Trabajo Práctico 0

A- Sistemas de numeración

A.01 *Escriba los dígitos que componen los siguientes sistemas de numeración (se brindan como ejemplo los vistos en la teoría)*

Base	Dígitos
2	0 , 1
3	
4	0 , 1 , 2 , 3
7	
8	0 , 1 , 2 , 3 , 4 , 5 , 6 , 7
10	0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9
12	
16	0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , E , F

A.02 *Escriba la definición de dígitos y de base de un sistema de numeración.*

A.03 *Convierta los siguientes números a su valor en decimal (como el ejemplo planteado).*

- $4301_5 = 4 * 5^3 + 3 * 5^2 + 0 * 5^1 + 1 * 5^0 = 4 * 125 + 3 * 25 + 0 + 1 = 576_{10}$

4301_6	4301_7	4301_{16}
$3FFF_{16}$	10100110_2	6502_8
8000_{16}	111_2	10_{16}
$7A3C_{16}$	10_2	F_{16}
00011000_2	10_4	20_{16}

A.04 *Complete los casos faltantes de conversión entre bases.*

Base 2	Base 16	Base 10
01100000		
11111111111	7FF	
		65535
		65536
		255
		256
	7F	
10000000		
110000		

A.05 *Escriba los siguientes números como potencias de 2*

$65536 \rightarrow 2^{16}$, $4096 \rightarrow 2$, $1024 \rightarrow 2$, $512 \rightarrow 2$, $256 \rightarrow 2$, $128 \rightarrow 2$, $16 \rightarrow 2$

32768 → 2 , 16384 → 2 , 1048576 → 2 , 4294967296 → 2

A.06 *Escriba los siguientes números como potencias de 2 simplificadas*

2^{16} → 64K , 2^{20} → , 2^{10} → , 2^{32} → , 2^{40} →

A.07 *Convierta los siguientes números decimales a binario (como el ejemplo dado) indicando cuántos bits son necesarios para la representación en binario.*

Número	Base	Resultado	Resto
19	2	9	1
9	2	4	1
4	2	2	0
2	2	1	0
1	2	0	1

10011
5 bits

16+2+1 = 19

128 , 127 , 65540 , 65536 , 65535 , 20 , 63 , 64 , 70, 1000 , 1023 , 1024 , 1030

A.08 *Indique si los siguientes números son par o impar (sin dividir por 2).*

10100110_2 **PAR**	10100111_2 **IMPAR**	4301_{16}
$3FFF_{16}$	10110_2	6502_8
8000_{16}	10111_2	10_{16}
$7A3C_{16}$	$7BB7_{16}$	F_{16}
00011000_2	$DE3B_{16}$	20_{16}

A.09 *Indique la cantidad de elementos que pueden representarse y cual es el menor y el mayor de ellos para sistemas sin signo que utilizan*

Sistema	Cantidad de elementos	Menor	Mayor
2 dígitos decimales	100	0	99
4 dígitos decimales			
2 bits	4	0	3 (11_2)
4 bits			
7 bits			
8 bits			
16 bits			
24 bits			
32 bits	4 G Elementos		
2 dígitos base 16	256 elementos	0	255 (FF_{16})
4 dígitos base 16			
6 dígitos base 16			

8 dígitos base 16			

A.10 *Utilice la fórmula generalizada para calcular la cantidad de elementos y el mayor siendo N la cantidad de dígitos y B la base (todas las combinaciones)*

N= 1 , 2 , 5 , 7 , 9 , 12 , 20

B= 2 , 5 , 7 , 10 , 12 , 16

A.11 *Represente los siguientes números en 8 bits y en 16 bits (sin signo)*

25_{10} , 12_{10} , 8_{10} , 65_{10} , 96_{10} , 255_{10} , 250_{10} , 128_{10} , 127_{10} , 129_{10}

A.12 *Complete la tabla con todos los valores posibles en 4 bits sin signo y su valor decimal*

4 bits	Decimal
0000	0
1111	15

A.13 *Dados A y B (c/u 4 bits sin signo) realice la operación e indique el resultado*

