Nelson J. Ramirez

2021-0360

Electrónica Digital
Practica III
ITLA

1- Contadores

1- Implemente dos circuitos donde se pueda visualizar las iniciales de su nombre y apellido (usando leds o display) y en el que se pueda observar una secuencia de encendido y apagado basado en los contadores Jhonson y Anular (use 5 flip flops).

Anexe diagrama logico, diagrama de tiempo de ambos, fotos de la implementacion en multisim funcionando (3 de cada circuito) y breve explicaicon del diagrama de tiempo y del funcionamiento de ambos circuitos. Ej: ODJHC

Contador Jhonson

A	B	C	D	E		A	B	C	D	E	F	G	
0	0	0	0	0	N	0	0	1	0	1	0	1	0
1	0	0	0	0	J	0	1	1	1	1	0	0	16
1	1	0	0	0	R	1	1	1	1	1	1	1	24
1	1	1	0	0	A	1	1	1	0	1	1	1	28
1	1	1	1	0	M	1	1	1	0	1	1	0	30
1	1	1	1	1	I	1	0	0	1	1	0	0	31
0	1	1	1	1	R	0	0	0	0	1	0	1	15
0	0	1	1	1	E	1	0	0	1	1	1	1	7
0	0	0	1	1	Z	1	0	0	1	0	0	1	3
0	0	0	0	1	M	1	1	1	0	1	1	0	1

Entradas:

A	B	C	D	E
0	0	0	0	0
1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1

Lo que debe salir:

N

J

R

A

M

I

R

E

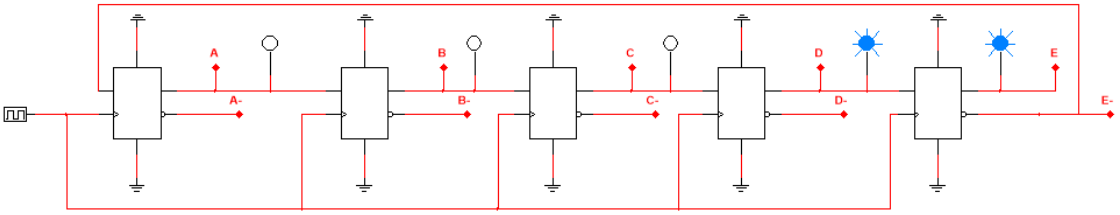
Z

M

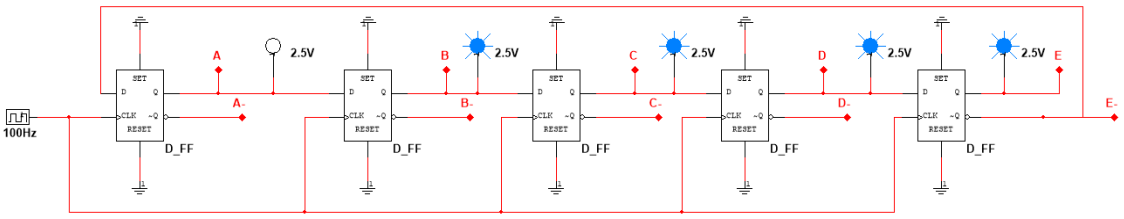
Display:

A	B	C	D	E	F	G	
0	0	1	0	1	0	1	0
0	1	1	1	1	0	0	16
1	1	1	1	1	1	1	24
1	1	1	0	1	1	1	28
1	1	1	0	1	1	0	30
1	0	0	1	1	0	0	31
0	0	0	0	1	0	1	15
1	0	0	1	1	1	1	7
1	0	0	1	0	0	1	3
1	1	1	0	1	1	0	1

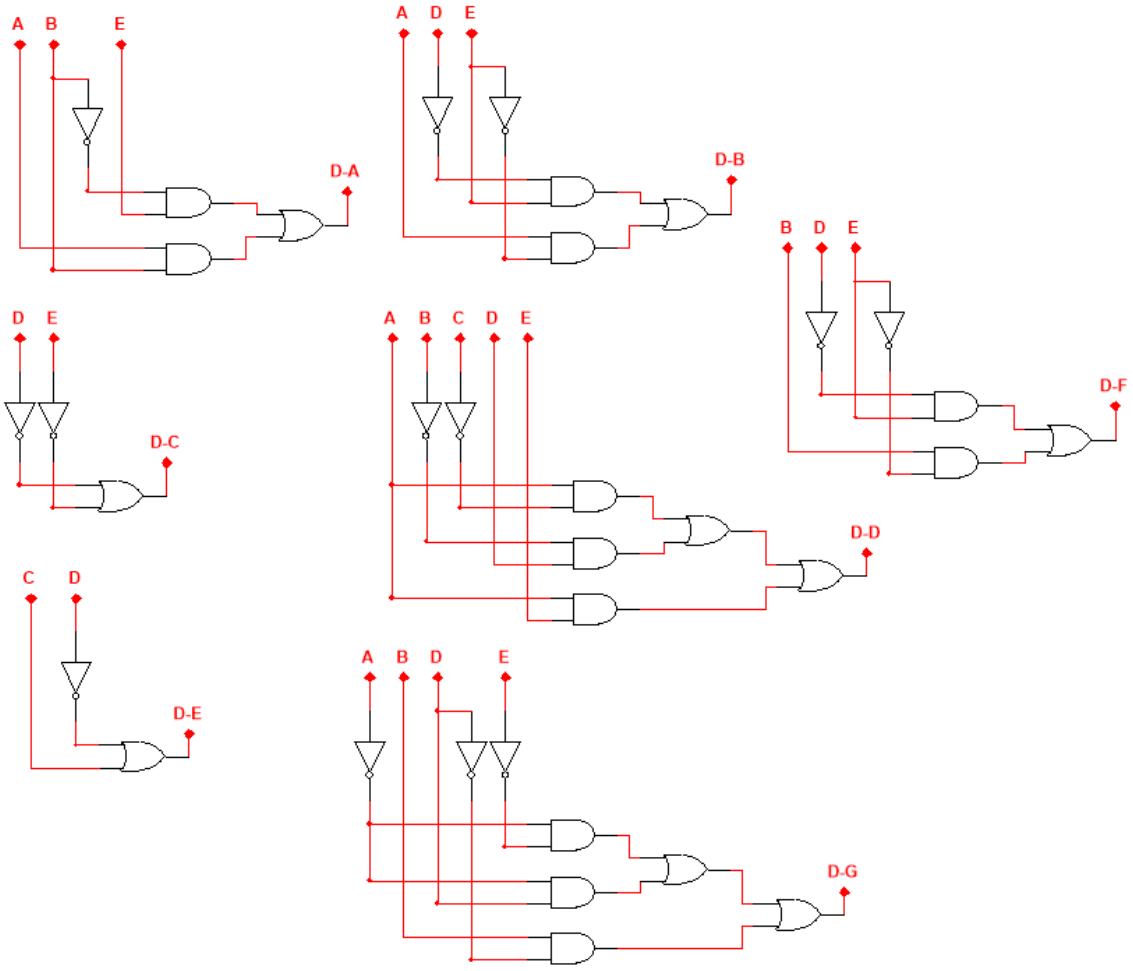
Circuito del contador:

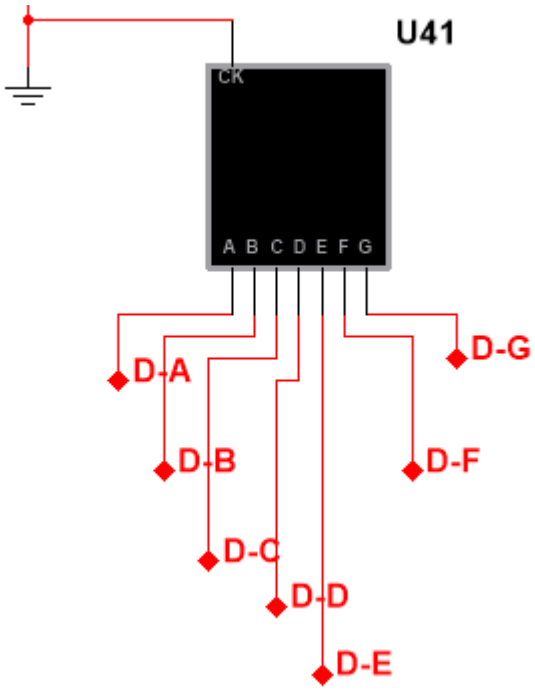


Con los datos de los contadores:

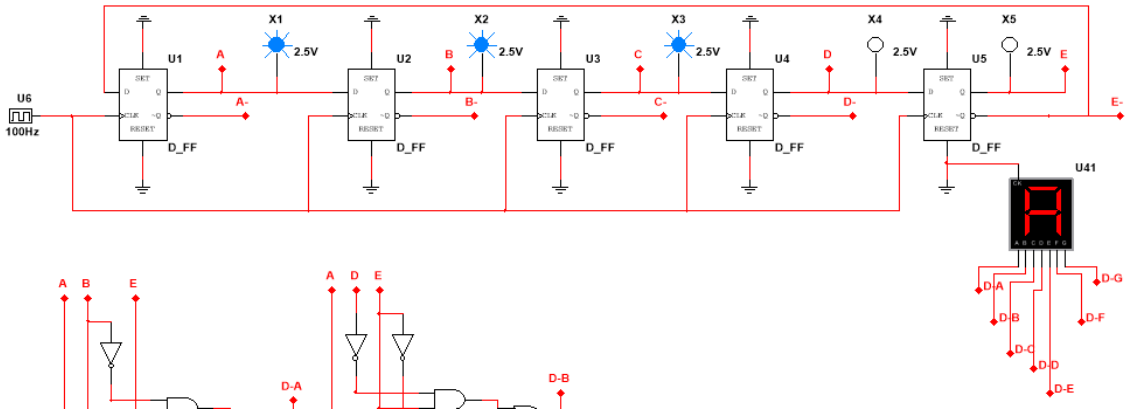


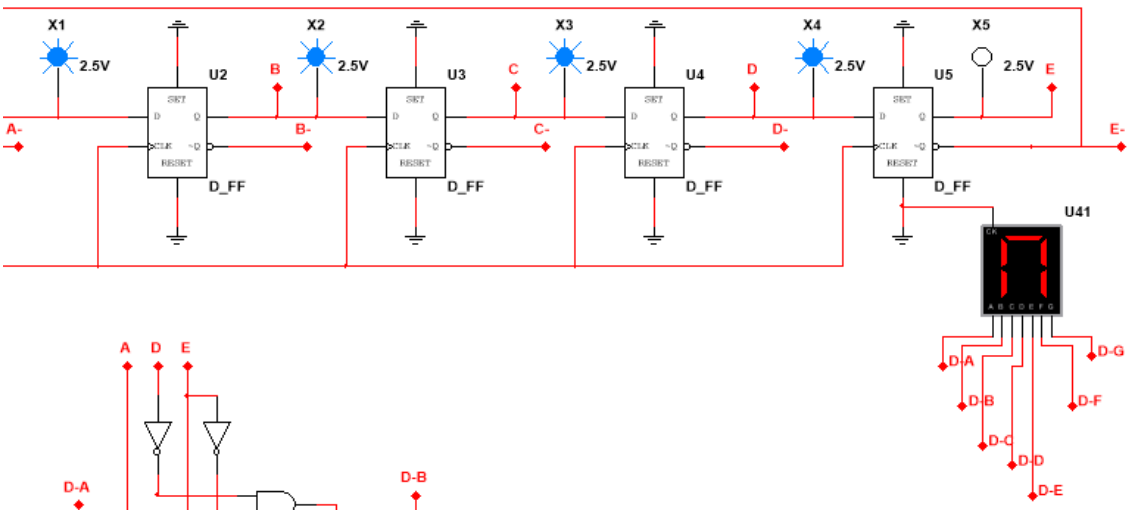
DISPLAY FUNCIONES:

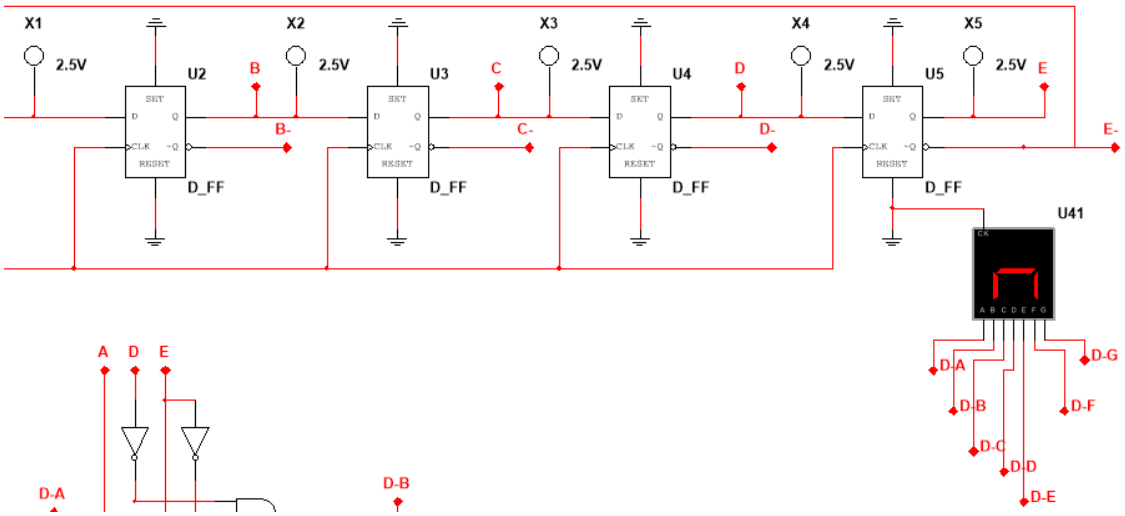


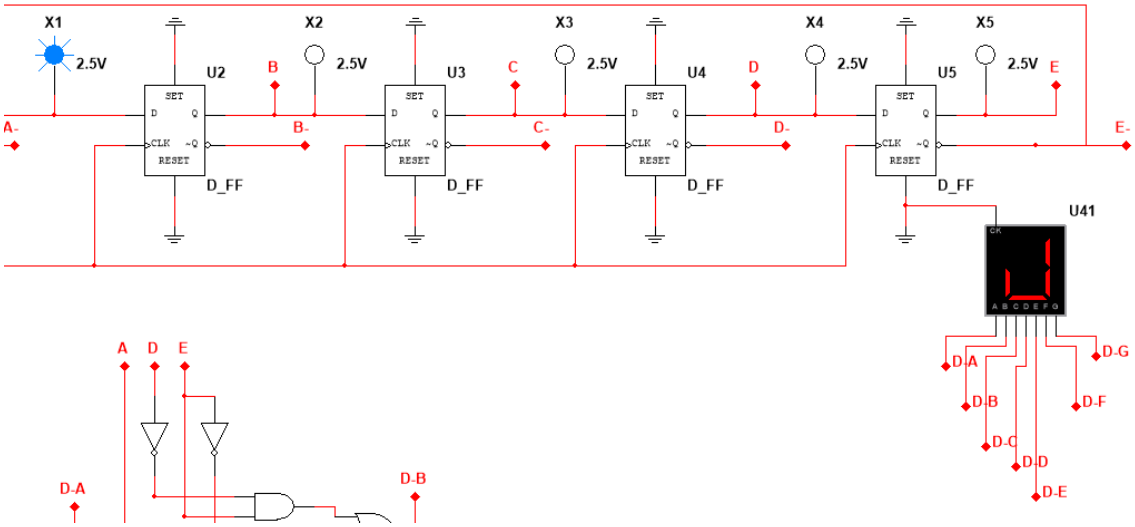


FUNCIONANDO CIRCUITO:

