Capítulo 4.

DISEÑO DEL ENCODER

4.1 ARQUITECTURA

Un encoder no es más que un elemento que traduce un código termométrico a código binario de N bits. A la hora de diseñar un encoder tenemos varias opciones y arquitecturas. En nuestro diseño el encoder deberá tener en cuenta los siguientes aspectos:

1. La reasignación de comparadores, es decir, cualquier comparador puede ser reasignado a cualquier otro código.

2. Solamente serán válidas un grupo de salidas de comparadores (las de los comparadores seleccionados en la etapa de calibración). Las salidas de los comparadores no seleccionados deben ser ignoradas.

3. Las salidas de los comparadores pueden no formar un código termométrico debido a los errores de burbuja.

4. El número de comparadores será mayor de 63 debido a los valores extra de referencias de tensión añadidas en los extremos del rango de tensiones a la entrada para asegurar una densidad uniforme de referencias de tensión. Estas referencias de tensión se añadirán colocando más resistencias en la escalera, tanto por arriba como por abajo, con lo cual el número de comparadores de la escalera aumentará.

En la figura 4.1 podemos ver algunas soluciones arquitecturales para un encoder. En la figura 4.1(a) no hay reasignación de comparadores con lo que podemos usar una estructura tradicional para nuestro encoder. En la figura 4.1(b) las referencias de tensión de los comparadores 4 y 5 están intercambiadas debido a que hay una burbuja en la salida de uno de los comparadores. Para corregir esto los comparadores 4 y 5 deben ser reasignados. Una posible solución para esto consiste en reasignar explícitamente las salidas de los comparadores 4 y 5 para formar un código termométrico (ver figura 4.1(b)). Después de esto podemos usar un esquema de encoder tradicional para completar el proceso de conversión. El problema de esta solución es que requiere una gran matriz de conexiones que consumirá mucha área y potencia. Esta matriz será lógicamente mayor, cuanto mayor sea el número de bits (N) de nuestro código binario.

Un esquema de encoders muy usado para conversiones A/D de alta velocidad consiste en contar el número de “1” sumando. El hecho de contar los “1” sumando, hace que el encoder sea inmune a los errores de burbuja. Como podemos ver en la figura 4.1(c), obtenemos una salida digital totalmente correcta con el simple hecho de sumar los “1”. A la hora de usar esta arquitectura debemos tener en cuenta la redundancia de comparadores de manera que sólo se puedan contar como máximo un total de NC unos, siendo NC el número de comparadores sin hacer redundancia, es decir, las salidas de los comparadores no seleccionados deben ser ignoradas. Esto no es difícil de implementar en nuestro diseño ya que los comparadores no seleccionados tienen por defecto su salida a “0” y no contribuyen por tanto al resultado de la suma. Un encoder diseñado mediante suma, se encarga implícitamente también de la reasignación de comparadores. Es por

esta facilidad y sencillez de operación por lo que hemos escogido un encoder basado en suma para nuestro esquema.

