple en base a una compuerta OR y un NOT.

2. Diseñar un circuito XOR, un OR exclusivo, con compuertas AND, OR y NOT. La salida es verdadera si una, y solo una, de las entradas es verdadera. La salida es falsa si las dos entradas son iguales.

3. La salida de un circuito normal puede ser alto o bajo. En muchos casos es conveniente tener también un estado adicional, en que la salida no es alto ni bajo, pero simplemente no conduce. Esta característica es muy importante en busses de datos y comunicaciones. Diseñar un circuito inversor (NOT) con transistores, que tiene una salida Y y dos entradas, X y En (habilitación). La salida Y es la negación de X cuando En es alto. La salida no conduce (ni alta ni baja) cuando En es bajo. Esto significa que ambos transistores arriba y abajo de la salida están apagados cuando En no está habilitado.

4. Escribir la ecuación lógica correspondiente a la siguiente figura. El resultado es una forma llamado 'suma de productos' que se va a ver en los próximos capítulos en lógica booleana.



Chapter 2

Boolean logic

En el primer módulo se mostró la forma de tratar voltajes y corrientes como números binarios, 0 y 1. Después se vió como manipularlos en compuertas para generar expresiones lógicas como el AND, OR, NOT y NOR. Estas compuertas forman la base conceptual de la álgebra [1] booleana. [2]

2.1 Teoremas Lógicas

2.1.1 Teoremas fundamentales

$x * 0 = 0$	$x + 0 = x$
$x * 1 = x$	$x + 1 = 1$
$x * x = x$	$x + x = x$
$x * \overline{x} = 0$	$x + \overline{x} = 1$

El símbolo '*' significa AND, '+' significa OR. Así como en álgebra normal es común obviar el símbolo '*' en una expresión.

2.1.2 Teoremas de múltiples variables

$x + y = y + x$	$(w + x)(y + z) = wy + xy + wz + xz$
$x * y = y * x$	$x + xy = x$
$x + (y + z) = (x + y) + z = x + y + z$	$x + \overline{x}y = x + y$
$x(yz) = (xy)z$	$\overline{x} + xy = \overline{x} + y$
$x(y + z) = xy + xz$	

[1] La palabra *álgebra* tiene su raíz en la palabra *al-jabr* en Arabe, que significa restoration. Fue parte del título de un libro del matemático Mohammad ibn-Musa al-Khowarismi, aprox. 825 AD, que también dió su nombre a nuestro término *algoritmo*.

[2] George Boole (1815-1864) nació en una pobre familia obrera inglesa, pero por su fuerza de personalidad y aprecio por la matemática, logró alcanzar los más altos niveles académicos de sus tiempos. Su trabajo cambió el estudio de la lógica desde una orientación filosófica y le dió una base matemática, que es lo que nos permite hoy en día tecnología digital.

2.1.3 Teoremas de DeMorgan

$\overline{(x+y)} = \overline{x} * \overline{y}$ $\overline{(x*y)} = \overline{x} + \overline{y}$

2.2 Simplificación de expresiones con álgebra booleana

$z = \overline{(\overline{a}+c)*(b+\overline{d})}$

a una expresión que contiene unicamente variables simples invertidas.

$$\begin{aligned} z &= \overline{(\overline{a}+c)} + \overline{(b+\overline{d})} \\ z &= \overline{\overline{a}} * \overline{c} + \overline{b} * \overline{\overline{d}} \\ z &= a * \overline{c} + \overline{b} * d \end{aligned}$$

Simplificar:

$$z = \overline{a + \overline{b} * c}$$

Resultado:

$$\begin{aligned} z &= \overline{a} * \overline{(\overline{b} * c)} \\ z &= \overline{a} * (\overline{\overline{b}} + \overline{c}) \\ z &= \overline{a} * (b + \overline{c}) \\ z &= \overline{a} * b + \overline{a} * \overline{c}) \end{aligned}$$

Simplificar:

$$w = \overline{(a + b * c) * (d + e * f)}$$

Resultado:

$$w = \overline{a} * \overline{b} + \overline{a} * \overline{c} + \overline{d} * \overline{e} + \overline{d} * \overline{f}$$

2.3 Mapas de Karnaugh

Mapas Karnaugh [3] se utilizan para poder simplificar ecuaciones lógicas en forma gráfica. Se prepara una planilla de las variables, en que las filas o columnas colindantes difieren solamente en el estado de una variable. Por ejemplo, entre la columna 2 y 3 de una planilla, una variable va a figurar negada en una columna y directa en la la otra. Las otras variables tienen el mismo estado en las dos columnas.

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Tabla de verdad
$X = \overline{A} * \overline{B} + A * B$
Ecuación lógica

	$\overline{B}$	B
$\overline{A}$	1	0
A	0	1

Diagrama de Karnaugh

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Tabla de Verdad
$X = \overline{A} * \overline{B} * \overline{C} +$
$\overline{A} * \overline{B} * C +$
$\overline{A} * B * \overline{C} +$
$A * B * \overline{C}$
Ecuación lógica

	$\overline{C}$	C
$\overline{A}\overline{B}$	1	1
$\overline{A}B$	1	0
AB	1	0
$A\overline{B}$	0	0

Diagrama de Karnaugh

[3] Maurice Karnaugh, recibió su PhD en Física de Yale en 1952, trabajó en Bell Labs, y después en IBM, hasta 1993. Inventor del primer sistema digital 'switching' de comunicaciones

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Tabla de Verdad

$X = \overline{A} * \overline{B} * C * \overline{D} +$
$\overline{A} * \overline{B} * C * D +$
$\overline{A} * B * C * \overline{D} +$
$\overline{A} * B * C * D +$
$A * \overline{B} * C * \overline{D} +$
$A * B * \overline{C}$

Ecuación lógica

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	1	1
$\overline{A}B$	0	0	1	1
AB	0	0	0	0
$A\overline{B}$	1	0	0	1

Mapa de Karnaugh

Exercises

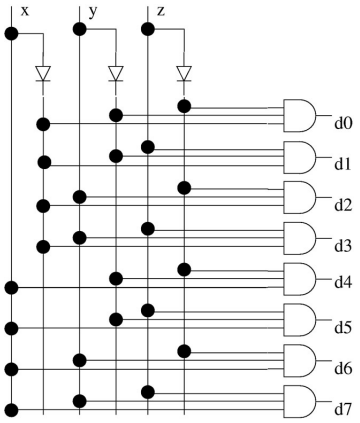
- Verificar que la simplificación mediante álgebra booleana da los mismos resultados que las mapas Karnaugh.
- Que ventaja tiene álgebra booleana con respecto a mapas K.?

Chapter 3

Combinatorial circuits

3.1 Decodificadores

Un decodificador n x m convierte un número binario de n bits a una línea única de salida de un máximo de m líneas de salida.

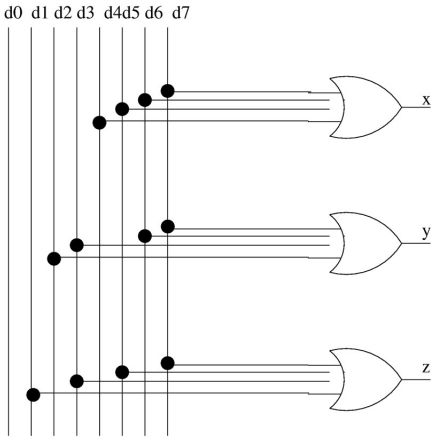


x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

3.2 Codificadores

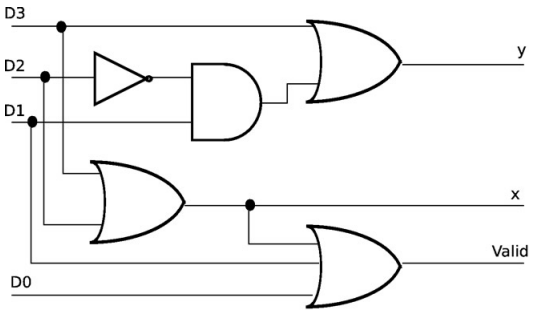
Codificadores (encoders) realizan la operación inversa de los decodificadores. Su salida es una de m líneas que es activada de acuerdo a cual de los 2^m líneas de entrada es activada. Una falla del codificador común es que no diferencia entre una entrada con todos ceros, y una entrada con un

1 en la línea 0. Resolver este problema requiere una entrada adicional de habilitación cuando una de las entradas es activada.



