Ej. 1 — (2.0 puntos) Diseñar el diagrama ASM, la ruta de datos y la tabla de salidas de la unidad de control de un sistema algorítmico que ordene de mayor a menor 16 números de 8 bits. Los números están guardados de forma no ordenada en las 16 primeras palabras de una memoria SRAM síncrona de 32x8 bits y deben almacenarse en orden descendente en las líneas de memoria de la 16 a la 31.
El sistema tiene como entradas: `clk`, `rst` e `ini`. Y como salida una única señal `fin`. El sistema comienza a funcionar cuando la señal `ini` se pone a 1. Cuando se complete el cálculo el sistema volverá al estado inicial, en el que la señal `fin` es igual a 1. Además de la memoria SRAM síncrona de 32x8 bits, READ-FIRST, con un solo puerto, podemos usar en la ruta de datos: contadores, registros, una única ALU con operaciones de suma y resta (r/s=0 es suma y r/s=1 es resta) y los elementos combinacionales que se necesiten.
La ALU, además del resultado, tiene como salidas las señales: `zero`, resultado de la última operación es un cero, y `neg`, resultado de la última operación es negativo.
El algoritmo es el siguiente:

1. Encontramos el valor máximo de los 16 números iniciales y lo guardamos en la dirección 16. En la posición inicial de este máximo colocamos un cero.
2. Encontramos el valor máximo de los 15 números restantes y lo guardamos en la dirección 17. En la posición inicial de este máximo colocamos un cero.
3. Encontramos el valor máximo de los 14 números restantes y lo guardamos en la dirección 18. En la posición inicial de este máximo colocamos un cero.
4. Repetimos este proceso hasta tener ordenados todos los números en las líneas 16-31 y el valor cero en las líneas 0-15.

El seudo-código y la definición de los interfaces son:

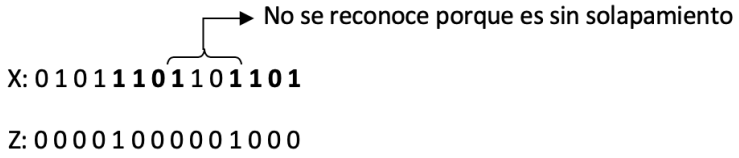
```
ind1 ← 16 ;
while ind1 ≤ 31 do
    ind2 ← 0 ;
    tempMax ← 0 ;
    indTempMax ← 0 ;
    while ind2 ≤ 15 do
        if SRAM(ind2) > tempMax then
            tempMax ← SRAM(ind2) ;
            indTempMax ← ind2 ;
        ind2++ ;
    end
    SRAM(ind1) ← tempMax ;
    SRAM(indTempMax) ← 0 ;
    ind1++
end
fin ← 1 ;
```

```
entity asm is
  port (clk   : in  std_logic;
        rst   : in  std_logic;
        ini   : in  std_logic;
        fin   : out std_logic
        );
end asm;
```

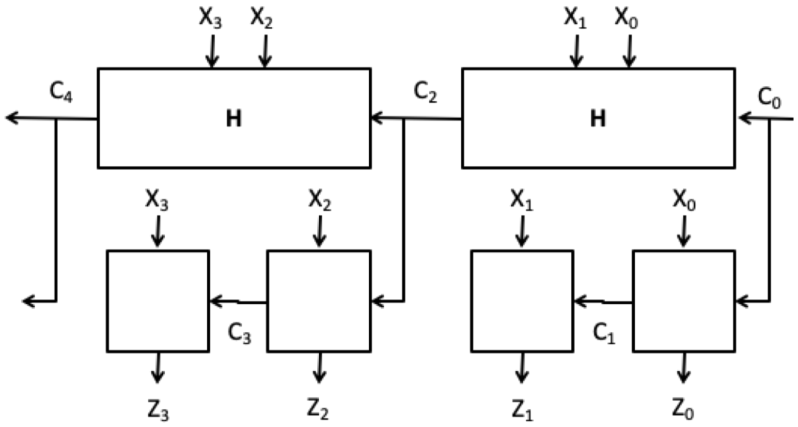
```
entity sram is
  port (clk   : in  std_logic;
        dina  : in  std_logic_vector(7 downto 0);
        addra : in  std_logic_vector(4 downto 0)
        wea   : in  std_logic;
        ena   : in  std_logic;
        douta : out std_logic_vector(7 downto 0)
        );
end sram;
```

