

Sistemas Digitales

Índice

Tema 1: Introducción a los Sistemas Digitales Electrónicos [Tema 1 : Introducción a los Sistemas Digitales Electrónicos](#).

Tema 2: Representación Digital de la Información [Tema 2 : Representación Digital de la Información](#).

Tema 3: Algebra de Boole. Puertas Lógicas.[Tema 3 : Algebra de Comutación ó Boole. Funciones Lógicas](#).

Tema 4: Lenguaje de Descripción Hardware VHDL.

Tema 5: Circuitos Aritméticos.

Tema 6: Otros Circuitos Combinacionales.

Tema 1 : Introducción a los Sistemas Digitales Electrónicos

- Presentación
 - Profesor
 - Calendario
 - Sistemas Digital: TAC
 - Electrónica
 - Procesamiento de señales eléctricas
- Organización Académica
 - Programa
 - Prácticas
 - Ejercicios
 - Evaluación
 - Metodología

Profesorado

- Prof. Cándido Aramburu Mayoz.
 - Doctor Ingeniero Telecomunicación (UPNA-Universidad Politécnica de Madrid)
 - Empresa Ikusi S.A. (Sistemas de Telemedida 1989)
 - Profesor Titular UPNA (Dpto Ingeniería Electrónica y Comunicaciones 2000)
- Profesor Prácticas: Aitor Urrutia
- Profesor Euskera: Marko Galarza
- Profesor Inglés: Ignacio del Villar
- <https://www.etsit.upm.es/>
- <https://www.velatia.com/es/empresas-que-forman-velatia/ikusi/>
- <https://www.unavarra.es/eu/sites/Portada/home.html>

Contacto

- Despacho: Edificio Los Tejos 2 Planta: Despacho 2028 (Prof. Candido Aramburu)
- Miaulario → correo interno
- Clase
 - G1: A112 : Lunes (12-14) y Jueves (10-12)
 - G2: A012 : Lunes (10-12) y Miércoles (12-14)
- Tutorías
 - Lunes (14-17) y Miércoles (9-12)
 - Cita Previa

Calendario

Febrero

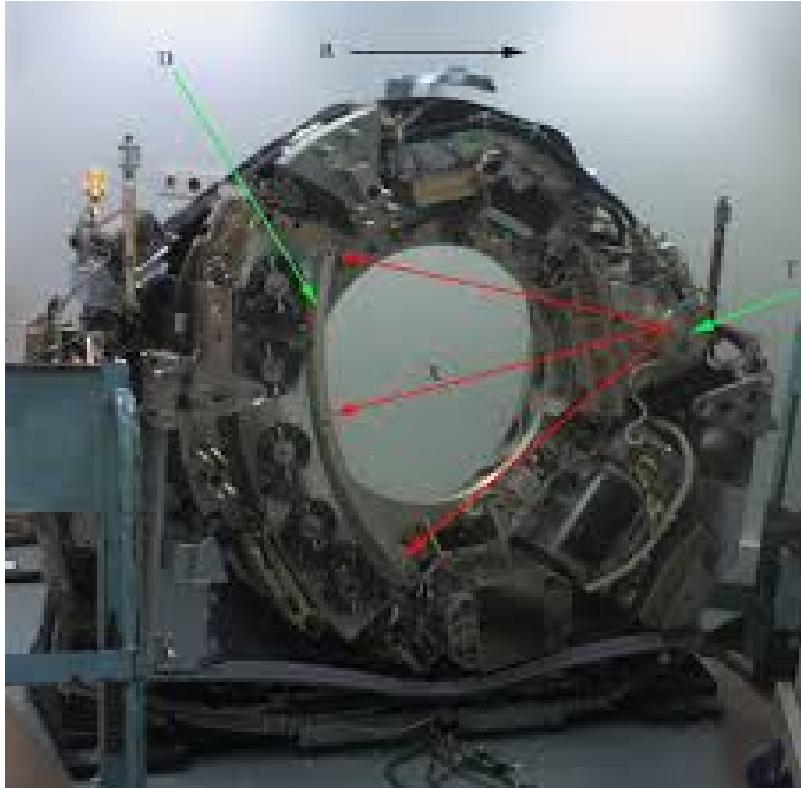
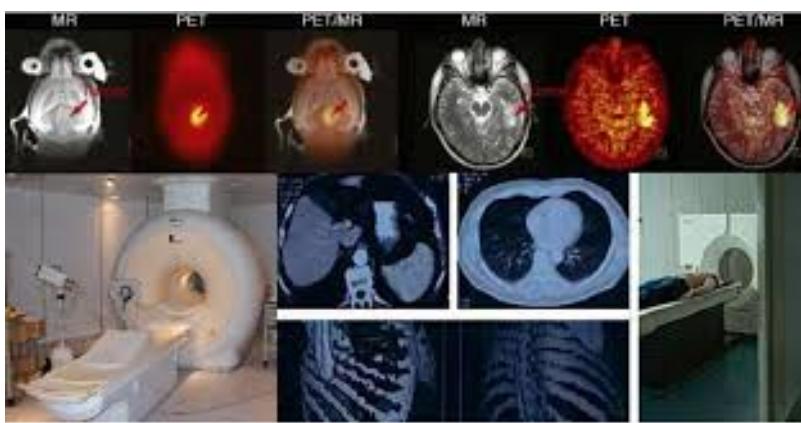
Lunes	Miércoles
30-Presentación.-Repre_info: bases	1-Repre_Info:Enteros.C2:Expansion Signo
6-C2:overflow.Boole:Morgan	8-Boole:Morgan.DK:examples
13	15
20	22
27	

Marzo

Lunes	Miércoles
	1
6	8
13	15
20	22
27	29

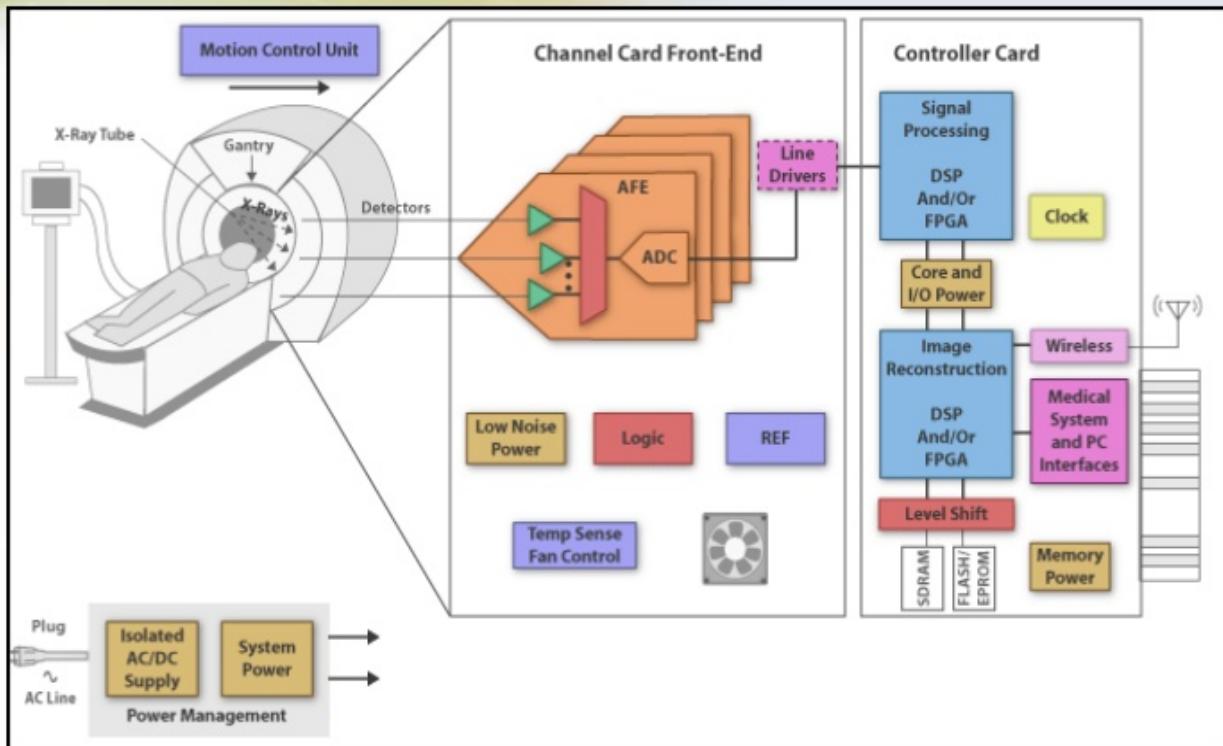
Sistemas Digitales

Tomografía axial computarizada



Sistema Digital

Solution From TI for CT Scanner



Entrada **Análogica** → Sensores Magnéticos.

Conversor A/D: Señal Analógica a Señales Digitales.

Circuitos **Lógicos**: multiplexores, filtros, codificadores, etc ...

Procesadores lógicos: procesamiento de las señales digitales para obtener la imagen.

FPGA : Field Programming Gate Array.

DSP : Digital Signal Processing.

CPU : Centra Procesor Unit.

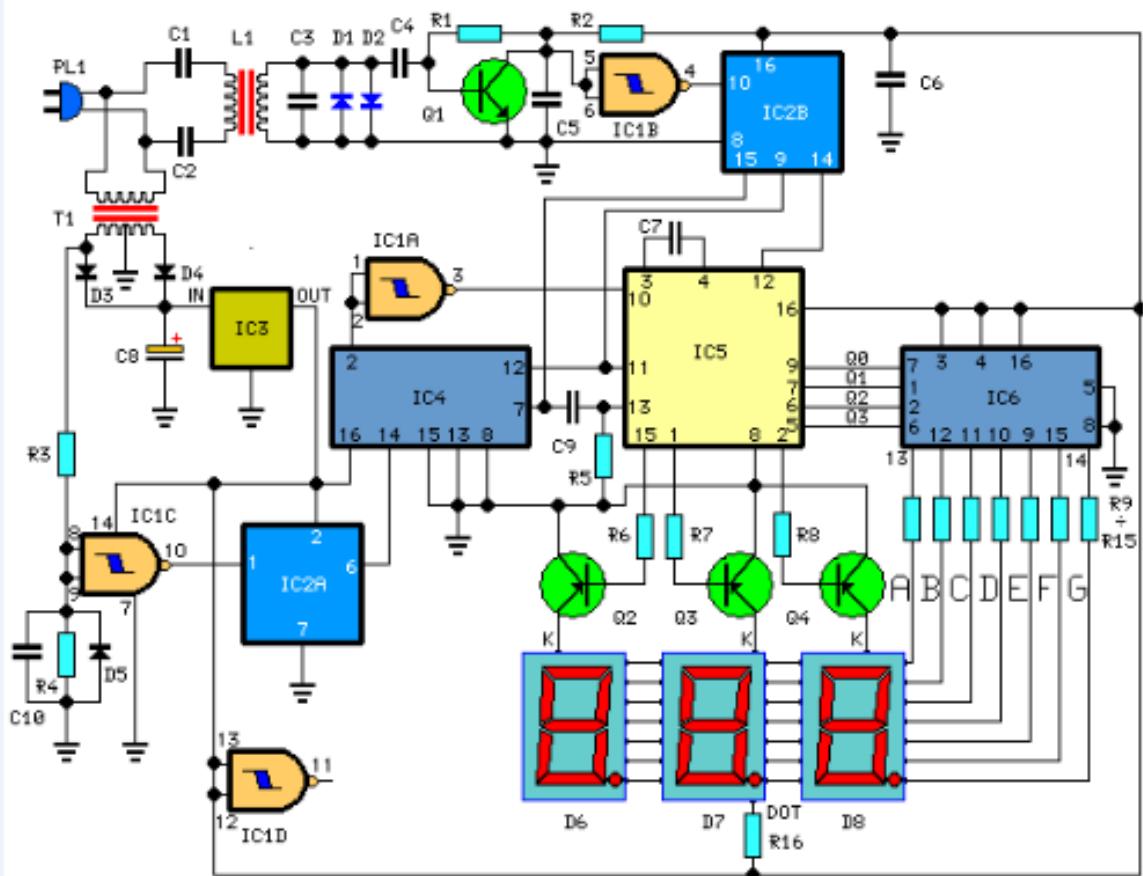
GPU : Graphic Procesor Unit.

