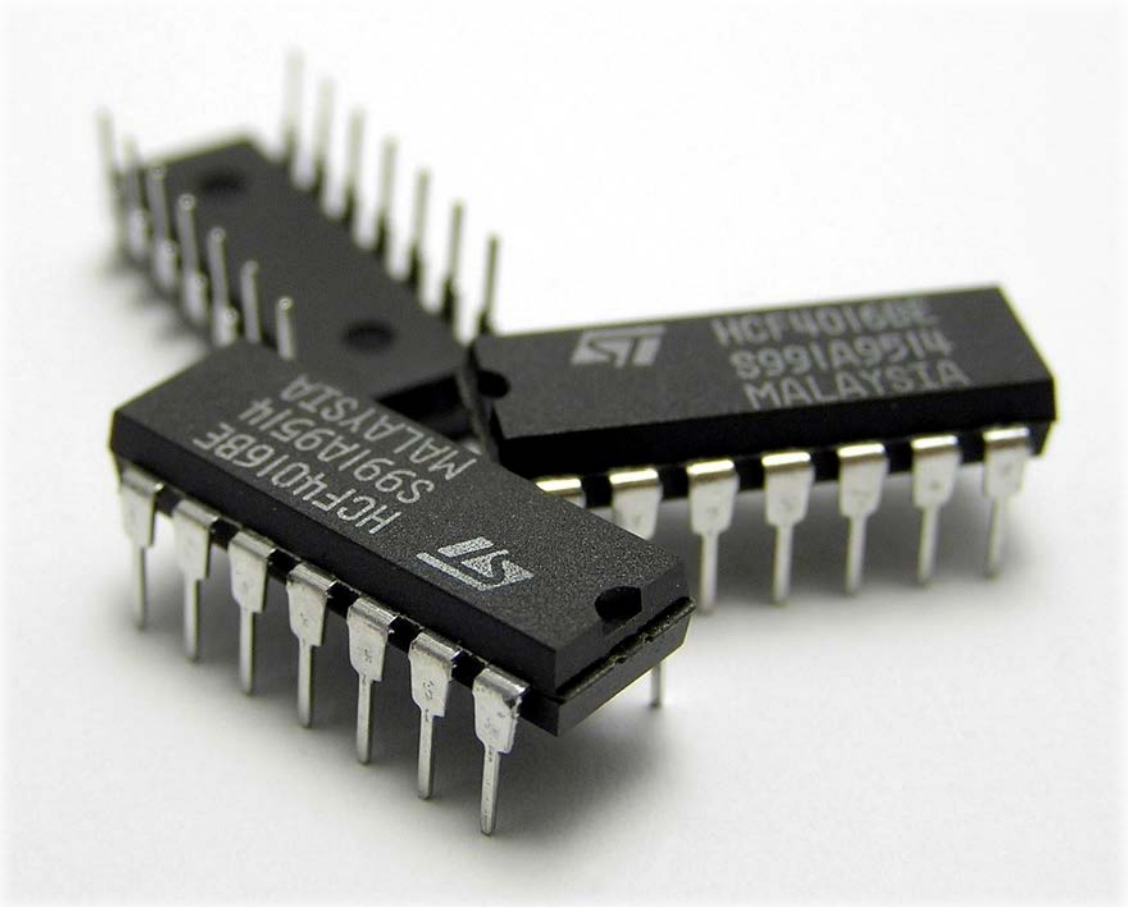


ALU Report

DETAILED VERSION

anas khedr | CSE324-S16 | April 29, 2016

Introduction:

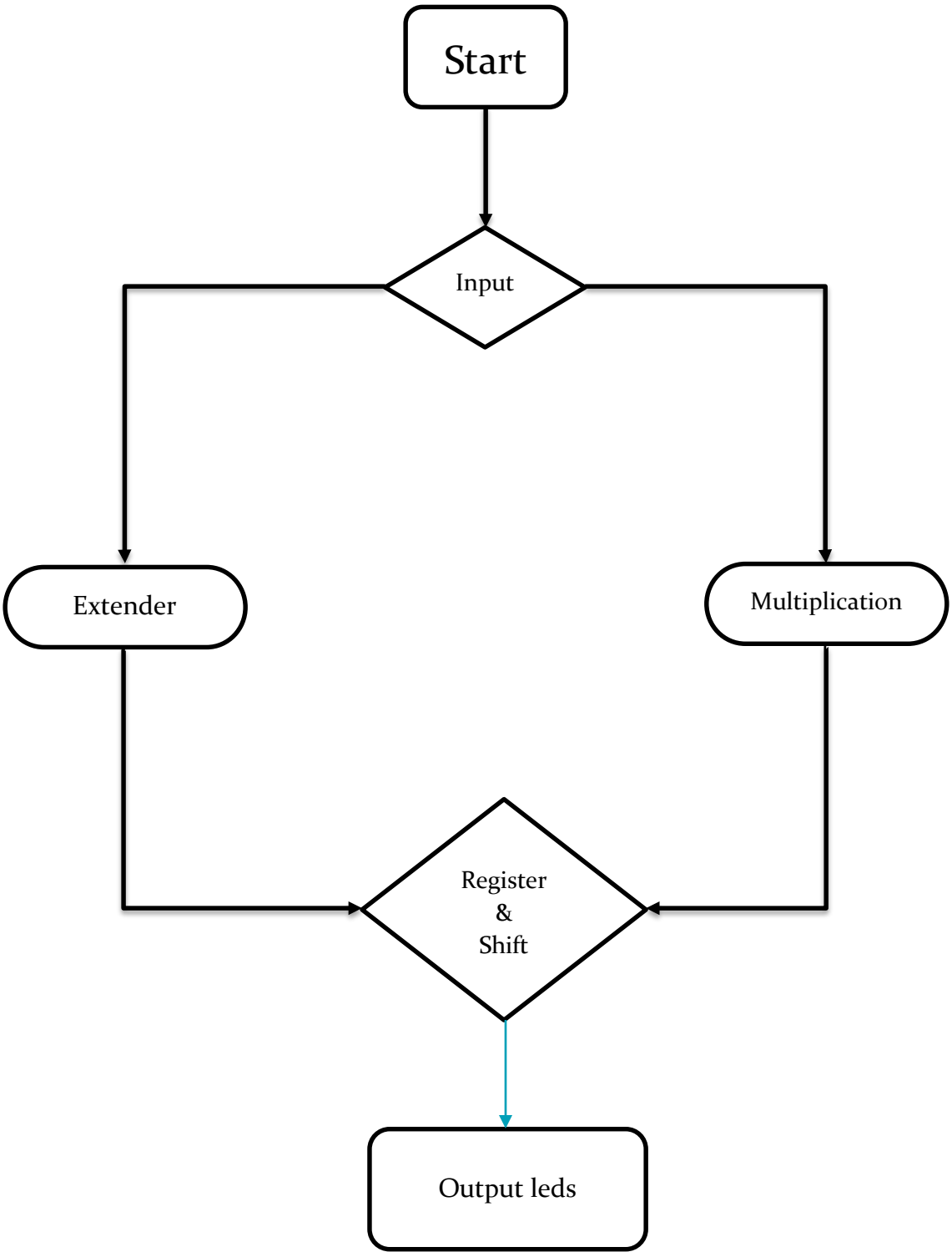
- The 4-BIT ALU presented in this report is one multiple designs we worked on.
- All the details explaining the operation concept of every function will be covered.
- Comparison between all other designs will also be covered.
- All the designs were made using Proteus 7 and 8 (mostly 7)

Primary version (V3):