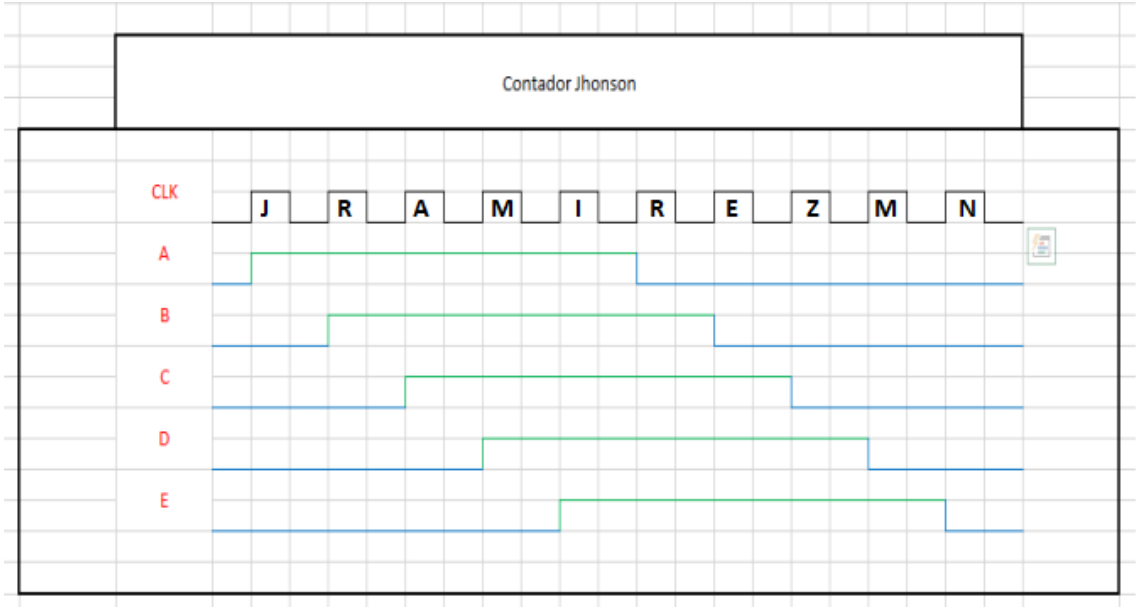





X1
2.5V
U2
B
X2
2.5V
U3
C
X3
2.5V
U4
D
X4
2.5V
U5
X5
2.5V
E
B-
C-
D-
E-
D_FF
U41
A D E
D-A
D-B
D-G
D-A
D-B
D-F
D-C
D-D
D-E


X1
2.5V
U2
B
X2
2.5V
U3
C
X3
2.5V
U4
D
X4
2.5V
U5
X5
2.5V
E
A-
B-
C-
D-
E-
D_FF
U41
A D E
D-A
D-B
D-G
D-A
D-B
D-F
D-C
D-D
D-E

DIAGRAMA


Contador Jhonson
CLK
J R A M I R E Z M N
A
B
C
D
E

CONTADOR ANULAR:

A	B	C	D	E		A	B	C	D	E	F	G	
1	0	0	0	0	N	0	0	1	0	1	0	1	16
0	1	0	0	0	J	0	1	1	1	1	0	0	8
0	0	1	0	0	R	0	0	0	0	1	0	1	4
0	0	0	1	0	M	1	1	1	0	1	1	0	2
0	0	0	0	1	2	1	1	0	1	1	0	1	1

ENTRADAS:

A	B	C	D	E
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

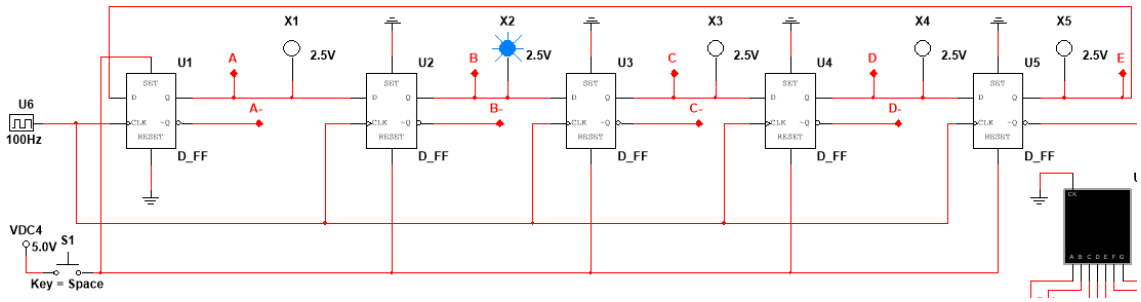
LO QUE DEBE SALIR:

N
J
R
M
2

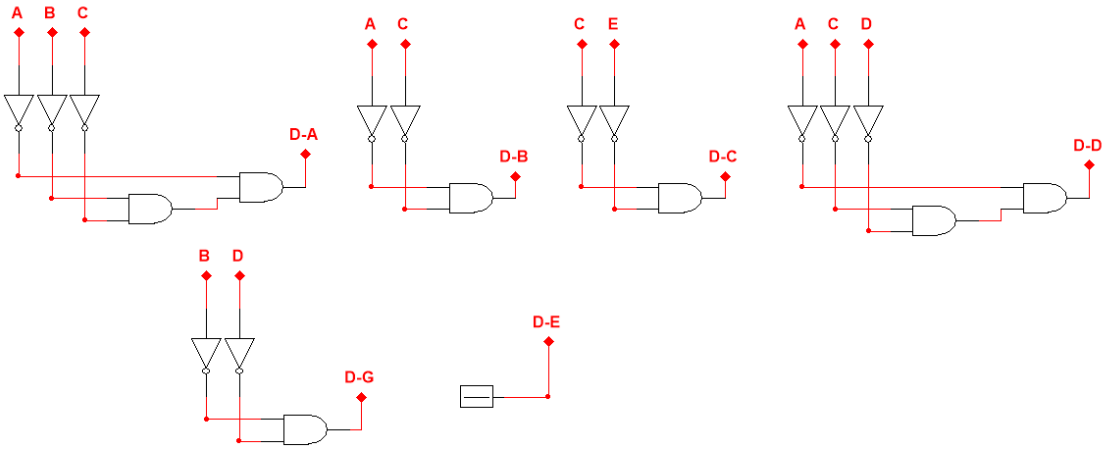
DISPLAY:

A	B	C	D	E	F	G	
0	0	1	0	1	0	1	16
0	1	1	1	1	0	0	8
0	0	0	0	1	0	1	4
1	1	1	0	1	1	0	2
1	1	0	1	1	0	1	1

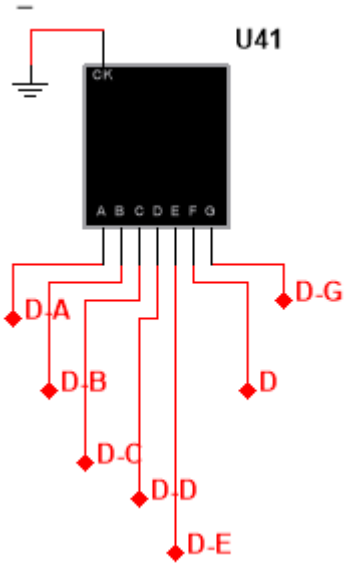
CIRCUITO CONTADOR ANULAR:



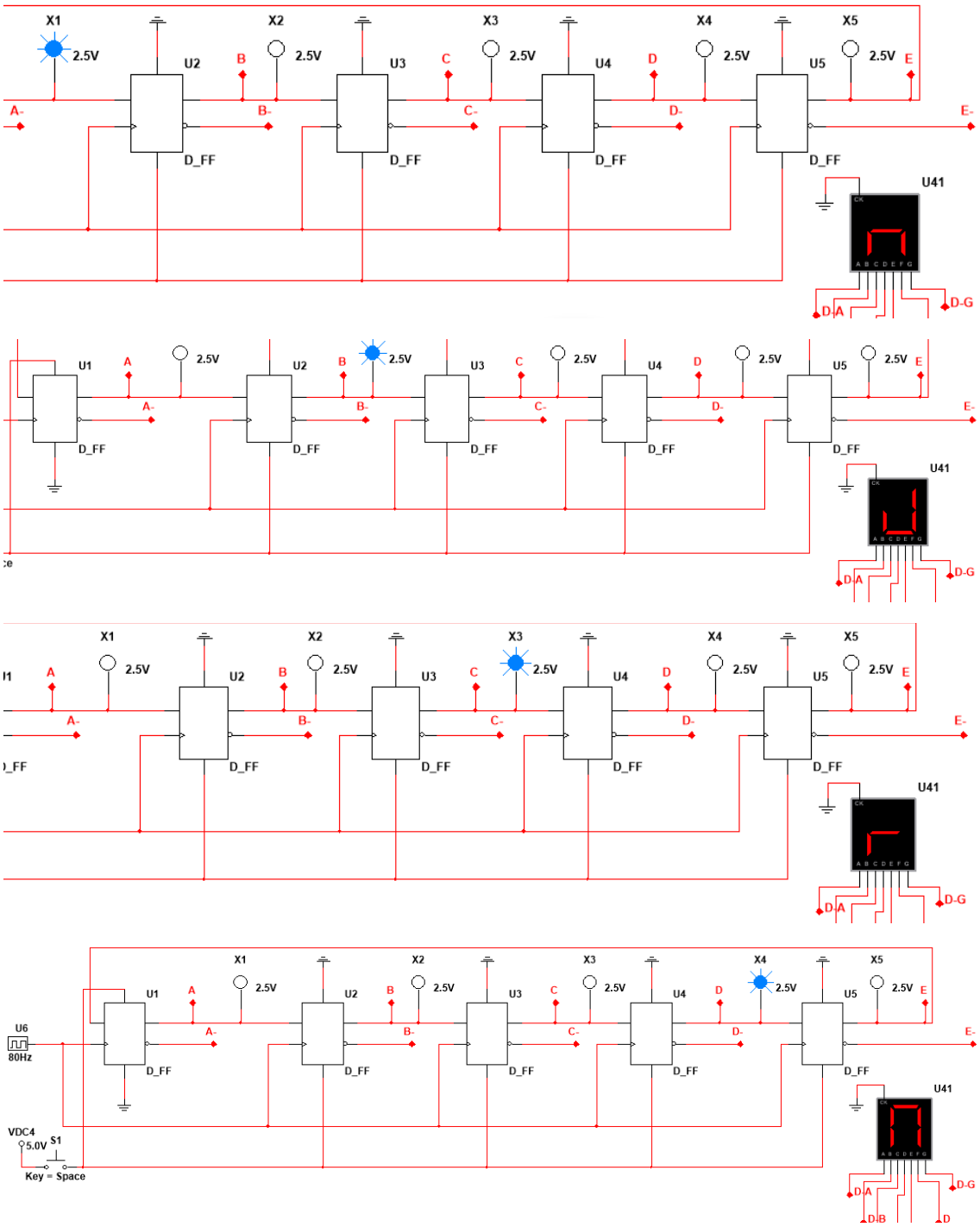
FUNCIONES DISPLAY



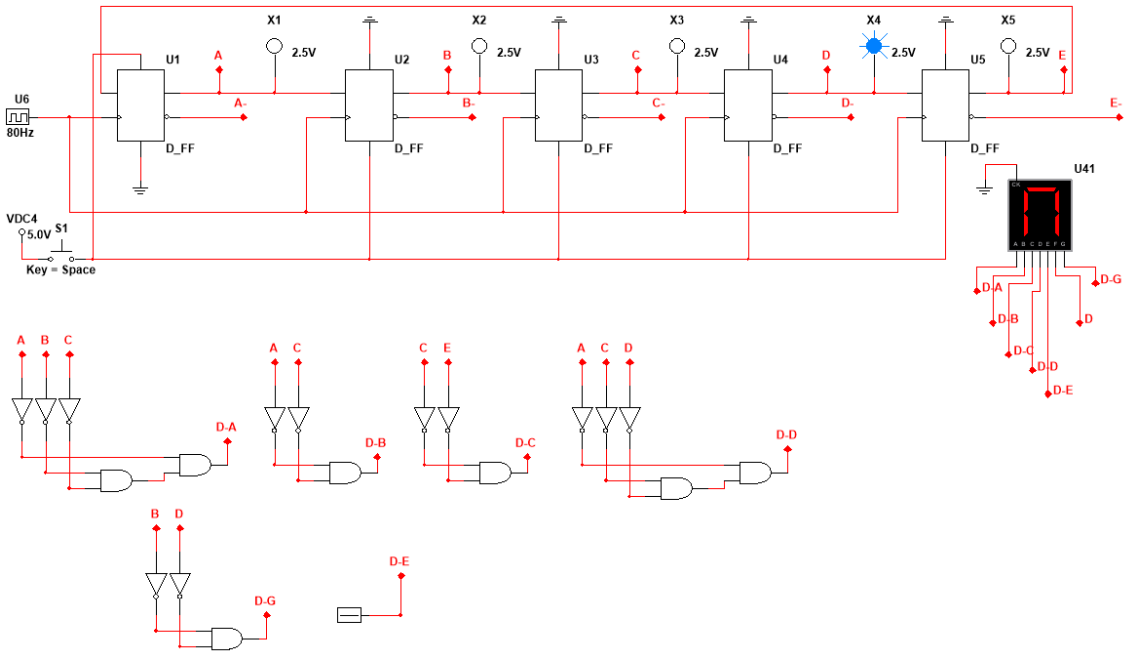
DISPLAY:

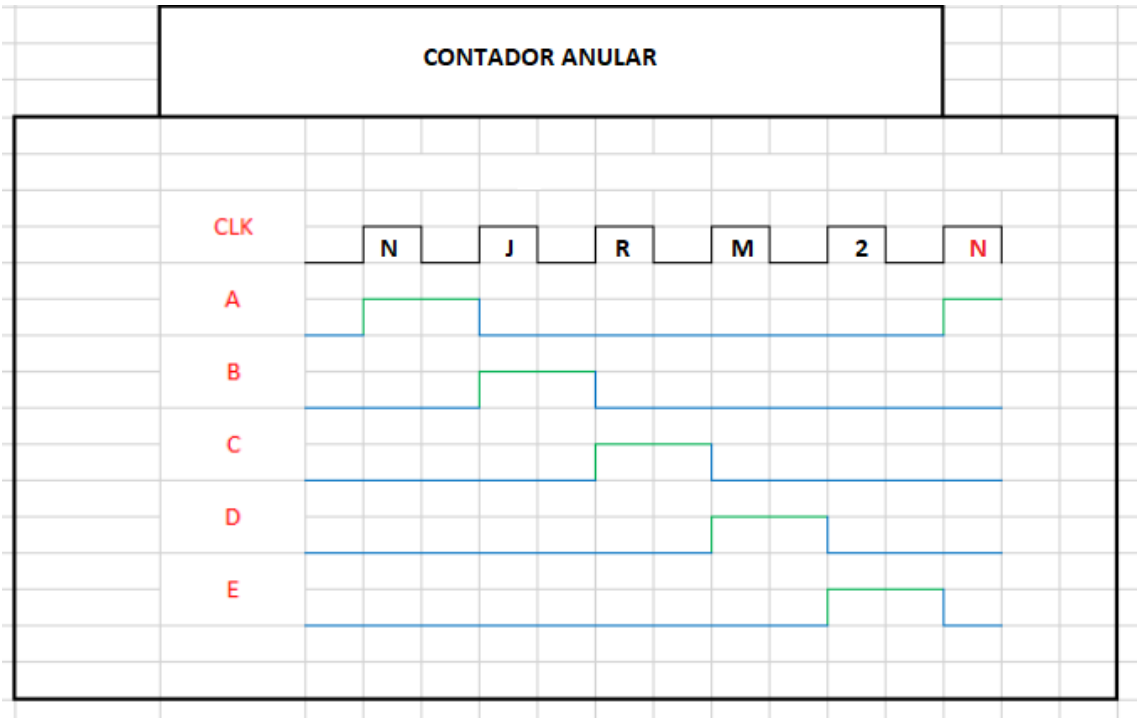


CIRCUITO FUNCIONANDO:



CIRCUITO COMPLETO:





2- Memorias

2- Realizar los siguiente ejercicios extraidos del libro de principios de diseño de sistemas digitales, Guillermo B. y Pablo F. Capitulo 5

Ejercicio 1: Explica la diferencia entre *dirección* de memoria y *posición* de memoria.

Ejercicio 2: La capacidad de las siguientes unidades de memoria se especifica por el número de palabras multiplicado por el número de bits por palabra. ¿Cuántas líneas de direccion y líneas de entrada-salida de datos se necesitan en cada caso? ¿Cuál es el número de bytes de capacidad de cada una?

1. $4K \times 8$
2. $2Gx \times 32$
3. $16M \times 16$
4. $256K \times 64$

Ejercicio 3: En la memoria SRAM de 4×4 bits de capacidad de la Figura 1.7 se usa un decodificador binario natural de 2 entradas y cuatro puertas OR de cuatro entradas cada una. Si queremos expandir la capacidad de esa memoria hasta 256×8 bits, ¿cómo serían el decodificador y las puertas OR necesarias?

Ejercicio 1:

Una dirección de memoria identifica un espacio concreto de la memoria. En una dirección de memoria se puede guardar una serie de bits (usualmente, 8 bits).