Electrónica

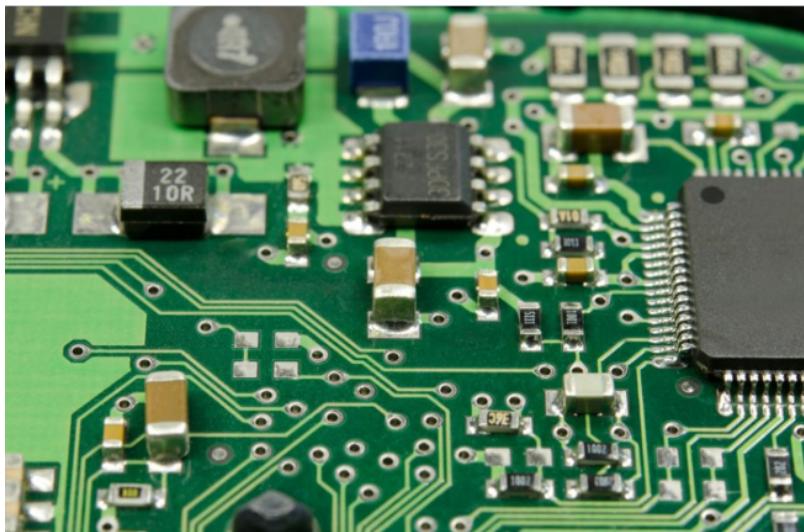
Equipos de Electrónica



Esquema Eléctrico



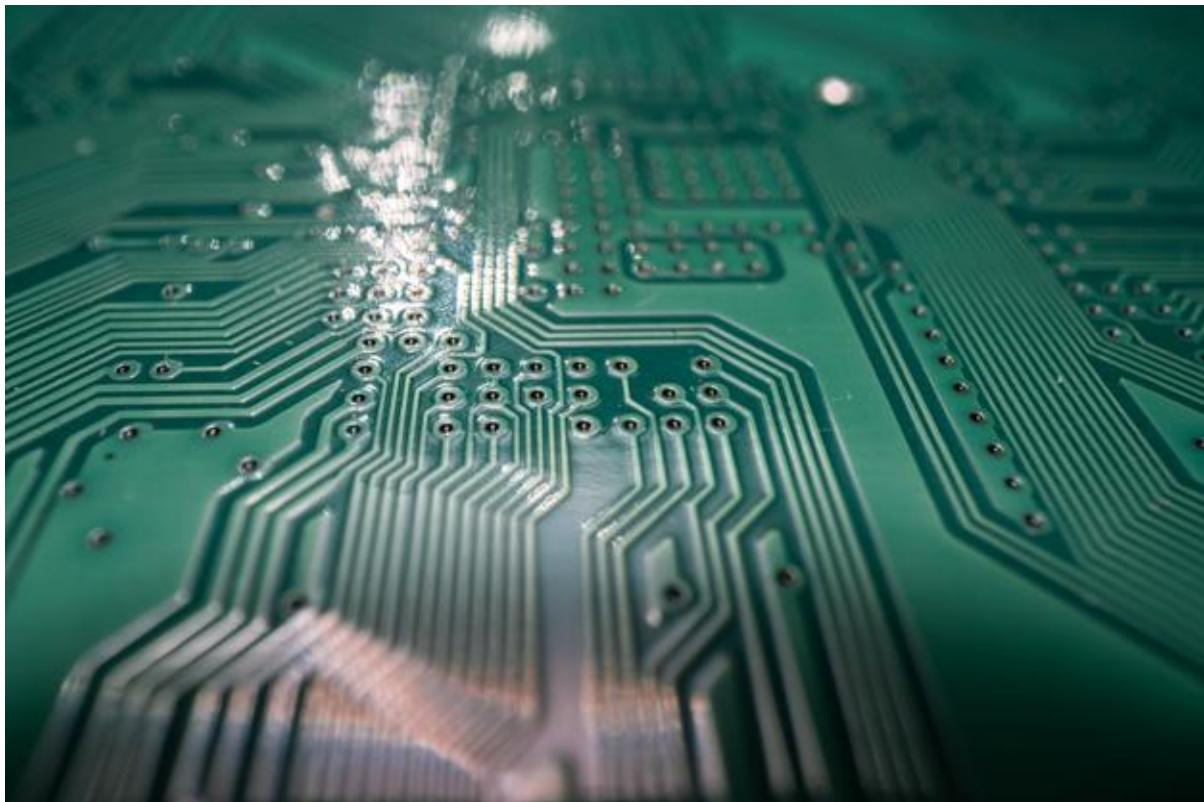
Componentes de una tarjeta de circuito impreso



Componentes:

- Discretos: resistencias, condensadores, transistores, transformadores, etc.
- Integrados ("chips", microelectrónica).
- material de semiconductor: Silicio.
- el componente básico es el transistor → un procesador puede tener cientos de millones.
- los microcircuitos hechos de transistores pueden ser tanto circuitos analógicos (un amplificador) como digitales (puertas lógicas)

Printed Circuit Board



Instrumentación



La Electrónica en la Profesión

Digital Electronics Engineer

Ref No.	EMP439757
Location	Twickenham, London, England
Job type	Permanent
Job Status	Closed

You can not apply for this job as its status is **Closed**.

 Save  Email

Share:   

Introduction

Are you interested in working for a continuously growing company that really value their staff and offer a fantastic range of benefits?

Important

Digital electronics, FPGA, VHDL/Verilog, DSP, PCB

The Job

This opportunity is for a Senior Electronics Engineer to join a team that have tripled in size over the last 10 years due to continued success. The company design and manufacture sensors that use state-of-the-art acoustic resonance technology, and are distributed internationally across a number of industries including marine and defence.

Técnico



¿ Profesiones relacionadas con la Electrónica?

Fases de Diseño de Circuitos Electrónicos Binarios

1. Funcional (manual): abstracción matemática

2. Automatización del proceso matemático
 - a. Herramientas de Diseño con ayuda del Computador (EDA)
 - b. Simulación del Diseño del Circuito Electrónico antes de fabricar el prototipo: Depuración
3. Fabricación del prototipo
 - a. Instrumentación
 - b. Verificación del funcionamiento en el Laboratorio
 - c. Verificación del funcionamiento en Campo
4. Comercialización
5. Producción

La Electrónica en la Carrera Universitaria

- Conocimientos de Electrónica
 - ¿ Para ?
 - Tecnología Hardware
 - Fabricación de Prototipos
 - Diseño de Prototipos : Conceptos Teóricos y Herramientas de diseño por computador
 - Desarrollo de Sistemas: Equipos, Plataformas
 - Comercialización
 - Usuario: Equipos, Plataformas

Representación Científica y Prefijos de las Unidades

Table 1. Prefijos

Prefijos	Tera	Giga	Mega	Kilo	mili	micro	nano	pico
Base 10 → magnitudes:m,gr,Hz, ..	10^{12}	10^9	10^6	10^3	10^{-3}	10^{-6}	10^{-9}	10^{-12}
Base 2 → magnitudes: Byte	2^{12}	2^9	2^6	2^3	2^{-3}	2^{-6}	2^{-9}	2^{-12}

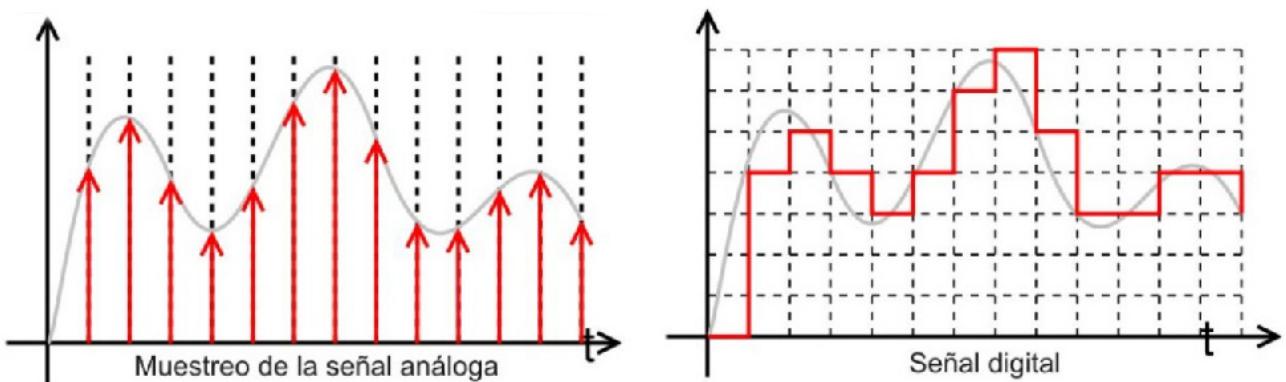
- Ejemplo: representar la magnitud=1000000000Hz debidamente
 - Notación científica → 10^9 Hz
 - Debidamente: Notación científica con prefijos $f=1\text{GHz} \rightarrow T=1/f=10^{-9}\text{seg}= 1\text{ns}$

Señales: Conversión Analógica Digital

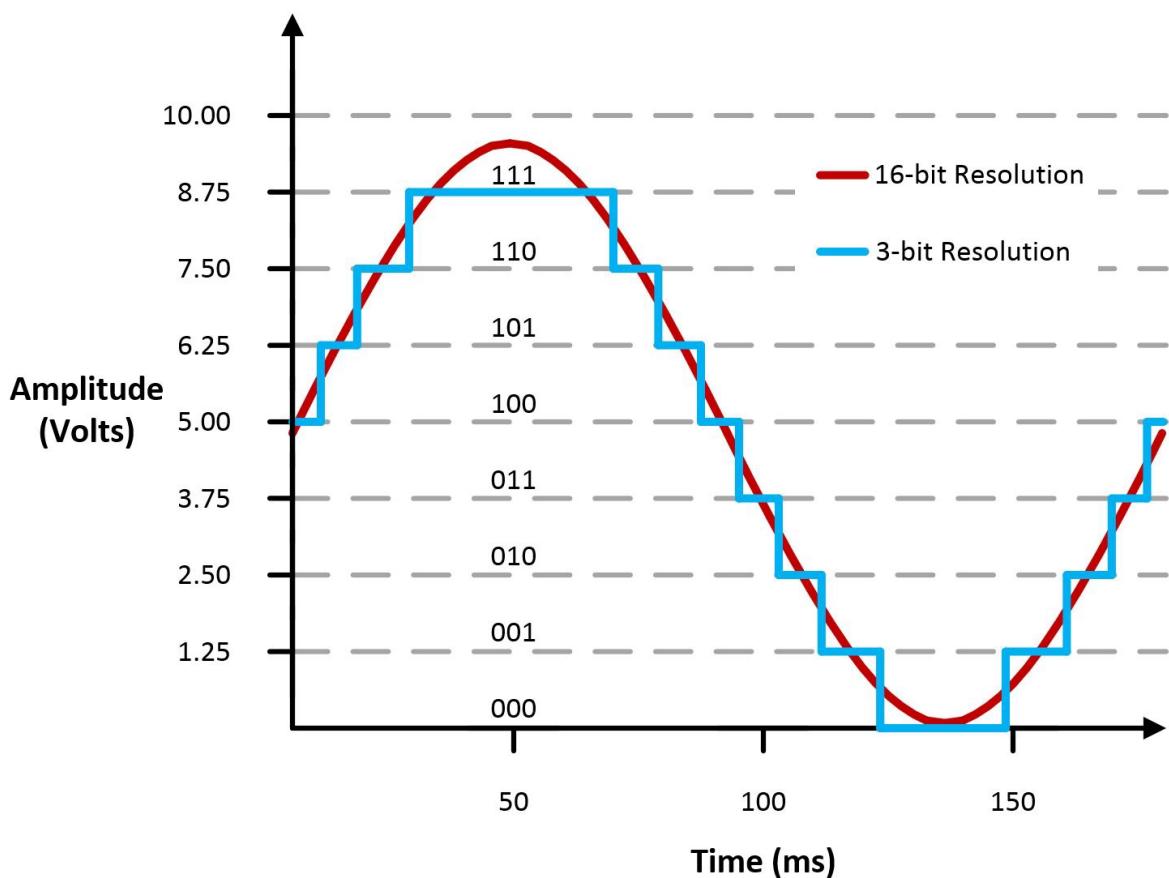
Analogica vs Digital

- Señal Continua
 - Amplitud: *oo* valores posibles en el rango
 - Tiempo: *oo* valores posibles en el rango
- Señal Discreta ó Digital
 - Amplitud: finitos valores posibles en el rango
 - Tiempo: finitos valores posibles en el rango

Señales : Muestreo y Cuantificación



Codificación



Calcular para las resoluciones de 3 bit y 16 bits cual es el mínimo incremento de señal codificable o error de cuantificación: con 3 bits el número de niveles es $2^3=8$ niveles y el mínimo relativo es $2^{-3}=1/8$; con 16 bits el número de niveles es 2^{16} y el mínimo relativo es $2^{-16}= 1/65536$.

Representación de los números en código binario : [Tema 2 : Representación Digital de la Información](#)

Señales Binarias : Abstractas

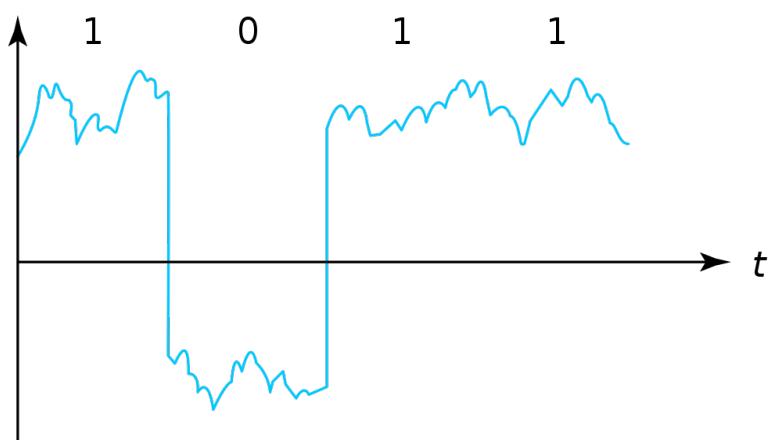


Eje ordenada: valores abstractos (0/1, High/Low, ON/OFF, etc ...)

Cronograma: Representación temporal de las señales digitales binarias.

Esa representación típica de los libros de texto, pizarra de clase, etc ... es ideal ya que físicamente siempre habrá distorsión.

Señales Binarias : Físicas



Eje ordenada: magnitudes físicas (mV ó mA).

La señal física está distorsionada por causas como pej: línea larga de transmisión (efectos capacitivos e inductivos).

Un ejemplo típico de distorsión son los tiempos de subida y bajada, que no son nulos sino del orden de unos nanosegundos.

La distancia considerable entre los dos niveles (binario) a la entrada del receptor hace fácil la discriminación entre el '0' y el '1'.

Digitalización de las Señales

Ventajas

- Calidad: Fácil de recuperar a pesar de la distorsión
- Almacenamiento: Fiabilidad, Diversidad Formatos

- Compatibilidad: Diversidad de Equipos (PC, móvil, coche, etc)
- Procesamiento: Sencillo, Flexible
- Coste: Barato (componentes)

Abstracción

- Niveles: el 0 y el 1
- Lógica binaria
 - Matemáticas: Algebra de Boole

Organización Académica

Programa de la Asignatura

- [Ficha Web Upna](http://www.unavarra.es/ficha-asignaturaDOA/?languageld=100000&codPlan=246&codAsig=246110&anio=2022) [<http://www.unavarra.es/ficha-asignaturaDOA/?languageld=100000&codPlan=246&codAsig=246110&anio=2022>]
- Programa en 3 partes
 - i. **Circuitos Combinacionales**
 - ii. **Circuitos Secuenciales**
 - iii. Otros: Números, Lógica Programable (VHDL), Teoría Tecnología
- Bibliografía

Prácticas

- Tipo de prácticas:
 - Diseño manual
 - Simulación con la herramienta software Quartus de Intel.
 - Captura gráfica de Esquemas Electrónicos
 - Descripción del Circuito mediante el Lenguaje VHDL. Fabricación del Circuito en tecnología FPGA

Ejercicios

- Tipo de problemas: Libro Verde → Ejercicios tipo examen → Sin calculadora y sin libros

El libro verde se adquiere en el edificio de rectorado, en la sección de comunicación,
que se encuentra en planta baja del edificio.
El horario: 8 a 14:30. Precio 8.5\$.

