ENEE640: Designing 8x4-bit Sorting Circuit (Exam 1)

Nirvan Mishra
nmishra1@umd.edu

***Abstract*—This document describes the design of a combinational sorting circuit for eight unsigned 4-bit numbers (X0..X7). The circuit takes these eight inputs and outputs the same eight numbers, but in sorted order, sorting is obtained by using the Batcher Odd - Even Mergesort Algorithm from least to greatest (Y0..Y7). The report outlines the design approach, analyzes its power consumption and worst-case propagation delay, and provides a transistor-level schematic for the core sorting element.**

I. Introduction

The basic principle of Batcher's Odd-Even Mergesort is a merge algorithm that joins two sorted sequence halves to produce a fully sorted sequence. Batcher's Odd-Even Mergesort is popularized as an efficient sorting method for graphics-processing hardware. It's used in parallel computing and sorting networks. Batcher's method constructs sorting networks of size $O(n(logn)^2)$ and depth $O((logn)^2)$, where n represents the number of items to be sorted.

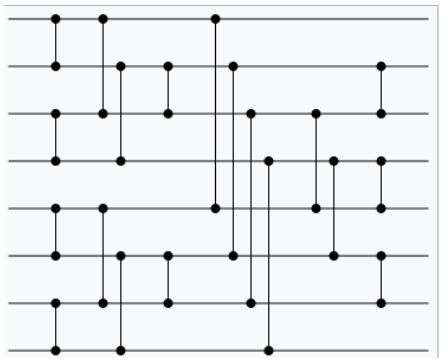
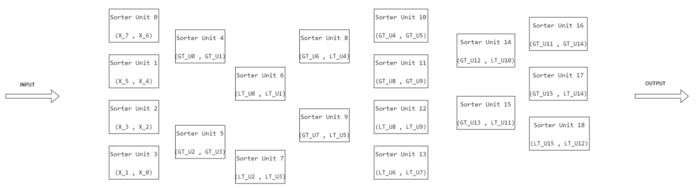


Fig. 1. Batcher Odd Even Mergesort


INPUT
Sorter Unit 0 (X_7 , X_6)
Sorter Unit 1 (X_5 , X_4)
Sorter Unit 2 (X_3 , X_2)
Sorter Unit 3 (X_1 , X_0)
Sorter Unit 4 (GT_U0 , GT_U1)
Sorter Unit 5 (GT_U2 , GT_U3)
Sorter Unit 6 (LT_U0 , LT_U1)
Sorter Unit 7 (LT_U2 , LT_U3)
Sorter Unit 8 (GT_U6 , LT_U4)
Sorter Unit 9 (GT_U7 , LT_U5)
Sorter Unit 10 (GT_U4 , GT_U5)
Sorter Unit 11 (GT_U8 , GT_U9)
Sorter Unit 12 (LT_U8 , LT_U9)
Sorter Unit 13 (LT_U6 , LT_U7)
Sorter Unit 14 (GT_U12 , LT_U10)
Sorter Unit 15 (GT_U13 , LT_U11)
Sorter Unit 16 (GT_U11 , GT_U14)
Sorter Unit 17 (GT_U15 , LT_U14)
Sorter Unit 18 (LT_U15 , LT_U12)
OUTPUT
Fig. 2. Sorting Network

The design in Fig. 1 utilizes a series of sorting units at each stage. Each sorting unit is composed of two key components: a comparator and a swapping unit.
The **comparator unit** acts as the brain of the sorting circuit, responsible for determining the relative order of two input numbers. It takes two 4-bit unsigned numbers (A and B) as inputs and produces three outputs:

- **A greater than B** A_GT_B: logic 1 if A is greater than B, else logic 0.
- **A less than B** A_LT_B: logic 1 if A is less than B, else logic 0.
- **A equals to B** A_EQ_B: logic 1 if A equals B, else logic 0

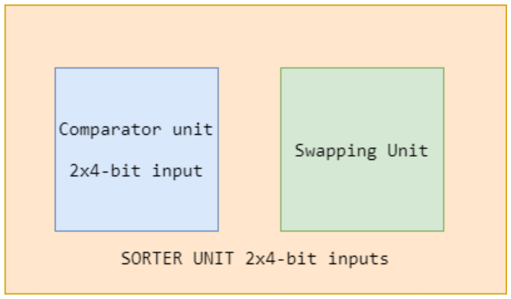


Fig. 3. Sorting Unit

The **swapping unit** is a crucial component that takes the outputs of the comparator unit and, based on those signals, rearranges the two input numbers if necessary to ensure proper sorting order. It acts like a controlled switch, manipulating the data flow depending on the comparison results.

