UNIVERSIDAD DE ANTIOQUIA

LAB1: Diseño de sistemas digitales

Jonatan Stiven Restrepo Lora

Otro Nombre

Universidad de Antioquia

April 8, 2024

1 Introducción

Se desea crear un circuito que permita hacer la suma de 16 números guardados en un vector, siguiendo el siguiente algoritmo:

$$\sum_{n=0}^{16}(A_n + A_{n+1})$$

Entre los componentes que se deben utilizar para desarrollar la solución están los Flip-Flops tipo ***JK***

2 Almacenamiento

2.1 Creación del Vector

Se creó una unidad de almacenamiento el cual pudiera contener un número de 8 bits y a la vez permitirá almacenar un número predeterminado, hasta que llegara el nuevo número por el que se reemplazaría.

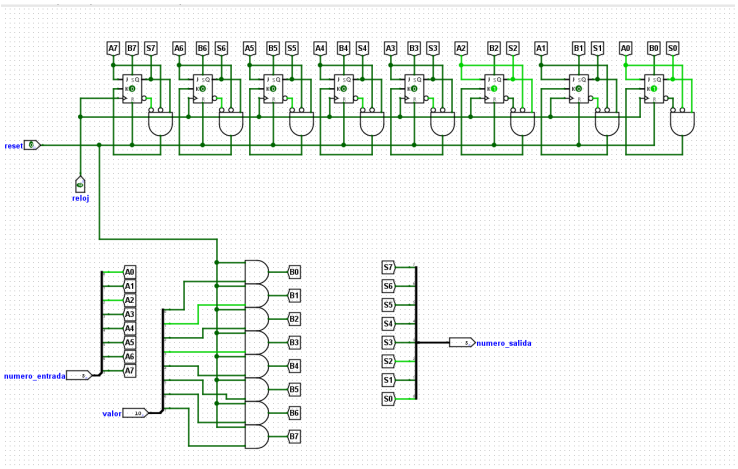

Figure 1: Unidad de almacenamiento

Así, de esta manera se podría crear de forma sencilla el vector de 16 posiciones y para el cual se permitiría almacenar números predeterminados. Además, se implementó una lógica para que solo escribiera en los arreglos correctos según la posición solicitada por el algoritmo y diario se retorne los numeros A_n y A_{n+1}

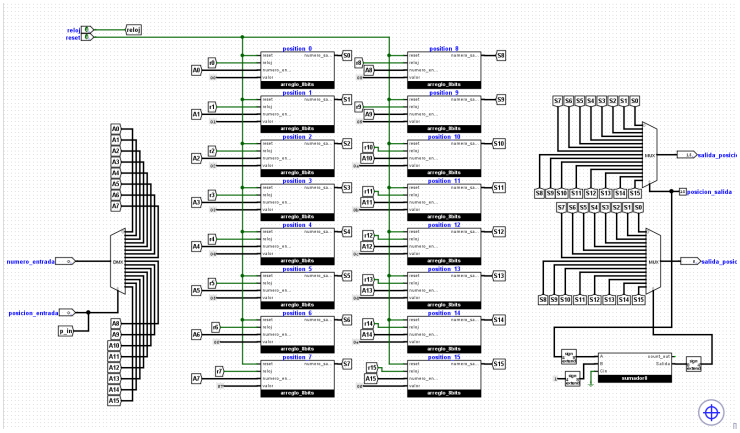

Figure 2: Vector de 16 posiciones

3 Sumador

Se construye primero un sumador de un bit full y con base a este se procede a construir uno para 8 bits que es la cantidad máxima para este ejercicio.