- * Capítulo 1: 1.1, 1.2, 1.4, 1.5, 1.6, 1.8, 1.9
- * Capítulo 2: 2.1
- * Capítulo 3: 3.2 3.3 -> 2º parcial
- * Capítulo 4: 4.2, 4.4, 4.6
- * Capítulo 5: 5.2, 5.3, 5.4
- * Capítulo 6: 6.1, 6.2 -> 2º parcial

- * Capítulo 7: 7.2, 7.3 y 7.4 -> 2º parcial
- * Capítulo 8: 8.1, 8.3 y 8.5 -> 2º parcial

- Los ejercicios del tema 2 (Representación de la Información) no están en el libro verde
 - Miaulario → Recursos → Ejercicios

Evaluación

- Sistema de Evaluación:
 - 75% teoría y 25% prácticas
 - Evaluación continua Teoría: dos parciales (30% 1º parcial y 45% 2º parcial). Nota mínima en el 2º parcial: 5. El Primer parcial se realizará el sábado 25 de Marzo a las 9:00, el segundo parcial el 24 de Mayo a las 8:00 y la recuperación el 12 de Junio a las 8:00
 - Recuperación Teoría: Entra todo. Nota mínima: 5.
 - Evaluación Prácticas: Un único examen el sábado XX de Mayo, no recuperable.

Metodología

- Trabajo en clase: principalmente Ejercicios con su teoría asociada
- Trabajo en casa
 - Teoría desarrollada en los apuntes PDF en mi aulario
 - Prácticas
 - En casa: Ejercicios de diseño manual
 - En casa: Utilización de Quartus y Memorias
- Tutorías
 - Resolución de dudas

Tema 2 : Representación Digital de la Información

Índice

- Información: números, caracteres, imagen, sonido, etc ..
- Números
 - Sistemas posicionales: base 10 (decimales), base 2 (binaria)
 - Naturales: bases 10,2,8,16 . Conversión entre bases
 - Enteros: Signo Magnitud, Complemento a la base-1, Complemento a la base
 - Operaciones aritméticas: Suma,Resta
 - Reales: coma fija y coma flotante
- Caracteres
 - Alfanuméricos y Signos de Puntuación
 - ASCII standard y extendido

- Unicode: UTF-8

Representación de los Números

Representación de los Números Decimales

- Decimal
 - 10 dígitos : 0,1,2,3,4,5,6,7,8,9
 - Pesos con base 10 : 10^n donde n es la posición del dígito dentro del número
- Ejemplo: número 5421

Table 2. Número 5421

Representación:	los símbolos 5421			
Posiciones:	3	2	1	0
Pesos:	$10^3 \rightarrow 1000$	$10^2 \rightarrow 100$	$10^1 \rightarrow 10$	$10^0 \rightarrow 1$
Dígitos:	5	4	5	1
Valores : ponderación	$5 \cdot 1000 =$ cinco mil	$4 \cdot 100 =$ cuatrocientos	$5 \cdot 10 =$ cincuenta	$1 \cdot 1 =$ uno
Valor:	$5 \cdot 1000 + 4 \cdot 100 + 5 \cdot 10 + 1 =$ cinco mil cuatrocientos cincuenta y uno			

Representación de los Valores Enteros en Código Binario

- ¿Número? ¿Valor? ¿Código? ¿Representación?
 - 2 dígitos : 0,1
 - Pesos con base 2 : 2^n donde n es la posición del dígito dentro del número:-1024-512-256-128-64-32-16-8-4-2-1...
- Ejemplo: número 0b1011

Table 3. Número 0b110011

Representación:	los símbolos 1011			
Posiciones:	3	2	1	0
Pesos:	$2^3 \rightarrow 8$	$2^2 \rightarrow 4$	$2^1 \rightarrow 2$	$2^0 \rightarrow 1$
Dígitos:	1	0	1	1
Valores : ponderación	$1 \cdot 8 =$ ocho	$0 \cdot 4 =$ cero	$1 \cdot 2 =$ dos	$1 \cdot 1 =$ uno
Valor:	ocho+cero+dos+uno= once			

Representación de los Valores Enteros en Código Binario

- ¿Cómo se representa en binario el valor 123.125? b1111011.001
- ¿Cómo se calcula el valor del número binario b1111011.001?
- Parte Entera: divisiones sucesivas por la base 2
- Parte Fracción: multiplicaciones sucesivas por la base 2

Representación de los Valores Enteros en Código Octal

- Dígitos: 0,1,2,3,4,5,6,7
- Posiciones y Pesos
- ¿Cómo se representa en octal el valor 123.125? 0o173.1
- ¿Cómo se calcula el valor del número octal 0o173.1?
- Parte Entera: divisiones sucesivas por la base 8
- Parte Fracción: multiplicaciones sucesivas por la base 8

Representación de los Números en Hexadecimal

- Dígitos: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F _ el valor de A es 10, B→11, C→12, D→13, E→14, F→15
- Posiciones y Pesos
- ¿Cómo se representa en hexadecimal el valor 123.125? 0x7B.2
- ¿Cómo se calcula el valor del número octal 0x7B.2?
- Parte Entera: divisiones sucesivas por la base 16
- Parte Fracción: multiplicaciones sucesivas por la base 16

Calculadora de Python

Python Interpreter Shell [\[https://www.programiz.com/python-programming/online-compiler/\]](https://www.programiz.com/python-programming/online-compiler/)

```
bin(123)
oct(123)
hex(123)
int(0b1111011)
int(0o173)
int(0x7B)
```

Conversiones entre el sistema binario y sistemas con base potencia de 2

- Conversión Binaria-Hexadecimal
 - base 16=2⁴
 - grupos de 4 bits empezando por la dcha
 - b1111011 → 111 - 1011 → 0x7B

- Conversión Hexadecimal-Binaria
 - grupos de 4 bits
 - Conversión Binaria-Octal
 - base $8=2^3$
 - grupos de 3 bits empezando por la dcha
 - $b1111011 \rightarrow 1 - 111 - 011 \rightarrow 0o173$
 - Conversión Octal-Binaria
 - grupos de 3 bits

Suma binaria

- Suma $10011011 + 00011011 = 10110110$

```

Llevadas -->      1 1   1 1

          1 0 0 1 1 0 1 1 <--sumando
+ 0 0 0 1 1 0 1 1 <--sumando

Valor suma      1 3 2 1 3 2
*****  

Resultado -->  1 0 1 1 0 1 1 0 <--suma

```

- Cuando la suma en una posición específica tiene un valor es mayor o igual a la base hay que restar n veces la base y el valor n será la llevada a sumar en la posición siguiente.

Resta binaria

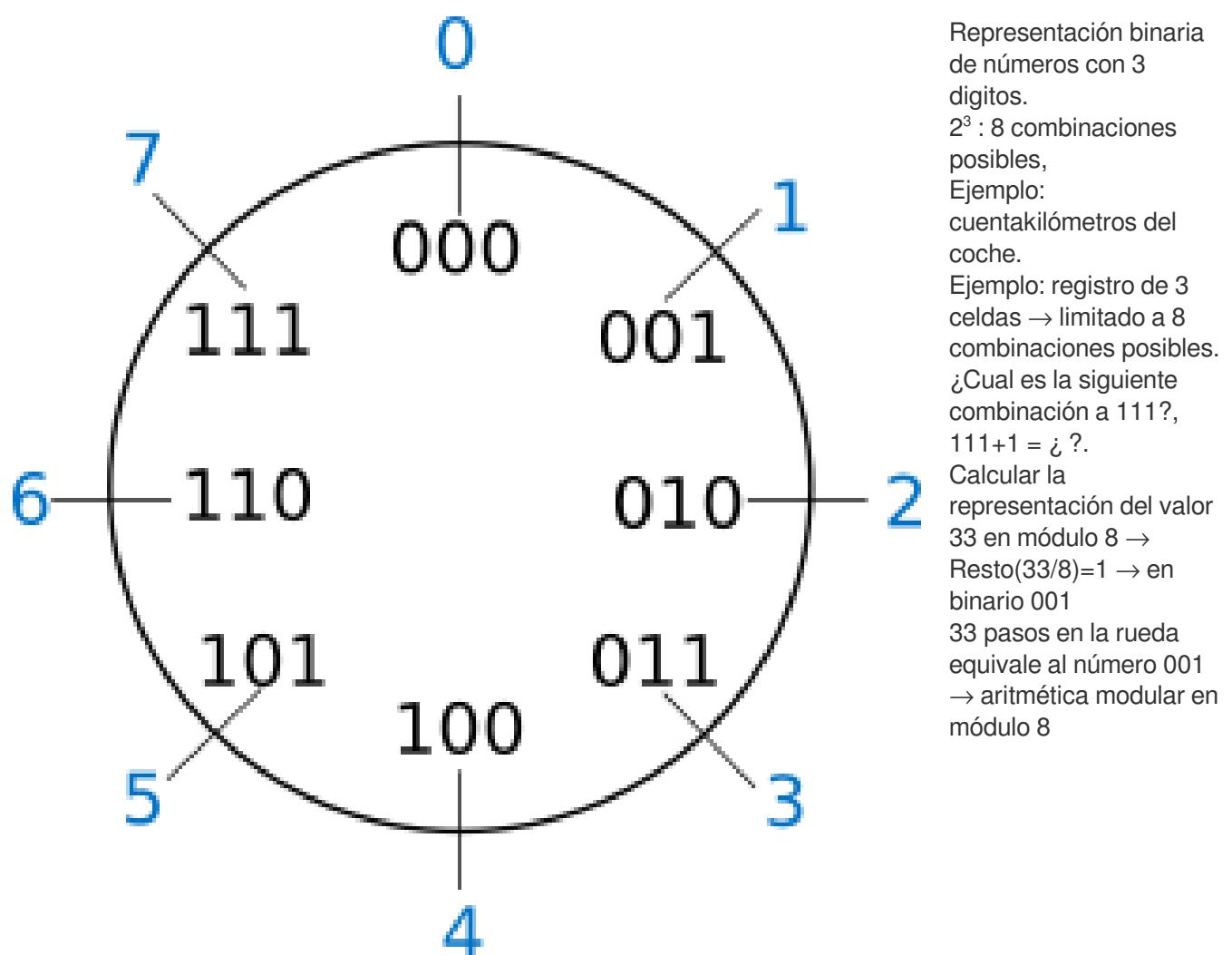
- Resta $10110110 - 10011011 = 00011011$

Sumar crédito al minuendo	2 2	2 2	
	1 0 1 1 0 1 1 0		<--minuendo
	- 1 0 0 1 1 0 1 1		<--sustraendo
Sumar llevada al sustraendo	1 1	1 1	

Resta	0 0 0 1 1 0 1 1		

- Cuando en una posición específica el minuendo es menor que el sustraendo se suma la base al minuendo antes de realizar la resta y se suma la llevada al sustraendo de la posición siguiente.

Aritmética Modular: la rueda



Representación binaria de números con 3 dígitos.

2^3 : 8 combinaciones posibles,

Ejemplo:
cuentakilómetros del coche.

Ejemplo: registro de 3 celdas → limitado a 8 combinaciones posibles.

¿Cuál es la siguiente combinación a 111?,
 $111+1 = ?$.

Calcular la representación del valor 33 en módulo 8 → Resto($33/8$)=1 → en binario 001

33 pasos en la rueda equivale al número 001 → aritmética modular en módulo 8

Operaciones aritméticas: Octal y Hexadecimal

- Base Octal
 - $0o675+0o304 = 0o1201$
 - $0o632-0o374 = 0o236$
- Base hexadecimal
 - $0xD1B+0xAF = 0x181A$
 - $0xE53-0xBA = 0x2A9$

Representación de Números con Valores Enteros

- Signo-Magnitud
- Complemento a la base menos 1
- Complemento a la base

Representación en Signo-Magnitud

- Signo → un dígito
- Base 10:
 - valores positivos: el signo el dígito 0 en la posición MSD (More Significant Digit) y resto de dígitos representa el módulo

- valores negativos: el signo el dígito 9 (base-1) en la posición MSD (More Significant Digit) y resto de dígitos representa el módulo
- Ejemplo +123 → 0123 y -123 → 9123

Representación en Signo-Magnitud

- Signo → un bit (Binary digit)
- Base 2 :
 - valores positivos: el signo el bit 0 en la posición MSB (More Significant Bit) y resto de bits representa el módulo
 - valores negativos: el signo el bit 1 (base-1) en la posición MSB (More Significant Bit) y resto de bits representa el módulo
 - Ejemplo +123 → 0b01111011 y -123 → 0b11111011
 - Dibujar la tabla y la rueda con todos los valores con sus representaciones.
 - ¿Cuántas representaciones son posibles? ¿Es simétrico el rango de valores representado? ¿Cuántas representaciones tiene el cero?
 - Extender el número de bits del número sin cambiar su valor

Representación en complemento a la base menos 1. C9

- Base 10: Complemento a 9 → C9
- Signo → un dígito
- Valores positivos: igual que los valores positivos en código Signo-Magnitud
- Valores negativos: Hay que restar el código del valor en positivo del minuendo 99999999 (base-1)
 - Ejemplo +123 → 0123 y -123 → 9999-0123 = 9876
- El C9 de un número positivo es el código de su valor en negativo
- El C9 de un número negativo es el código de su valor en positivo

Representación en complemento a la base menos 1. C1

- Base 2 : base-1=1 → Complemento a 1 → C1
- Signo → un dígito
- Valores positivos: igual que los valores positivos en código Signo-Magnitud
- Valores negativos: Hay que restar el código del valor en positivo del minuendo 11111111 (base-1)
 - Ejemplo '+123' → 0b01111011 y -123 → 11111111-01111011 = 10000100
 - El código del valor negativo se puede calcular invirtiendo los bits del código del valor positivo
- El C1 de un número positivo es el código C1 de su valor en negativo y del de un número negativo es el código C1 de su valor en positivo
 - Dibujar la tabla y la rueda con todos los valores con sus representaciones.
 - ¿Cuántas representaciones son posibles? ¿Es simétrico el rango de valores representado? ¿Cuántas representaciones tiene el cero?
 - Extender el número de bits del número sin cambiar su valor

Representación en complemento a la base 10 : C10

- Signo → un dígito
- Base 10: Complemento a 10 → C10
- Valores positivos: igual que los valores positivos en código Signo-Magnitud
- Valores negativos: Hay que restar el código del valor en positivo del minuendo 0000000 (base)
 - Ejemplo '+123' → 0123 y -123 → 0000-0123 = 9877
- El C10 de un número positivo es el código de su valor en negativo
- El C10 de un número negativo es el código de su valor en positivo

Representación en complemento a la base 2 : C2

- Signo → un dígito
- Base 2: Complemento a 2 → C2
- Valores positivos: igual que los valores positivos en código Signo-Magnitud
- Valores negativos: Hay que restar el código del valor en positivo del minuendo 0000000 (base)
 - Ejemplo +123 → 0b01111011 y -123 → 00000000-01111011 = 0b100000101
 - El código del valor negativo se puede calcular invirtiendo los bits del código del valor positivo y después sumarle 1
 - Equivale a calcular el C1 y sumarle 1
 - El código del valor negativo se puede calcular a partir del código del valor positivo
 - empezando por la dcha repetir los bits hasta el primer uno e invertir el resto de bits

Representación en complemento a la base 2 : C2

- El C2 de un número positivo es el código C2 de su valor en negativo
- El C2 de un número negativo es el código C2 de su valor en positivo
 - Dibujar la tabla y la rueda con todos los valores con sus representaciones.
 - ¿Cuántas representaciones son posibles? ¿Es simétrico el rango de valores representado? ¿Cuántas representaciones tiene el cero?
 - Extender el número de bits del número sin cambiar su valor → Extensión del bit de SIGNO

Extensión del signo en C2

Table 4. Razonamiento de la extensión de signo de un número negativo: números de 3 bits

Valor	C2 sin extensión	C2 con extensión
+33	0100001	00100001
-33	0000000 -0100001 ----- 1011111	00000000 -00100001 ----- 11011111

Se observa que en el C2 con extensión, al hacer la resta y extender con un 0 más el minuendo y el substraendo, provoca la extensión con un bit más en la resta de valor 1 en el dígito más significante. Segundo

añado ceros al minuendo y sustraendo, aparecen unos en la resta sin alterar su valor.