- **A Greater Than B**: When the comparator unit determines that A is greater than B (A_GT_B is high), the swapping unit takes no action. It keeps the original order of the two inputs. In essence, it acts like a straight-through wire, passing A and B to the next stage unchanged.
- **A Less Than B** : Conversely, when the comparator unit indicates that A is less than B (A_LT_B is high), the swapping unit actively swaps the two inputs. It essentially reverses the order of A and B before sending them to the next stage.
- **A Equal To B**: If the comparator unit detects that A and B are equal (A_EQ_B is high), the swapping unit also maintains the original order.

The circuit utilizes a multi-stage architecture employing comparators for bit-by-bit comparisons and a swapping unit controlled by the comparator outputs to achieve the desired order. The analysis of power consumption and worst-case propagation delay will be addressed in further sections.

II. Design

An 8-input (4bit) sorter circuit designed to arrange eight unsigned 4-bit numbers in descending order. The circuit operates using a bottom-up approach, where smaller building blocks are designed first and then combined to achieve the overall functionality.

The core element of the sorting circuit is the sorter unit. Each sorter unit takes two 4-bit numbers (A and B) as inputs and outputs two sorted 4-bit numbers. Internally, a sorter unit comprises two crucial components:

Comparator Circuit: This circuit performs a bit-by-bit comparison between A and B. It determines which number is greater, less than, or equal to the other, and generates corresponding signals A_GT_B, A_LT_B, and A_EQ_B.

Swapping Circuit: Based on the comparator's outputs, the swapping circuit selectively rearranges the inputs. If A is greater than B, the swapping circuit keeps the original order. However, if A is less than B, the swapping circuit actively swaps A and B, ensuring the larger value moves forward in the sorting process.

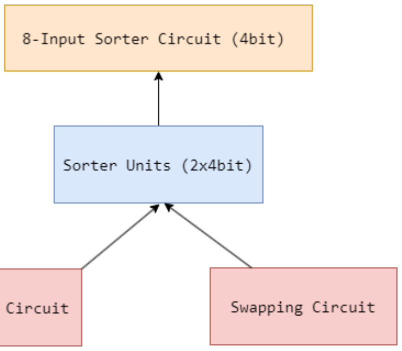


Fig. 4. Architecture of Sorter Circuit

A. Comparator Circuit

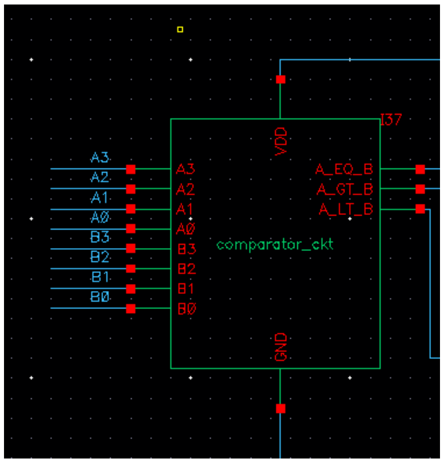


Fig. 5. Comparator Circuit

Working of comparator:
This comparator takes 2x4 bit inputs and produces three outputs: A greater than B; A less than B, and A equals B. The comparator compares MSB of A with respect to B, if A is high, and B is low, we get A greater than B, conversely, if A is low and B is high we get, A greater than B. But if both the MSB's are 1, it checks the next bit, and so on. If all the bits are equal in both the inputs, we have A equals B.

Example:

- For case A greater than B = 1: Let A be 1101 i.e., 13; and B be 1100 i.e., 12.

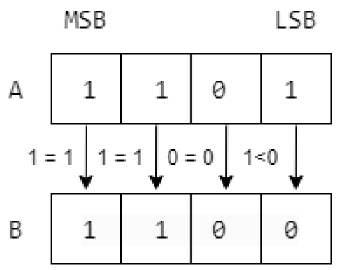


Fig. 6. A greater than B = 1

- For case A less than B = 1: Let A be 1100 i.e., 12; and B be 1101 i.e., 13.

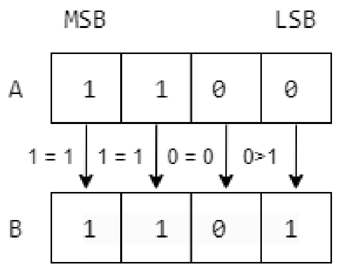


Fig. 7. A less than B = 1

- For case A equals B = 1: Let A be 1100 i.e., 12; and B be 1100 i.e., 12.