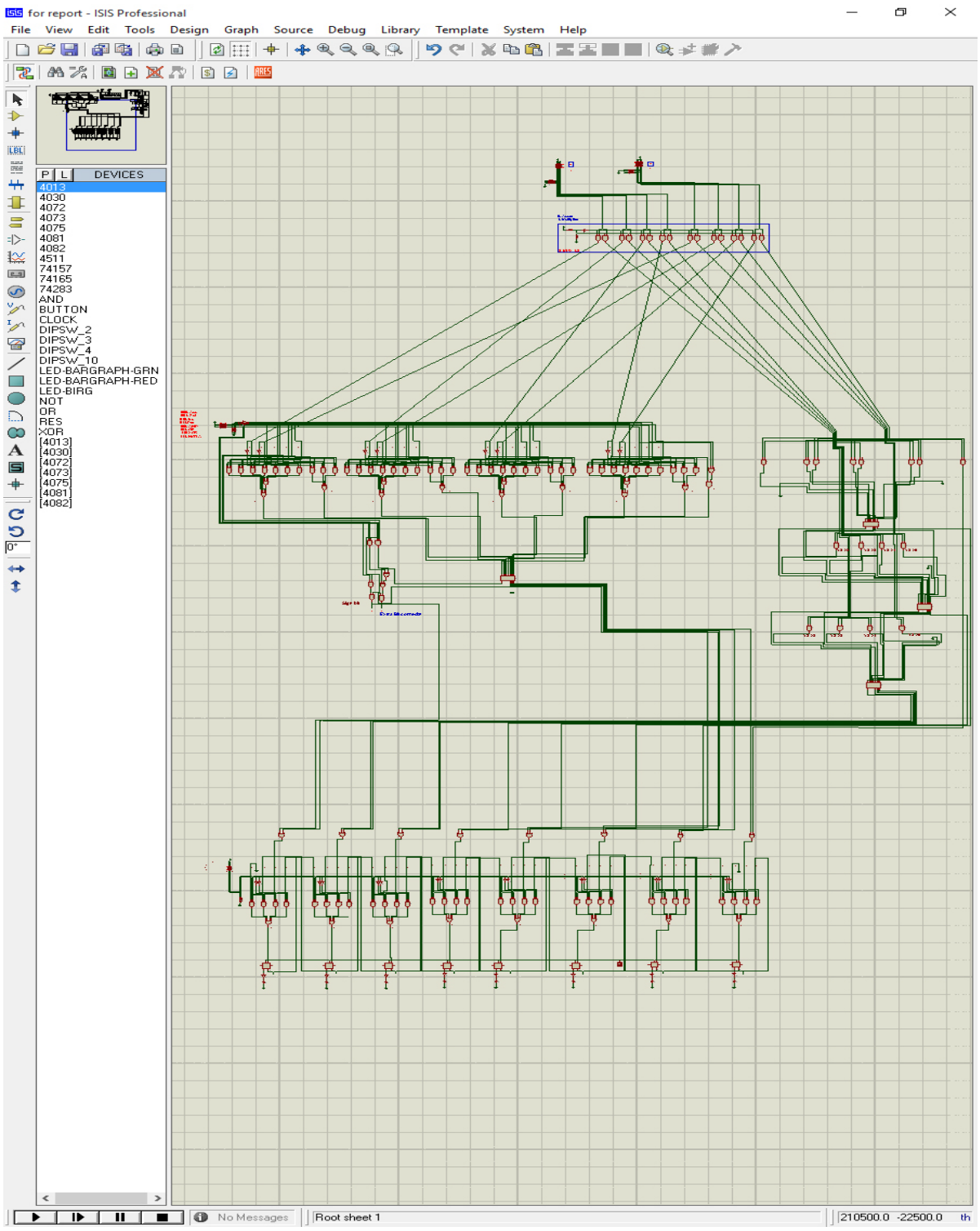
- Like any other version this one has the flowing operation.
    1. Add & Subtract
    2. Logic: AND, OR, XOR, NOT.
    3. Multiplication.
    4. Logic shift right and Logic shift left.
    5. Memory register.

Add, Subtract and Logic operations are implemented using extender.
Multiplication unite works independently from the rest of the ALU
The input is directed to either of the two unites using a multiplexer

Flow chart:


Start
Input
Extender
Multiplication
Register
&
Shift
Output leds

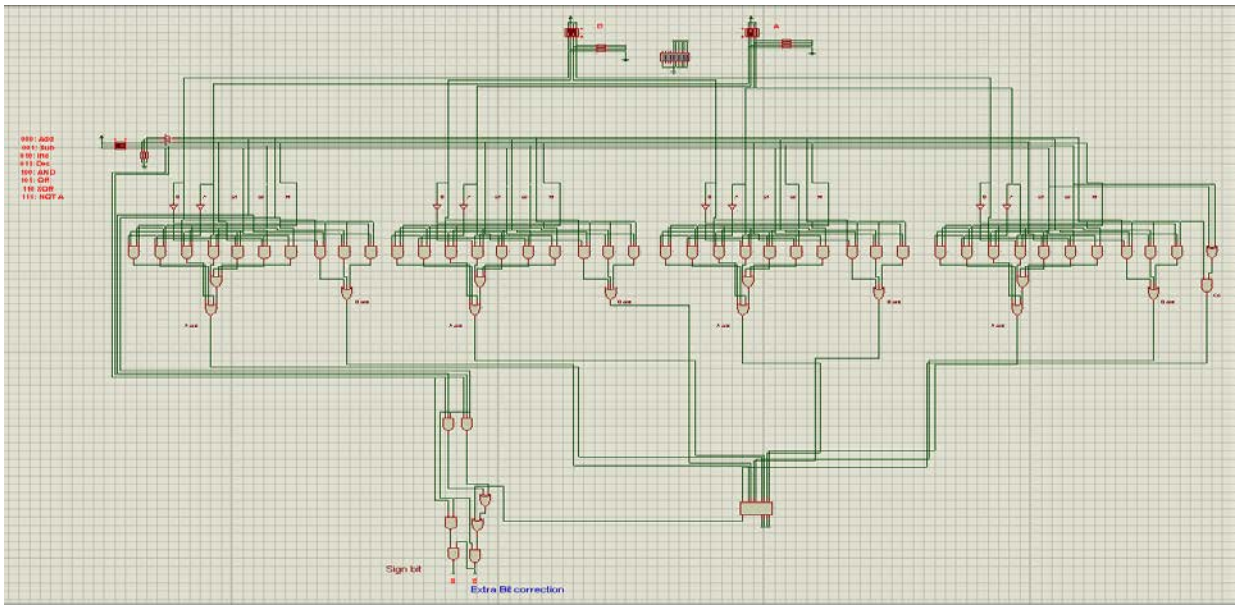
The complete design:



The extender

The extender overview:

- The extender can do the following:
  .1. Add & Subtract
  .2. Increment and decrement
  .3. AND & OR
  .4. XOR & NOT(A)
- There are 8 different operations that the extender can do, so we need 3-BITs to choose among them.
- The 3-BITs are M(choose between Logic&Addtion), S0 and S1
- Here is how the design looks (for better view refer to the design file, Name: extender and adder(Full Adder IC))



The extender's truth table:

M	S0 S1	A B	Aout Bout	C0	Function
0	00	00	00	0	
0	00	01	01	0	ADD
0	00	10	10	0	(A+B)
0	00	11	11	0	
0	01	00	01	1	
0	01	01	00	1	SUBTRACT
0	01	10	11	1	(A-B)
0	01	11	10	1	
0	10	00	00	1	
0	10	01	00	1	INCREMENT
0	10	10	10	1	(A+1)
0	10	11	10	1	
0	11	00	01	0	
0	11	01	01	0	DECREMENT
0	11	10	11	0	(A-1)
0	11	11	11	0	
1	00	00	00	0	
1	00	01	00	0	AND
1	00	10	00	0	(A.B)
1	00	11	10	0	
1	01	00	00	0	
1	01	01	10	0	OR
1	01	10	10	0	(A+B)
1	01	11	10	0	
1	10	00	00	0	
1	10	01	10	0	XOR
1	10	10	10	0	(A⊕B)
1	10	11	00	0	
1	11	00	10	0	
1	11	01	10	0	NOT
1	11	10	00	0	(~-A)
1	11	11	00	0	

- A and B are the input lines for the extender
- Every 2 bits (A and B) have their own extender.
- Since were working with 4 bits , then we'll be needing 4 extenders.
- The idea of the extender is to manipulate the input bits to get an output that will satisfy the operation we want to do if applied to a Full Adder.
- Say we want to subtract A-B (code: 001) then the Full Adder will need the number A as it is but we'll need to NEG the number B _which we do _.

- Aout and Bout are the manipulated output which will satisfy the required operation.
- Most of the truth table were solved using Karnaugh map online solver url: http://www.32x8.com/ and verified by hand.