D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

A priority encoder is an encoder that will only react to one of the inputs. You can imagine a problem with the normal encoder that would occur if several inputs were activated at the same moment. A priority encoder resolves this problem.

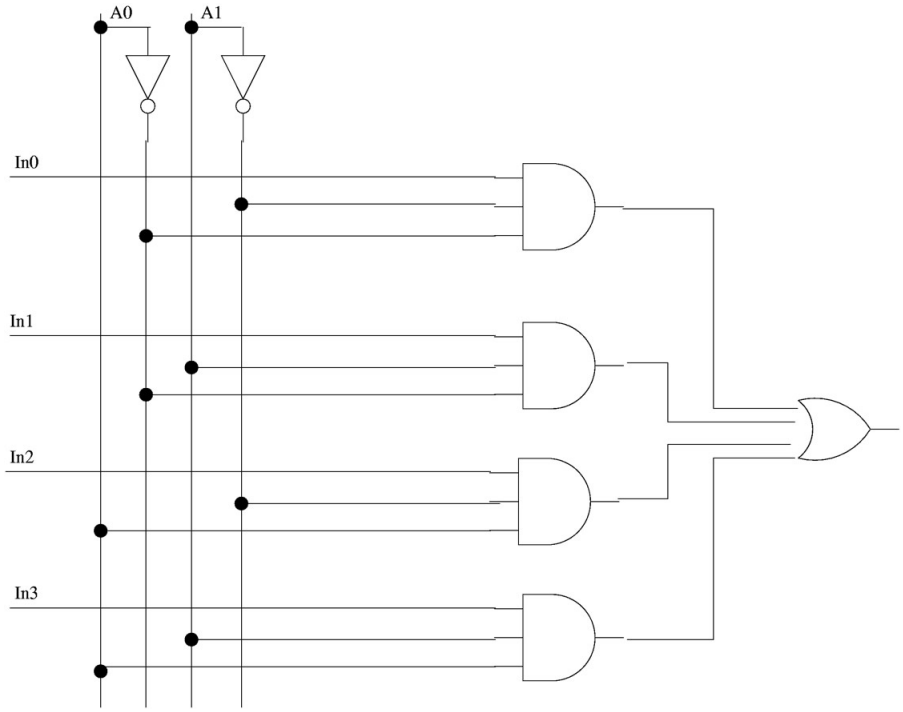


The Boolean equations for this circuit are:

$$x = D_2 + D_3$$
$$y = D_3 + D_1\overline{D_2}$$
$$V = D_0 + D_1 + D_2 + D_3$$

3.3 Multiplexores

Un multiplexor es similar a un decodificador. Tiene dos tipos de entradas; una entrada es un conjunto de líneas representando datos, y la otra entrada es una dirección que selecciona cual línea de entrada aparece en la salida. Es similar a un equipo de música que tiene varios micrófonos como entrada, y selecciona uno para enviar a los parlantes. Tiene solo una línea de salida. Los multiplexores son muy usados en microprocesadores y memorias.



Exercises

- Definir la tabla de verdad de un codificador de prioridad que activa la línea correspondiente a la entrada más alta. Utiliza una salida de validéz igual a cero si ninguna de las entradas es activida.
- Diseñar un circuito para el codificador de prioridad de 4 entradas.
- Diseñar un circuito para convertir código Gray de 4 bits a código binario.
- Diseñar un circuito para comparar 2 números de 4 bits y generar un 1 si son iguales, o un 0 si difieren.
- Dibujar un decodificador de 2 x 4, con una línea de habilitación, utilizando unicamente compuertas NOR.

Chapter 4

Digital arithmetic

La presentación de aritmética digital empieza con representación de números en binario y hexadecimal. Después se describen circuitos para realizar sumas, y la representación de números en complemento a dos, lo que facilita la realización de restas en hardware.

4.1 Sumas en binario y en hex

Realizar una suma en binario es simple, pero muy tedioso. Se tiene que recordar simplemente que $10_2 = 2_{10}$ y que el segundo dígito es acarreo. Por ejemplo:

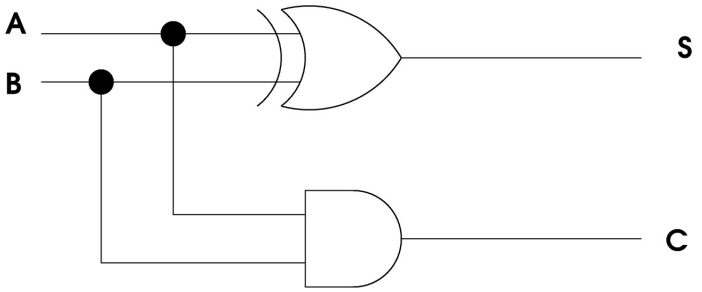
	1	1	0	1	13_{10}
+	0	1	0	1	5_{10}
1	0	0	1	0	18_{10}

Este ejemplo muestra también que los dos números que se suman se pueden representar en 4 dígitos binarios, mientras el resultado requiere 5. Esto no tiene que ver con que se está sumando números binarios, pero sí tiene que ver con el hecho que las sumas se hacen en un computador con una representación de números con precisión finita y limitada. Por ejemplo, un sistema con 16 bits para enteros positivos y negativos (1 bit para el signo), puede representar $2^{16} = 65536$ números diferentes, tipicamente de -32768 hasta 32767 (no se olvida el cero).
Sumas en hex son escencialmente lo mismo. Uno se encuentra sumando y restando en hex cuando está analizando el stack en un programa para determinar desfazajes y bifurcaciones. Por ejemplo, si el puntero a la dirección de una línea de código (el PC, program counter) está en 0x8440, y una bifurcación apunta a treinta sentencias más adelante ($30_{10} = 0x1E$) o sea 240 ($240_{10} = 0xF0$) bytes más adelante en una máquina de 32 bits, entonces tiene que sumar:

$$\begin{array}{r} \text{0x8440} \\ +\ \text{0xF0} \\ \hline \text{0x8530} \end{array}$$

4.2 Medio sumador

Para generar un circuito para realizar sumas, sumas de solo un bit, es conveniente empezar con la tabla de verdad, donde se suman A y B para dar la suma S, y el acarreo C:



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

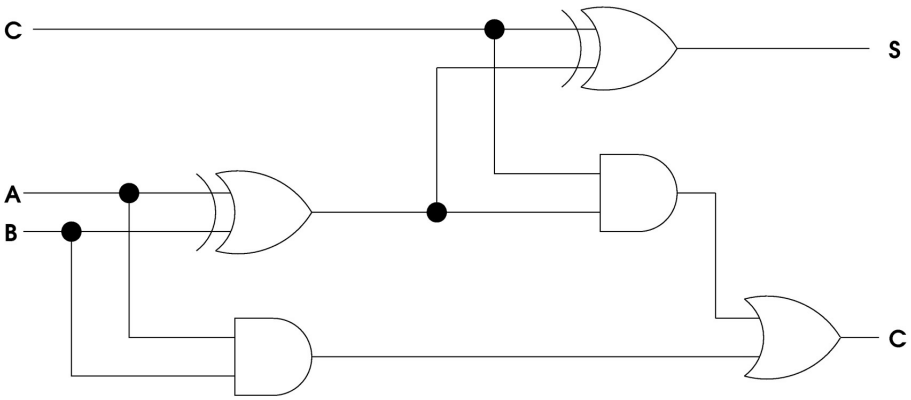
Al mirar la tabla de verdad es claro que el acarreo es simplemente un AND de A y B, mientras la suma es el XOR de A y B. Este circuito no tiene un acarreo entrante.

4.3 Sumador completo

Se inicia el análisis de un sumador completo con su tabla de verdad. Las entradas son A y B, al igual que el medio sumador, y también tiene un acarreo entrante, que vamos a llamar C_i. La salida del circuito es la suma, S, y el acarreo saliente, C_o.

A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Para resolver esta tabla de verdad, se puede simplificar utilizando los teoremas booleanos, o mapas de Karnaugh. Las ecuaciones son:
$S = \overline{A} * \overline{B} * C_i + \overline{A} * B * \overline{C_i} + A * \overline{B} * \overline{C_i} + A * B * C_i$
$C_o = A * B + A * C_i + B * C_i$



El circuito ilustrado aquí es el conjunto de dos sumadores medios más una compuerta OR. La suma es 1 cuando el número de entradas en 1 es impar, por eso el XOR del XOR. El acarreo saliente es un AND de las entradas, o un AND del XOR de las entradas y el acarreo entrante.

