PRÁCTICAS “TECNOLOGÍA Y ORGANIZACIÓN DE COMPUTADORES”

1º Grado en Informática – Grupo C

Departamento de Arquitectura y Tecnología de Computadores

Francisco Javier Caracuel Beltrán
caracuel@correo.ugr.es

Índice

PRÁCTICA 1.

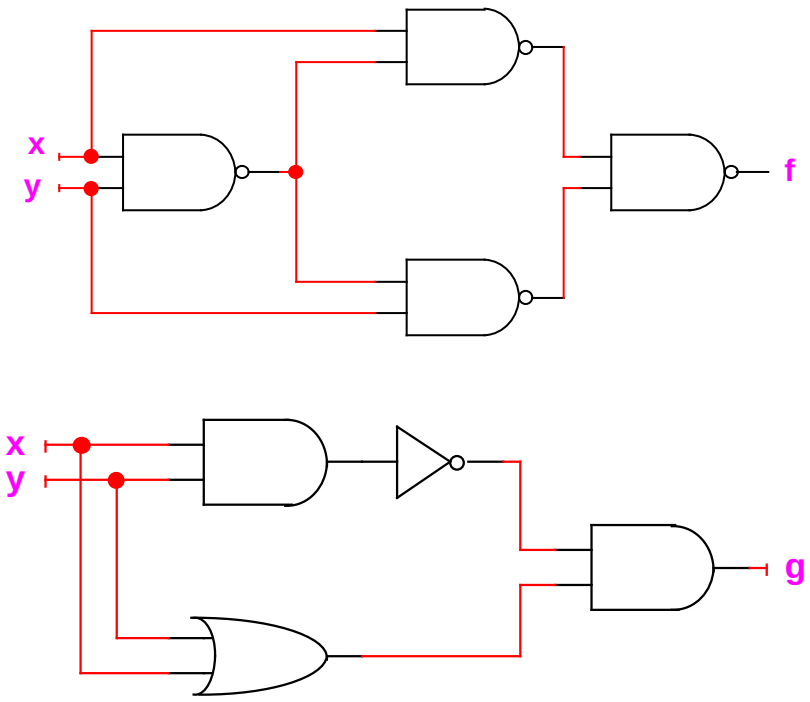
ANÁLISIS Y DISEÑO DE CIRCUITOS COMBINACIONALES CON PUERTAS LÓGICAS.

Objetivos:

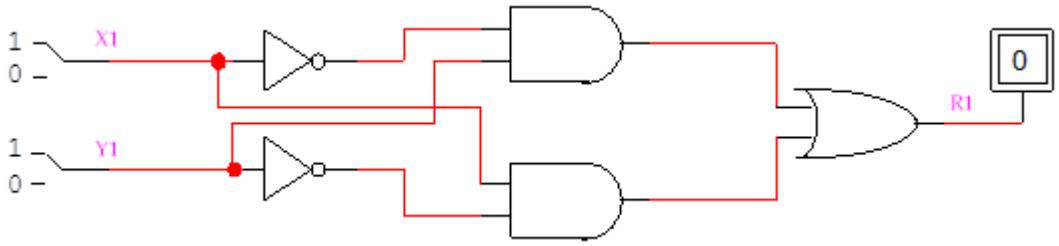
- *Estudiar el funcionamiento de circuitos combinacionales sencillos.*
- *Poner en práctica métodos de simplificación de funciones de conmutación.*
- *Implementar funciones de conmutación con diferentes tipos de puertas lógicas.*

1. Análisis de un sistema combinacional.

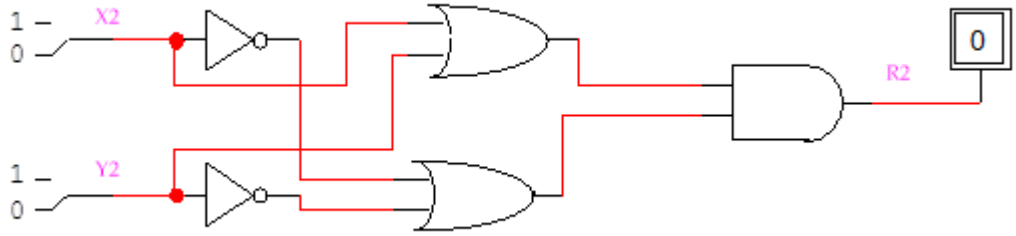
Analice los circuitos de la figura mediante su implementación en el simulador lógico. Obtenga experimentalmente las expresiones algebraicas y las tablas de verdad de las funciones de conmutación *f* y *g* resultantes en la simulación. Minimícelas y obtenga la expresión algebraica mínima de las funciones en un circuito combinacional equivalente mínimo en forma AND/OR y OR/AND. Implemente estos circuitos equivalentes mínimos en el simulador y compruebe que las funciones *f* y *g* resultantes en cada caso son iguales a las de los circuitos originales de la figura.



AND/OR



OR/AND



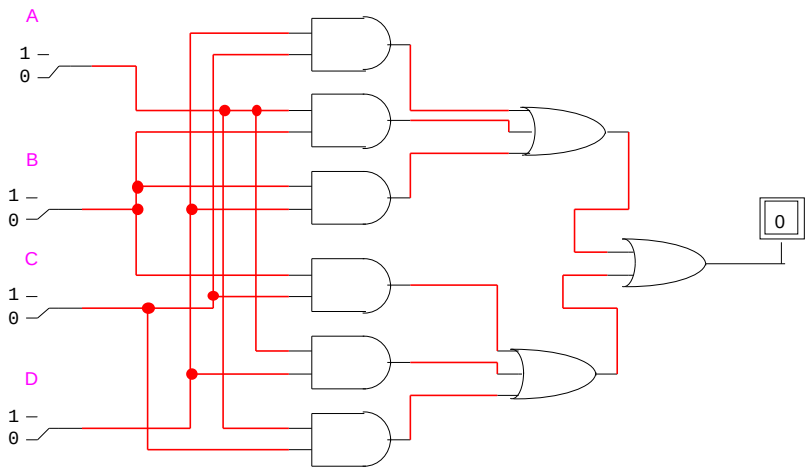
2. Diseño de un sistema combinacional.

El objetivo de esta práctica es el diseño de un circuito lógico combinacional partiendo del enunciado de un problema, modelando el mismo mediante funciones de conmutación y realizando sus tablas de verdad, simplificación e implementación de diferentes formas equivalentes entre si y con la misma funcionalidad. El enunciado del problema es el siguiente:

> "Un jurado consta de cuatro miembros que deben evaluar el examen de un candidato. El candidato aprobará el examen si y sólo si recibe dos o más votos favorables del jurado. Para votar, los miembros del jurado disponen cada uno de ellos de un interruptor (A, B, C ó D) de manera tal que pulsándolo (interruptor = 1) dan su voto favorable al candidato y no pulsándolo (interruptor = 0) dan su voto negativo al candidato."

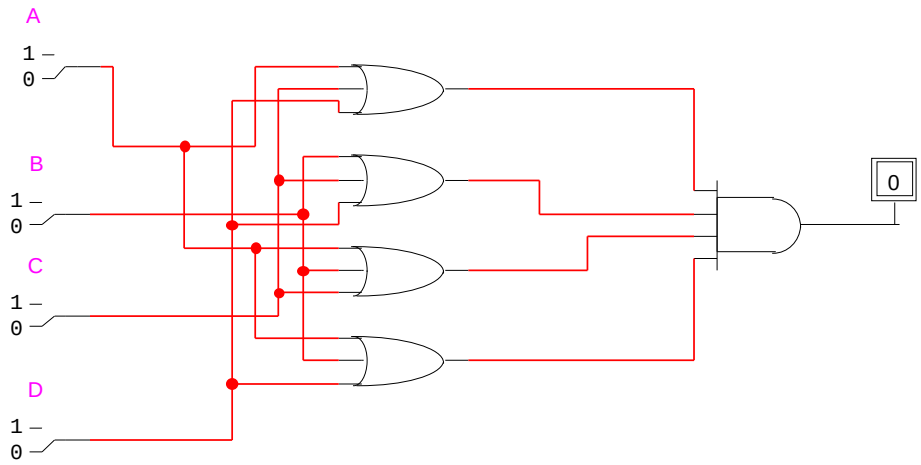
Implemente un circuito lógico mínimo que genere la función que permita determinar si aprueba o suspende un candidato tomando como entradas los cuatro pulsadores A, B, C y D de que dispone el tribunal. Para ello, realice la tabla de verdad de la función, minímicela e implemente dicha función empleando:

a) Síntesis AND/OR. Mediante el simulador lógico.



b) Síntesis NAND/NAND. Mediante el entrenador lógico.

c) Síntesis OR/AND. Mediante el simulador lógico.



d) Síntesis NOR/NOR. Mediante el simulador lógico.

Emplee en cada caso las puertas lógicas de que dispone en el laboratorio de prácticas (simulador o entrenador lógico), haciendo las transformaciones y/o agrupaciones de puertas que estime oportunas en caso de no disponer de las puertas lógicas que resulten de la minimización de la función que representa al problema.

PRÁCTICA 2.

DISEÑO DE CIRCUITOS ARITMÉTICOS.

SUMADORES Y RESTADORES.

Objetivos:

- *Montar y verificar el funcionamiento de sumadores binarios con propagación del acarreo en cascada a partir de semi-sumadores.*
- *Construir un sumador/restador binario a partir de un sumador binario y puertas lógicas.*

2.1. Circuito semi-sumador:

Utilizando el simulador lógico, realice y compruebe el funcionamiento de un semi-sumador binario cuya tabla de verdad se representa en la Tabla 2.1. Cree un símbolo para el semi-sumador como el que se representa en la Figura 2.1.

