VLSI Milestone I:
High Performance ALU Design

Abdallah Mahmoud
Malak Ahmed
Noor Motaz Elsaban

Outline

1. Overview on ALUs
2. Supply Voltage
3. Operating Frequency
4. Adder
5. Shifter
6. Comparator
7. References

ALU: overview

- The Arithmetic Logic Unit (ALU) is a fundamental digital circuit and the core "calculator" inside every Central Processing Unit (CPU).
- It is the component that actually executes all mathematical and logical instructions
- An ALU takes data from registers and a command from the control unit, performs an operation, and delivers a result back to the registers
- Inputs:
  - Operand A: The first data word (e.g., 4-bit)
  - Operand B: The second data word (e.g., 4-bit)
  - Opcode (Control Signal): A set of bits that tells the ALU which operation to perform (e.g., 001 for ADD, 010 for SUBTRACT)
- Outputs:
  - Result: The single output data word (e.g., 4-bit)
  - Status Flags: Single bits that report on the result (Carry)

ALU: Overview

- An ALU is not a single, giant component; it is built by combining several specialized functional blocks
- A large multiplexer (MUX) selects the output from the correct block based on the Opcode
- The key blocks required to build a high-performance ALU are:
  - Adder: used to perform addition, subtraction, incrementing, and decrementing
  - Shifter: a dedicated block for performing logical shifts, arithmetic shifts, and rotations
  - Logical Unit: a set of gates that performs all bitwise operations (e.g., A AND B, A OR B)
  - Comparator: a block that checks the relationship between two operands (e.g., A > B, A = B)
- ALUs can either be high performance, low power or hybrid
- ALUs be used in applications like gaming and pacemakers
- Our project focuses on designing a 4-bit ALU using 65nm CMOS technology
- The design philosophy of a high performance ALU is focused on minimizing the critical path, the slowest possible signal path through the circuit, which determines its maximum clock speed

Supply Voltage

	Typical Range	Benefit
High Performance	1.0-1.8V	Maximum Speed, balanced and industry standard
Near Threshold	0.6-0.9V	Energy efficient (low dynamic power)
Sub-threshold	0.4-0.5V	Least dynamic power consumption

Supply Voltage

- **Rejection of Sub-Vth (0.4 V – 0.5 V):** The resulting performance using Sub-Vth voltage is functionally unusable for a standard digital ALU.
- **Rejection of NTV (0.6 V – 0.9 V):** While achieving the highest energy efficiency, it is very slow (around 8 times slower than the case in which we use 1.2V) and this is generally too severe for a core ALU. More importantly, the dramatically reduced noise margins and high sensitivity to process variation make it unsuitable for our application.

Supply Voltage

- Rationale for selecting Vdd = 1.8 V

	Explanation	Reason (from the map)
High speed/ performance	Capable of multi-GHz operation	Higher overdrive gives us more driving current (Id) which lowers the RC delay (CL charges quicker) so overall we have a low propagation delay (fast ALU)
Robustness	Highest margin/ tolerance against PVT and noise	Large voltage swing ensures sufficient logic noise margins (NML and NMH)
Excellent leakage control	The ability to use transistors with thick oxide and high Vth reduces the leakage current	Thicker gate oxide minimizes gate tunneling current; higher Vth limits subthreshold current
Acceptable energy efficiency	Dynamic power dissipation is high but the Power-Delay Product (PDP) is optimized for high throughput efficiency	Dynamic power scales higher but the short delay minimizes active switching time, resulting in high overall efficiency

Operating Frequency

- The selection of the operating frequency for a digital circuit using a fixed supply voltage (Vdd=1.8V) is primarily a trade-off between maximal throughput (Performance) and overall energy consumption, as dynamic power scales linearly with frequency.

	Typical Range	Applications	Key Performance Metrics
Ultra-Low Speed	150 KHz to 10 MHz	Specialized reference circuits, sensor interfaces, or low-speed I/O pads	Maximizing standby battery life; minimizing power noise
Nominal/High-Throughput	1.0 GHz to 2.0 GHz	Standard processor cores, Digital Signal Processors or integrated logic blocks requiring fast data processing	Balanced power-performance; high computational yield
Maximum Performance	3.6 GHz to 5.0 GHz	Highly optimized microprocessor cores, clock dividers, or critical path comparators.	Achieving absolute minimum latency; usually requires very high power density

Operating Frequency

- The best operating frequency for a robust, standard 4 bit ALU using 1.8V supply voltage is typically in the range of 1.0 GHz to 1.2 GHz
- This frequency range is optimal because it achieves the best compromise between performance and energy efficiency

