Electrónica Digital 2
Programa de Ingeniería Electrónica
Universidad de Antioquia

Examen: tema 1^A – bloques de diseño digital (15%)
Fecha: viernes 15 de septiembre de 2023
Duración: una hora y cuarenta y cinco minutos (1h:45m)

NOMBRE: **Solución**

EJERCICIOS PARA RESOLVER

1) (16%) La operación R=A-B de un sistema de cómputo, así como las entradas y salidas, han sido descritas usando lenguaje SystemVerilog de la siguiente manera:

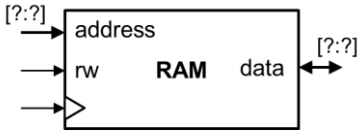
```
input  logic [3:0] A, B;
output logic [3:0] R;
output logic cout;
always_comb
    {cout, R} = {1'b0, A} + ({1'b0, ~B} + 1'b1);
```

Sí a las entradas **A** y **B** se les asigna los valores dados en la tabla de la derecha, indique, en decimal, el valor de la salida R y sí en la operación con los datos dados se presenta condición de *Overflow*:

A	B	R (decimal)	*Overflow*
-2	7	7	Si
-4	-8	4	No
7	-1	-8	Si

2) (16%) Una empresa de la ciudad lo quiere contratar a usted para que genere el código en SystemVerilog de una memoria RAM de **4kbytes** de capacidad, con bus de datos compartido (**data**) de 32-bits. Para probar sus habilidades, se le solicita definir los **puertos de entrada y salida** del módulo RAM, así también como la definición del **arreglo de almacenamiento** para la memoria llamado **mRAM**:

module ram (clk, rw, address, data);
    // Definición del bus de direcciones (address)
    input logic [9:0] address;
    // Definición del bus de datos compartido (data)
    inout logic [31:0] data;
    // Definición de las señales de *read/write* (rw) y reloj (clk)
    input logic clk, rw;
    // Definición del arreglo (mRAM)
    logic [31:0] mRAM [0:210-1];**
    …
endmodule



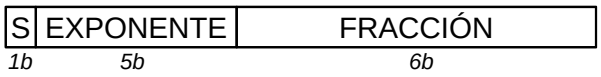
3) (8%) Un grupo de estudiantes ha cometido un error en la definición del tipo de dato **State**, empleada para definir las señales **currentState** y **nextState** de una máquina de 5 estados en SystemVerilog. Indique el error y brinde una solución.

```
typedef enum logic [1:0] {S0, S1, S2, S3, S4} State;
State currentState, nextState;
```

Problema: **Número de bits incorrecto, se requieren 3 bits**
Corrección: **`typedef enum logic [2:0]`**…

4) (30%) 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, 5 bits para el exponente y 6 bits para la fracción de la mantisa normalizada como se muestra a continuación:

Representación de punto flotante (12-bits)



Las siguientes son las características del formato:

a. Exponente sesgado. Rango del exponente sesgado (se suma 15): $[1, 2^5 - 2]$.
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 6 bits.

Sabiendo lo anterior, responda las siguientes preguntas dados los valores **A = 0xCAD** y **B = 0x50D** en representación de punto flotante de 12-bits.

a) (4%) Indique los exponentes de cada operando sin sesgo: ExpA = **3**, ExpB = **5**
b) (5%) Indique las mantisas de cada operando (binario): MantA = **1.101101**, MantB = **1.001101**
c) (2%) Indique cada operando en valor decimal: DecA = **-13.625**, DecB = **38.5**

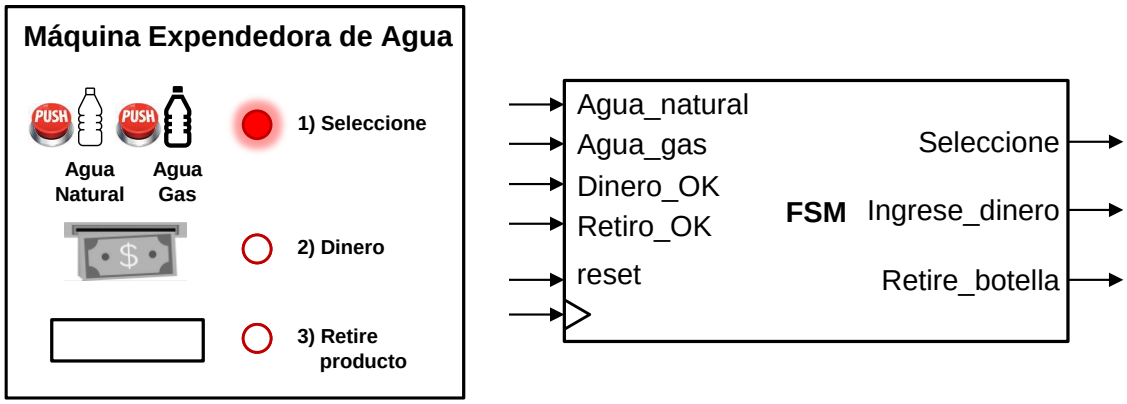
Realice la operación **R = A * B** y responda las siguientes preguntas:

d) (5%) Indique el exponente de R sin sesgo: ExpR = **9**
e) (8%) Indique la mantisa de R (binario): MantR = **1.000001**
f) (2%) Indique el valor decimal de R: DecR = **-520**

Responda las siguientes preguntas en relación con el formato de punto fijo de 12-bits:

g) (2%) En la expresión del nuevo formato de 12-bits: $Dec = \pm mantisa \; x \; 2^{EXP}$, indique el rango de valores para EXP: **[-14, 15]**
h) (2%) Valor máximo que se puede representar en el nuevo formato (decimal): **65024**

5) (30%) Una empresa de la ciudad requiere desarrollar el controlador de la máquina expendedora de botellas de agua natural y con gas que se muestra en el lado izquierdo de la siguiente figura:



La entidad del controlador a desarrollar se muestra en la parte derecha de la figura anterior. Dicho módulo se llama FSM y corresponde a una máquina de estados finitos. Este módulo cuenta con las señales de salida:

Seleccione, **Ingrese_dinero** y **Retire_botella**, y de entrada: **Agua_natural**, **Agua_gas**, **Dinero_OK** y **Retiro_OK**. A continuación, se detalla cómo el módulo FSM deberá generar las señales de salida de acuerdo con las acciones del cliente:

a. Cuando la señal de **reset** es igual a 1'b1, el módulo FSM pondrá la señal **Seleccione** en 1'b1. Las demás salidas del módulo FSM deberán estar a 1'b0.
b. Cuando el cliente seleccione el tipo de botella de agua a comprar, el módulo FSM deberá poner la señal **Ingrese_dinero** en 1'b1. El módulo FSM será notificado del evento mediante el valor en 1'b1 en la señal **Agua_natural** o en la señal **Agua_gas**. Las demás salidas del módulo FSM deberán estar a 1'b0.
c. Cuando el cliente introduzca el valor adecuado de dinero, el módulo FSM deberá poner la señal **Retire_botella** en 1'b1. El módulo FSM será notificado del evento mediante el valor en 1'b1 en la señal **Dinero_OK**. Las demás salidas del módulo FSM deberán estar a 1'b0.
d. Cuando el cliente retire la botella de la máquina, el módulo FSM pondrá nuevamente la señal **Seleccione** en 1'b1, iniciándose una nueva compra. El módulo FSM será notificado del evento mediante el valor en 1'b1 en la señal **Retiro_OK**. Las demás salidas del módulo FSM deberán estar a 1'b0.

A usted se le encarga completar las siguientes porciones de código para tener listo el módulo FSM:

a) (8%) Complete la definición del tipo de dato State con los estados de la FSM y bits requeridos:

```
typedef enum logic [1:0] {S_seleccion, S_dinero, S_retiro} State;
State currentState, nextState;
```

b) (4%) Complete la descripción del proceso de actualización de currentState

```
always_ff @(posedge clk, posedge reset)
    if (reset)
      currentState <= S_seleccion;
    else
      currentState <= nextState;
```

c) (10%) Complete la descripción del proceso de actualización de nextState

```
always_comb begin
    nextState = currentState;
    case (currentState)
      S_seleccion:
        if (Agua_natural || Agua_gas)
          nextState = S_dinero;
      S_dinero:
        if (Dinero_OK)
          nextState = S_retiro;
      S_retiro:
        if (Retiro_OK)
          nextState = S_seleccion;
    endcase
end
```

d) (8%) Complete la descripción del proceso de actualización de las salidas

```
always_comb begin
    Seleccione = 1'b0;
    Ingrese_dinero = 1'b0;
    Retire_botella = 1'b0;
```

```
    case (currentState)
      S_seleccion:
        Seleccione = 1'b1;
      S_dinero:
        Ingrese_dinero = 1'b1;
      S_retiro:
        Retire_botella = 1'b1;
    endcase
end
```