Introduction to VLSI – Final Project part 1

4-bit ALU – Planning Stage

Roy Milshtein,

Tsuf Zamet,

Itay Havdala,

Inbal Shalit.

1. Planning Stage

This report documents our detailed planning process, which includes the logical design of the ALU, floor planning considerations, and area estimation. Our goal is to showcase the fundamental principles and techniques used to develop a functional and efficient 4-bit ALU. We will outline the step-by-step approach taken to convert the desired functionality of the ALU into an optimized physical design, enabling successful arithmetic and logical operations on 4-bit data.

1.1. Project I/O

$Input\ Registers$	$Output\ Registers$
$A[3:0], B[3:0], Opcode[5:0]$	$Y[3:0]$

1.2. Opcode Assignment

$Opcode[5:0]$	$Opcode\ Assignment$		$Notes$
$X10000$	$Barrel\ Shift\ Right$	$i = 1$	$Y \leftarrow A \gg i$
$X10001$	$Barrel\ Shift\ Right$	$i = 2$	
$X10010$	$Barrel\ Shift\ Right$	$i = 3$	
$X10011$	$Barrel\ Shift\ Right$	$i = 4$	
$X00001$	$Bitwise - XOR$		$Y = A \bigoplus B$
$X00011$	$Bitwise - AND$		$Y = A \cdot B$
$X00010$	$Bitwise - OR$		$Y = A + B$
$X00100$	ADD		$\langle Y \rangle = \langle A \rangle + \langle B \rangle$
$X01100$	SUB		$\langle Y \rangle = \langle A \rangle - \langle B \rangle$
$X00000$	NEG		$\langle Y \rangle = -\langle A \rangle$
$MSB = 0$	$Select$ A		$A[3:0]$
$MSB = 1$	$Select\ Feedback$		$A[3:0] \equiv Y_{n-1}[3:0]$

1.3. Block Design and Methodology

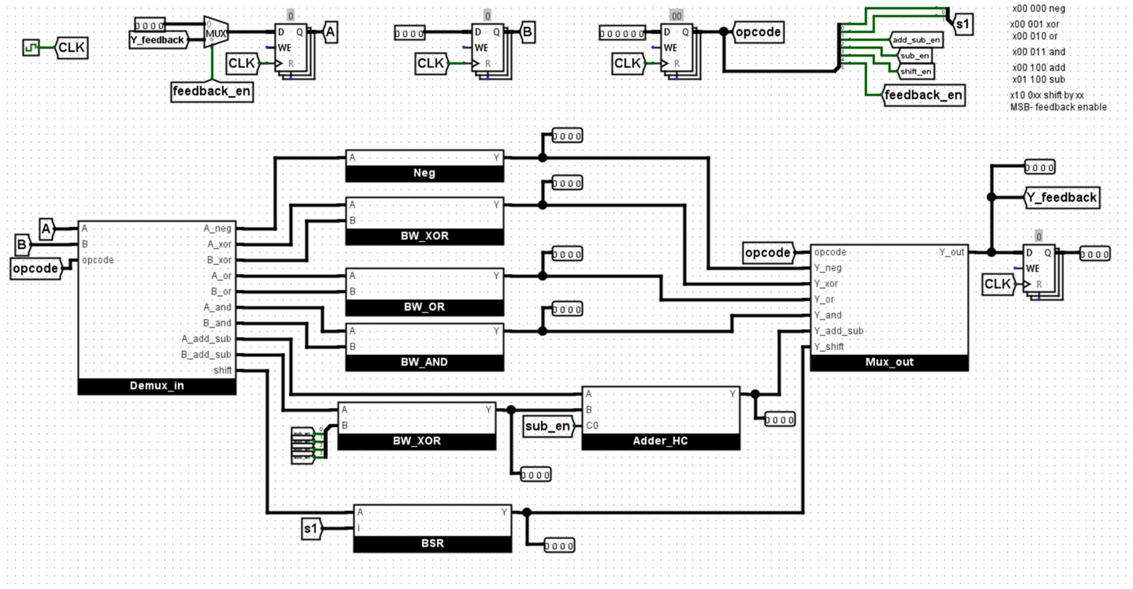
1.3.1. <u>High-level block design methodology</u>

The high-level design is based three main sections:

- The “$Input$” section, connects between the inputs A and B to the desired arithmetic/ logic block determined by the $opcode$.
- The “$Logic$" section, which contains all the ALU blocks: $ADD, NEG, BSR, OR, AND, XOR$.
- The "$Output$" section, getting the result from the relevant ALU block and determines the $feedback$.