4.4 Complemento a dos

La representación de números negativos es un aspecto importante de aritmética digital. La representación simple, llamado signo - magnitud, es de utilizar un bit para el signo, y los otros bits para la magnitud. Por ejemplo, si utilizamos 5 bits total, podemos representar números de la siguiente manera:

0	0	1	0	0	4_{10}
1	0	1	0	0	-4_{10}
0	1	1	0	0	12_{10}
1	1	0	1	0	-10_{10}

El problema con la representación signo-magnitud es que no podemos sumar números positivos y negativos juntos. Por ejemplo, el resultado de 12 + (-10) es 6 (0110 en 4 bits) y no se sabe que hacer con el bit del signo.

Una alternativa (no la única) es la representación en complemento a dos que se utiliza en los computadores desde los años '80. Números con signo se representan en este formato porque es muy simple y rápido convertir de una representación a otra, y permite realizar aritmética con o sin signo en un solo circuito.

Para determinar un número en complemento a dos, se escribe su magnitud (el valor absoluto), se complementan todos los bits del número (los 1's se cambian a 0's y vice versa), y se le suma 1. Por ejemplo, para encontrar la representación de -12_{10} en complemento a dos, con 4 bits más un bit para el signo:

$$
\begin{array}{r|rr}
12_{10} \rightarrow 01100_2 & & \\
\hline
 & 01100 & 10100 \\
 & 10011 & 01011 \\
 & +\ 1 & +\ 1 \\
\hline
 & 10100 & 01100
\end{array}
$$

Para encontrar la magnitud de un número de complemento a dos se hace exactamente el mismo procedimiento; complementar cada dígito y sumarle 1. Para comprobar que la conversión ha sido correcta, se puede sumar los 2 números:

$$
\begin{array}{r}
01100 \\
+\ 10100 \\
\hline
100000
\end{array}
$$

El resultado es cero, como corresponde, pero con un acarreo demás, que tenemos que descartar.

Exercises

- Asegúrese que está familiarizado con la conversión entre números binarios, decimales y hexadecimales.
- Diseñar un medio sumador utilizando 3 compuertas AND y 1 OR sobre las entradas directas y las entradas negadas.
- Diseñar un medio sumador utilizando 2 compuertas AND, 1 OR y 1 NOT sobre las entradas directas y las entradas negadas.
- Diseñar un sumador completo utilizando 3 compuertas AND y 1 OR para la suma, y 3 AND más 1 OR para el acarreo.
- Practicar realizando conversiones de signo-magnitud a complemento a dos.

Chapter 5

Arithmetic and logical unit: ALU

5.1 Sumador

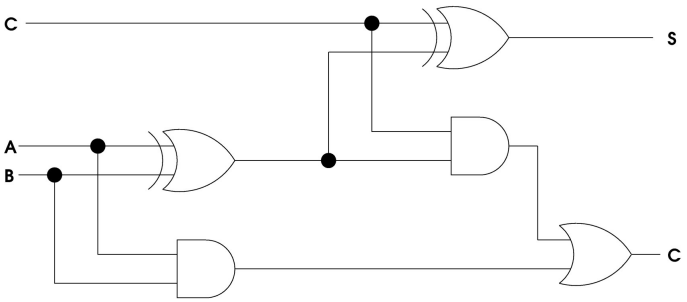
Recordando el sumador completo:

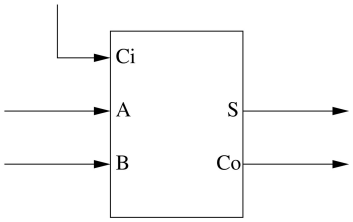
A	B	C_{in}	Σ	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$\Sigma = C_{in} \bigotimes (A \bigotimes B)$

$C_{out} = A * B + C_{in} * (A \bigotimes B)$
$C_{out} = \overline{A} * B * C_{in} + (A * (B + C_{in})$

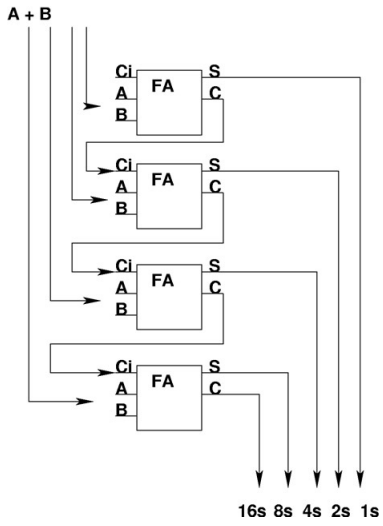
La tabla de verdad del sumador completo puede implementarse de varias maneras, un ejemplo simple utiliza las compuertas XOR. Este sumador sirve para sumar solamente 1 bit. Si se desea sumar multiples bits, por ejemplo los 32 bits de una palabra en un PC, se pueden juntar 32 sumadores de 1 bit.





Sumador Completo Representación en bloque

El conjunto de sumadores se llama 'ripple', o paralelo, porque hay una propagación de sumas desde el primer bit hasta el último. Nótese que la suma del primer bit tiene que terminar antes que se puede iniciar la suma del segundo bit, lo que hace que este sumador es más lenta con generación de acarreo anticipada.



Para reducir la demora en un sumador paralelo, se puede diseñar un circuito para generar el acarreo anticipadamente. Definamos dos variables intermedias en el diagrama del sumador completo:
$P_i = A_i \bigotimes B_i$
$G_i = A_i * B_i$
entonces la suma y acarreo pueden representarse como:
$S_i = P_i \bigotimes C_i$
$C_{i+1} = G_i + P_i * C_i$
El acarreo en C_{i+1} va a generarse siempre cuando G_i sea 1. También hay una propagación del acarreo desde la anterior suma cuando P_i sea 1. Las ecuaciones llegan a ser largas, pero permiten calcular el acarreo rapidamente.

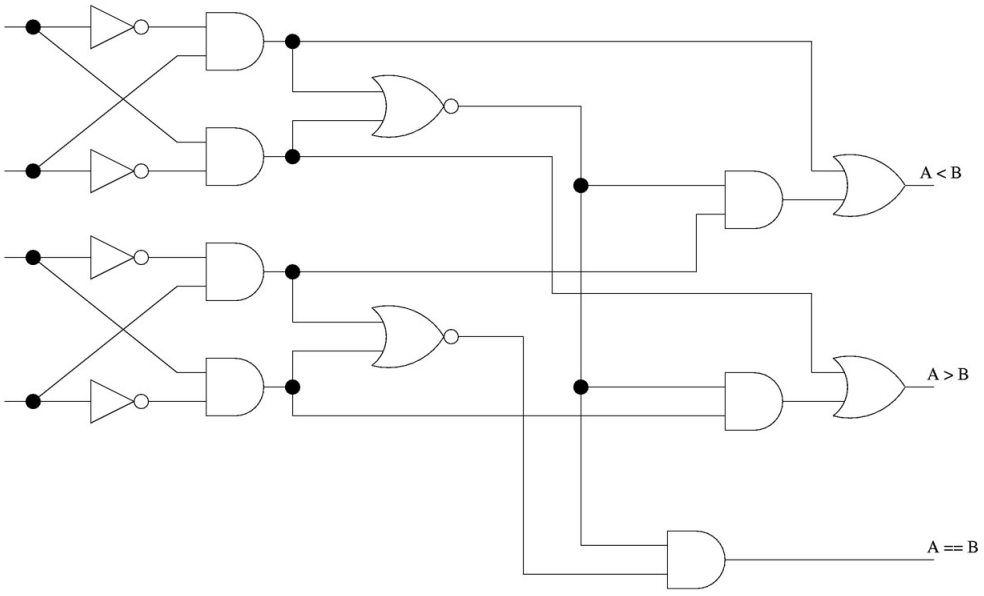
$C_2 = G_1 + P_1 * C_1$
$C_3 = G_2 + P_2 * C_2$
$= G_2 + P_2 * (G_1 + P_1 * C_1)$
$= G_2 + P_2 * G_1 + P_2 * P_1 * C_1$
$C_4 = G_3 + P_3 * G_2 + P_3 * P_2 * G_1 + P_3 * P_2 * P_1 * G_1$
y así sucesivemente.

5.2 Comparación de magnitudes

La comparasión de magnitudes de números es fundamental para la evaluación de condiciones en la ejecución de programas. Por ejemplo, código puede pedir que se ejecute un bloque si un número es igual o menor que otro. Al traducir este código a ensamblador resulta en una bifurcacón condicional; si el primero es mayor, entonces saltar a otra instrucción más adelante.