Operaciones aritméticas en C2

- Suma
 - Se realiza como se ha visto para números naturales.
 - Si hay llevada en el MSBit, no se tiene en cuenta, se elimina.
 - $A=0b11011011$. Suma $A+A$

```
Llevadas -> 1 1   1 1   1 1  
  
          1 1 0 1 1 0 1 1 (Valor -37)  
          + 1 1 0 1 1 0 1 1 (Valor -37)  
  
Valor suma      2 1 3 2 1 3 2  
*****  
Resultado --> 1 0 1 1 0 1 1 0<--(Valor -74)
```

- Resta
 - La resta de números con signo se puede realizar de dos formas: $A-B$ ó $A-B = A+(-B)$
 - $A = 0b00110110$ y $B = 0b10011011$
 - Si hay llevada en el MSBit, no se tiene en cuenta, se elimina.

```
Crédito 2 2   2 2   2 2  
  
          1 0 1 1 0 1 1 0<--(Valor -74)  
          - 1 1 0 1 1 0 1 1<--(Valor -37)  
  
LLevada 1 1 1   1 1   1 1  
*****  
Resta     1 1 0 1 1 0 1 1 (Valor -101)
```

Operaciones aritméticas C2: Overflow o Desbordamiento

- $A = 0b00110110$ y $B = 0b10011011 \rightarrow$ Calcular $A-B$
- Con 8 bits el máximo valor es 01111111 de valor $2^7-1=128-1=127$
- La resta $A-(B)=A+(-B)=54+103=157>127 \rightarrow$ **Overflow o Desbordamiento**

```
Crédito 2      2 2   2 2  
  
          0 0 1 1 0 1 1 0<--(Valor = 54)  
          - 1 0 0 1 1 0 1 1<--(Valor = -103)  
  
LLevada 1 1   1 1
```

Resta 1 0 0 1 1 0 1 1 (Valor -101)

- El valor -101 en lugar de la resta correcta +157 es debido a que el resultado esta fuera de rango →
- Observarmos que hemos hecho la SUMA de dos números POSITIVOS y el resultado ha sido NEGATIVO

Operaciones aritméticas C2: Overflow

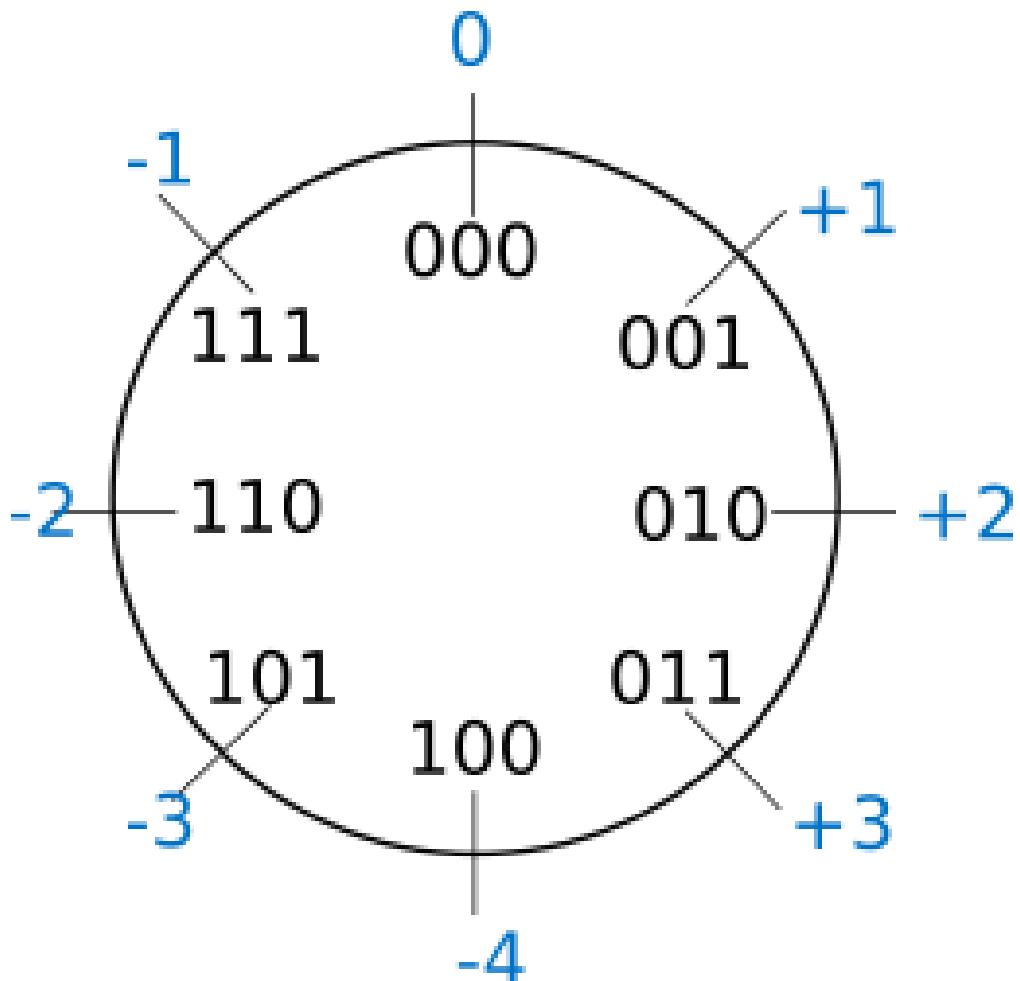


Al realizar la suma de dos valores con el mismo signo si el resultado es de signo contrario hay overflow

Operaciones aritméticas C2: Overflow

- Overflow: la operación requiere operandos con mayor número de bits manteniendo el valor para que el resultado sea correcto.
- Si dos operandos a sumar tienen diferente signo nunca hay overflow
- Si dos operandos a sumar tienen el mismo signo y resultado tiene signo contrario : **Error** de Overflow.
- Ejemplo:
 - Operandos de 1 byte : $01111111+01111111=11111110$ → sumandos positivos y resultado negativo
 - Solución: **Extensión del signo** : Operandos 9 bits → $001111111+001111111=011111110$
 - la repetición del bit más significativo no altera el valor de la representación
 - el bit más significativo es 0 si es positivo y 1 si es negativo. Por lo tanto, 01010 equivale a 01010 ó 001010 ó 0....0001010. Por lo tanto, 1010 equivale a 11010 ó 111010 ó 1....1111010

C2: Representación gráfica del Overflow



Si a partir de la posición 010 nos movemos dos posiciones en sentido horario llegamos a la posición 100.
 Si a 010 le sumamos el valor 2 nos da como resultado 100
 Por lo tanto $010+010=100$, es decir, $2+2=4 \rightarrow \text{overflow}$ ya que el +4 necesita 4 bits y estamos trabajando con 3 bits únicamente.

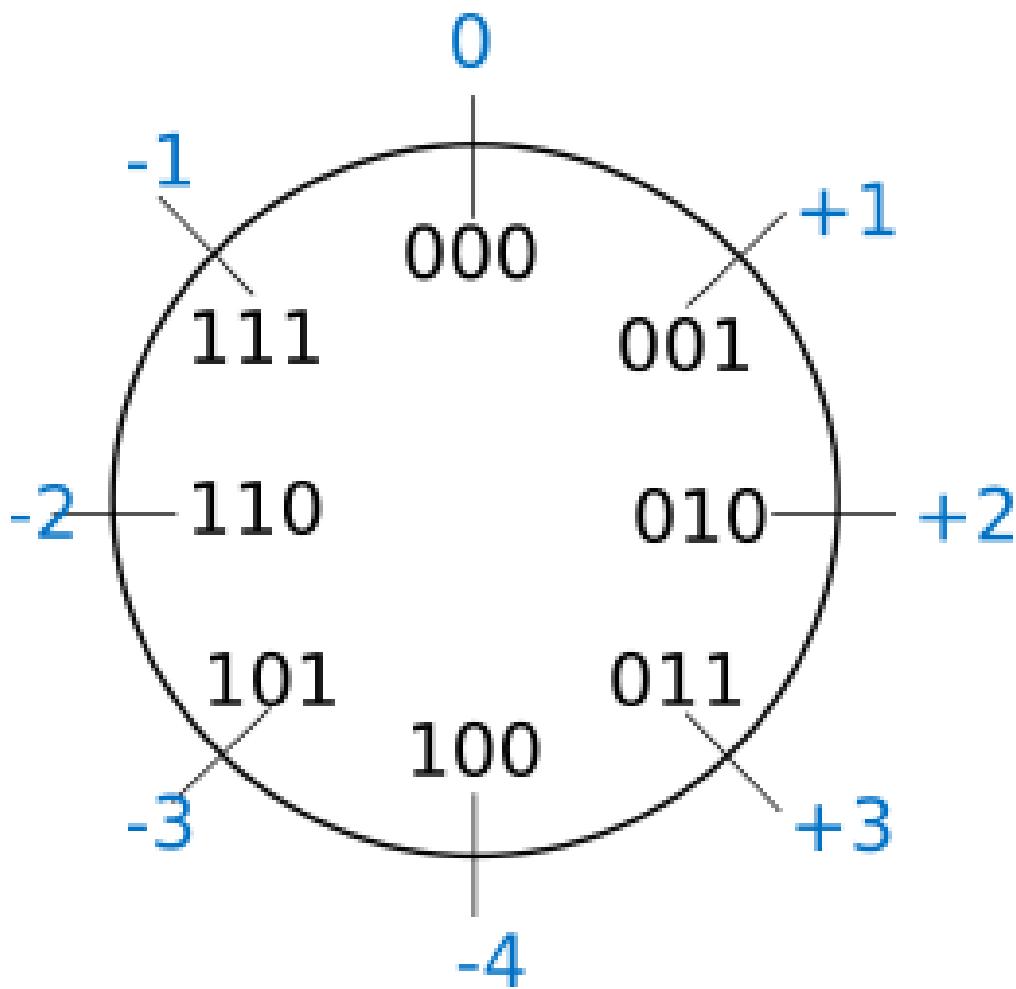
Asimetría del rango en C2: -4 con 3 bits

- Con números de 3 bits los formatos S-M y C1 son simétricos con valores en el rango (+3,-3), en cambio el formato C2 tiene el rango (+3,-4)
- En C2 el valor +4 se representa como 0b0100 y necesita por lo tanto 4 bits, no se puede representar con 3 bits, y el valor -4 se representa con el C2(0100), es decir, 1100 también con 4 bits. El 1100 se puede comprimir ya que tiene el signo extendido con la repetición de 1 de bit más significativo, por lo que la representación 100 es la representación del -4

Complemento a 2 : Ejemplos

- 0b101010101 está en C2 → ¿Cuál es su valor?
 - como es negativo no es un sistema posicional
 - tenemos que calcular el valor negativo a través del valor positivo
 - La representación del valor positivo es el C2 del valor negativo
 - $C2(0b101010101) = 0b010101011$ cuyo valor es $2^7+2^5+2^3+2^1+2^0=128+32+8+2+1=+171$
 - El valor de 0b101010101 es -171
- Si la representación de -123 es 0b100000101 ¿cuál es la de '+123' ?
 - $C2(0b100000101)=0b011111011$ representa el valor '+123'

Aritmética Modular de valores representados en Complemento a 2



Representación de números binarios de 3 bits en C2
Operaciones de suma y resta modular → método gráfico
A partir de la posición 001 si nos movemos en sentido horario (SUMA modular) 2 posiciones obtenemos la posición 011, es decir, $1+2=3$
A partir de la posición 110 si nos movemos en sentido horario (SUMA modular) 9 posiciones obtenemos la posición 111, es decir, $-2+9=-1$
A partir de la posición 110 si nos movemos en sentido antihorario (RESTA modular) 4 posiciones obtenemos la posición 010, es decir, $-2-4=+2$
Los errores de **overflow** se resuelven aumentando el número de bits de la

representación, pero siempre existirá un rango que si lo traspasamos dará overflow.

Comparación S-M, C1 y C2

Table 5. Números de 3 bits

Valor	S-M	C1	C2
+3	011	011	011
+2	010	010	010
+1	001	001	001
0	000	000	000
	100	011	---
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	-	-	100

Número en complemento a 2 y base hexadecimal



Un número binario se puede representar en hexadecimal y hacer la interpretación en complemento a 2. Hay que tener cuidado con las extensiones del signo

- Calcular el valor del número 0xAAA si dicho número tiene formato en complemento a 2
 - si lo convertimos a binario el número empieza por 1, luego es negativo
 - para saber su valor calculo su complementario C2 y tendrá la representación del positivo
 - $0x000-0xAAA = 0x556 \rightarrow 5*16^2+5*16^1+5*16^0 = 5*256+5*16+5 = 1280+80+5 = '+213' \rightarrow 0xAAA$ tiene de valor -213

Número en complemento a 2 y base hexadecimal

- Realizar la suma de los números en formato complemento a 2: 0x80+0x80
 - sumar sin extender el signo de los operandos ¿Hay overflow?