Por hacer un símil sencillo ejemplo, imaginemos un hotel que tiene 10 habitaciones numeradas del 0 al 9. En recepción tienen un armarito con 10 huecos para guardar las llaves. Cada uno de esos huecos está identificado con un número, por lo que para buscar la llave de una habitación concreta, solo tienen que buscar ese número y coger la llave del cajetín.

Mientras que la posición de memoria es la función que permite recordar el lugar ocupado por un objeto en una lista ordenada, sin tener que memorizar la lista.

Ejercicio 2:

Ejercicio 2:

1. $4k * 8 = \ 4 * 2^{10} * 8 \rightarrow 4 * 1024 * 8 = 32\,768\ bit$

Capacidad: $\frac{32738}{8} \rightarrow 4\,096\ bytes = 4k\ bytes$

Lineas de dirección: $4K \rightarrow 4 * 2^{10} = 2^2 * 2^{10} = 2^{12} \rightarrow 12\ bits\ de\ dirección$

Lineas de entrada y salida: $Long\ 8\ bits = 8\ bits\ de\ entrada\ y\ salida.$

2. $2G * 32 = 2 * 2^{30} * 32 \rightarrow 2 * 1\,073\,741\,824 * 32 = \ 68\,719\,476\,736$

Capacidad: $\frac{68,719,476,736}{8} = 8589934592\ bytes \rightarrow 8G\ bytes$

Lineas de dirección: $2G = 2 * 2^{30} = 2^{31} \rightarrow 31\ bits\ direccion$

Lineas de entrada y salida: $Long.\ de\ pallabra\ 32\ bit \rightarrow 32\ bits\ de\ entrada - salida.$

3. $16M * 16 = 16 * 2^{20} * 16 \rightarrow 16 * 1{,}048{,}576 * 16 = 268{,}435{,}456$

Capacidad: $\frac{268,435,456}{8} = 33{,}554{,}432\ bytes = 32\ M\ bytes$

Lineas de dirección: $16\ M = 16 * 2^{20} \rightarrow 2^4 * 2^{20} = 2^{24} \rightarrow 24\ bits\ de\ dirección$

Lineas de entrada y salida: $Long.\ de\ palabra\ 16\ bits = 169\ bits\ de\ entrada - salida$

4. $256K * 64 = 256 * 2^{10} * 64 = 256 * 1024 * 64 \rightarrow 16{,}777{,}216\ bits$

Capacidad: $\frac{16,777,216}{8} = 2{,}097{,}152\ bytes \rightarrow 2G\ bytes$

Lineas de dirección: $256\ K \rightarrow 256 * 2^{10} = 2^8 * 2^{30} = 2^{30} \rightarrow 38\ bits\ de\ dirección.$

Lineas de entrada y salida: $Long.\ de\ palabra\ 64\ bits = 64\ bits\ de\ entrada - salida.$

Ejercicio 3:

En la figura del libro se puede observar como las salidas del decodificador corresponden a la señal de una serie de celdas binarias, las correspondientes a una palabra. Por lo tanto, si queremos 256 palabras, se necesitan 256 líneas a la salida del decodificador. Un decodificador binario natural de 256 = 2 8 salidas tienen 8 bits de entrada. Además, como la señal salida de cada celda binaria de una palabra está unida a una puerta OR, si tenemos 8 bits de longitud de palabra, necesitamos 8 puertas OR, cada una con 256 entradas, tantas como palabras tenemos.

3-HDL

3- Realice un comparador de 4 bits usando verilog, use sintaxis assign y operadores lógicos y vuelva a realizarlo en verilog pero en este caso a caso a compuertas. Anexe diagrama lógico y de tiempo del mismo (Test bench). Anexe codigo de verilog, diagrama lógico y de tiempo.

DISEÑO:

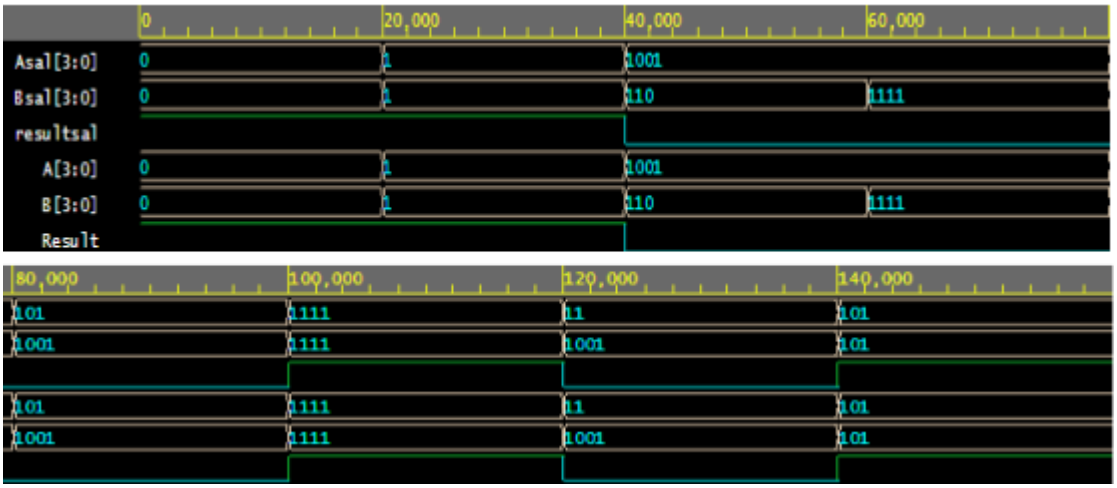
VHDL Design

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity comparator is port (
6    A     ,B: in std_logic_vector (3 downto 0);
7    Result : out std_logic);
8  end comparator;
9
10
11 architecture bool of comparator is
12 begin
13   Result <= not (A(3) xor B(3)) and
14             not (A(2) xor B(2)) and
15             not (A(1) xor B(1)) and
16             not (A(0) xor B(0));
17  end bool;
```

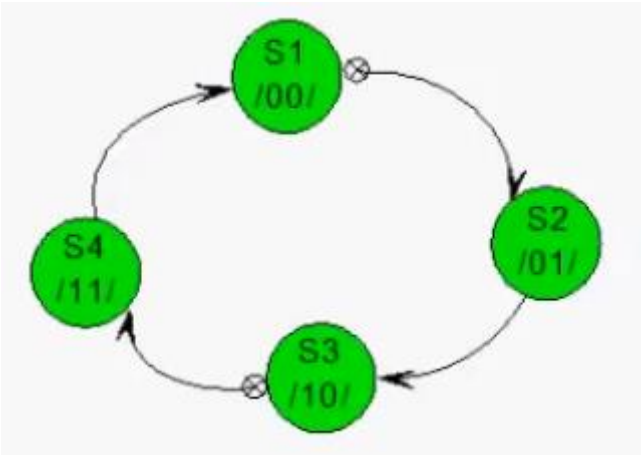
Testbench

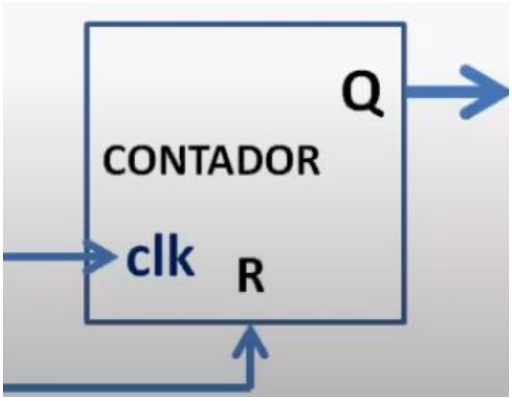
```
library ieee;
use ieee.std_logic_1164.all;

entity simulacion is
end simulacion;

architecture behavorial of simulacion is


Component comparator
port (A,B : in std_logic_vector (3 downto 0);
Result : out std_logic);
end component;

signal Asal,Bsal : std_logic_vector (3 downto 0):="0000";
signal resultsal : std_logic;

begin
U0: comparator Port map(A => Asal, B => Bsal, Result => resultsal);



process begin

Asal <= "0000";
Bsal <= "0000";

wait for 20 ns;

Asal <= "0001";
Bsal <= "0001";

wait for 20 ns;

Asal <= "1001";
Bsal <= "0110";

wait for 20 ns;

Asal <= "1001";
Bsal <= "1111";

wait for 20 ns;


Asal <= "0101";
Bsal <= "1001";

wait for 20 ns;
Asal <= "1111";
Bsal <= "1111";
Asal <= "1111";
Bsal <= "1111";

wait for 20 ns;