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 la salida Z sera igual a 1 si en esa celda se ha detectado el patrón 1101 sin solapamiento y 0 en caso contrario. El patrón se reconoce de derecha a izquierda del vector de entrada. Ejemplo:



2. Diseñar una red iterativa con anticipación de operandos H sólo entre cada grupo de 2 bits, mientras que el acarreo dentro de cada grupo de 2 bits se hace por propagación de operandos, tal y como se observa en la siguiente figura.



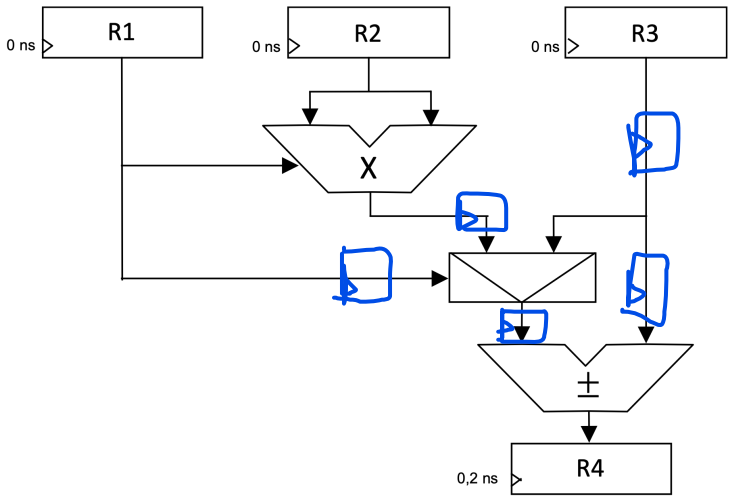
3. Suponiendo que todas las puertas de dos entradas tienen un retardo t, las puertas de tres entradas un retardo $2 \cdot t$ y las puertas de cuatro entradas tienen un retardo $3 \cdot t$, calcular el retardo del camino crítico en el diseño del apartado anterior para $n = 16$. En el caso de haber usado inversores, suponer que su retardo es despreciable.

Ej. 3 — (1.25 puntos) En el siguiente circuito, los valores de propagación de los elementos combinacionales son los siguientes:

- Retardo(Multiplicador) = 3 ns
- Retardo(Multiplexor) = 0.5 ns
- Retardo(Sumador) = 2.5 ns

Los valores de las líneas de reloj corresponden a los retardos de propagación desde la fuente del reloj hasta el registro correspondiente. Los parámetros de los componentes secuenciales son los siguientes:

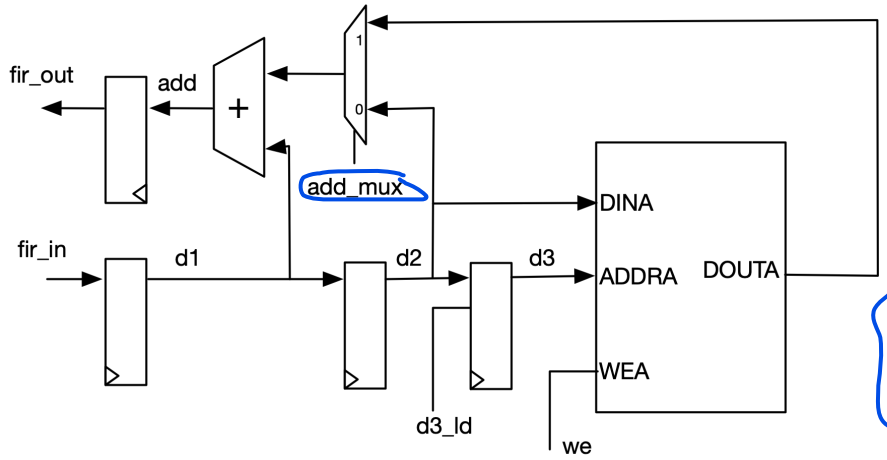
- tclk-2-q = 0.2ns
- tsetup = 0.1ns
- thold = 0.15ns