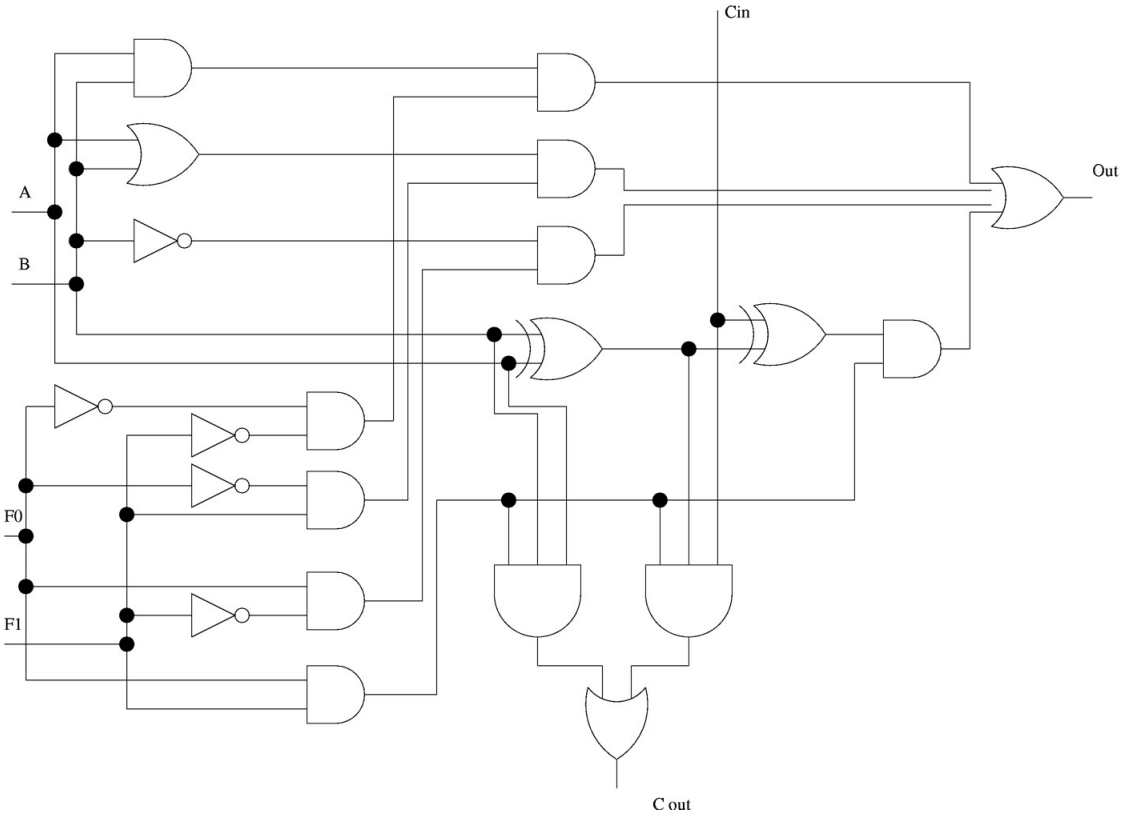
Tomemos:
$A = A_3A_2A_1A_0$
$B = B_3B_2B_1B_0$
Los dos números son iguales si:
$x_i = \overline{(A_i \bigotimes B_i)} = A_i * B_i + \overline{A_i} * \overline{B_i} = 1$
para todo i. Para determinar si $(A > B)$, hay que evaluar cada columna de los números sucesivemente:
$(A > B) = A_3 * \overline{B_3} + x_3 * A_2 * \overline{B_2} + x_3 * x_2 * A_1 * \overline{B_1} + x_3 * x_2 *_1 *A_0 * \overline{B_0}$

El circuito no es tan complejo como podría parecer porque se pueden utilizar las mismas compuertas para cálcular x_i como para las inigualdades.



5.3 Integración de la ALU

Con los circuitos aritméticos, de comparasión y las operaciones lógicas, se completan los componentes básicos de una ALU. Es necesario integrarlos, y poder seleccionar que operación realizar. Tomemos como ejemplo una ALU de cuatro funciones. Entonces el control de ejecución de una instrucción en ensamblador requiere 2 líneas para selección de la operación, a través de un multiplexor (2 líneas de dirección, 4 líneas de salida). El ejemplo graficado es una ALU de 1 bit, con las operaciones:
$AandB$
$AorB$
$\overline{B}$
$A + B$
Las líneas F_0, F_1 controlan que resultado aparece a la salida de la ALU.



Exercises

- Diseñar un circuito para generación de acarreo para 4 bits.
- Verificar los valores intermedios del comparador de magnitudes de 2 bits.
- Diseñar un circuito para comparar la magnitud de dos números de 4 bits.

- Para reducir el consumo energético de una ALU conviene no activar compuertas que no intervienen en una operación. Existe una posibilidad de reducir las compuertas actividades en la ALU graficada?

Chapter 6

Eagle

Objetive

- Conocer el procedimiento para fabricar un circuito impreso utilizando herramientas de software.

Circuit design is a process that includes

- Decide what your circuit must do
- Create a general overview of your circuit
- Design the schematic, with components and connections
- Simulate the circuit to make sure it does what you want
- Export the schematic to a printed circuit board software
- Place the components on the PCB and route the networks
- Send the finished PCB design to a circuit manufacturer
- Populate the PCB with components
- Install the system, charge your customer and celebrate

Many software tools are available to help in these processes, commonly called EDA (Electronic Design Automation) tools, which are in the general category of CAD (Computer Aided Design). This class / laboratory will introduce one such software product, *Eagle*.

Eagle is a software product produced by *Cadsoft*, a German firm, that is available for both Linux and Windows, through their site at *www.cadsoft.de*. Eagle includes both schematic capture and PCB layout. Cadsoft offers a free version that is limited in that it can be used for making circuits up to 80mm x 100mm, but no more.

The best way to learn Eagle is to follow their guided tour available at *http://www.cadsoft.de/Tour/tour01.htm* where a couple NAND gates are placed on the schematic, and a PCB is set up with a PIC microprocessor and a few discrete components. Follow the tour, it's easy.

Exercises

- Find out what is the software called *Spice*, and if it would be useful for you.
- What tools are inclued in gEDA?
- The most important exercise is to design and illustrate your circuit.

Chapter 7

Flip flops

Objetive

- Describir mecanismos básicos de lógica secuencial
- Identificar diferentes tipos de flip-flops

References

- Ronald J. Tocci (2001), *Digital Systems: Principles and Applications, 8^{th} Edition*, Prentice Hall, Capítulo 5
- Patterson and Hennesey (1997), *Computer Organization and Design, 2^{nd} Edition*, Morgan Kauffman, Appendix B

7.1 Lógica combinacional vs. secuencial

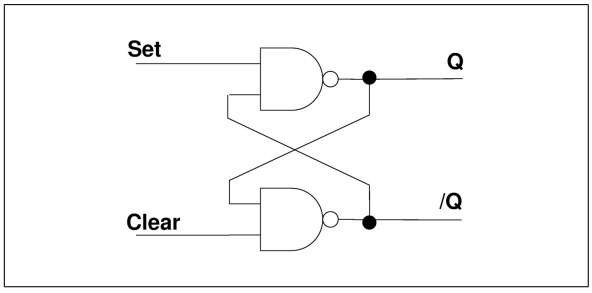
Circuitos de lógica combinacional tienen salidas que dependen exclusivemente de las entradas al circuito. Ejemplos de circuitos combinacionales son multiplexores, codificadores y circuitos aritméticos. La salida de un circuito de lógica secuencial depende de las entradas, pero también depende del estado anterior del circuito. Ejemplos de circuitos secuenciales son contadores, registros y memorias. Circuitos secuenciales se basan en un componente llamado el *flip-flop (FF)*, que significa rebote, o dar vuelta. El término formal para el FF es *biestable multivibrador*, pero este nombre suena más como un nuevo producto para relajación personal que un circuito lógico.

7.2 S-C (S-R) NAND FF

El FF más simple se compone de dos compuertas NAND (o dos NOR's). Recordemos la tabla de verdad de un NAND.

A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

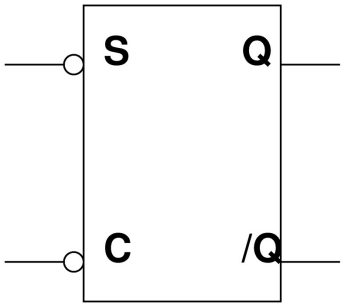
El S-C FF se compone de dos NANDs con la salida retroalimentando la entrada de la otra compuerta.