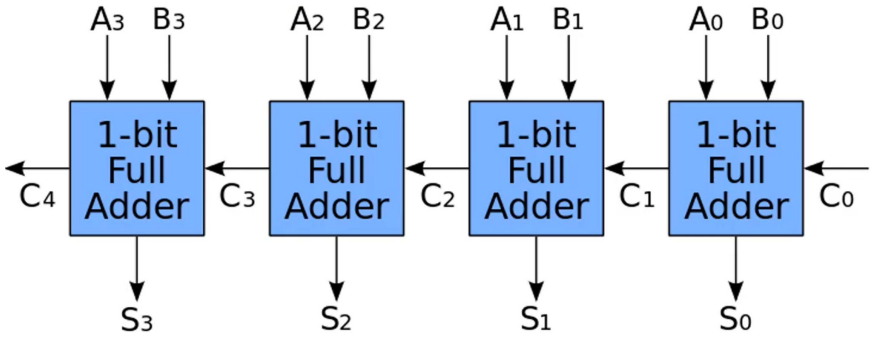
Reason	Explanation
Computational efficiency (throughput)	1 GHz means the 4-bit result completes in a worst case delay of around 1 ns which is a very low latency per operation
Standard Cell and IP Maturity	This range is widely adopted as the default performance target for general-purpose digital cores. This standardization ensures high yield, access to proven standard cell libraries, and predictability
Static leakage reduction	Operating at high speed is crucial because it minimizes the clock cycle time (1/f), reducing the duration over which the static current leaks for any given computation
Robustness	Operating the ALU at 1.2V and 1.0-1.2 GHz results in achieving high throughput reliably, without risking the functional failures or high variation sensitivity that occurs when using a frequency that is too close to the process's absolute maximum frequency ceiling

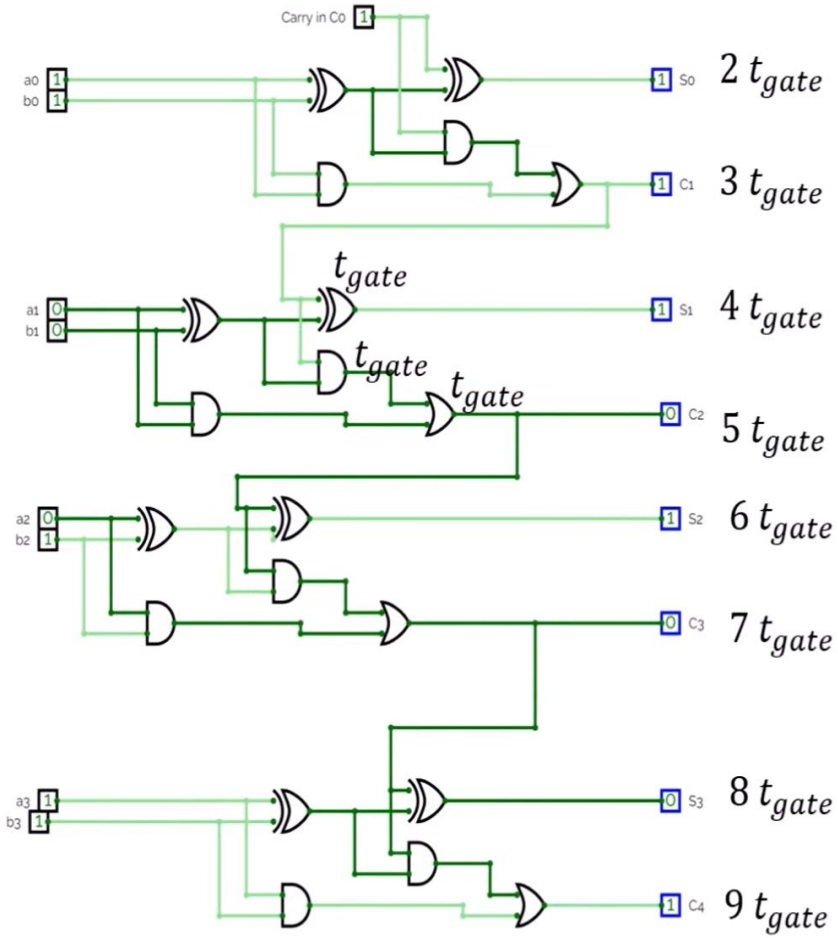
Different Adder Architectures

Different Searched Architectures

- Ripple Carry
- Carry Look-Ahead
- Carry Select
- Carry Skip

Ripple Carry Adder





- Eah carry has to wait for the previous carry to be generated, so carries are calculated serially, which causes accumulated delay.