Asal <= "0011";
Bsal <= "1001";

wait for 20 ns;

Asal <= "0101";
Bsal <= "0101";

wait for 20 ns;

wait;
end process;

end behavorial;
```

DIAGRAMA:



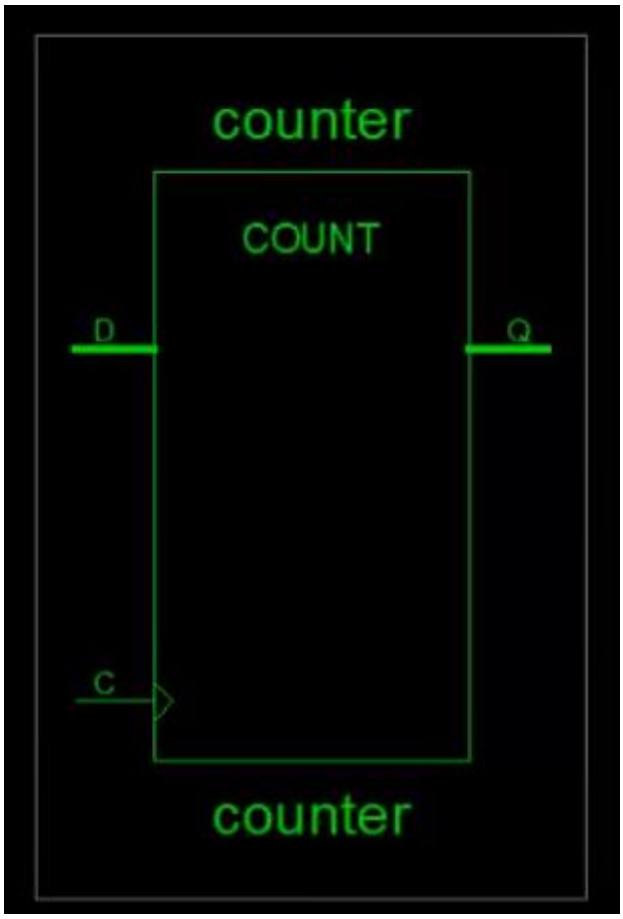
4- Realice un contador del 1-15 usando verilog).Anexe codigo de verilog, diagrama lógico y de tiempo

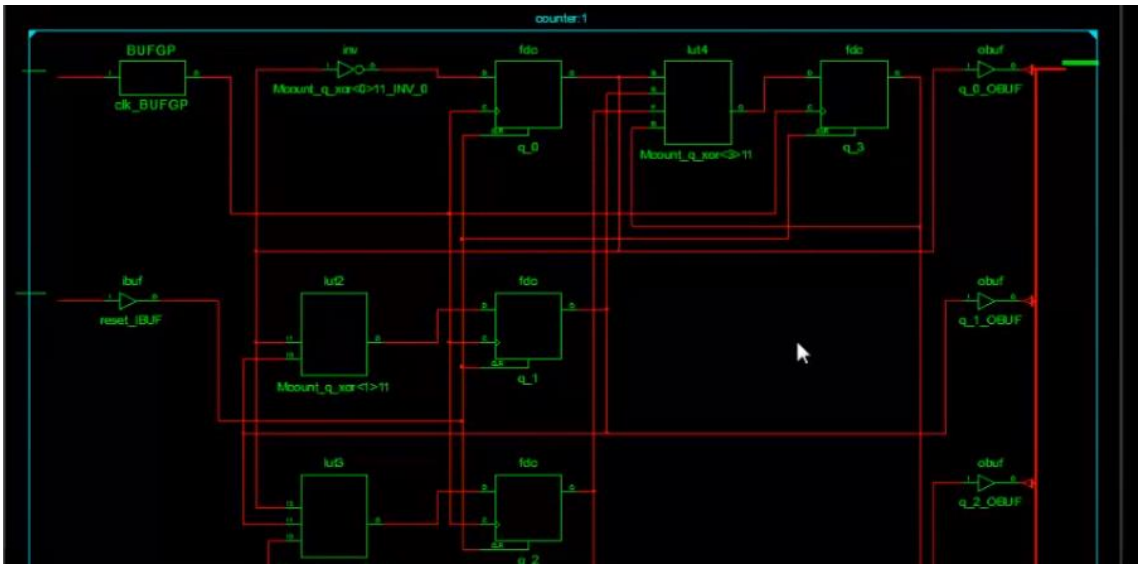


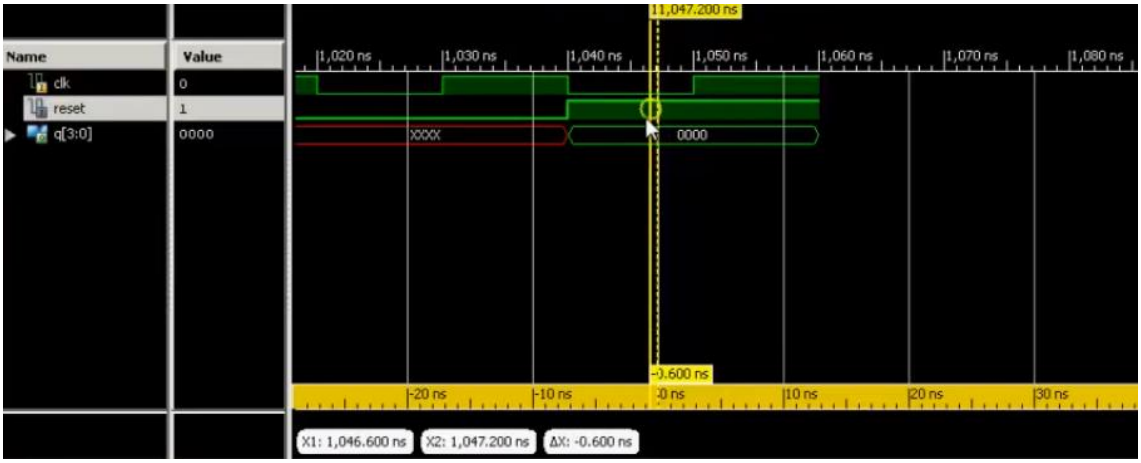


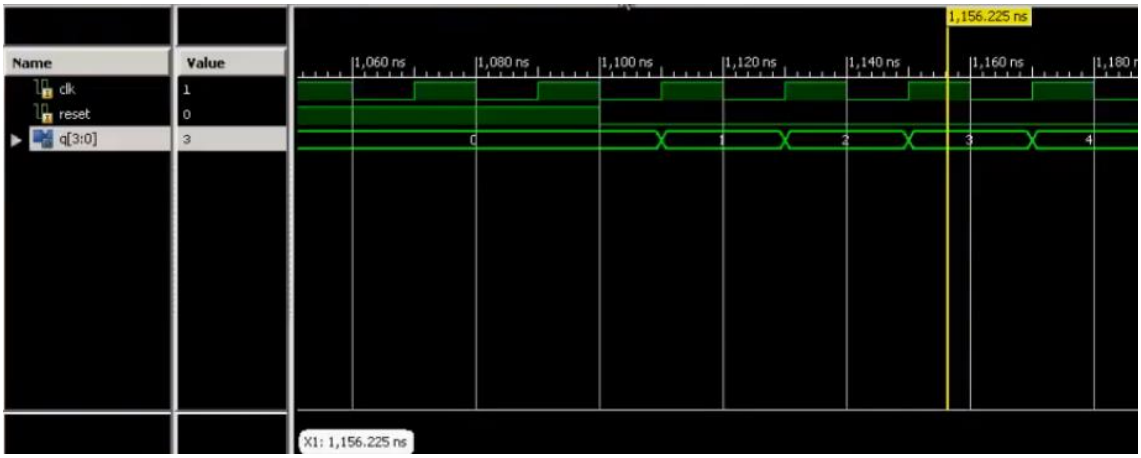
```
1  module counter(clk, reset, q);
2  input clk, reset;
3  output reg [3:0] q;
4
5     always@(posedge reset, posedge clk)
6        if(reset)
7        q<= 0;
8        else
9        q<= q+ 1;
10 endmodule
```

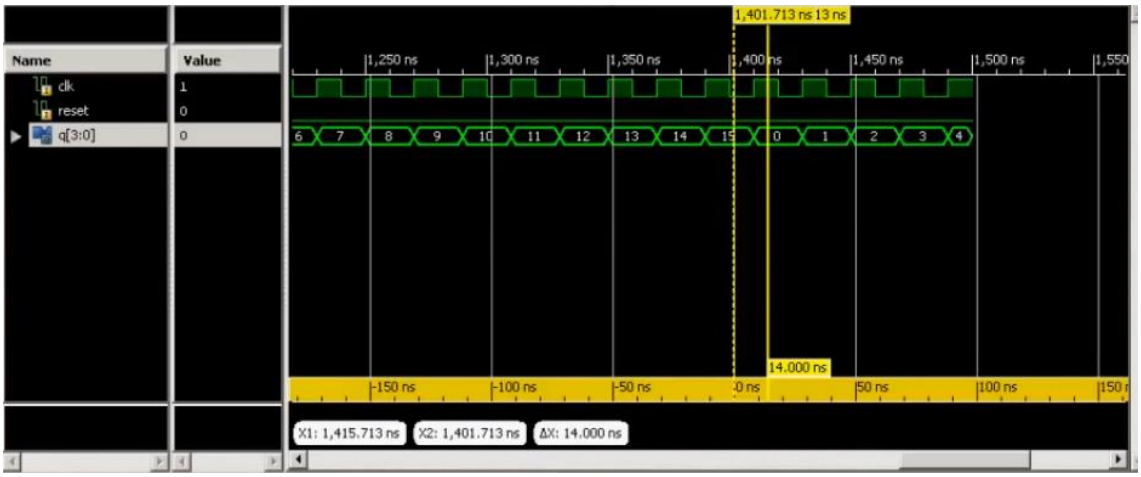
RTL




counter:1
BUFGP
clk_BUFGP
inv
Mcount_q_xor<0>11_INV_0
fdc
q_0
lut4
Mcount_q_xor<3>11
fdc
q_3
obuf
q_0_OBUF
ibuf
reset_IBUF
lut2
Mcount_q_xor<1>11
fdc
q_1
obuf
q_1_OBUF
lut3
fdc
obuf
q_2_OBUF


Name
Value
clk
0
reset
1
q[3:0]
0000
11,047.200 ns
1,020 ns
1,030 ns
1,040 ns
1,050 ns
1,060 ns
1,070 ns
1,080 ns
XXXX
0000
-0.600 ns
-20 ns
-10 ns
0 ns
10 ns
20 ns
30 ns
X1: 1,046.600 ns
X2: 1,047.200 ns
ΔX: -0.600 ns


Name
Value
clk
1
reset
0
q[3:0]
3
1,156.225 ns
1,060 ns
1,080 ns
1,100 ns
1,120 ns
1,140 ns
1,160 ns
0
1
2
3
4
X1: 1,156.225 ns

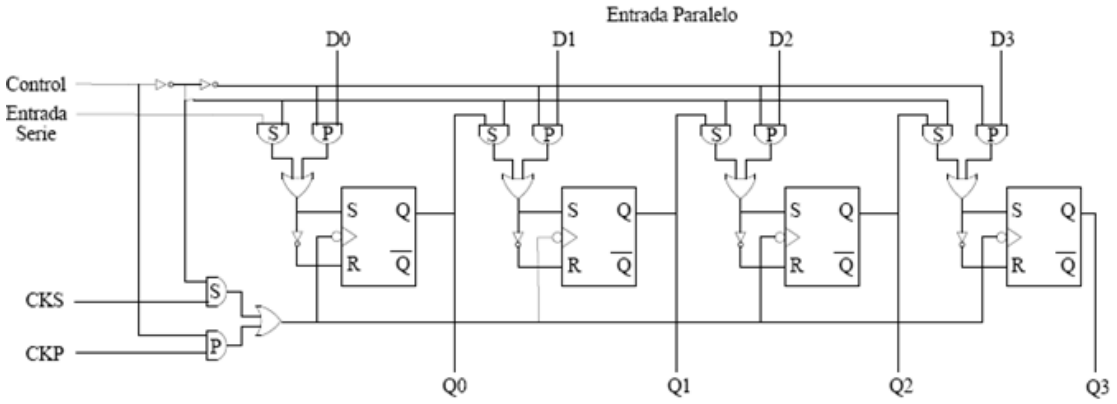


4- Registros

5- Explique el funcionamiento de un registro Universal anexe diagrama y defina los diferentes tipos de registros que podemos encontrar.

Un registro es universal cuando es bidireccional, tanto las entradas como las salidas de los datos pueden ser en paralelo y en serie, y tiene capacidad para inhibir su funcionamiento.

A continuación, mostramos el esquema de un registro universal de desplazamiento:



Funcionamiento del registro universal:

- **Funcionamiento como registro serie-serie**: para ello la entrada "Control" debe ser igual a 0, además de la puerta lógica "AND-P" debe estar anulada. La señal de reloj "CKS" debe ser activa y la entrada de datos se debe realizar por la línea "Entrada Serie". La salida serie se obtendría por "Q3" en siete impulsos de reloj.
- **Funcionamiento como registro serie-paralelo:** para ello la entrada "Control" debe ser igual a 0, además de la puerta lógica "AND-P" debe estar anulada. La señal de reloj CKS debe ser activa y la entrada de datos se debe realizar por la línea "Entrada Serie". La salida paralelo se obtendría por "Q0", "Q1", "Q2"y "Q3", en cuatro pulsos de reloj.
- **Funcionamiento como registro paralelo-serie:** lo primero que debemos hacer es cargar los datos presentes en las entradas "D0-D4" de los biestables, para ello en la línea de "Control" debe ser igual a 1 y activamos un pulso de reloj "CKP". Para

desplazar los bits a lo largo de los biestables y en consecuencia la salida de dichos bits por la salida "Q3", pondremos "Control" = 0 y activaremos el reloj "CKS".

- **Funcionamiento como registro paralelo-paralelo:** la entrada de datos se realiza por "D0−D3", en cada pulso de reloj "CKP" se transfiere el dato a la salida. La salida de los datos se produce en cada pulso de reloj "CKP" por "Q0-Q3".

