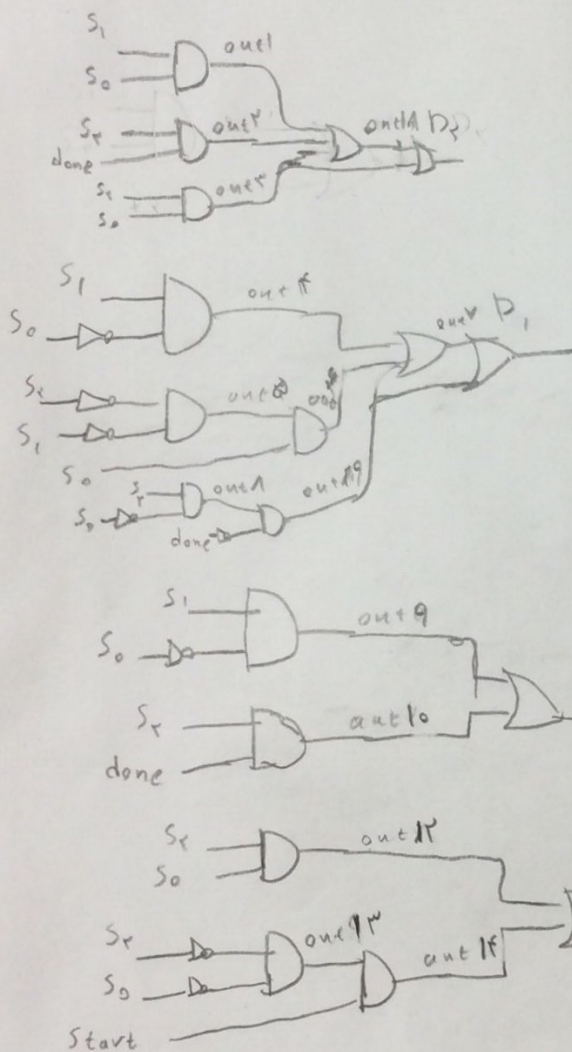
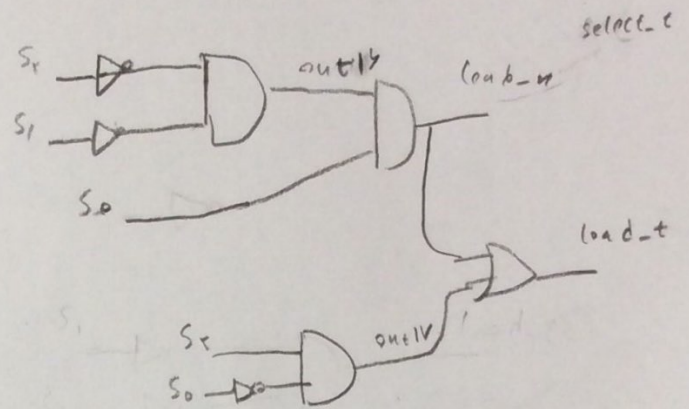
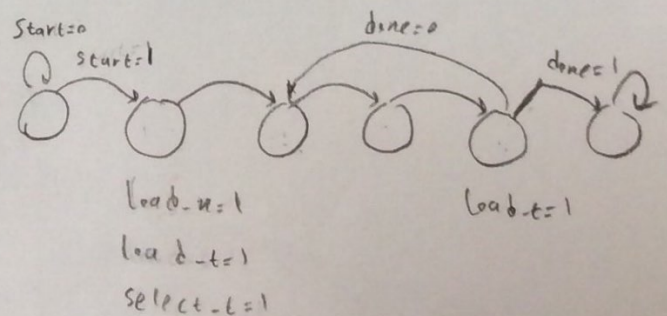


$S_r S_i S_o$	load_w	load_t	select_t
000	0	0	0
001	1	1	1
010	0	0	0
011	0	0	0
100	0	1	0
101	0	0	0



A B C	P	E	D_r, D_i, D_o
$S_r S_i S_o$	start	done	
000	0	-	00 b
000	1	-	00 1
001	-	-	010
010	-	-	011
011	-	-	100
100	-	0	010
100	-	1	101
101	-	-	101