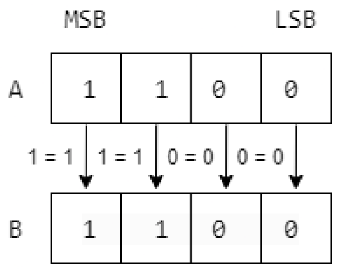


Fig. 8. A equals B = 1

- **Truth Table for Comparator circuit:**

A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$A = B$	$A > B$	$A < B$
$A_3 > B_3$	x	x	x	0	1	0
$A_3 < B_3$	x	x	x	0	0	1
$A_3 = B_3$	$A_2 > B_2$	x	x	0	1	0
$A_3 = B_3$	$A_2 < B_2$	x	x	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	x	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	x	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	1	0	0

- **Boolean expression:**

  1) For A = B:
     $X_3 = (A_3 \text{ xnor } B_3)$; $X_2 = (A_2 \text{ xnor } B_2)$; $X_1 = (A_1 \text{ xnor } B_1)$; $X_0 = (A_0 \text{ xnor } B_0)$

$$X_3 \cdot X_2 \cdot X_1 \cdot X_0 \tag{1}$$

  2) For $A > B$:

$$A_3\cdot\overline{B_3}+X_3\cdot A_2\cdot\overline{B_2}+X_3\cdot X_2\cdot A_1\cdot\overline{B_1}+X_3\cdot X_2\cdot X_1\cdot A_0\cdot\overline{B_0} \tag{2}$$

  3) For $A < B$:

$$\overline{A_3}\cdot B_3+X_3\cdot\overline{A_2}\cdot B_2+X_3\cdot X_2\cdot\overline{A_1}\cdot B_1+X_3\cdot X_2\cdot X_1\overline{A_0}\cdot B_0 \tag{3}$$

- **Optimization:** Instead of implementing the $A < B$ boolean expression, we can use NOR gate to determine the output for $A < B$, as when $A = B$ and $A > B$ = 0, $A < B$ = 1.

$A = B$	$A > B$	$A < B$
0	0	1
0	1	x
1	0	x
1	1	x

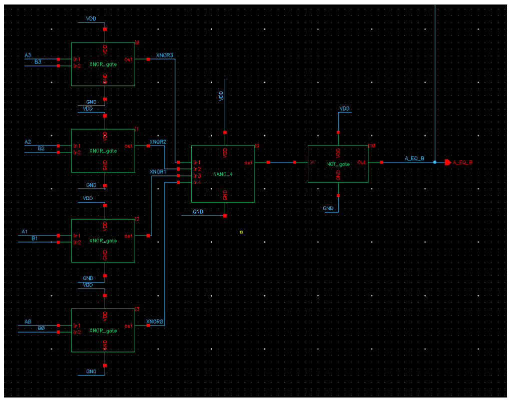


Fig. 9. A equals B

From Fig.12, when A=0 and B=1 at *0ns* A_LT_B is high, until A reaches to 0, and B to 1. As soon as A reaches to 1 and B to 0 A_GT_B is high. Next, when B transitions from 0 to 1 at *5ns* both A and B are equal now, and we see A_EQ_B is high, until A drops to 0 and again A_LT_B becomes high.A_LT_B is high.

