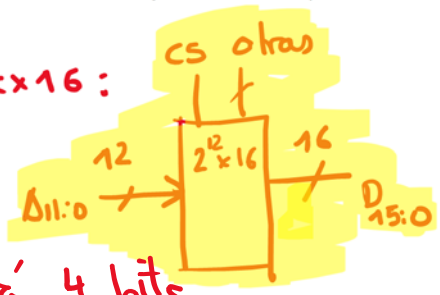
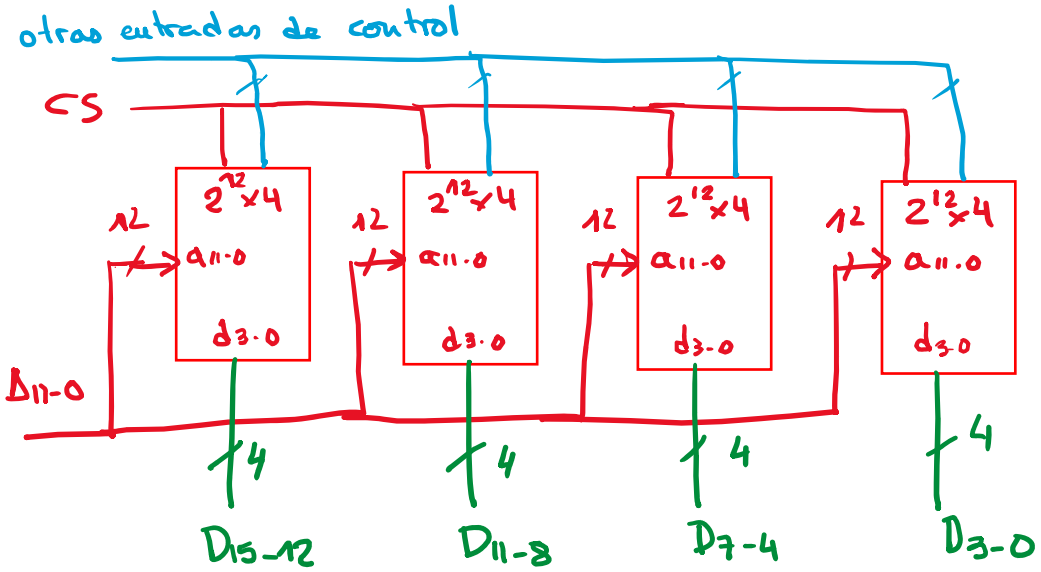
**ALUMNO:**_____

1. Considere una memoria de 4Kx16 y otra de 16Kx4,
   a. ¿Cuál de ellas tiene mayor capacidad? (**0.5 puntos)**
      Las dos tienen la misma capacidad 64kbits o 65536 bits
   b. ¿Cuántas palabras pueden almacenarse en la memoria de 4Kx16?, ¿y en la de 16Kx4? (**0.5 puntos)**
      En la memoria de 4Kx16 se pueden almacenar 4096 palabras de 16 bits ($4x2^{10}=2^2x2^{10}=2^{12}$)
      En la memoria de 16Kx4 se pueden almacenar 16384 palabras de 4 bits ($16x2^{10}=2^4x2^{10}=2^{14}$)
   c. A partir de memorias de 4Kx4 y los circuitos combinacionales que necesite, obtenga una memoria de 4Kx16 **(1 punto)**

Para obtener la memoria de 4kx16:
asociamos 4 memorias de 4k x 4, que deben funcionar simultáneamente.
Cada una de ellas proporcionará 4 bits y juntas nos proporcionan los 16 bits necesarios