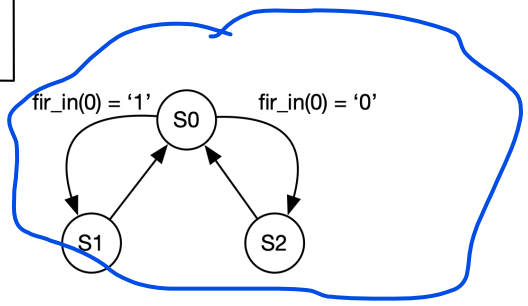


1. Indicar cuál es el camino crítico y calcular la máxima frecuencia de reloj de este circuito con las características descritas anteriormente. 6,1ns f = 163,9 MHz
2. Utilizar segmentación de tal manera que la frecuencia de funcionamiento máxima se incremente lo máximo posible. Indicar cuál es la nueva frecuencia máxima de funcionamiento ahora. Los nuevos registros tendrán una latencia de reloj de 0.2 ns. 3,1 ns f = 322,6 MHz
3. Indicar si ha habido algún efecto colateral como consecuencia de la segmentación aplicada en el apartado anterior. Si procede, indicar cómo puede solucionarse.

Violación hold −0,15 ns / Buffers

Ej. 4 — (1.25 puntos) Completar el cronograma correspondiente al siguiente sistema e indicar el contenido final de la memoria SRAM. Las entradas al sistema son `fir_in`, `clk` y `rst` (reset de los registros, está a '0' en el cronograma) y la salida es `fir_out`; el resto de señales son internas según tabla adjunta. Considerar (1) que la memoria es síncrona y funciona en modo WRITE_FIRST y ALWAYS_ENABLE y (2) que los registros que no poseen señal de carga (`ld`) cargan en cada flanco de reloj.



