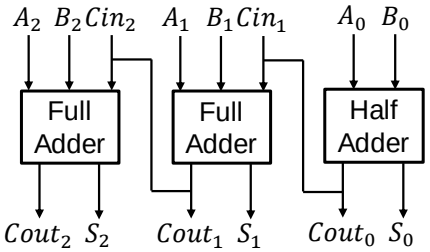
Electrónica Digital 2
Programa de Ingeniería Electrónica
Universidad de Antioquia

Examen: tema 1 – bloques de diseño digital (15%)
Fecha: miércoles 20 de marzo de 2024
Duración: una hora y cuarenta y cinco minutos (1h:45m)

NOMBRE: **SOY LA SOLUCIÓN**

EJERCICIOS PARA RESOLVER

1) (20p) Para el sumador de 3-bits con propagación de acarreo que se muestra a continuación indique, para cada fila de la tabla, los valores de las salidas conforme se propagan en el tiempo (C_{outx} y S_x), dados los valores A = 3'b011 y B = 3'b001. El tiempo de propagación de las señales ($delay$) es de $12ns$. Asuma que las salidas arrancan en 1'b0. ¿En qué tiempo la respuesta será correcta? **36** (ns)



	C_{out2}	S_2	C_{out1}	S_1	C_{out0}	S_0
$t_1 = delay$	0	0	0	1	1	0
$t_2 = delay * 2$	0	0	1	0	1	0
$t_3 = delay * 3$	0	1	1	0	1	0
$t_4 = delay * 4$	0	1	1	0	1	0

2) (20p) Defina una memoria RAM de 16Mbytes con bus de datos independiente para lectura (rd) y escritura (wd), línea de habilitación de escritura (we) y 64 bits por palabra, empleando el lenguaje SystemVerilog:

```
module ram (clk, we, addr, rd, wd);
    // (3p) Definición del bus de direcciones (addr)
    input logic [20:0] addr;
    // (2p) Definición del bus de datos (rd y wd)
    output logic [63:0] rd;
    input logic [63:0] wd;
    // (2p) Definición de las señales de write-enable (we) y reloj (clk)
    input logic clk, we;
    // (4p) Definición del arreglo (mRAM)
    logic [63:0] mRAM [0:2**21-1];
    always_ff @(posedge clk) begin
    // (6p) Escritura en mRAM únicamente cuando we = 1'b0
        if (!we)
            mRAM[addr] <= wd;
    end
    // (3p) Lectura síncrona (con relación al reloj) de mRAM
    always_ff @(posedge clk)
        rd <= mRAM[addr];
endmodule
```

3) (10p) Escriba una línea de código en SystemVerilog que lleve el complemento a dos de la señal {1'b0, **value**} a la señal **result**, si la línea llamada **negative** es igual a 1'b1; en caso contrario, lleve el contenido de la señal {1'b0, **value**} a **result**. Las señales fueron definidas así: **logic** [5:0] result y **logic** [4:0] value. Asuma que la línea de código estará dentro de un bloque **always_comb**.

Respuesta: **result** = (negative) ? (~{1'b0, **value**} + 1'b1) : {1'b0, **value**};

4) (30p) Un nuevo formato para representar números de punto flotante empleando 12-bits ha sido generado. En este nuevo formato se reservó un bit para el signo, 4 bits para el exponente y 7 bits para la fracción de la mantisa normalizada como se muestra a continuación:

Representación de punto flotante (12-bits)

S	EXPONENTE	FRACCIÓN
1b	*4b*	*7b*

Las siguientes son las características del formato:

a. Exponente sesgado. Rango del exponente sesgado: $[1, 2^4 - 2]$.
b. El valor en decimal se representa como: $Dec = (-1)^S \; x \; 1.fracción \; x \; 2^{exponente - SESGO}$.
c. La mantisa está normalizada. En el formato se guarda la fracción en 7 bits.
d. Representación de los valores +/- cero: S (signo) = 0/1, exponente = 0, fracción = 0.
e. Representación de los casos +/- infinito: S (signo) = 0/1, exponente = $2^4 - 1$, fracción = 0.
f. Representación del caso NaN: S (signo) = 0, exponente = $2^4 - 1$, fracción ≠ 0.

Sabiendo lo anterior, responda las siguientes preguntas dados los valores **A = 0xCC5** y **B = 0x4CF** en representación de punto flotante de 12-bits.

a) (4p) Indique los exponentes de cada operando sin sesgo (dec): ExpA = **+2**, ExpB = **+2**
b) (4p) Indique las mantisas de cada operando (binario): MantA = **1.100_0101**, MantB = **1.100_1111**
c) (2p) Indique cada operando en valor decimal: DecA = **-6.15625,** DecB = **+6.46875**

Realice la operación **R = A + B**, mostrando el procedimiento, e indique el valor **R** en hexadecimal:

d) (6p) Valor R en Hexa = **0x2A0**
e) (14p) Procedimiento operación **R = A + B**:

1) $Exp_A = 1001 = 2, Exp_B = 1001 = 2, Mant_A = 1.1000101, Mant_B = 1.1001111$

2) Ambos exponentes iguales, no se necesita desplazar alguna de las mantisas

3) Sumar mantisas:

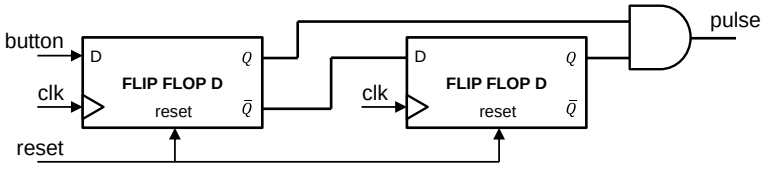
$$\begin{matrix} 1.1000101 \\ 1.1001111 \end{matrix} \rightarrow \begin{matrix} 001.1000101(+) \\ 001.1001111(+) \end{matrix} \rightarrow \begin{matrix} 110.0111011(-) \\ \overline{001.1001111(+)} \\ 000.0001010(+) \end{matrix}$$

4) Normalizar mantisa resultante y ajustar exponente:

$$000.0001010 \rightarrow 1.0100000, Exp_R = 2 - 4 = -2$$

$$\text{Hex:} 0\_0101\_010\_0000 = 0x2A0$$

5) (20p) En algunas aplicaciones es necesario generar un pulso de duración un ciclo de reloj (**pulse**), en el momento en que se active la correspondiente señal de entrada (**button**), sin importar si ésta se queda activa por más de un ciclo de reloj. El circuito dado de manera esquemática realiza la mencionada función. Reprodúzcalo empleando SystemVerilog:



```
module pulseGenerator (input logic clk, reset, button, output logic pulse);
      logic q1, q2;

      always_ff @(posedge clk, posedge reset) begin
        if (reset) begin
          q1 <= 0;
          q2 <= 0;
        end else begin
          q1 <= d;
          q2 <= ~q1;
        end
      end
      assign pulse = q1 & q2;
endmodule
```