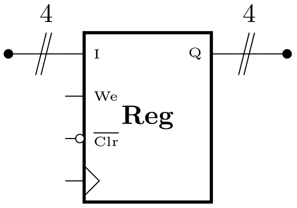
Hardware - Tutorial 4
Memory Design

Exercice 1 : Register design

A register is a CPU component that can hold a small amount of data. You can see it as a memory that can only hold one value at a time. Its capacity is defined by its length in bits.

This exercice is about designing a 4 bits register. It will consist in a component with the following I/O:

- 4 input bits
- 4 output bits
- One We control bit : When it is **high**, the input bits are stored in the internal memory
- One $\overline{Clr}$ control bit : When it is **low**, the internal memory is reset to 0
- One clock input : It will provide a clock signal for the internal flipflops



As the internal flipflops states are not inter-dependent, we can simplify the designing process by first designing a one bit register and then scale it up.

1) Write a table to describe the different states of the register depending on its inputs.

State	$\overline{Clr}$	WE	Next State
Idle	1	Ø	Clear
Idle	1	0	Idle
Idle	1	1	Load
Clear	Ø	Ø	Idle
Load	Ø	Ø	Idle

Now we can design the logic that will drive the internal flipflop of the register. You can choose any flipflop to design the register but try to think about what makes a type preferable to another.

2) Write a truth table for the inputs of the chosen flipflop to ensure the output matches the state table.

- D flipflop version:

$\overline{Clr}$	WE	I	Q	D
0	Ø	Ø	Ø	0
1	0	Ø	0	0
1	0	Ø	1	1
1	1	0	Ø	0
1	1	1	Ø	1

- JK flipflop version:

$\overline{Clr}$	WE	I	Q	J	K
0	Ø	Ø	0	0	Ø
0	Ø	Ø	1	Ø	1
1	0	Ø	0	0	Ø
1	0	Ø	1	Ø	0
1	1	0	0	0	Ø
1	1	0	1	Ø	1
1	1	1	0	1	Ø
1	1	1	1	Ø	0

3) Fill up a K-Map for each of theses inputs and extract their logic expressions.

- D flipflop version:

D		**IQ**			
		00	01	11	10
$\overline{\text{Clr}}$ WE	00	0	0	0	0
	01	0	0	0	0
	11	0	0	1	1
	10	0	1	1	0

$$D = \overline{Clr}\cdot WE\cdot I + \overline{Clr}\cdot \overline{WE}\cdot Q$$

- JK flipflop version:

J		**IQ**			
		00	01	11	10
$\overline{\text{Clr}}$ WE	00	0	Ø	Ø	0
	01	0	Ø	Ø	0
	11	0	Ø	Ø	1
	10	0	Ø	Ø	0

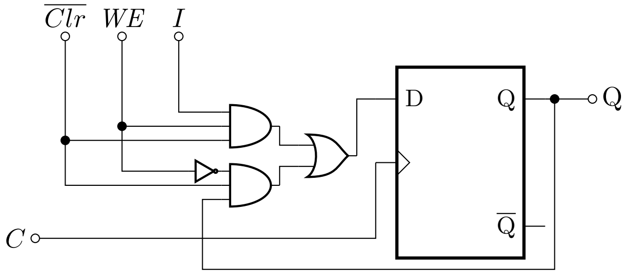
K		**IQ**			
		00	01	11	10
$\overline{\text{Clr}}$ WE	00	Ø	1	1	Ø
	01	Ø	1	1	Ø
	11	Ø	1	0	Ø
	10	Ø	0	0	Ø

$$J = \overline{Clr}\cdot WE\cdot I$$

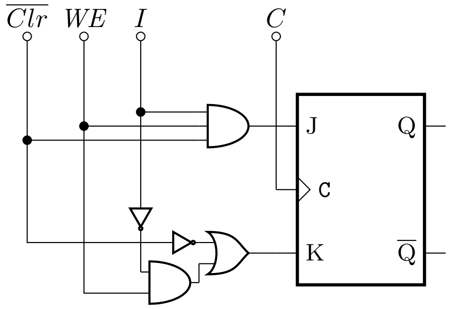
$$K = \overline{\overline{Clr}} + WE\cdot \bar{I}$$

4) Draw the circuit of your 1 bit register.

