PROYECTO N°1

Implementación del diagrama lógico de un circuito controlador - TEMA B

Arquitectura de Computadoras

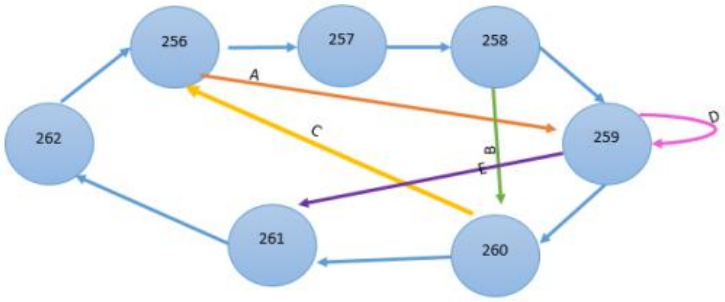
Aravena Joaquín
Graziabile Stéfano

Grupo 12
Tema B

Informe del Proyecto

Introducción:

El circuito es un controlador de estados que permite pasar entre los estados de acuerdo a la figura enunciada.



Las entradas (A-E) representan las líneas externas que, en caso de estar alguna activa, cambia el controlador a su modo alternativo.

Este modo alternativo realiza saltos especiales a estados específicos, a diferencia del normal que cicla automáticamente por todos los estados en orden desde el 256 al 262.

El circuito fue implementado en la versión Logisim Evolution 3.8.0

Etapa 1:

Tabla de estados del circuito:

A	B	C	D	E	Estado Actual	Codificación Actual	Próximo Estado	Próxima Codificación
0					256	000	257	001
1					256	000	259	011
					257	001	258	010
	0				258	010	259	011
	1				258	010	260	100
			0	0	259	011	260	100
			0	1	259	011	261	101
			1	0	259	011	259	011
		0			260	100	261	101
		1			260	100	256	000
					261	101	262	110
					262	110	256	000

Las columnas “Codificación Actual” y “Próxima Codificación” las adoptamos como una optimización al circuito. Esto nos permite recorrer todo el circuito utilizando únicamente los últimos 3 bits, dejando los primeros 6 como constantes. Es decir, cada estado resulta de la suma de 256 y nuestra codificación

Aravena Joaquín
Graziabile Stéfano

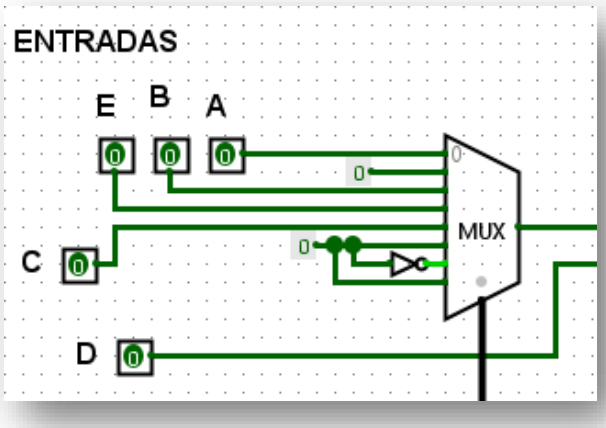
Grupo 12
Tema B

Premisas del proyecto:

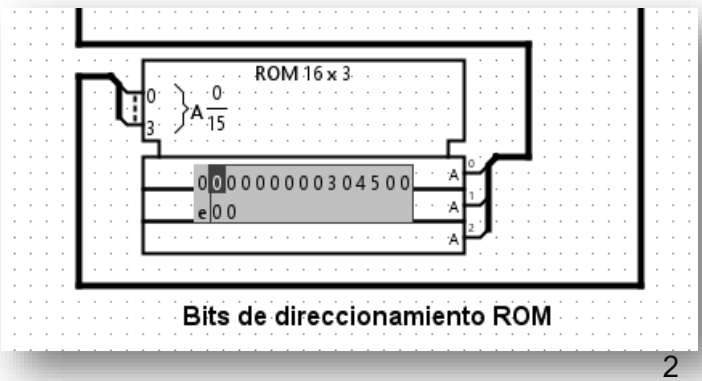
- Se asume que no se pueden activar dos líneas externas en simultáneo.
- Codificamos la “Salida” del circuito en base decimal para que sea más fácil de verificar y visualizar.
- El contador se mantiene constantemente activo por defecto, excepto cuando se encuentra en el estado 259 y se activa la línea D, donde se apaga el enable del contador para que se mantenga en el mismo estado.