A_0 B_0	S_0 (suma)	C_1 (acarreo)
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

Tabla 2.1

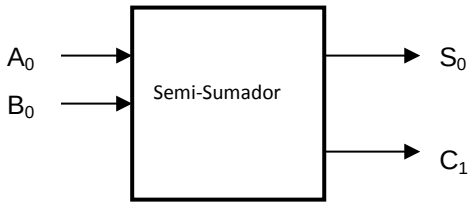
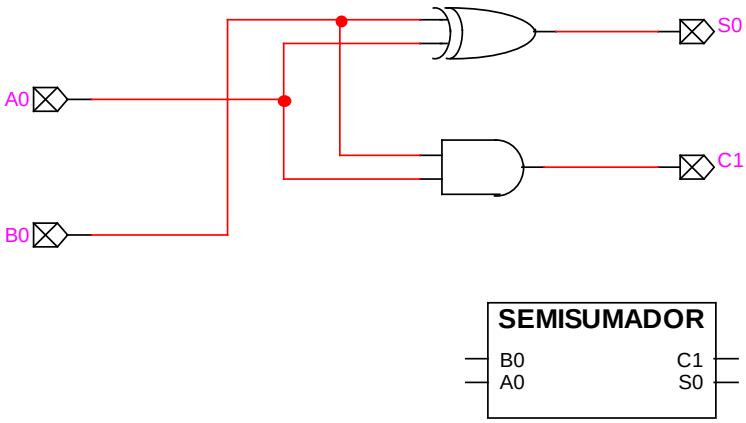
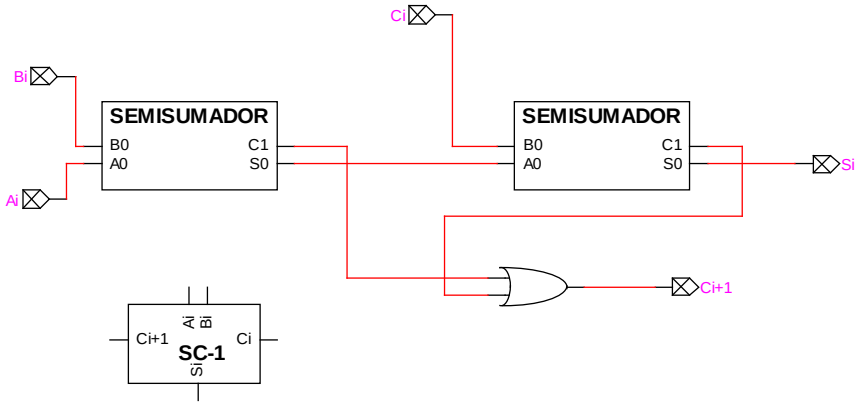


Figura 2.1



2.2. Circuito sumador completo de 1 bit:

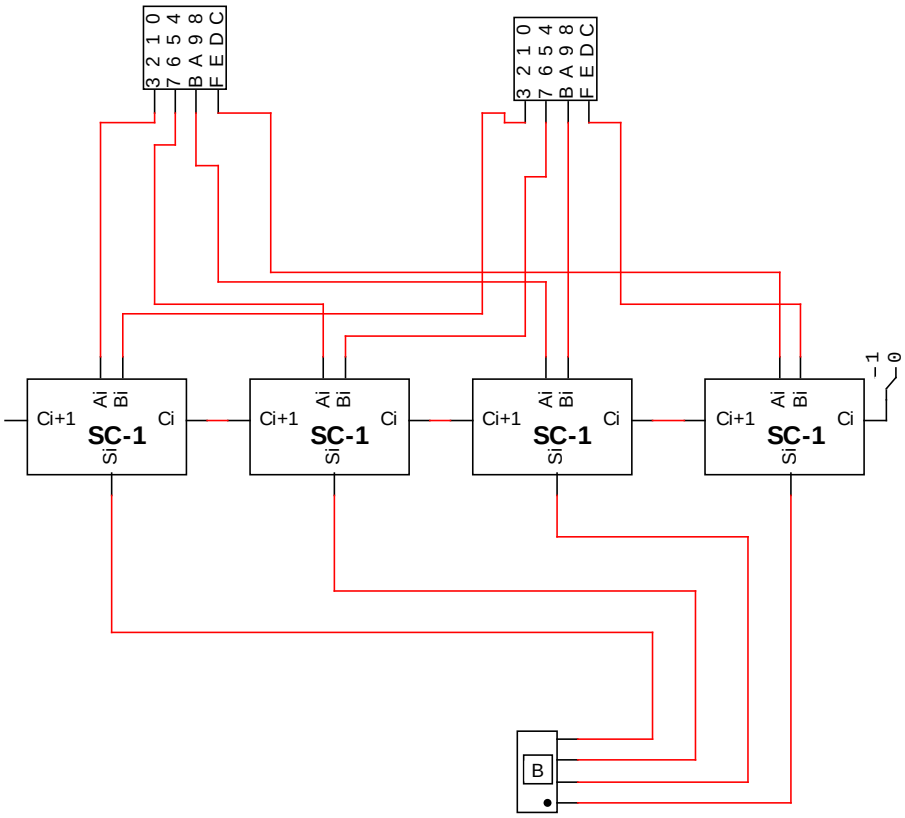
A partir de dos semi-sumadores (como el realizado en el apartado 2.1) y las puertas lógicas que considere oportunas del simulador, ***construya un sumador completo de 1 bit****. El símbolo del sumador completo se muestra en la Figura 2.2. Guarde dicho circuito con el nombre SC1.cct y añada su símbolo en una librería denominada* milib.clf*.*



Ai	Bi	Ci	Ci+1	Si
0	0	0	0	**0**
0	0	1	0	**1**
0	1	0	0	**1**
0	1	1	1	**0**
1	0	0	0	**1**
1	0	1	1	**0**
1	1	0	1	**0**
1	**1**	**1**	**1**	**1**

2.3. *Circuito sumador completo de 4 bits:*

Utilizando cuatro sumadores completos de 1 bit como el diseñado en el apartado 2.2, ***realice un sumador completo*** *para datos de 4 bits. Utilice los componentes HEX_DISPLAY y HEX KEYBOARD wo/STB de la biblioteca* Simulation IO.clf *para realizar el test del circuito. Guarde dicho circuito con el nombre ADD_4.cct y añada su símbolo a la librería* milib.clf.



2.4. Circuito sumador/restador de 4 bits:

Realice un sumador/restador *de 4 bits, añadiendo al sumador binario de 4 bits realizado en el apartado 2.3 las puertas lógicas que considere necesarias. En el caso de la resta, ésta se realizará sumando al minuendo el complemento a dos del sustraendo. Guarde dicho circuito con el nombre ADD_SUB_4.cct y añada su símbolo a la librería* milib.clf.

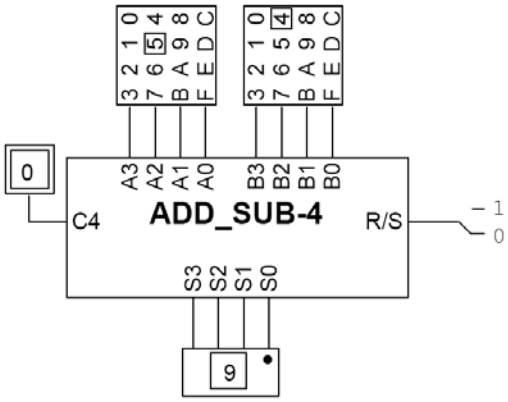
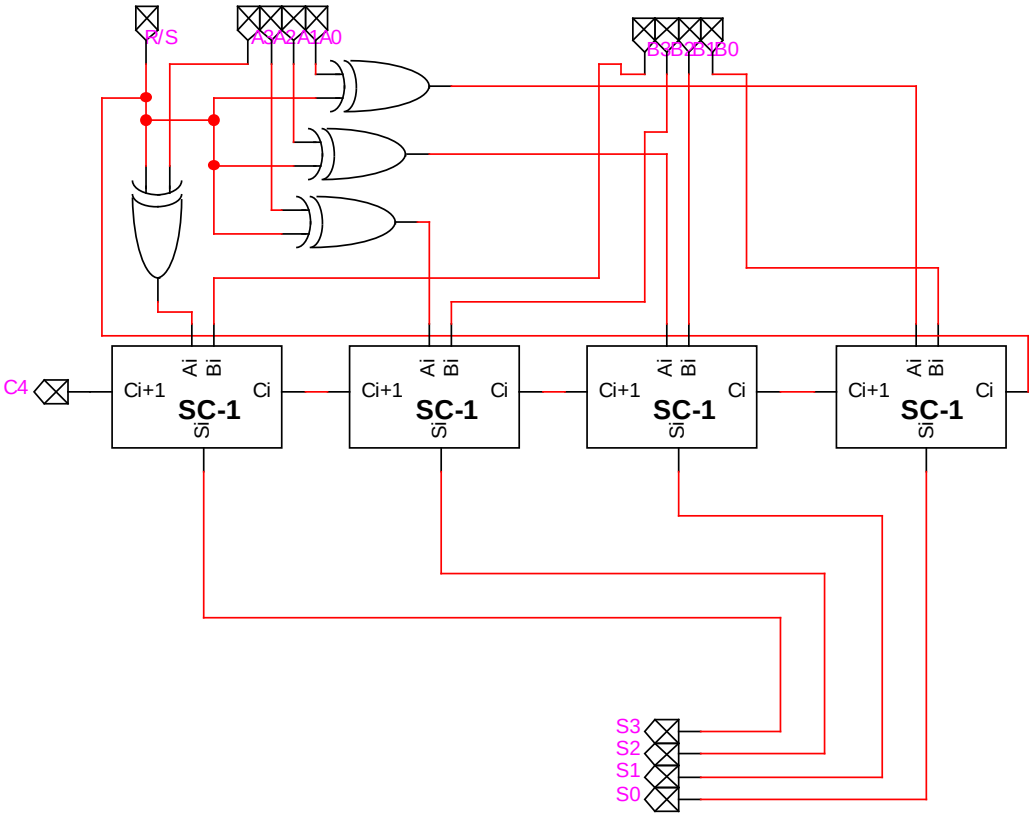


Figura 2.3



PRÁCTICA 3.

ANÁLISIS Y REALIZACIÓN DE UNA UNIDAD ARITMÉTICO-LÓGICA (ALU) DE 4 BITS.

Objetivos:

- *Analizar teóricamente una Unidad Aritmético-Lógica de 4 bits a partir de los esquemáticos de los circuitos que la implementa. Montar y verificar su funcionamiento en un simulador lógico.*

3.1. Estudio preliminar:

Se pretende analizar, realizar y verificar, utilizando un simulador lógico, una Unidad Aritmético-Lógica (ALU) que sea capaz de operar con 2 datos A y B de 4 bits. Esta ALU proporcionará las operaciones aritméticas y lógicas indicadas en la *Figura 3.1*. Cada operación se seleccionará según el valor que tomen (1 ó 0) unas señales de control S_2, S_1, S_0 y un acarreo de entrada C_{in}.