1.3.2. <u>ALU top block diagram</u>

The following diagram represents the ALU top block which includes the inputs/outputs registers of the circuit, feedback, and clock as well as the combinatorial logic of the circuit. The combinatorial logic used to implement the $Logic$ section’s blocks will be shown in later sections.

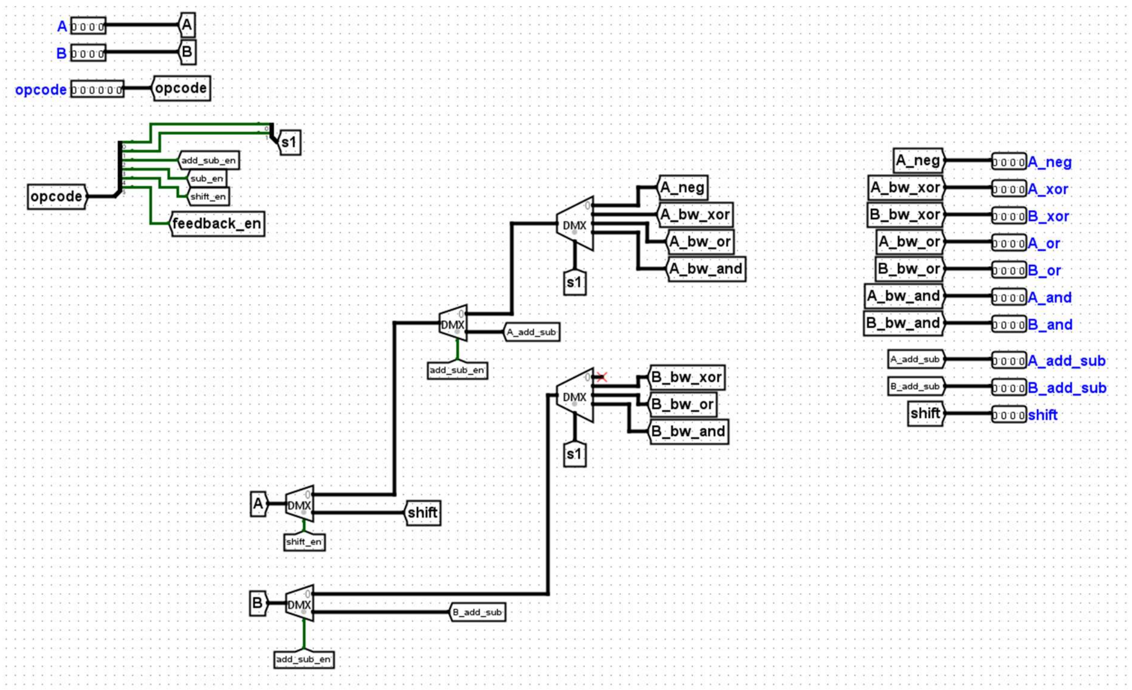


$D.1.$ $ALU\ top\ block\ diagram$

1.3.3. Detailed hierarchy design

- *"Input" Section* – consist of two 4-bit registers for A and B inputs, 6-bit register for the opcode, 4-bit $MUX2:1$ for the feedback and a block named $Demux\_IN$.
  Denote that we decided to feed the circuit with the opcode using register, in order to have better control over the calculations and prevent some issues that can occur if the opcode changes during calculations.
  We will use DFF for all registers in the project.
  The $Demux\_IN$ block is meant to navigate the inputs (A and B) to the relevant logic block, according to the given opcode, while inserting zero inputs to all the logic blocks that should not be active in the current operation.