Decisiones tomadas en el contador:

- La entrada “Enable” está cableada a la negación de la salida 110 y la compuerta “D”, ya que en dicha situación el contador se mantiene en el estado actual. En cualquier otro momento el contador trabaja normalmente.
- La entrada “Up/Down´” está cableada a una constante con valor “1” ya que el circuito siempre realiza un cambio de estado hacia adelante, o bien realiza un salto utilizando la entrada “Load”.
- Las entradas “Carry” y “Clear” no están cableadas ya que no es necesaria su utilización.
- La entrada “Load” se encuentra cableada a una compuerta OR, utilizando un MUX, el cuál activa esta entrada cuándo se debe realizar un salto en el Modo Alternativo, y también se utiliza el bit más significativo de la salida del contador para poder codificar el inicio del contador en el estado 256.
- Los bits de salida del contador son utilizados como líneas de selección del MUX, comobits de entrada de la ROM y como los bits de salida de todo el circuito.



ROM: siempre codifica los saltos especiales que debe realizar el circuito, los cuáles van a ser activados cuando se active la entrada “Load” del contador. La ROM codifica dichos saltos en hexadecimal, tomando como entrada 4 bits, el primer bit representa el más significativo del contador y es utilizado para distinguir cuándo el contador se encuentra en 0 y es necesario realizar el salto al estado inicial 256.
Los 3 bits restantes respetan la convención adoptada de codificación para cargar el estado al que debe saltar el contador.



MUX: es utilizado para activar la entrada “Load” del contador cuándo el circuito entra en Modo Alternativo. Mientras que la ROM codifica el estado al cuál se va a realizar el salto, el MUX determina cuándo se va a realizar. Las líneas de selección corresponden a los 3 bits menos significativos del estado actual del contador.

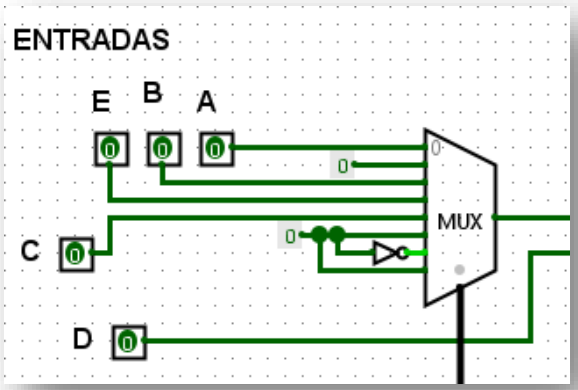
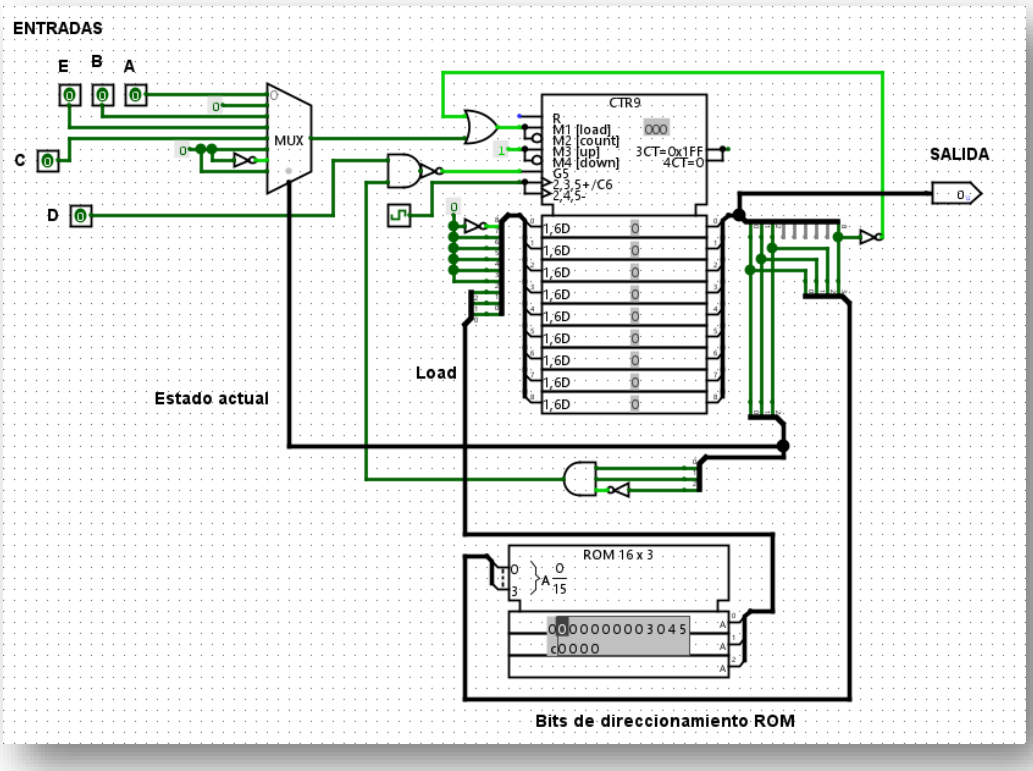