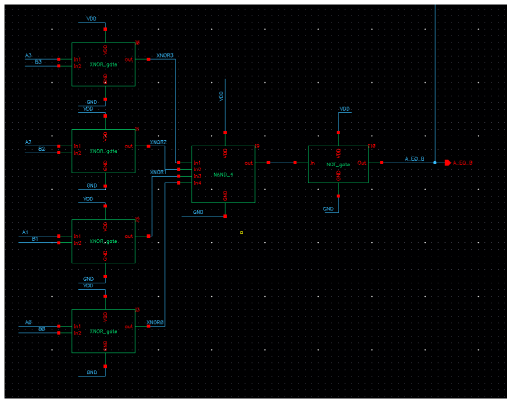


Fig. 10. A greater than B

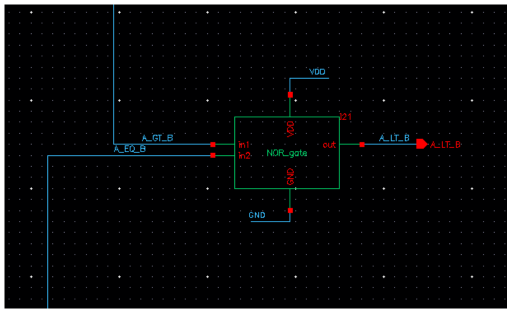


Fig. 11. A less than B

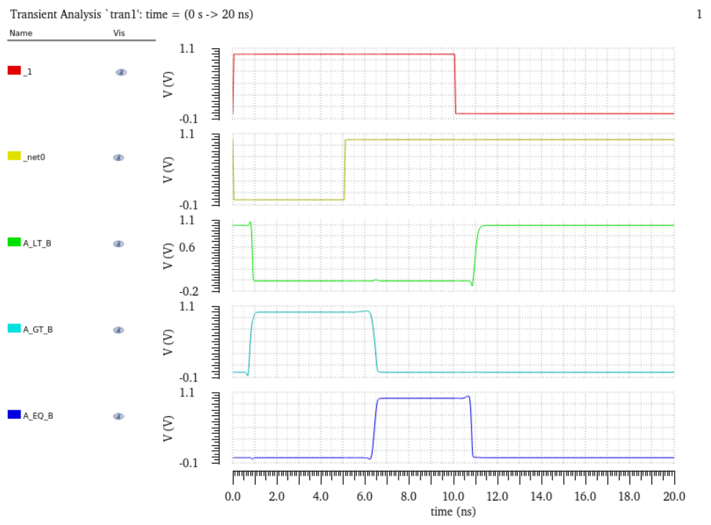


Fig. 12. Comparator's Output

B. Swapping Circuit

The swapping circuit takes 2x4bit inputs along with the output of comparator circuit, as selection lines. The output of

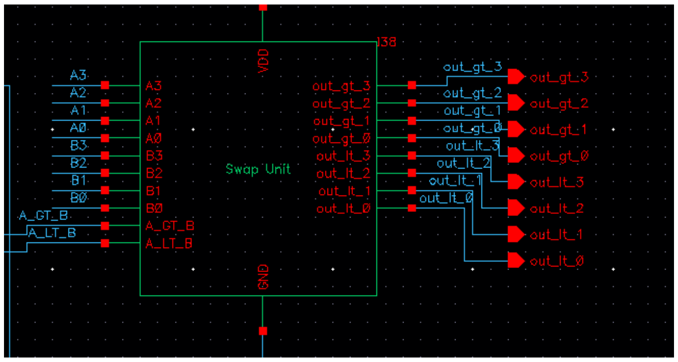


Fig. 13. Swapping Circuit

swapping circuit is the greater value and then lesser value, as shown in Fig.14.

- **Truth Table for Swapping Circuit:**

A_i	B_i	$A > B$ (i.e., S1)	$A < B$ (i.e., S0)	Output
0	0	0	0	0
0	1	0	1	swap
1	0	1	0	no change
1	1	1	1	no change

- **Boolean Expression:**
  1) For greater value *out_gt*:

$$\overline{A_i} \cdot B_i \cdot \overline{S1} \cdot S0 + A_i \cdot \overline{B_i} \cdot S1 \cdot \overline{S0} + A_i \cdot B_i \quad (4)$$

  2) For lesser value *out_lt*:

$$\overline{A_i} \cdot B_i \cdot S1 \cdot \overline{S0} + A_i \cdot \overline{B_i} \cdot \overline{S1} \cdot S0 + A_i \cdot B_i \quad (5)$$

- **Example**
  - **When $A < B$:**

    Let A be 4, and B be 8, here A_LT_B i.e., S0 = 1 and A_GT_B i.e., S1 = 0, and the supposed output of the unit must be *out_gt* = 8 and *out_lt* = 4:

    1) $out\_gt\_3 = \overline{A_3} \cdot B_3 \cdot \overline{S1} \cdot S0 + A_3 \cdot \overline{B_3} \cdot S1 \cdot \overline{S0} + A_3 \cdot B_3 = \mathbf{1}$
    2) $out\_gt\_2 = \overline{A_2} \cdot B_2 \cdot \overline{S1} \cdot S0 + A_2 \cdot \overline{B_2} \cdot S1 \cdot \overline{S0} + A_2 \cdot B_2 = \mathbf{0}$
    3) $out\_gt\_1 = \overline{A_1} \cdot B_1 \cdot \overline{S1} \cdot S0 + A_1 \cdot \overline{B_1} \cdot S1 \cdot \overline{S0} + A_1 \cdot B_1 = \mathbf{0}$
    4) $out\_gt\_0 = \overline{A_0} \cdot B_0 \cdot \overline{S1} \cdot S0 + A_0 \cdot \overline{B_0} \cdot S1 \cdot \overline{S0} + A_0 \cdot B_0 = \mathbf{0}$
    5) $out\_lt\_3 = \overline{A_3} \cdot B_3 \cdot S1 \cdot \overline{S0} + A_3 \cdot \overline{B_3} \cdot \overline{S1} \cdot S0 + A_3 \cdot B_3 = \mathbf{0}$
    6) $out\_lt\_2 = \overline{A_2} \cdot B_2 \cdot S1 \cdot \overline{S0} + A_2 \cdot \overline{B_2} \cdot \overline{S1} \cdot S0 + A_2 \cdot B_2 = \mathbf{1}$
    7) $out\_lt\_1 = \overline{A_1} \cdot B_1 \cdot S1 \cdot \overline{S0} + A_1 \cdot \overline{B_1} \cdot \overline{S1} \cdot S0 + A_1 \cdot B_1 = \mathbf{0}$
    8) $out\_lt\_0 = \overline{A_0} \cdot B_0 \cdot S1 \cdot \overline{S0} + A_0 \cdot \overline{B_0} \cdot \overline{S1} \cdot S0 + A_0 \cdot B_0 = \mathbf{0}$

    Hence, the result from the swapping unit is *out_gt* = 1000, i.e., 8, and *out_lt* = 0100, i.e., 4.

