Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (I/2012)

Solución Interrogación 1

Pregunta 1

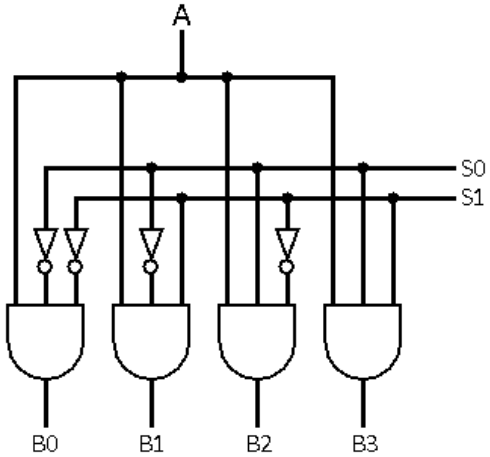
a) ¿Qué se puede decir en relación a las operaciones de suma y multiplicación en los números de punto flotante, con respecto a la pérdida de precisión? **(1 pto.)**

Solución: La suma, debido al proceso de equilibrio de exponentes, genera una mayor pérdida de precisión que la multiplicación. Luego, es preferible primero multiplicar y luego sumar.

b) ¿Que cosas se debe conocer de un arreglo para extraer e interpretar sus datos correctamente de una memoria?**(1 pto.)**

Solución: Se debe conocer la dirección de inicio del arreglo, su largo, el tipo de dato de sus elementos y su endiannes.

c) Diseñe un De-Multiplexor con bus de datos de 1 bit y bus de control de 2 bits.**(1 pto.)**
Solución:



d) ¿Que permite que un flip-flop D funcione en un flanco y no en un estado? **(1 pto.)**

Solución: Esto lo permite la conexión de 2 latches D en serie, donde la señal de control de uno es la negación de la del otro.

e) ¿Cuantos conectivos ternarios distintos se pueden definir en la lógica booleana?**(1 pto.)**

Solución: La tabla de verdad de cada conectivo binario tendrá $2^3 = 8$ entradas y cada entrada puede tomar 2 valores (lógica booleana). Luego, existen $2^8 = 256$ conectivos ternarios.

f) ¿Que consideración tiene el estándar IEEE754 con respecto al significante normalizado sin 1 y al número 0? **(1 pto.)**

Solución: Dado que el significante siempre contiene un 1, el estándar define al número 0 como el número que tiene sólo ceros en el exponente y el significante. El bit de signo del significante es libre, por lo que existe el +0 y el -0.

Pregunta 2

Dada una lógica ternaria con valores V, F y ?, y las siguientes tablas de verdad para los conectivos $\wedge$, $\vee$ y $\neg$:

A $\wedge$ B	V	F	?
V	V	F	?
F	F	F	F
?	?	F	?

A$\vee$B	V	F	?
V	V	V	V
F	V	F	?
?	V	?	?

A	$\neg$A
V	F
F	V
?	?

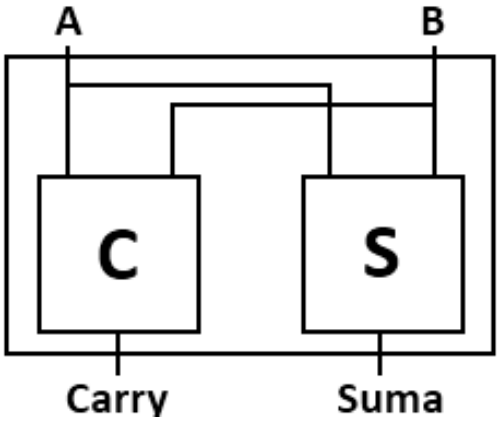
Conteste las siguientes preguntas:

a) Asigne los símbolos de la lógica ternaria antes descrita a los números 0, 1 y 2 y diseñe un full-adder que sume dos números de un trit (ternary digit). Puede utilizar los conectivos binarios de trits que estime conveniente para diseñar las compuertas. **(4 ptos.)**

