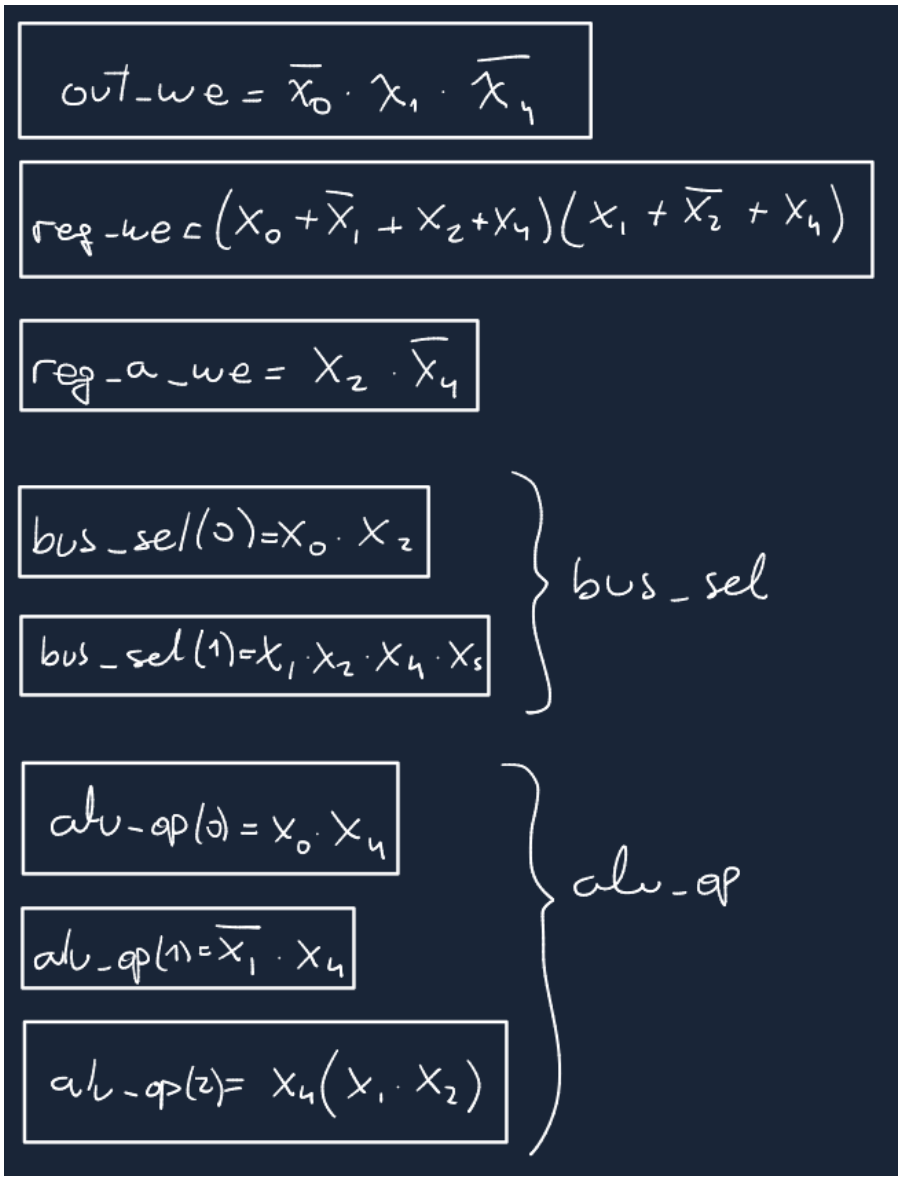
1.

Instruction	Instruccion En Binario	*bus_sel*	*alu_op*	*reg_a_we*	*out_we*	*reg_we*
IN (0X01)	00000001	10	000	0	0	1
OUT (0X02)	00000010	00	000	0	1	0
MOV(0X03)	00000011	00	000	0	0	1
LDA(0X04)	00000100	00	000	1	0	0
LDI(0X05)	00000101	01	000	1	0	0
ADD(0X10)	0001000	00	010	0	0	1
SUB(0X11)	00010001	00	011	0	0	1
AND(0X12)	00010010	00	100	0	0	1
OR(0X13)	00010011	00	101	0	0	1
XOR(0X14)	00010100	00	110	0	0	1
SHL(0X20)	00100000	00	000	0	0	1
SHR(0X21)	00100001	00	000	0	0	1

Reducción de las funciones de cada bit:

Bits de las instrucciones: input(0) = x_0, input(1) = x_1, etc...