Para analizar el funcionamiento del FF, se supone que las salidas Q y $\overline{Q}$ tienen valores invertidos; cuando Q == 1, entonces $\overline{Q}$ == 0, y vice versa. Las entradas S y C van a estar normalmente en el estado alto (== 1) y se pulsan abajo para modificar las salidas. Cuando S = C = 1, hay dos estados de las salidas igualmente probables, y que dependen de los valores anteriores de Q y $\overline{Q}$. Este hecho, que la salida depende del valor anterior, es lo que hace que los FF sean los componentes básicos de las memorias.

Suponiendo que S = C = 1, entonces el valor de Q será 0 si es $\overline{Q}$ también era 1, sino Q será 1. Igualmente con esta entrada, $\overline{Q}$ será 0 si Q era 1, sino $\overline{Q}$ será 1. Conviene trazar estos valores por el circuito para convencerse de su funcionamiento.

Representación en bloque de un S-C FF



Resumen de funcionamiento NAND latch

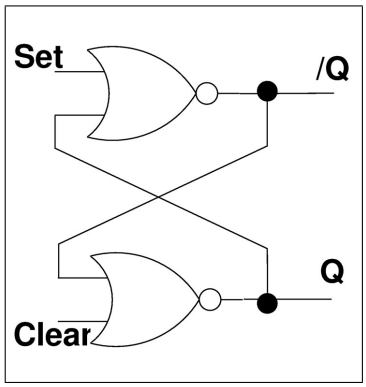
S	C	Salida
1	1	sin cambio
0	1	Q = 1
1	0	Q = 0
0	0	inválido

Se dice que la entrada S = C = 0 es inválida no porque no se le puede aplicar la combinación, en esta situación las salidas son Q = $\overline{Q}$= 1. Se prefiere mantener las salidas con valores complementarios,

entonces se trata de evitar esta entrada. Si las dos entradas, estando en niveles bajos, se levantan simultaneamente, el valor que toma la salida es dificil determinar, al final una de las entradas va a subir antes que la otra, y el valor de la salida va a determinarse cual sube más lentamente.

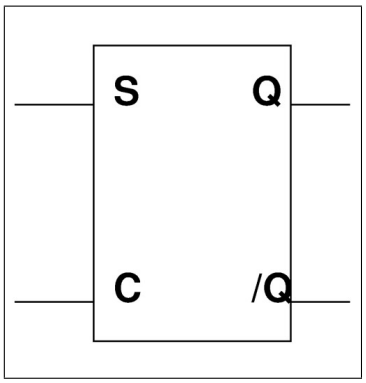
7.3 NOR latch

Un S-C flip-flop puede fabricarse de dos compuertas NOR de una manera muy parecida a un NAND latch. La diferencia es que las entradas son invertidas; S = C = 0 es la posición normal de reposo en que las salidas no cambian, y se activa las entradas subiendo S o C a 1. El estado S = C = 1 es el estado inválido.



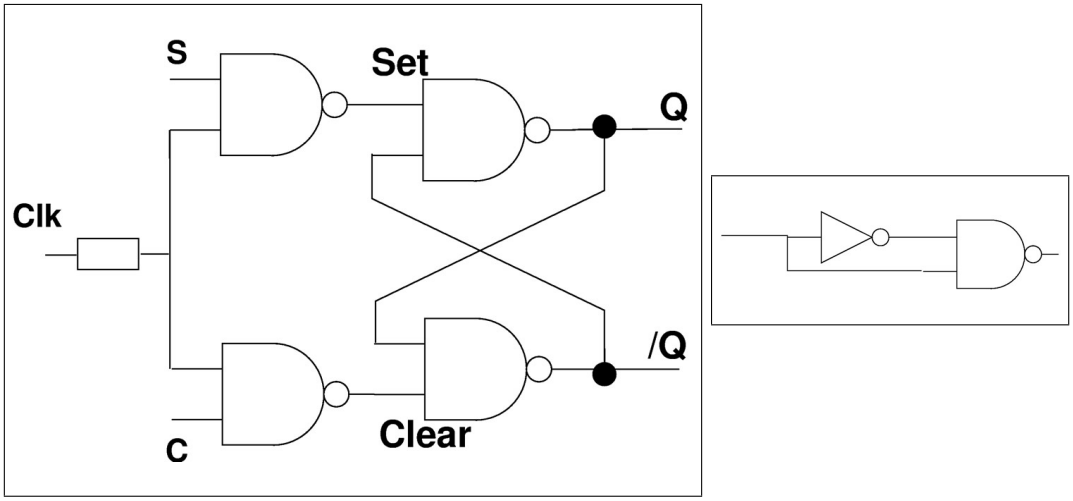
Resumen de funcionamiento
NOR latch

S	C	Salida
0	0	sin cambio
1	0	Q = 1
0	1	Q = 0
1	1	inválido



7.4 S-C flip-flops con reloj

Para reducir la posibilidad de ocurrencia de estados de entrada inválidos, se puede gatillar la entrada con un pulso de reloj. Los valores de entrada al FF (Set y Clear en el diagrama) se mantienen en 1 siempre que el reloj esté abajo. En este estado, las salidas no van a cambiar, de acuerdo a la tabla de funcionamiento (verdad) del circuito.
Este circuito sigue con el problema de indefinición de la salida: si S = C = 1 y el reloj sube a 1, después baja, entonces no se sabe que valores van a estar en la salida.



El otro circuito ilustrado es un generador de flancos. Cuando la entrada cambia, la salida va a cambiar durante un período muy breve, equivalente al tiempo que demora la compuerta NOT en cambiar de valor.

Exercises

- Un circuito para remover rebotes de un interruptor puede fabricarse de un FF NAND. El interruptor tiene dos posiciones, ambas conectadas a entradas al FF S-C. Las dos entradas S-C están llevado arriba a través de una resistencia. Como funciona este circuito?

Chapter 8

J-K Flip flops

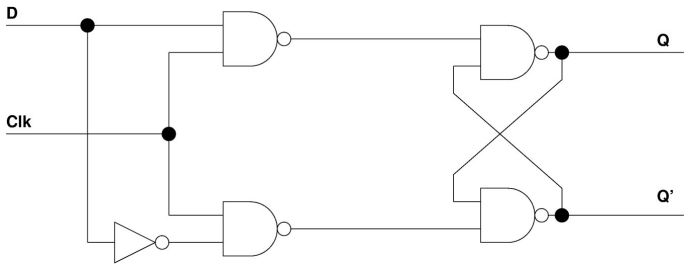
Objetive

- Describir mecanismos básicos de lógica secuencial
- Identificar diferentes tipos de flip-flops

References

- Ronald J. Tocci (2001), *Digital Systems: Principles and Applications, 8^{th} Edition*, Prentice Hall, Capítulo 5
- Patterson and Hennesey (1997), *Computer Organization and Design, 2^{nd} Edition*, Morgan Kauffman, Appendix B

8.1 D flip-flops

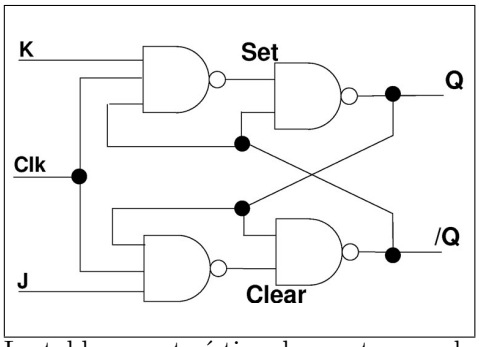


Un FF llamado 'D', por datos, elimina el estado indefinido, porque tiene solo una entrada. Este tipo

de FF es comunmente utilizado en registros y memorias, porque el dato de entrada es mantenido hasta el próximo pulso del reloj de la entrada.

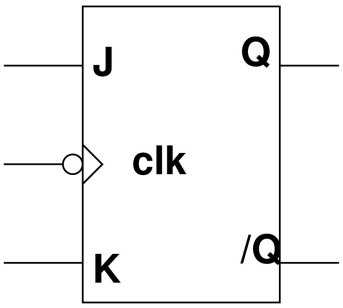
8.2 J-K flip-flops (toggle)

El J-K FF extiende la retroalimentación de las salidas para también ingresarla a las entradas, ahora llamas J y K. Cuando las entradas J y K están en 1, y se aplica un pulso al reloj, entonces las salidas se complementan, se invierten.