$G = A - 1$	Decrementar A
$G = A + B$	Sumar: A más B
$G = A$	Transferir A
$G = \overline{A}$	Transferir complemento de A
$G = A \wedge B$	Operación A AND B
$G = A + B + 1$	Sumar A más B más 1
$G = A \oplus B$	Operación: A EXOR B
$G = A + \overline{B}$	Sumar A con el complemento a 1 de B
$G = A + 1$	Incrementar A
$G = A \vee B$	Operación: A OR B
$G = A + \overline{B} + 1$	Restar: A –B (en complemento a 2)

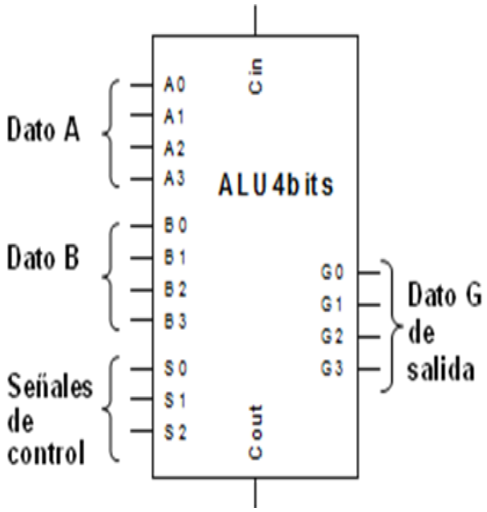
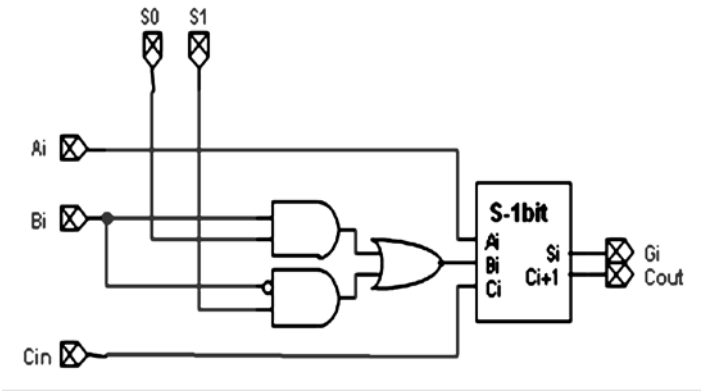
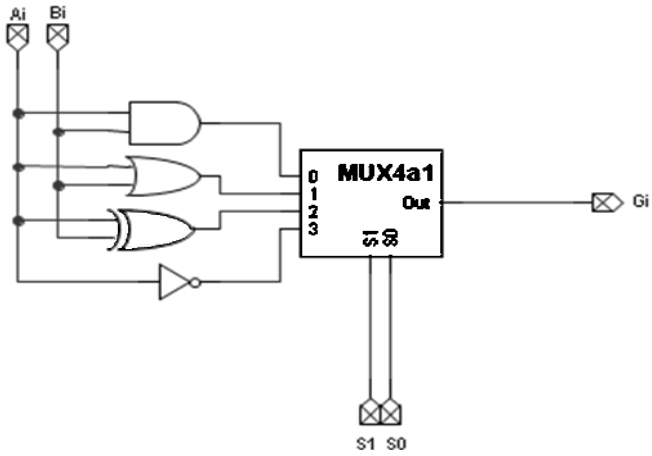


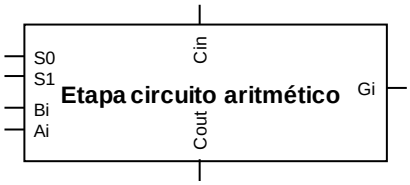
Figura 3.1: Operaciones que realiza la ALU. Esquema general de entradas y salidas.

La estructura jerárquica modular mediante Logic Works de la ALU se establece en las *Figuras 3.2,*

3.3 y 3.4. Se parte de los circuitos de la *Figura 3.2(a) y 3.2(b)* que corresponden a las etapas aritmética y lógica, respectivamente. Ambas operan sobre datos de un bit. La etapa aritmética incluye a un sumador completo de un bit (S-1bit) y la etapa lógica un multiplexor 4:1. Estos módulos se "encapsulan" dando lugar a los bloques indicados en la parte inferior de las *Figura 3.2(a) y (b).*







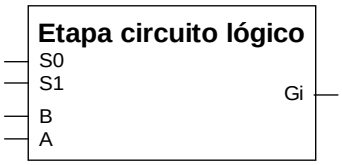
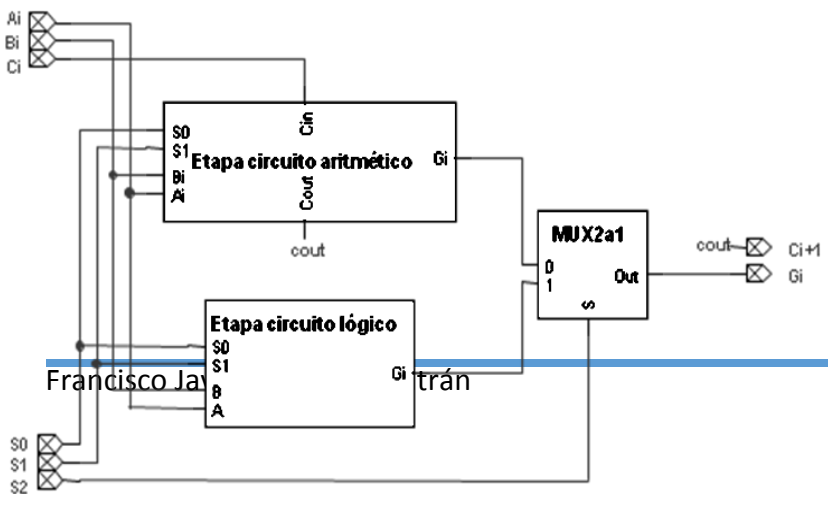


Figura 3.2(a)

Figura 3.2(b)

Figura 3.2. Estructura modular de cada elemento aritmético o lógico de 1 bit.



En la *Figura 3.3* se ilustra el diseño y "encapsulado" de la Etapa ALU de un bit que integra los módulos anteriores de la *Figura 3.2*.

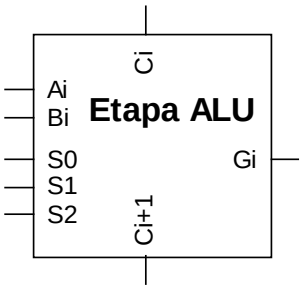


Figura 3.3. Estructura de la Etapa ALU de un bit.

En la *Figura 3.4* se ilustra el diseño y "encapsulado" de una ALU de 4 bits, utilizando 4 Etapas ALU de un bit, como las diseñadas en la *Figura 3.3*.

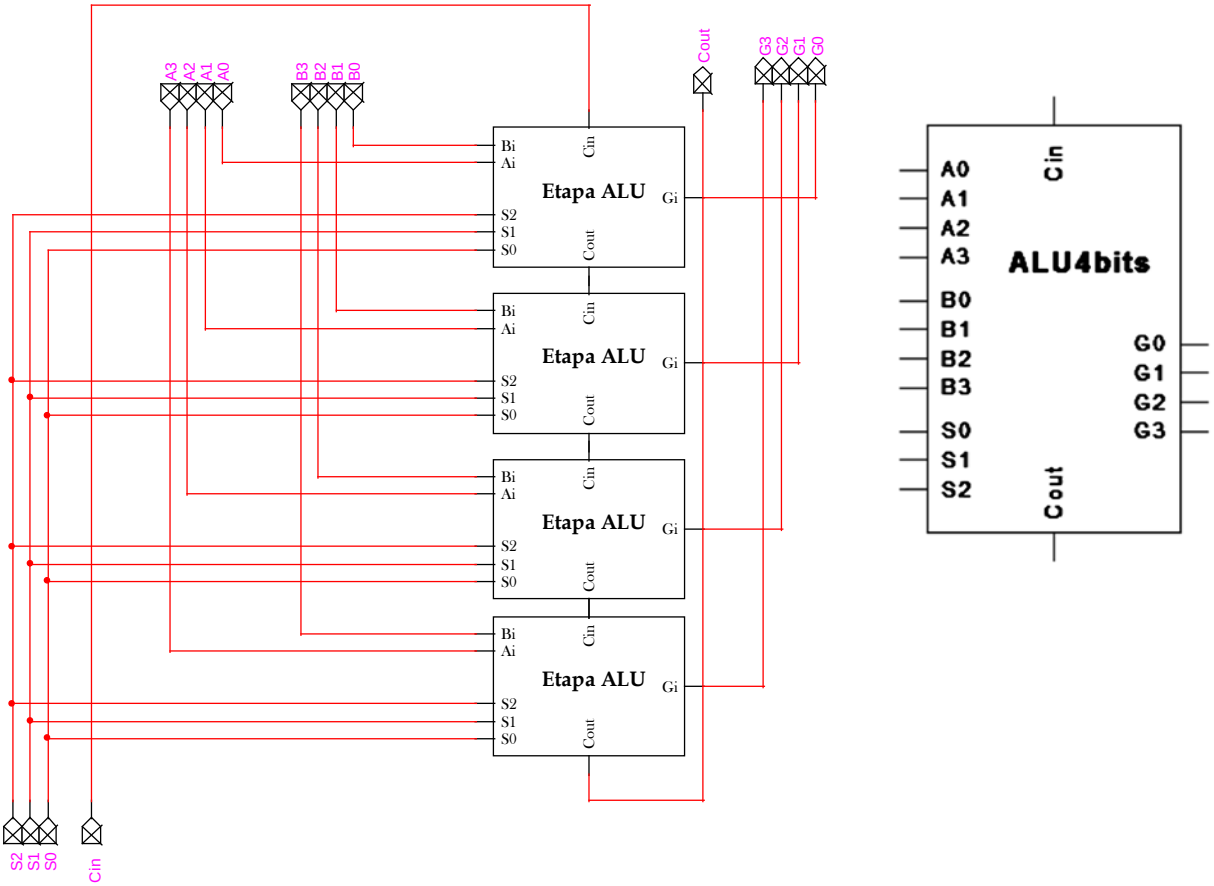


Figura 3.4. Estructura completa de la ALU para palabras de 4 bits.