  - **When $A > B$:**

    Let A be 4, and B be 8, here A_LT_B i.e., S0 = 0 and A_GT_B i.e., S1 = 1 and the supposed output of the unit must be *out_gt* = 8 and *out_lt* = 4:

    1) $out\_gt\_3 = \overline{A_3} \cdot B_3 \cdot \overline{S1} \cdot S0 + A_3 \cdot \overline{B_3} \cdot S1 \cdot \overline{S0} + A_3 \cdot B_3 = \mathbf{1}$
    2) $out\_gt\_2 = \overline{A_2} \cdot B_2 \cdot \overline{S1} \cdot S0 + A_2 \cdot \overline{B_2} \cdot S1 \cdot \overline{S0} + A_2 \cdot B_2 = \mathbf{0}$
    3) $out\_gt\_1 = \overline{A_1} \cdot B_1 \cdot \overline{S1} \cdot S0 + A_1 \cdot \overline{B_1} \cdot S1 \cdot \overline{S0} + A_1 \cdot B_1 = \mathbf{0}$
    4) $out\_gt\_0 = \overline{A_0} \cdot B_0 \cdot \overline{S1} \cdot S0 + A_0 \cdot \overline{B_0} \cdot S1 \cdot \overline{S0} + A_0 \cdot B_0 = \mathbf{0}$
    5) $out\_lt\_3 = \overline{A_3} \cdot B_3 \cdot S1 \cdot \overline{S0} + A_3 \cdot \overline{B_3} \cdot \overline{S1} \cdot S0 + A_3 \cdot B_3 = \mathbf{0}$
    6) $out\_lt\_2 = \overline{A_2} \cdot B_2 \cdot S1 \cdot \overline{S0} + A_2 \cdot \overline{B_2} \cdot \overline{S1} \cdot S0 + A_2 \cdot B_2 = \mathbf{1}$
    7) $out\_lt\_1 = \overline{A_1} \cdot B_1 \cdot S1 \cdot \overline{S0} + A_1 \cdot \overline{B_1} \cdot \overline{S1} \cdot S0 + A_1 \cdot B_1 = \mathbf{0}$
    8) $out\_lt\_0 = \overline{A_0} \cdot B_0 \cdot S1 \cdot \overline{S0} + A_0 \cdot \overline{B_0} \cdot \overline{S1} \cdot S0 + A_0 \cdot B_0 = \mathbf{0}$

    Hence, the result from the swapping unit is *out_gt* = 1000, i.e., 8, and *out_lt* = 0100, i.e., 4.

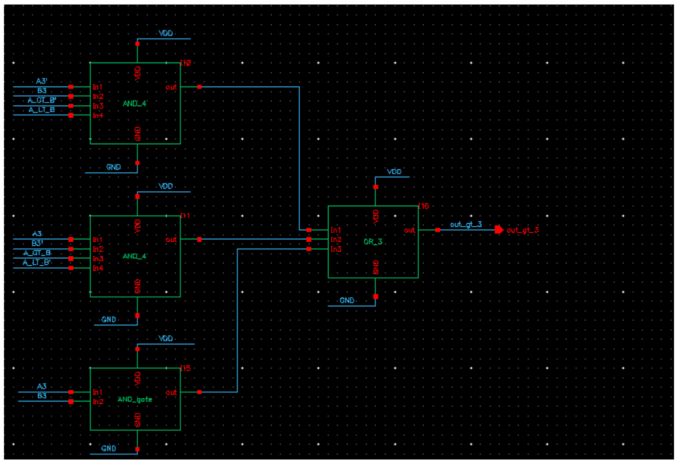


Fig. 14. *out_gt_3*

Fig. 14, shows the swapping schematic for the equation 4, that yields the *out_gt_3*, similarly for *out_gt_2*, *out_gt_1*, and *out_gt_0*.