J	K	Q_1
0	0	Q_0
1	0	1
0	1	0
1	1	$\overline{Q}_0$

La tabla característica demuestra que la salida cambia de valor cuando J = K = 1. El diagrama de bloque de un FF J-K demuestra que la entrada del reloj es sobre el flanco de bajada del pulso del reloj.



Exercises

- Realizar el seguimiento de los estados internos de los FF J-K, y D.
- Dibujar la traza de salida de un FF J-K cuando J = K = 1, y se aplica un tren de pulsos al reloj.
- Conectar la salida Q de un J-K al reloj de un segundo J-K, mantener las entradas en 1. Dibujar las trazas de las salidas Q de ambos FF junto con un tren de pulsos al reloj.
- Un circuito para remover rebotes de un interruptor puede fabricarse de un FF NAND. El interruptor tiene dos posiciones, ambas conectadas a entradas al FF S-C. Las dos entradas S-C están llevado arriba a través de una resistencia. Como funciona este circuito?

Chapter 9

Counters

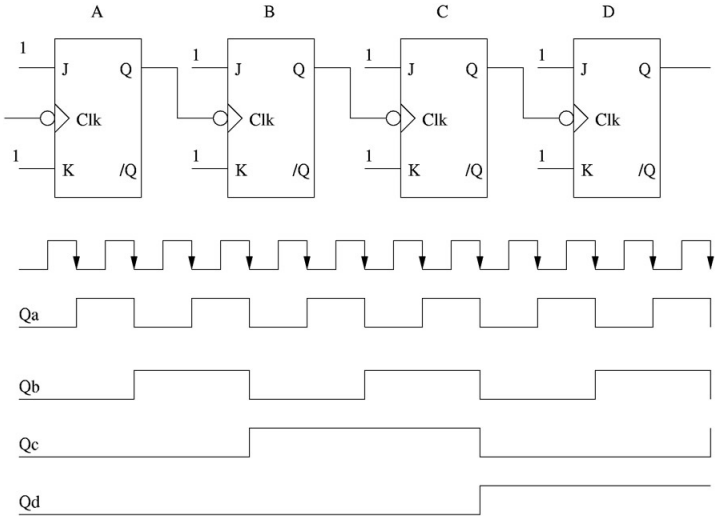
Objetive

- Entender división de frecuencias para contadores
- Entender contadores de diferentes módulos
- Entender contadores asincrónicos y paralelos

Reference

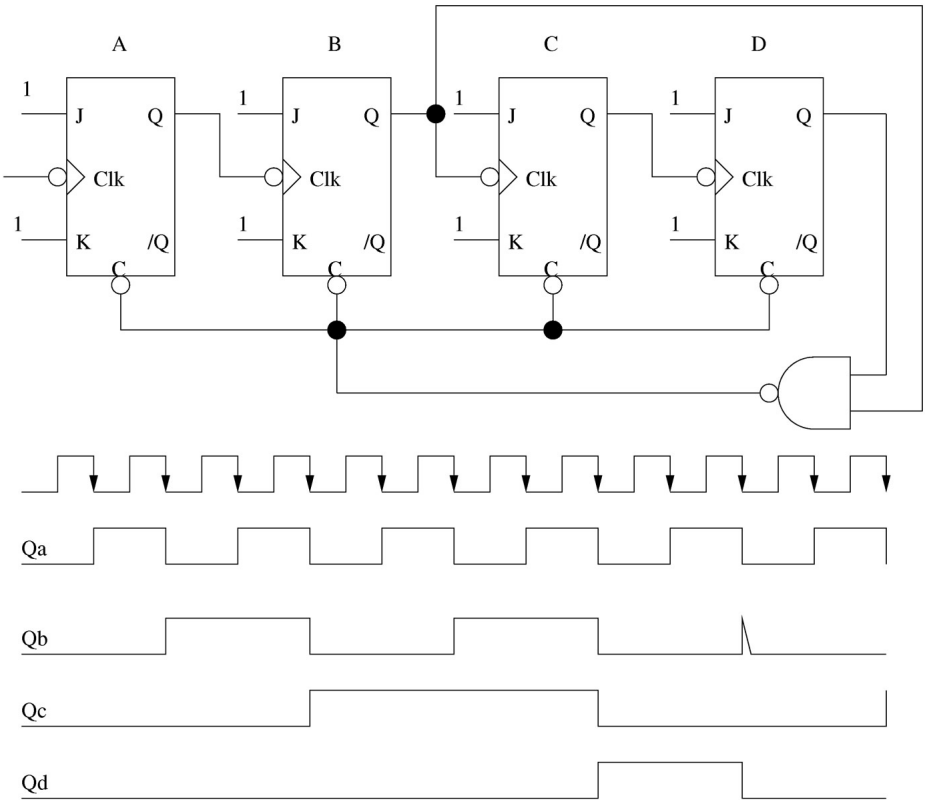
Digital Systems, Ronald Tocci, Cap. 7

Divisor de frecuencia



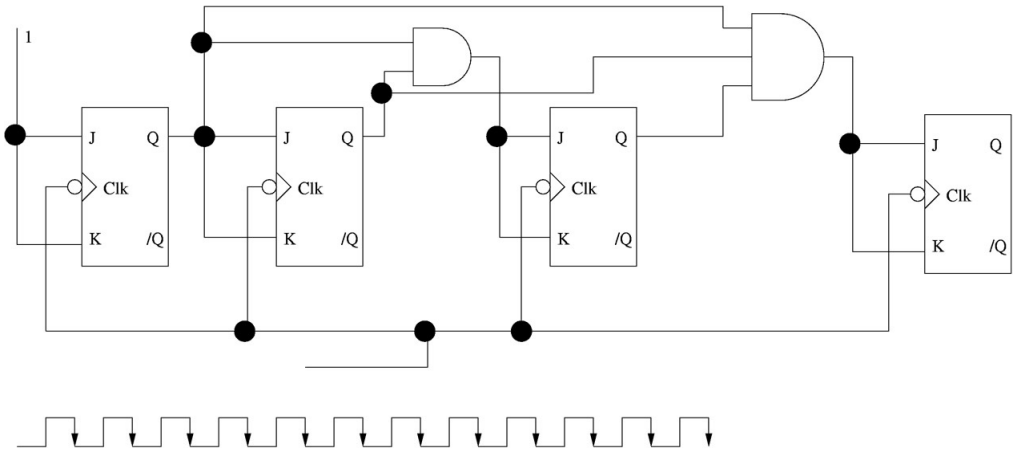
El período de oscilación de la salida de cada f-f es doble de la salida del f-f anterior en la cadena. Si se toman un corte vertical a las salidas se obtiene un contador binario. En este caso, con 4 f-f's el contador llega hasta 16: se llama un contador módulo 16. Este tipo de contador es un contador asincrónico, o 'ripple' por que los pulsos (y los cambios de estado) se propagan secuencialmente a lo largo del circuito. Los cambios en estado de cada f-f no son sincronizados.

Contador módulo 10



El contador de módulo != 2^n se efectúa utilizando las entradas no-sincrónicas de los f-f. Las salidas (Q) de los f-f se ingresan a una compuerta NAND, y la salida del NAND entra a la entrada 'CLR' de los f-f. Cuando la salida de los f-f equivale al número deseado, la salida del NAND será 0, y consecuentemente se activa el CLR de los f-f, y el conteo se inicia de nuevo desde 0.

Contador sincrónico

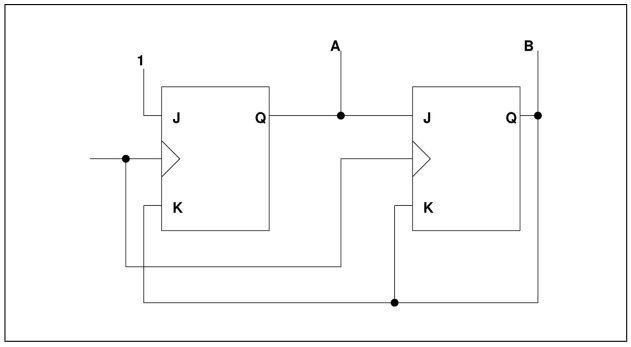


Con este circuito los intercambios de estado de las salidas de los f-f están sincronizados porque las entradas del reloj son en un solo pulso. El cambio de estado de las salidas de los f-f está habilitado cuando todos los f-f anteriores están en 1. Sino, las entradas J-K están en cero, y los f-f se mantienen en su estado anterior.

