Transcripciones de las presentaciones de clases de teoría 2018

IMPORTANTE: Estas notas de clases sirven como complemento de los apuntes ya editados por esta cátedra y no deben ser considerados como el material didáctico final a estudiar. Se aprovecha en las mismas refrescar ciertos conceptos vertidos en los mismos, complementarlos y actualizarlos.

TEMA 8: Circuitos aritméticos:

Filmina 1: Dentro de la variada gama de circuitos digitales, tenemos los denominados circuitos aritméticos. Estos tienen como objetivo realizar operaciones aritméticas en formato binario o BCD, punto fijo o punto flotante. Dependiendo de la aplicación se utilizarán unos u otros.

Filmina 2: Desde el punto de vista de cómo se procesan los datos tendremos que pueden ser del tipo "serie" o "paralelo". En el primer caso los datos se van presentando al circuito de a un bit por vez, generalmente comenzando primero con el LSB (bit menos sgnificativo). En el segundo, los datos se presentan en formato paralelo, es decir, todos los bits simultáneamente.

Dependiendo de la función a realizar, tenemos sumadores, restadores, multiplicadores, divisores y funciones combinadas de los mismos para realizar operaciones complejas como por ejemplo el cálculo de raíz cuadrada, exponenciales, etc.

Si bien es posible generar funciones complejas en base al uso de un microprocesador, a través de algoritmos que se corren en un programa, la posibilidad de generar dichas funciones en hardware, en muchos casos, presentan ventajas en cuanto a velocidad y/o el empleo de menores recursos lógicos, como es el caso de la ausencia de un micro para realizarlas.

Filmina 3: Un sumador en formato paralelo en punto fijo, es un circuito que generalmente realiza la operación de suma entre dos o más operandos de longitud de palabra N.

La suma puede ser con o sin signo, en punto fijo o punto flotante.

De las estructuras más conocidas tenemos las: "ripple-carry", "look-ahead-carry ", " carry-select" y "carry-save", cada una con características particulares.

Algunas presentan ventajas en cuanto a velocidad de respuesta mientras que otras requieren menor uso de recursos lógicos.

Filmina 4: Un sumador ripple-carry de N bits se basa en la utilización de sumadores de un bit que se van encadenando a fin de realizar la operación de suma.

Es una estructura sencilla que se repite tantas veces como bits tengan los operandos.

Su más grande desventaja es la de acarrear retardos de propagación lo que lo constituye en un dispositivo lento dependiente del número de bits con los que se trabaje.

Filmina 5: Un semi-sumador (ó half-adder) es un sumador de un bit entre dos operandos A y B, tal que como salidas tiene el “carry” (o bit de acarreo) y el bit de “suma”.

Generalmente se usa para la suma de los bits menos significativos, ya que no se necesita entrada auxiliar de carry alguno.

Su síntesis se realiza con una compuerta Or-exclusiva y una AND.

Filmina 6 a 9: Un sumador de un bit completo (full-adder), requiere de una entrada de carry.

La suma nuevamente emplea compuertas Or-exclusiva, mientras que la síntesis de la salida de acarreo para la etapa siguiente puede tener diferentes soluciones. Aquí se presentan dos de ellas. Una de ellas presenta menor cantidad de niveles de retardo por lo que será más rápida.

Filmina 10 a 13: Para armar un sumador de N bits basta con concatenar a N sumadores de un bit.

En ese caso, el menos significativo deberá tener su entrada de carry en “0”.

Si consideramos por ahora que una compuerta común (AND, NAND, OR, NOR, NOT) tienen un retardo tpd y una Or-exclusiva, 2tpd, tendremos que si usamos el circuito de generación de carry que tiene dos niveles de compuertas, la salida en un sumador de un bit tarda 4tpd mientras que el carry a la etapa siguiente tarda 2tpd.

En la estructura del sumador se pude observar que el carry de salida está siempre 2tpd con más retardo que el de entrada dado que todos los Ai y Bi sin importar el órden, se considera estables en tiempo t=0.

Si esos retardos se van acumulando tendremos que por ejemplo en un sumador de 5 bits, el último bit de suma se estabilizará en 10tpd al igual que el último carry.

A medida que crece el número de bits, aumentará el retardo.

Cabe aclarar que el cálculo hecho se toma para la peor condición que es cuando la operación de suma hace cambiar hasta el último bit de suma.

NOTA: Para tener coherencia con la literatura aquí se ha considerado que una compuerta OR-EXCLUSIVA tiene el doble de retardo que una normal (INV, OR, NOR, AND, NAND). Pero eso dependerá de la familia lógica que la represente como se verá en el Tema 9.

Filmina 14: Un diseño alternativo que ya tiene varios años, es el denominado “look-ahead carry” o acarreo por adelanto. Significa que la forma de realizar los cálculos del bit de suma y carry son diferentes de tal forma que se pueda preveer cual será el valor de éstos antes de tener las tres entradas estabilizadas. Si bien esta arquitectura es más rápida que la de ripple-carry paga un precio en cuanto a la necesidad de mayor lógica para generar al sumador.