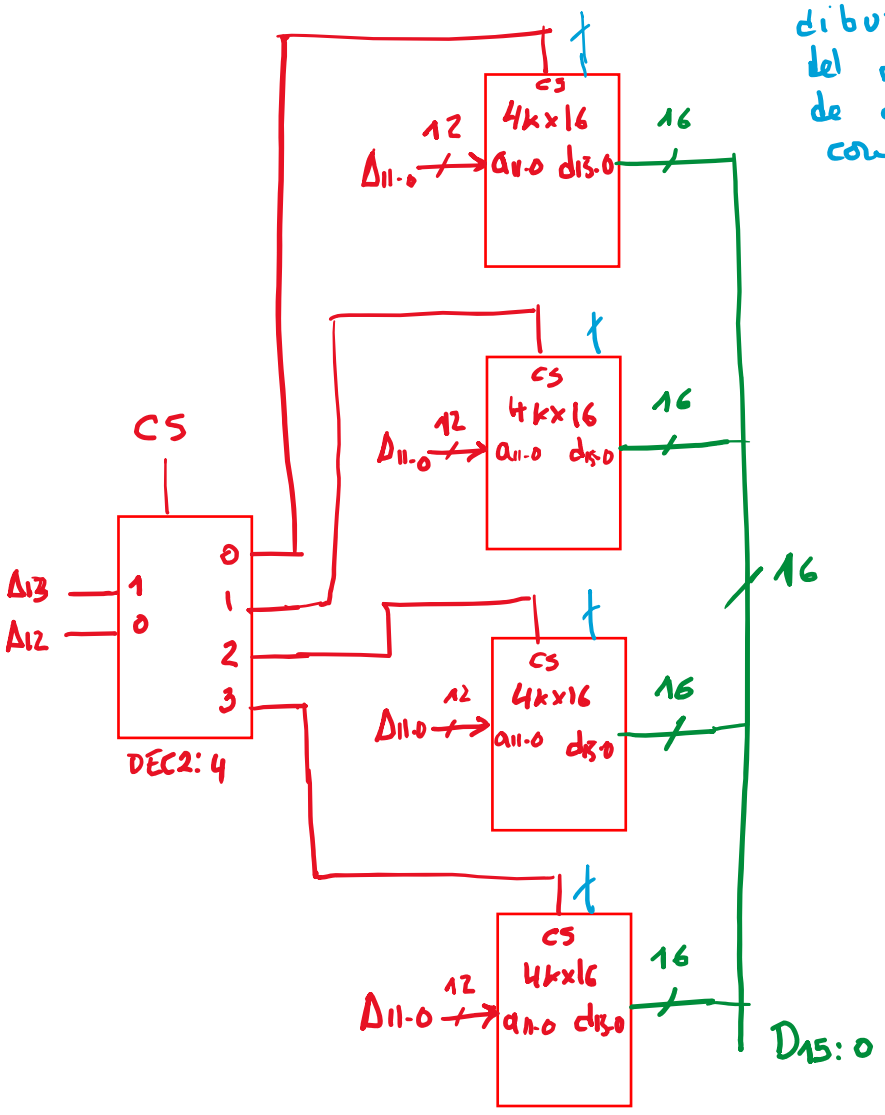




   d. A partir de memorias de 4Kx16 y los circuitos combinacionales que necesite obtenga una memoria de 16Kx16 **(1 punto)**

En este caso conectaremos 4 memorias de 4kx16 de modo que los 16 K palabras se proporcionan en 4 bloques de 4k, cada uno de ellos cubierto por una de las 4 memorias.

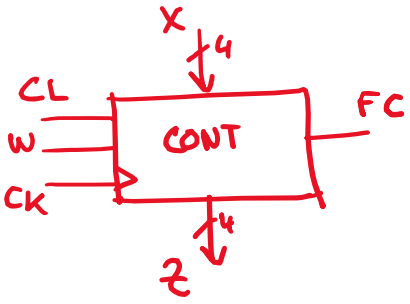
dos buses de salida de las memorias se conectan a un único bus de salida y hay que conseguir que en cada acceso a memoria solo una de ellas pueda estar seleccionada y volcando datos al bus.