3.2. Realización práctica:

1. *Trabajo para realizar ANTES de la sesión de prácticas (debe venir preparado):*

1.1. *A**nalice teóricamente** los esquemáticos de los circuitos y etapas de la ALU mostradas en las* Figuras 3.2, 3.3 *y* 3.4. *Deduzca, razonadamente, para qué combinaciones de las*

señales de control S_2, S_1, S_0 y valor de C_{in} se realiza cada una de las operaciones indicadas en la Figura 3.1*. El resultado indíquelo en la columna de la* Tabla 3.1 *correspondiente a "Operaciones de la ALU deducidas Teóricamente".*

1.2. *Ponga algunos ejemplos, con datos (A y B) de entrada de 4 bits, y obtenga teóricamente, para cada una de las operaciones dadas en la F*igura 3.1 *el resultado (dato G) que debería obtenerse.*

SEÑALES $S_2S_1S_0C_{in}$	Operaciones de la ALU, deducidas Teóricamente	Operaciones de la ALU deducidas Experimentalmente
0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1		
0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1		
1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1		
1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1		

Tabla 3.1

2. *Realice en LogicWorks la ALU de la* Figura 3.4 *y* ***el circuito de prueba de la*** Figura 3.5*. Para ello, use (de la biblioteca* Simulation IO.clf *) dos componentes HEX KEYBOARD wo/STB para generar los datos $A_3A_2A_1A_0$ y $B_3B_2B_1B_0$, un componente HEX_DISPLAY para visualizar las salidas $G_3G_2G_1G_0$, cuatro BINARY SWITCH para generar las señales $S_2S_1S_0C_{in}$, y un BINARY PROBE para visualizar la salida C_{out} y así realizar la prueba experimental del circuito.*

3. *En el Laboratorio, introduzca los ejemplos, deducidos teóricamente en el apartado (1.2), en el circuito de prueba y* ***deduzca experimentalmente*** *los valores de las señales de control S_2,S_1,S_0 y valor de C_{in} correspondiente a cada una de las operaciones indicadas en la* Figura 3.1 *y* ***rellene la última columna*** *de la* Tabla 3.1

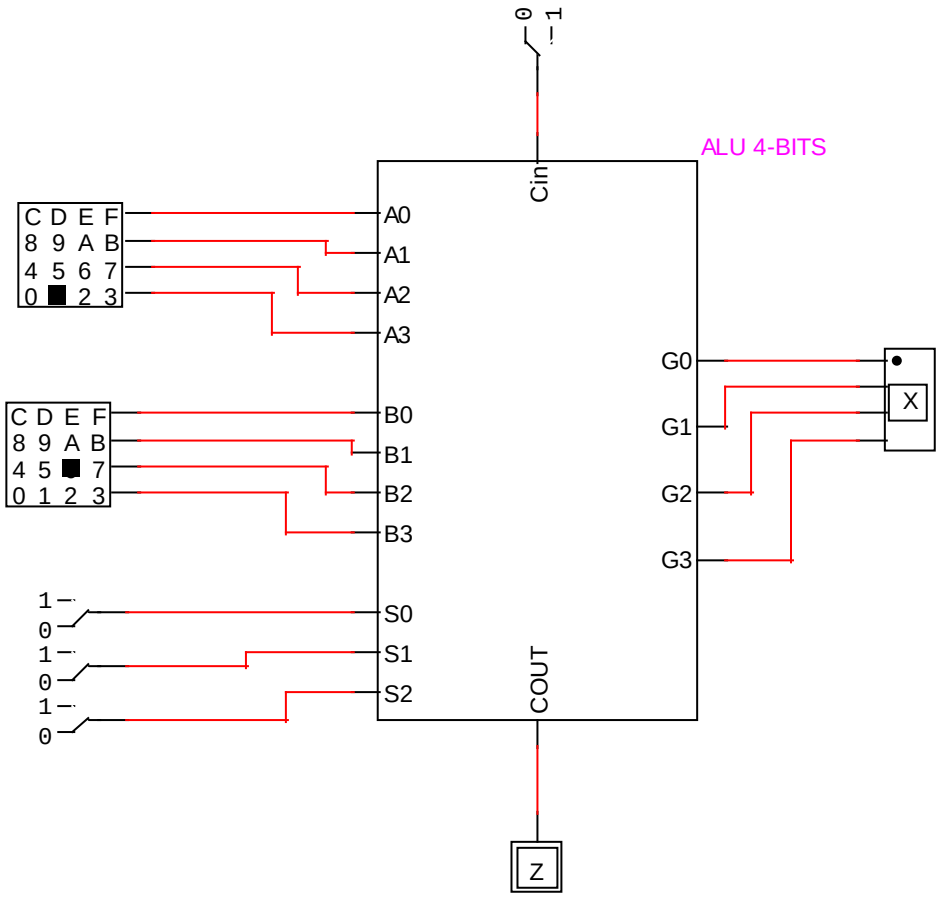
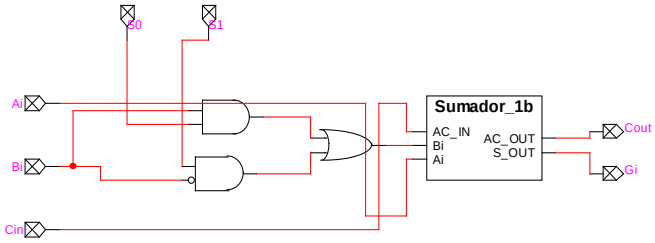
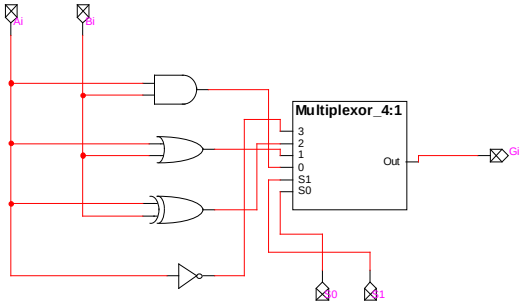


Figura 3.5. Circuito de prueba de la ALU de 4 bits.

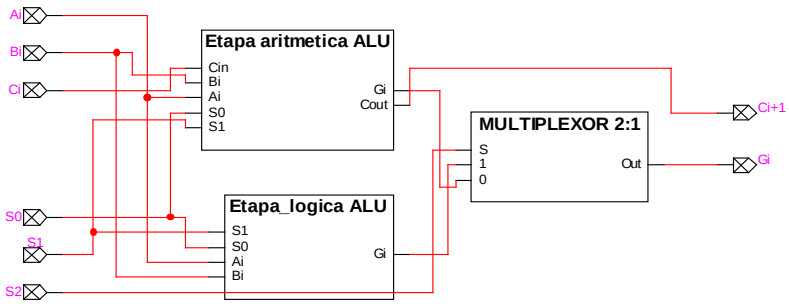
Etapa aritmética



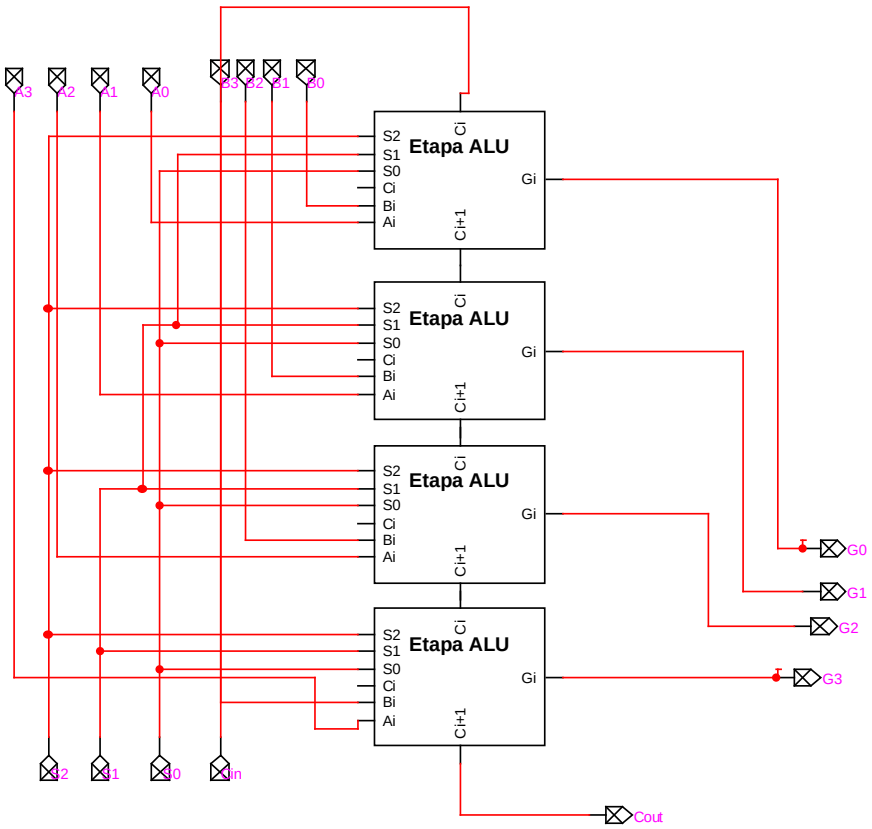
Etapa lógica



Etapa ALU



ALU 4 bits


A3
A2
A1
A0
B3
B2
B1
B0
S2
S1
S0
Ci
Bi
Ai
Etapa ALU
Gi
Ci+1
G0
G1
G2
G3
B2
B1
B0
Bin
Cout

PRÁCTICA 4.
FUNCIONAMIENTO DE CODIFICADORES/DECODIFICADORES Y MULTIPLEXORES/DEMULTIPLEXORES.

Objetivos:

- *Realizar codificadores y decodificadores sencillos.*
- *Realizar multiplexores y demultiplexores sencillos.*
- *Aprender a utilizar multiplexores como generadores de funciones de conmutación.*
- *Comprender la correspondencia entre demultiplexores y decodificadores con señal de habilitación.*

4.1. Decodificadores y codificadores sencillos.

Realice y simule en Logic Works los siguientes circuitos:

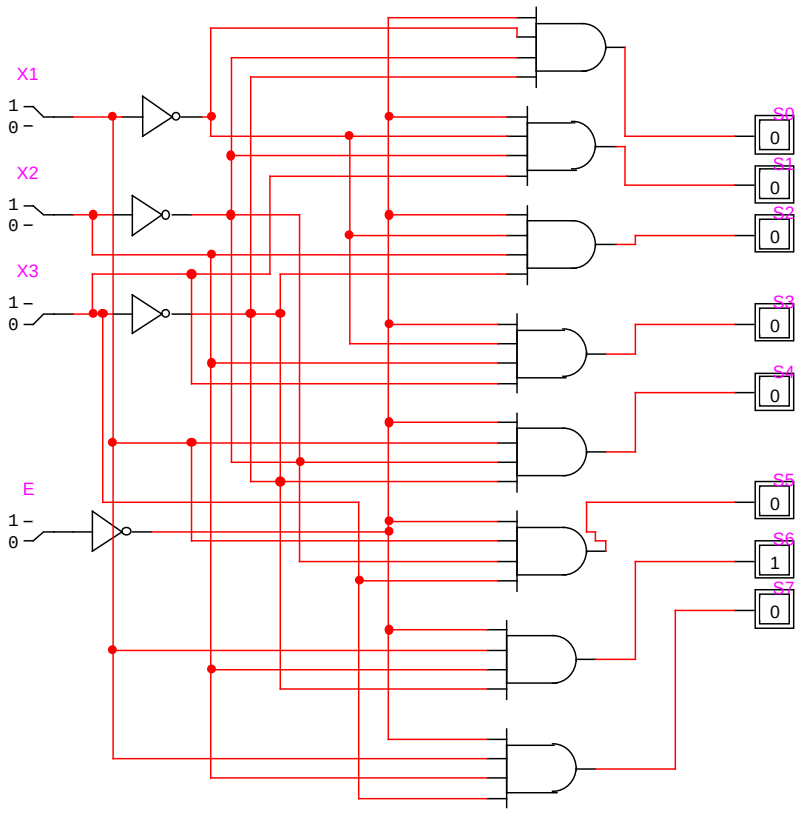
a) Un decodificador binario de 3 entradas y 8 salidas con entrada de habilitación CE.