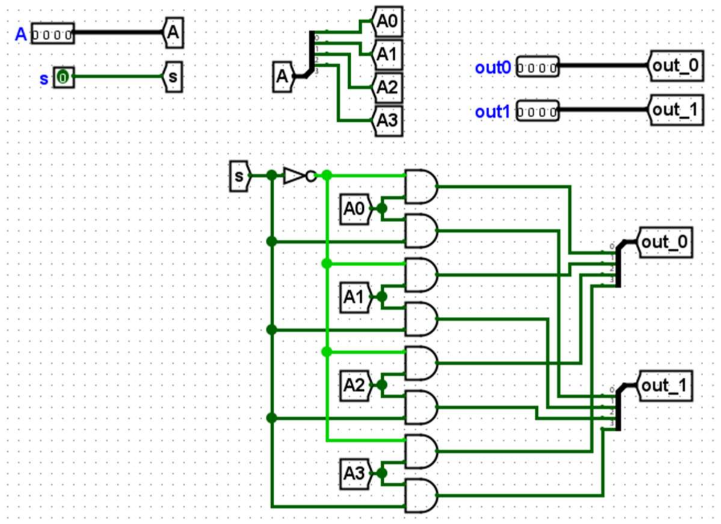
  Logic diagram of $Demux\_IN$ block:



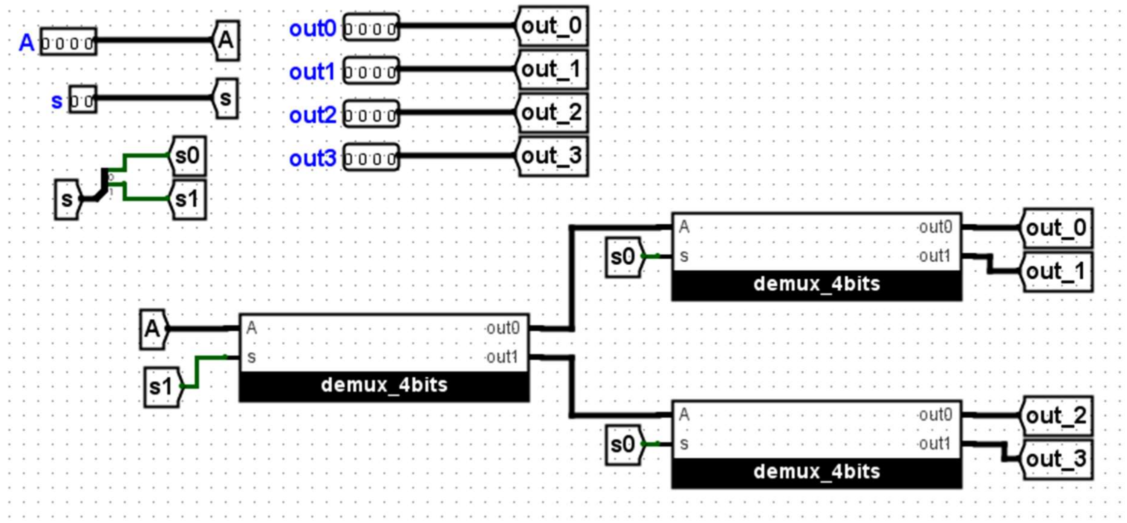
$D.2. Demux\_IN\ diagram$

We built this block using $DEMUX\ 1:2$ and $DEMUX\ 1:4$ gates. The small "DMX" instance in the diagram corresponds to $DEMUX\ 1:2$ and the large "DMX" instance corresponds to the $DEMUX\ 1:4$ gate. Denote that both gates are getting input of 4 bits (implementation of those $Demux$ gates using basic logic gates is shown below).

