report: adder and subtractor

Lorenzo Ramella, Alessandro Matteo Rossi, Marco Tambini

June 15, 2021

Abstract

Contents

1 Basics concepts

To be able to realize calculation a circuit need to be able to do logic operation, in parrticular in digital electronics we usually use boolean algebra.

To be able to create a circuit that is able to work with boolean algebra we first need to define The various component. The number 0 and 1 have to be proprieties of an electric circuit that can be "moved"; the easiest of this proprieties to use is the voltage so we can assign the number 0 to a low voltage and the number 1 to a high voltage.

For example if we define $0V$ as low, negative or 0 and $5V$ as high positive or 1 we can define a threshold voltage is exactly in the middle so that voltge under $2.5V$ will be considered 0 and voltage above will be considered 1.

Once 1 and 0 are defined we need to define the operation:

- "!" is the negation and can be represented by a NOT gate.
- " + " is the addition and can be represented by a OR gate.
- " ∗ " is the multiplication and can he represented by an AND gate.

2 Logic gate

When we tal about a logic gate we are talking about a circuit that can take a certain number of input and give a single output depending on the input readed; the output needs to be readable by another logic gate of the same family.

The main logic gate are the following:

2.1 N-mos

To crete the logic gate we need to first know how to use a MOSFET. In our case we used only N-MOS.

The N-MOS is a transistor that get in input a gate voltage, a drain voltage, a source voltage and a body voltage; in most cases the source and the body are internally connected since the body needs to be at the lowest voltage and the source is usually at mass.

When a positive voltage is aplied between the drain and source a deplition layer block the passage of current is formed and there is no passage of current. If we then start aplying a positive voltage between gate and body the electron will start "balancing" the gaps in the P substrate but there will still be no passage of current.

After V_{GS} surpass a threshold voltage the current will start to flow from drain to source. At the start the ratio between the current and V_{DS} is linear but, when V_{DS} is big enough, the ratio stop to grow and become almost linear as seen in Immage and we find ourself in the region of saturation.

The voltage of drain we used for our circuits was $5V$ so whenever the current passed we where in the condition of saturation. The threshold voltage vary between transistor and usually go from $0.5V$ to $5V$. The transistor we used is the $IRF822$ and we found in lag that the threshold voltage is approssimatively , we decided to use $5V$ for V_{GS} to make, as we will see later, input and output aproximativaly the same.

2.2 NAND gate

The first gate we realized in the laboratory was the NAND gate.

2.3 NOR gate

2.4 Other gate

3 16-bit calculator

3.1 Input with 16-bit

One of the problem that arise when we want to use more memory space is an increase in complexity; this one make the circuit more expensive and also take space that could be used for other pourpose.

While in a small scale we could use a 2-bit priority encoder for our project on the simulator we wanted to create a calculator tht could operate with 16 bit in input and give an output just as big, excluding sign bit; to do that a priority encoder would be inpractical and unnecessarely more complex. Another problem that would araise in that of the phisical input since priority encoder would require as many input as the number of decimal number we want.

To solve both problem we decided to use a keyboard to take the input throught some button intead of the levers we used in the laboratory, since this method could solve both the problem with the number of input and the problem of the complexity of the decoder.

3.2 Keyboard

The keyboard part connect a point at high voltage to the rest of the circuits; the button we used where: the number form 0 to 9, the operation + the operation − the = sign to get the result and a clear button that reset the circuit.

3.3 4-bit encoder

The first part of our encoder is a 4-bit encoder without priority.

As you can see in Immage this part can be subdivided in 3 part

The red part is a 4-bit encoder without priority. As we were saying in the paragraph above, the advantage of using a keyboard is that we will, under normal condition, only get 1 input at a time making the usage of the encoder without priority more advantageous.

The encoder works by checking with a OR if a number to wich the bit correspond is inputted. For example if we press the button 6 only the second and third bit will output a high voltage since the first and fourth input are not connected.