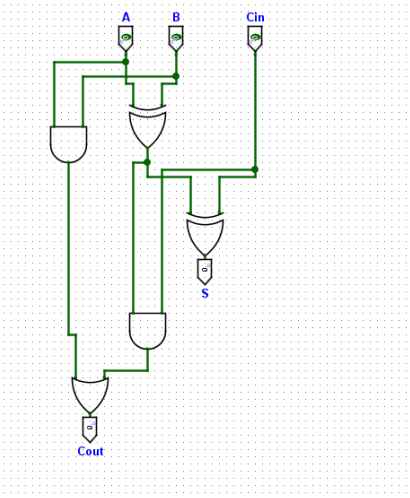

Figure 3: Sumador completo de 1 bit

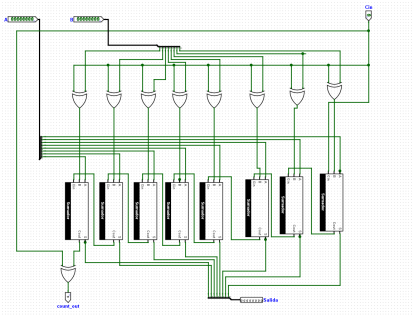

Figure 4: Sumador completo de 8 bits

4 Contadores

4.1 Contador unidad

Para este ejercicio se creó un contador de 4 bits (16 posiciones posibles) con el flip-flop tipo Jk conectados en cascada, el cual ayudaría para saber qué posición del vector guardar los datos, en que fase del algoritmo se encuentra y el estado de la máquina.

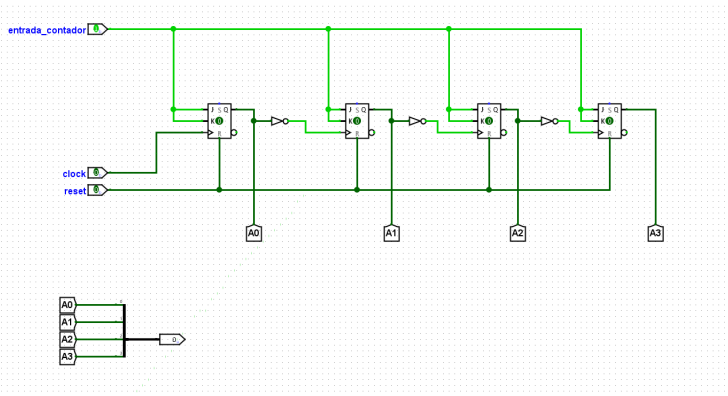

Figure 5: Contador de 4 bits

4.2 Contador par

Se creó un contador par, el cual da las posiciones $\eta = \{0, 2, 4, 6, 8, 10, 12, 14\}$, esto con el fin de seguir con la lógica implementada en el vector donde se retorna los números en las posiciones A_n & A_{n+1}.

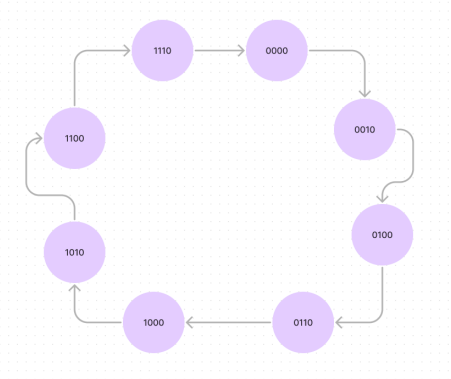

Figure 6: Diagrama contador par

Para llevar a cabo la implementación de dicho contador sé continuo con la creación de la tabla extendida y la simplificación con los mapas K.

