Ej. 1 — (2.0 puntos) Diseñar la ruta de datos y la unidad de control (diagrama ASM y tabla de salidas) de un sistema algorítmico que calcule hasta el n-ésimo término, C_n, de la relación de recurrencia de Segner. Las ecuaciones (1-2) definen la relación de recurrencia que genera la sucesión y el seudo-código se incluye a continuación. Supóngase que cuando se solicita el n-ésimo término, ya se encuentran cargados en memoria los términos anteriores.

$$C_0 = 1 \quad (1)$$

$$C_{n+1} = \sum_{i=0}^{n} C_i \cdot C_{n-i} \quad (2)$$

$C \leftarrow 0$;
for $i{=}0,\ i < n, i{+}{+}$ **do**
  $C \leftarrow C + \text{mem}(i) \cdot \text{mem}(n-i)$;
end
$\text{mem}(n) \leftarrow C$;

El sistema tiene tiene cuatro entradas y dos salidas. Las entradas son: `clk`, `rst`, `ini` y `n`. Supóngase que el valor de `n` está codificado en binario puro. Las salidas son: `seg` y `fin`, de forma que la salida `seg` sea el (n+1)-ésimo elemento de la sucesión. El sistema comienza a funcionar cuando la señal `ini` se pone a 1. En ese instante se almacena el valor de `n` en un registro interno del camino de datos. Cuando se complete el cálculo el sistema volverá al estado inicial, en el que la señal `fin` es igual a 1. Los términos de la sucesión se irán almacenando en una memoria SRAM síncrona de 32x64 bits true dual-port y modo de funcionamiento WRITE_FIRST.

```
entity asm is
  port (clk   : in  std_logic;
        rst   : in  std_logic;
        ini   : in  std_logic;
        n     : in  std_logic_vector(4 downto 0);
        seg   : out std_logic_vector(63 downto 0);
        fin   : out std_logic
        );
end asm;

entity ram is
  port (clk   : in  std_logic;
        dina  : in  std_logic_vector(63 downto 0);
        addra : in  std_logic_vector(4 downto 0)
        wea   : in  std_logic;
        ena   : in  std_logic;
        douta : out std_logic_vector(63 downto 0)
        dinb  : in  std_logic_vector(63 downto 0);
        addrb : in  std_logic_vector(4 downto 0);
        web   : in  std_logic;
        enb   : in  std_logic;
        doutb : out std_logic_vector(63 downto 0)
        );
end ram;
```

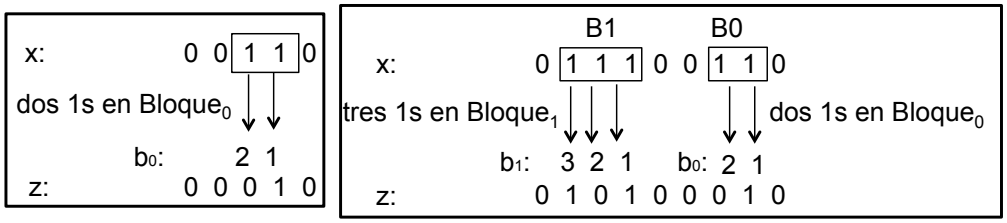
En la ruta de datos se puede usar una memoria RAM[(1)] síncrona true dual-port, registros, contadores, multiplicador, un **único** sumador/restador, y los elementos combinacionales adicionales que se consideren necesarios.
[(1)]: memoria SRAM síncrona de 32x64 bits, true dual-port y modo de escritura WRITE_FIRST.