The implementations of both $DEMUX\ 1{:}2$ and $DEMUX\ 1{:}4$ gates:

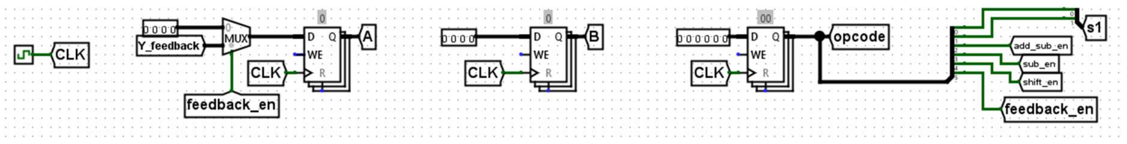


$D.3. DEMUX\ 1{:}2\ diagram\ "demux\ \_4bits"$



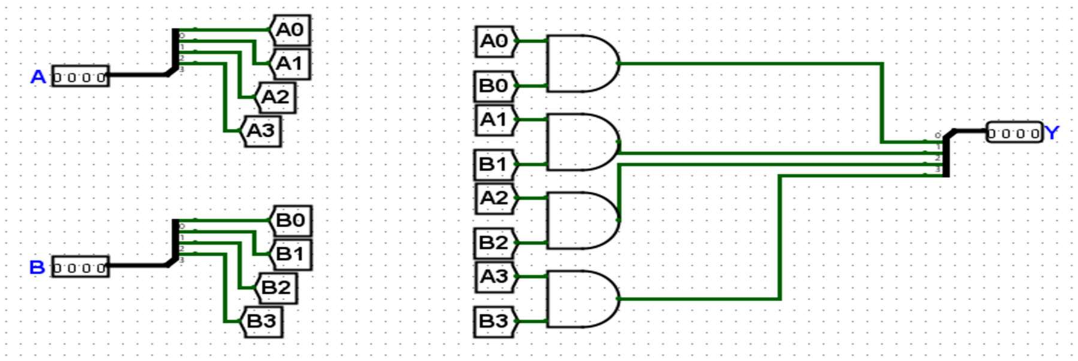
$D.4. DEMUX\ 1{:}4\ diagram$

Input registers logic:

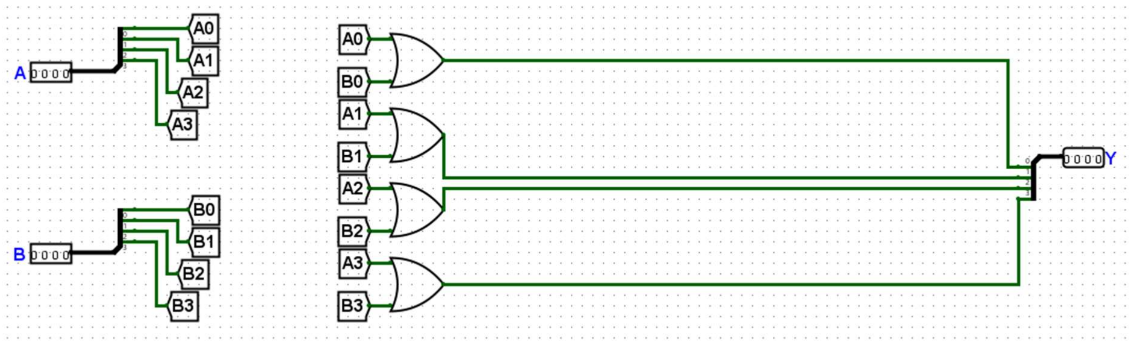


$D.5. input\ registers$

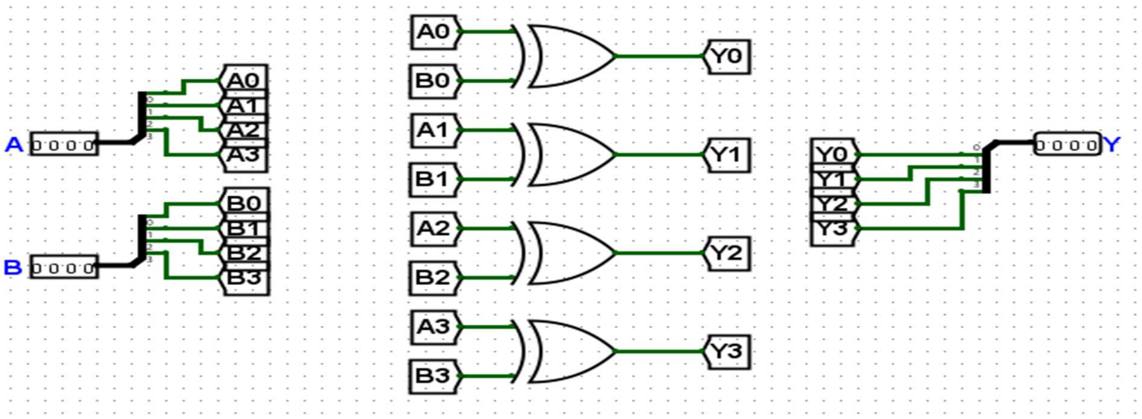
- *"Logic" Section* – consists of all logical operations blocks:
    - Bitwise operations: a simple implementation of the logical bitwise operations - OR, AND, XOR. The following operations will be implemented using a 4-bit parallel cells of the desired logic.



