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

IIC2343 – Arquitectura de Computadores (I/2015)

Solución Interrogación 1

Pregunta 1

a) Describa una codificación para números enteros binarios distinta al complemento a 2, que utilize bit de signo y tenga una única representación para el cero. No es necesario que en esta codificación se cumpla que $A + -A = 0$ **(1 pto.)**
Solución: Una posible solución es la codificación ZigZag. Este esquema utiliza los números naturales para codificar de manera alternada los números negativos y positivos, *i.e.*: 0 ⇒ 0, -1 ⇒ 1, 1 ⇒ 2, -2 ⇒ 3, 2 ⇒ 4, etc. De esta manera, los números negativos siempre tendrán un 1 como bit menos significativo y los positivos tendrán un 0, y el cero será representado únicamente como 0.

b) Un reloj digital fue programado para hacer sonar su alarma a los 30 minutos, pero esta sonó a los 7680 minutos. ¿Por qué se produjo este problema? **(1 pto.)**
Solución: El problema ocurrió porque el número fue interpretado en un endianness incorrecto. El número 30 en hexadecimal, usando 2 bytes, se representa como 0x001E. Luego si el número fue leído en un endianness incorrecto, será interpretado como 0x1E00 = 7680

c) ¿Qué consideración hay que tener en una ALU para que esta pueda operar números con y sin signo (no simultáneamente)? **(1 pto.)**
Solución: No hay que tener ninguna consideración, la ALU opera números binarios sin importar si son positivos o negativos.

d) En la ALU vista en clases, el principal factor que limita la velocidad es la propagación del carry. ¿Es posible minimizar este problema? Fundamente su respuesta, o de ejemplos, en ambos casos. **(1 pto.)**
Solución: Para solucionar este problema, es posible utilizar una capa de lógica previa al sumador de ALU, que se encargue de predecir el valor del carry para cada *Full-Adder*, en función de las entradas anteriores. Esta técnica se conoce como carry-lookahead.

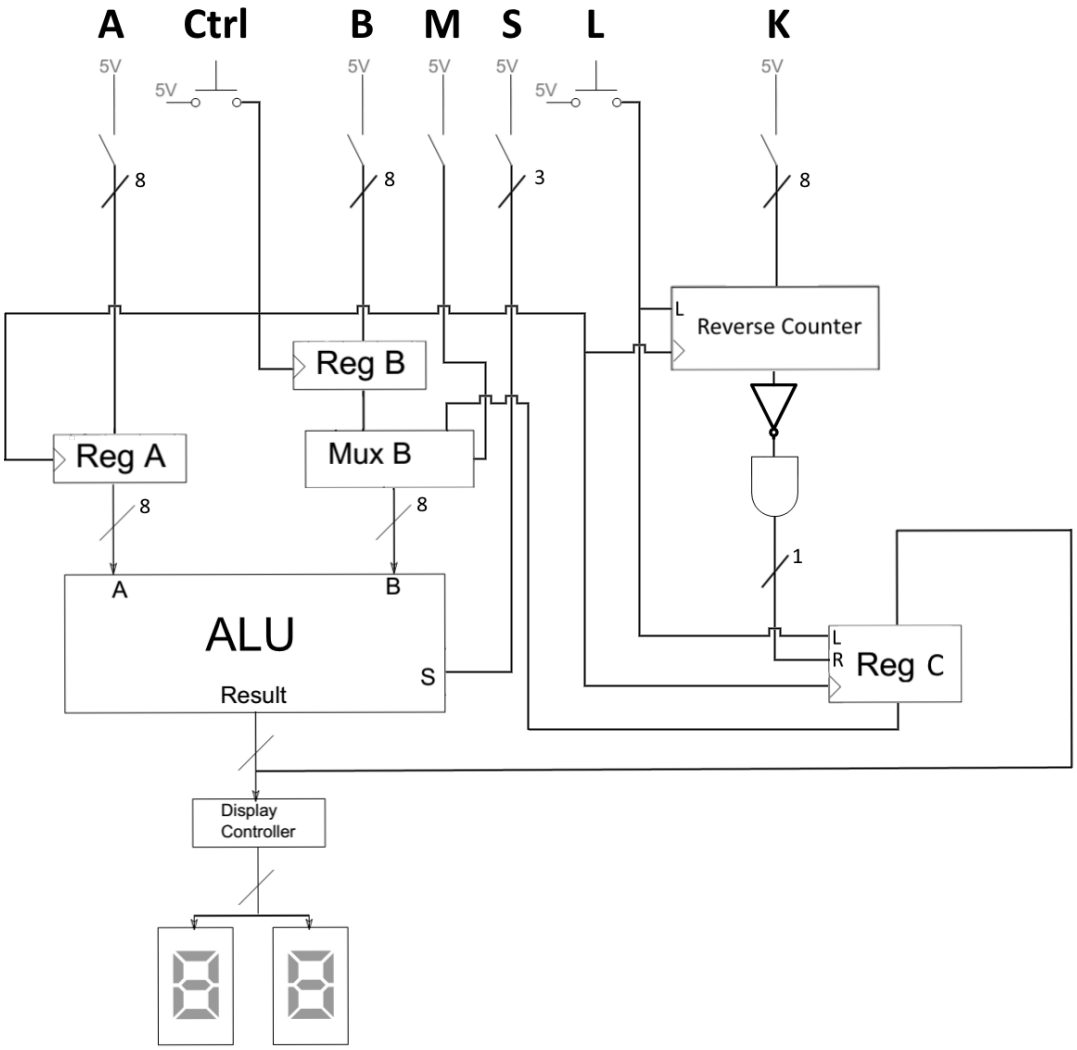
e) ¿Por qué es más lento realizar operaciones sobre números de punto flotante de base decimal, que sobre los de base binaria? **(1 pto.)**
Solución: Un posible motivo es la imposibilidad de realizar los cálculos directamente en binario, debido al uso de otra base, lo que implica, entre otras cosas, que el punto (coma) no puede moverse de manera simple al momento de realizar multiplicaciones o sumas.