Diagrama del circuito:



Conclusión:

El proyecto fue de utilidad para comprender más a fondo el funcionamiento de un contador, aplicándolo particularmente a este controlador de estados. Además, pudimos combinarlo con otros circuitos, como la ROM y el MUX para que estos trabajen en conjunto y en sincronía, mediante el uso de un reloj.

Podríamos haber hecho uso de un contador de menos bits, usando solo los últimos tres bits. Esta solución permite ahorrar en hardware, pero resulta poco extensible en el caso de necesitarlo.

Etapa 2:

Esta etapa del proyecto implica la implementación total de un sumador CLAA (Carry - Lookahead Adder) en configuración ripple. El fin de este sumador es calcular el próximo estado de los saltos del modo alternativo del circuito.

Los saltos a realizar y las sumas de los mismos se detallan en la siguiente tabla:

A	B	C	D	E	Estado actual	Suma hexadecimal	Próximo Estado
0					256	-	257
1					256	03	259
					257	-	258
	0				258	-	259
	1				258	02	260
			0	0	259	-	260
			0	1	259	02	261
			1	0	259	-	259
		0			260	-	261
		1			260	FC	256
					261	-	262
					262	FA	256

CIRCUITO-PG:

Se utiliza para calcular todos los carries de entrada de las diferentes posiciones, en paralelo (C0 a C3) de un CLAA-4Bits. Esto se logra en dos niveles de compuertas.

Esta generación de los carries en paralelo y en dos niveles de compuertas es una de las ventajas claves de optar por la impmentación del CLAA.

Las **entradas** de este circuito corresponden al carry inicial (c0), los carries propagados(p0-p3), y los generados(g0-g3) en las posiciones previas.

Conociendo c0, el circuito se encarga de calcular los restantes carries de entrada como se detalla en la siguiente imagen:

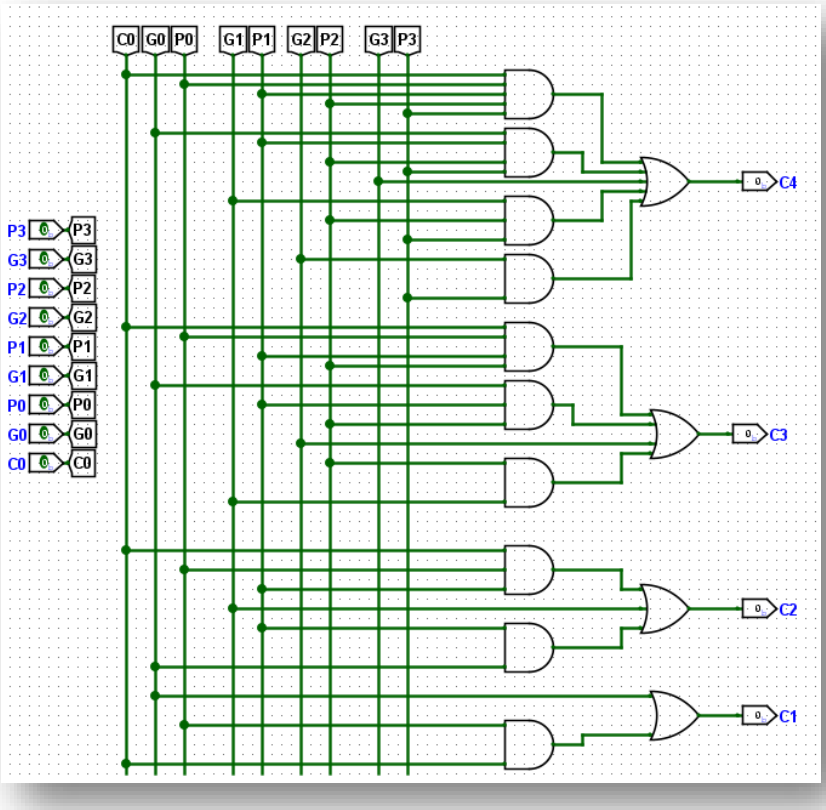
$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

