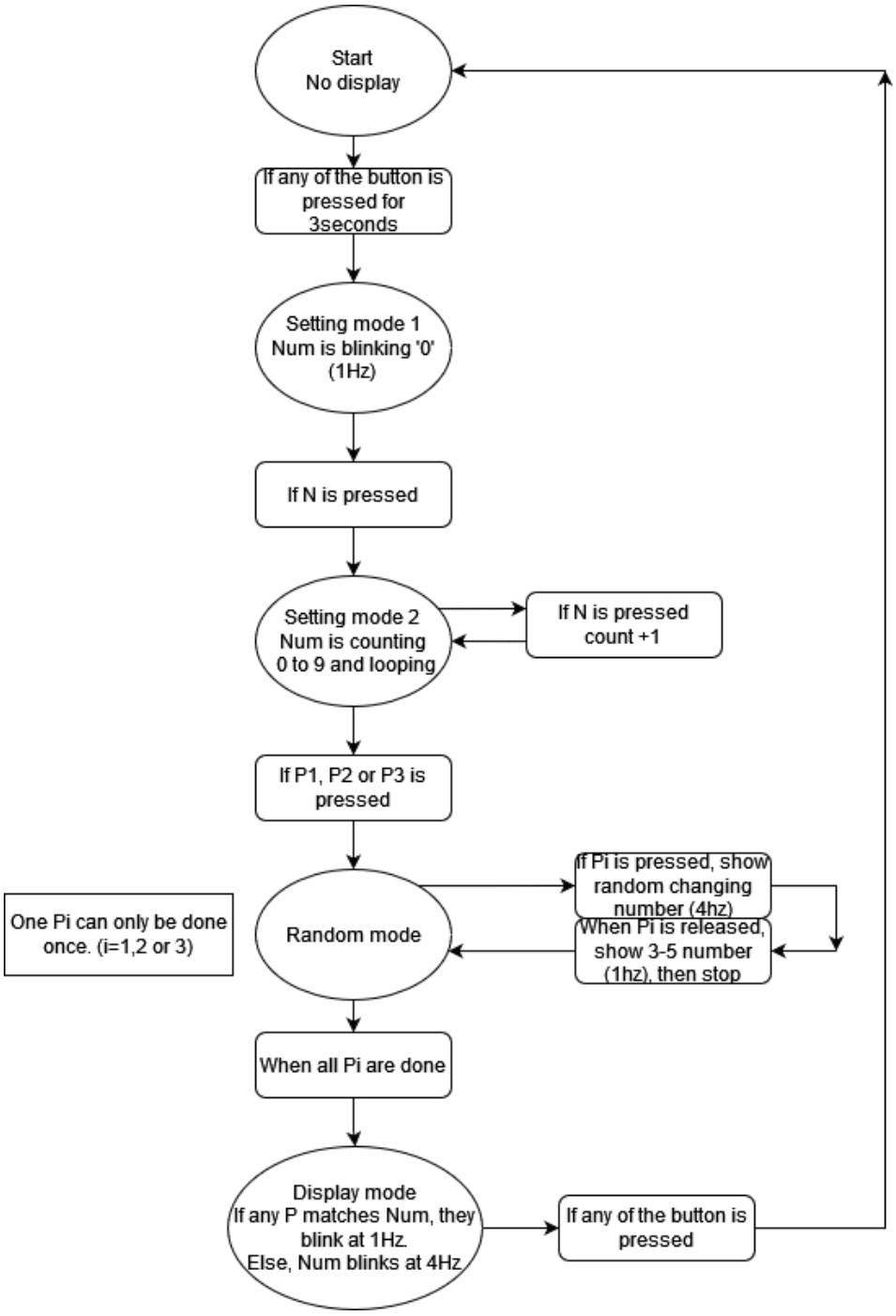
01205241 Digital Circuits and Logic Designs