Tipos de registros

Los registros del procesador se dividen o clasifican atendiendo al propósito que sirven o a las instrucciones que les ordenan.

Registros de datos

Guardan valores de **datos numéricos,** como son los **caracteres** o **pequeñas órdenes.** Los procesadores antiguos tenían un registro especial de datos: el **acumulador,** el cual era usado para operaciones determinadas.

Registro de datos de memoria (*MDR*)

Es al que hacíamos referencia antes, se trata de un registro que se encuentra en el procesador y que está conectado al **bus de datos.** Tiene poca capacidad y una velocidad alta por la que escribe o lee los datos del bus que van dirigidos a la **memoria** o al **puerto E/S**, es decir, un periférico.

Registros de direcciones

Guardan direcciones que son usadas para acceder a la **memoria principal** o **primaria,** que solemos conocer como **ROM** o **RAM.** En este sentido, podemos ver procesadores con registros que se usan solo para guardar direcciones o valores numéricos.

Registros de propósito general (*GPRs*)

Son registros que sirven para **almacenar direcciones** o **datos generales.** Se trata de una especie de **registros mixtos** que, como su propio indica, no tienen una función específica.

Registros de propósito específico (*SPRs*)

En esta ocasión, estamos ante registros que guardan datos del estado del sistema, como puede ser el registro de estado o el ***instruction pointer***. Pueden estar combinados con el **PSW** (*Program Status Word*).

Registros de estado

Sirven para guardar valores reales cuya función es determinar cuándo una instrucción debe ejecutarse o no. También se le conoce como **CCR** (*Condition Code Register)*. Dentro de este tipo de registros, encontramos el siguiente:

- Registro de bandera o «***FLAGS***«. Lo encontramos en los procesadores **Intel** con arquitectura **X86.** Estamos ante un registro con **16 bits** de ancho. Pero, tiene 2 sucesores:
  - **EFLAGS,** con **32 bits** de ancho.
  - **RFLAGS,** con **64** bits de ancho.

Registros de coma flotante

Primero, convendría explicar qué es una **coma flotante.** La coma flotante es una representación, en forma de fórmula, de números reales de distintos tamaños que sirve para realizar operaciones aritméticas. Nos encontraremos con ella en sistemas que requieren sistemas de procesados muy rápidos.

5- ADC

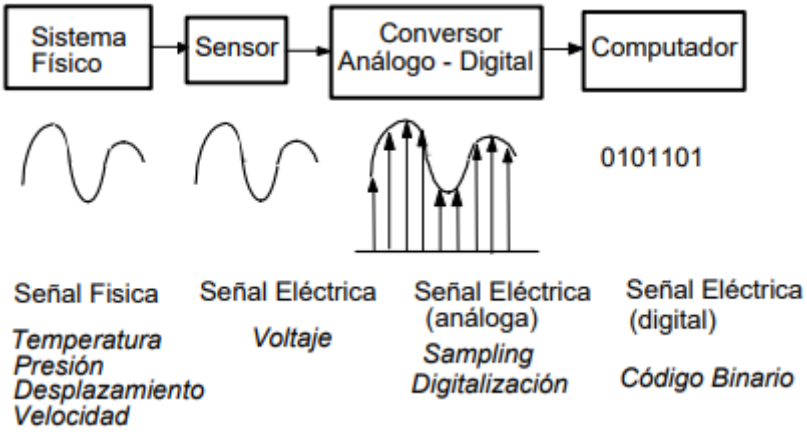
6- Defina que es un ADC (Analog Digital converter) anexe explicacion minimo 3 parrafo, anexe fotos y diagramas y 10 aplicaciones donde se utilizan.

Anexe info sobre un IC que sea un ADC.

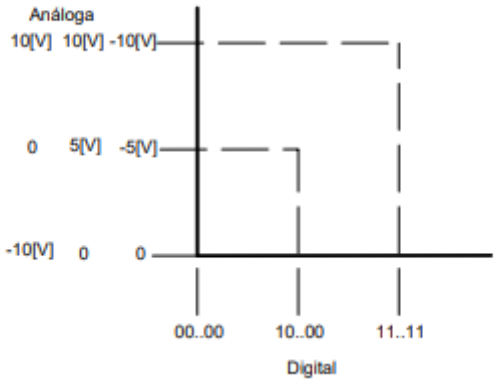
El desarrollo de los microprocesadores y procesadores digitales de señal (DSP), ha permitido realizar tareas que durante años fueron hechas por sistemas electrónicos analógicos. Por otro lado, como que el mundo real es análogo, una forma de enlazar las variables analógicas con los procesos digitales es a través de los sistemas llamados conversores de analógico - digital (ADC- Analogue to Digital Converter) y conversores digital - analógico (DAC- Digital to Analogue Converter). El objetivo básico de un ADC es transformar una señal eléctrica análoga en un número digital equivalente. De la