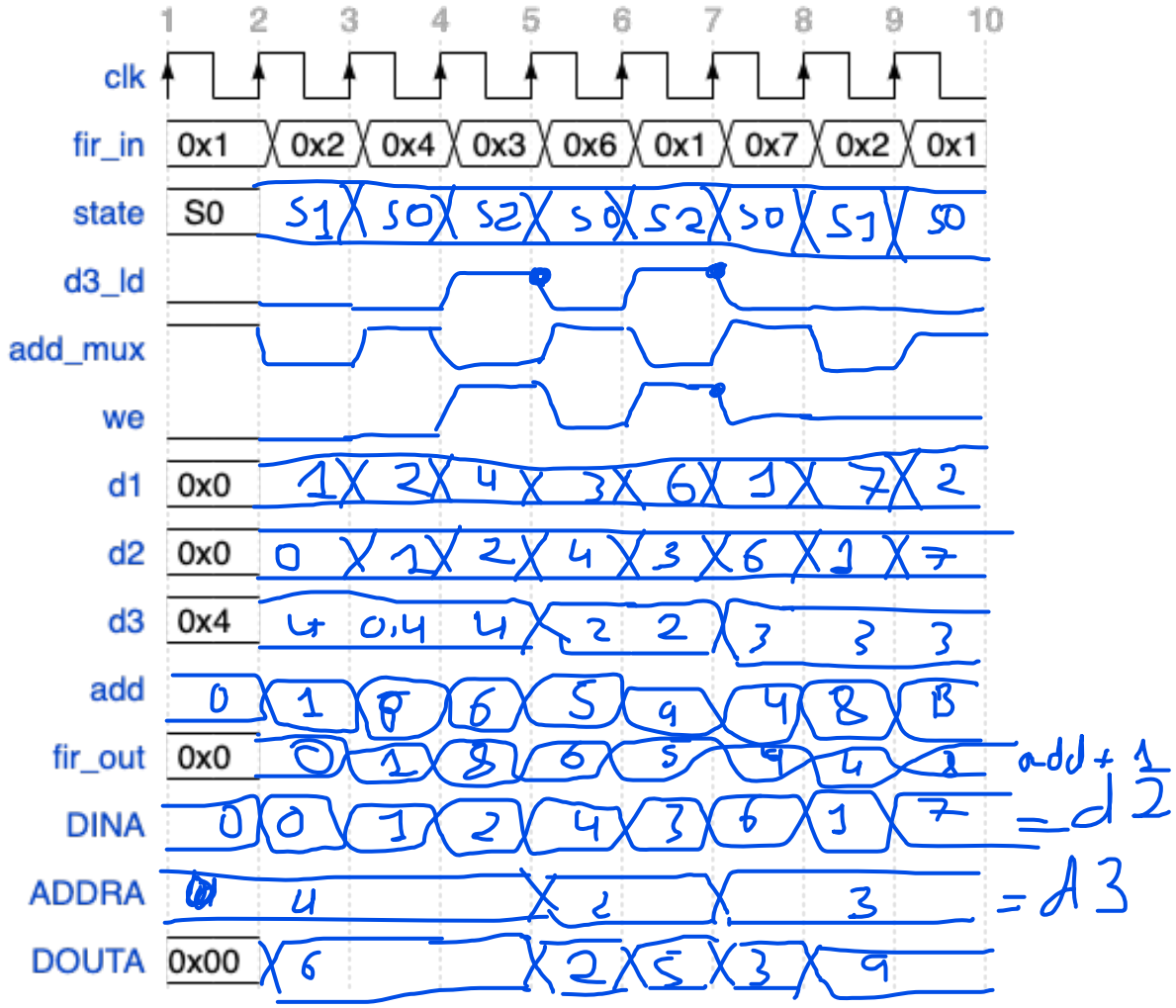


SRAM

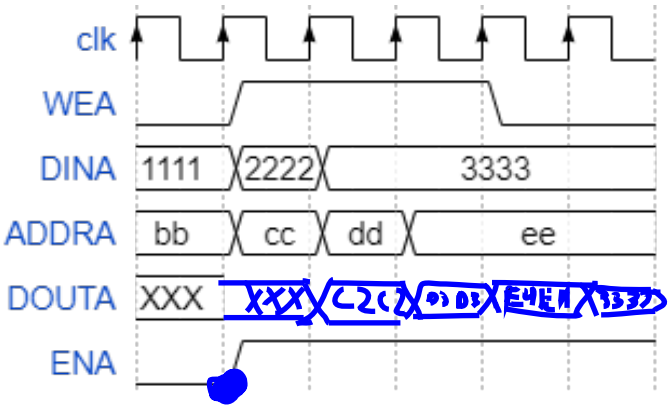
ADDR	DATA
0x00	0x03
0x01	0x07
0x02	~~0x05~~ 3
0x03	0x09
0x04	~~0x06~~ 2
0x05	0x02
0x06	0x03
0x07	0x0C

Tabla de salidas UC

State	d3_ld	add_mux	we
S0	0	1	0
S1	0	0	0
S2	1	0	1



Ej. 5 — (0.5 puntos) Completar el cronograma siguiente, asumiendo que las señales corresponden a una Block RAM que funciona en modo READ_FIRST. Asumir que los contenidos iniciales de la Block RAM se indican en la tabla a continuación.



Address	Data
aa	A0A0
bb	B1B1
cc	~~C2C2~~ 2222
dd	~~D3D3~~ 3334
ee	~~E4E4~~ 3335