Las cuatro salidas (c1-c4) corresponden a los cuatro carries de entrada generados en paralelo.



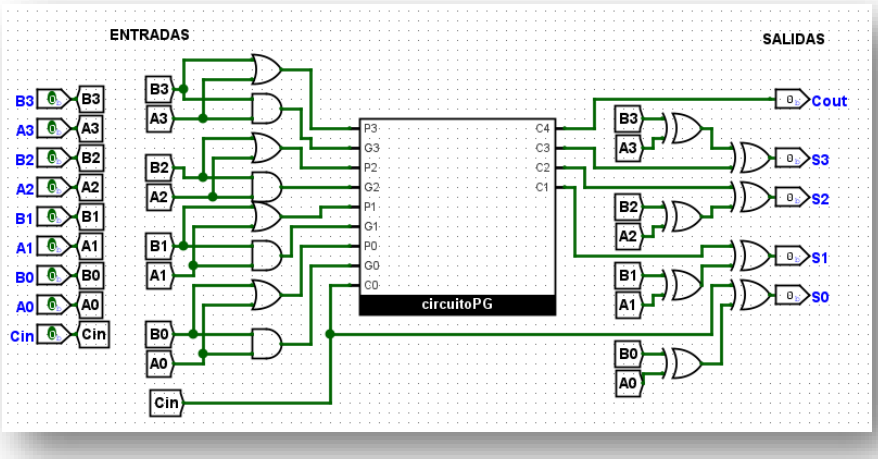
CLAA-4Bits:

Este circuito toma como **entrada** los ocho bits correspondientes a las entradas A y B, los cuales representan los números a sumar en binario; y el carry de entrada inicial (Cin).

Estos ocho bits se agrupan de a dos, (A0, B0 - A3, B3), siendo A0 y B0 los bits menos significativos de las entradas, y A3 y B3 los más significativos.

Con estas entradas se calculan todos los carries de generación y propagación, los cuales sirven de entrada para el circuito **circuitoPG.** Luego, se combinan los carries generados por el dicho circuito, los cuales se utilizan, en combinación con los bits de

entrada, para calcular los **4 bits del resultado** de la suma(S0-S3), y el carry de salida, en caso de haber. El bit de salida S0 corresponde al bit menos significativo del resultado.



CLAA-24Bits:

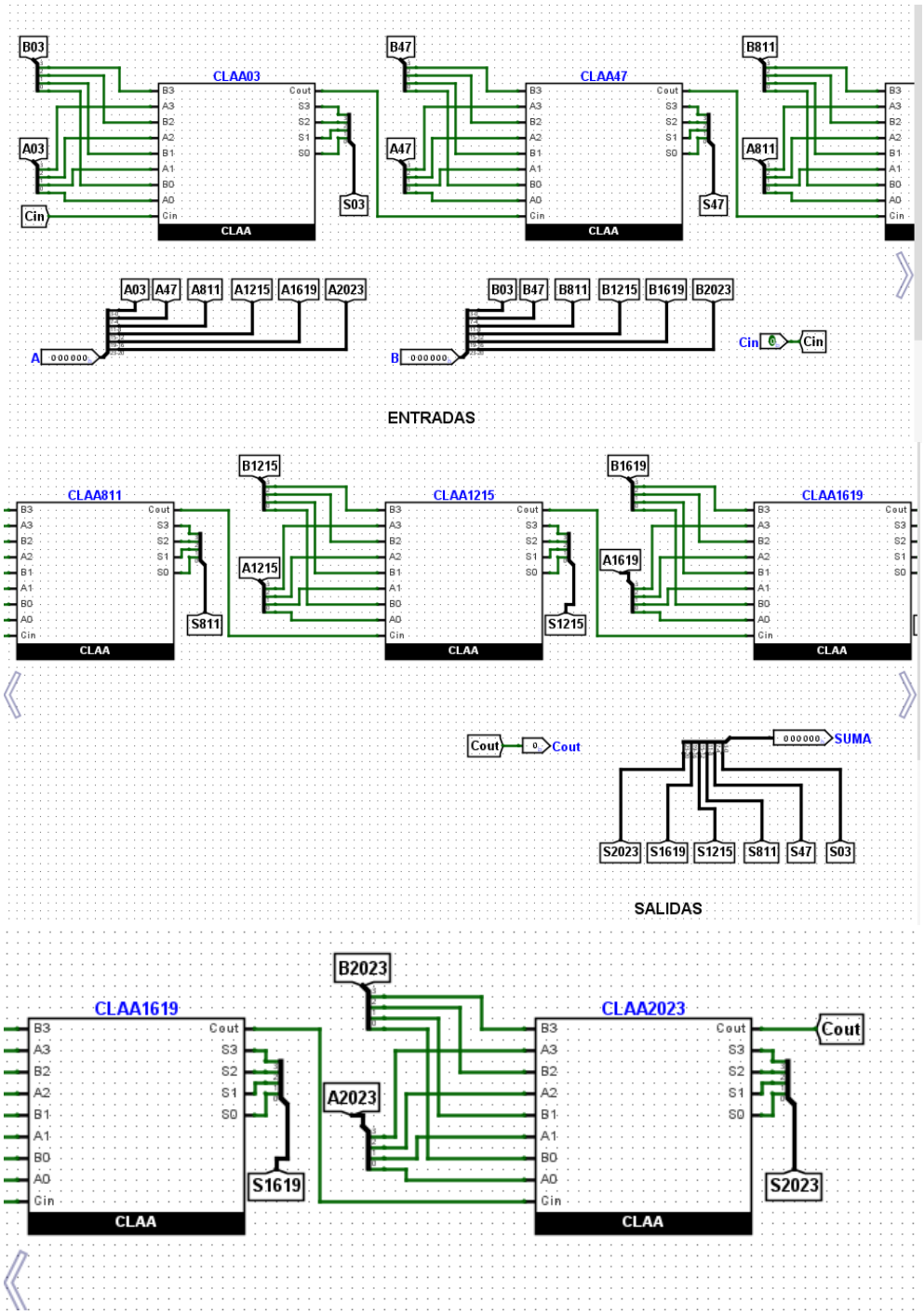
Para lograr un circuito sumador de 24 bits, utilizamos 6 CLAA-4Bits en configuración Ripple.

Este circuito toma como **entradas** los 24 bits correspondientes a la entrada A(A0-A23) y los 24 bits de la entrada B(B0-B23). Las entradas se ingresan en hexadecimal para facilitar la visualización del circuito, evitando largas tiras de 24 bits en los pines de entrada.

Luego, el funcionamiento del mismo es análogo al CLAA de 4 bits, con la diferencia que, en esta implementación, el carry de salida (Cout) sirve como carry de entrada (Cin) del próximo CLAA de 4 bits.

La **salida** del circuito corresponde a los 24 bits del resultado de la suma entre A y B (S0-S23).

Quedan adjuntadas las imágenes correspondientes a este circuito:



Tiempo del CLAA-4Bits:

- 1 retardo compuerta para calcular todos los P y G
- 2 retardos de compuertas para calcular todos los carry dentro del circuitoPG
- 2 retardo de compuerta para computar la suma
- Total: 5 retardos de compuerta.