CAD Project 2

Mohammed Reza Ahvi
810100253

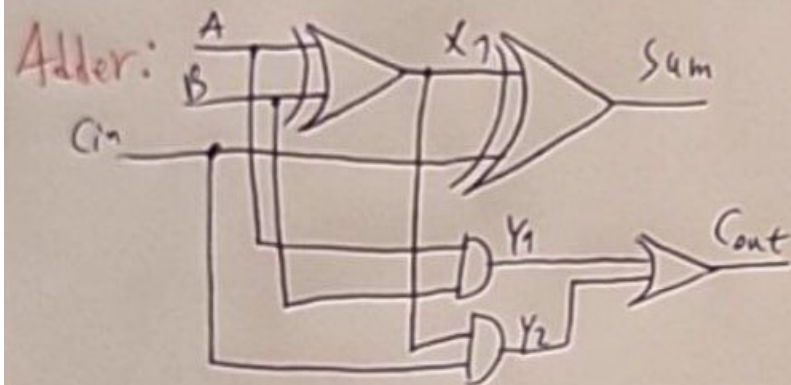
Kasra Noorbaks
810100230

Lets break throw the parts of datapath:

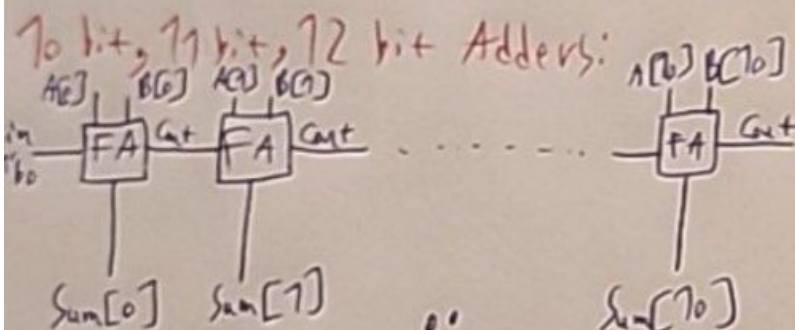
we have 8 5-bit registers with enable that 4 of them are used to be the inputs of PU and 4 of them for final answer. first using Actel modules we wrote the code for 1-bit register with enable and then cascaded it.

we have 4, 5-bit 2to1 MUX's and 1, 5-bit 4to1 MUX.

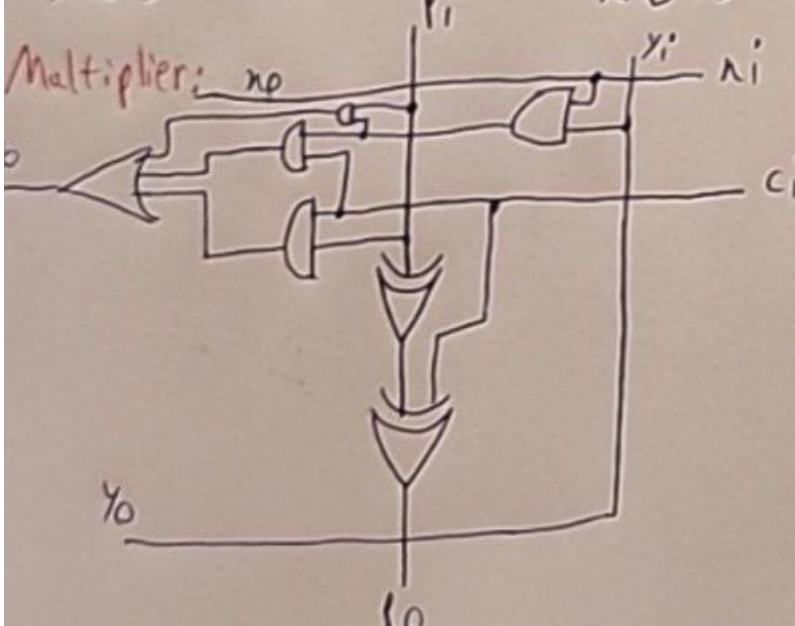
for checking that if a Neuron has become 0 we have 4, 5-bit Or gates.



using this logical gates, we implemented a 1-bit full-adder. we wrote the code of and, or gates with Actel modules and for Xor we used 4 NAND gates.