misma forma, un DAC transforma un número digital en una señal eléctrica análoga.

Esta función exige que los pasos intermedios se realicen de forma optima para no perder información. Según el tipo de componente y su aplicación existen distintos parámetros que lo caracterizan, éstos pueden ser: la velocidad de conversión, la resolución, los rangos de entrada, etc.. Por ejemplo, una mayor cantidad de bit, implica mayor precisión, pero también mayor complejidad. Un incremento en un solo bit permite disponer del doble de precisión (mayor resolución), pero hace más difícil el diseño del circuito, además, la conversión podría volverse más lenta. Dentro de las de aplicaciones de estos sistemas está el manejo de señales de vídeo, audio, los discos compactos, instrumentación y control industrial. En los siguientes apartados se describen los conceptos básicos de conversión de señal, técnicas de implementación para los ADC o DAC, características y parámetros que los definen. Se revisarán las configuraciones más clásicas, atendiendo a criterios de velocidad y manejo de datos, como también los nuevos productos disponibles en el mercado.
Un transductor permite relacionar las señales del mundo real y sus análogas eléctricas. Para compatibilizar la información con un sistemas digital, se requiere de convertidores de datos del tipo ADC o DAC, segun corresponda. El diagrama de bloques de la Fig.1 muestra la secuencia desde que la variable física entra al sistema hasta que es transformada a señal digital (código binario). Para dicha señal ingrese al convertidor análogo - digital, ésta debe ser muestreada, es decir, se toman valores discretos en instantes de tiempo de la señal análoga, lo que recibe el nombre de sampling. Matemáticamente es el equivalente a multiplicar la señal análoga por una secuencia de impulsos de periodo constante. Como resultado se obtiene un tren de impulsos con amplitudes limitadas por la envolvente de la señal analógica



Cada conversor ADC ó DAC, esta determinado por una función de transferencia ideal de *entrada - salida* (ver Fig. 4), que muestra la equivalencia entre el mundo digital y el análogo.



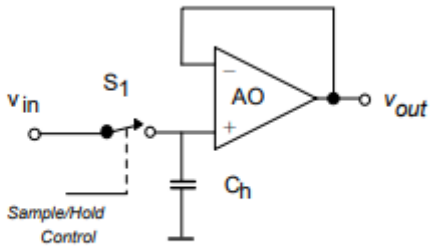


Fig. 9. Circuito Sample and Hold.

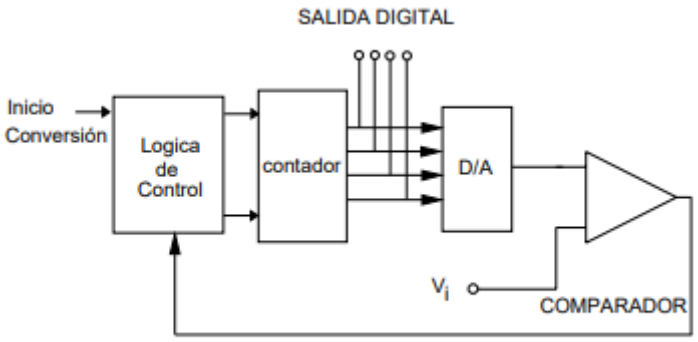


Fig. 13. Conversor de rampa escalera.

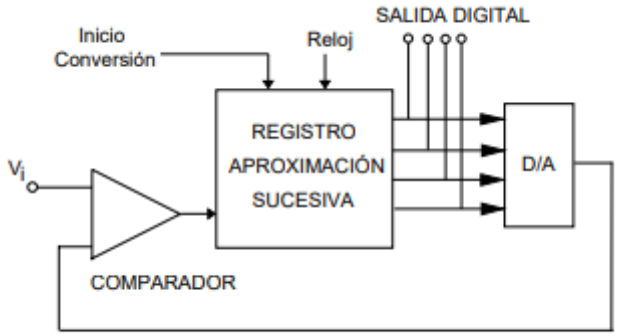


Fig. 14. Conversor de aproximación sucesiva.

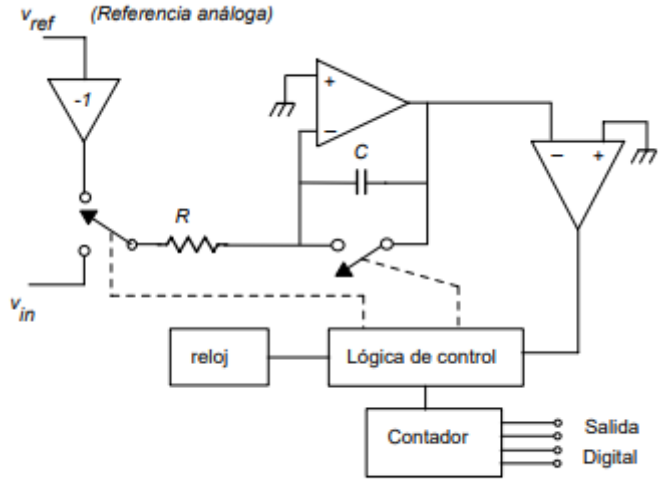


Fig. 16. Convesor doble rampa.

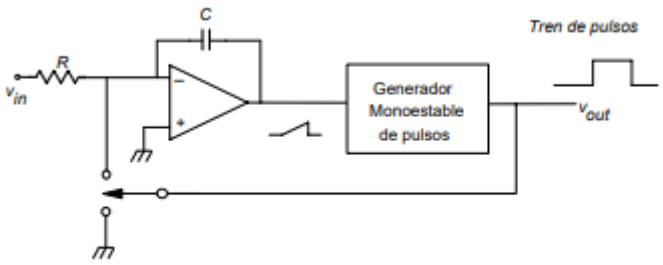


Fig. 17. Convesor Voltaje - Frecuencia

Algunas de las aplicaciones importantes de un ADC son:

-Grabadora de audio.

-Tarjetas sintonizadoras de televisión.

-Radares

-Arduino

-Osciloscopio Digital

-FPGA

-Detector de Humo.

-Puertas de seguridad por sensor de movimiento.

-Sistema de alarma de carro.