The Multiplier

The multiplier overview:

- The Multiplier uses multiplication by addition.
- Note the following:

			X3	X2	X1	X0	×	First number
			Y3	Y2	Y1	Y0		second number
			Y0*X3	Y0*X2	Y0*X1	Y0*X0	+	L1
		Y1*X3	Y1*X2	Y1*X1	Y1*X0			L2
	Y2*X3	Y2*X2	Y2*X1	Y2*X0				L3
Y3*X3	Y3*X2	Y3*X1	Y3*X0					L4

- This is how the multiplication is done.
- We multiply 2 bits at a time and then we add them together.
- We can multiply 2 bits using 2 input AND.
- As for the addition, we use full adder
- There are 4 levers of multiplication output that need to be added so we use 3 Full Adders.
- 3 full adders:

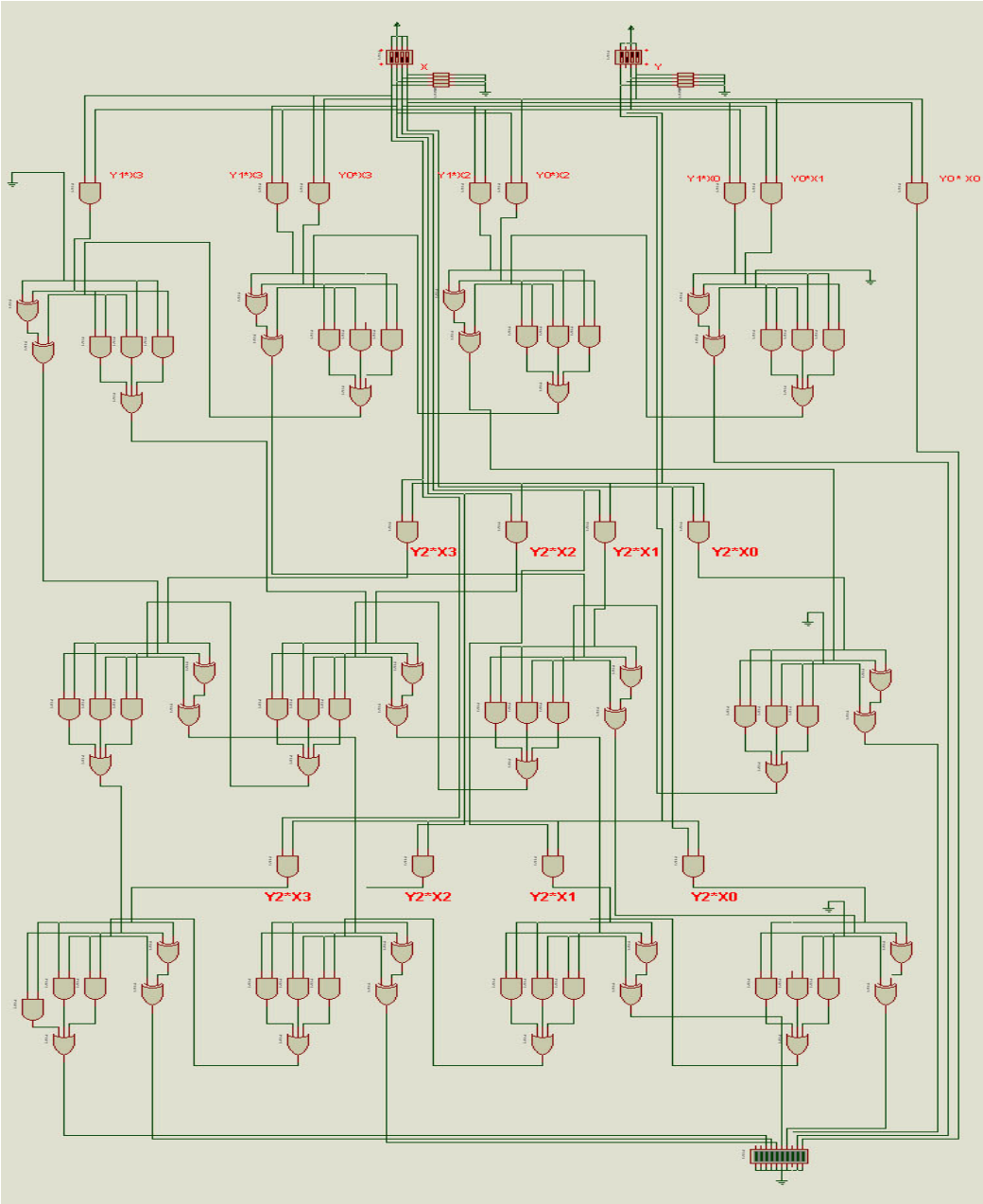
  .1. Adds L1 + L2

.2. Adds L3+ L4

.3. Adds the output of 1 and 2

- Note that there might be carry from the last addition
- This carry is a higher order (12 ➔ the 1 is higher than 2).
- So the last carry of the first Full Adder is added as Co for the next full adder.
- The max output will be represented in 8-Bits, so we need to modify the output of the extender to be represented in 8-Bits as well.
- We use a small circuit (for the extender) to determine if the output is negative or positive and based on it's finding , it fills the extra bits.

The design:



for better view refer to the design file, Name: Multi 4x4 (working)