Fig. 15, shows the swapping schematic for the equation 5, that yields the *out_lt_3*, similarly for *out_lt_2*, *out_lt_1*, and *out_lt_0*. Fig.16 shows the input pins, for the swapping unit

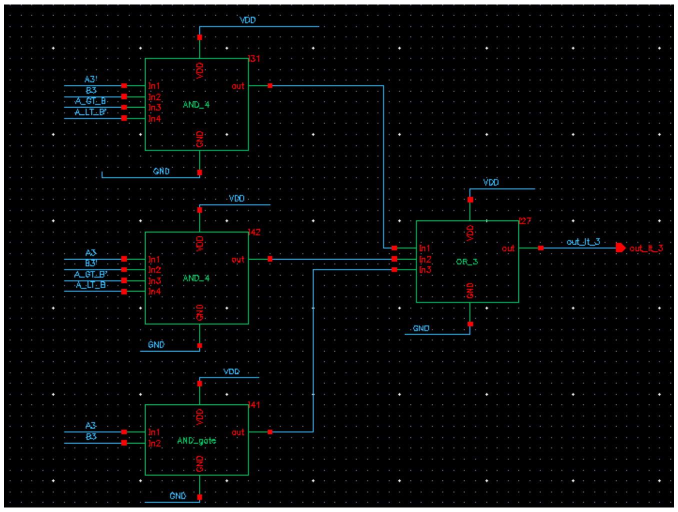


Fig. 15. *out_lt_3*

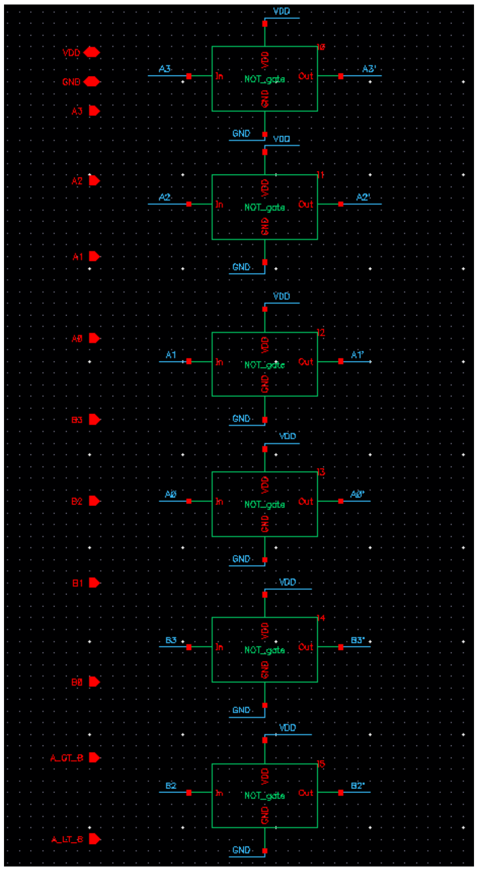


Fig. 16. Input

C. Sorter Unit

The sorter unit, consists of comparator and swapping circuit.
In the sorter unit, first the comparator circuit takes the two inputs, compares them, produces A_GT_B; A_LT_B; and A_EQ_B. Along with the inputs and these outputs of

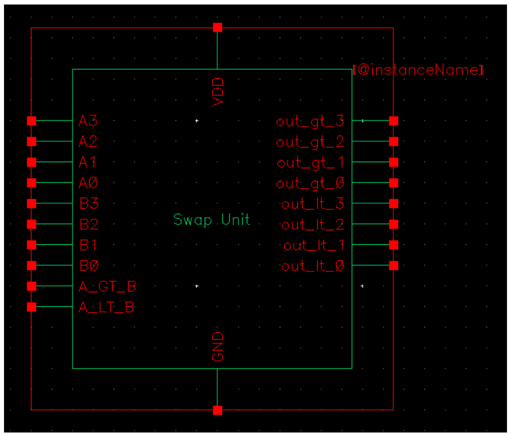


Fig. 17. Sorter Unit

comparator circuit as selection lines, to the swapping circuit, swaps the two inputs depending on which selection line is high.

