EdC-TI Grupo 1 **Prueba 1** **Curso 22/23**

ALUMNO:_____

1.1 Razone cuántas memorias de tamaño 2Kx4 son necesarias para obtener:

a. Una memoria de 2Kx16
   Hacen falta 4 memorias pues queremos obtener una memoria con el mismo número de palabras que la original pero con un ancho de palabra 4 veces superior.
b. Una memoria de 16Kx4
   Hacen falta 8 memorias pues queremos obtener una memoria que tiene 8 veces el número de palabras de la memoria original.
c. Una memoria de 16Kx8
   Hacen falta 16 memorias pues en este caso queremos multiplicar por 8 el número de direcciones y duplicar el tamaño de los datos.

Para el caso *b* diga cuántas líneas tienen el bus de dirección y el bus de datos y cuántas palabras contiene esa memoria.

16K direcciones son $16x2^{10}= 2^4x2^{10}=2^{14}$ direcciones, por tanto, hay 14 líneas en el bus de de direcciones, el bus de datos contiene 4 líneas (ancho de palabra igual a 4) y el número de palabras contenidas en la memoria es 2^{14}, es decir, 16384.

1.2 Describa en Verilog un registro de 8 bits con entradas y salidas separadas (IN y OUT), puesta a cero asíncrona (CL) y carga en paralelo (W).

```
module registro (input [7:0] IN, input CL,W,clk, output [7:0] OUT)

reg [7:0] q;
always @(posedge clk, posedge CL)
        if (CL) q <= 8'b0;
        else if (W) q <= IN;
assign OUT = q;

endmodule
```

1.3 ¿Qué elemento describe el siguiente módulo Verilog?

```
module ABC (input a, b, c, output reg z);
always @(posedge a or negedge  b)
        if (!b) z <= 0;
        else z <= c;
endmodule
```

Se trata de un biestable D (entrada c y salida z) con entrada de puesta a cero asíncrona (b) activa en bajo. El reloj es a.

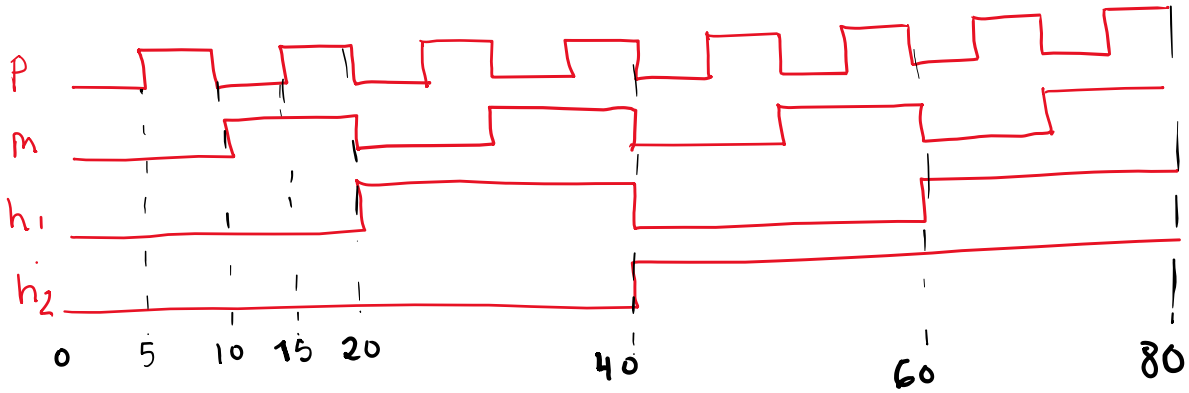
1.4 Dibuje las ondas correspondientes al siguiente testbench:

```
module test();
    reg p, m, h1, h2;
    wire f;
    project uut (.f(f), .p(p), .m(m), .h1(h1), .h2(h2));

initial begin
    p = 0;
    m = 0;
    h1 = 0;
    h2 = 0;
   #40 h2=1;
   #40 $finish;
end
always
    #5 p = ~p;
always
    #10 m = ~m;
always
    #20 h1 = ~h1;

endmodule
```

Se trata de un test para un circuito de 4 entradas y una salida.
En los tres “always” se definen 3 de las entradas y son señales periódicas de periodos 5, 10 y 20 que como podemos ver en el “initial” comienzan en valor 0.
La cuarta entrada se define en el “initial” y comienza valiendo 0 para cambiar a 1 en el instante 40.
40 unidades de tiempo más tarde la simulación debe terminar.