2. Código hecho en el archivo “Decoder.vhd”.

Las sentencias concurrentes usadas para cada salida de la Unidad de Decodificación están basadas en las funciones reducidas obtenidas en la primera consigna del trabajo practico. En el caso de las salidas compuestas por más de un bit, se definió una función para cada bit.

El circuito implementado es combinacional, ya que solo depende de la combinación de los valores de entrada para definir un valor de salida. Para la misma combinación, siempre se obtiene el mismo valor de salida.

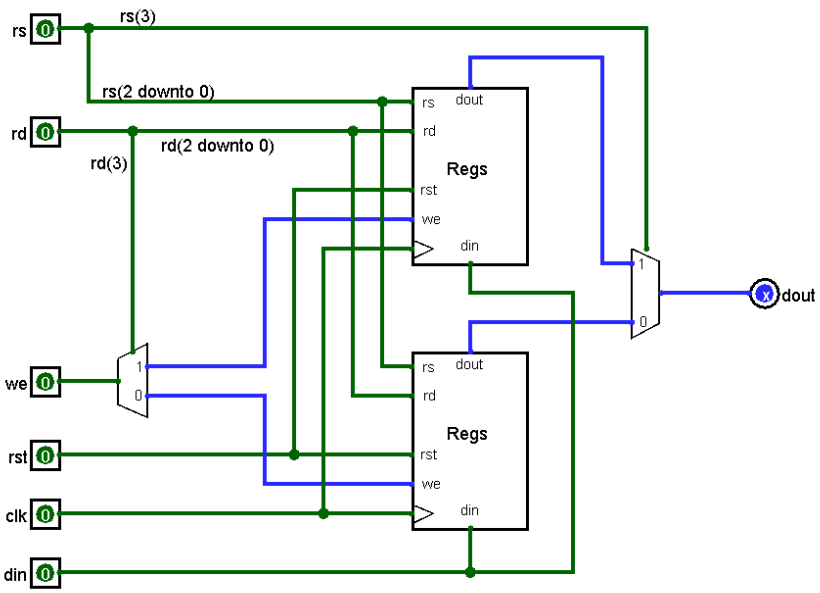
3. Código del registro hecho en el archivo “regs_16.vhd”, el del multiplexor y demultiplexor se encuentran en los archivos “mux2_4.vhd” y “dmux2_1” respectivamente.

Para implementar el banco de 16 registros que se pide haciendo uso del banco de 8 registros dado, es necesario conectar ambos por medio de un demultiplexor y un multiplexor para que los registros funcionen en conjunto.

Se implementaron un DMUX2_1 (dmux2_1.vhd) para dirigir la señal we y un MUX 2_4 (mux2_4.vhd) para elegir cuál de las salida de banco es la que se envía a la salida del banco de 16 registros.