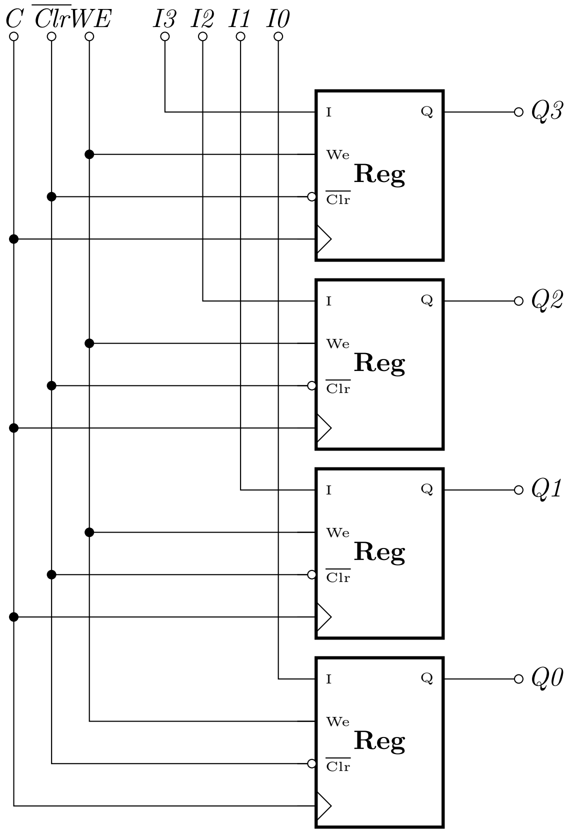
- D version:



- JK version:



5) Using 4 of your previously designed circuit, draw the circuit of a 4 bits register.



Exercice 2 : Memory design

A RAM memory is a component that can hold a fixed amount of binary words. Each of theses words can be accessed by using an unique number: its address. A memory is defined by two properties :

- The width represents the size of each stored binary word
- The depth represents the number of words that can be stored

The goal of the following exercice is to design a memory that can store 16 words of 4 bits.

- How many bits should be used to encode each of its addresses ?

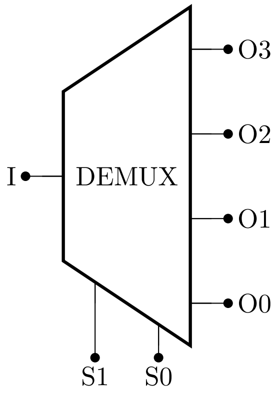
→ 16 stored words means we need at least 16 addresses, so addresses must be written on at least four bits.

The memory will be made of 16 4bits registers. Upon read or write operations, only one of them will be active at a given time.

a) Demultiplexer

Design a combinatorial circuit that has a one bit data input, two select bits and 4 outputs. For each possible combination of the select bits, the input bit must be copied on a different output bit. The other output bits must stay low.

Here is how this component is drawn in circuits :



Correction:

Truth Table

I	S_1	S_0	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

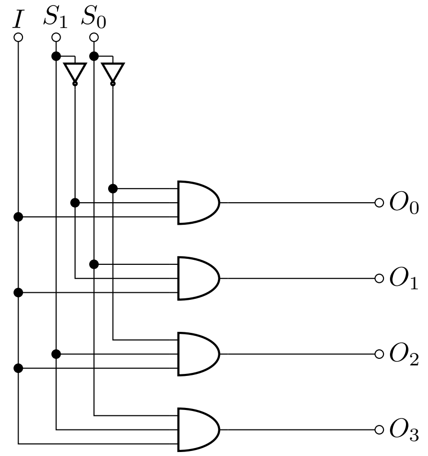
$O_0 = I{\cdot}\overline{S_1}{\cdot}\overline{S_0}$

$O_1 = I{\cdot}\overline{S_1}{\cdot}S_0$

$O_2 = I{\cdot}S_1{\cdot}\overline{S_0}$

$O_3 = I{\cdot}S_1{\cdot}S_0$

Circuit



b) Assembly

Your memory will consist in a matrix of 16 registers. You will use a slightly different version than the one you designed earlier. It will have no CLR input, but one CS (Chip Select) input. While the CS input is low, the register ouptuts are all tight low and it does not respond to any other input. While the CS is high, the register works acts normally.