Mini Project 2 : Sequential logic circuit design

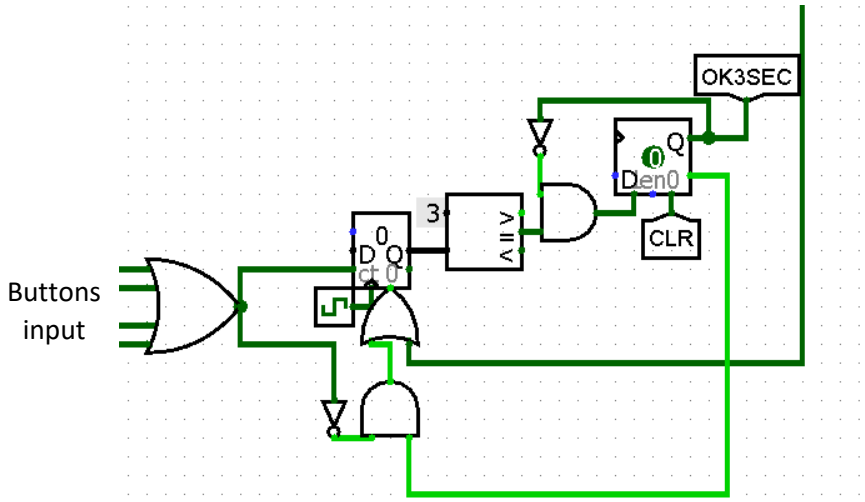
First, we draw the state diagram of our circuit :



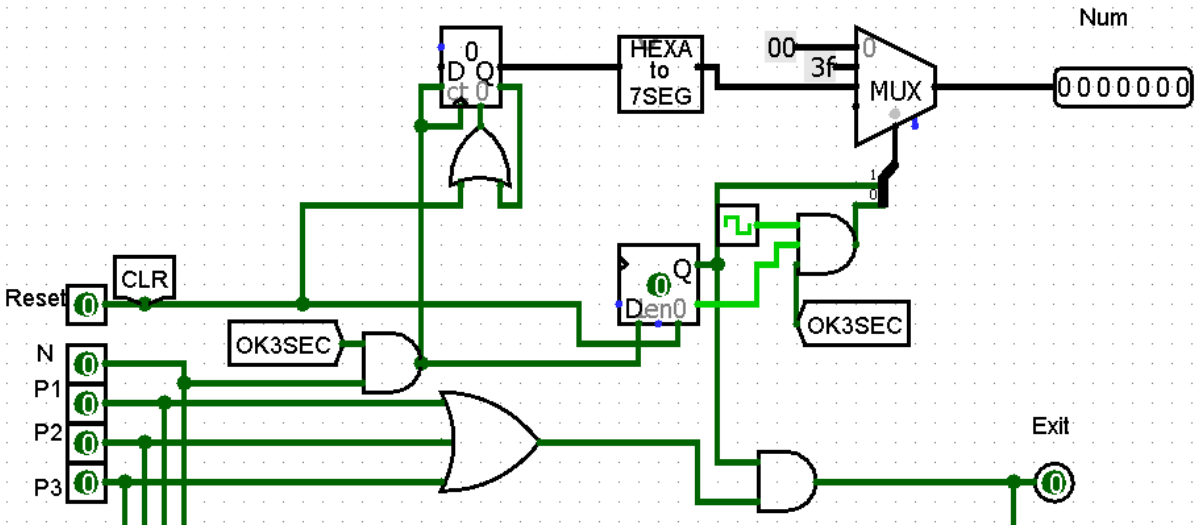
We know that we'll have to use clock for our circuit to make the display blink and count time. We set the simulation clock at 8Hz, so we can achieve both 4Hz blinking and 1Hz blinking :

- For 4Hz blinking we use a clock that takes 1 tick for high duration and 1 tick for low duration. After 8 ticks, it will have blinked 4 times.
- For 1Hz blinking we use a clock that takes 4 ticks for high duration and 4 ticks for low duration. After 8 ticks, it will have blinked 1 time.