A	B	Operación	Carry / Borrow	Resultado	A=B	A>B	A<B
1001	1010	A - B	**Borrow**	1111	No	No	**Si**
		A + B	Carry	0011			
0100	0011	A - B	**No**	0001	No	**Si**	No
		A + B	No	0111			

1000	1000	A - B					
		A + B					
0111	1000	A - B					
		A + B					
1101	0110	A - B					
		A + B					
0000	1111	A - B					
		A + B					
1011	1101	A - B					
		A + B					
0111	0111	A - B					
		A + B					

A.14 *Dados A y B (c/u 8 bits sin signo) realice la operación e indique el resultado*

A	B	Operación	Carry / Borrow	Resultado	A=B	A>B	A<B
00101000	01000000	A - B	**Borrow**	11101000	No	No	**Si**
		A + B	No	01101000			
0100000	00101000	A - B	**No**	00011000	No	**Si**	No
		A + B	No	01101000			
10001010	10001110	A - B					
		A + B					
01111010	01111010	A - B					
		A + B					
11011111	01101111	A - B					
		A + B					
00000000	11111111	A - B					
		A + B					
10111100	11010001	A - B					
		A + B					
01111010	01110101	A - B					
		A + B					

A.15 *Indique para cada número su base, base menos uno y calcule ambos complementos.*

Número	Base	Base - 1	C.B. -1	C. B.
15234_{10}	100000_{10}	99999_{10}	84765_{10}	84766_{10}
7245_{8}				
100110_{2}				
$FD300A14_{16}$				
$BA2321A0_{12}$				
10000000_{2}				

A.16 *Calcule los complementos acotados a 8 dígitos*

Número	Base	Base - 1	C.B. -1	C. B.
89234120_{10}	100000000_{10}	99999999_{10}	10765879_{10}	10765880_{10}
$ABEF7240_{16}$				
90000000_{10}				
10000001_{2}				
10000000_{2}				
01111111_{2}				
01000000_{2}				
11111111_{2}				

A.17 *Complete la tabla con los complementos para los números de 4 bits e indique su valor en decimal si se utilizan números signados en complemento a la base*

4 bits	C.B.-1	C.B.	Decimal
0000	1111	0000	**0**
0001	1110	1111	**1**
0111	1000	1001	**7**
1000	0111	1000	**-8**

1110	0001	0010	**-2**
1111	0000	0001	**-1**

A.18 *Indique la cantidad de elementos que pueden representarse y cual es el menor y el mayor de ellos para sistemas signados en C.B. que utilizan*

Sistema	Cantidad de elementos	Menor	Mayor
2 bits	4	-2 (10_2)	1 (01_2)
4 bits			
7 bits			
8 bits			
16 bits			
24 bits			
32 bits	4 G Elementos		

A.19 *Convierta los siguientes números representados en 8 bits signados en C.B. a 16 bits signados en C.B. Ej: 11000101 → 11111111 11000101 ; 01111111 → 00000000 01111111*
10101010 ; 11100010 ; 01110001 ; 00000000 ; 1000000 ; 00000010 ; 11111111

A.20 *Dados A y B (4 bits signados en C.B.) realice la operación e indique el resultado*

A	B	OP	C/B	Res	V	N	A=B	A>B	A<B
1100 (-4)	**0**101 (5)	A-B	No	**0**111 (7)	**Si**	**No**	No	No	**Si**
1100 (-4)	0101 (5)	A+B	C	0001 (1)	No	No	No	No	Si
0011 (3)	**1**001 (-7)	A-B	B	**1**010 (-6)	**Si**	Si	No	**Si**	No
0011 (3)	1001 (-7)	A+B	No	1100 (-4)	No	Si	No	Si	No
0001	0001	A-B							
0111	1000	A-B							
1000	1000	A-B							
1000	0001	A+B							
1000	0000	A-B							

1111	1110	A-B							
1110	1111	A-B							
1111	1111	A+B							
0000	0000	A-B							

A.21 *Dados A y B (8 bits signados en C.B.) realice la operación e indique el resultado*