Extender el número 0x80. ¿ Por qué hay que tener cuidado ?

- sumar extendiendo un dígito el signo de los operandos 0x80

Extensión del signo en C2: problema de la BASE

Table 6. Extensión del Signo del N° 0x80 en C2 en binario, hexadecimal y octal

NºBits	Binario	Hexadecimal	Octal
8	10000000	1000_0000 → 0x80	110_000_000 → 0o600
9	110000000	1111_1000_0000 → 0xF80	110_000_000 → 0o600
10	1110000000	1111_1000_0000 → 0xF80	111_110_000_000 → 0o7600
11	11110000000	1111_1000_0000 → 0xF80	111_110_000_000 → 0x7600
12	111110000000	1111_1000_0000 → 0xF80	111_110_000_000 → 0x77600
13	1111110000000	1111_1111_1000_0000 → 0xFF80	111_111_110_000_000 → 0x77600

Números Reales Binarios

- Coma Fija
 - 123.125 → b1111011.001
- Coma flotante
 - Notación científica: potencias en la base del sistema (decimal,binario,etc)
 - En decimal → $1.23125*10^2$
 - En binario → $1.111011001*2^6$
 - el factor que no es potencia se denomina mantisa

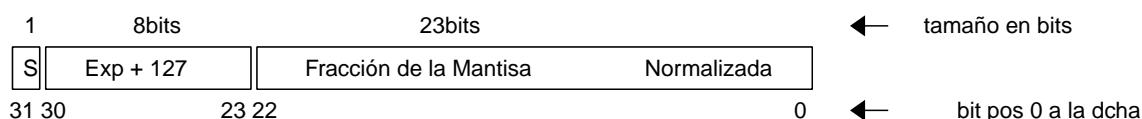
- Se dice que el número real en binario y en notación científica está normalizado si la parte entera de la mantisa vale 1.

Números Reales Binarios: Norma IEEE-754

- Campos del formato en notación científica : Signo, parte entera de la mantisa, parte fracción de la mantisa, base , exponente (módulo y signo)
- ¿Es necesario representar los **seis** campos del formato de la notación científica?
 - Si esta normalizado la parte entera de la mantisa siempre vale 1
 - La base del factor potencia siempre vale 2
 - Por lo tanto la parte entera y la base no son necesario representarlas. Son implícitas a la representación.
 - Hay una forma de no tener que representar el signo del exponente del factor potencia
 - es sumarle una cantidad para que al representarlo en EXCESO siempre sea positivo
- Resumiendo, sólo es necesario representar: el signo del número , la fracción de la mantisa y el exponente en exceso. Por lo tanto el formato IEEE-754 tiene 3 campos.

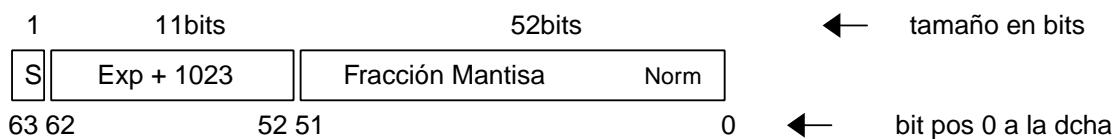
Formato IEEE-754 simple

- representación de 32 bits: 1 bit para el signo / 8 bits para el exponente en exceso a 127 / 23 bits para la fracción
- bit de signo: 0 si es positivo y 1 si es negativo



Formato IEEE-754 doble

- representación de 64 bits: 1 bit para el signo / 11 bits para el exponente en exceso a 1023 / 52 bits para la fracción
- bit de signo: 0 si es positivo y 1 si es negativo



El N° -5.5/1024 en los dos Formatos IEEE-754

- Signo negativo
- Conversión binaria del módulo
 - módulo: $5.5/1024 = 5.5 \cdot 2^{-10} = 101.1 \cdot 2^{-10}$
 - Normalización de la mantisa → $1.011 \cdot 2^{-8}$

- Formato Simple de 32 bits
 - Signo negativo: bit 1
 - Exponente en exceso $127 = -8 + 127 = 119 = 01110111$
 - Fracción de la mantisa=011
 - Solución:
 - 1_01110111_01100000000000000000000000000000
 - 0b101110111011000000000000000000000000000000
 - **0xBBB00000**
 - [calculador ieee](http://weitz.de/ieee/) [<http://weitz.de/ieee/>]
- Formato Doble de 64 bits
 - Signo negativo: bit 1
 - Exponente en exceso $1023 = -8 + 1023 = 1015 = 01111110111$
 - Fracción de la mantisa=011
 - Solución:
 - 1_01111110111_0110...0
 - 0b1011111101110110...0
 - **0xBF76000000000000**

Representación de los Caracteres

Representación de los Caracteres

- Tipos de Caracteres:
 - Alfanuméricos: a,b,...z,0,1,...9,A,B...Z
 - Signos de Puntuación: !"#\$%&/()=
 - de Control: Salto de Línea (\n), Find de Fichero (EOF), Fin de String (\00, ...)
- Formatos
 - ASCII: standard y extendido
 - Unicode: UTF-8

ASCII Standard

2	3	4	5	6	7	30	40	50	60	70	80	90	100	110	120
0:	0	@	P	`	p	0:	(2	<	F	P	Z	d	n	x
1:	!	1	A	Q	a	q	1:)	3	=	G	Q	[e	y
2:	"	2	B	R	b	r	2:	*	4	>	H	R	\	f	p
3:	#	3	C	S	c	s	3:	!	+	5	?	I	S]	g
4:	\$	4	D	T	d	t	4:	"	,	6	@	J	T	^	h
5:	%	5	E	U	e	u	5:	#	-	7	A	K	U	_	i
6:	&	6	F	V	f	v	6:	\$.	8	B	L	V	'	j
7:	'	7	G	W	g	w	7:	%	/	9	C	M	W	a	k
8:	(8	H	X	h	x	8:	&	0	:	D	N	X	b	l
															v

9:) 9 I Y i y	9: ' 1 ; E 0 Y c m w
A: * : J Z j z	
B: + ; K [k {	
C: , < L \ l	
D: - = M] m }	
E: . > N ^ n ~	
F: / ? O _ o DEL	

American Standard Code for Information Interchange

Alfabeto anglosajón

7 bits → $2^7=128$ caracteres : 0x00 hasta 0x1F son 32 caracteres de control y el resto alfanuméricos

En hexadecimal rango [0x00-0x7F]

En decimal rango [0-127]

Upna : 0x55706E61

año 2023: 0x61—6F2032303233

ASCII Extendido

- Para poder representar caracteres de otras culturas Europeas es necesario expandir el standard con 1 bit más
- ASCII 8 bits → $2^8 = 256$ caracteres
- [Python Interpreter Shell](https://www.programiz.com/python-programming/online-compiler/) [<https://www.programiz.com/python-programming/online-compiler/>]

```
ord('A')
hex(ord('A'))
hex(ord('\n'))
chr(65)
chr(0x41)
[hex(ord(c)) for c in "Hola"]
[chr(c) for c in [0x48, 0x6f, 0x6c, 0x61, 0x20, 0x4d, 0x75, 0x6e, 0x64, 0x6f]]
[hex(ord(c)) for c in "ñ"]
[hex(ord(c)) for c in "\n \t"]
```

- La ñ tiene el código ASCII 0xF1

UTF-8

- [Character Set, HTML Converter, etc ...](https://www.charset.org/utf-8) [<https://www.charset.org/utf-8>]
- Unicode Transformation Format (UTF)
- UTF-8: Esta orientado a la transmisión de palabras de 1 byte
- Los caracteres pueden tener entre 1 y 4 bytes → 2^{21} code points ≈ 2 millones;
- The dominant encoding on the World Wide Web and on most Unix-like operating systems
- En linux comando **localectl status** : informa sobre el sistema del teclado
- ñ:
 - hex code 0xC3B1
 - unicode point U+00F1 → los primeros 256 caracteres equivalen al ascii extendido

Unicode Points

- [html css js online](https://html-css-js.com/) [https://html-css-js.com/]: ñ
- U+2228: ✓
- U+22bc: ✗
- U+22bd: ✘
- U+22a6: ⊢
- U+1f60b: 🎉
- U+00f1: ñ
- OrduU+00F1a: Orduña
- U+2190: ←
- U+2192: →

Tema 2: Ejercicios

- Miaulario/Recursos/Ejercicios
- Fundamentos de sistemas digitales Thomas Floyd

Tema 3 : Algebra de Conmutación ó Boole. Funciones Lógicas.

Matemática Lógica Binaria

- Valores Lógicos Binarios : "0" , "1"
 - representa dos estados: los estados de una señal binaria (High/Low), los estados de una bombilla (encendido/apagado), de un conmutador (on/off), de una condición (verdadero/falso), etc, cualquier situación que se pueda modelar mediante dos estados.
- Variables lógicas: ...u, x1, x2, y, v1, u2, ...
 - Una variable independiente que puede tomar los valores "0" y "1"
- Función lógica: z1, z2, z3, F,
 - Una función lógica expresa una relación lógica o/y aritmética o/y comparativa o/y etc entre las variables independientes a través de unos operadores matemáticos.
- Operadores
 - Operadores aritméticos: suma, resta, multiplicación, ...
 - Operadores lógicos: or (suma), and (producto), negación, or exclusiva, etc...
 - Operadores comparadores: > , >, ==, etc

Tablas de la Verdad de los operadores NOT, OR, AND, XOR

Table 7. NOT

x	z=̄x
---	------

0	1
1	0

Table 8.
OR

x	y	$z = x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Table 9.
AND

x	y	$z = x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Table 10.
XOR

x	y	$z = x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Tablas de la Verdad de los operadores NOR, NAND, XNOR

Table 11.
NOR

x	y	$z = x + y$
0	0	1
0	1	0
1	0	0

1	1	0
---	---	---

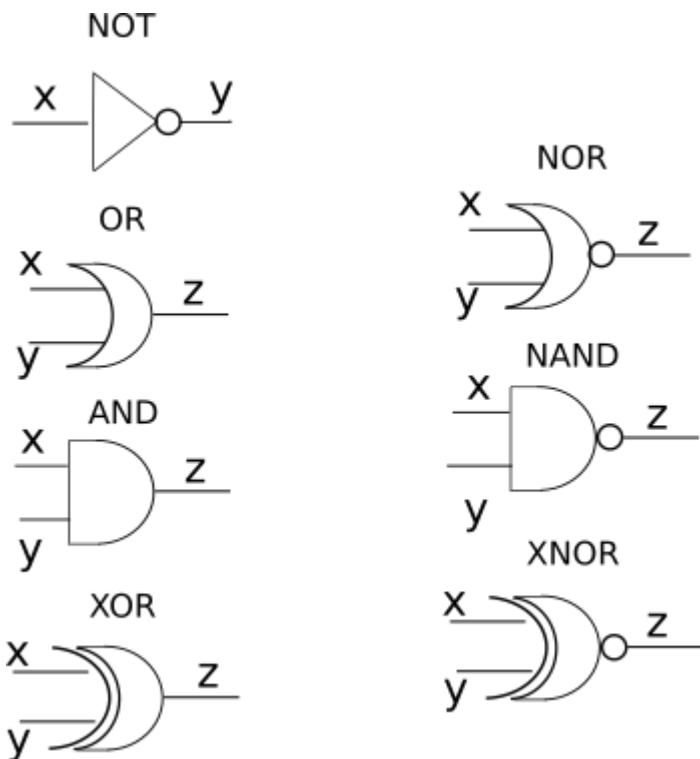
Table 12.
NAND

x	y	$z =$ $x \cdot y$
0	0	1
0	1	0
1	0	0
1	1	0

Table 13.
XNOR

x	y	$z =$ $x \oplus y$
0	0	1
0	1	0
1	0	0
1	1	1

Puertas Lógicas

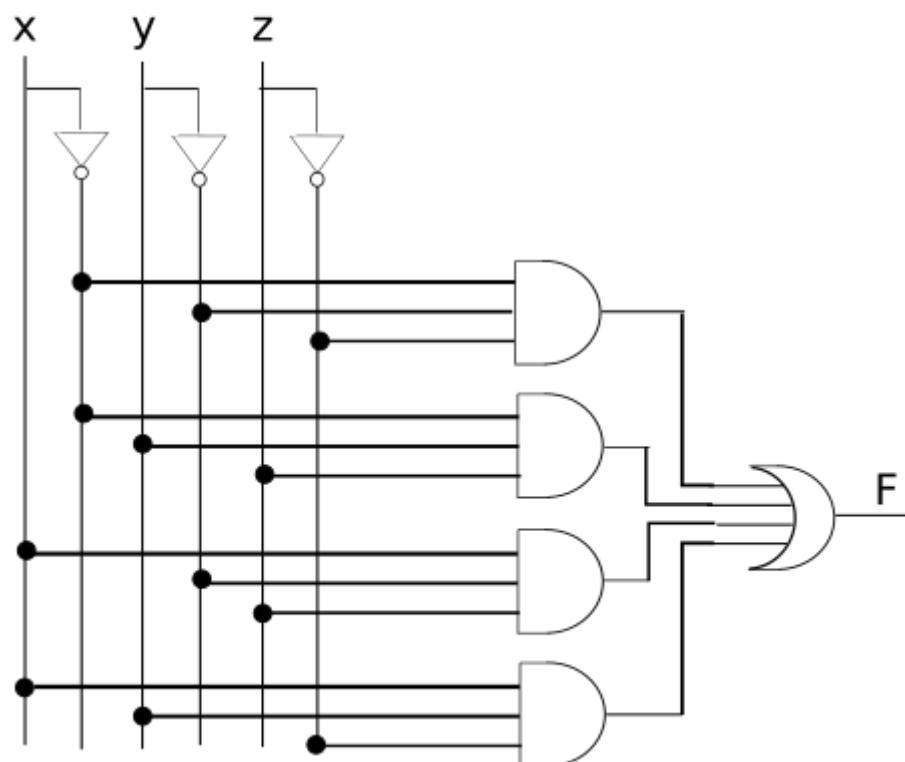
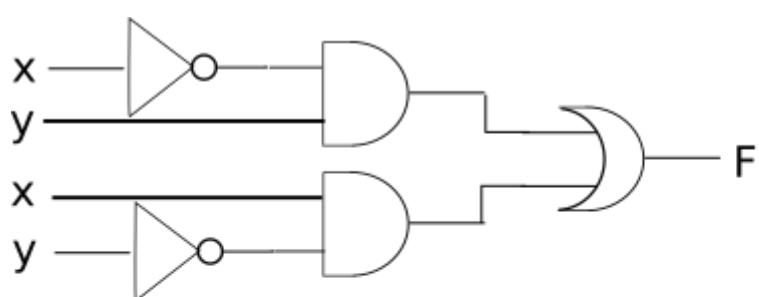


Circuitos Digitales : Expresiones Lógicas

$$F(x,y) = \bar{x}y + x\bar{y}$$

$$F(x,y,z) = \bar{x}\bar{y}z + xy\bar{z} + \bar{x}yz + xy\bar{z}$$

Circuito digital en 3 niveles: not-and-or.