Tabla de verdad.

E	X1	X2	X3	S0	S1	S2	S3	S4	S5	S6	S7
0	0	0	0	1	0	0	0	0	0	0	**0**
0	0	0	1	0	1	0	0	0	0	0	**0**
0	0	1	0	0	0	1	0	0	0	0	**0**
0	0	1	1	0	0	0	1	0	0	0	**0**
0	1	0	0	0	0	0	0	1	0	0	**0**
0	1	0	1	0	0	0	0	0	1	0	**0**
0	1	1	0	0	0	0	0	0	0	1	**0**
0	1	1	1	0	0	0	0	0	0	0	**1**
1	0	0	0	0	0	0	0	0	0	0	**0**
1	0	0	1	0	0	0	0	0	0	0	**0**
1	0	1	0	0	0	0	0	0	0	0	**0**
1	0	1	1	0	0	0	0	0	0	0	**0**
1	1	0	0	0	0	0	0	0	0	0	**0**
1	1	0	1	0	0	0	0	0	0	0	**0**
1	1	1	0	0	0	0	0	0	0	0	**0**
1	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

$S0 = \overline{E}*X1*X2*X3$ $S1 = E*\overline{X1}*X2*X3$ $S2 = E*X1*\overline{X2}*X3$ $S3 = E*X1*X2*\overline{X3}$

$S4 = \overline{E}*\overline{X1}*X2*X3$ $S5 = E*\overline{X1}*\overline{X2}*X3$ $S6 = E*X1*\overline{X2}*\overline{X3}$ $S7 = \overline{E}*X1*X2*X3$

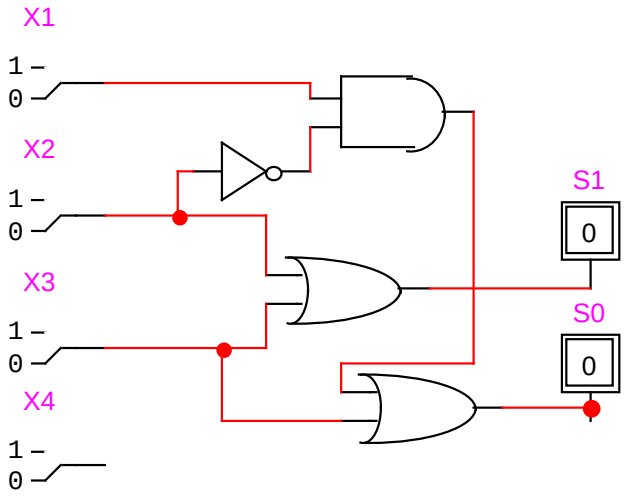


b) Un codificador binario con prioridad de 4 entradas y 2 salidas.

Tabla de verdad

X1	X2	X3	X4	S0	S1
0	0	0	0	-	-
0	0	0	1	0	**0**
0	0	1	0	0	**1**
0	0	1	1	0	**1**
0	1	0	0	1	**0**
0	1	0	1	1	**0**
0	1	1	0	1	**0**
0	1	1	1	1	**0**
1	0	0	0	1	**1**
1	0	0	1	1	**1**
1	0	1	0	1	**1**
1	0	1	1	1	**1**
1	1	0	0	1	**1**
1	1	0	1	1	**1**
1	1	1	0	1	**1**
1	**1**	**1**	**1**	**1**	**1**

S1= X2+X3 S0= X3+$\overline{X2}$*X1



4.2. Conversor a siete segmentos.

Un conversor de códigos es un circuito combinacional con *n* entradas y *m* salidas tales que para cada combinación de entradas se genera una y sólo una combinación de salida. En esta práctica se va a realizar un conversor de código para asignar a dígitos decimales del 0 al 7 un código que permita encender o apagar los led's de un visualizador de 7 segmentos. Realice, utilizando Logic Works un conversor de código para un visualizador de 7 segmentos. Para su realización hay que saber:

a) Los ocho dígitos se codifican en binario con los valores 000 para el 0 hasta el 111 para el 7 utilizando los conmutadores binarios de Logic Works (Binary Switch).
b) Un visualizador de 7 segmentos tiene 7 led's que se encienden o se apagan dependiendo de si hay un 1 (encendido) o un 0 (apagado) en su entrada. Para simularlo, se utiliza el visualizador de 7 segmentos disponible en la biblioteca "SIMULATION IO" de Logic Works.
c) Teniendo en cuenta que dichos led's en el display de 7 segmentos reciben un nombre (a, b, c, d, e, f y g, ver figura 4.1), se tendrán que realizar 7 funciones de 3 variables 2 para conseguir la codificación adecuada. Complete para su realización la tabla de verdad de las 7 funciones (Tabla 4.1) y minimice dichas funciones. Implemente la expresión mínima de dichas funciones.

CÓDIGO ABC	Nº	a	b	c	d	e	f	g
000	**0**	1	1	1	1	1	1	**0**
001	**1**	0	1	1	0	0	0	**0**
010	**2**	1	1	0	1	1	0	**1**
011	**3**	1	1	1	1	0	0	**1**
100	**4**	0	1	1	0	0	1	**1**
101	**5**	1	0	1	1	0	1	**1**
110	**6**	1	0	1	1	1	1	**1**
111	7	**1**	**1**	**1**	**0**	**0**	**0**	**0**

AB C	**00**	**01**	**11**	**10**
0	1	1	1	0
1	0	1	1	1

$$a=B+AC+A\bar{C}$$

AB C	**00**	**01**	**11**	**10**
0	1	1	0	1
1	1	1	1	0

$$b=A+\bar{B}C+B\bar{C}$$

AB C	**00**	**01**	**11**	**10**
0	1	0	1	1
1	1	1	1	1

$$c=C+B+A$$

AB C	00	01	11	10
0	1	1	0	1
1	0	1	0	1

$d = \bar{A}C + A\bar{B} + \bar{A}\bar{B}$

AB C	00	01	11	10
0	1	1	1	0
1	0	0	0	0

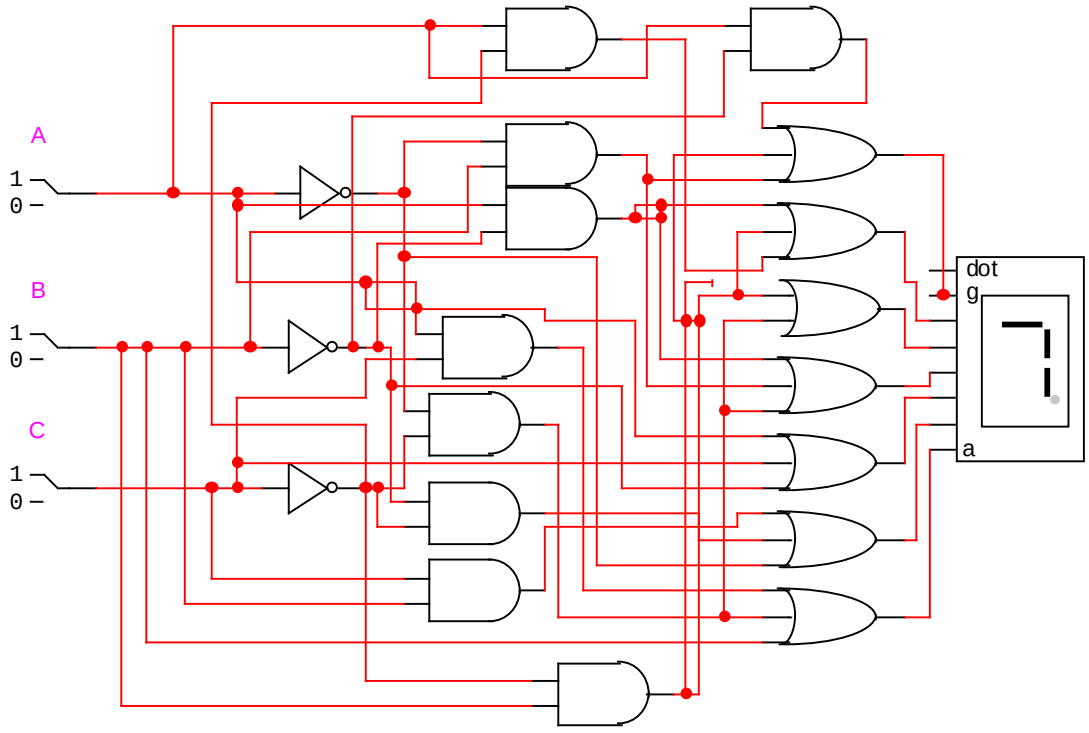
$e = \bar{A}C + B\bar{C}$

AB C	00	01	11	10
0	1	0	1	1
1	0	0	0	1

$f = \bar{A}C + \bar{B}\bar{C} + A\bar{B}$

AB C	00	01	11	10
0	0	1	1	1
1	0	1	0	1

$$g=A\bar{B}+\bar{B}C+\bar{A}\bar{B}$$

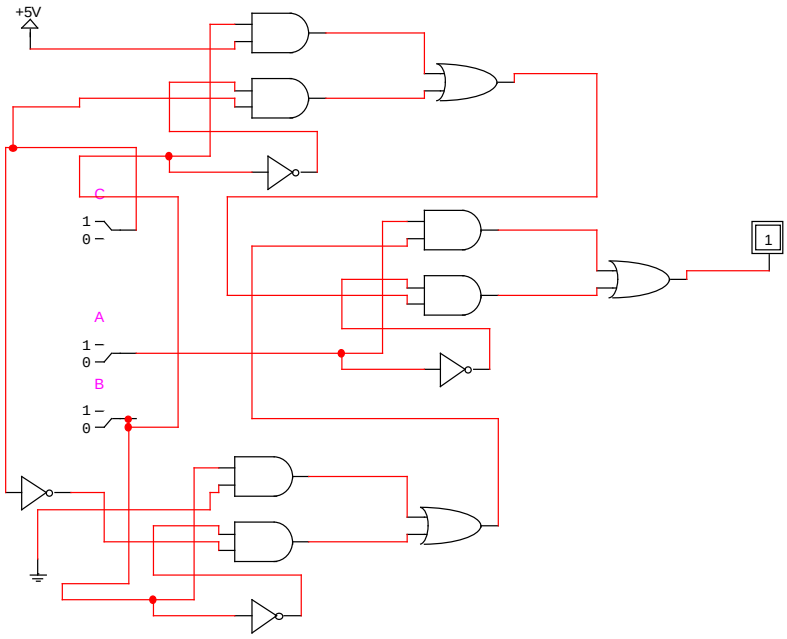


4.3. Síntesis de funciones lógicas con multiplexores.

Implemente la función de tres variables f(A, B, C) cuya tabla de verdad se presenta en la Tabla 4.3, utilizando multiplexores de 2 a 1. Debe realizar cada multiplexor a partir de las puertas lógicas de que dispone en el simulador lógico.