$D.6.$ $Bitwise\ AND$

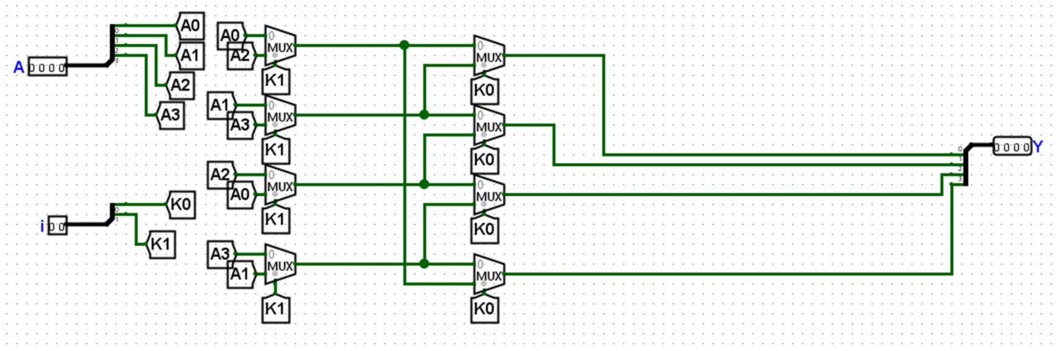


$D.7. Bitwise\ OR$



$D.8.$ $Bitwise\ XOR$

- <u>Barrel shift Right (BSR) -</u>
  We chose to implement this block using 8 $Mux2:1$ gates as shown below:

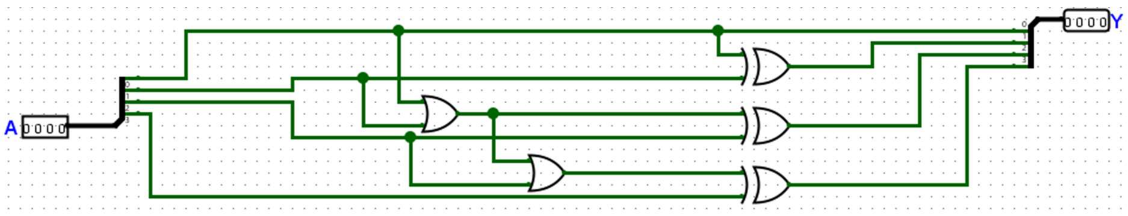


$D.9.$ *Barrel Shift Right block*

  Denote that it's a cyclic shifter, so shift by 4 is the same as shift by 0, which means no change to the input.