f) Si una memoria RAM es construida utilizando los elementos de memoria vistos en clases, ¿a qué tipo de memoria RAM corresponde? **(1 pto.)**
Solución: Corresponde a una SRAM (Static RAM).

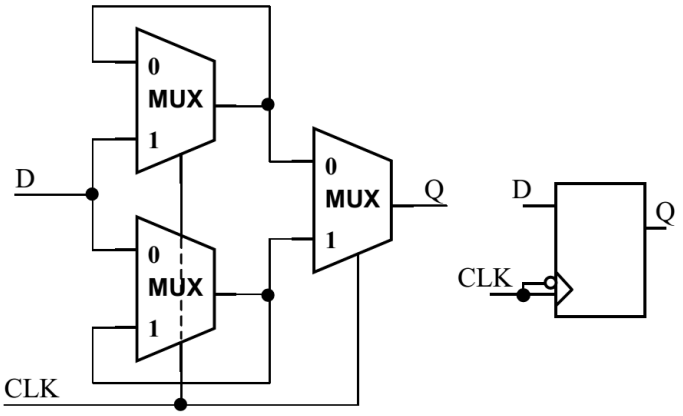
Pregunta 2

a) Diseñe una calculadora de 8 bits y 8 operaciones, que permita visualizar los resultados de las operaciones. Además, esta calculadora debe permitir respaldar el resultado de **una** operación, para ser usado posteriormente. Este resultado almacenado sólo estará disponible durante las K siguientes operaciones, donde K es un número natural definido por el usuario, no mayor a 255. Este resultado almacenado podrá ser sobrescrito por el usuario, durante cualquier de las K operaciones siguientes, en cuyo caso, el proceso comenzará nuevamente de cero. **(3 ptos.)**