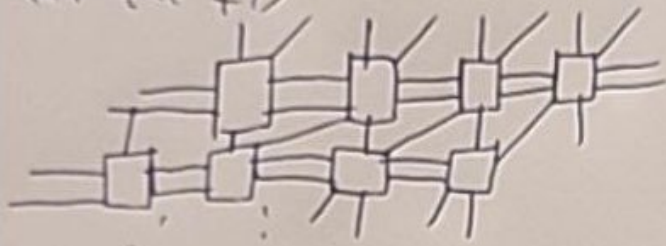
whenever we needed a 11 or 12 bit adder we cascaded out 1 bit full adder to make it!



Using this logic for bit by bit Multiplication we wrote down the code for an 5-bit array multiplier.

in multiplication we first calculate the 2's complements of our 2 inputs and then we have a Max that decides to be the original input or 2's complement of it

After multiplication, if both inputs were + or -, we just output our result, but if the result were -, we output the 2's complement of the result. (2's complementing is offered by a not gate and an adder for the +1)



the schematic of array multiplier is shown.

Encoder: this module takes the result of 4-5-bit or gates and calculates the select of our 4-to-1 Mux.

cd	ab	00	01	11	10
00		0	0		
01		1			
11					
10					

out1

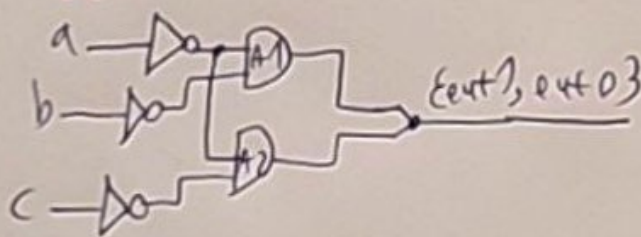
$$\text{out1} = \bar{a} \bar{b}$$

cd	ab	00	01	11	10
00		1			
01					
11					
10					

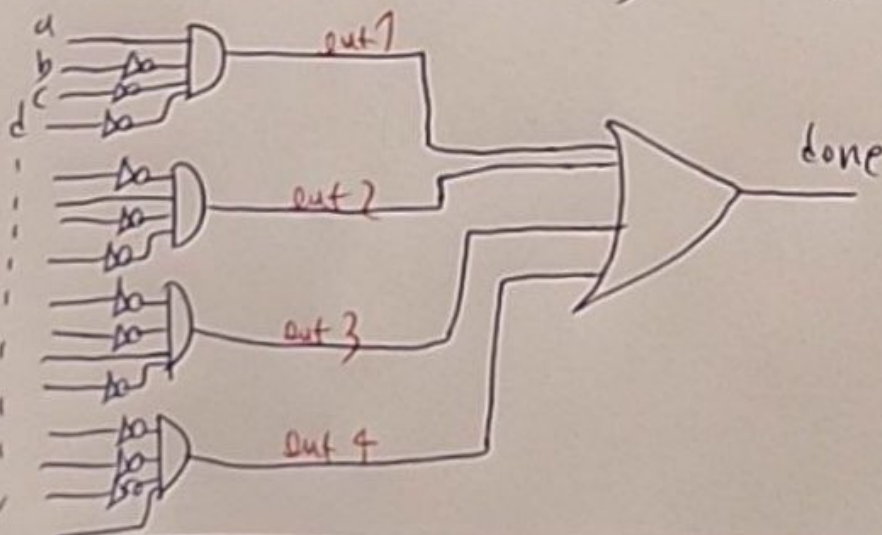
out0

$$\text{out0} = \bar{a} \bar{c}$$

(we wrote the code for all logical gates with Actel modules.)



done checker: this module determines that whether the maximum number is found or not using this logic:



when we multiply two 5-bit numbers the result is 10-bit.
when we want to add these two 10-bit numbers, we first must sign extend them because the carry-out bit!

(CPU modules also have 4 registers R1-R4)

goes back to first ALU

The diagram illustrates a 4-to-1 multiplexer circuit with a parity checker. On the left, four data inputs labeled x_1, x_2, x_3, x_4 are connected to the select lines of a 4-to-1 MUX. The MUX has four data inputs labeled 00, 01, 10, and 11. The output of the MUX is labeled $out/4t$. This output is connected to a block labeled 'parity checker'. The parity checker has a 'done' output. The circuit also includes four comparators (A) and four processors (PU). Each comparator A is connected to the output of the parity checker and one of the data inputs x_1, x_2, x_3, x_4 . The processors (PU) are connected to the outputs of the comparators and the data inputs. The entire circuit is labeled 'data path:' at the top left.