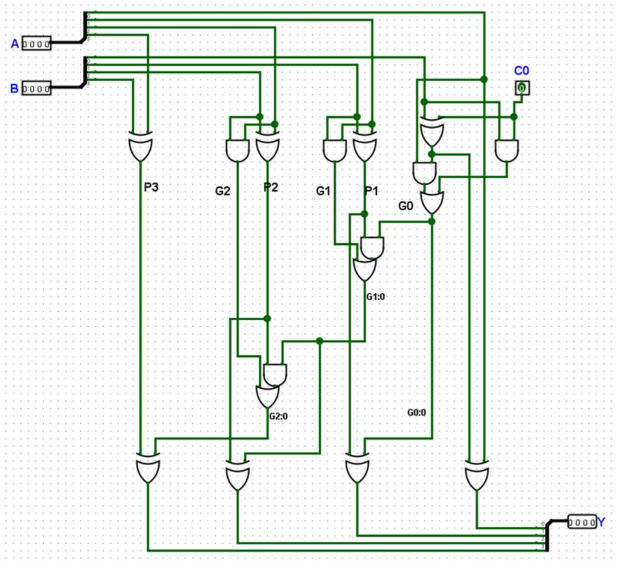
- <u>Neg –</u>

  This block is getting one 4-bit input and returns a binary representation of its negation (in two's complement):



$D.10.$ *Neg block*

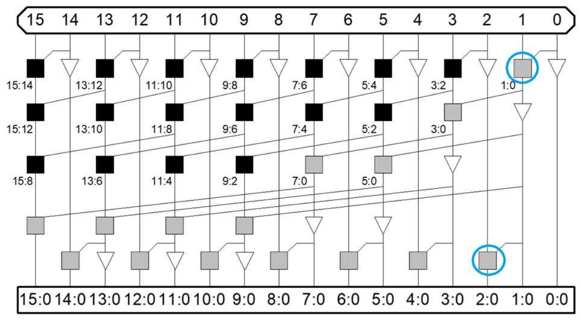
- <u>Adder –</u>
  Our adder is based on Han-Carlson algorithm, with additional logic of carry in in order to support substruction. Some more explanations below the diagram.



$D.11.$ *Han – Carlson adder*

The first part of the circuit is the PG logic. On the right, there are additional few logic gates to support carry in, which allows us to use this circuit also as a subtracter (explained later). The center part of the adder is a group of PG logic according to our algorithm. The section below is the sum logic, which consists of 4 XOR gates.
Our adder isn't supposed to support carry out so we didn't had to use the AND and OR gates which was supposed to be in this section too.

The algorithm we needed to implement is Han- Carlson:

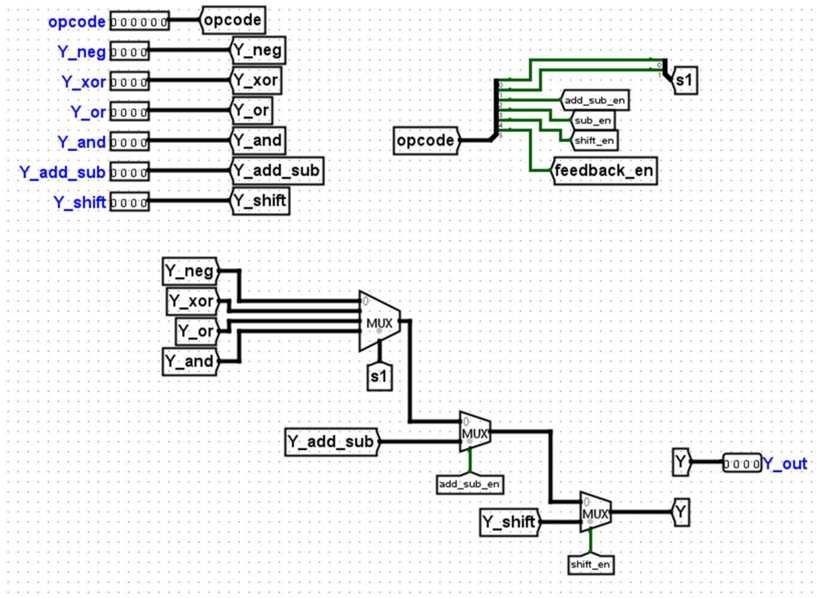


Because we don't have carry out, only the 2 circled cells are relevant, and we implemented them.

In order to use the adder also to preform subtraction, as pointed before, we added some logic adding a carry in bit to the circuit. While using the circuit for addition- the carry in bit will be zero, and for subtraction it will be 1→ the carry in bit is the sub_enable bit from the opcode.
The B input of the adder will be the output of $bitwise\ xor$ between the original B input and sub_enable (extended to 4 bits) in order to flip all bits before substraction ($bitwise\ xor$ with ones) or push original B to the adder for addition ($bitwise\ xor$ with zeros).