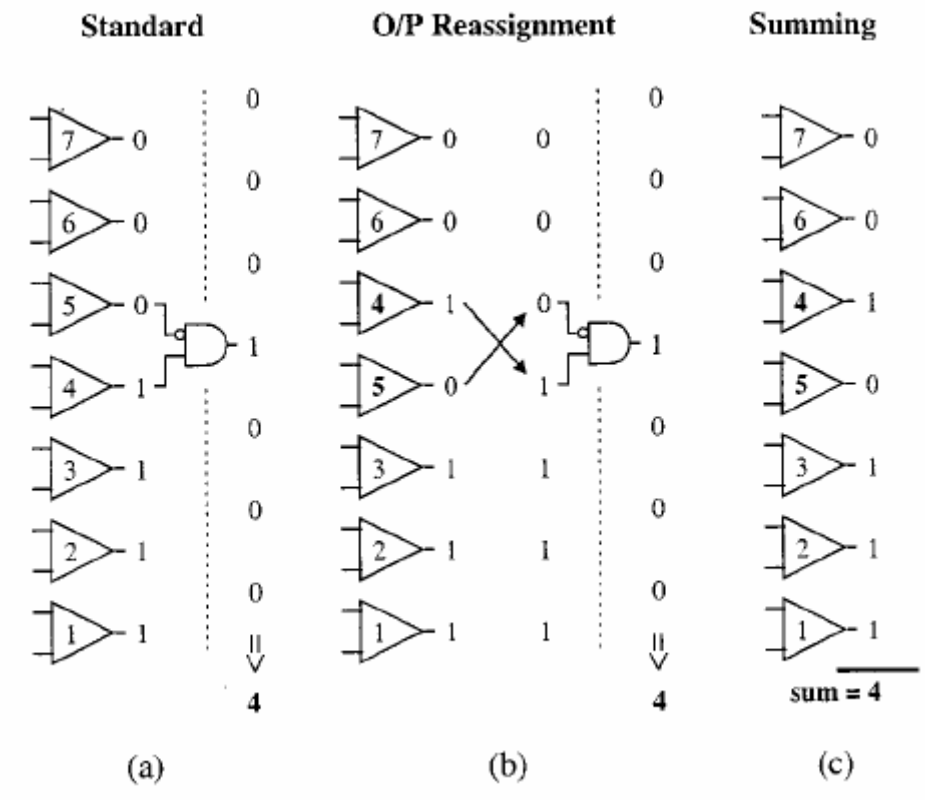

Figura 4.1. Soluciones para el encoder

A la hora de comenzar a diseñar nuestro encoder, es necesario tener en cuenta dos aspectos esenciales comentados anteriormente.

1. Referencias de tensión extra: Las referencias de tensión que vamos a añadir a cada extremo de nuestro tango de tensiones, implicará que se tengan que añadir resistencias a cada extremo de nuestra escalera. Debido a qque el número de referencias de tensión está ligado al número de comparadores mediante la expresión.

$$NC = NR - 1$$

Donde NC=número de comparadores y NR=número de resistencias de la escalera de resistencias.

En principio el número de comparadores de nuestro Flash es de 63 ya que:

$$NC = 2^N - 1$$

Siendo N=número de bits. En nuestro caso N=6 tendremos, NC=63 y por tanto NR=64.

Como ya comentamos en el capítulo anterior, debemos añadir 10 referencias de tensión extra por arriba y por debajo de nuestra escalera aproximadamente. Para simplificar nuestro diseño del encoder, tomaremos un número de referencias extra a añadir que sea múltiplo de 2^N. Como 10 no es múltiplo de 2^N, cogeremos el número inmediatamente inferior que sea múltiplo de 2^N, es decir, tomaremos 8 referencias extras de tensión por cada lado. Lo que implica un total de:

$$NR = 8 + 64 + 8 = 80$$

$$NC = 79$$

Tendremos entonces un total de 80 resistencias y 79 comparadores en nuestro Flash sin hacer redundancia.

2. Redundancia: como ya vimos en el capítulo anterior, la redundancia de comparadores es un aspecto básico en nuestro diseño. Como ya dijimos también en ese capítulo, el número idóneo para hacer redundancia era 4, debido a que con esta cantidad de redundancia obtenemos buenos valores de SNDR, pudiéndonos permitir en los comparadores un offset bastante alto (de unos 5LSB). Por tanto nuestro esquema Flash básico formado por escalera de resistencias y comparadores deberá ser repetido hasta cuatro veces en nuestro circuito. Obviamente, y ya que la escalera de resistencias es la misma para los cuatro bloques, no implementaremos cuatro escaleras para nuestro esquema ya que esto incrementaría el consumo mucho. Lo que haremos será utilizar la misma escalera de resistencias para los cuatro bloques de comparadores.

En la figura 4.2 podemos observar el esquema resultante una vez tenidos en cuenta estos dos aspectos fundamentales para el diseño del encoder. Como podemos observar en la gráfica, el número total de entradas que irán al encoder será de 315 entradas. Así mismo, podemos ver como en el cuarto bloque de comparadores no tenemos 8 resistencias extras por debajo sino que sólo tenemos 7. Esto está relacionado con el número de entradas que van al encoder, ya que como veremos más adelante detalladamente, con 315 entradas el circuito del encoder se simplifica mucho más que si tenemos 316 entradas.

Una vez que hemos elegido el método de suma como método idóneo para el encoder, debemos diseñar la estructura de dicho encoder. Para ello nos basaremos en un artículo de R. Kanan[1]. A pesar de que en este artículo no encontramos un encoder que se ajuste a nuestras necesidades, si que nos da una idea de cómo diseñarlo. Tras descartar otras posibilidades, el esquema final con el que nos quedamos para nuestro diseño es el mostrado en la figura 4.3.