A	B	OP	C/B	Res	V	N	A=B	A>B	A<B
11000000	**0**1010000	A-B	No	**0**1110000	**Si**	**No**	No	No	**Si**
11000000	01010000	A+B	C	00010000	No	No	No	No	Si
11110001	11110001	A-B							
11110111	11111000	A-B							
11111001	11111000	A-B							
11111000	11110001	A-B							
11111000	11111000	A+B							

A.22 *Exprese los siguientes números binarios en decimal*

Ej: $1,11010_2 = 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4} + 0 * 2^{-5} = 1 + ½ + ¼ + 1/16 = 1,8125_{10}$

$101,00110_2$	$1010,0111_2$	$1010011,0_2$
$10,100110_2$	$0,10110_2$	$101001,10_2$
$10100,110_2$	$0,10111_2$	$10100,111111_2$

A.23 *Exprese los siguientes números decimales en binario. Utilice como máximo 8 bits en su parte fraccionaria. En caso de no poder representarse indique qué valor se representa.*

Ej: $8,51_{10} = 1000,10000010_2 = 8,5078125_{10}$

$25,320_{10}$	$1023,0078125_{10}$	$0,01171875_{10}$
$16,625_{10}$	$256,03125_{10}$	$75,75_{10}$
$40,0625_{10}$	$0,00390675_{10}$	$2,2_{10}$

A.24 *Realice las siguientes sumas y restas de números binarios sin signo. Indique si la operación es imposible en números sin signo.*

1101,110011 + 11001,00101	11,001010 – 01,100101	110101,01 + 0,11
1111 – 0,1111	110100,10101 – 1100111,101	1010,1010 – 1010,1010
0,1 – 1,0	11,11 – 01,11	0,001+0,0001

A.25 *Exprese los siguientes números decimales en notación científica*
Ej: 1234567 = 1,234567 * 10^6 = 1,234567e6 ; 0,00123 = 0,123 * 10^{-2} = 0,123e-2

1320322_{10}	$1023,00781_{10}$	$12112,01171875_{10}$
1_{10}	$0,1_{10}$	0000000000075_{10}
$0,000000000625_{10}$	$0,00390675_{10}$	$2,2_{10}$

A.26 *Dados los siguientes números en punto flotante IEEE754 de simple precisión , indique su valor en decimal. (Son simples, no utilice una calculadora o conversor)*

Signo	Exponente	Mantisa	Decimal
0	10000001	1100000 00000000 00000000	7
0	10000010	1100000 00000000 00000000	
1	10000010	1100000 00000000 00000000	
0	01111111	1100000 00000000 00000000	

A.27 *Convierta los siguientes números decimales a IEEE754 de simple precisión, en caso de no poder representarlo indique si el resultado tiene error. Represente también en resultado como un conjunto de dígitos hexadecimales*

Número	Signo	Exponente	Mantisa	Error	Hexadecimal
1024,75	0	10001001	0000000 00011000 00000000	no	44801800
65535,2					
32768,1					
1,25					

A.28 *Convierta los siguientes números representados en IEEE754 de simple precisión a IEEE754 de doble precisión (64 bits)*

1 10001001 01100000000000000000000

0 10000011 11100000000000000000000

A.29 *Explique como el siguiente programa imprime en pantalla el número 1024,75 (solo para alumnos que hayan cursado Programación)*

```
#include <stdio.h>

void main(){

    unsigned int a = 0x44801800; // 1024,75

    float f = *((float*)&a); // Contenido de Puntero a float apuntando a unsigned int.

    printf("%f\n",f); // Imprime 1024,75
```

}

A.30 Caso real. Investigue los conceptos que desconozca.

En una línea de empaque automático se detectó una anomalía en el peso obtenido de una balanza dinámica. La cajas cuyo peso rondaba los 30 kg se pesaban normalmente, sin embargo algunas cajas cuyo peso rondaba los 35-36 kg eran rechazadas por no llegar al peso mínimo. El peso en el display de la balanza se presentó siempre normalmente, pero cuando se transmitía la *cadena de peso* al PLC se informaba como de 3 o 4 kgs. ¿Cuál era la causa? ¿Qué solución aportaría?