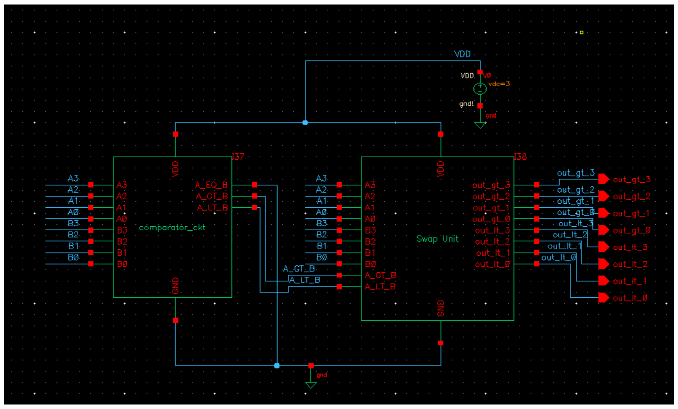


Fig. 18. Sorter Schematic

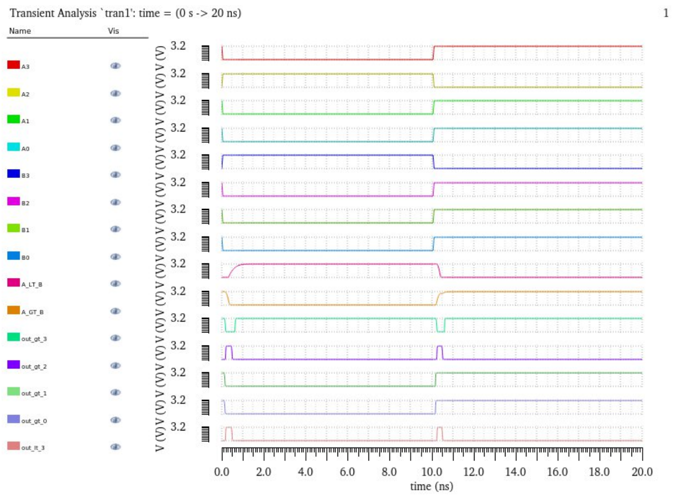


Fig. 19. Sorting Unit output

Fig. 19 shows the sorting process for inputs A (binary 4) and B (binary 8). The comparator starts by comparing the most significant bits (MSBs). Since A's MSB (0) is less than B's MSB (1), the comparator sets A_LT_B high. This triggers the swapping unit to swap A and B. After the swap, the output lines (out_gt and out_lt) reflect the swapped order: out_gt becomes 8 (original B) and out_lt becomes 4 (original A).

D. 8 - Input Sorter Circuit (4-bit)

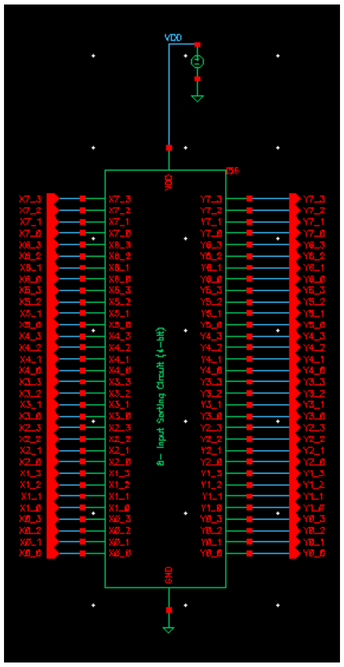


Fig. 20. Sorting Circuit

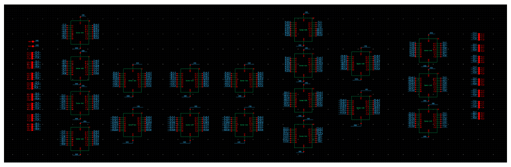
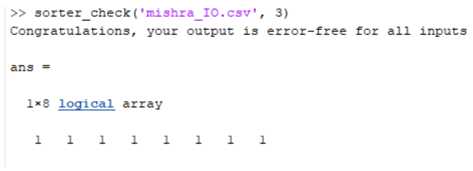


Fig. 21. Sorting Schematic



```
>> sorter_check('mishra_IO.csv', 3)
Congratulations, your output is error-free for all inputs

ans =

  1×8 logical array

   1   1   1   1   1   1   1   1
```

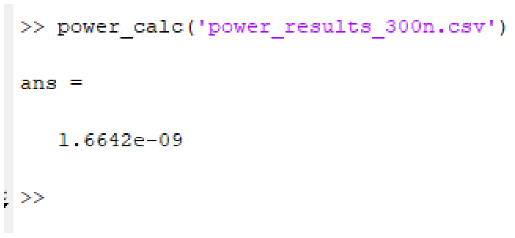
Fig. 22. Result of sorter

III. Result

A. Power Analysis

Width (n)	Energy (in Joules)	Power (in mW)
300	1.6642e-09	6.93
250	1.4529e-09	6.05
215	1.3433e-09	5.59
200	1.2253e-09	5.08

1) Power at width = 300n: 1.6642/240 = 6.93 mW



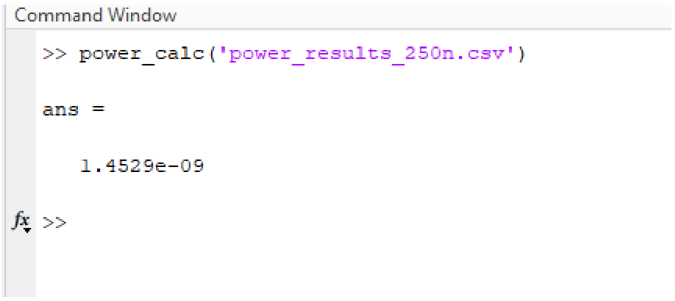
```
>> power_calc('power_results_300n.csv')

ans =

   1.6642e-09

>>
```