Ej. 2 — (1.50 puntos) Diseñar utilizando puertas lógicas:

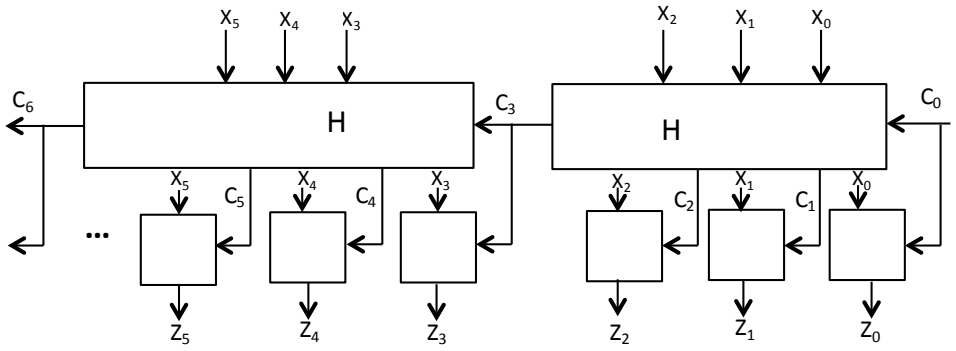
1.Una red iterativa 1D que dado un vector de entrada, x, de n bits genere una salida, z, de n bits tal que:

$$z_i = \begin{cases} 0 & \text{si } x_i = 0 \\ b_j \bmod 2 & \text{si } x_i = 1 \end{cases} \tag{3}$$

donde b_j es el número de unos consecutivos del j-ésimo bloque hasta x_i, con $x_i \in b_j$. Un bloque es un conjunto de unos consecutivos. Es decir, sin ceros entre sí. Las figuras que se presentan a continuación ilustran el funcionamiento de la red iterativa para dos vectores de entrada.



2.Una red iterativa con anticipación de operandos, tal y como se observa en la siguiente figura.



3.Suponiendo que todas las puertas de dos y tres entradas tienen un retardo t, calcular el retardo del camino crítico en el diseño del apartado anterior para $n = 15$. En el caso de haber usado inversores, suponer que su retardo es despreciable.

Ej. 3 — (1.25 puntos) Se ha sintetizado el siguiente código VHVL con las señales a, b, c, d, e y f viniendo de registros de 8 bits.

```
p_add : process(clk, rst)
begin
  if (rst = '1') then
    g <= (others => '0');
  elsif rising_edge(clk) then
    g <= a + b + c + d + e + f;
  end if;

end process p_add;
```

El informe de síntesis contiene la siguiente información:

```
Macro Statistics
# Adders/Subtractors                                   : 5
 8-bit adder                                           : 5
# Registers                                            : 7
 8-bit register                                        : 7
```