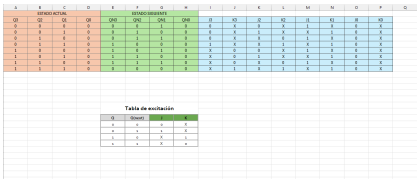

Figure 7: Tabla extendida contador par

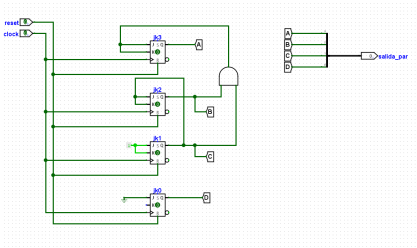

Figure 8: Circuito contador par

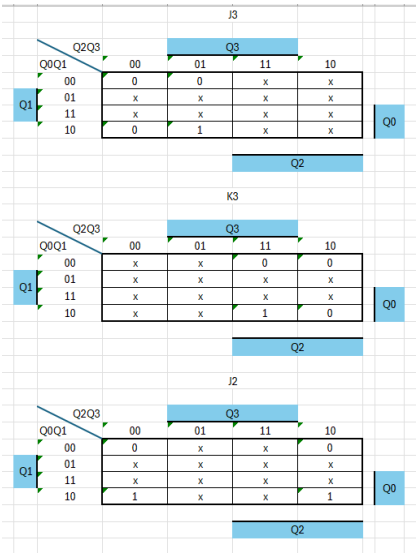

Figure 9: Mapas K contador par

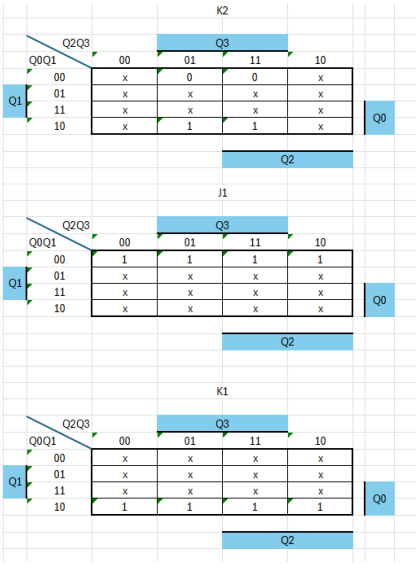

Figure 10: Mapas K contador par

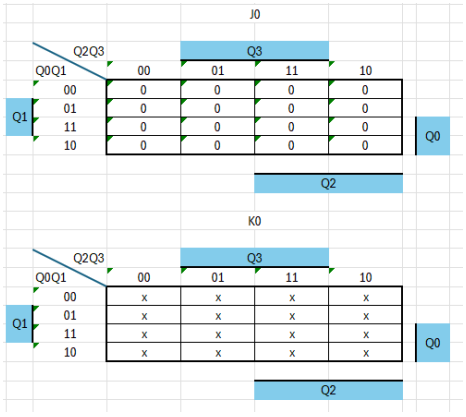

Figure 11: mapas k contador par

Las expresiones simplificadas son las siguientes:

- $J3 = Q2 \cdot Q1$
- $K3 = Q2 \cdot Q1$
- $J2 = Q1$
- $K2 = Q1$
- $J1 = 1$
- $K1 = 1$
- $J0 = 0$
- $K0 = X$

5 Visualizador

Para poder cumplir con las especificaciones de poder ver los números A_n & A_{n+1} en cada ciclo del algoritmo, la suma correspondiente al mismo ciclo, la face del algoritmo y el estado actual de la máquina, fue necesario crear los siguientes componentes:

5.1 Decodificador 7 segmentos

Se creó un decodificador de 7 segmentos el cual recibe 4 bits y representa los valores de 0 al 9.

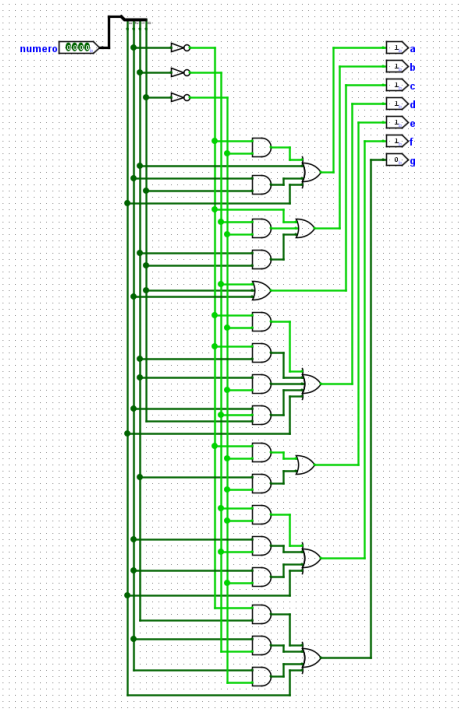

Figure 12: Circuito Decodificador 7 segmentos

5.2 Complemento a 2

Se construyo un circuito que hiciera el complemento a 2, para una entrada de 8 bits. Dicho complemento funciona solo para números negativos.