Tiempo del CLAA-24Bits:

- 1 retardo de compuerta para calcular todos los P y G, de los 6 CLAA a la vez
- 12 retardos de compuerta para propagar el carry a través de todos

los CLAA

- 2 retardos de compuerta para calcular la suma final de todos los CLAA
- Total: 15 retardos de compuerta

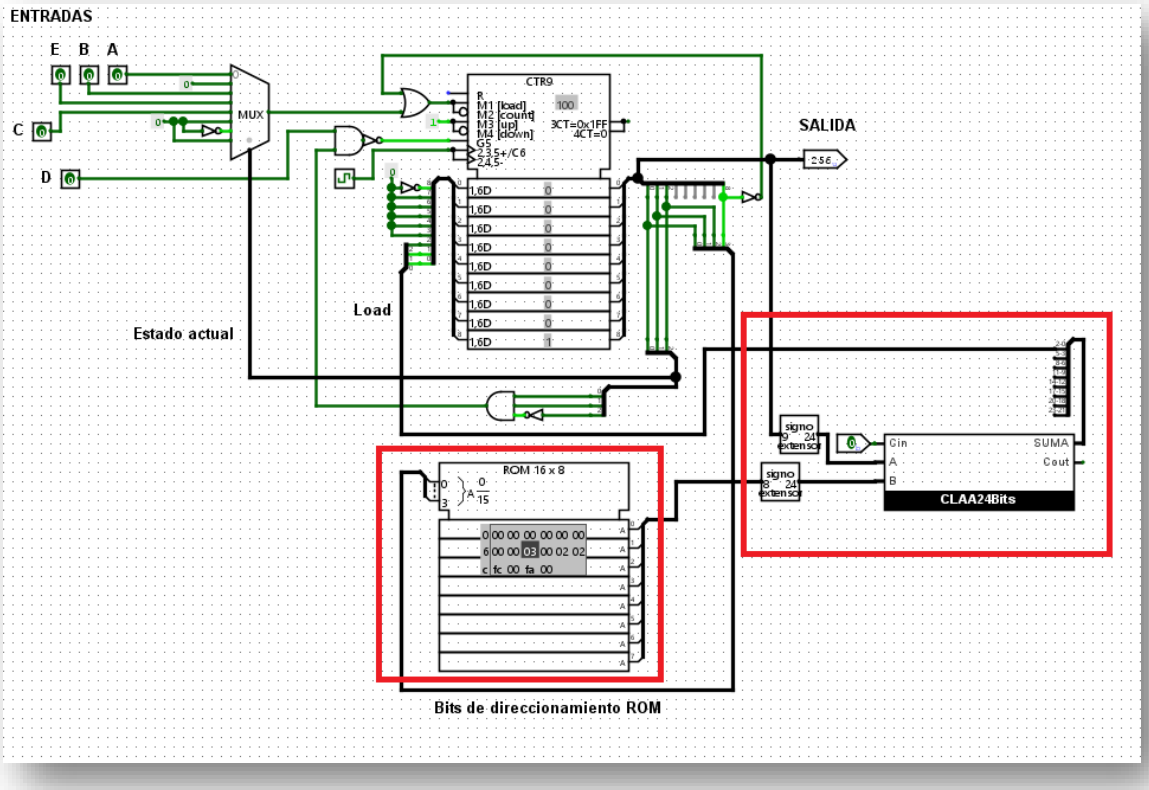
Diagrama del circuito original actualizado:

El circuito original fue modificado para adaptarlo a los requerimientos de la Etapa 2 del proyecto. Se agregó el CLAA-24bits, el cual recibe como entradas el **estado actual**, proveniente del contador, y el número al que se le debe sumar para obtener el estado correcto correspondiente al salto.

Este segundo número se codifica en la ROM, cuyo uso fue modificado para que, en lugar de codificar el estado al que se quiere realizar el salto, codifique el número que se debe sumar al estado actual para realizar el salto correcto. Si el salto es hacia un estado anterior, se ingresa el número a sumar en representación complemento a la base.

Utilizamos extensores a 24 bits para adaptar las salidas, tanto del contador como de la ROM, a 24 bits, para que se adapte a la entrada del CLAA.

Se adjunta a continuación el diagrama actualizado del circuito:



Conclusión:

Se podría haber hecho más eficiente el circuito utilizando un CLAG que genere en paralelo todos los carries de entrada a cada uno de los 6 CLAA-4Bits, en lugar de configurar dichos CLAA en Ripple. De igual manera, el uso de un CLAG es más rápido, a cambio de una mayor demanda a nivel hardware.