Por claridad no dibujo la unión del resto de entradas de control, que irían conectadas entre sí



2. Describa en Verilog un contador de módulo 10 con señal de puesta a cero síncrona activa en alto, carga en paralelo y señal de fin de cuenta. Debe especificar claramente los nombres y función de cada una de las señales que utilice. **(2 puntos)**

Descripción del contador que voy a diseñar:




| CL | W | CONT ← |
|---|---|---|
| 1 | x | 0 |
| 0 | 1 | x |
| 0 | 0 | CONT+1 \|md.10 |

z = [CONT]

FC = 1 si [CONT] = 1001

```
module cont (input cl, w, ck, input [3:0] x,
        output fc, output [3:0] z);

reg [3:0] q;

always @(posedge ck)
          if(cl==1)
             q<= 4'b0000;
          else if (w==1)
              q<=x;
              else if (q<9)
                   q<=q+1;
                  else
                    q<=0;

assign z=q;
assign fc=q[3]*q[0];
endmodule
```

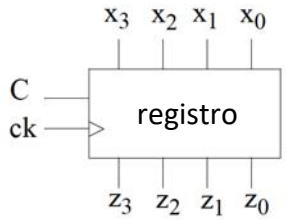
describo el módulo con las entradas y salidas con los nombres elegidos

para almacenar el estado del contador

recoge lo que he puesto en la tabla, el clear es síncrono y activo en alto

la señal de fin de cuenta se activa en 1001 (9)

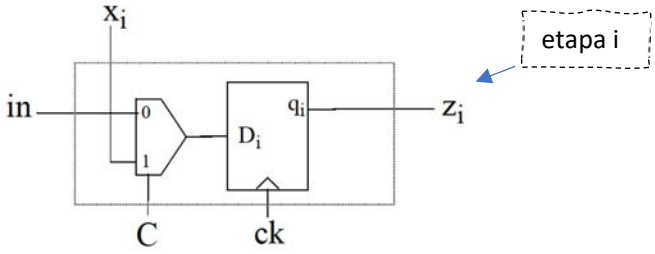
3. Considere el siguiente registro y descríbalo en Verilog utilizando descripción procedimental **(1 punto)**

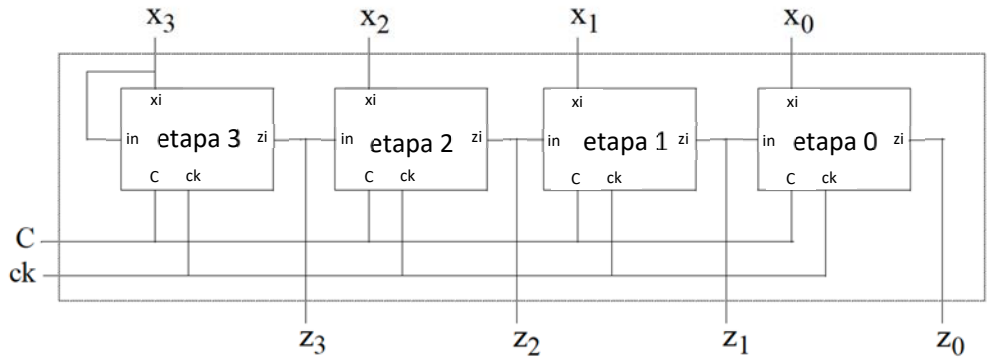



| C | operación |
|---|---|
| 0 | registro ← SHR(registro,$x_3$) |
| 1 | registro ← X |

```
module registro (input C,ck, input [3:0] x, output [3:0] z);

reg [3:0] q;

always @(posedge ck)
        if (C==0)
             q<={x[3],q[3:1]};
            else q<=x;

assign z=q;

endmodule
```

4. A continuación se muestra un diseño modular del registro del ejercicio anterior. Los dibujos muestran la etapa básica (etapa i) y el circuito correspondiente al registro de 4 bits como interconexión de 4 de estas etapas. Como puede observar, la etapa básica consta de un multiplexor y un biestable D.