B- Códigos

B.01 *Escriba el código de Gray para 16 elementos. Indique la distancia del código.*

B.02 *Escriba el código Johnson para 10 elementos. Indique el módulo del código Johnson para N bits. Indique la distancia del código.*

B.03 *Investigue el código "one-hot" e implemente uno para 10 elementos. Indique el módulo del código para N bits. Indique la distancia del código.*

B.04 *Escriba en Binario, Decimal y hexadecimal los valores para los siguientes elementos codificados en ASCII (8 bits):*

0,1,2,3,4,5,6,7,8,9, A,B,C,X,Y,Z,a,b,c,x,y,z

B.05 *Escriba el código BCD Exceso 3. Indique la distancia del código.*

B.06 *Realice las siguientes sumas representando los dígitos en BCD.*

345+608 ; 1+999 ; 1923+8931 ; 1162 + 895

B.07 *Realice las siguientes sumas representando los dígitos en BCD Exceso 3.*

345+608 ; 1+999 ; 1923+8931 ; 1162 + 895

B.08 *Agregue el bit de paridad de modo que las siguientes palabras queden con paridad par en los unos.*

001010101 ; 010110101; 11111111; 11111 ; 10010101010101010; 0 ; 1 ; 101010

B.09 *Se desean transmitir los siguientes elementos codificados en ASCII 8 bits. Indique qué valor representa en ASCII y luego genere un código de hamming de 12 bits para cada elemento.*

00110110 ; 00110101 ; 00110000 ; 00110010

B.10 *Se recibieron los siguientes 5 elementos codificados en ASCII 8 bits (extendidos con bits de paridad hamming). Indique que se recibió (en ASCII) haciendo correcciones de ser necesario.*

110111010010

010011001001

010100010010

100111000101

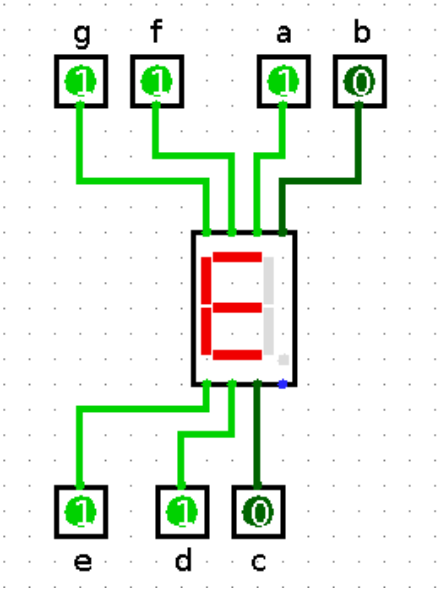
110111110010

C - Álgebra de Boole y circuitos combinatorios

C.01 *Utilizando postulados y teoremas simplifique las siguientes funciones.*

$$F(A,B)=A+A.B$$
$$F(A,B)=A+\overline{A}.B$$
$$F(A,B,C)=(A+B).(A+C)$$

C.02 *Utilizando logisim, conecte el siguiente circuito utilizando un display 7 segmentos y 7 pines de entradas.*



C.03 *Dadas las siguientes funciones a,b,c,d,e,f y g para las entradas N_3 ,N_2 ,N_1 y N_0 . Escriba cada función en sus dos formas canónicas, simplifique ambas utilizando Karnaugh e implemente el circuito con compuertas en la representación más conveniente.*

Entradas				Salidas						
N_3	N_2	N_1	N_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1