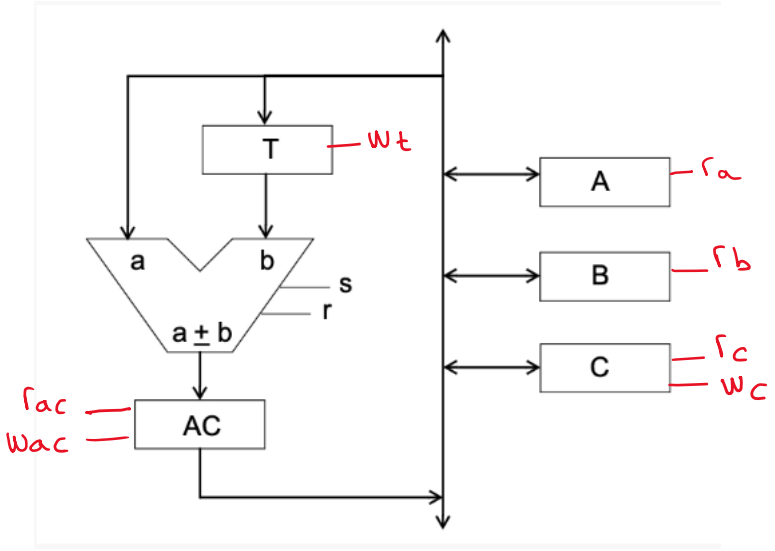
Son formas de onda que dan todas las combinaciones posibles de las 4 entradas.

2. Mediante la unidad de datos de la figura se desea realizar dos operaciones diferentes en función del valor de una variable de entrada S.

S = 0	S=1
C ← A - 3B	C ← B - 3A

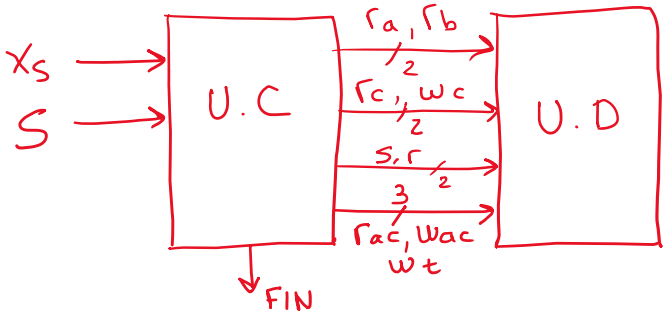
Se pide:

a. Completar los registros con las señales de control necesarias.



Todos los registros que conectan sus salidas al bus compartido deben tener lectura condicional y por tanto señal de lectura, estos son: A, B, C y AC.
Los registros T, AC y C necesitan también señal de escritura porque son, o bien, destino de los datos (C) o necesarios para operar (T y AC).

b. Dibujar un diagrama de bloques donde muestre claramente las entradas y salidas de las unidades de datos y de control.