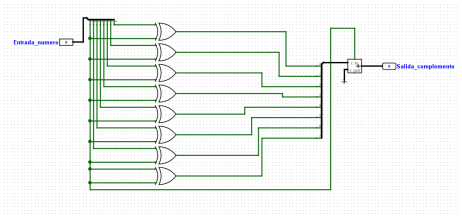

Figure 13: Circuito Complemento a dos

5.3 Construcción Visualizador

Los componentes anteriores fueron necesarios para poder crear un componente que me permitiera obtener las unidades, decenas y centenas de un número, así como también su signo:

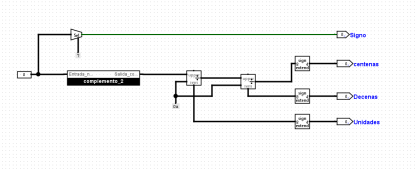

Figure 14: Circuito desglosé número

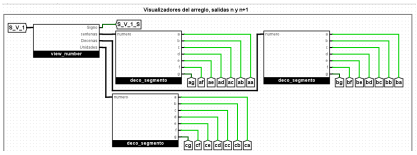

Figure 15: Circuito unificado

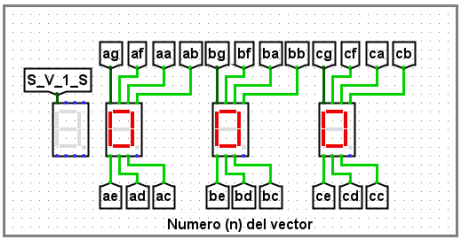

Figure 16: Resultado Visualizador

6 Máquina de estados

Para poder llevar a cabo el algoritmo de reducción, se construyo una máquina de estados, dependiendo de la entrada de 2 bits que, representa la fase en la que se encuentra, realiza cierta cantidad de ciclos y cuando termine genera una salida de **1**, para de esta forma poder controlar el reset de los contadores y otros componentes:

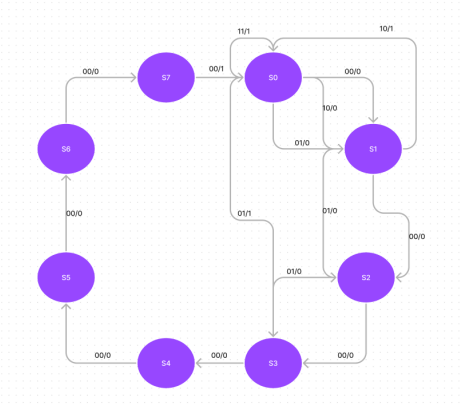

Figure 17: Diagrama máquina de estados

Como el algoritmo para la suma es de complejidad algorítmica $O(\log_2(n))$, significa que para las 16 posiciones del vector, solo usaremos como máximo 8 estados para la primera iteración, 4 para la segunda, 2 para la tercera, 1 para la última fase, teniendo en cuanta lo anterior, se evidencia solo 8 estados en la máquina.

	Entrada		ESTADO ACTUAL			ESTADO SIGUIENTE			z							
	X	Y	Q2	Q1	Q0	Q2N	Q1N	Q0N	Salida	J2	K2	J1	K1	J0	K0
3	0	0	0	0	0	0	0	1	0	0	X	0	X	1	X
4	0	0	0	0	1	0	1	0	0	0	X	1	X	X	1
5	0	0	0	1	0	0	1	1	0	0	X	X	0	1	X
6	0	0	0	1	1	1	0	0	0	1	X	X	1	X	1
7	0	0	1	0	0	1	0	1	0	X	0	0	X	1	X
8	0	0	1	0	1	1	1	0	0	X	0	1	X	X	1
9	0	0	1	1	0	1	1	1	0	X	0	X	0	1	X
10	0	0	1	1	1	0	0	0	1	X	1	X	1	X	1
11	0	1	0	0	0	0	0	1	0	0	X	0	X	1	X
12	0	1	0	0	1	0	1	0	0	0	X	1	X	X	1
13	0	1	0	1	0	0	1	1	0	0	X	X	0	1	X
14	0	1	0	1	1	0	0	0	1	0	X	X	1	X	1
15	1	0	0	0	0	0	0	1	0	0	X	0	X	1	X
16	1	0	0	0	1	0	0	0	1	0	X	0	X	X	1
17	1	1	0	0	0	0	0	0	1	0	X	0	X	0	X