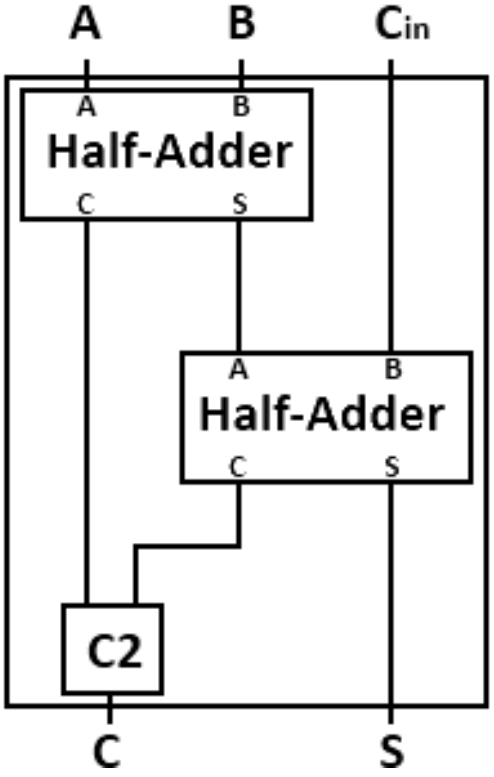
Solución: La asignación de símbolos a los números puede ser cualquiera, siempre y cuando las tablas de verdad usadas respeten esta asignación. En este caso, se utilizó la siguiente asignación: $? = 0$, $V = 1$, $F = 2$. Para construir un full-adder de trits, el primero paso consiste en construir un half-adder y luego, usando dos half-adders construir un full adder. Usando la asignación de símbolos a números anterior, la tabla de verdad para las salidas de carry (**C**) y suma (**S**) de un half-adder, dados trits en las entradas **A** y **B**, es la siguiente:

A	B	C	S
0	0	0	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	0	2
1	2	1	0
2	0	0	2
2	1	1	0
2	2	1	1

Luego, si creamos una compuerta de lógica ternaria para la columna **C** y otra para la columna **S**, el diagrama del circuito de un half-adder es el siguiente:



A continuación, para construir el full-adder basta con observar que el carry de un half adder toma como valor máximo un 1 y que además, al agregar la señal de carry-in a un full-adder, el resultado máximo de esta suma sigue necesitando 2 trits. Luego, podemos diseñar el full-adder ternario usando un esquema similar a un full-adder binario:



donde la compuerta **C2** tiene la siguiente tabla de verdad:

A	B	C2
0	0	0
0	1	1
1	0	1

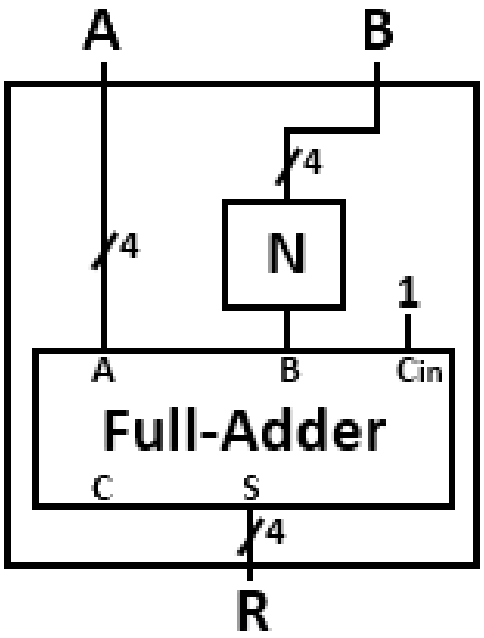
Cabe destacar que dado que nunca los carry que entegan los half-adders serán 1 al mismo tiempo, sólo es necesario definir el valor de **C2** para las tres combinaciones descritas en la tabla.

b) Utilizando lo desarrollado en el item anterior, diseñe un restador de dos números de cuatro trits cada uno. **(2 ptos.)**

Solución: Para construir un restador de trits utilizando un esquema similar al visto en clases para el restador de bits, es necesario que este funcione en complemento a 3, que es el análogo al complemento a 2 pero en ternario. Para esto, es necesario definir una compuerta que entregue el complemento del sustraendo (B), tal como el negador entrega el complemento de un número binario. La tabla para esta compuerta es la siguiente:

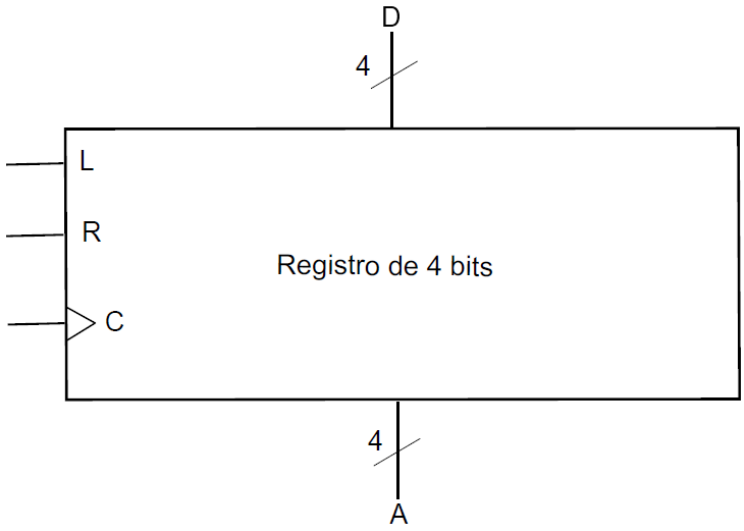
A	N
0	2
1	1
2	0

Teniendo definida la compuerta N, el diagrama para el restador de trits es el siguiente:

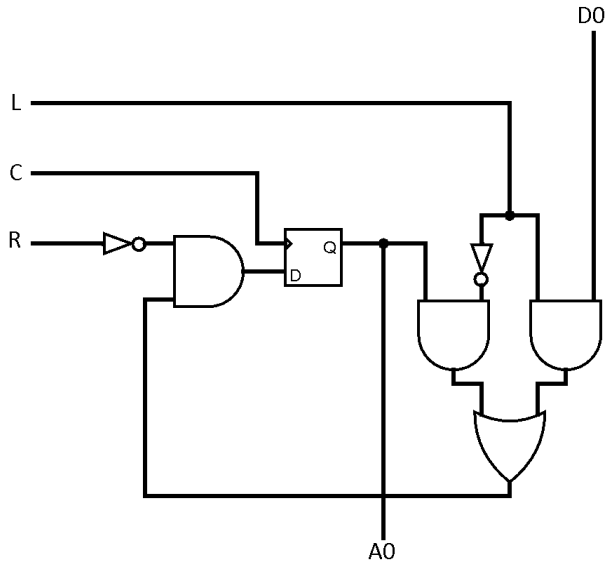


Pregunta 3

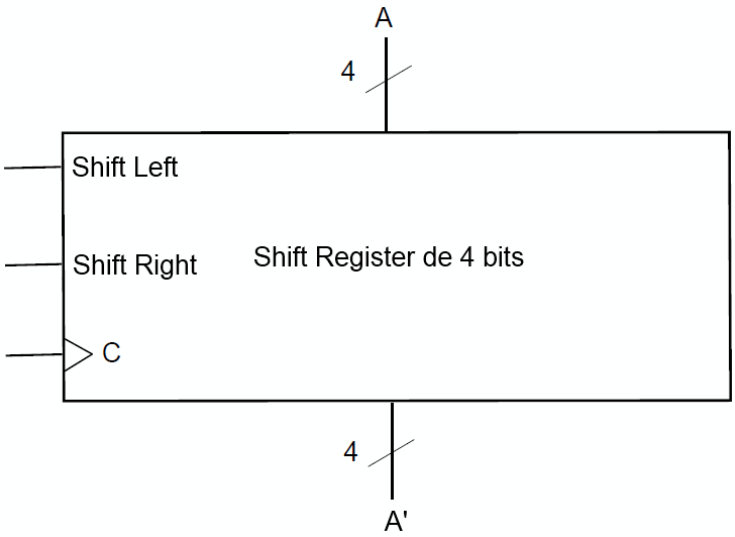
a) Implemente mediante compuertas lógicas y flip-flops tipo D, el registro de la figura, con señales de control (C), carga (Load) y reset (Reset), que funciona con flanco de subida. **(3 ptos.)**