Transparencias PDF: Miaulario/Recursos/Apuntes

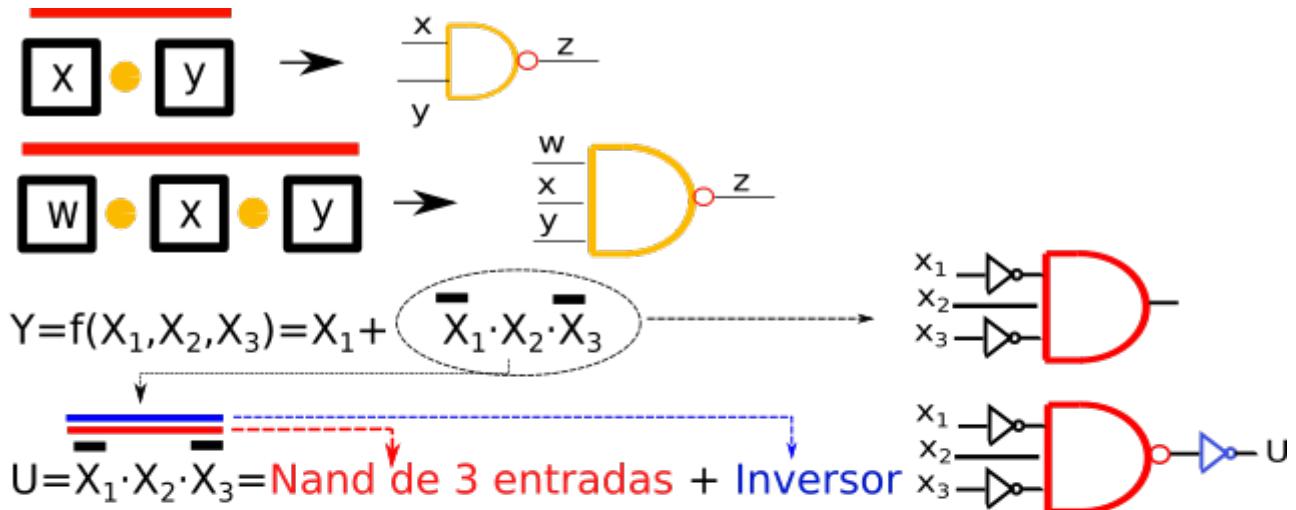
- Postulados del Algebra de Boole
- Teoremas del Algebra de Boole (Leyes de Morgan)
- $(a^*b)+(c^*d); a+a^*b$
- Generación de funciones con puertas lógicas: Ejemplo 1 a)yb)
- Simplificación de funciones mediante Teoremas: Al final
- Formas canónicas: Sum of Products (SOP) y Product of Sums (POS)
 - minitérminos y maxitérminos
 - Ejemplos básicos
- Diagramas de Karnaugh (DK)
 - Agrupar celdas adyacentes en potencias de 2^n
 - Ejemplos básicos
- Relación SOP-POS
 - ejemplo1: $a+ab$
 - ejemplo2: general 3 variables x,y,z

- Simplificación de funciones mediante Teoremas
 - Extender los términos como minitérminos
 - Dibujar DK y agrupar celdas equivale a sacar factor común

Link: álgebra de conmutación funciones.pdf

- [Algebra de Boole. Funciones Lógicas](#) [./PDF/03_algebra_de_commutacion_funciones_logicas.pdf]

Generación de Funciones mediante puertas Lógicas NAND



$$= X_1 + \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} =$$

$$= \overline{X_1} \cdot \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} =$$

$$= \overline{X_1} \cdot \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} =$$

$$= \overline{X_1} \cdot \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} =$$

Formas Canónicas de una Función: Síntesis por minitérminos y maxitérminos

- Hay dos formas canónicas (standard) de expresar una función
 - suma de productos (SOP) de variables
 - producto de sumas (POS) de variables

Lógica Positiva/Negativa: Relación y/o con */+

- Lógica positiva → ¿Cuando vale 1 una función, una expresión, una variable, etc ?

- Lógica negativa → ¿Cuando vale **0** una función, una expresión, una variable, etc ?

Table 14.

OR

x	y	$z = x + y$
0	0	0
0	1	1
1	0	1
1	1	1

 $Z = 0$ si "X" e "Y" valen 0 → $z = (x+y)$ $Z = 1$ si "X" o "Y" valen 1 → $z = (x+y)$

Table 15.

AND

x	y	$z = x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

 $Z = 0$ si "X" o "Y" valen 0 → $(z=x \cdot y)$ $Z = 1$ si "X" e "Y" valen 1 → $(z=x \cdot y)$

Forma Canónica SOP: Suma de Minitérminos

Table 16. Tabla de la Verdad de la Función $F(x_1, x_2, x_3)$

x1	x2	x3	F	minitérminos
0	0	0	0	$m_0 : \bar{x}_1 \bar{x}_2 \bar{x}_3$
0	0	1	0	$m_1 : \bar{x}_1 \bar{x}_2 x_3$
0	1	0	0	$m_2 : \bar{x}_1 x_2 \bar{x}_3$
0	1	1	1	$m_3 : \bar{x}_1 x_2 x_3$
1	0	0	1	$m_4 : x_1 \bar{x}_2 \bar{x}_3$
1	0	1	1	$m_5 : x_1 \bar{x}_2 x_3$
1	1	0	0	$m_6 : x_1 x_2 \bar{x}_3$
1	1	1	0	$m_7 : x_1 x_2 x_3$

Lenguaje natural → F vale **1** (lógica positiva) si $m_3 \text{ o } m_4 \text{ o } m_5$ vale **1** → **suma**Lenguaje natural → m_3 vale **1** (lógica positiva) si \bar{x}_1 y x_2 y x_3 valen **1** → **multiplicación**Lenguaje lógico → $F = \text{SOP} = m_3 + m_4 + m_5$. $F(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_3$.

Cada minitermino se sintetiza mediante una puerta AND.

La síntesis de la función F tendría un nivel de puertas AND de 3 entradas y un nivel con una puerta OR con tantas entradas como minitérminos hacen 1 a la función.

Forma Canónica POS: Producto de Maxiterminos

Table 17. Tabla de la Verdad de la Función $F(x_1, x_2, x_3)$

x1	x2	x3	F	maxiterminos
0	0	0	0	M0 : $x_1+x_2+x_3$
0	0	1	0	M1 : $x_1+x_2+\bar{x}_3$
0	1	0	0	M2 : $x_1+\bar{x}_2+x_3$
0	1	1	1	M3 : $x_1+\bar{x}_2+\bar{x}_3$
1	0	0	1	M4 : $\bar{x}_1+x_2+x_3$
1	0	1	1	M5 : $\bar{x}_1+x_2+\bar{x}_3$
1	1	0	0	M6 : $\bar{x}_1+\bar{x}_2+x_3$
1	1	1	0	M7 : $\bar{x}_1+\bar{x}_2+\bar{x}_3$

Lenguaje natural → F vale 0 (lógica negativa) si M0 ó M1 ó M2 ó M6 ó M7 vale 0 → multiplicación

Lenguaje natural → M1 vale 0 (lógica negativa) si x_1 y x_2 y \bar{x}_3 valen 0 → suma

Lenguaje lógico → $F = \text{POS} = M0M1M2M6M7$.

$$F(x_1, x_2, x_3) = (x_1+x_2+x_3)(x_1+x_2+\bar{x}_3)(x_1+\bar{x}_2+x_3)(\bar{x}_1+\bar{x}_2+x_3)(\bar{x}_1+\bar{x}_2+\bar{x}_3).$$

Cada maxitermino se sintetiza mediante una puerta OR.

La síntesis función F tendría un nivel de puertas OR de 3 entradas y un nivel con una puerta AND con tantas entradas como maxiterminos hacen 0 a la función.

Relación entre la forma canónica SOP y POS

- Ejemplo $F = F(x_1, x_2, x_3) = m_3 + m_4 + m_5$
- $F = m_0 + m_1 + m_2 + m_6 + m_7$
- $F = m_0 + m_1 + m_2 + m_6 + m_7 =$
- $F = \bar{m}_0 \cdot \bar{m}_1 \cdot \bar{m}_2 \cdot \bar{m}_6 \cdot \bar{m}_7$
- $F = M_0 \cdot M_1 \cdot M_2 \cdot M_6 \cdot M_7 = F$

Simplificación de las funciones mediante los Diagramas de Karnaugh (DK)

- El Diagrama de Karnaugh es una representación gráfica multidimensional (2D, 3D, etc) mediante celdas de los minitérminos y maxiterminos de la tabla de la verdad unidimensional 1D
- Ejemplo $F(x_1, x_2, x_3) = \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3$.
 - los minitérminos y maxiterminos siguen la secuencia unidimensional 000-001-010-011-100-101-110-111
 - Los reorganizamos en una matriz de celdas **adyacentes**, donde dos celdas adyacentes tienen todas las variables comunes **excepto una**
- **Simplificación:**
 - Agrupar celdas adyacentes en grupos de un número de celdas potencia de dos → 2^n : 2, 4, 8, etc ...
 - Cuanto mayor sea el número de celdas agrupadas mayor será el número de variables y términos simplificados.

Diagrama de Karnaugh de la función $F(x_1, x_2, x_3)$

celdas adyacentes: todas las variables comunes menos una.

$x_1 x_2$	00	01	11	10
x_3	0	0	0	1
	1	0	1	0

miniter/maxiter adyacentes: todas las variables comunes menos una

$x_1 x_2$	00	01	11	10
x_3	m_0/M_0	m_2/M_2	m_6/M_6	m_4/M_4
	m_1/M_1	m_3/M_3	m_7/M_7	m_5/M_5

- Son adyacentes las celdas de la misma columna o de la misma fila con todas las variables comunes **menos una**. Por eso la tercera columna ha de ser 11
- Observar que cada celda equivale a un minitérmino y un maxitérmino de la Tabla de la verdad
- Por lo tanto, el diagrama DK representa las formas canónicas SOP y POS.

Simplificación de la Función mediante DK

celdas adyacentes: todas las variables comunes menos una.

$x_1 x_2$	00	01	11	10
x_3	0	0	0	1
	1	0	1	1

miniter/maxiter adyacentes: todas las variables comunes menos una

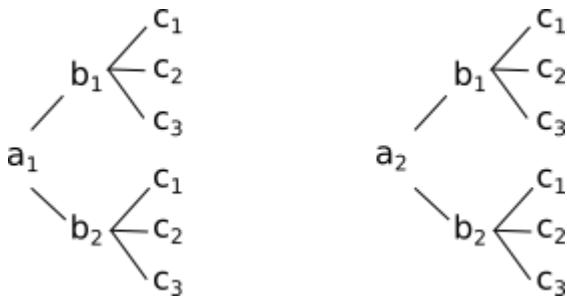
$x_1 x_2$	00	01	11	10
x_3	m_0/M_0	m_2/M_2	m_6/M_6	m_4/M_4
	m_1/M_1	m_3/M_3	m_7/M_7	m_5/M_5

Si sumamos los minitermos de la 4^a columna $Y = f(x_1, x_2, x_3) = m_4 + m_5 = x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 = x_1\bar{x}_2(\bar{x}_3 + x_3) = x_1\bar{x}_2$, se ha simplificado el número de variables de 3 a dos. La función simplificada es $Y = \bar{x}_1x_2x_3 + x_1\bar{x}_2$. Sumar dos minitermos adyacentes equivale a agrupar dos celdas adyacentes y reducir una variable.

Ejercicios básicos matemáticos en el dominio del Algebra de Boole

- $x * 0 = 0; x * 1 = x; x * \bar{x} = 0; x + \bar{x} = 1$
- $x = x * 1 = x * (y + \bar{y}); x = x + 0 = x + y * \bar{y}$
- $x \cdot (x + u + v + ...) = x; x + (x \cdot u \cdot v \cdot ...) = x$
- Transformar una suma de productos de variables lógicas en producto de sumas de variables lógicas
 - $F = y\bar{z} + x\bar{y} + x\bar{y}z$
 - cambio de nomenclatura para facilitar la explicación $F = a_1a_2 + b_1b_2 + c_1c_2c_3$

Ejercicios básicos matemáticos en el dominio del Algebra de Boole



$$F = a_1a_2 + b_1b_2 + c_1c_2c_3 = (a_1 + b_1 + c_1)(a_1 + b_1 + c_2)(a_1 + b_1 + c_3) \cdot \cdot \cdot (a_1 + b_2 + c_1)(a_1 + b_2 + c_2)(a_1 + b_2 + c_3) \cdot \cdot \cdot (a_2 + b_1 + c_1)(a_2 + b_1 + c_2)(a_2 + b_1 + c_3) \cdot \cdot \cdot (a_2 + b_2 + c_1)(a_2 + b_2 + c_2)(a_2 + b_2 + c_3)$$

$$F = y\bar{z} + x\bar{y} + x\bar{y}z = (y + x + x)(y + x + \bar{y})(y + x + z) \cdot \cdot \cdot (y + \bar{y} + x)(y + \bar{y} + \bar{y})(y + \bar{y} + z) \cdot \cdot \cdot (\bar{z} + x + x)(\bar{z} + x + \bar{y})(\bar{z} + x + z) \cdot \cdot \cdot (\bar{z} + \bar{y} + x)(\bar{z} + \bar{y} + \bar{y})(\bar{z} + \bar{y} + z)$$

Ejercicios básicos matemáticos en el dominio del Algebra de Boole

Simplificación

$$F = (y + x)(y + x + z)(1)(1)(\bar{z} + x)(\bar{z} + x + \bar{y})(1)(\bar{z} + \bar{y} + x)(\bar{z} + \bar{y})(1) = (y + x)(y + x + z)(\bar{z} + x)(\bar{z} + x + \bar{y})(\bar{z} + \bar{y} + x)(\bar{z} + \bar{y})$$

POS → Expansión para que tenga cada término las 3 variables

$$F = (y + x + z\bar{z})(y + x + z)(\bar{z} + x + y\bar{y})(\bar{z} + x + \bar{y})(\bar{z} + \bar{y} + x)(\bar{z} + \bar{y} + x\bar{x})$$

Aplico la propiedad Distributiva a cada término

$$F = (y + x + z)(y + x + \bar{z})(y + x + z)(\bar{z} + x + y)(\bar{z} + x + \bar{y})(\bar{z} + x + \bar{y}) \cdot (\bar{z} + \bar{y} + x)(\bar{z} + \bar{y} + x)(\bar{z} + \bar{y} + \bar{x}) = (y + x + z)(y + x + \bar{z})(\bar{z} + x + \bar{y})(\bar{z} + \bar{y} + \bar{x}) = (x + y + z)(x + y + \bar{z})(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + \bar{z}) = M_0M_1M_3M_7$$

Ejercicios básicos matemáticos en el dominio del Algebra de Boole

- F en la 1^a forma canónica

$$\bullet F = y\bar{z} + x\bar{y} + x\bar{y}z = y\bar{z} \cdot (x + \bar{x}) + x\bar{y} \cdot (z + \bar{z}) + x\bar{y}z = y\bar{z}x + y\bar{z} \cdot \bar{x} + x\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z = xy\bar{z} + \bar{x}y\bar{z} + x\bar{y}z + x\bar{y}\bar{z} = m_6 + m_2 + m_5 + m_4$$

Nominación Teoremas

- conmutativa $a \cdot b =$
- idempotencia $a + a =$
- identidad $a \cdot 1 =$
- complementario $a + \bar{a} =$
- absorción $a + ab =$
- distributiva $ab + cd =$
- $a+1$
- a^*0

Simplificación de funciones mediante axiomas y teoremas del Algebra de Boole



celdas adyacentes equivale a minitérminos con factores comunes, que pueden ser agrupados y simplificados.