Figure 18: Tabla extendida de la maquina de estados

J2

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	0	1	0	x	x	x	x	x
01	0	0	0	x	x	x	x	x
11	0	x	x	x	x	x	x	x
10	0	0	x	x	x	x	x	x

K2

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	x	x	x	x	0	1	0	0
01	x	x	x	x	x	x	x	x
11	x	x	x	x	x	x	x	x
10	x	x	x	x	x	x	x	x

J1

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	0	1	x	x	x	x	1	0
01	0	1	x	x	x	x	x	x
11	0	x	x	x	x	x	x	x
10	0	0	x	x	x	x	x	x

K1

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	x	x	1	0	0	1	x	x
01	x	x	1	0	x	x	x	x
11	x	x	x	x	x	x	x	x
10	x	x	x	x	x	x	x	x

J0

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	1	x	x	1	1	x	x	1
01	1	x	x	1	x	x	x	x
11	0	x	x	x	x	x	x	x
10	1	x	x	x	x	x	x	x

K0

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	x	1	1	x	x	1	1	x
01	x	1	1	x	x	x	x	x
11	x	x	x	x	x	x	x	x
10	x	1	x	x	x	x	x	x

Z

X,Y \ Q2,Q1,Q0	000	001	011	010	110	111	101	100
00	0	0	0	0	0	1	0	0
01	0	0	1	0	x	x	x	x
11	1	x	x	x	x	x	x	x
10	x	1	x	x	x	x	x	x

Figure 19: Mapas K de la máquina de estados

Expresiones minimizadas:

$$
\begin{aligned}
J0 &= X' + Y' \\
J1 &= X' + Q0 \\
J2 &= Y' \cdot Q1 \cdot Q0 \\
K0 &= 1 \\
K1 &= Q0 \\
K2 &= Q1 \cdot Q0 \\
Z &= Q2 \cdot Q1 \cdot Q0 + Y \cdot Q1 \cdot Q0 + X \cdot Q0 + X \cdot Y
\end{aligned}
$$

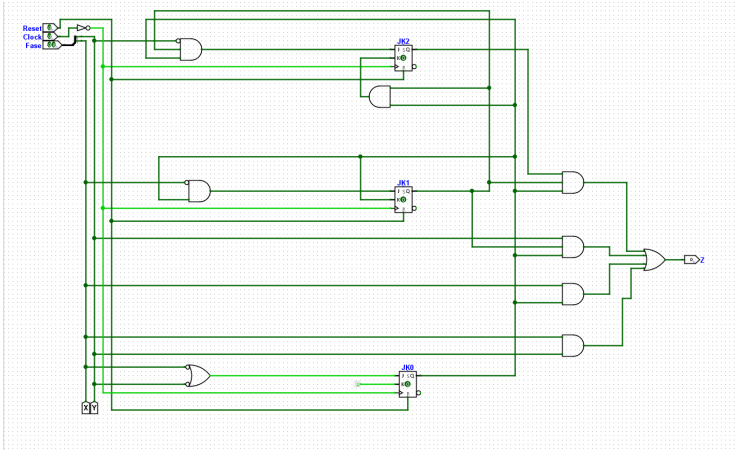

Figure 20: Circuito maquina de estados

7 Ensamble circuito

Teniendo en cuenta los componentes antes presentados, se procede a ensamblar cada uno de ellos para poner en funcionamiento el algoritmo de suma para un vector de 16 posiciones con 8 bits de entrada.