Knowing this we can start designing the first stage of our circuit : detecting if a button is being held down more than 3 seconds. For this we use a counter, that only counts when a button is being held down, and resets if it releases before reaching 3 counting. Once it reaches 3, it validates the input by switching a D-FF to 1 asynchronously.

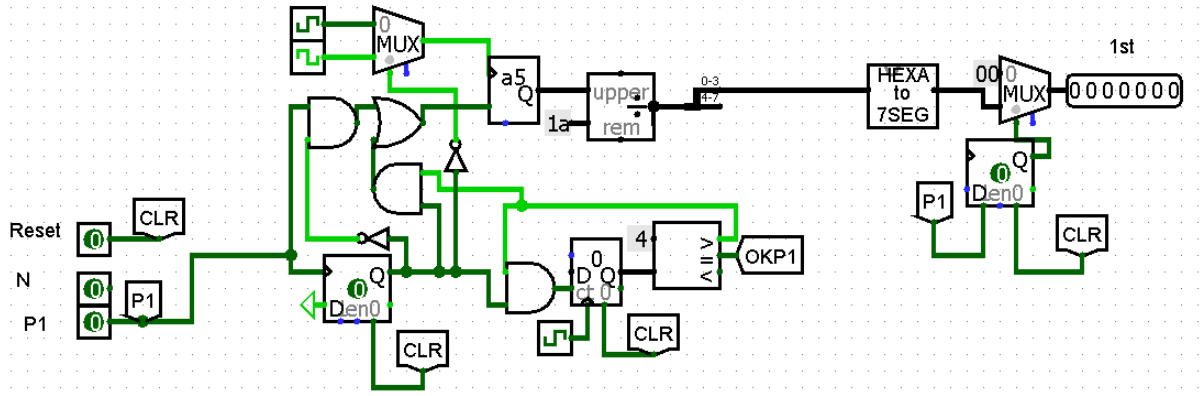


Once the transition has been made (OK3SEC=1), Num blinks 0 and we must detect if N is pressed. If the latter, we stop blinking 0 and start counting, by changing a D-FF asynchronously. For counting we use a counter that wraps around when reaching 10, so it only counts 0 to 9 and back to 0. Also when we get to counting stage, we have to detect whether P1, P2 or P3 is pressed, to go to the next stage (Exit).

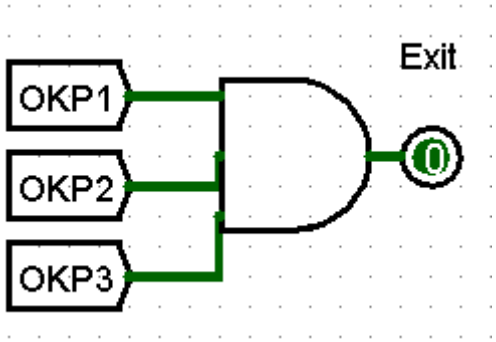


The next stage is random mode. In this stage, once a P button is pressed and held down, the display changes randomly between numbers at a frequency of 4Hz, and when released, it continues changing for 3 to 5 values at 1Hz. Once a P is done it cannot be remade.

To do so, we use a falling edge D-FF. When the button P is held, it allows the random generator to refresh on the 4Hz clock. Once the button is released, it changes the clock selection and starts counting to allow the random generator to change for 4 more clock cycles.