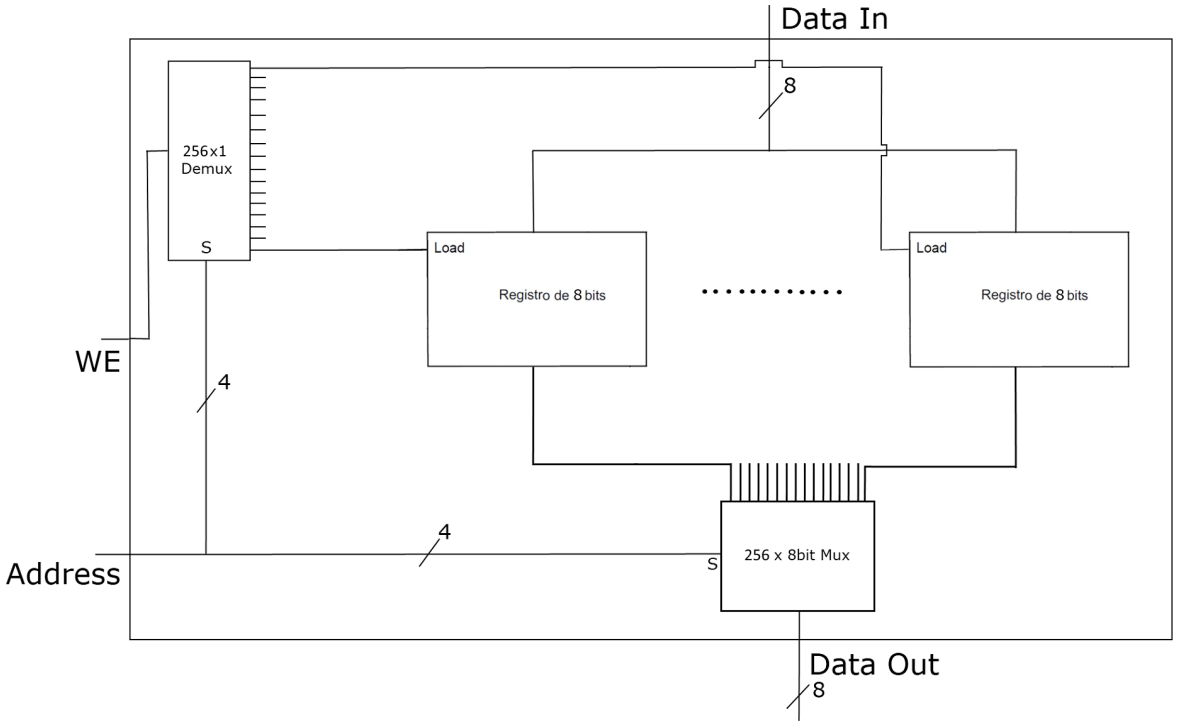
Solución:



b) Diseñe una memoria RAM DDR de 256B. Sólo puede utilizar, como elementos de memoria, flip-flops y registros. **(3 ptos.)**
Solución: Este problema puede dividirse en 2 partes. La primera parte, y más compleja, es diseñar un flip-flop que se active en ambos flancos (2 ptos.). La segunda, y más sencilla, es diseñar una memoria RAM utilizando esos flip-flops (1 pto).
Una posibilidad de diseño para el flip-flop activado en ambos flancos es la siguiente:



En la figura, a la izquierda se muestra la construcción del flip-flop activado en ambos flancos, mientras que a la derecha, el símbolo de este flip-flop. Luego, utilizando este flip-flop, construir un registro que se active en ambos flancos es trivial (igual que para el caso activado en el flanco de subida).
Finalmente, una posible solución para la construcción de la memoria, que utiliza registros activados en ambos flancos, es la siguiente:



Pregunta 3

a) Implemente, utilizando el lenguaje que más le acomode, la clase **Float**, que emula el comportamiento del tipo de datos *float* (IEEE754). Su implementación debe cumplir lo siguiente:

- incluir métodos para sumar, restar y multiplicar.
- un constructor que reciba como entrada un string que describe el número en formato decimal (no notación científica).
- métodos para obtener el signo, el exponente y el significante.
- manejar los casos especiales.