A partir de la descripción de los módulos *mux21* y *biestD* que se le proporciona, realice un diseño jerárquico. Para ello, siga los siguientes pasos:

a. Obtenga el módulo *etapai* a partir de *mux21* y *biestD* mediante descripción estructural **(1,5 puntos)**

Respetando los nombres de las señales en los módulos y definiendo un cable de unión entre la salida del multiplexor y la entrada de dato del biestable, tenemos:

```
module etapai (input in, xi, C, ck, output zi);
    wire outmux;

    mux21 inst1 (.a(in),.b(xi),.sel(C), .z(outmux));
    biestD inst2 (.d(outmux),.ck(ck),.q(zi));

endmodule
```

```
module mux21 (input a ,b, sel, output z);
        assign z = ~sel&a | sel&b;
endmodule
```

```
module biestD (input ck ,d, output reg q);
        always @ (posedge ck)
                q <= d;
endmodule
```

b. Obtenga la descripción del registro a partir de la conexión de 4 instancias de *etapai* **(1 punto)**

Simplemente instanciamos 4 veces la etapa i y conectamos los módulos entre sí. No es necesario definir cables pues los cables de conexión entre módulos son también salidas (z0,z1,z2 y z3).

```
module registro (input C,ck, input [3:0] x,
                 output [3:0] z);

  etapai inst0 (.xi(x[0]),.C(C),.ck(ck),.in(z[1]),.zi(z[0]));
  etapai inst1 (.xi(x[1]),.C(C),.ck(ck),.in(z[2]),.zi(z[1]));
  etapai inst2 (.xi(x[2]),.C(C),.ck(ck),.in(z[3]),.zi(z[2]));
  etapai inst3 (.xi(x[3]),.C(C),.ck(ck),.in(x[3]),.zi(z[3]));

  endmodule
```

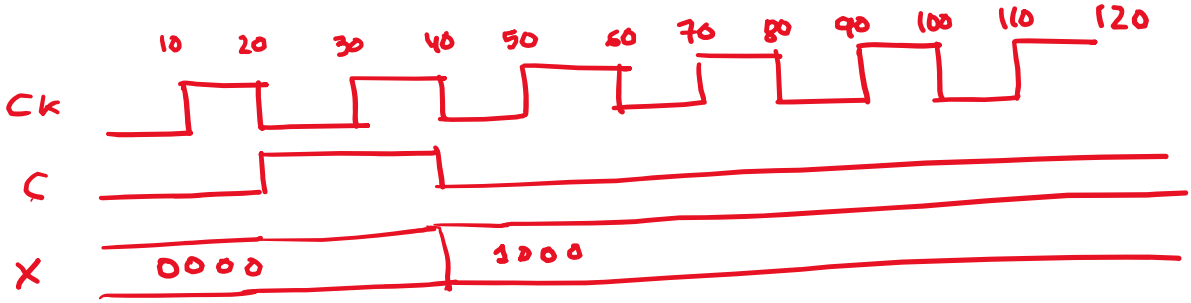
5. Complete el testbench (correspondiente al registro de los apartados 3 y 4) de modo que:
    a. inicialmente todas las entradas parten de valor 0
    b. se define una señal de reloj de 20 ns de periodo
    c. la señal C ha de valer 1 desde t=20ns hasta t=40ns
    d. la señal $x_3$ ha de valer 1 desde t=40ns hasta t=80ns
    e. la simulación debe terminar cuando t=120ns

    **(1,5 puntos)**

```
module registro_tb;

  reg [3:0] x;
  reg C, ck;
  wire [3:0] z;

  registro  uut (.x(x),.C(C),.ck(ck),.z(z));


//completar//
```

```
initial begin
   x=4'h0;
   ck=0;
   C=0;
   $dumpvars(1,uut);
   #20 C=1;
   #20 C=0;
   x[3]=1;
   #40;
   x[3]=0;
   #40 $finish;
  end

  always #10 ck = ~ck;
endmodule
```

https://www.edaplayground.com/x/W42h




También añadiríamos al principio (antes de la definición del módulo):

`timescale 1ns / 1ps