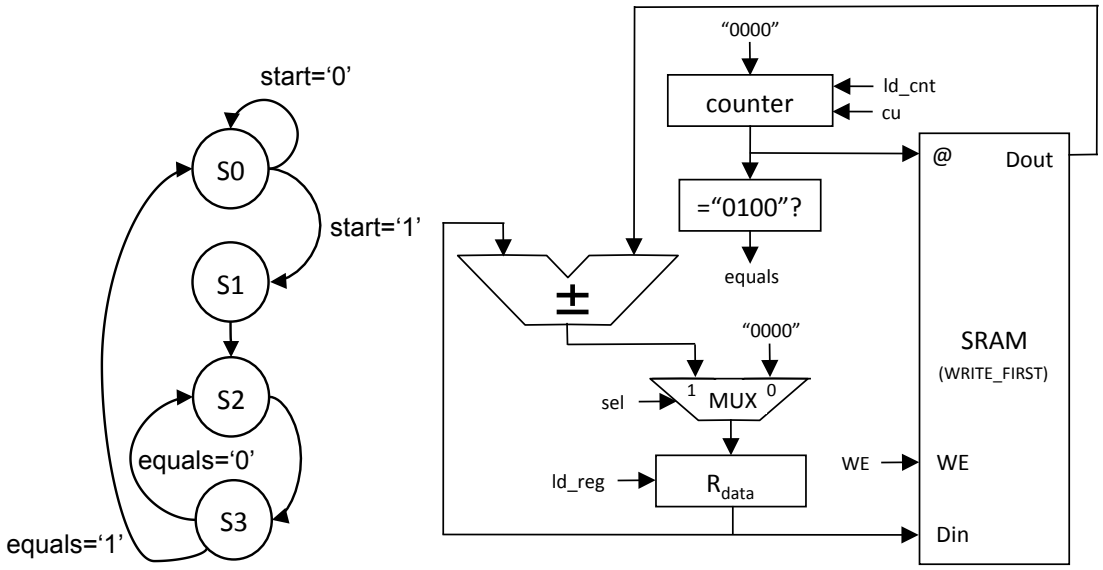
Y el detalle del camino crítico es el siguiente:

```
All values displayed in nanoseconds (ns)
=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Total number of paths / destination ports: 5870 / 8
-------------------------------------------------------------------------
Delay:               6.291ns (Levels of Logic = 12)
  Source:            b_1 (FF)
  Destination:       g_7 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: b_1 to g_7
                                Gate
       Cell:in->out      fanout   Delay  Logical Name (Net Name)
       ----------------------------------  ------------
        FDC:CLK->Q            2   0.626  b_1 (b_1)
ADDSUB01 LUT3:I0->O           1   0.479  Madd_g_addsub0001C1 (Madd_g_addsub0001C)
        LUT4:I3->O            1   0.479  Madd_g_addsub0001_Madd_lut<2> (Madd_g_addsub0001_Madd_lut<2>)
        XORCY:LI->O           2   0.541  Madd_g_addsub0001_Madd_xor<2> (g_addsub0001<2>)
        LUT3:I2->O            1   0.479  Madd_g_addsub0003C11 (Madd_g_addsub0003C1)
        LUT4:I3->O            1   0.479  Madd_g_addsub0003_Madd_lut<3> (Madd_g_addsub0003_Madd_lut<3>)
ADDSUB03 MUXCY:S->O           1   0.435  Madd_g_addsub0003_Madd_cy<3> (Madd_g_addsub0003_Madd_cy<3>)
        MUXCY:CI->O           1   0.056  Madd_g_addsub0003_Madd_cy<4> (Madd_g_addsub0003_Madd_cy<4>)
        MUXCY:CI->O           1   0.056  Madd_g_addsub0003_Madd_cy<5> (Madd_g_addsub0003_Madd_cy<5>)
        XORCY:CI->O           1   0.786  Madd_g_addsub0003_Madd_xor<6> (g_addsub0003<6>)
        LUT2:I1->O            1   0.479  Madd_g_add0000_lut<6> (Madd_g_add0000_lut<6>)
  ADD00 MUXCY:S->O            0   0.435  Madd_g_add0000_cy<6> (Madd_g_add0000_cy<6>)
        XORCY:CI->O           1   0.786  Madd_g_add0000_xor<7> (g_add0000<7>)
        FDC: t_setup              0.176  g_7
       ----------------------------------------
       Total                      6.291ns
```

1. Dibujar un posible esquemático del circuito implementado teniendo en cuenta los datos del informe de síntesis (macro statistics) y señalar cuál es el camino crítico. **Nota: hay varias posibles soluciones.**
2. Calcular el margen de setup de este camino si la frecuencia de reloj es 180 MHz y el skew de reloj para los registros del camino es 0.125 ns.
3. Suponiendo que los retardos de cada sumador del camino crítico se calculan en base a los retardos de las LUT, XOR y los MUXes señalados en el esquema, ¿podría funcionar el circuito correctamente a 200 MHz? Si la respuesta es negativa, indíquese qué solución podría tomarse para que pudiera funcionar a la nueva frecuencia. Dibujar la solución propuesta en el esquemático del apartado 1.

Ej. 4 — (1.25 puntos) Completar el cronograma correspondiente al siguiente sistema e indicar el contenido final de la memoria. Las entradas al sistema son: `start`, `clk` y `rst` (reset del registro y contador, puesto a '0' en el cronograma); la salida del sistema es `end`; y el resto de señales son señales internas según tabla adjunta. Considerar que la memoria es síncrona tanto para lectura como para escritura, y que funciona en modo WRITE_FIRST. Suponer también que el módulo `counter` es un contador módulo 8 ascendente con carga paralela.