Nótese la diferencia con el divisor de frecuencia, o el contador módulo 10 en que las entradas J y K se mantienen arriba, y la salida de cada f-f es la entrada de reloj al f-f siguiente. En este caso las entradas J y K se mantienen en cero (sin cambio a la salida) salvo cuando el AND de las salidas anteriores esté en uno. Esto genera un intercambio (toggle) de la salida del f-f.

Exercises

- Diseñar un contador módulo N, es decir un contador en que el usuario puede elegir hasta que número cuenta.
- Elegir un cristal de frecuencia conocida y calcular de que manera podría obtener de ella una frecuencia de 9600Hz.
- Que salidas tiene este circuito? Es un contador?



Chapter 10

Registros

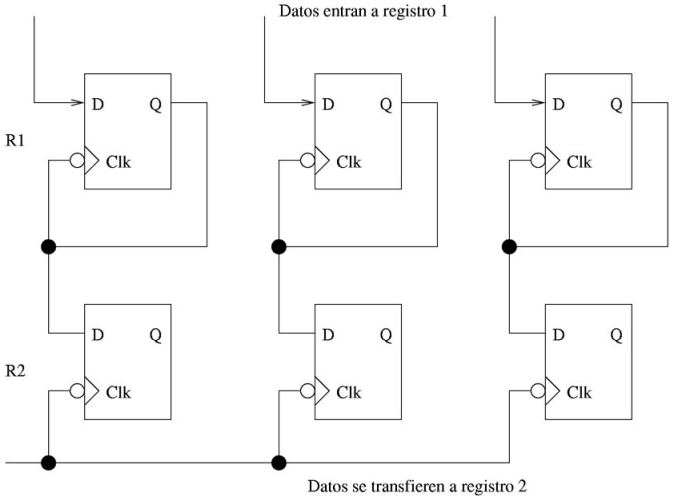
Objetive

- Entender como registros pueden almacenar y transferir datos
- Diferenciar entre registros paralelos y seriales

Reference

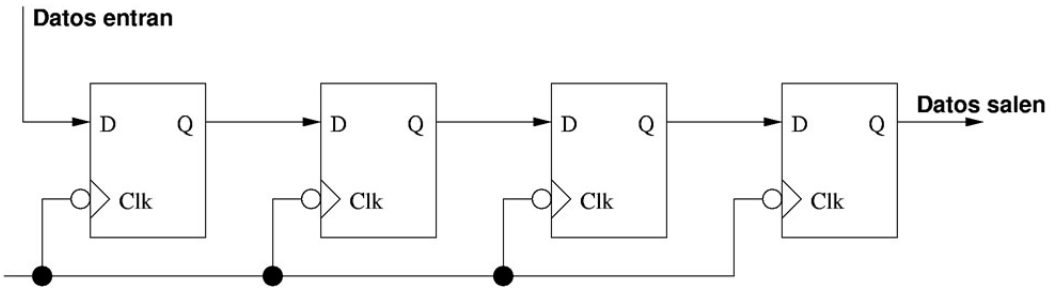
Digital Systems, Ronald Tocci, Capítulo 5-[17,18,19]

Registro de transferencia paralelo



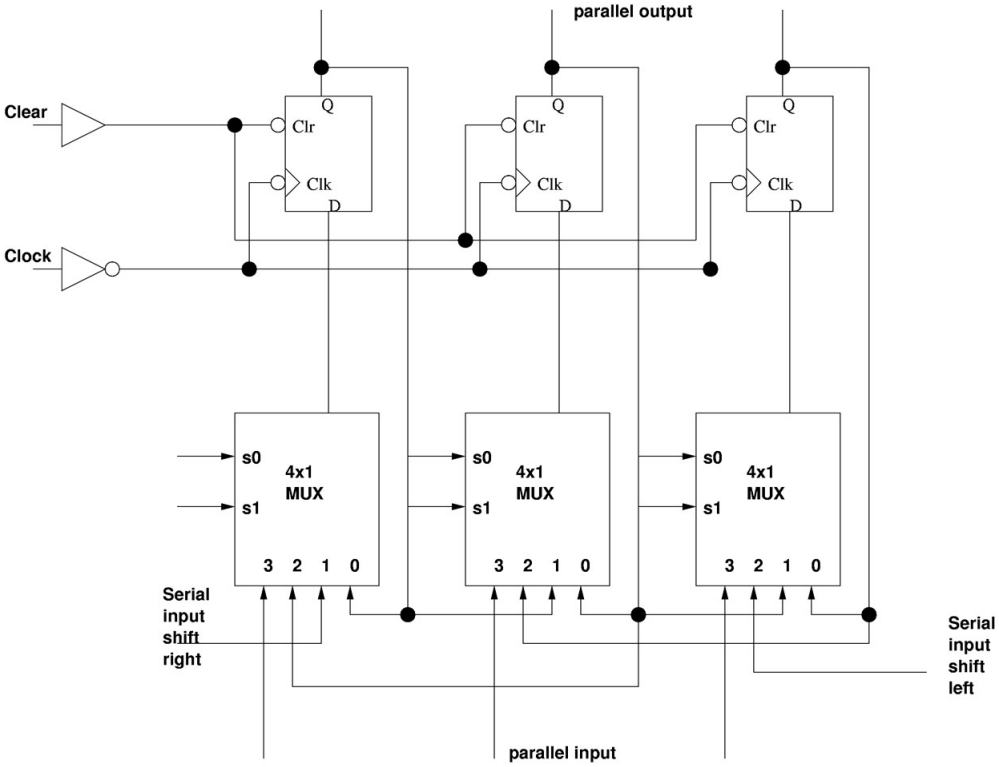
Cuando se pulsa el reloj, los datos se transfieren desde el registro superior al registro inferior.

Registro de transferencia serial, escurrimiento



Cuando se pulsa el reloj, los datos se transfieren desde el un f-f al f-f adyacente.

Registro de transferencia serial bidireccional, carga paralelo



De ida y vuelta, izquierda o derecha

Las comunicaciones seriales tradicionales en PC's ocupaban UART's (Universal Asynchronous Receiver-Transmitter), por ejemplo el 8250, 16450, 16550, 16550A. Bluetooth también ocupa UART's. Microprocesadores modernos frecuentemente incorporan UART's, por ejemplo los PIC.

El propósito de la UART es transformar un conjunto de bits en formato paralelo a un secuencia de bits en formato serial. Contiene un registro paralelo / serial. Una USART hace lo mismo, pero agrega una línea con reloj para sincronizar los dos extremos de la línea de datos.

Una comunicación seriales normales envían un bit indicando inicio del byte (start bit), un conjunto de bits para el byte de datos en formato LSB primero, después un bit opcional de paridad, y termina con un bit de término. El valor de start bit debe ser el contrario al estado normal de la línea, mientras el stop bit debe ser el estado normal. Por ejemplo, si la línea normalmente está con cero voltios, el start bit debe ser alto. El nivel de voltaje es establecido por otro chip, por ejemplo el MAX232.

Para evitar confusiones, es conveniente expresar la tasa de comunicaciones en términos de bits por segundo, y no ocupar 'baud'. Este último se refiere a cuantas transiciones en estado (por ejemplo, voltaje) hay por segundo. Sistemas modernos de comunicación codifican multiples bits en cada transición para poder aumentar los bits por segundo sin modificar el baud.

10.1 Memories

Celda La unidad básica que almacena un bit. Puede ser un flip-flop, un capacitor, un toroide magnético, un punto sobre el CD

Palabra La unidad básica de direccionamiento de memoria compuesto de un conjunto de celdas. Tipicamente va desde 8 bits hasta 64 bits.

BigEndian / LittleEndian Ordenamiento de los bytes en una palabra. Internet, Motorola son BigEndian, Intel es LittleEndian.

Ejemplo de 'Leo Rojas 21 4', Nombre, edad, sala...