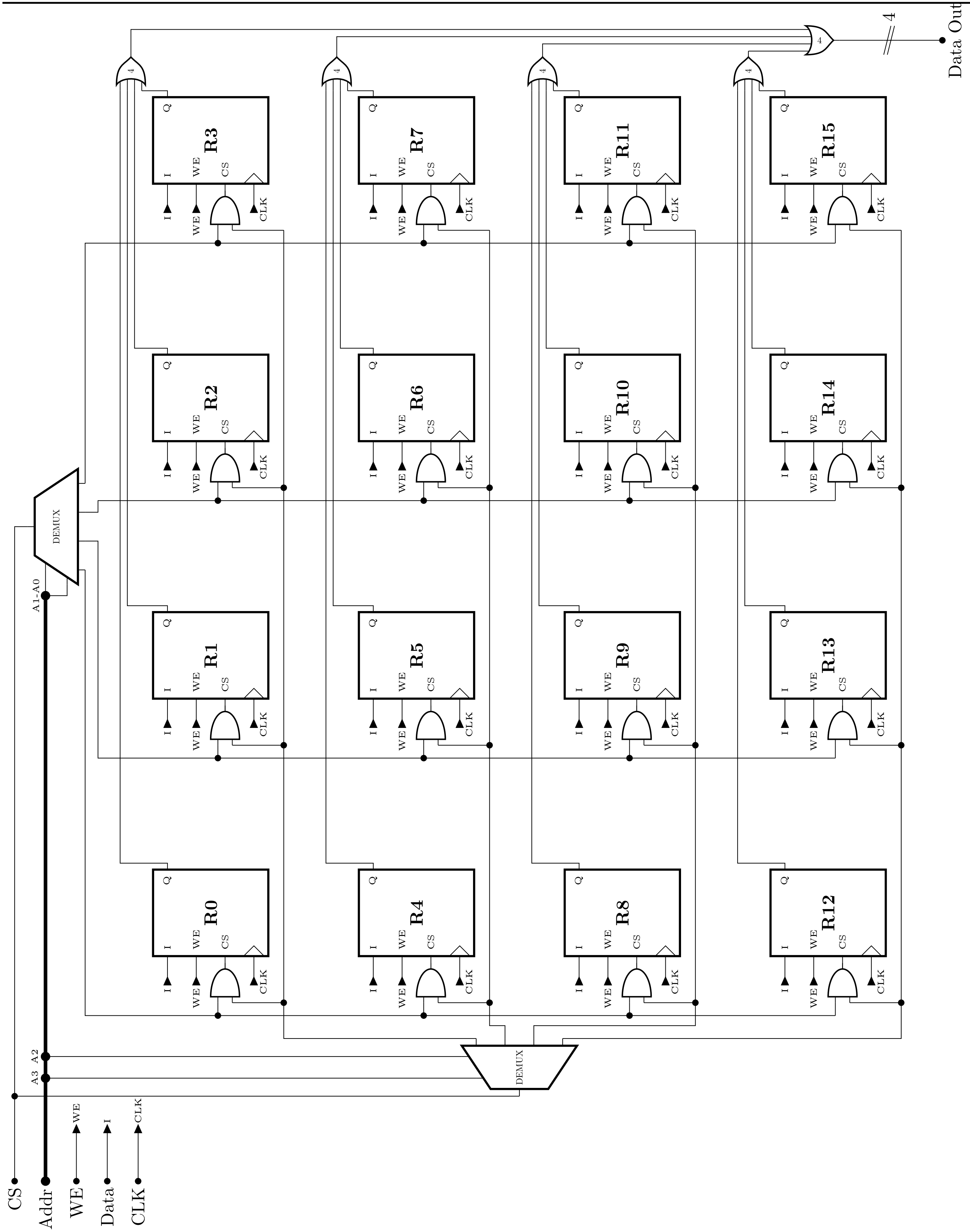
In order to select which register is active for an operation, you will need two of the previously designed demultiplexers.

- Complete the following schematic to have a fully functionnal memory

⇒ Start by adding the multiplexers and connect them. One is used to select a column of registers, the other selects a line.

⇒ Then connect all outputs to the main output (Remember that multiple logic outputs cannot be connected together by a simple cable).

⇒ Finally Connect the input databus and all the remaining register inputs.


4
Data Out
4
4
4
4
Q
R3
I
WE
CS
CLK
Q
R7
Q
R11
Q
R15
Q
R2
Q
R6
Q
R10
Q
R14
DEMUX
A1-A0
Q
R1
Q
R5
Q
R9
Q
R13
Q
R0
Q
R4
Q
R8
Q
R12
DEMUX
A2
A3
WE
I
CLK
CS
Addr
WE
Data
CLK