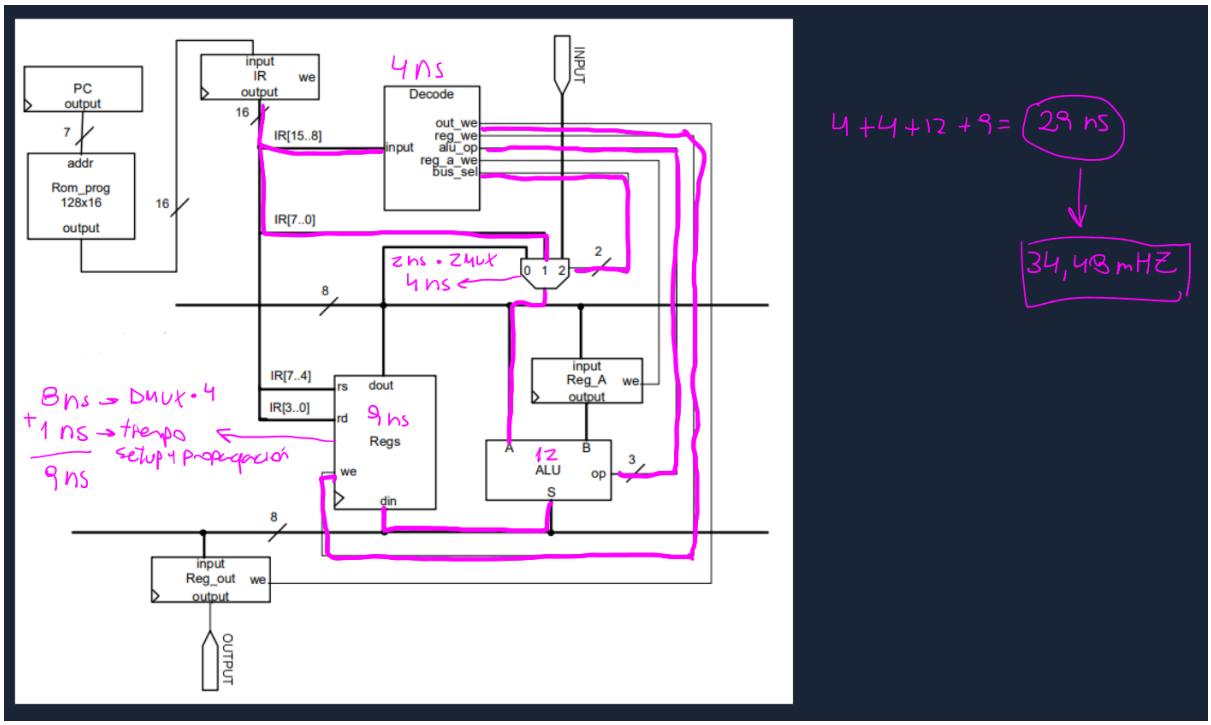
El DMUX recibe la entrada we y usa el bit de mayor peso de rd como selector para elegir cuál de los bancos regs se habilita para la escritura, ya que esto representa una tabla de combinaciones dividida a la mitad, que solo se diferencia por ese último bit. El MUX recibe la salida dout de ambos bancos regs y usa el bit de mayor peso de rs para seleccionar cuál de los dos es la salida final del componente.

Las entradas rst, clk y din se conectan tal cual a las entradas correspondientes de ambos bancos regs. El resto de los bits de las entradas rs y rd (del bit 2 al bit 0) se conectan a las entradas rs y rd de los bancos.



4. La frecuencia de operación optima a la que puede trabajar es de 34.48 mHz.

El tiempo se calcula según la instrucción que más tiempo tome en ejecutar, las cuales son las instrucciones que utilizan la ALU (ADD, SUB, AND, OR, etc...).



5. En cada fila de la tabla se especifica la instrucción dada y su código hexadecimal que se cargaría en la ROM del programa, y se resaltan los registros que participan en la operación.

codigo hexa	instrucción	r0	r1	r2	r3	r4	r5	r7	r14	reg_A	reg_out
Inicializacion		0	0	0	0	0	0	0	0	0	0
0x053	ldi 3	0	0	0	0	0	0	0	0	3	0
0x1010	add r1, r0	0	3	0	0	0	0	0	0	3	0
0x055	ldi 5	0	3	0	0	0	0	0	0	5	0
0x1021	add r2, r1	0	3	8	0	0	0	0	0	5	0
0x1132	sub r3, r2	0	3	8	3	0	0	0	0	5	0
0x1241	and r4, r1	0	3	8	3	1	0	0	0	5	0
0x1451	xor r5, r1	0	3	8	3	1	6	0	0	5	0
0x1371	or r7, r1	0	3	8	3	1	6	7	0	5	0
0x03E3	mov r14, r3	0	3	8	3	1	6	7	3	5	0
0x022	out r2	0	3	8	3	1	6	7	3	5	8
0x023	out r3	0	3	8	3	1	6	7	3	5	3
0x024	out r4	0	3	8	3	1	6	7	3	5	1
0x025	out r5	0	3	8	3	1	6	7	3	5	7
0x027	out r7	0	3	8	3	1	6	7	3	5	8
0x02E	out r14	0	3	8	3	1	6	7	3	5	3