- Ejemplo 1: $Y=f(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3$
- Dibujar la TV y el DK de la función Y
- Simplificar la función Y mediante el agrupamiento de celdas en el DK
- Partiendo del agrupamiento DK razonar la simplificación de la función Y mediante los **axiomas y teoremas del algebra de Boole**.

Simplificación de funciones mediante el Diagrama de Karnaugh

- Agrupar celdas adyacentes en grupos de un número de celdas 2^n : 2, 4, 8, etc ...
- Cuanto mayor sea el número de celdas agrupadas mayor será el número de variables y términos simplificados.
- $y = f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \dots$
- $y = f(x_1, x_2, x_3, x_4) = \sum (m_0 + m_1 + m_3 + m_4 + m_5 + m_7 + m_9 + m_{11} + m_{13} + m_{14} + m_{15})$
- $y = f(x_1, x_2, x_3, x_4) = \sum (0, 1, 3, 4, 5, 7, 9, 11, 13, 14, 15)$
- Simplificar la función "y" tanto simplificando la forma SOP como simplificando la forma POS y dibujar el resultado de la síntesis.

Ejercicios matemáticos en dos dominios Gráfico/Algebra de Boole

- $F = f(x_1, x_2) = x_1 = x_1 + x_2\bar{x}_2$
 - obtener la forma canónica SOP y POS mediante TV y DK
 - obtener la forma canónica SOP analíticamente: propiedad identidad
 - obtener la forma canónica POS analíticamente: propiedad distributiva

- convertir la forma canónica POS a SOP mediante la equivalencia entre miniterminos y máxiterminos
- convertir la forma canónica POS a SOP analíticamente
- $F = f(x_1, x_2, x_3) = x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_2x_3$
 - obtener analíticamente la forma canónica SOP
 - primero expandir por la propiedad distributiva
 - simplificar cada factor
 - simplificar factores repetidos
 - extender cada factor para que tenga las 3 variables
 - aplicar la propiedad distributiva

Cuaderno de Ejercicios: Capítulo 1

- 1.1, 1.2, 1.4, 1.5, 1.6, 1.8, 1.9
- Metodología: antes de comenzar a resolver el ejercicio hay que describir el método a seguir para resolver el ejercicio.

Planteamiento de los Ejercicios Capítulo 1

- Ejercicio 1.1
 - Resolverlo primero por DK
 - asociar DK con álgebra de Boole
 - SOP,POS,factor común,ordenar,simplificar
- Ejercicio 1.2
 - Análisis, TV(combinaciones repeticiones)
 - variable indiferente → valor X
 - variable nula → TV y DK reducidas
- Ejercicio 1.3
 - lenguaje natural → lenguaje lógico
 - $F=SOP$
 - lógica positiva ($o/y \rightarrow */+$) → $F=1$ si ...
 - lógica negativa ($o/y \rightarrow */+$) → $F=0$ si ...
 - deducir máxiterminos y miniterminos
 - $F=X$ si ...
 - Función: valor no definido: X
 - DK : definición libre para simplificar: 0 ó 1

Planteamiento de los Ejercicios Capítulo 1

- Ejercicio 1.4
 - Resolverlo por DK
 - Formato ajedrez → Factor Común → XOR

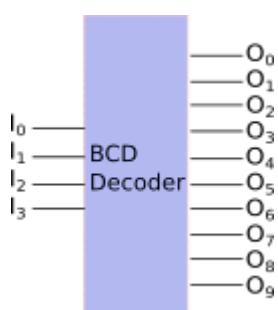
- Ejercicio 1.5
 - lenguaje natural → lenguaje lógico
 - lógica positiva - lógica negativa
 - o/y → */+
 - Condiciones → miniterminos o maxiterminos

Planteamiento de los Ejercicios Capítulo 1

- Ejercicio 1.6
 - lenguaje natural → lenguaje lógico
 - lógica positiva - lógica negativa
 - o/y → */+
 - Condiciones → miniterminos o maxiterminos
- Ejercicio 1.8
 - Escenificación → Diferentes Casos
 - Entro al pasillo por la izda y salgo por la dcha
 - Entro al pasillo por la izda y salgo por la izda
- Ejercicio 1.9
 - Señal binaria: Relación de aspecto
 - Período: Duración nivel alto respecto nivel bajo
 - Módulos o subcircuitos:
 - Anidamiento de funciones → subfunciones

Binary Coded Decimal (BCD)

- El código binario BCD codifica, cada dígito decimal de un número, de forma directa con 4 bits para cada dígito decimal.
- Ejemplos
 - 23 → 0010 0011
 - 87045 → 1000 0111 0000 0100 0101
- Diseñar un circuito digital simplificado que decodifique el código binario BCD en uno de los diez dígitos: 0,1,...,9



- códigos y dígitos

Código	Dígito Decimal
0000	0
0001	1
0010	2
0011	3
0100	4

- códigos y dígitos

Código	Dígito Decimal
0101	5
0110	6
0111	7
1000	8
1001	9

- códigos y dígitos

Código	Dígito Decimal
1010	X
1011	X
1100	X
1101	X
1101	X
1111	X

Tema 4: Lenguaje de Descripción Hardware VHDL

Very high speed integrated circuits Hardware Description Language (VHDL)

- HDL: Hardware Description Languages
- NO son lenguajes de programación sino de **descripción de Hardware**. Es una lenguaje que está pensado para describir circuitos de la misma forma que otras formas de describir un circuito digital: mediante un esquema eléctrico, mediante una tabla de la verdad, mediante diagramas de secuencias de estados, etc ...
- También sirve para describir las formas de onda cuadradas de las señales binarias de entrada de un circuito digital
- ... y por supuesto también tiene sentencias y estructuras de programación que no describen circuitos digitales, por ejemplo imprimir en la pantalla una frase como "Hello World".

Descripción del Hardware de un circuito digital.

```
-- Sistemas Digitales : Descripción VHDL Curso Primavera 2023
-- Circuito light.vhd: Puerta lógica XOR extendida
entity of light_bit is
  port (
    x,y : in bit;
    z   : out bit
  );
end entity;

architecture rtl of light_bit is
  signal s,t,u,v : bit;
begin
  s <= not x;
  t <= not y;
  u <= x and t;
  v <= y and s;
  z <= u or v;
end rtl;
```

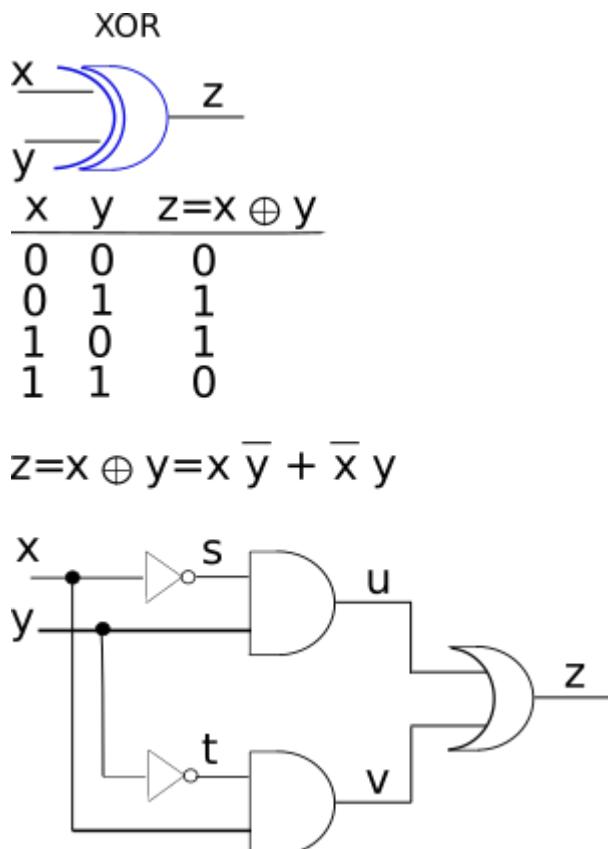
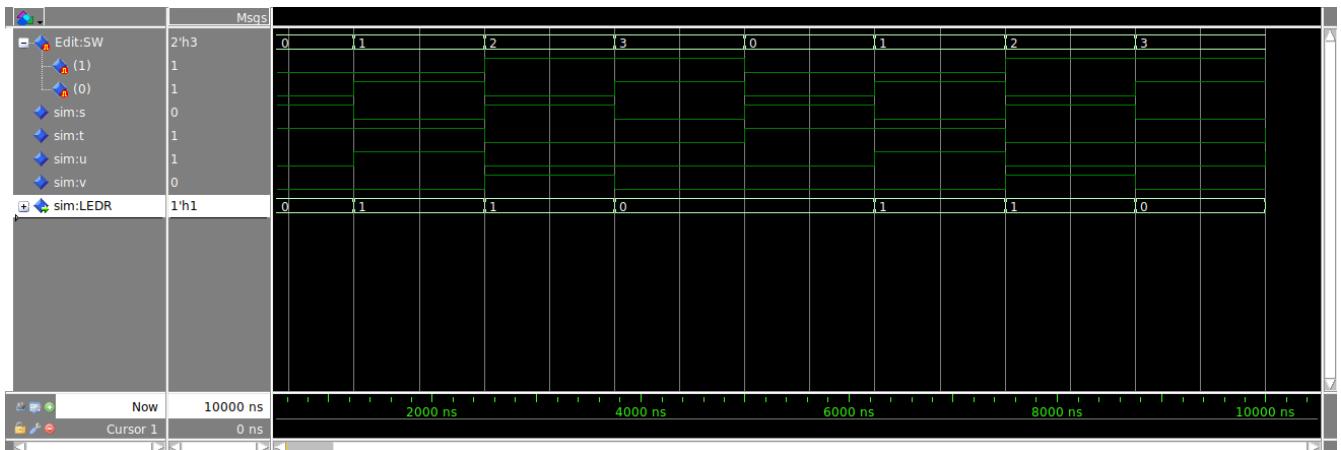


Figure 1. Circuito light_bit.vhd

Cronogramas



Señales VHDL :tipos

- Hay dos **tipos de señales** en el lenguaje vhdl:
 - PORT : x,y,z
 - son señales de acceso al circuito: su **modo** puede ser de entrada (IN) ó de salida (OUT)
 - una señal de entrada tipo IN no puede conectarse a la salida de una puerta lógica
 - una señal de salida tipo OUT no puede conectarse a la entrada de una puerta lógica
 - SIGNAL : s,t,u,v
 - son señales internas al circuito y son bidireccionales: pueden conectarse tanto a la entrada como a la salida de una puerta lógica

Señales VHDL : tipos de datos

- hay diferentes **tipos de datos** para las señales
- tipo de dato bit único : BIT : admite únicamente dos valores: el ' 0 ' y el ' 1 '
- en VHDL los valores de los bits hay que entrecollarlos para diferenciarlos de los datos de tipo INTEGER
- tipo de dato secuencia de bits: " 010001010101 " → doble entrecillado si el dato se representa con más de un bit.

Señales VHDL : Buses

- Físicamente un Bus es un conjunto de pistas metálicas que sirven para transportar señales conectando dos unidades
- Por ejemplo el "bus de direcciones" de 32 hilos ó pistas de la memoria RAM sirve para seleccionar una dirección de 32 bits de la memoria. La dirección **0110011001100110011001100110** se transporta desde la CPU hasta la memoria RAM a través de un bus de 32 pistas. Al bus de direcciones de memoria (address bus) se le podría llamar **A** y a cada hilo del bus $A_{31}, A_{30}, \dots, A_1, A_0$.
- Desde el punto de vista lógico un bus es un vector o un array de dimensión "n", por ejemplo n=32.
- El tipo de datos de los buses **A** y **B** de 32 bits se podrían declarar como:
 - signal A,B :bit_vector(31 downto 0); donde el bit MSB(más a la izquierda) sería el hilo A_{31} y el bit LSB(más a la derecha) el bit A_0 y lo mismo con el bus B
 - signal A,B :bit_vector(0 to 31); donde el bit MSB(más a la izquierda) sería el hilo A_{0} y el bit LSB(más a la derecha) el bit A_{31} y lo mismo con el bus B

Sentencias VHDL : Asignación Concurrente

- CAS : Concurrent Assignment Sentence
- La sentencia CAS se representa mediante el símbolo \Leftarrow
- El valor resultante de **evaluar** la expresión a la derecha del símbolo \Leftarrow se asigna a la señal a la izquierda del símbolo \Leftarrow

```
s <= not x;
t <= not y;
u <= x and t;
v <= y and s;
z <= u or v;
```

Sentencias Concurrentes

Concepto de concurrencia: ¿ CUANDO se ejecuta una sentencia concurrente? cuando hay un **evento** en una de las **señales sensibles** de la sentencia. En el caso de la sentencia CAS la señales sensibles son las señales a la derecha del símbolo \Leftarrow .