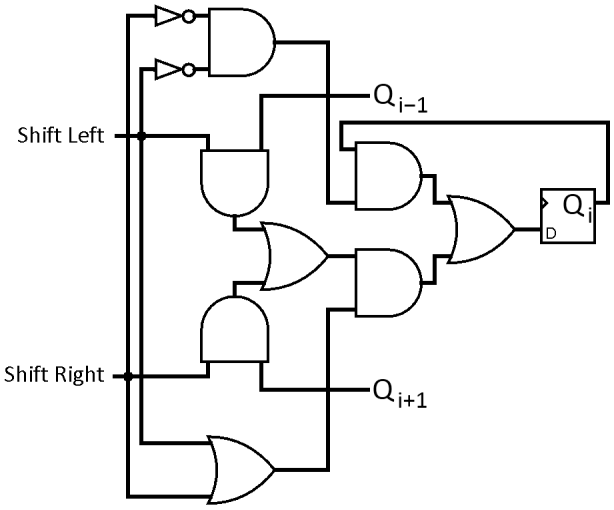
Solución: Por motivos de espacio, se presenta la solución para el bit menos significativo. Para el resto de los bits el diagrama es el mismo.



b) Implemente mediante compuertas lógicas y flip-flops tipo D, el shift-register de la figura. Este registro realiza shifting lógico con su contenido, de acuerdo a las señales Shift Left y Shift Right, cuando la señal C se encuentra en flanco de subida. **(3 ptos.)**



Solución: Por motivos de espacio, se presenta la solución para el bit i. Cuando $i = 0$ (bit menos significativo), $Q_{i-1} = Q_3$, mientras que si $i = 3$ (bit más significativo, $Q_{i+1} = Q_0$.



Pregunta 4

Dada la siguiente función, donde `var` representa un tipo básico desconocido:

```
var func(var a, var b)
{
    var c = 0, d = 0;
    var n = sizeof(var)*8;
    for (var i = n-1; i >= 0; i--)
    {
        d = d << 1;
        d(0) = a(i);
        if (d >= b)
        {
            c = c - b;
            c(i) = 1;
        }
    }
    return c;
}
```

Donde `x`(i) entrega el bit de `x` que está en la posición i. Suponga además que se tiene una memoria de 10 entradas, con palabras de 1 byte con los valores que se muestran a continuación:

Dirección	Palabra
0	$0x0A$
1	$0x00$
2	$0x01$
3	$0x00$
4	$0x12$
5	$0x05$
6	$0x10$
7	$0x00$
8	$0x00$
9	$0x06$

Calcule el valor de retorno de la función para los siguientes casos:

a) Las variables `a` y `b` son de tipo `byte`, `a` está almacenada a partir de la dirección 0 y `b` está almacenada a partir de la dirección 5. **(2 ptos.)**

b) Las variables `a` y `b` son de tipo `unsigned short` (entero sin signo de 2 bytes), `a` está almacenada a partir de la dirección 3 y `b` está almacenada a partir de la dirección 8. Ambas variables ocupan orden big endian. **(2 ptos.)**

c) Las variables `a` y `b` son de tipo `unsigned short`, `a` está almacenada a partir de la dirección 1 y `b` está almacenada a partir de la dirección 6. Ambas variables ocupan orden little endian. **(2 ptos.)**

Solución: El algoritmo copia el valor de la variable a en la variable d secuencialmente, desde los bits más significativos hasta los menos, mediante shifts a la izquierda de una posición. Esto significa que pone un bit en la posición 0 de d y luego lo mueve hacia la izquierda con shifts. El valor de retorno c se modifica sólo a partir del momento en que d es mayor que b. De ahí en adelante, por

cada iteración se le resta a c el valor de b. Como c parte en 0 y en los tres casos no tiene signo, esto es análogo a que c tome el valor $2^8 = 256$ cuando es del tipo *byte* y que tome el valor $2^16 = 65536$ cuando es *unsignedshort*. Además, junto con restar b, se pone en 1 el bit i de c.

a) $a = 10$ y $b = 5$, valor de retorno 247.

b) $a = 18$ y $b = 6$, valor de retorno 65525.

c) $a = 256$ y $b = 16$, valor de retorno 65471.