[1] *"A 640mW high accuracy 8-bit lGHz flash ADC encoder"*

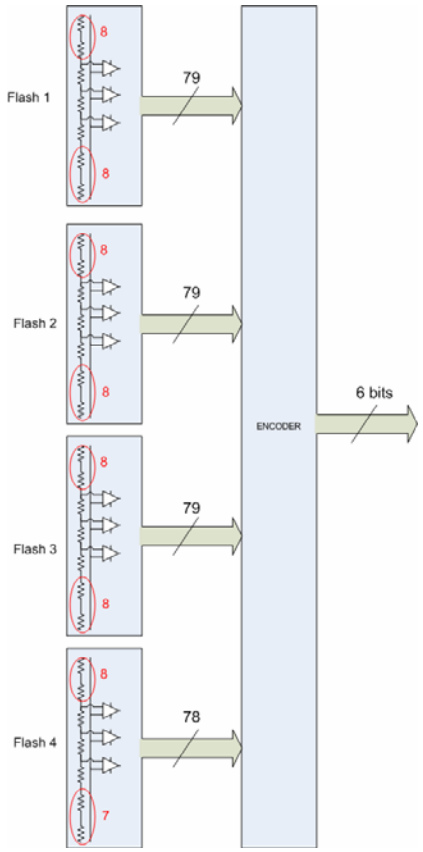

Figura 4.2. Esquema con referencias de tensión extras y redundancia

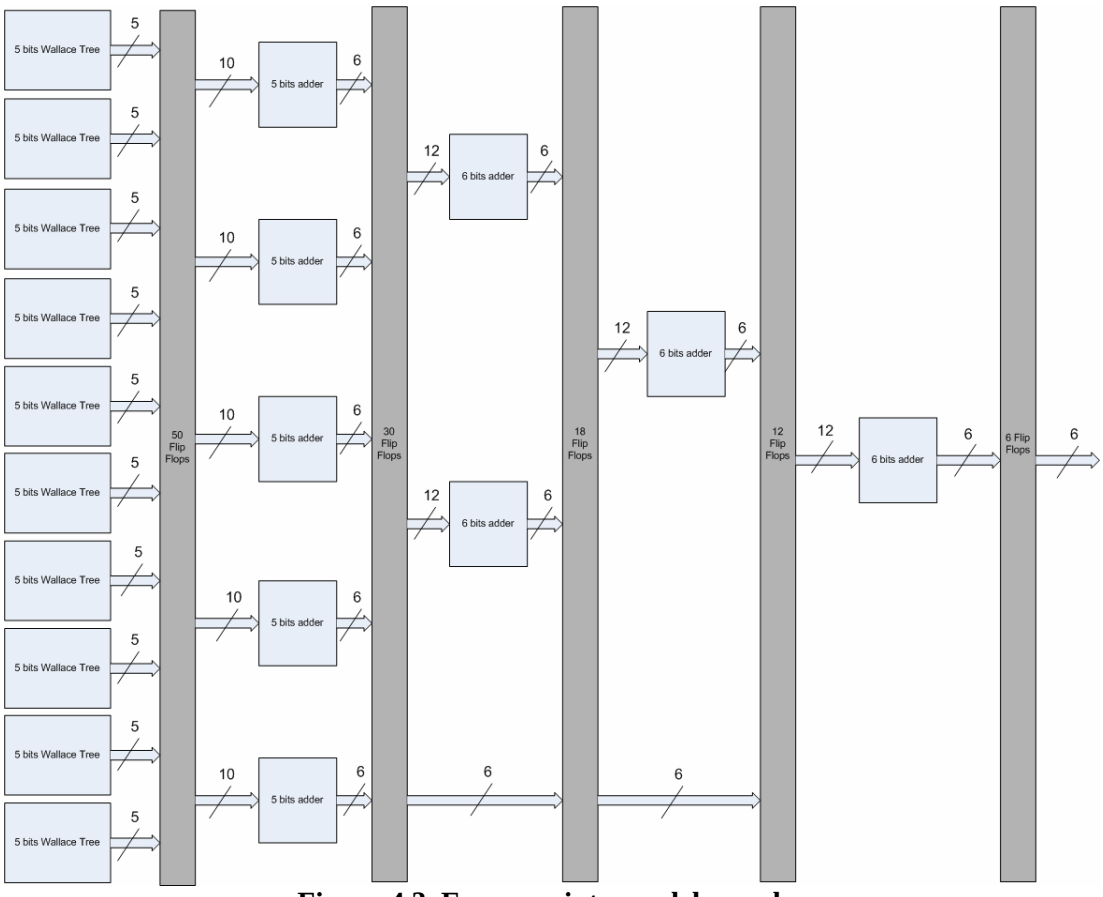

Figura 4.3. Esquema interno del encoder

Como vemos en el esquema anterior, el objetivo es realizar varias cuentas de forma paralela y después sumarlas todas en paralelo también, con una estructura en pipeline.

Cada etapa está separada de la etapa posterior y anterior mediante un bloque de biestables con lo que conseguimos una perfecta estructura en pipeline. El inconveniente de esta estructura en pipeline es que tardaremos más en obtener la primera salida del encoder, concretamente, tardaremos 5 ciclos de reloj en obtener dicha salida ya que tenemos 5 etapas de pipeline. La gran ventaja es que después de estos 5 ciclos obtendremos las salidas en cada ciclo de reloj sin tener que esperar a que las señales atraviesen todo el retraso combinacional de las puertas que hay desde la entrada hasta la salida.