El algoritmo se basa en definir dos funciones denominadas “G” ó generate y “P” ó propagate. Ambas sólo dependen de los operandos del bit en cuestión.

Un sumador de un bit con este algoritmo queda como figura en la filmina 15.

Se observa que siempre los Gi y Pi dependen sólo de las entradas de datos LAS CUALES SIEMPRE ESTARÁN ESTABLES EN TIEMPO “CERO” ya que son la referencia temporal y por lo tanto estarán estabilizadas sus salidas en 2tpd y tpd respectivamente sin importar su posición.

Los carry se van generando en base a estas funciones. Se observa que a medida que crece la posición del carry (CO, C1, C2, C3, etc.) mayor cantidad de lógica se necesita para sintetizarlos.

En la filmina 17 tenemos los circuitos sintetizados de dichas ecuaciones, siendo la de C4 el más complejo.

Si analizamos las respuestas temporales podemos ver que C1 se estabiliza en 4tpd al igual que el resto.

Los bits de suma se estabilizan en 6tpd salvo S0 que tiene un retardo de 4tpd.

Es decir, que para un sumador de 4 bits, en 6tpd las salidas están todas estabilizadas, a diferencia del sumador ripple-carry que era de 10tpd.

La desventaja es con respecto al aumento de la lógica para la síntesis de los carry.

Filmina 19 a 20: El 74HC283 es un sumador comercial de 4 bits que utiliza esta arquitectura.

Su circuito es un poco diferente pero sobre la misma base. Los retardos difieren ya que los retardos reales de las compuertas Or-exclusivas son diferentes a los 2tpd que adoptamos aquí para el cálculo de los retardos.

En el caso de necesitar ampliar la cantidad de bits, se puede realizar la cascada de dos sumadores de 4 bits con lo cual si bien no se conseguirá la misma velocidad que uno de 4 bis, será sensiblemente más veloz que su contraparte tipo “ripple-carry”.

Filmina 21 a 22: El carry-select de N bits está basado en tres sumadores completos de N/2 bits, uno para la parte baja y los otros dos para generar la parte alto de la suma, junto con un MUX 2:1 de longitud de palabra N/2.

La ventaja es que los bits de suma de la parte alta se estabilizan casi al mismo tiempo que los de la parte baja, salvo que hay que sumar el tiempo de respuesta del MUX.

Los sumadores mencionados pueden ser de cualquier tipo.

Filminas 23 a 25: El sumador carry-save tiene una filosofía de suma muy particular, donde se realizan dos operaciones de sumas en paralelo y luego se adicionan los resultados.

En el primer caso se suman los operandos sin considerar sus acarreos por ejemplo hacer 12+19 daría como resultado 21. Por otro lado se consideran los carry solos en la suma; para el ejemplo tendremos como resultado: 10. Posteriormente se suman ambos resultados parciales dando 21+10=31 que es el resultado correcto.

La ventaja sobre el ripple-carry es su menor retardo. En la filmina 25 se tiene una implementación donde se realiza la suma de 3 operandos en el mismo circuito. Con los esquemas anteriores habría que tener dos niveles de sumadores: el primero sumando A+B y el segundo sumando el resultado de ambos con el operando restante, C.

Filmina 26 a 27: Se muestran los circuitos semi-restador y restador completo de un bit.

Para sintetizar un restador de N bits ripple-carry se procede de igual manera que en el caso del sumador, es decir, realizando la cascada de restadores de un bit.

Filmina 28: Este circuito implementa un restador en complemento a 2.

Recordar que A-B es lo mismo que hacer A + “el complemento de B”, por lo que se puede en este caso complementar en CA1 a B y luego sumar un “1”.

La solución para esto es sencilla ya que complementar en CA1 es igual que invertir todos los bits, es decir, podemos usar compuertas Or-exclusiva en cada bit de B para invertir esos valores.

Para ello hay que poner una de las entradas a la Or-exclusiva en “1”.

Si podemos unir todas esas entradas a una línea en común podríamos diseñar un sumador ó restador con esa línea de control. Hay que considerar que en ese caso también hay que poner una Or-exclusiva a la entrada del carry0 del sumador ya que para el sumador debe estar en “0” dicha entrada.

Filmina 29: Implementar multiplicadores y divisores de un número sin signo por otro que sea potencia de 2, es sumamente fácil ya que sólo hay que correr la coma para un lado u otro.

Existe al menos dos formas de implementarlo: una es empleando un registro de desplazamiento de carga paralelo y salida serie y luego aplicar tantos ciclos de reloj para desplazar los bits a derecha (para dividir) o a izquierda (para multiplicar) agregando “ceros” en los extremos.

Otra forma es empleando estructuras basadas en barrel-shifter con desplazamiento aritmético.