➔ Flipping all of B bits and adding 1 (carry in) will allow us to preform <A>+(-<B>) as wanted.

- *Output Section:*
  This section consists of a block named MUX_out and the output register.
  The MUX_out block is getting all logic block's outputs and choses the output of the circuit, according to the opcode (determines the operation preformed).
    - Mux_out diagram:



$D.12.\ Mux\_out\ \ block$

Denote that the small MUX gates in the diagram are $MUX2{:}1$ basic gates, getting 4-bit inputs, and the bigger MUX gate is $MUX4{:}1$ gate, getting 4-bit inputs.

1.4 Cell Number and Area Estimation

To obtain accurate results, we conducted a comprehensive analysis involving cell counting and area estimation. Our approach involved calculating the individual areas of all logic gates and memory gates we want to use. Relevant information is summarized in the table below:

$\boldsymbol{Name\ of\ Gate}$	$\boldsymbol{Width}\ [\mu m]$	$\boldsymbol{Height}\ [\mu m]$	$\boldsymbol{Area}\ [(\mu m)^2]$
$XOR2X1$	1.6	1.71	$\mathbf{2.736}$
$AND2X1$	0.8	1.71	$\mathbf{1.368}$
$OR2X1$	0.8	1.71	$\mathbf{1.368}$
$MX2X1$	1.4	1.71	$\mathbf{2.394}$
$MX2X2$	1.8	1.71	$\mathbf{3.078}$
$MX4X2$	5	1.71	$\mathbf{8.55}$
$DFFQX1$	3.4	1.71	$\mathbf{5.814}$
$INVX1$	0.4	1.71	$\mathbf{0.684}$

$T.1.\ gate\ sizes$

After dividing our schematic into sections using the hierarchical design outlined in the "Detailed hierarchy design" section, we proceeded to estimate the area for each block. The table provided below presents the results of our block area estimation:

Section	**Block**	**Gate**	**# of Gates**	**Gate Area**	**Block Area Estimation**	**Section Area Estimation**
Input	*Input regs*	*DFFQX1*	14	5.814	81.396	198.792
	Demux 1X2 (4 bits)	*AND2X1*	8	1.368	11.628	
		INVX1	1	0.684		
	Demux 1X4 (4 bits)	*Demux 1X2 (4 bits)*	3	11.628	34.884	
	Demux in	*Demux 1X2 (4 bits)*	3	11.628	104.652	
		Demux 1X4 (4 bits)	2	34.884		
	Feedback Mux	*MX2X2*	4	3.078	12.312	
Logic	*ADDER*	*AND2X1*	6	1.368	34.2	97.128
		OR2X1	3	1.368		
		XOR2X1	8	2.736		
	NEG	*OR2X1*	2	1.368	10.944	
		XOR2X1	3	2.736		
	$BW-XOR$	*XOR2X1*	4	2.736	10.944	
	$BW-AND$	*AND2X1*	4	1.368	5.472	
	$BW-OR$	*OR2X1*	4	1.368	5.472	
	BSR	*MX2X1*	8	2.394	19.152	
output	*Mux out*	*MX2X2*	8	3.078	58.824	82.08
		MX4X2	4	8.55		
	Output reg	*DFFQX1*	4	5.814	23.256	
Total	-	-	157	-	-	378