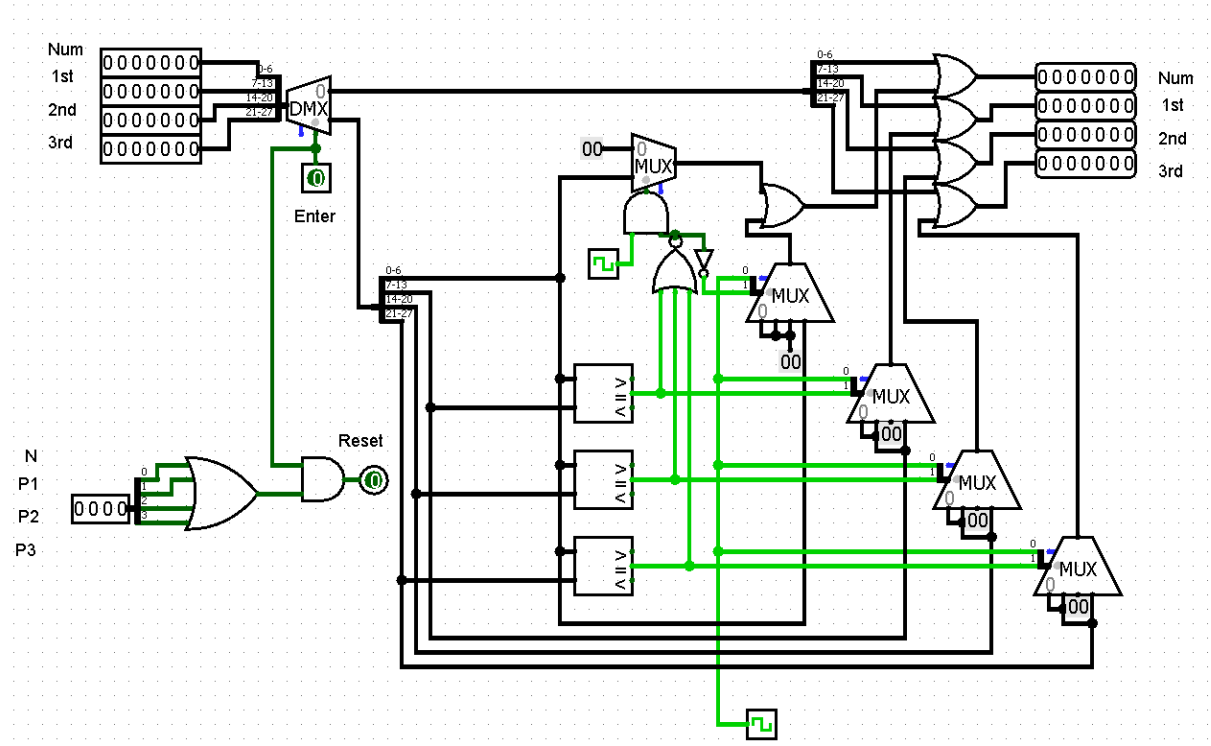


To have a good random generator, we generate a number between 0 to 255, and divide it by 26 and round to the lower. This way we have a fairly shared probability of having a number between 0 and 9. Each random generator used has a different random seed, so they don't generate the same random numbers. To go to the next stage, we have to have finished every Pi.