Internamente, la clase **NO** puede utilizar tipos de dato de punto flotante (float, double, etc.). Asuma la existencia de funciones que realizan la transformación entre strings, números binarios, decimales, etc. **(4 ptos.)**

Solución:

```
public class Float
{
  public int sign = 0;
  public int exp = 127;
  public String significant = toBinaryString(0, 23);
  public String decimal;

  // funcion toBinaryString(String number) retorna un string con la
  // representacion binaria del numero racional contenido en number.

  // funcion toBinaryString(int number, int length) transforma un
  // numero entero su representacion binaria de largo length

  public Float(String number)
  {
    this.decimal = number;
    String binaryString;
    if (number.charAt(0) != '-')
    {
      this.sign = 0;
      binaryString = toBinaryString(number);
    }
    else
    {
      this.sign = 1;
      binaryString = toBinaryString(number.substring(1));
    }
    if (binaryString.charAt(0) == 0)
    {
      if (binaryString.length() == 1)
      {
        this.exp = 0;
        this.significant = toBinaryString(0, 23);
        return;
      }
      else
      {
```

```
          int i = 2;
          for (; binaryString.charAt(i) != '1'; i++)
          {
            this.exp--;
          }
          i++;
          for (int j = 0; i < binaryString.length() || i < 25; i++,j++)
          {
            this.significant.charAt(j) = binaryString.chaAt(i);
          }
          return;
        }
      }
      else
      {
        int i = 1;
        for (; binaryString.charAt(i) != '.'; i++)
        {
          this.exp++;
          this.significant.charAt(i-1) = binaryString.charAt(i);
        }
        i++;
        for (int j = 0; i < binaryString.length() || i < 25; i++,j++)
        {
          this.significant.charAt(j) = binaryString.chaAt(i);
        }
        return;
      }
    }
    public Float add(Float f)
    {
      return new Float(addFloat(this.decimal, f.decimal));
    }
    public Float sub(Float f)
    {
      return new Float(subFloat(this.decimal, f.decimal));
    }
    public Float mul(Float f)
    {
      return new Float(mulFloat(this.decimal, f.decimal));
    }
    public String getSign()
    {
      return String.valueOf(this.sign);
    }
    public String getExp()
    {
      return toBinaryString(this.exp, 8);
    }
    public String getSignificant()
    {
      return this.significant;
    }
  }
```

b) Describa un procedimiento para realizar la multiplicación de dos números naturales binarios de 8 bits, usando sólo las operaciones disponibles en la ALU vista en clases. **(2 ptos.)**
Solución: Asumiendo que *Producto* = *Multiplicando* * *Multiplicador*, el procedimiento se resume de la siguiente manera:

a) Inicialmente se tiene almacenado el multiplicando y el multiplicador y se tiene espacio reservado para almacenar el producto, inicializado en cero.

b) Verificar si bit menos significativo del multiplicador es 1.

- Si es 1, sumar el multiplicando al producto.

c) Hacer shift-left del multiplicando.

d) Hacer shift-right del multiplicador.

e) Verificar si este proceso se ha repetido 8 veces

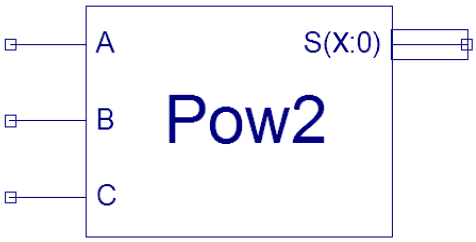
- Si no se ha repetido 8 veces, volver al paso *b)*.
- Si se ha repetido las 8 veces, se termina el proceso.

Pregunta 4

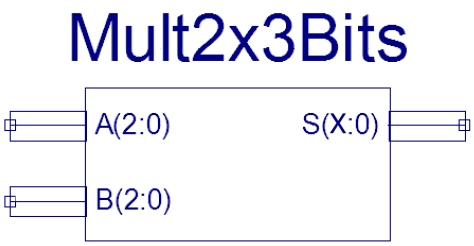
Para las preguntas a) y b) de esta sección, considere el componente de la Figura 1):

Figura 1: Constante 1 de 3 entradas de 1 bit.