BigEndian

Dirección	Valor			
bytes:	0	1	2	3
0	L	e	o	
4	R	o	j	a
8	s			
12				21
26				4

LittleEndian

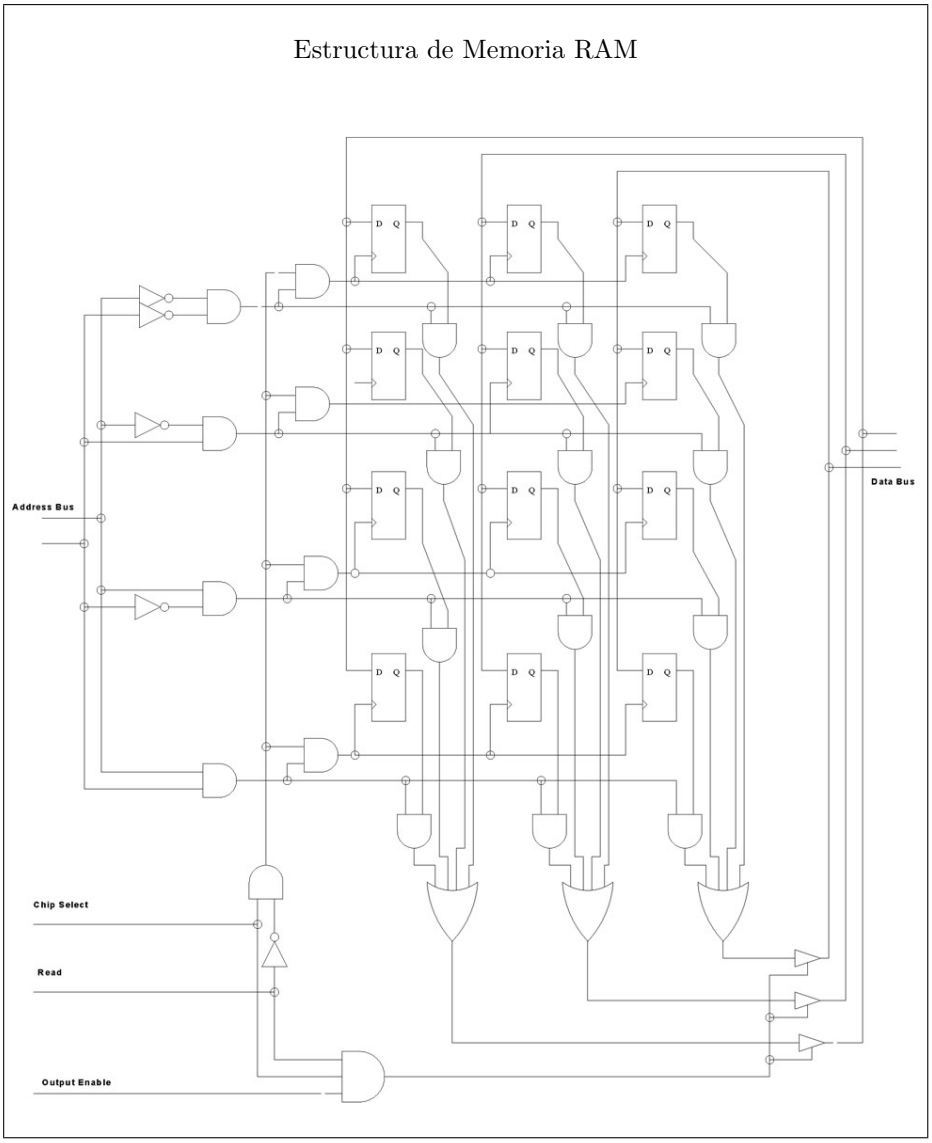
Dirección	Valor			
bytes:	3	2	1	0
0		o	e	L
4	a	j	o	R
8				s
12				21
26				4

Densidad Especificación de la capacidad de almacenamiento de memoria. Tipicamente se refiere al número de palabras por número de bits por palabra. Por ejemplo, 32K x 16 significa 32 * 2^{10} palabras y que cada palabra tiene 16 bits.

10.2 Estructura de memorias

Memoria puede considerarse como un conjunto de registros organizados como palabras, en filas y columnas. Se puede especificar una dirección que se decodifica para habilitar lectura o grabación de una fila de memoria. Los datos a grabar o leídos entran o salen por un bus de datos (a la derecha en el dibujo). El bloque de memoria (o chip) está habilitado por una línea llamada 'chip select'. Cuando el chip no está habilitado, no se puede leer ni grabar la memoria. Cuando lectura

está seleccionado, el chip está seleccionado y se habilita la salida, entonces se puede leer las filas de la memoria. Para ello, cada fila es habilitado por la dirección decodificada.



10.3 Operaciones sobre memorias

10.3.1 Write

1. CPU coloca dirección de memoria sobre bus de direcciones
2. CPU coloca datos sobre bus de datos
3. CPU activa las líneas correspondientes para grabación

4. IC's de memoria decodifican dirección para determinar que palabras se seleccionen.
5. Datos sobre el bus de datos se transfieren a la ubicación seleccionada.

10.3.2 Read

1. CPU coloca dirección de memoria sobre bus de direcciones
2. CPU activa las líneas correspondientes para lectura
3. IC's de memoria decodifican dirección para determinar que palabras se seleccionen.
4. IC's de memoria colocan los datos al bus de datos para la CPU.

10.4 ROM, SRAM, DRAM, EEPROM, Flash

Read Only Memory Procesadores chicos de sistemas embutidos fabricados en gran volmenes tienen sus programas quemados en ROM en el procesador durante la fabricación del chip. Un MOSFET sirve para cada celda.

PROM, EPROM, EEPROM ROM's que el usuario puede programar, cuando se aplica un voltaje alto a las conexiones a cero del source del MOSFET se quema la conexión, y así dejar la celda en alto. Erasable PROM's. Se pueden borrar, tipicamente con luz ultravioleta. El circuito tiene una ventanita transparente. Estos circuitos eran los más populares para aplicaciones de pocas unidades. Electrically Erasable PROM. Puede borrar palabras con una corriente eléctrica, no necesariamente con ultravioleta. Esto significa que lo puedes borrar dentro del circuito de aplicación.

Flash Estructuralmente es semejante al EPROM, pero el 'gate' tiene una capa de óxido más delgado, por lo que puede borrarse eléctricamente dentro del circuito de aplicación.

DRAM Dynamic RAM. SRAMs (memoria estática) guarda el dato en flip-flops. Celdas DRAM son capacitores, que gradualmente pierden su carga, por lo que tiene que refrescarse periodicamente. Los capacitores tiene un amplificador para detectar la carga sobre el capacitor. Son más pequeñas y consumen menos energía, pero no tan rápidas. Memoria principal de los PC's es DRAM. Cache segundario es SRAM, alguna memoria de video también.

En DRAMs actuales, las direcciones son multiplexados. Dos líneas, CAS (Column Address Select) y RAS (Row Address Select) reemplazan el CS de las memorias más simples. Por ejemplo, una DRAM de 16M x 1 requiere 24 bits de direccionamiento en forma simple ($s^{24} = 16M$). Con multiplexado requiere 12 bits (en 2 pasos, primero para columnas y después para filas), require también CAS y RAS, pero no requiere CS. Es un total de 11 bits menos.

FPM DRAM Fast Page Mode DRAM permite acceso más rápido a memoria en la misma 'página' de acceso anterior porque solo los bits menos significativos cambian

EDO DRAM Extended Data Output DRAM es un FPM DRAM con un 'latch' después del amplificador que permite CAS cambiar su valor cuando el latch recibe el valor de la celda.

SDRAM Synchronous DRAM lee lecturas de datos secuenciales mediante el reloj del bus, no CAS. La primer lectura es la misma velocidad que DRAM normal, pero las lecturas secuenciales son mucho más rápidas

DDRSDRAM Double Data Rate SDRAM, opera con el reloj del sistema, pero realiza una lectura en el flanco de subida y una en el flanco de bajada

SLDRAM Synchronous Link DRAM, una mejora al DDRSDRAM. Funciona con relojes de bus hasta 200MHz.

DRDRAM Direct Rambus DRAM, integra un bus en el módulo de memoria.

Exercises

- Desarrollar las señales necesarias para control el registro de carga paralelo y descarga serial.
- Definir los circuitos necesarios para calcular la paridad en una UART.
- Definir los circuitos necesarios para determinar el start y stop bits en una UART.
- Cual es mejor, Little Endian, o Big Endian?
- Que diferencias hay entre FLASH y EPROM?
- Como podía ser usado un anillo magnético para memoria?
- Un bloque de memoria contiene 2048 palabras de 16 bits. Cuantas líneas de control, entrada y salida requiere.
- Cual es más conveniente: un bloque de memoria de 1024 palabras de 32 bits, o 2048 palabras de 16 bits?
- Como funciona el 'pendrive'?
- Que es 'flash puppy'?
- Que es una 'MMU', Memory Management Unit?