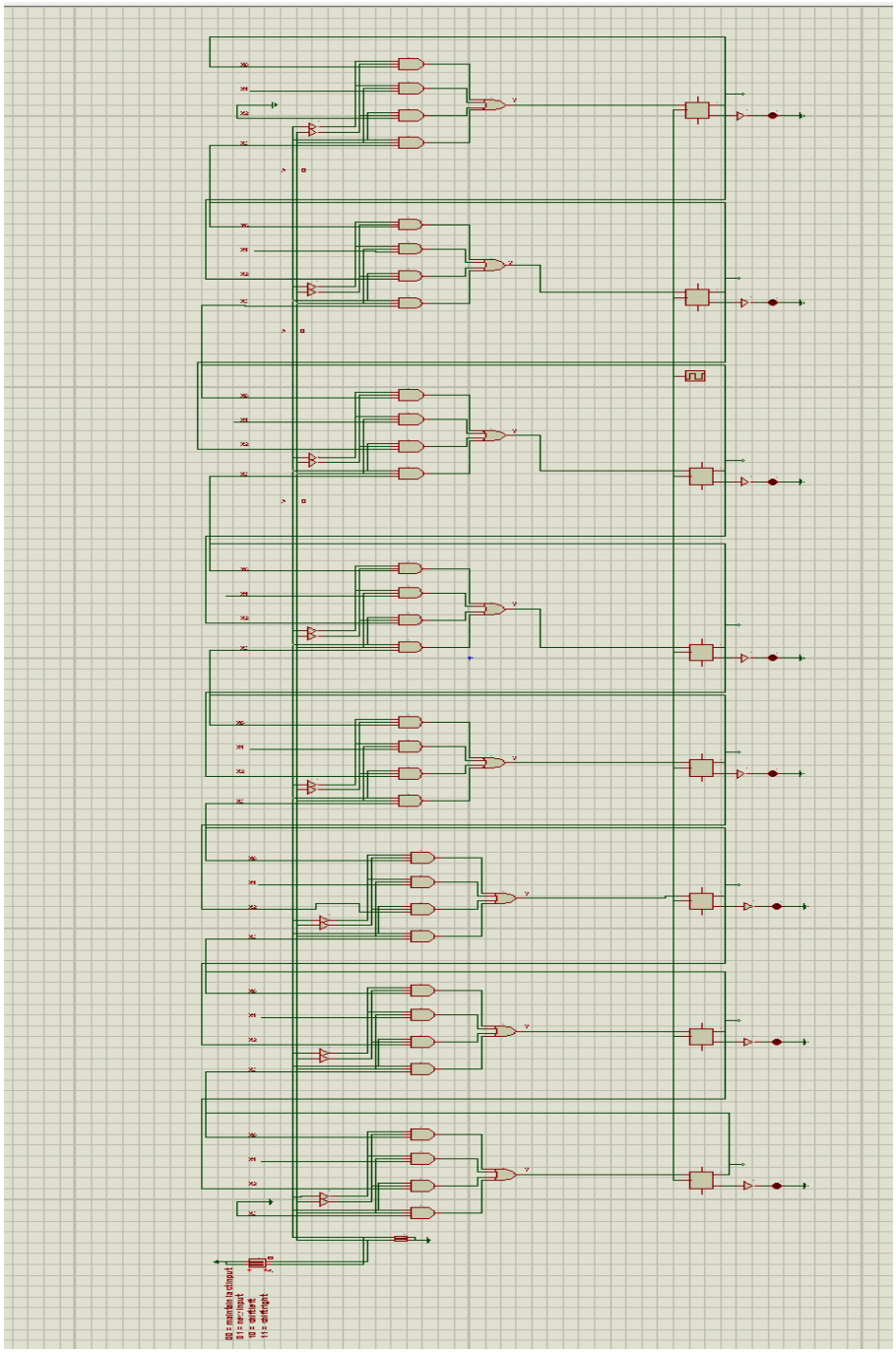
The Register

Registers overview:

- The registers can do the following:
  .1. Logic Shift right
  .2. Logic Shift left
  .3. Store value
- The registers consist mainly of flib-flobs.
- We use additional multiplexers in order to perform the logical shift in the number stored in the register.
- We use a slow clock signal (1 HZ) seeing that we are humans and we won't be able to see the changes if the clock is typical 10^6 HZ (1 MHZ).
- A and B are the operation selectors

A	B	FUNCTION
0	0	maintain last input
0	1	new input
1	0	shift left
1	1	shift right

The design:



For better view, refer to the design file.
Name: Registers.

The comparison

V3:

- All the above designs are from V3
- V2 is the final design that we're making its hardware.
- So V3 is the best design?
  No, not by a long shot (V1 is the best)
- What about all the other design

OK, let's dive into it

- Before I start listing pros and cons you need to understand that we were gonna use a power saving mode feature.
- The idea was to place a transistor between ever IC's Vcc (or ground) pin and the power supply.
- A small circuit was supposed to choose when to turn the transistor on or off, and thus cut the power completely from parts of the circuit.
- This feature was gonna enable us to save tremendous amount of unnecessary power dissipation (relatively speaking).
- Off course the tradeoff is the performance.

why didn't we implement this idea?

- This idea can only be implemented in V1(which was the initial design) seeing how logical and arithmetic operation are separated.
- If we were to use the arithmetic operation, we would have cut the power off from the logical and the multiplication circuit.
- But in V3 the arithmetic and logical are now tied together with the extender.
- If we cut the power off from a single IC, then will have illegal output.
- Even without power saving mode, V3 will still consume more power than V1 seeing that V3 has more ICs than V1.