2) Power at width = 250n: 1.4529/240 = 6.05 mW



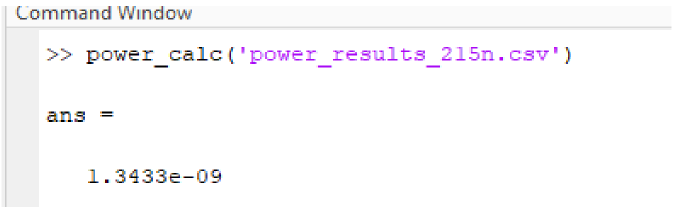
```
Command Window
>> power_calc('power_results_250n.csv')

ans =

   1.4529e-09

fx >>
```

3) Power at width = 215n: 1.3433e/240 = 5.59 mW

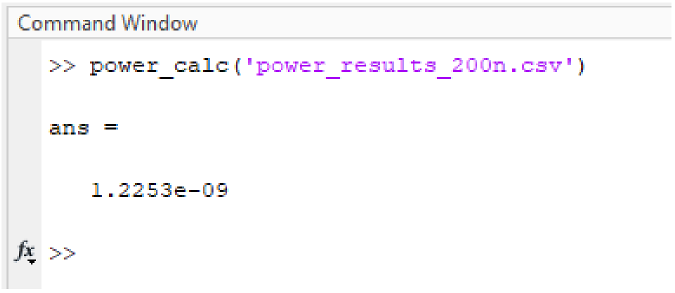


```
Command Window
>> power_calc('power_results_215n.csv')

ans =

   1.3433e-09
```

4) Power at width = 200n: 1.2253/240 = 5.08 mW



```
Command Window
>> power_calc('power_results_200n.csv')

ans =

   1.2253e-09

fx >>
```

B. Delay Analysis

Width (n)	Output	Worst Case Delay (in ns)
300	$Y3_1$	1.9447
250	$Y3_1$	1.8629
225	$Y3_1$	1.81
200	$Y3_1$	1.82

1) Worst case delay at 300n:

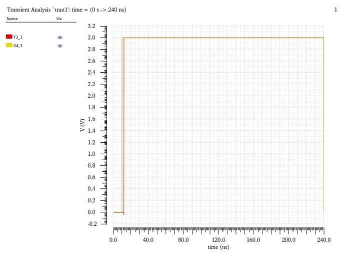


Fig. 23. Worst Case Delay at 300n

2) Worst case delay at 250n:

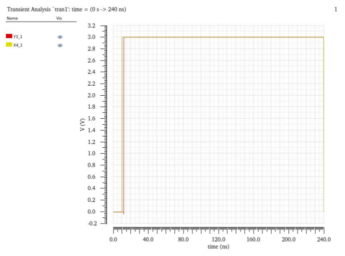


Fig. 24. Worst Case Delay at 250n

3) Worst case delay at 200n:

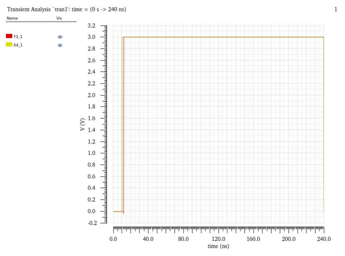
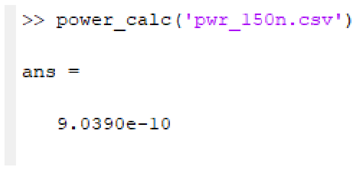


Fig. 25. Worst Case Delay at 200n

C. Trade-off Analysis: Width Reduction

- Expected Benefits:

  Reduced Power Consumption: Narrower transistors generally lead to lower leakage currents and dynamic power dissipation during switching. As when the width reduced to 150n, the power delay was 3.76 mW.



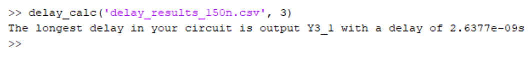
```
>> power_calc('pwr_150n.csv')

ans =

   9.0390e-10
```

- Expected Drawbacks:

  Increased Delay: Narrower transistors offer higher resistance, hindering current flow and increasing the time it takes for the voltage to reach its desired level at the output. This translates to a higher propagation delay, potentially impacting sorter performance.



```
>> delay_calc('delay_results_150n.csv', 3)
The longest delay in your circuit is output Y3_1 with a delay of 2.6377e-09s
>>
```

- Analysis:

  Impact on Power Consumption: Reducing the width from 300nm to 150nm can lead to a significant decrease in leakage current due to the smaller channel area. Dynamic power consumption, which depends on switching capacitance and switching frequency, might also be reduced due to lower capacitance with narrower transistors.

  Impact on Delay: The increased resistance of narrower transistors will likely lead to a noticeable increase in delay. This could potentially affect the overall sorting speed of the circuit.

IV. References

- https://youtu.be/PGA52YSVKdI?si=OpbhslfvbICcOBQX
- Batcher's Odd Even Mergesort