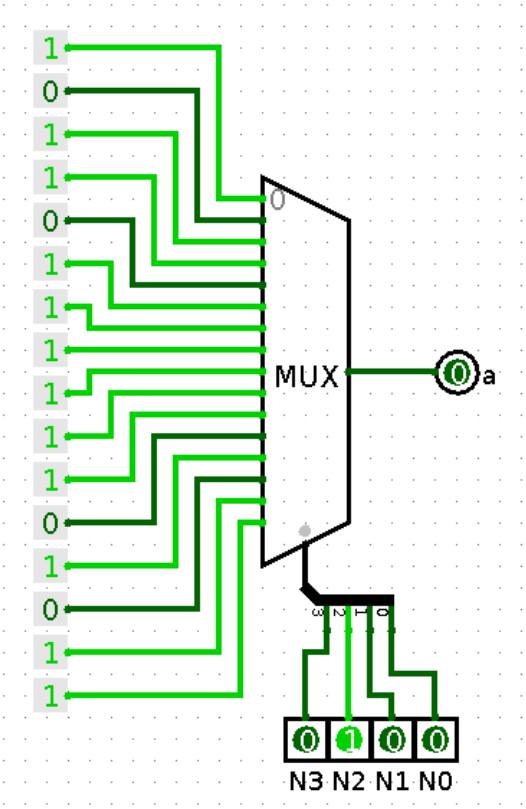
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

C.04 *Implemente cada una de las funciones del punto anterior en logisim (en el formato que prefiera, sea suma de productos o producto de sumas). Las entradas N3,N2,N1 y N0 se implementan como pines de entrada. Las salidas (a,b,c,d,e,f y g) se conectan a las entradas de un display 7 segmentos (vea el punto C.02 para saber que pin corresponde con qué función). Verifique el funcionamiento correcto representando todas las entradas y verificando que el display represente correctamente el valor de las entradas. Es decir cuando la entrada es por ejemplo 0000 el display 7 segmentos representa el 0. Cuando la entrada es 1000 el 7 segmentos representa el 8. Este nuevo circuito posee 4 entradas y 7 salidas y debe llamarse BCDa7SEG.*

C.05 *Implemente un codificador simple para 16 entradas (0 a F). Implemente el mismo en logisim utilizando 16 pines de entrada y 4 pines de salida. Luego conecte la salida del codificador a la entrada del circuito del punto anterior. Verifique el funcionamiento probando todas las entradas válidas del codificador (una entrada en uno a la vez).*

C.06 *Conecte en paralelo los 16 pines de entrada del circuito anterior a una compuerta NOR de 16 entradas. Nombre la salida NADA. Luego utilice la salida NADA en el circuito implementado en el punto anterior de forma tal que si NADA=1 entonces el display 7 segmentos se encuentra todo apagado. En el momento que un pin de entrada se convierte en 1 el display 7 segmentos debe representar que pin está activado. Si ningún pin está activado el display debe estar todo apagado. Logre esto con una compuerta AND en cada entrada del 7 segmentos de forma que sea entrada AND !NADA.*

C.07 *Implemente las funciones a,b,c,d,e, f y g con un multiplexor de 4 entradas de selección como se representa en la imagen. Las entradas del multiplexor son constantes. Note que para conectarse a las entradas de selección se necesita de un BUS de 4 cables. Utilice un splitter con ancho 4 como se ve en la imágen para conectar las entradas N. Luego conecte las salidas de los multiplexores al display 7 segmentos.*



C.08 *Implementar en logisim un sumador de números de dos números de 4 bits. Como entradas tiene los números A (A_3 A_2 A_1 A_0) y B (B_3 B_2 B_1 B_0). La salida es un número de 4 bits llamado S más un bit de carry.*

C.09 *Modifique el sumador del ejercicio anterior sabiendo que los números de entrada (A y B) son dígitos BCD. El sumador debe tener como salida dos dígitos BCD (representando desde 0000 0000 hasta 0001 1000). Sugerimos detectar aquellos casos donde la suma excede el rango BCD (0000 ~ 1001) y corregir sumando 6.*

C.10 *Implementar un circuito en logisim que tenga como entrada una palabra de 8 bits y como salida genere los 4 bits de paridad hamming.*

C.11 *Implementar un circuito en logisim que reciba 12 bits (8 + 4 paridades hamming) como entrada y se encargue de validar si hay errores en los 12 bits. Conecte la salida de este circuito a un conversor BCD a 7 segmentos y represente en el 7 segmentos 0 si no hay errores o el número de bit (desde 1 hasta C) que tiene un error.*

C.12 *Implementar un circuito en logisim que reciba 12 bits (8 + 4 paridades hamming) como entrada y genere 8 bits de salida los cuales se encuentran corregidos en caso de*

haber un error en los 8 bits de datos. Recomendamos implementar con compuertas XOR para invertir los bits de datos y un decodificador para seleccionar cual se invierte.