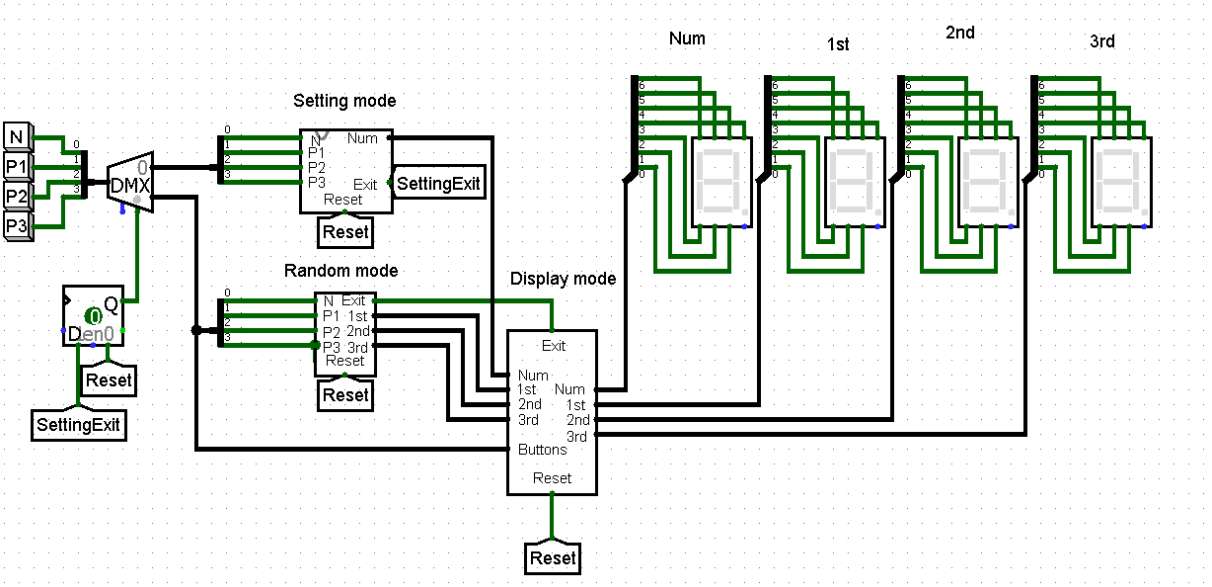


This exit is used in the next stage, Display mode :

When Enter is not triggered by the previous exit, this circuit only takes what is input and outputs it back. When Enter is high, it redirect the output to the blinking mode. There we compare each display with Num, if none are equal to num, it will blink at 4Hz, else, the ones equal to num and num will blink at 1Hz. And we allow the reset to be activated by any button press.



The whole circuit together is built this way. Having Setting exit allows to change the input into the Random mode, then exit of random mode activates the display mode. When the reset is activated in the display mode, it resets everything.

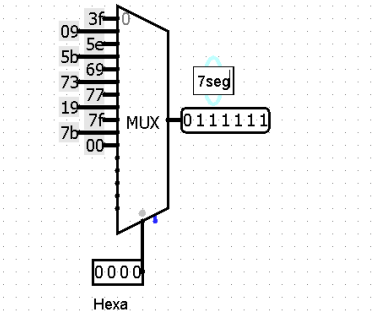


This is the total number of devices used in this project :

Component	Library	Simple	Unique	Recursive
Setting mode	Projet2	1	1	1
hexa to 7seg	Projet2	0	4	4
Random mode	Projet2	1	1	1
Display mode	Projet2	1	1	1
Splitter	Wiring	7	19	19
Pin	Wiring	0	29	35
Tunnel	Wiring	6	33	33
Clock	Wiring	0	13	13
Constant	Wiring	0	28	61
Power	Wiring	0	3	3
NOT Gate	Gates	0	9	9
AND Gate	Gates	0	17	17
OR Gate	Gates	0	13	13
NOR Gate	Gates	0	1	1
Multiplexer	Plexers	0	13	16
Demultiplexer	Plexers	1	2	2
Divider	Arithmetic	0	3	3
Comparator	Arithmetic	0	7	7
D Flip-Flop	Memory	1	9	9
Counter	Memory	0	5	5
Random Generator	Memory	0	3	3
Button	Input/Output	4	4	4
7-Segment Display	Input/Output	4	4	4
Label	Base	7	37	37
TOTAL (without project's subcircuits)		30	252	294
TOTAL (with subcircuits)		33	259	301

The problems encountered during this project were many :

- I could not work my way around a classic method of designing this project. I mainly used the state diagram and used trial and error to design each state and transition. Thus, why each subcircuit is a spaghetti mess.
- I had to use classic 7seg display. While the hex 7seg display is very handy, it does not allow to display nothing. If I want to turn it off, it will display '-'. Therefore, I use Hexadecimal to 7 segment converters, which allows to use the counter and random generator in hexadecimal, and then display in classic 7 segment.



- When blinking 0, if you want to press N to stop it, it will directly go to 1, whereas if you press any of the P, it stops blinking and stays at 0.
- Finally, I used buttons and not pins as input to make sure two are not pressed at the same time, this is easier and avoids any unwanted input that could break the circuit.