A	B	F
0	0	**C**
0	1	**1**
1	0	$\overline{\mathbf{C}}$
1	**1**	**0**

3 MULTIPLEXORES 2 A 1



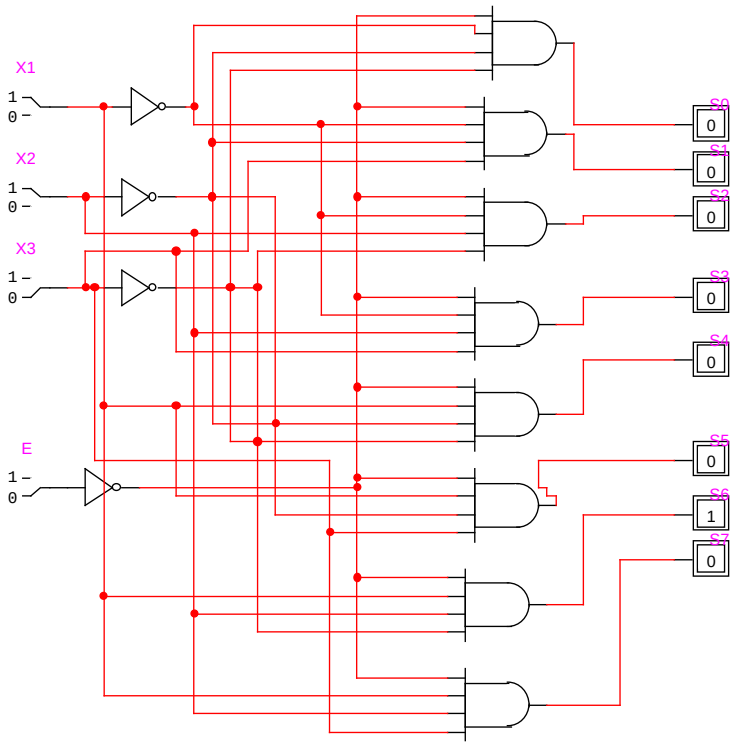
4.4. Realización de demultiplexores.

Realice un demultiplexor de 1 a 8. Compare este circuito con el decodificador binario de 3 entradas y 8 salidas con entrada de habilitación de chip (CE) implementado

Z	X1	X2	X3	S0	S1	S2	S3	S4	S5	S6	S7
0	0	0	0	1	0	0	0	0	0	0	**0**
0	0	0	1	0	1	0	0	0	0	0	**0**
0	0	1	0	0	0	1	0	0	0	0	**0**
0	0	1	1	0	0	0	1	0	0	0	**0**
0	1	0	0	0	0	0	0	1	0	0	**0**
0	1	0	1	0	0	0	0	0	1	0	**0**
0	1	1	0	0	0	0	0	0	0	1	**0**
0	1	1	1	0	0	0	0	0	0	0	**1**
1	0	0	0	0	0	0	0	0	0	0	**0**
1	0	0	1	0	0	0	0	0	0	0	**0**
1	0	1	0	0	0	0	0	0	0	0	**0**
1	0	1	1	0	0	0	0	0	0	0	**0**
1	1	0	0	0	0	0	0	0	0	0	**0**
1	1	0	1	0	0	0	0	0	0	0	**0**
1	1	1	0	0	0	0	0	0	0	0	**0**
1	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

$S0 = \bar{Z}*X1*X2*X3$ $S1 = Z*\bar{X1}*X2*X3$ $S2 = Z*X1*\bar{X2}*X3$ $S3 = Z*X1*X2*\bar{X3}$

$S4 = \bar{Z}*\bar{X1}*X2*X3$ $S5 = \bar{Z}*X1*\bar{X2}*X3$ $S6 = \bar{Z}*X1*X2*\bar{X3}$ $S7 = \bar{Z}*X1*X2*X3$



PRÁCTICA 5.

COMPROBACIÓN EXPERIMENTAL DEL FUNCIONAMIENTO DE LOS BIESTABLES BÁSICOS. IMPLEMENTACIÓN Y FUNCIONAMIENTO DE REGISTROS.

Objetivos:

- Comprobar el funcionamiento de biestables y registros.
- Construir un registro con posibilidad de carga paralelo y desplazamiento.

5.1. Utilización de los biestables del equipo de prácticas:

En el equipo entrenador de prácticas SIDAC DET 2020 existen dos módulos conteniendo, cada uno de ellos, 4 biestables J-K (FF-JK), con entrada de reloj (C) y puesta a cero asíncrona (R). Su símbolo se representa en la Figura 5.1. La entrada externa asíncrona R (CLEAR) se activa por nivel alto, es decir, cuando se pone CLEAR=1 la salida del biestable toma el valor cero (Q = 0).

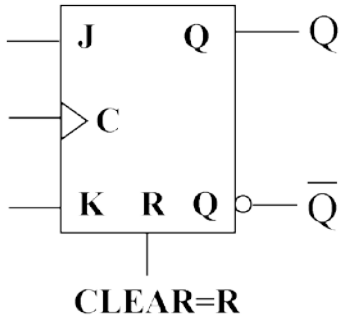


Figura 5.1

Durante el desarrollo de esta práctica y en todas aquellas en la que no se indique lo contrario, la señal de reloj de los biestables del equipo entrenador de prácticas se va a generar de forma manual utilizando uno de los conmutadores “sin rebote” de los que dispone el equipo de prácticas (referido como “PULSOS” en el panel de éste). La justificación es bien simple: en un conmutador convencional, cada vez que se realiza una conmutación se genera un tren de pulsos hasta alcanzar su valor estable, esto

es debido al “rebote” mecánico de sus contactos. Por tanto, si se usa este tipo de conmutador como señal de reloj en un biestable disparado por flanco, éste cambiará de estado por cada flanco del tren de pulsos falseando los resultados experimentales y comportamientos de los circuitos secuenciales que se realicen.

Monte los circuitos de las Figuras 5.2 y 5.3 utilizando los biestables J-K de que se dispone en el quipo de prácticas DET 2020 y asigne un conmutador sin rebotes para la entrada de reloj Ck (“PULSOS”) y un conmutador convencional para la entrada X (referido como “PROGRAMADORES” en el panel del equipo). Reproduzca con dichos conmutadores las secuencias de entradas X y Ck del cronograma de la Figura 5.4 representando en éste las correspondientes salidas Q_1 y Q_2 de los circuitos de las Figuras 5.2 y 5.3.

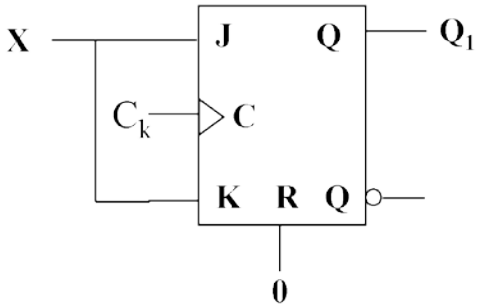


Figura 5.2

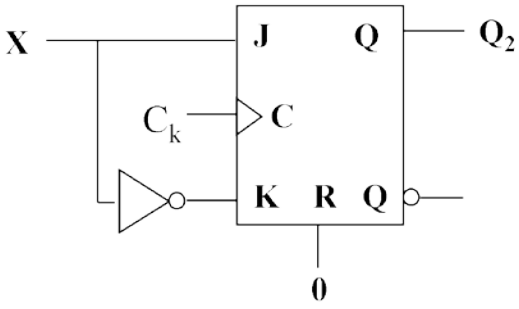


Figura 5.3

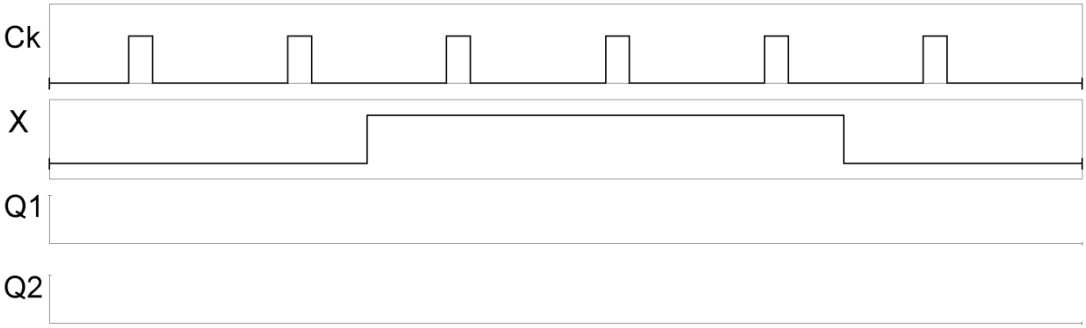


Figura 5.4

A la vista de los resultados obtenidos, responda a las siguientes cuestiones:

a) ¿Son los biestables J-K del equipo de prácticas activos por flanco de subida, flanco de bajada o por nivel alto o bajo?

Las 2 son falco de subida y nivel alto.

b) Indique a qué tipo de biestables (T ó D) corresponden las configuraciones de las Figuras 5.2 y 5.3.