C.13 *Implementar un circuito en logisim que reste dos números signados A y B de 4 bits c/u. El restador debe poseer una salida de resultado, una de Borrow y 6 salidas de comparación (A=B, A!=B , A<B , A<=B, A>B y A >=B).*

C.14 *Implementar un circuito que complemente un número de 4 bits a la base.*

C.15 *Combine un restador de dos números de cuatro bits (C.13) con un complementador (C.14) y un conversor BCD a 7 Segmentos (C.07 por ejemplo) para presentar el resultado de la resta. Si el número es negativo utilice un segundo display 7 segmentos donde enciende el segmento g para indicar el signo menos.*

D - Secuenciales

No puede utilizar componentes de logisim como FF, contadores, etc. Solo compuertas y dispositivos de E/S. Arme sus propios componentes como circuitos y reutilicelos.

D.00 *Implemente en logisim los circuitos vistos en la presentación teórica:*

- Flip Flop Tipo D Flanco Ascendente
- Flip Flop Tipo D Flanco Descendente
- Flip Flop Tipo T Flanco Ascendente
- Flip Flop Tipo T Flanco Descendente
- Implemente las entradas forzadas (preset fuerza un 1, reset fuerza un 0) en todos los circuitos de manera sincrónica y asincrónica.

Estos componentes deberán ser cada uno en cada variante un circuito de logisim, y los mismos serán reutilizados en los ejercicios de esta práctica (D.04 en adelante).

D.01 *Dibuje el circuito FF RS utilizando compuertas NAND. Realice las transiciones paso a paso cómo se explica en el apunte teórico (el cual usa NOR). Explique en qué transiciones se producen estados inestables. Luego convierta este FF RS NAND en un FF tipo D.*

D.02 *Realice un diagrama de tiempo donde se reflejen las transiciones en las señales del punto anterior.*

D.03 *Diseñe e implemente un circuito detector de flanco descendente. Realice un diagrama de tiempos donde se represente la detección del flanco.*

D.04 *Diseñe e implemente un registro de desplazamiento que tome un número de 4 bits por una entrada serie (IN) y posea una entrada llamada NotLeft, de forma que si NotLeft=0 el número se desplace a la izquierda y si NotLeft=1 el número de desplace a la derecha. Cada FF debe tener una salida en paralelo. Ejemplos: Dado el número 0011, ingresa por serie 0, luego 0, luego 1, luego 1.. si la entrada NotLeft=0 entonces la salida es 0011, si NotLeft=1 entonces la salida es 1100.*

D.05 *Diseñe e implemente un circuito que tenga entrada paralelo y salida serie. El mismo posee 4 entradas en paralelo (número de 4 bits) el cual se almacena en los FF cuando la señal de entrada PAR=1 y se produce un flanco ascendente de clock. Cuando PAR=0 y se produce un flanco ascendente el contenido de los FF se desplaza hacia la derecha un bit. El circuito debe poseer solo una salida conectada a la salida Q del último FF. Este circuito debe poder recibir un número, por ejemplo: 1100, y sacar en serie 0,0,1,1.*

D.06 *Diseñe e implemente un registro con 2 entradas de control, las cuales definen las siguientes 4 operaciones: 00- Mantiene valor, 01- Desplaza izquierda, 10-Desplaza derecha, 11- Carga en paralelo. Posee 4 entradas de datos, una entrada serie, una salida serie y 4 salidas de datos.*

D.07 *Diseñe e implemente un contador binario sincrónico ascendente módulo 10. Luego diseñe e implemente un contador binario sincrónico ascendente módulo 6. Utilizando lógica combinacional detecte cuando el contador módulo 10 pasa de 1001 a 0000 y utilice esta lógica para incrementar el contador módulo 6. Indique qué valores puede contar este contador en conjunto.*