Como podemos ver en el esquema, nuestro encoder estará compuesto por:

- Árboles de Wallace de 5 bits: que serán los circuitos encargados de contar los "1".
- Sumadores de 5 y 6 bits: que serán los encargados de sumar las diferentes cuentas que se realizan de manera independiente.
- Biestables tipo D: serán los encargados de dividir nuestro circuito en varias etapas con el objetivo de hacer una estructura en pipeline.

El diseño de estos elementos se detalla apliamente a continuación.

4.2 ÁRBOL DE WALLACE

Como ya hemos comentado anteriormente y hemos mostrado en el esquema del encoder, usaremos la estructura de árbol de Wallace para contar los "1" debido a que es la estructura más rápida para la cuenta de unos. En el artículo de R. Kanan mencionado anteriormente podemos encontrar el esquema de un árbol de Wallace de 4 bits. No obstante nosotros necesitamos uno de 5 bits con lo que será necesario hacer un diseño propio nuevamente.

El árbol de Wallace se basa únicamente en celdas elementales de suma. En la primera etapa lógica del árbol se cuenta el número de "1" a la entrada y expresa el resultado a la salida con un código binario de 2 bits. La segunda etapa consiste en sumar las salidas de 2 bits de las celdas adyacentes dos a dos, dando una salida binaria de 3 bits, y así sucesivamente hasta obtener el código binario final a la salida para el árbol. El número de celdas básicas de un árbol de Wallace de N bits viene dado por la expresión:

$$n = 2^N - N - 1$$

Con lo que un árbol de Wallace de 5 bits como los que necesitamos tendrá un total de 26 celdas elementales. Siguiendo el esquema del árbol de Wallace de 4 bits expuesto en el artículo mencionado anteriormente, podemos llegar a un árbol de 5 bits como el mostrado a continuación. Como podemos ver en la gráfica, el árbol de Wallace de 5 bits ha sido obtenido a partir del árbol de Wallace de 4 bits, duplicando la estructura, con esto conseguimos un bit más de resolución. Esta técnica de duplicar la estructura para conseguir un bit más de resolución es muy empleada a la hora de diseñar circuitos digitales ya que en un árbol de Wallace de N bits de resolución tenemos:

$$2^N \text{ posibles valores de entrada}$$

Si duplicamos la estructura, en el nuevo árbol que obtenemos tendremos ahora:

$$2 \times 2^N = 2^{N+1} \text{ posibles valores de entrada}$$

Es decir, la resolución de este nuevo árbol construido duplicando la estructura será N+1 bits, con lo que vemos que duplicando la estructura, la resolución ha aumentado en 1 bit. Siguiendo este razonamiento pasamos del árbol de Wallace de 4 bits explicado en el artículo anterior al árbol de Wallace de 5 bits mostrado a continuación:

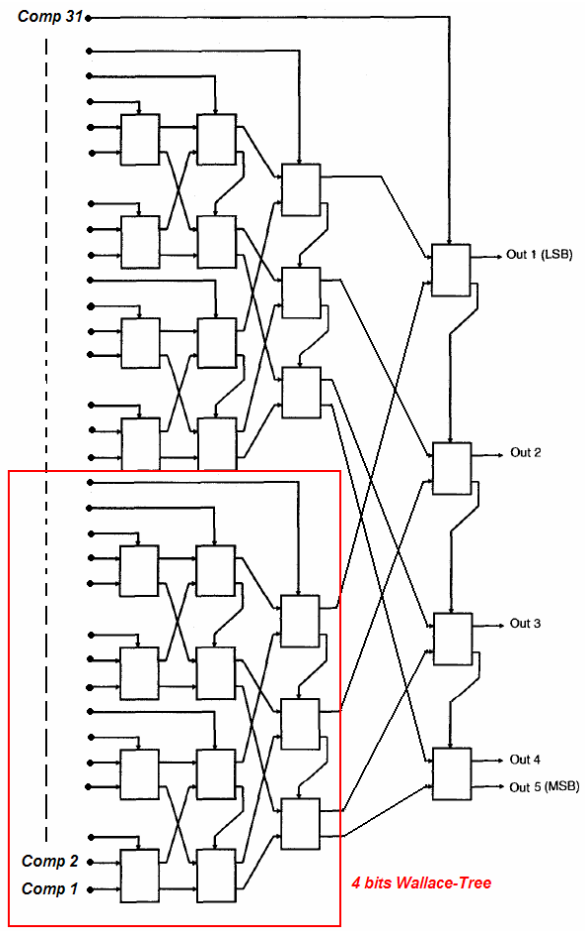

Figura 4.4. Árbol de Wallace de 5 bits

Como ya comentamos en el apartado anterior, la estructura de nuestro encoger se simplifica mucho si el número de entradas es 315 en lugar de 316, es por eso por lo que hemos eliminado una referencia de tensión en el bloque de comparadores más bajo. Este hecho, está relacionado con lo que acabamos de explicar referente al duplicado de estructuras. Como vemos en la figura 4.3, la entrada de nuestro encoger está formada por árboles de Wallace de 5 bits. Cada árbol de Wallace tiene 2^N-1 entradas, es decir que cada árbol de Wallace de 5 bits tiene 31 entradas. Además debemos saber que por cada dos árboles de Wallace podemos incluir otra entrada extra, que irá directamente conectada a la segunda etapa de celdas unidad del árbol. Este fenómeno puede observarse en la figura 4.4. Como vemos al duplicar dos árboles de Wallace de 4 bits no hemos obtenido 30 entradas sino 31. Esa entrada extra ha sido conectada directamente a la segunda etapa lógica de celdas del árbol de 5 bits como podemos ver en la figura. Exactamente lo mismo ocurre cuando duplicamos árboles de 5 bits, por cada dos árboles conseguimos una entrada extra. Entonces si llamamos NW al número de árboles de Wallace de 5 bits usados en la primera etapa de nuestro encoder y NE al número de entrdas disponibles en nuestro encoder, podemos calcular el número de entradas disponibles para nuestro encoder mediante la siguiente expresión:

$$NE = 31 \cdot NW + \frac{NW}{2}$$

En nuestro caso tenemos:

$$NE = 315$$

Es por esto por lo que el número de entradas ha sido reducido de 316 a 315, así nos evitamos tener que volver a duplicar la estructura con lo que se añadirían unos cuantos árboles de Wallace más que sería desaprovechados ya que sólo usaríamos una de sus entradas. Con 315 entradas conseguimos una estructura óptima.

Como vemos en el esquema anterior del árbol de 5 bits, el árbol está formado únicamente por celdas elementales de tres entradas y dos salidas. Las salidas de estas celdas dependerán del número de unos que tengamos a la entrada de ellas. La relación entre las entradas y las salidas en cada celda es la siguiente:

Number of ONE at the entries	Output binary code
0	00
1	01
2	10
3	11

Figura 4.5. Relación entre las entradas y las salidas de una celda unidad

Con esta relación entre entradas y salidas, y usando un mapa de Karnaugh, podemos sacar con facilidad la expresión de la celda unidad mediante puertas lógicas. Si llamamos A, B y C a las entradas y S1 y S2 a las salidas obtenemos:

A	B	C	S1	S2
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

A B \ C	0	1
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	1

A B \ C	0	1
0 0	0	1
0 1	1	0
1 0	1	0
1 1	0	1

S1: (A and B) or [(A or B) and C] = AB + AC + BC

S2: [(A xor B) and not C] or [(A xnor B) and C] = (A$\oplus$B)notC + not(A$\oplus$B)C = A$\oplus$B$\oplus$C

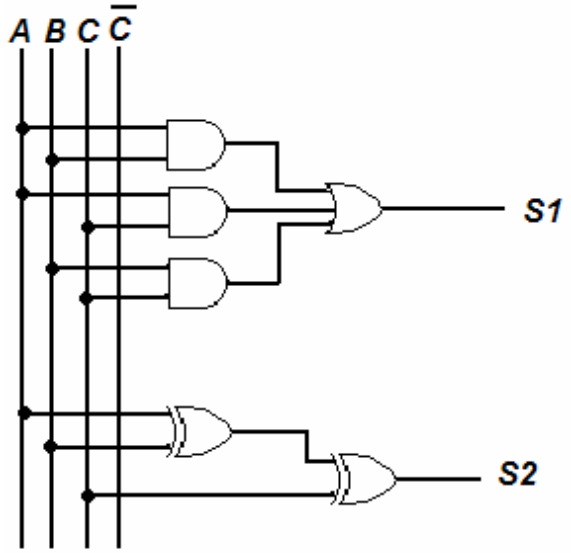

Figura 4.5. Celda unidad obtenida mediante mapas de Karnaugh

4.3 SUMADORES

Como ya hemos mencionado anteriormente y mostrado en la figura 4.3, serán necesarios distintos tipos de sumadores para nuestro encoder. En concreto serán necesarios sumadores de 5 y 6 bits. Como podemos ver en la figura 4.3, estos sumadores se encargan de ir sumando en paralelo .los "1" contados por los árboles de Wallace. La suma de los "1" contados no se hace toda de una sola vez sino que se divide en etapas separadas una de otra mediante biestables formando una estructura en pipeline, como ya explicamos anteriormente.

Normalmente los sumadores suelen ser circuitos bastante lentos ya que aunque realizan con bastante rapidez el cálculo de una suma, tiene que esperar a que les llegue el acarreo de la etapa anterior con lo cual el retraso combinacional se va incrementando hasta hacerse significativo. Debido a que en nuestro circuito necesitamos sumadores de la mayor velocidad posible, optaremos por sumadores que minimicen este retraso de los acarreos. Estos sumadores son los sumadores con selección de acarreo.