Para las preguntas d), e), f) y g) de esta sección, considere los componentes de la Figura 2a) y 2b):



(a) Eleva al cuadrado 1 entrada de 3 bits.



(b) Multiplicador de 2 entradas de 3 bits.

Figura 2

a) Construya la tabla de verdad del componente Const1, considerando las entradas A, B y C, y la salida S. **(0.5 ptos.)**

Solución: Una posible solución se presenta en la siguiente tabla:

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

Figura 3: Tabla de verdad Const1.

b) Utilizando el componente de la Figura 1) como referencia, la simbología de Xilinx y el área de trabajo adjunta, construya una posible implementación del mismo utilizando sólo compuertas lógicas. No puede

utilizar el símbolo de constante y debe usar todas las entradas. **(0.5 ptos.)**

Solución: Una posible solución se presenta en la siguiente figura:

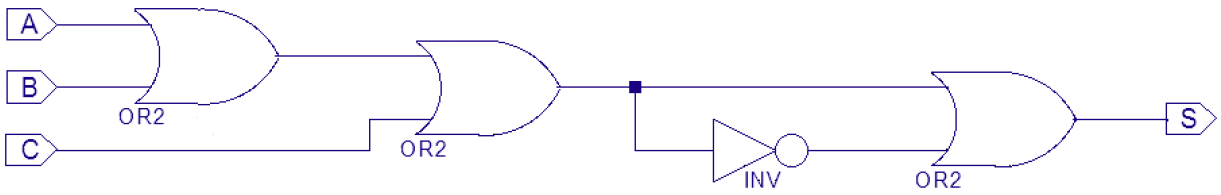


Figura 4: Posible implementación Const1.

c) En Xilinx, ¿Qué es y para qué sirve un Bus Tap?, ¿Cuáles son los requisitos, sobre un Bus de Datos y los Bus Tap, para su correcto funcionamiento? **(0.5 ptos.)**

Solución: Un Bus Tap es un símbolo de Xilinx que permite interactuar con los Buses de Datos, introduciendo o sacando bits, específicos o rangos de ellos, de los mismos. Para su correcto funcionamiento es importante considerar el tamaño del Bus de Datos con el que se está trabajando, que los nombres de ambas estructuras coincidan y el orden en que entran o salen los bits del Bus de Datos.

d) Considerando las salidas S(X:0) de los componentes de la Figura 2), ¿cuáles son los largos en bits de las palabras que salen de cada uno? Justifique sus respuestas. **(0.5 ptos.)**

Solución: Para el componente Pow2, el largo de la palabra es de 6 bits, pues en el peor caso $A = B = C = 1$. Dado lo anterior, esto se interpretaía como el número 7, luego su cuadrado es $49 = 110001$. Para el componente Mult2x3Bits, el largo de la palabra también es 6 bits, pues en el peor caso $A = B = 111$. Luego la multiplicación $A * B = 49 = 110001$.

e) Construya la tabla de verdad del componente Pow2, considerando las entradas A, B y C, y la salida S. **(0.5 ptos.)**

Solución: Considerando que A es el bit más significativo y C es el menos significativo, un posible solución se presenta en la siguiente tabla:

A	B	C	S5	S4	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

Figura 5: Tabla de verdad Pow2.

f) Utilizando el componente de la Figura 2a) como referencia, la simbología de Xilinx y el área de trabajo adjunta, construya un componente capaz de elevar al cuadrado una entrada de 3 bits, de forma que al procesarlo se vea como el de la Figura 2a). Para esto no puede utilizar el componente de la Figura 2b). **(1.5 ptos.)**

Solución: Una posible solución se presenta en la siguiente figura:

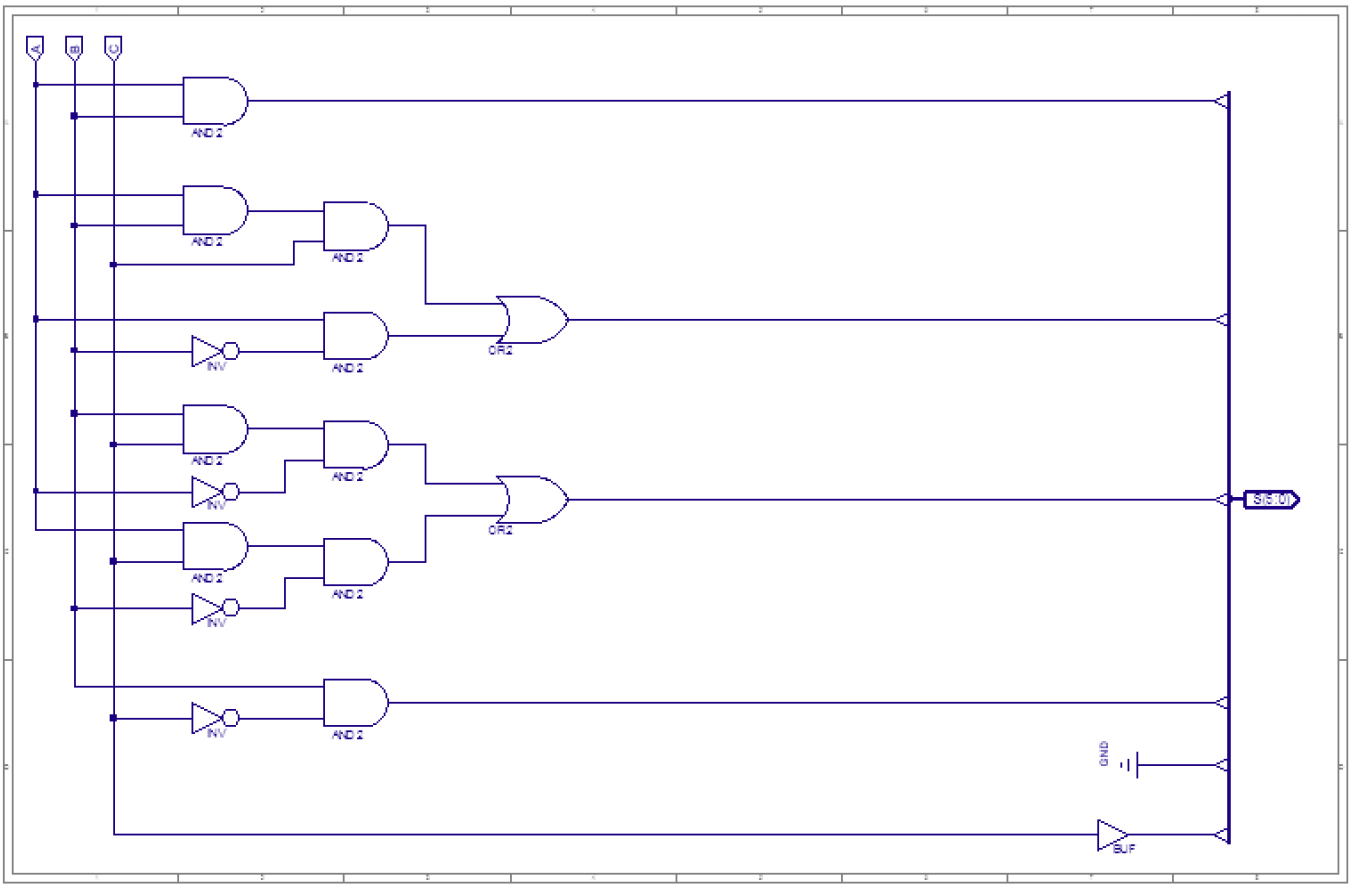


Figura 6: Posible implementación Pow2.

g) Utilizando los componentes de la Figura 2) como referencia, la simbología de Xilinx y el área de trabajo adjunta, construya un nuevo símbolo con 2 entradas de 3 bits cada una, A y B, y una única salida S, tal que $S = (A + B)^2$. Para esto puede basarse en la construcción realizada para el proyecto del curso. **(2 ptos.)**