7.1 Visualizador

Con base a los vizualizadores pasados, se construyo uno por cada elemento requerido: ***Número*** A_n ***y*** A_{n+1} ***del vector, Suma de los números, Fase del algoritmo, Estado actual de la máquina de estados.***

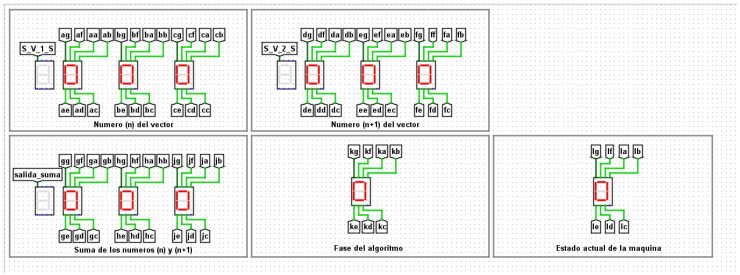

Figure 21: Visualizador de los elementos

7.2 implementación

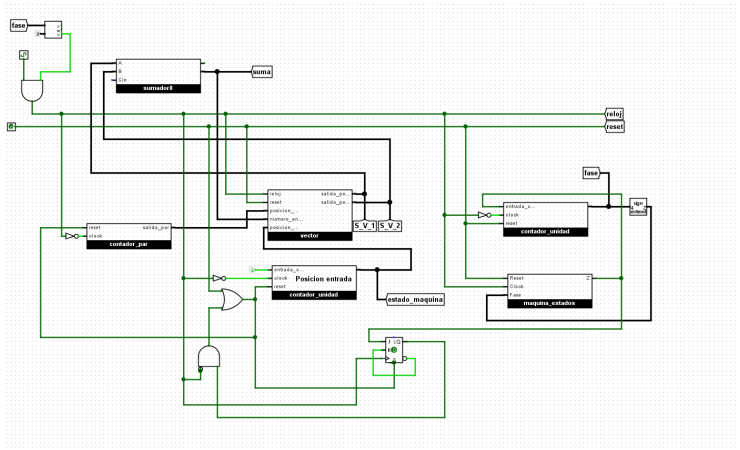

Figure 22: Implementacion del circuito

Todos los **contadores** funcionan con flancos de reloj de bajada, para así permitir que el resultado de sumar el número A_n & A_{n+1} se pueda guardar en la posición η del vector. El vector guarda los números con flancos de subida del reloj, de esta manera diario estará guardando los datos en la posición correcta. Con respecto al flip-flop Jk usado, este permite que la última suma correspondiente al estado de la máquina se guarda en la posición correcta, dado que por el diseño de la misma cuando está en el último estado activa la salida que produce un reset en los contadores.

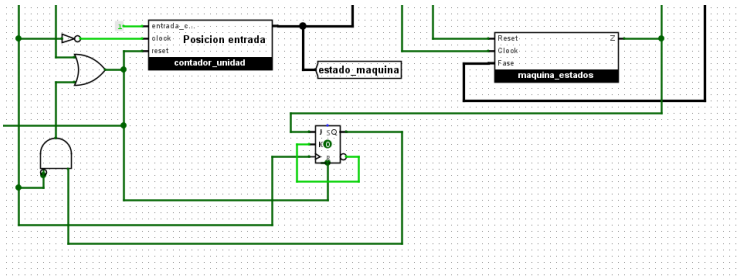

Figure 23: Uso del flip-flop para retardar el reset.

Por último se configuró el circuito para que se detuviera cuando estuviera en la última fase del algoritmo y así evitar que este corra indefinidamente provocando desbordamiento.

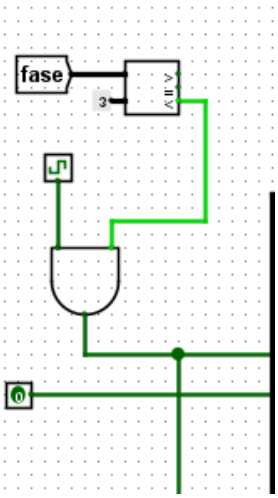

Figure 24: Control fin del algoritmo