Filminas 30 a 31: Un multiplicador basado en el algoritmo convencional, realiza productos parciales y luego los suma considerando los pesos de cada uno, es decir, con los corrimientos en las posiciones que correspondan (el producto parcial menos significativo no tiene corrimiento, el siguiente debe estar corrido un lugar a izquierda y así siguiendo).

Una opción más eficiente en cuanto a velocidad es el multiplicador que se basa en el algoritmo de Booth.

Su implementación emplea multiplexores y sumadores lo que lo hace interesante en estructuras típicas de circuitos lógicos programables tales como las FPGA (Field Programmable Gate Array) que son muy “ricos” en multiplexores.

Filmina 32 a 33: El algebra de Booth aquí se explica para el caso de multiplicar dos números sin signo de 4 bits cada uno X con sus bits [X3 X2 X1 X0] e Y con sus bits [Y3 Y2 Y1 Y0].

Se trabaja aquí con Y dividido en parte alta [Y3 Y2] y parte baja [Y1 Y0].

Se realizan dos productos parciales: uno es [X3 X2 X1 X0] por [Y3 Y2] y por otro lado

[X3 X2 X1 X0] por [Y3 Y2 0 0], éste último con dos lugares corridos a la izquierda a fin de respetar el peso que tiene en la representación.

Para realizar estas multiplicaciones lo que se hace en realidad es plantear los diferentes valores que puede tener Y. Por ejemplo para [Y1 Y0], los valores posibles son: 0, 1, 2 ó 3.

Para [Y3 Y2 0 0] los valores posibles también son los mismos.

Entonces se puede sintetizar para esta parte del multiplicador, un circuito que dé a la salida un valor tal que si X se multiplica por [Y1 Y0]=0 dé 0, si es por [Y1 Y0]=1 será X, si es por [Y1 Y0]=2, 2X ó [Y1 Y0]=3, 3X. Esto aquí se logra con un MUX 4:1 con entradas de 0, X, 2X y 3X y salida de 6 bits, entrando con [Y1 Y0] en los bits de selección del MUX.

Para lograr que esto funcione, obviamente debe de generarse antes las funciones de 2X y 3X por si se necesiten. Eso se puede lograr empleando barrel-shifter y MUXs como se verá más adelante en esta clase.

El mismo razonamiento se utiliza para generar el producto parcial entre [X3 X2 X1 X0] y

[Y3 Y2 0 0].

El resultado definitivo de la multiplicación se obtiene sumando ambos productos parciales.

Filmina 34 a 35: Empleando barrel-shifter aritméticos es posible implementar multiplicaciones entre un número y otro potencia de 2.

Para el caso de un número arbitrario de la forma (m+1)N se puede utilizarse además un sumador. Se puede agregar más flexibilidad empleando más de un barrel-shifter .

Filmina 36: Este circuito es el que podría utilizarse en el ejemplo visto del multiplicador con algoritmo de Booth, donde se puede programar la multiplicación de una variable por 0, 1, 2 ó 3.

Filmina 37 a 38: El chip presentado es un circuito integrado (bastante antiguo) que cumple la función de unidad aritmético-lógica (ALU abreviado en inglés), es decir, por programación de ciertas entradas de control se pueden realizar operaciones aritméticas o lógicas entre dos variables de entrada de 4 bits cada una.

NOTA: En el módulo de "lógica programable" se verá un ejemplo de diseño de una ALU mucho más versátil que este ejemplo.

Filminas 39 a 40: Como ejemplos de circuitos aritméticos serie, presentamos el caso de un sumador de N bits y un circuito complementador según el sistema CA2 del tipo serie.

El sumador posee sólo un sumador de un bit y un FF tipo "D" para almacenar el carry de entrada en la nueva iteración de suma. Cada ciclo de reloj que se aplique al circuito se carga un nuevo valor de bit de suma en el RD (RD: Registro de Desplazamiento) y se le suma a los operandos Ai Bi el carry anteriormente calculado en la etapa i-1.

De esta manera el circuito sirve para realizar la suma entre dos operandos sin signo sin importar la cantidad de bits que tengan siempre que alcance la longitud de los RDs necesarios.

A mayor cantidad de bits, mayor será la cantidad de ciclos de reloj necesarios para completar la suma.

La ventaja de este circuito es la simplicidad circuital, necesitándose menos recursos de lógica aunque el precio a pagar es la velocidad de respuesta.

El circuito complementador funciona como copiador del dato de entrada. El primer bit que entre debe ser el LSB y así siguiendo hasta el MSB. Hasta que no se detecte un "1" en la entrada el circuito copia. Si se detecta un "1", éste se copia pero todos los bits posteriores serán negados hasta completar los N bits a complementar. Esto sucede porque al recibir un "1" la OR, quedará su salida también en "1" y obligará al FF inferior a copiar permanentemente un "1" en su salida, la cual se inyecta a una de las patas de la Or-exclusiva la cual tendrá el efecto de negar siempre el dato de entrada. Eso seguirá así hasta que se haga el reset de los FFs.