The green part is a OR connected to all the number input to check when a button corresponding to a number is pressed since, if 0 was pressed, it would not result in any binary input but should still be readed like in the number 10.

The blue part is a small "flash memory"; this part, whose work will be discussed in the next part, is a simple "security" method to make sure that the input of the button correctly arrive correctly to the next memory. This part could be eliminated if all the tempistic are correct but we prefered keeping it to make sure no problem arise.

3.4 Memory and successive inputs

This is the part that make possible to get subsequent input and it's formed by two memories and two adder.

in this part the input is stored and, after a new input is pressed, the stored number will be multiplied by 10 and then added to the new input. doing this we can abtain all number up till the memory limit.

It's inportant to understand that this part work by using the concepts of rising edge, where there is a transition from low to high of the input, and falling edge, where the input changes from high to low. Using this concept make possible to divide the action corresponding to a sigle input in two part.

3.4.1 Rising edge

When the first button is pressed the ciurcuit is in a state of rising edge. Right after a button is pressed the line that go to the AND of the second memory, highlighted in yellow Immage , switch to low blocking the and to let any signal pass from the first memory to the second memory.

The second line that change is the one under the first memory (highlighted in green) that also turn off; this is the line that is responsible of the clearing of the memory and, when it switch from high to low, it reset the memory.

After a little while the last line that change is the one on top of the first memory that let the input be memorized in the cleared memory.

3.4.2 Falling edge

After the button is released the circuit transit to falling edge. The process is similar to the preceden; the first line to change this time is the one below the second memory, this line has, like the one in the first memory,the job of resetting the memory.

Right after it's the line on top of the second memory that switch on permitting the last one to store the information outputted from the first memory.

The last line to switch is, like before, the one on top of the first memory that switch from high to low and stop the memory from being modified.

3.4.3 Multiplication

The work of the adders isn't during the falling edge or the rising edge but is instead between the two.

The second adder, highlighted in red, is the one responsible of multiplying 10 to the previous number and work between falling edge and rising edge

To do a multiplication in binary you need to shift the number as much as the bit of the multiplier and add all the result. For example let's take $6 \cdot 10$, writing it in binary we obtain $0110 \cdot 1010$. if we subdivide the multiplication if $0110 \cdot 1000$ and $0110 \cdot 0010$ we can simply shift the number. after this we add the two result $101000 + 001010 = 110010$ and the result that we obtain correspond with 60.

to do this operation we simply make the input go to the second and fourth entrance of the adder and take in output the result multiplied by 10. Since the multiplier is 10 we can see that the first bit of the result will always be low and that the first three number of the result don't need any addition so we can directly take result without passing by a full adder.

3.4.4 Adder

The first adder, highlighted in blue, has the job of adding the previous number, multiplied by 10, and the new number in input. this second adder work between the rising edge and the falling edge.

3.5 Sign bit

To do the subtraction we could have used the full subtractor but we decided to use the addition between positive and negative number in binary. To do the subtraction we needed a way to read if the second number is positive or negative. As visible in Immage wesimply used a flip flop. From the top the first line is the input +, the second is the input for − the third is the clear button that also reset the sign bit.

3.6 Memory

The final part of the input is the two memory in Immage .

The first memory, highlighted in green, is the one where we save the first number. Knowing that the first number is always positive we can save the memory and give the output to the processing part just as it is. Since we want this memory to be modified only when it has to register the first number we added the AND "line" that is controlled by the line exiting the OR in Immage; with this the memory will store information only when + or − are inputted. the line is also connected at the right side of the memory to a line that clear the memory from immagemaking possible to input the second number.

The second memory, highlighted in yellow, work in the same way as the first one but take the check at the input is done with the = sign instead.

One difference between the first and second memory are the XOR gate at the output of the second memory. The XOR take on one input the respective bit and on the other the sign bit. After XOR the output will stay the same if the sign bit is positive and invrted if negative; this is done to do the addition between a negative and a positive.

To see the complete process about the addition of positive and negative number check chapter