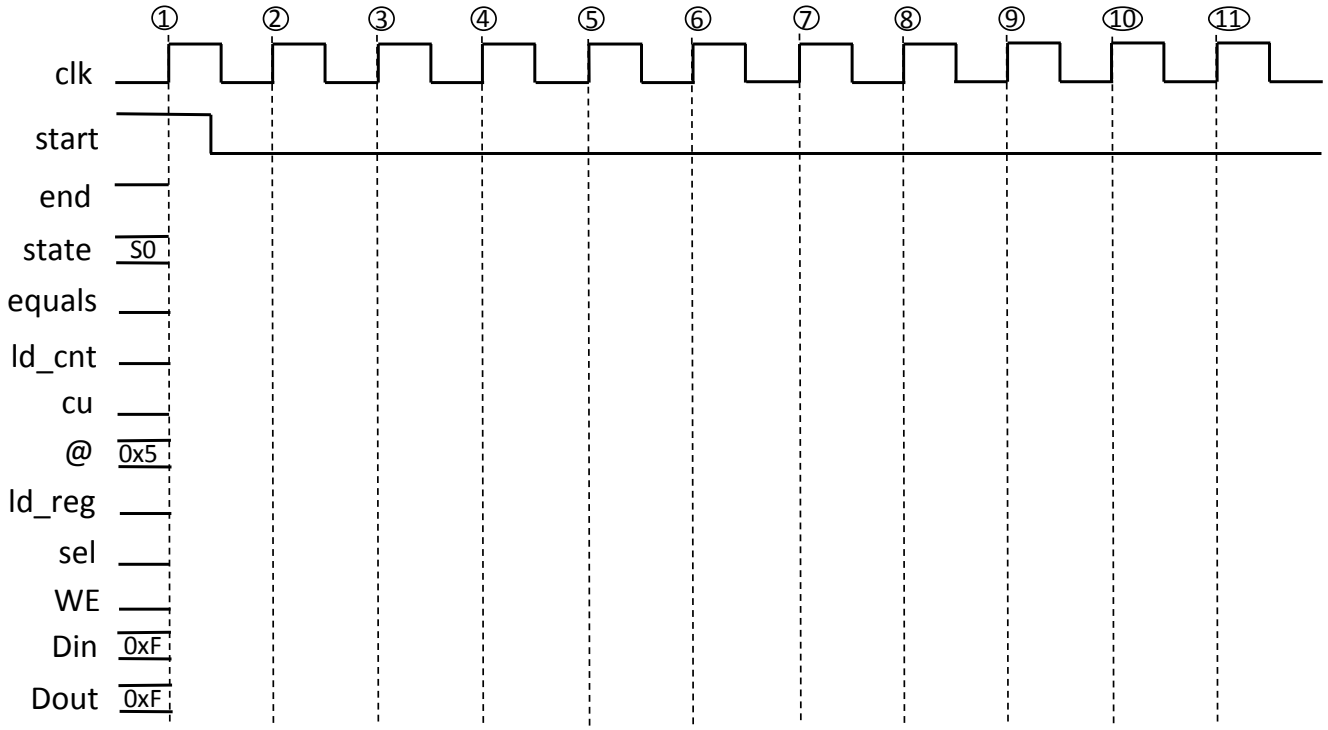


Contenido de la SRAM:

Address	Data
0x00	0x0A
0x01	0x0B
0x02	0x0C
0x03	0x0D
0x04	0x0E
0x05	0x0F
0x06	0x00

Tabla de salidas:

state	ld_cnt	cu	sel	ld_reg	WE	end
S0	0	0	0	0	0	1
S1	1	0	0	0	0	0
S2	0	1	1	1	0	0
S3	0	0	0	0	1	0



Ej. 5 — (0.5 puntos) Contestar brevemente las siguientes dos cuestiones:

1. Describir brevemente el proceso de lectura de una memoria DRAM.

2. Se desea implementar una memoria de tamaño 128Kx16 bits, a partir de una Block RAM de Xilinx de tamaño 256x8 bits de un único puerto. ¿Se podrían realizar las lecturas en un ciclo de reloj? ¿Cuál sería el hardware adicional necesario para realizar correctamente las operaciones de lectura sobre dicha memoria? Realizar un esquema indicándolo.