Figura 5.2 es tipo T
Figura 5.3 es tipo D

5.2. Registro de desplazamiento con recirculación:

Un registro de desplazamiento con recirculación de 4 bits tiene la estructura dada en la Figura 5.5.

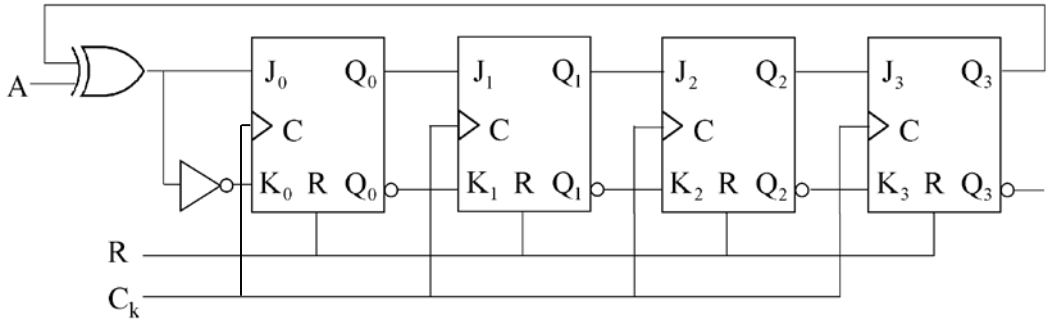
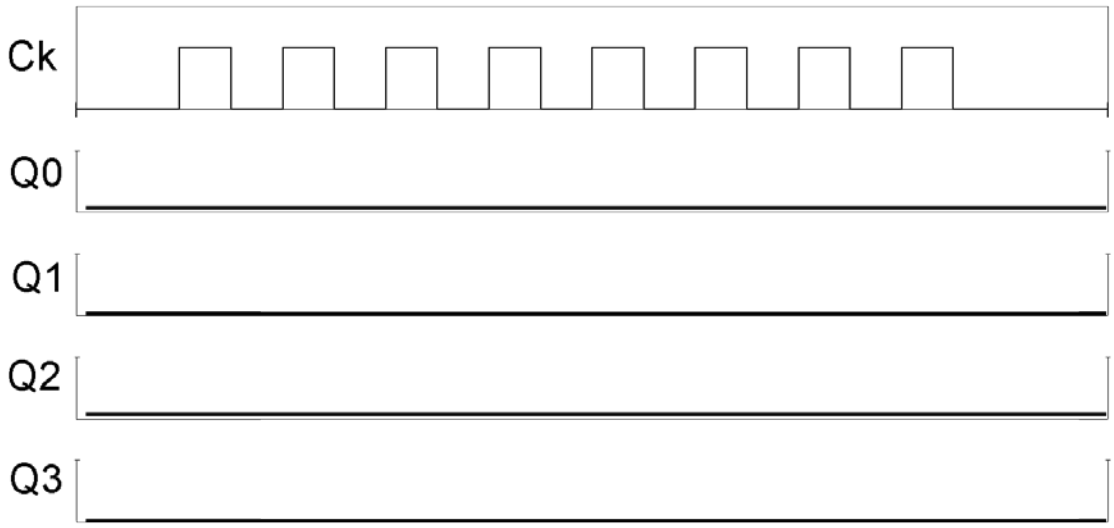


Figura 5.5

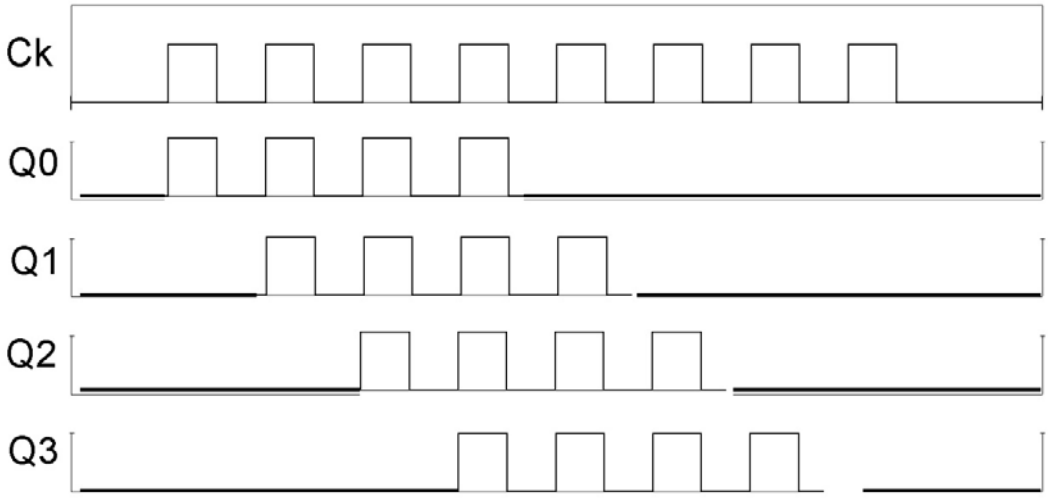
La información que se genera por Q_3 se vuelve a introducir al biestable Q_0 modificada o no por la puerta XOR, según sea el valor de la señal A.

Monte en el entrenador de prácticas SIDAC DET 2020 el circuito de la Figura 5.5, y con estado inicial $Q_0Q_1Q_2Q_3 = 1000$, realice las siguientes cuestiones:

a) Manteniendo A = 0 obtenga las salidas $Q_3Q_2Q_1Q_0$ durante 8 pulsos de reloj.



b) Manteniendo A = 1 obtenga las salidas $Q_3Q_2Q_1Q_0$ durante 8 pulsos de reloj. Realice en ambos casos los cronogramas de las salidas $Q_3Q_2Q_1Q_0$ durante dichos 8 ciclos de reloj en las Figuras 5.6 y 5.7 y comente el resultado obtenido. Como cuestión adicional, explique el procedimiento que ha utilizado para inicializar el registro al valor: $Q_0Q_1Q_2Q_3 = 1000$.

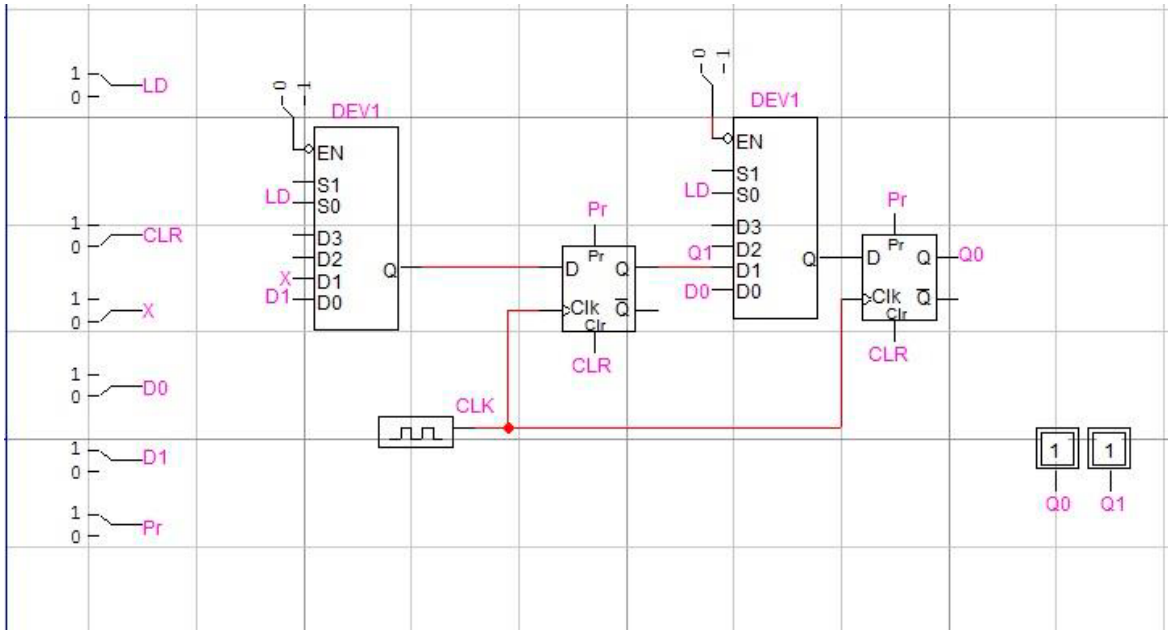


5.3. Registro de desplazamiento con carga paralelo síncrona:

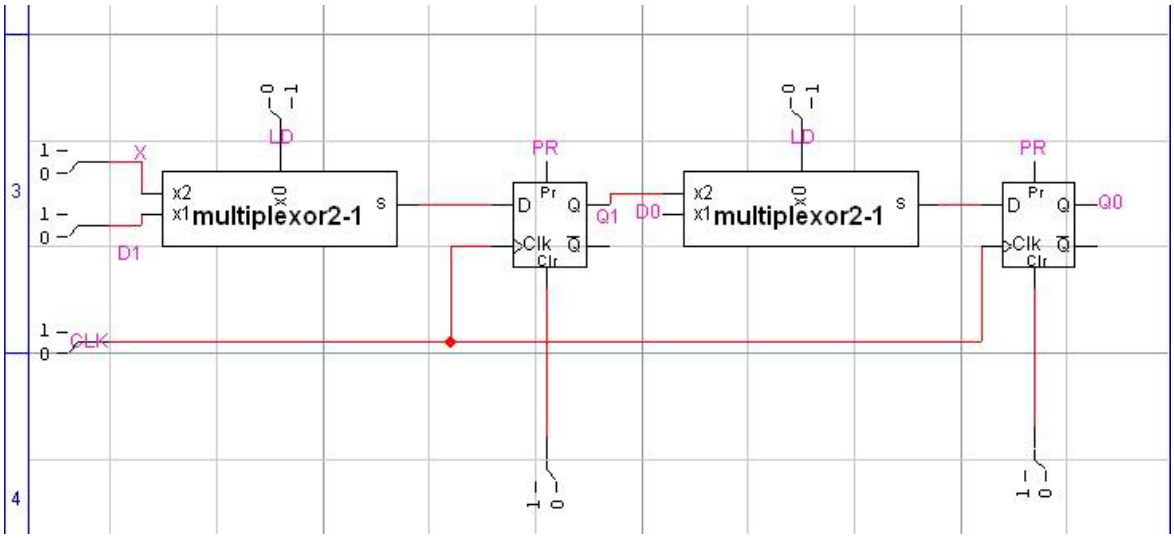
Con frecuencia, es necesario disponer de un registro de desplazamiento con la posibilidad de cargar datos en paralelo. Esto permite, por ejemplo, realizar conversiones no sólo de serie-paralelo o serie-serie, sino además conversiones paralelo-paralelo y paralelo-serie. La carga paralelo síncrona se obtiene añadiendo circuitos que permiten configurar las conexiones de las entradas de los biestables D, bien para conectar en cascada los biestables (operación de desplazamiento) o bien para conectar entradas externas con las entradas D de los biestables (carga paralelo síncrona).

