UNIVERSIDAD NACIONAL DE LA PLATA

E1212 - TÉCNICAS DIGITALES

TRABAJO PRÁCTICO Nº 5

FACULTAD DE INGENIERÍA

Circuitos Aritméticos

Ejercicio 1

Se requiere diseñar un sumador/restador de tipo Ripple Carry sin signo para dos operandos de 4 bits que cuente con una línea de control que permita seleccionar la operación a realizar.

1. Deducir las funciones lógicas que intervienen en las etapas del sumador. Repetir para el restador.
2. Realizar el diagrama esquemático de alto nivel y calcular el camino crítico teniendo en cuenta los siguientes parámetros. ¿Qué ocurre con el camino crítico si se aumenta la cantidad de bits a 8? ¿Y a 16?

$$t_{pd(XOR)} = 2\tau \qquad t_{pd(AND,NOT,OR)} = \tau$$

3. Codificar el sumador en VHDL parametrizando el diseño utilizando *componentes* y mediante el uso de las sentencias *generic* y *for generate* y simularlo para 8 bits. Analizar el esquema RTL.

Ejercicio 2

Dado el circuito integrado 74HC283 que implementa un sumador Look-ahead Carry:

1. Analizar el circuito esquemático identificando las funciones *generate* y *propagate* y verificar cualitativamente cómo funciona la predicción del carry.
2. Hallar la máxima frecuencia a la cual puede operar el sumador en los siguientes casos:
   2.1 No hay carry en la entrada del sumador.
   2.2 Hay carry en la entrada del sumador.
3. Si se quiere implementar un sumador de 16 bits utilizando estos circuitos integrados, indicar cómo los interconectaría y calcular la frecuencia máxima de operación en dicho caso.
4. Si se quiere implementar un sumador de 2 bits y otro de 1 bit totalmente independientes utilizando un único circuito integrado, indicar cómo se puede utilizar el chip para ello y cómo funciona la arquitectura.
5. Comparado con una arquitectura Ripple Carry, ¿cuáles son las ventajas y desventajas de utilizar una u otra?

Ejercicio 3

Dado un sumador de tipo Carry Select de 16 bits.

1. Realizar un esquema de alto nivel del sumador indicando claramente la cantidad de bits involucrados en cada etapa para los siguientes casos:
   1.1 Dividiendo los operandos en 2 suboperandos de 8 bits cada uno.
   1.2 Dividiendo los operandos en 4 suboperandos de 4 bits cada uno.
2. Utilizando como base el código del Ejercicio 1 realizar la implementación en VHDL del caso 1.1. Analizar el esquema RTL y verificar la arquitectura.

3. Comparando con otros tipos de sumadores, ¿Qué ventajas y desventajas tiene este tipo de implementación?
4. ¿Cuál sería la implementación óptima en el inciso 1 considerando las siguientes especificaciones?

$t_{pd(FA-Ci)} = 2\tau$ $t_{pd(FA-Si)} = 4\tau$ $t_{pd(MUX)} = 2\tau$

Ejercicio 4

Dado un sumador serie sincrónico de 4 bits sin signo:

1. Realizar el diagrama esquemático y describir cómo funciona.
2. Calcular la frecuencia máxima de operación considerando los siguientes parámetros.

   t_{setup}= 3ns t_{hold}= 0ns $t_{CLK->Q}$= 2ns $t_{pd(COMP)}$= 3ns $t_{pd(MUX)}$= 7ns $t_{pd(NOT)}$= 1ns
3. Implementar el diseño en VHDL y simularlo en Modelsim para verificar su funcionamiento.
4. Extender la implementación del inciso anterior a n bits utilizando la sentencia *generic.*

Ejercicio 5

Si se requiere multiplicar un número entero sin signo de 4 bits por un valor constante en el rango de 0 a 15.

1. Analizar la arquitectura más adecuada del multiplicador y realizar el diagrama esquemático genérico de alto nivel indicando claramente la cantidad de bits en la salida y las etapas intermedias.
2. Utilizar la arquitectura anterior para el caso en que el valor constante sea igual a 6.
3. Repetir el inciso anterior para el valor constante igual a 13.
4. Analizar qué ocurre si el número a multiplicar fuese con signo codificado en CA2. Describir qué debería cambiar en la arquitectura para realizar operaciones de este tipo.

Ejercicio 6

Se requiere implementar una unidad aritmética que realice la siguiente operación donde los operandos X e Y son de 5 y 3 bits sin signo respectivamente.

$$R = \frac{5X - Y}{(-4)}$$

1. Diseñar la lógica necesaria y realizar un diagrama de alto nivel indicando la cantidad de bits con que se realiza cada operación.
2. Utilizando varios chips de la unidad Aritmético Lógica integrada SN74AS181 describir cómo implementar el diagrama del inciso 1.

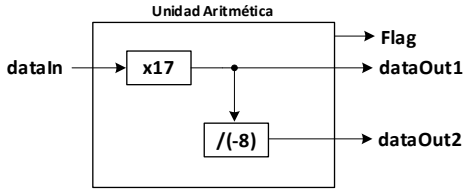
Ejercicio 7

Diseñar la lógica necesaria para implementar un multiplicador sin signo para operandos de 4 bits basado en el algoritmo de Booth.

1. Realizar un diagrama en bloques de alto nivel indicando la cantidad de bits involucrados en cada operación.
2. Verificar el diseño para la operación A=8 y B=13.