Ejemplo:

```
s <= not x;
t <= not y;
u <= x and t;
v <= y and s;
z <= u or v;
```

Ejemplo: Ver cronograma

- 0- "x" = "y" = **0** \Rightarrow s = t = **1** \Rightarrow u = v = **0** \Rightarrow z = **0**
- 1- Se produce un EVENTO (**0→1**) en la señal puerto "x"
- 2- "x" es una señal sensible en la línea 1 del código
y en la línea 3 del código
- 3- Se ejecutan las líneas 1 y 3 del código
- 4- Ejecución de la línea 1: "s" (**1→0**)
- 5- Ejecución de la línea 3: "u" (**0→1**)
- 6- Hay un evento en "s": se ejecuta la línea 4 : "v" no cambia \rightarrow no evento
- 7- Hay un evento en "u": se ejecuta la línea 5 : "z" cambia (**0→1**)
- 8- La señal z no es una señal sensible en ninguna de las sentencias \Leftarrow : FIN
- 9- FIN de la actualización de todas las señales hasta el próximo evento en "x" o/y "y"

Sentencias Concurrentes

Las sentencias concurrentes NO se ejecutan secuencialmente, sino **simultáneamente**, de la misma forma que en el circuito "**light_bit.vhd**" la puerta lógica OR procesa sus dos entradas al mismo tiempo que las puertas NOT y AND del mismo circuito.

En los 4 ejemplos siguientes la actualización de los valores de todas las señales, ante el evento de una de ellas, da el MISMO resultado, ya que la ejecución no es secuencial, sino que se ejecutan UNICAMENTE las sentencias concurrentes cuyas señales sensibles varían; y las sentencias que se ejecutan lo hacen SIMULTANEAMENTE.

```
s <= not x;
```

```
t <= not y;
u <= x and t;
v <= y and s;
z <= u or v;
```

```
z <= u or v;
v <= y and s;
u <= x and t;
t <= not y;
s <= not x;
```

```
u <= x and t;
v <= y and s;
z <= u or v;
s <= not x;
t <= not y;
```

```
u <= x and t;
t <= not y;
s <= not x;
z <= u or v;
v <= y and s;
```

Entidad

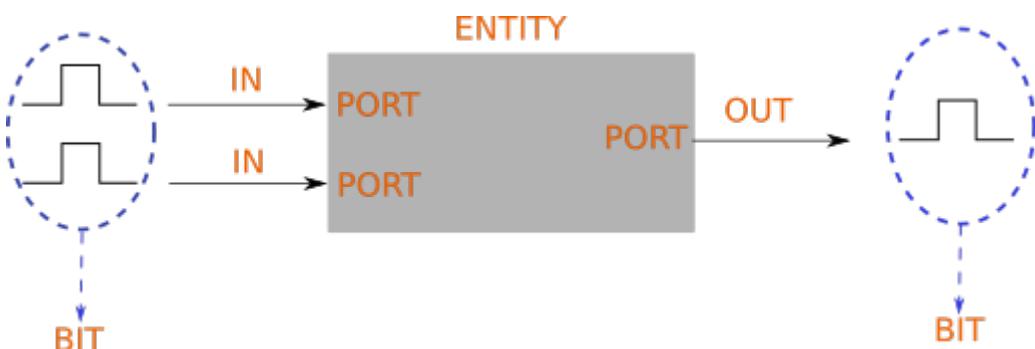
La **entidad** describe el comportamiento del circuito digital visto desde fuera, es decir, describe únicamente los accesos de entrada y salida del circuito. Los accesos de entrada y salida se realizan a través de señales digitales binarias denominadas **puertos**.

La entidad se define con el keyword ENTITY

La entidad que hay nominarla con un nombre. Este nombre condiciona el nombre del fichero donde se almacena, que ha de tener el mismo nombre con y la extensión **.vhdl**

Las señales tipo PORT pueden ser de entrada (IN) ó salida (OUT) ó salida_y_entrada (BUFFER).

Además del **modo** de la señal (IN-OUT-BUFFER) es necesario declarar el tipo de los datos (BIT)

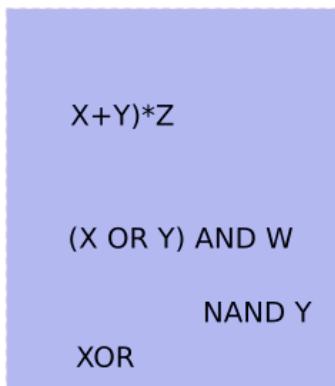


Las señales cuyos datos son de tipo BIT admiten los valores '0' y '1'

Sintaxis

```
entity of light_bit is
  port (
    x,y : in bit;
    z   : out bit
  );
end entity;
```

Arquitectura

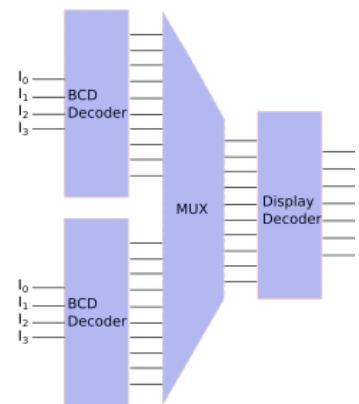
REGISTER TRANSFER LEVEL
RTL

BEHAVIORAL

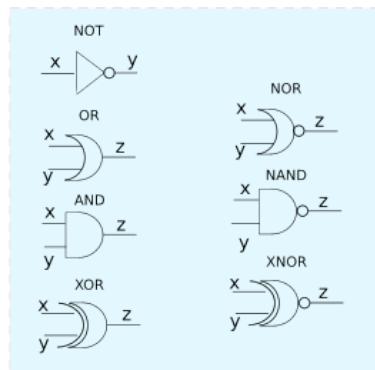
Si x e y valen 0 entonces la salida vale 0... ó si las entradas valen 1 la salida vale 0.....

x	y	$z = x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

STRUCTURAL



SINTESIS



3 Tipos de arquitecturas:

RTL: Expresiones lógicas

Behavioral o comportamental: funcionalidad

Structural: conectar subcircuitos

La descripción más sencilla es la **behavioral** ... pero también es la que exige un mayor esfuerzo al sintetizador.

Arquitectura

```

architecture rtl of light_bit is
  signal s,t,u,v : bit;
begin
  s <= not x;
  t <= not y;
  u <= x and t;
  v <= y and s;
  z <= u or v;
end rtl;

```

La arquitectura del circuito se declara con el keyword architecture

La arquitectura del circuito hay que nominarla con cualquier nombre: rtl, fun, etc... y relacionarla con una entidad

Las señales internas hay que declararlas con el keyword signal y definir el tipo de datos: pej bit

La relación entre las señales (puertos e internas) se define mediante "sentencias vhdl" entre los keywords begin y end

Hojas de Referencia

[Hoja de referencia simple](#) [./PDF/VHDL_Cheat_Sheet.pdf]

[Hoja de referencia completa](#) [./PDF/VHDL_QRC__01.pdf]

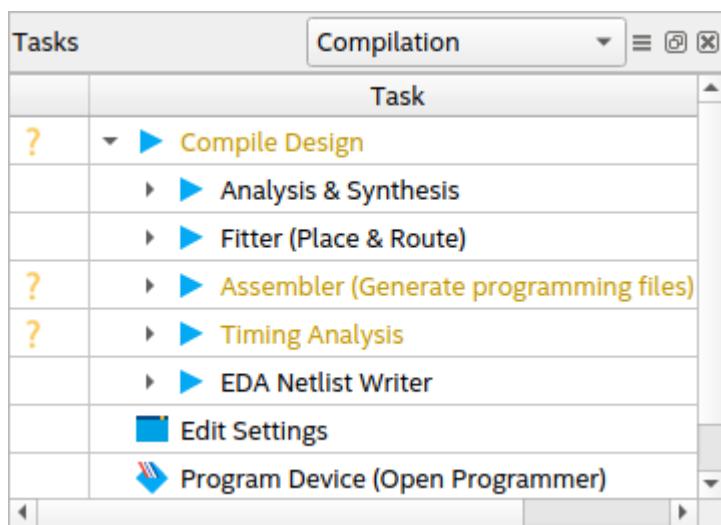
Síntesis: Herramienta Quartus

```

File Edit View Project Assignments Processing Tools Window Help
light_bit
Project Navigator Hierarchy Entity:Instance
Cyclone V: 5CSEMA5F31C6
abc light_bit
16 library std; -- La librería std no es necesario declararla
17 use std.standard.all; -- Si no se declara la librería std, tampoco el paquete standard
18
19
20 ENTITY light_bit IS
21   PORT(SW : IN bit_vector (1 downto 0);
22        LEDR : OUT bit_vector (0 downto 0));
23 END light_bit ;
24
25 ARCHITECTURE rtl OF light_bit IS
26   signal s,t,u,v : bit;
27 BEGIN
28   s <= NOT SW(0);
29   t <= NOT SW(1);
30   u <= SW(0) AND t;
31   v <= SW(1) AND s;
32   LEDR(0) <= u OR v;
33
34 END rtl ;
35

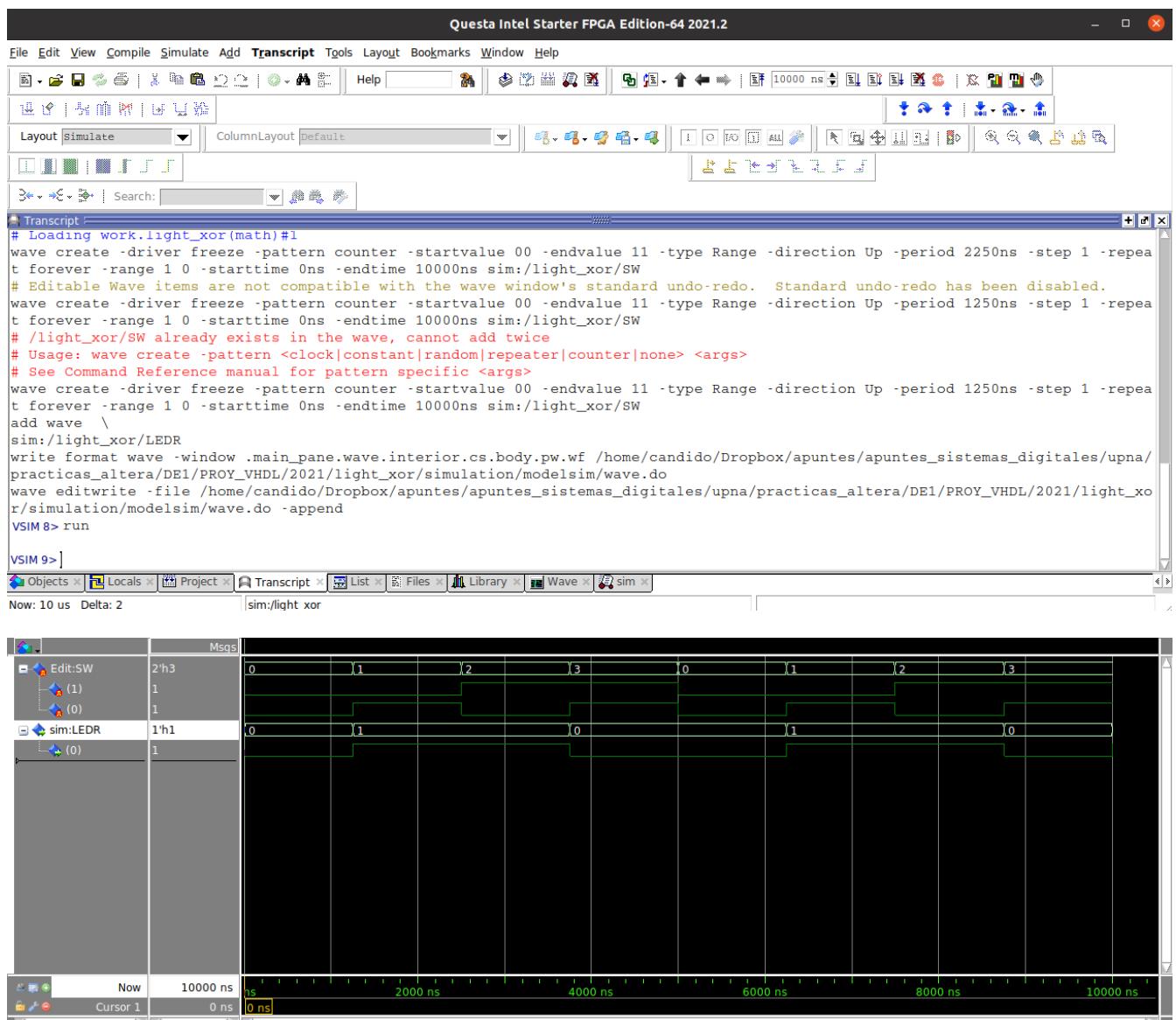
```

Entrada: código VHDL "light_bit.vhd"



Salida: simulación y síntesis : bitstream "quartus_light.sof"

Simulación: Herramienta Modelsim/Questa



Verificar el correcto funcionamiento del circuito antes de su Fabricación

Fabricación: FPGA Cyclone V de Intel

- Tarjeta de prototipado de Terasic DE1 SoC
 - La tarjeta contiene la **FPGA Cyclone V** y sus periféricos
 - El diseño "light_bit" se implementa en el chip FPGA (**Field Programmable Gate Array**)

Tema 5: Circuitos Aritméticos

Tema 6: Otros Circuitos Combinacionales

Fin del Primer Parcial

listas

matematicas

$Y = f(x_1, x_2, x_3) = x_1 + \bar{x}_1 x_2 x_3$ = transformar suma en producto con morgan
 $= x_1 + \bar{x}_1 x_2 x_3 =$ aplico morgan = $\bar{x}_1' \cdot \bar{x}_1 x_2 x_3$

x

$$\frac{\overline{x_1x_2x_3}}{\overline{x_1x_2x_3}}$$

Text Formatting

text with id and role

some text
some text

```
printf("Hello world %d\n", x)
```

step 1



Finally, a pro tip...



First, watch out for...



Second, don't forget...

step 2

The greatest glory in living lies not in never falling, but in rising every time we fall.
The greatest glory in living lies not in never falling, but in rising every time we fall.

step 3

blue

white

red

overline

$Z = X$

$Z = X$

$Z = X$

tablas