Solución: Es necesario notar que no se puede realizar la suma primero, pues no se podría utilizar el Pow2 con el resultado de esta. Luego es necesario utilizar la igualdad $(A + B)^2 = A^2 + 2 * A * B + B^2$. Una posible solución se presenta en la siguiente figura:

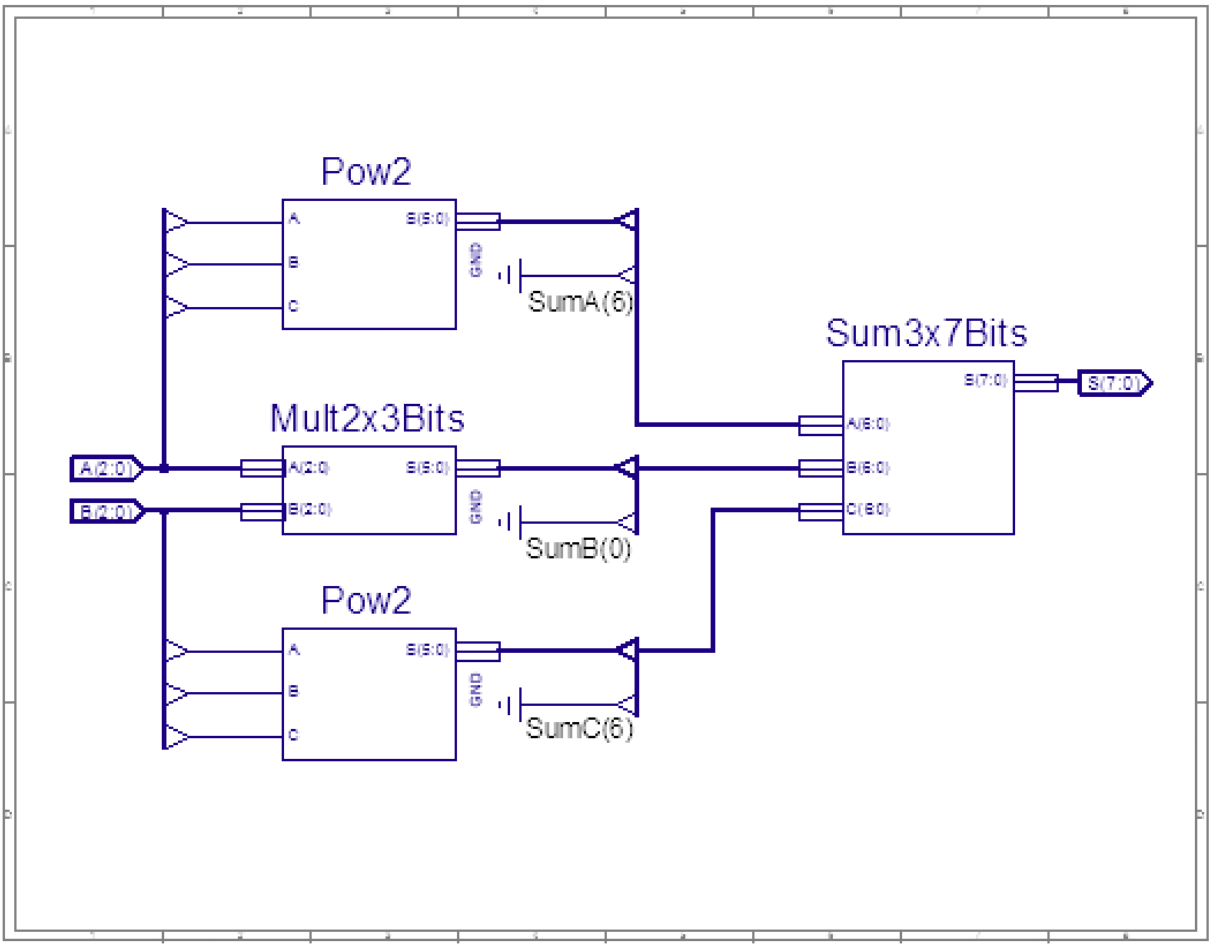


Figura 7: Posible implementación para $S = (A + B)^2$.