- The main reason that V3 has more ICs is the extender.
- The extender requires more transistors (ICs) than the separated logical and arithmetic circuit.

- Conclusion: V1 is better than V3 in every way I can think of, less power dissipated (power saving mode can also be implemented to improve the power consumption even more) and less ICs (cheaper).
- Then why did we go with V3 when we knew the original design was better?
- Our professor suggested we use extender to reduce the power consumption because V1 was consuming too much power, our counter proposal was to move the extender up to be directly after the input numbers (to prevent the other circuits from switching states and consume power due to their transistor nature) and limit the unnecessary power dissipation or we could use a power saving mode.
- The extender was actually a lot worse than we anticipated and there was no way to get less power dissipation, at least not to my knowledge.

What about the other versions?

- V2 is the same as V3 except that we changed the internal design of the Full Adder in the Multiplier with Full Adder IC.
- V4 and V4_1 were attempts to merge the multiplication with the extender, V4_2 was the successful one. But is was even worth than V3 in the power consumption and the cost.
- V5 was a desperate attempt to reduce the cost of V4_2 by replacing some AND gate with a tri-state buffer.
- All other versions were attempts in a certain direction but we never completed them.

Because we are big believers of open source, all data relation our project will be posted online for all other students to learn from our mistake, improve or build on our project.

You’re free to use the designs as you like, no need to ask for permission.

For any enquiry relating the project please don't hesitate to contact me

emails:
anaskhedr47@gmail.com
zero_2222_man@yahoo.com

Credit of the project goes to:

- Yassmine Ahmed abdl-basset
- Amira Ibrahim
- Sultan Ibrahim
- Menna Ashraff
- Anas Ahmed Fouad