Utilizando puertas lógicas y biestables de Logic Works, realice un registro de 2 bits con desplazamiento hacia la derecha y posibilidad de carga paralelo síncrona. El circuito final debe tener 2 entradas de control que denominadas Reset y Load, dos entradas D_1 y D_0 para introducir un dato de dos bits para la carga paralelo, y una entrada X para introducir datos en serie. Respecto a las señales de control, éstas deben operar del siguiente modo:

a) Si Reset es 1, entonces se ponen a cero los biestables de forma asíncrona, esto es, independientemente de la señal de reloj, en caso contrario (Reset = 0) el biestable opera según el valor que tome la entrada "Load".



b) Con Reset = 0, si Load es 1 se produce la carga paralelo y si Load es 0 se desplaza el contenido del registro, ambas operaciones ocurrirán de forma síncrona, es decir, tendrán lugar coincidiendo con el siguiente flanco activo de la señal de reloj.



PRÁCTICA 6.

IMPLEMENTACIÓN Y FUNCIONAMIENTO DE CONTADORES Y GENERADORES DE SECUENCIAS.

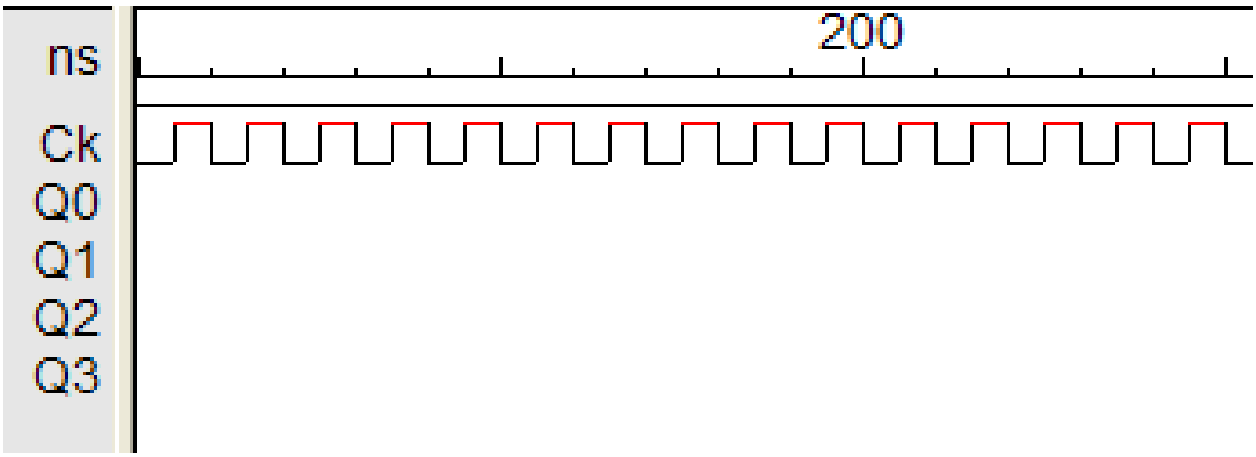
Objetivos:

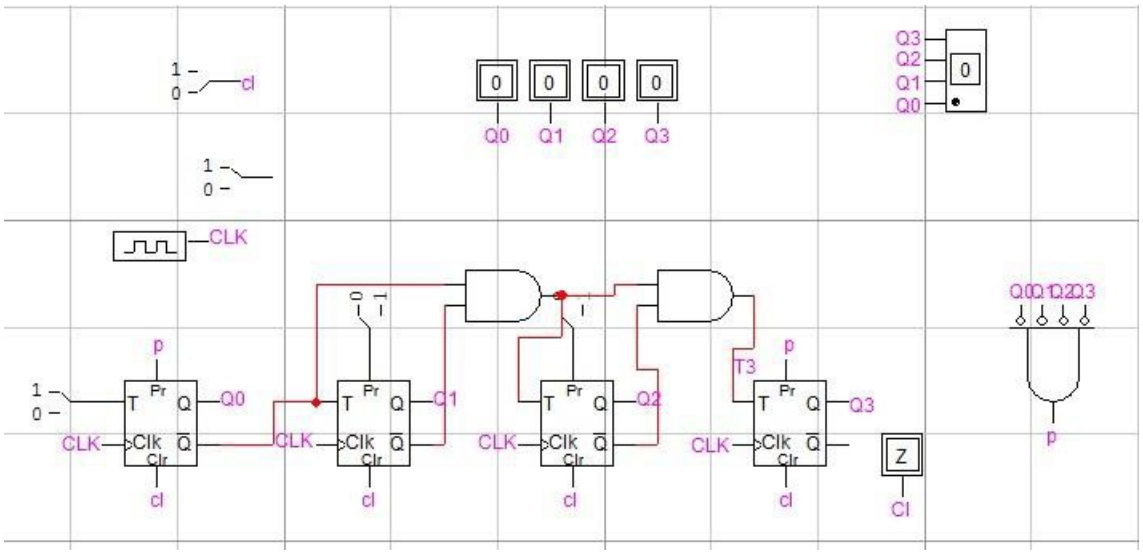
- Diseñar contadores y generadores de secuencias.
- Comprobar el funcionamiento de contadores y generadores de secuencias.

6.1. Contador Síncrono de módulo 10:

Diseñe un contador síncrono descendente módulo 10 que genere la cuenta (en binario)

9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9, 8, ... Simule el circuito utilizando Logic Works y compruebe su funcionamiento mediante un cronograma como el de la Figura 6.1.





6.2. Generador de secuencia síncrono:

Diseñe un generador de secuencia o secuenciador síncrono que produzca, de forma cíclica (en binario) la siguiente secuencia de salidas: 0, 1, 3, 0, 2, 0, 1, 3, 0, 2 ..... Simule el circuito utilizando Logic Works y extraiga un cronograma que refleje su buen funcionamiento. Para ello, implemente un circuito como el de la Figura 6.2 utilizando un generador de reloj (CLOCK) y un componente HEX_DISPLAY para visualizar las salidas. ¿Cuántos estados diferentes se presentan en este sistema secuencial?.

Presenta 16 estados diferentes.

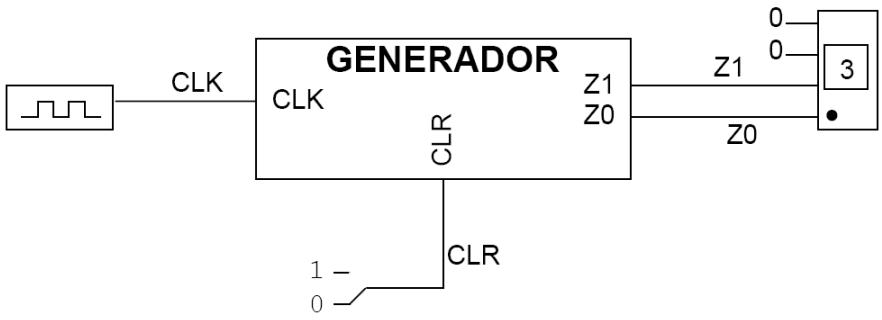
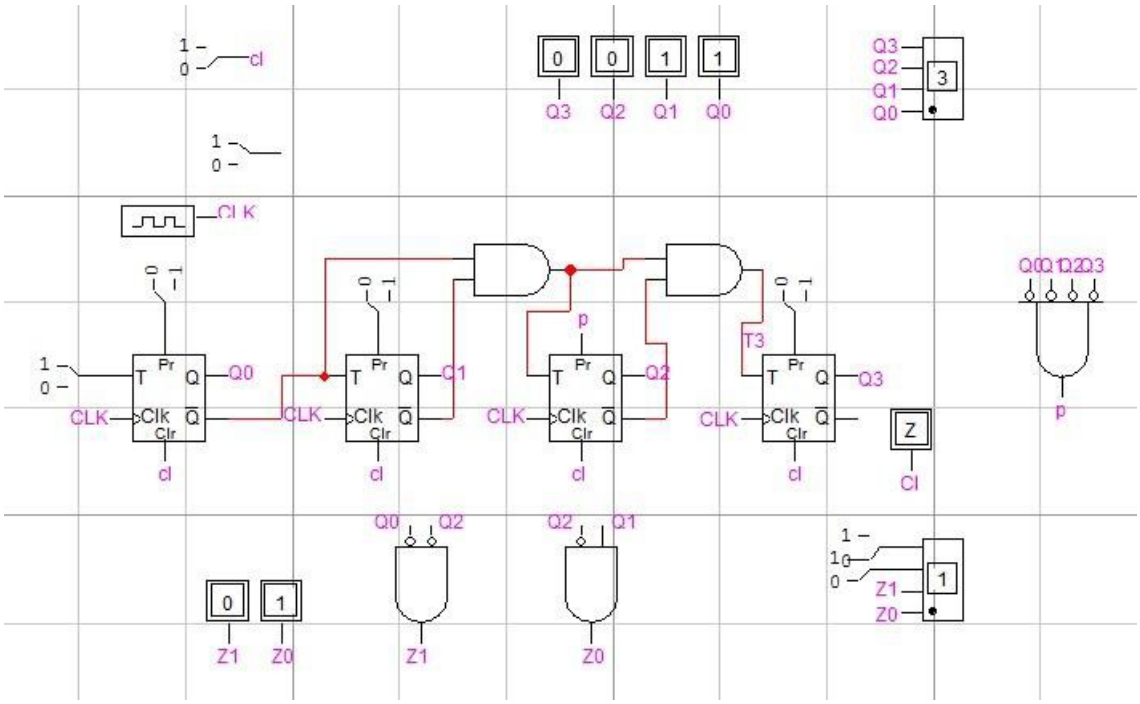


Figura 6.2


1
0
cl
0
0
1
1
Q3
Q2
Q1
Q0
3
CLK
Pr
T
Q
Clk
Clr
CLK
cl
Q0
Q1
Q2
Q3
p
T3
Z
Cl
Q0Q1Q2Q3
Q0
Q2
Q2
Q1
Z1
Z0
1
Z1
Z0