2)

a)

* La entrada clk representa el valor a alamacenar en los flip-flop D
* El force-input es una entrada de control que habilita el almacenado de valores en los flip-flop D y afecta los 3 al mismo tiempo. Las 3 entradas conectadas al input W en los flip-flop D tambien cumplen esta funcion, solo que actua cada una en un flip-flop D
* Las entradas conectadas al input en\_out en cada flip-flop D son de control y habilitan la salida del valor almacenado
* El en\_force\_input es una entrada de control que regula el paso del valor que entra en force\_input
* R0, R1 y R2 son un valor de salida y representan los valores almacenados en cada uno de los flip-flop D.
* Luego esta la salida general del componente que, en el caso de que no haya error devuelve los valores alamacenados en lo/s Ri y/o en el force\_input

b) sec\_activacion: Force\_Input -> en\_Force\_input -> (en Registro 1) w -> clk

En caso de que se desee alterar componendes, esta es la secuencia de pasos para no modifique a “registro\_1”: (en registro\_1) deshabilitamos w.

c) sec: en\_force\_input 1 (no nos importa el valor que tenga “Force\_input” previamente (0 o 1) -> (en “Registro\_0”) w 1-> clk 1 -> clk 0 -> en\_Force\_input 0 -> en\_out 1 (en “r\_0”) -> w 1 (en “R\_1”) -> clk 1 -> clk 0 -> en\_out (en “R\_0”) 0 -> w (en ”R\_1”) 0 -> en\_out (en “R\_1”) 0 -> (en “r\_0”) en\_out 0 -> (en “r\_2”) en\_out 1 -> clk 1 -> clk 0 -> en r\_0 0 -> en r\_2 w= 1 -> en r\_2 en\_out = 0 -> en r\_1 en\_out= 1-> clk 1 -> clk 0

3

ENTRADAS:

* Force\_input se encarga de ingresar el valor que queremos almacenar en alguna de las memorias
* en\_Force\_input es una entrada de control que habilita o no el paso del valor de Force\_input
* Regi\_Write con i entre 0 y 3, es una variable de control que permite o no en cada memoria el guardado de un valor
* Regi\_enableOut con i entre 0 y 3, es una variable de control que permite o no la salida del valor almacenado en cada memoria
* ALU\_N\_Write, con N =A || B es una variable de control que permite o no almacenar el valor de un registro (o varios con el mismo valor) en la ALU
* OP es una variable de control que le indica a la ALU que operacion realizar entre +,- ,and, or
* ALU\_enable\_out es una variable de control que permite la salida del valor resultante de la operacion realizada por la ALU
* El clock es una variable de control que permite almacenar valores en los registros y realizar operaciones en la ALU, al oscilar entre 0 y 1. Tambien de regular las salidas de los registros y la ALU al cambiar de valor

SALIDAS:

* N\_debug con N = A || B es una salida que nos permite ver en todo momento el valor almacenado en las entradas correspondientes al valor a o b de la ALU
* S\_debug es una salida que nos permite ver en todo momento el resultado de la operacion realizada por la ALU
* N es una salida que nos indica si el resultado de la operacion de la ALU fue negativo en complemento a 2
* Z es una salida que nos indica si el resultado de la operacion de la ALU fue 0
* V es una salida que nos indica si el resultado de la operacion de la ALU tuvo overflow en la suma interpretada en complemento a 2 (0 sino)
* C es una salida que nos indica si el resultado de la operacion de la ALU tuvo carry en la suma y la resta (0 sino)

b)

* Reg4\_debug (display de cada registro) es una salida que nos permite ver en todo momento el valor almacenado en un registro
* El Display inferior esta conectado a un cable que contiene el valor tanto del resultado de la operacion de la ALU, como de la entrada force\_input, por lo que siempre que no haya conflicto con valores, nos lo mostrara.

c) Primero de todo suponemos todos los valores seteados a 0. Dicho esto, a continuacion la secuencia:

Seteamos (0100) en Force\_input -> en\_Force\_input 1 -> Reg2\_write 1 -> clk 1 -> clk0 -> Reg2\_write 0 -> Seteamos (1101) en Force\_input -> Reg3\_write 1 -> clk 1 -> clk0

d)

| Valores de entrada | Resultado Op1 | Flags | Resultado Op2 | Flags |
| --- | --- | --- | --- | --- |
| (4,0) | 4 en ambas interpretaciones | N=0, Z=0, V=0, C=0 | 4 en ambas interpretaciones | N=0, Z=0, V=0, C=0 |
| (7,-1) | 8 en sin signo y -8 en complemento a 2 | N=1, Z=0, V=1, C=1 | 7 en ambas interpretaciones | N=0, Z=0, V=0, C=0 |
| (-8,-2) | 6 en ambas interpretaciones | N=0, Z=0, V=1, C=1 | 10 en sin signo y -6 en complemento a 2 | N=1, Z=0, V=1, C=1 |
| (8,-9)\* | N/A | N/A | N/A | N/A |

\*No se pueden realizar operaciones con estos valores, pues -9 no se puede escribir en complemento a 2 ya que este funciona en el rango [7,-8]

Veamos a continuacion la secuencia de operaciones para almacenar en R2 y R3 los resultados de OR y sub con (4,0) como valores:

Seteamos (0100) en force\_input → en\_force\_input 1 →Reg0\_write 1→clk 1→ clk 0→ Reg0\_write 0 → Seteamos (0000) en force\_input →Reg1\_write 1→clk 1→ clk 0→ Reg1\_write 0→ALU\_A\_Write 1→Reg0\_enable\_out 1→ clk 1→ clk 0→ Reg0\_enable\_out 0 →ALU\_A\_Write 0→ ALU\_B\_Write 1→Reg1\_enable\_out 1→ clk 1→ clk 0→ Reg1\_enable\_out 0→ALU\_B\_Write 0→op 11 →clk 1→ clk 0→ en\_force\_input 0 → ALU\_enable\_out 1→Reg2\_write 1→clk 1→ clk 0→ Reg2\_write 0→ op 01 →clk 1→ clk 0→ ALU\_enable\_out 1→Reg3\_write 1→clk 1→ clk 0→ Reg3\_write 0

e) Se niega la salida del clock para que en el mismo ciclo de encendido y apagado la ALU realice la operacion (clk 1) y el registro\_4bytes devuelva el resultado (clk 0, interpretado como 1 por el registro), bajando el tiempo de respuesta del componente, ya que si dicho NOT no estuviera negando al clk, entonces se neceseterian dos ciclos del mismo para cargar dos valores (suponiendo write 1 en ambos) y devolver el resultado.