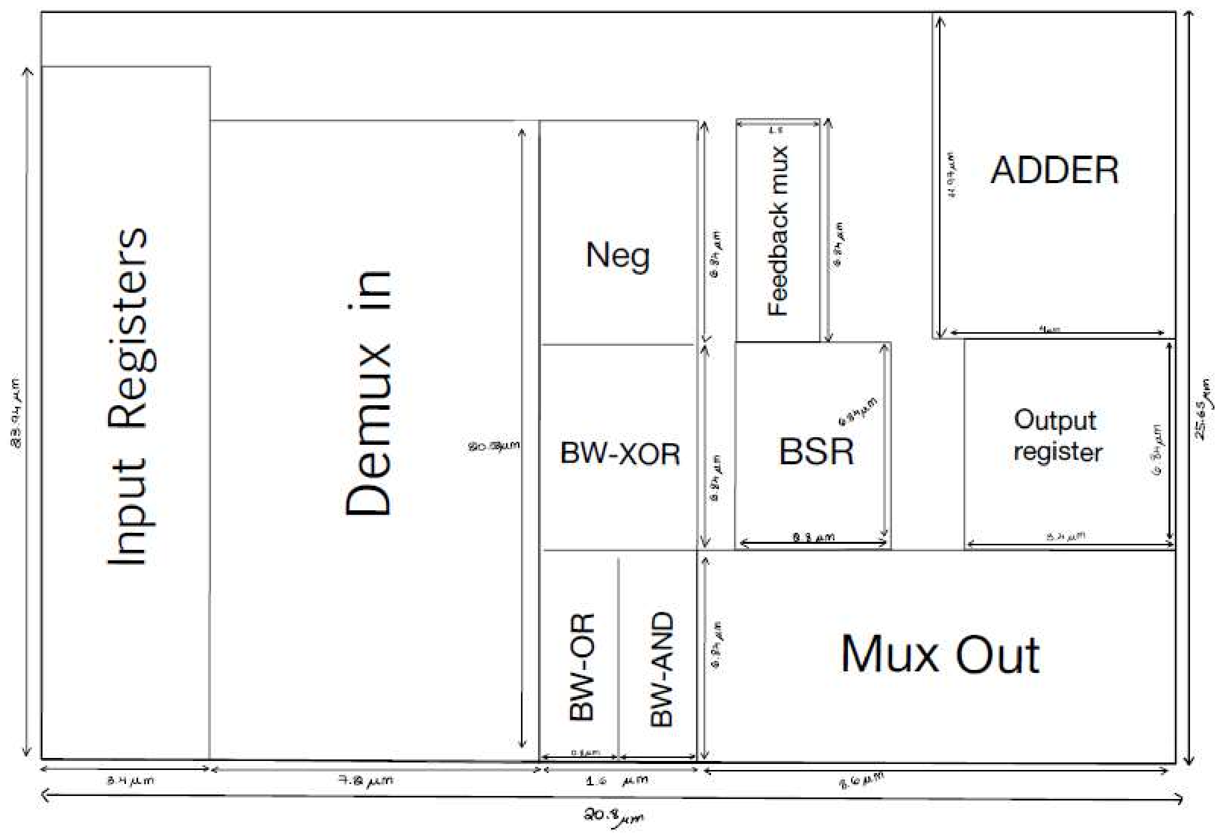
T.2. Estimated block sizes

To enhance the optimization of our floor plan, we included detailed block width and height measurements. This addition enables easier area optimization and facilitates efficient utilization.

Section	**Block**	**Block Width**	**Block Height**
$Input$	$Input\ regs$	3.4	23.94
	$Demux\ 1X2\ (4\ bits)$	3.6	3.42
	$Demux\ 1X4\ (4\ bits)$	3.6	10.26
	$Demux\ in$	7.2	20.52
	$Feedback\ Mux$	1.8	6.84
$logic$	$ADDER$	4	11.97
	NEG	1.6	6.84
	$BW-XOR$	1.6	6.84
	$BW-AND$	0.8	6.84
	$BW-OR$	0.8	6.84
	BSR	2.8	6.84
$output$	$Mux\ out$	8.6	6.84
	$Output\ reg$	3.4	6.84

$T.3. Block\ width\ and\ height$

1.5 Floor Plan



After preparing the layout for each block, we designed the floorplan for the full ALU.

We took into consideration the size of the blocks, their connections to other blocks, and the minimization of the total area of the ALU.

Our inputs are on the left side, and the data is flowing in a general direction from left to right. VCC, VDD are going horizontally, and connected between blocks to minimize space and wiring.

This floor plan has a total area of: $\mathbf{533.52\mu m^2}$

Which is **41**% extra area over the total area of the logic gates.