D.08 *Conecte el contador módulo 10 al conversor BCD a 7 segmentos de la parte de combinacionales.*

D.09 *Implemente un contador que cuenta la siguiente secuencia: 011, 101, 110.*

D.10 *Utilizando un multiplexor de 16 entradas de datos y un contador diseñe un circuito que genere la siguiente secuencia por una salida llamada F: 0110010100000010.*

D.11 *Diseñe e implemente un circuito que tome un número de 4 bits y lo desplace hacia la derecha una cantidad de posiciones representadas por otra entrada de 2 bits. Ej: Si el número es 1010 y la entrada de desplazamiento es 01 se desplaza a la derecha una posición dejando 0101. Si la entrada de desplazamiento es 10 se desplaza dos a la derecha dejando 0010. Si la entrada de desplazamiento es 11 se desplaza tres a la derecha dejando 0001. Si la entrada de desplazamiento es 00 no se desplaza nada y queda el número original 1010. Esto puede realizarlo utilizando lógica secuencial y combinacional mezclada… pero también puede implementarlo como un barrel shifter. Sugerimos hacerlo de las dos formas.*

D.12 *Modifique el circuito anterior para que realice un desplazamiento aritmético (manteniendo el signo como el ejercicio A.19) a la derecha, es decir, teniendo en cuenta el bit de signo.*

D.13 *Diseñe e implemente una FSM cuya única entrada sea una señal de clock (puede utilizar una señal de reset si lo desea para inicializar los FF en 0) y produzca como salida la siguiente secuencia: 010, 101, 110. La secuencia se repite todo el tiempo. Sugerimos implementar 3 estados y la tabla de transición de uno estado a otro.*

D.14 *Realice el diseño de una FSM (diagrama de bloques, tablas de verdad e implementación en logisim) que detecte cuando se ingresa $9 o más en un parquímetro. La entrada está formada por 3 bits que pueden valer 001 ($1), 010 ($2), 101 ($5) y una señal de clock que genera un flanco ascendente cuando la moneda ingresa. La salida es un bit (VÁLIDO) que vale 1 cuando se ingresaron $9 o más. Existe una entrada de Reset de forma tal que si Reset=1 y VÁLIDO=1 la FSM vuelve a contar $9 (VÁLIDO=0).*

D.15 *Realice el diseño de una FSM (diagrama de bloques, tablas de verdad e implementación en logisim) que reciba números separados por espacios en blanco (ASCII 0x20). Los mismos ingresan a la FSM un dígito a la vez usando la entrada D (D7,D6,D5...D0) codificados en ASCII. Solo los caracteres ASCII que representan números y el espacio en blanco deben ser tenidos en cuenta. La FSM posee dos salidas: "PAR" que toma el valor 1 si la cantidad de dígitos del número es par, e "IMPAR" que toma el valor 1 si la cantidad de dígitos del número es impar. Ambas deben actualizarse cuando se recibe el espacio que separa los números y deben volver a cero cuando ingresa el primer dígito del número siguiente. Cuando el dígito ingresado por la entrada D es válido, se produce un flanco ascendente en una entrada llamada CLOCK.*

D.16 *Realice el diseño de una FSM (diagrama de bloques, tablas de verdad e implementación en logisim) para una cerradura electrónica. La misma posee una entrada de teclado de 4 bits D (D3 D2 D1 D0) que representan la tecla presionada en el teclado numérico, y la misma vale 1111 cuando el usuario presiona el botón Numeral (#). Cuando el usuario presiona una tecla la entrada CLOCK genera un flanco ascendente (que permite leer el valor del teclado en ese instante). La salida es un bit llamado ABRIR que debe ponerse en 1 cuando el usuario ingresa el código 6502#. Cualquier combinación incorrecta hace que ABRIR=0. Una vez abierta la cerradura se vuelve a cerrar con el ingreso del próximo dígito.*