Ejercicio 8

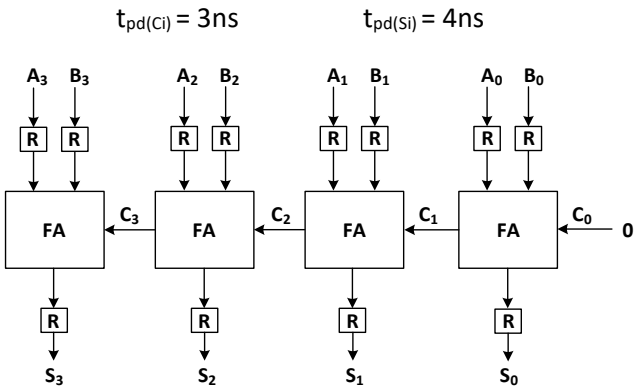
Se requiere implementar en VHDL una unidad aritmética que realice las operaciones que se muestran en el diagrama en bloques adjunto. La unidad recibe un número *dataIn* entero sin signo de 6 bits que se lo multiplica por 17 y cuyo resultado luego se divide por (-8). Posee dos salidas de datos *dataOut1* (que naturalmente será sin signo) y *dataOut2* que estará codificada en CA2. Por otro lado, cuenta con un *Flag* de salida que se debe poner en alto si el resultado *dataOut1* es mayor que 63.



1. Diseñar la lógica necesaria y realizar un diagrama de alto nivel indicando la cantidad de bits con que se realiza cada operación.
2. Implementar el diseño en VHDL, sintetizarlo en Quartus II y verificar el diagrama RTL.
3. Simular la implementación en Modelsim para diferentes valores de *dataIn*.

Ejercicio 9

Dado el siguiente sumador Ripple Carry con las siguientes especificaciones, donde los bloques R son registros tipo D:

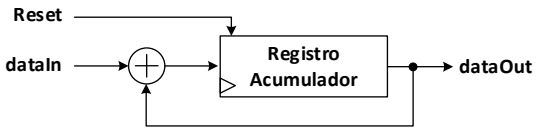


1. Analizar y calcular el camino crítico de la implementación.
2. Si los datos de entrada A y B ingresan a una tasa de 100MHz que coincide con la frecuencia de reloj de los registros, justificar por qué el sumador no funcionará correctamente. ¿Qué parte funcionará bien y qué parte lo hará de forma incorrecta?
3. ¿Cuál sería la máxima frecuencia de operación del sumador en ese caso?

4. La solución natural al problema del inciso 1 es acortar el camino crítico. Analizar posibles modificaciones con ese objetivo que permitan el correcto funcionamiento del sumador a una frecuencia de 100MHz. ¿Qué ventajas y desventajas tienen este tipo de soluciones?
5. Con las modificaciones propuestas calcular nuevamente la máxima frecuencia de operación.

Ejercicio 10

Un Oscilador Controlado Numéricamente (o NCO) permite generar una señal digital cuya frecuencia puede configurarse. Esto permite, agregando un decodificador, generar digitalmente cualquier forma de onda periódica. La estructura básica de un NCO consta de un sumador y un registro que hace las veces de acumulador como se muestra en el siguiente diagrama.



1. Aprovechando el código generado en el Ejercicio 1 diseñar en VHDL el NCO considerando 16 bits para *dataIn* y para el registro acumulador.
2. Simular la implementación en Modelsim para una frecuencia de reloj de 50MHz y los siguientes valores de *dataIn*:

   2.1 dataIn= “3333” (en hexa)

   2.2 dataIn= “00F7” (en hexa)

   2.3 dataIn= “051F” (en hexa)

   En Modelsim configurar el *radix* de *dataOut* como *unsigned* y el *format* como *analog(automatic).* Observar la forma de onda generada.
3. Responder a las siguientes cuestiones:

   3.1 ¿Cómo calcularía la frecuencia de *dataOut* en función del valor de *dataIn*?

   3.2 ¿Cuál es la mínima frecuencia que puede generar? ¿Y la máxima?

   3.3 ¿Qué sucede cuando la frecuencia a generar se acerca a la frecuencia del reloj?
4. Agregar el decodificador como se muestra en el siguiente esquema donde la entrada al mismo son los 4 bits más significativos del registro acumulador. Simular nuevamente en Modelsim para algunos de los valores de *dataIn* del inciso 2 configurando el *radix* y el *format* como *signed* y *analog(automatic)* respectivamente.

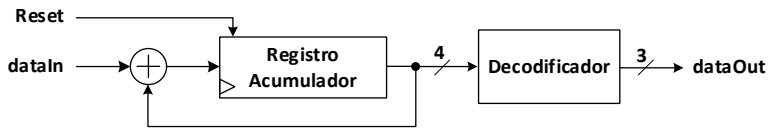


Tabla de verdad del decodificador			
Entrada	**Salida**	**Entrada**	**Salida**
0000	001	1000	110
0001	010	1001	101
0010	011	1010	100
0011	011	1011	100
0100	011	1100	100
0101	010	1101	101
0110	001	1110	110
0111	111	1111	111

5. ¿Qué forma de onda se observa? ¿Cuál es la resolución de fase de la forma de onda generada? ¿Qué debería hacer si se quisiera aumentar dicha resolución?
6. Modificar el diseño para generar una forma de onda que responda a $\cos^2()$ y configurar el NCO para generar dicha forma de onda con una frecuencia de 78KHz. Simular con Modelsim y verificar el comportamiento de *dataOut.*