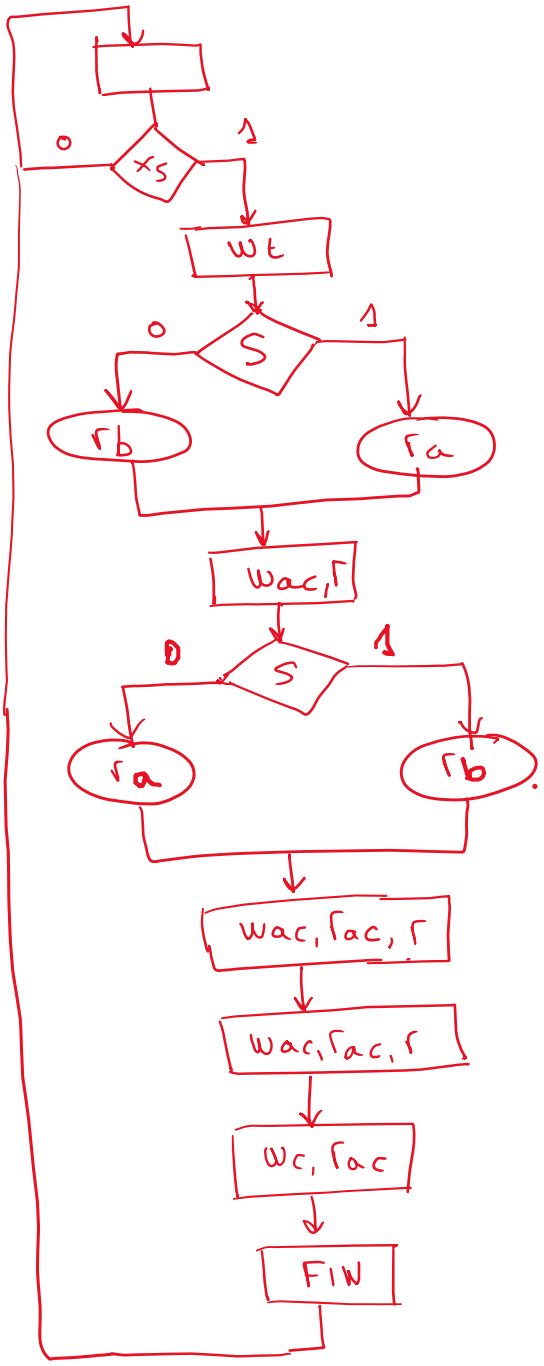


c. Mostrar la secuencia de microoperaciones a realizar para los dos valores de S.

S=0 C ← A - 3B	S= 1 C ← B - 3A
T ← B AC ← A - T AC ← AC - T AC ← AC - T C ← AC	T ← A AC ← B - T AC ← AC - T AC ← AC - T C ← AC

Son comunes a las dos macrooperaciones

d. Obtener la carta ASM de la unidad de control.



3. Se desea añadir al CS3 una nueva instrucción, que permita inicializar posiciones en la memoria de datos directamente sin utilizar ningún registro. Es la instrucción STI, cuya sintaxis es la siguiente: STI YoZ,dato.
   Dicha instrucción almacena una palabra de 8 bits: “dato” en la dirección de MEMDAT indicada por el puntero Y o Z.

La nueva instrucción permite introducir valores en la memoria de datos:
STI YoZ,dato realiza la operación: DATMEM(YoZ) ← dato

a. Asigne a la nueva instrucción el código de operación y el formato de código máquina que considere oportunos. Después como ejemplo, escriba el código máquina de la instrucción STI Z,43.

   El formato será el siguiente:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Código de operación					Rbase			dato							

   Elegiremos como código de operación uno que no esté usado, por ejemplo 10000.
   Rbase será 110 o 111 según si la instrucción usa Y o Z.
   Dato es el dato inmediato, que es una palabra de 8 bits.

   STI Z,43 → 10000 111 00101011 → $872B

b. Describa, si son necesarias, las modificaciones que se deben realizar en la unidad de datos para poderla implementar.
   No es necesario realizar ninguna modificación dado que existe un camino para llevar la dirección de memoria contenida en Z hasta el registro MAR y también hay un camino para llevar el dato inmediato hasta la memoria de datos.

c. Obtenga la secuencia de microoperaciones (ciclo de búsqueda y ejecución) de STI en la nueva unidad de datos (transferencias RT y señales a activar).
   Búsqueda:
   1. IR ← CODMEM, PC ← PC+1 (wir,ipc)

   Ejecución:
   1. AC ← R($IR_{10:8}$) (wac,ta)
   2. MAR ← AC, AC ← $IR_{7:0}$ (wmar,rac,wac,tb,inm)
   3. DATMEM(MAR) ← AC (wmem,rac)

d. Obtenga un programa que, haciendo uso de STI, y las instrucciones que necesite del CS3, inicialice la memoria de datos a partir de la dirección $A0 con los valores: -1, +127 y -128 .
   Varias posibilidades, por ejemplo:

```
LDI R6,$A0
STI Y,-1
LDI R6,$A1
STI Y, 127
LDI R6,$A2
STI Y,-128
```

   O también:

```
LDI R7,$A2
STI Z,-128
SUBI R7,1
STI Z, 127
SUBI R7,1
STI Z,-1
```