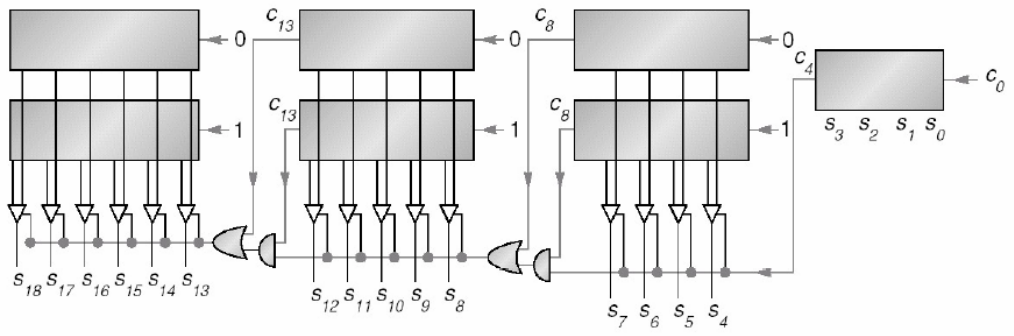

Figura 4.6. Sumador con selección de acarreo

Como podemos ver en la figura, en cada etapa se calcula la suma para los dos posibles acarreos, tanto para acarreo "0" como para acarreo "1". Después el valor correcto se elige dependiendo del acarreo de la etapa anterior mediante una lógica de selección que no es más que un multiplexor.

4.4 FUNCIONAMIENTO

Una vez que ya tenemos diseñada la arquitectura de nuestro encoder, pasaremos a montarlo y simularlo con CADENCE. En primer lugar vemos que en las librerías dadas por el fabricante, no disponemos de puertas AND ni OR, pero si NAND, con lo cual lo primero será expresar el circuito de la figura 4.5 con las puertas adecuadas a nuestra tecnología.

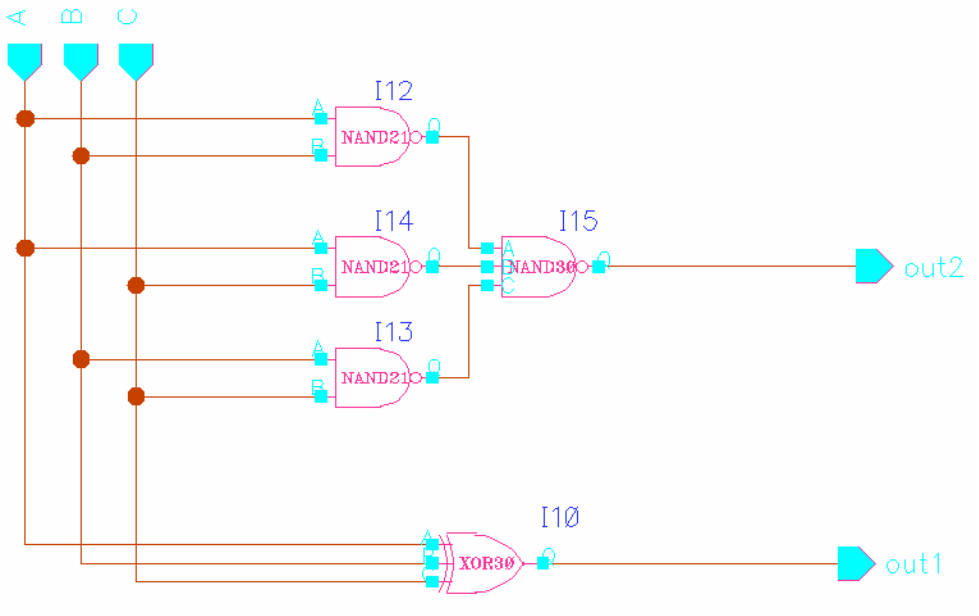

Figura 4.7. Celda unidad en CADENCE

Una vez que tenemos esto pasamos a montar los árboles de Wallace de 4 y 5 bits y a continuación pasamos a montar los sumadores y los bloques de biestables. Una vez que tenemos todos los bloques montados, pasamos a construir el esquema de la figura 4.3, obteniendo lo siguiente:

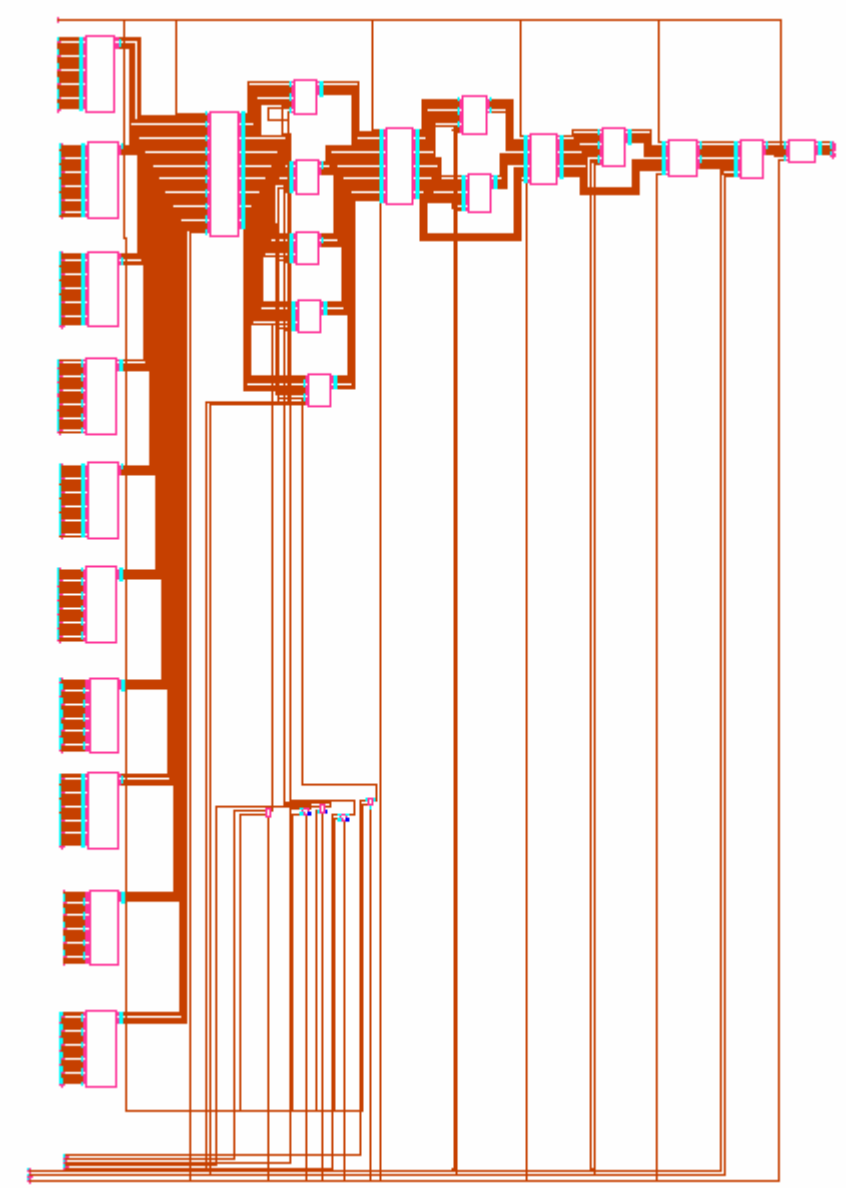

Figura 4.8. Esquema del encoder en CADENCE

Es importante señalar que las 5 entradas extras que sacamos fruto de la duplicación de los árboles de Wallace de 5 bits, deberán ser retrasadas mediante biestables antes de ser conectadas a los sumadores de 5 bits. Estos biestables serán necesarios ya que estas 5 señales no han pasado por el primer bloque grande de biestables con lo que irán adelantadas respecto a las 310 señales restantes.

Para probar el funcionamiento del encoder haremos u análisis variando las entradas. Conectaremos dos fuentes de tensión que den pulsos cuadrados con distinto periodo. Una de las fuentes la conectamos a las 6 entradas más bajas y la otra a las 9 entradas más altas, de manera que cuando las dos fuentes estén a “1” la salida deberá ser 15, cuando sólo una de ellas esté a “1”, la salida deberá ser 6 o 9, dependiendo de la que esté a “1” y cuando las dos estén a “0” la salida deberá ser “0”. El funcionamiento se detalla en el siguiente diagrama.

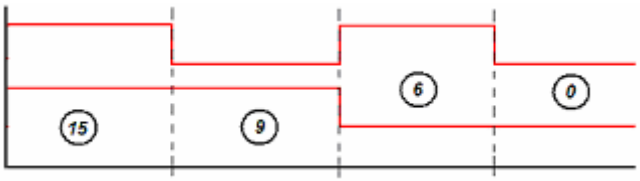

Figura 4.9. Cambio de las entradas y resultado que se debe obtener

Con estas entradas el resultado obtenido trabajando a una frecuencia de 300Mhz es el siguiente:

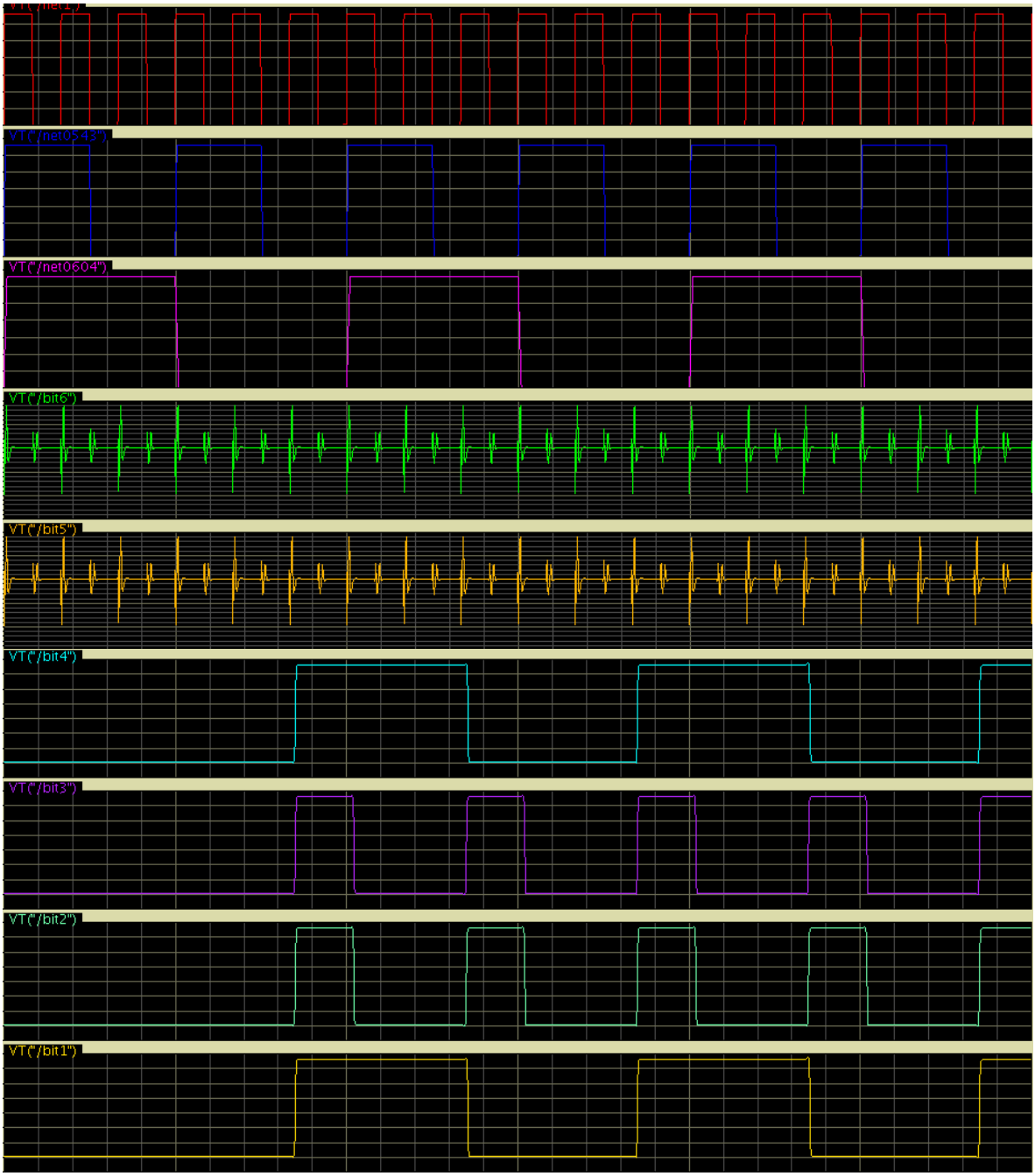

Figura 4.10. Resultado de simulación del encoder en CADENCE

Como vemos en la gráfica, en primer lugar tenemos el reloj, que en este caso es de 300Mhz. Esta ha sido la frecuencia más alta para la que hemos podido conseguir un correcto funcionamiento en simulación, por lo que esta será la frecuencia que limite la frecuencia de funcionamiento de nuestro convertidor por el momento. Como vemos las salidas correctas empiezan a producirse una vez pasados 5 ciclos de reloj como ya anunciamos anteriormente debido a la estructura en pipeline. A continuación tenemos los valores que van tomando las dos fuentes y por último tenemos las salidas de los 6 bits desde el MSB hasta el LSB en orden descendente. Como podemos observar si nos fijamos en los cuatro bits menos significativos, las salidas son las que ya esperábamos, es decir: 15, 9, 6 y 0.

Por tanto podemos concluir diciendo que a pesar de la buena arquitectura seleccionada (la más rápida posible), la frecuencia de nuestro encoder y por tanto de nuestro convertidor esta por ahora limitada a 300Mhz.