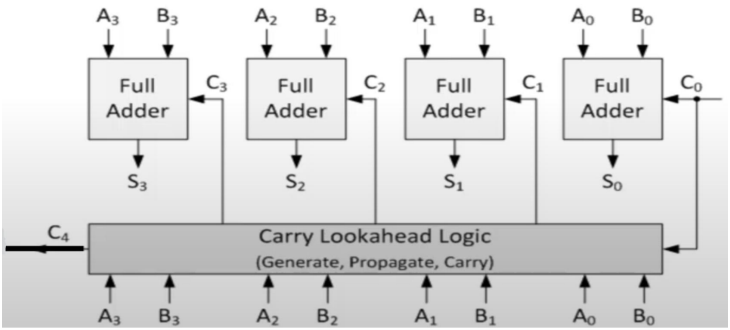
$$t_{sum} = 2n\ t_{gate}$$

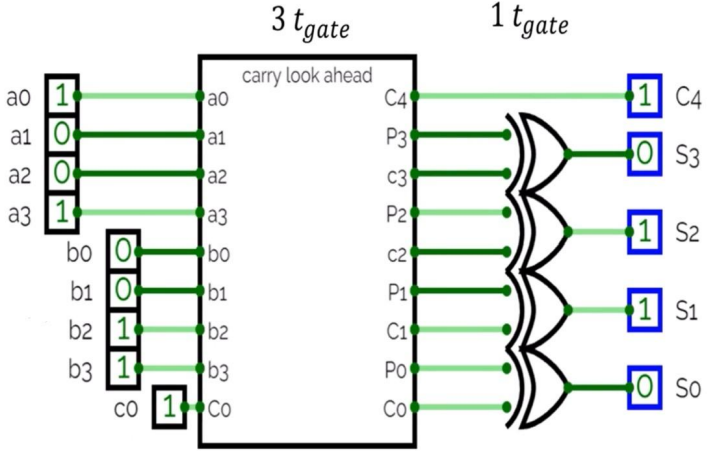
$$t_{carry} = (2n+1) t_{gate}$$

$$t_{c\ 4\ bitAdd} = 9\ t_{gate}$$

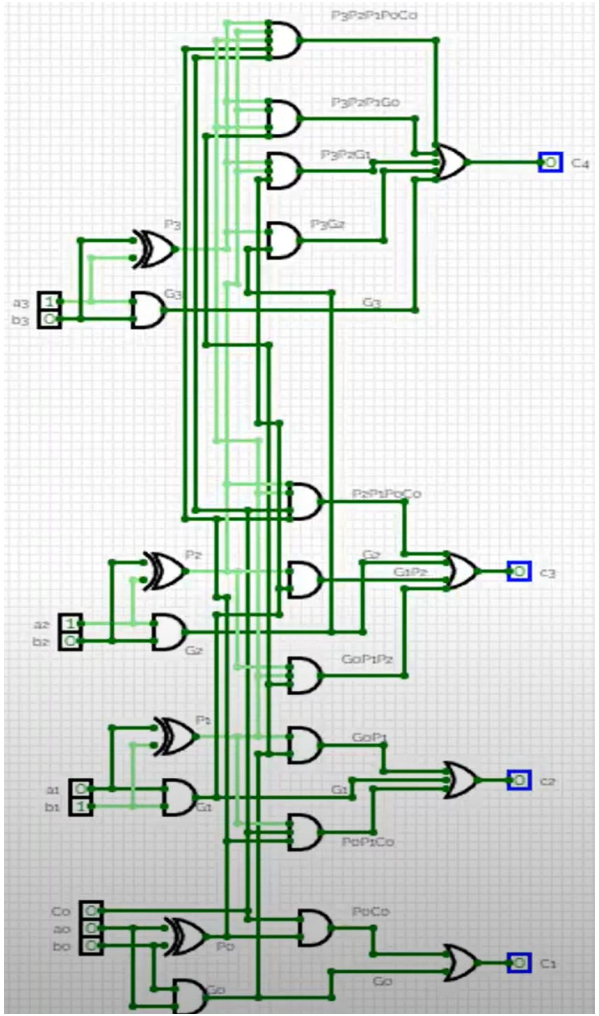
Carry Look-Ahead Adder

- Carry values are calculated using separate logic circuits.
- Carries are calculated in parallel circuits, which are also parallel to the sum bits calculation circuits.
- The total output delay time is approximately equivalent to 4 stages or 4 gate delays.



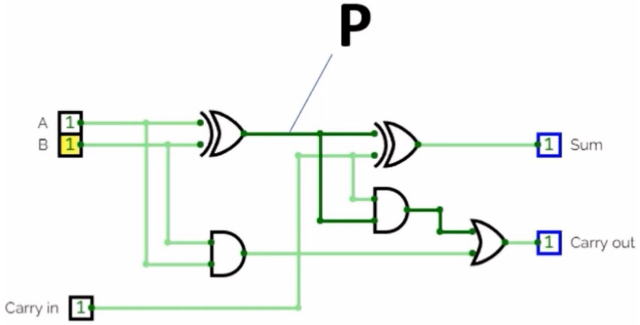


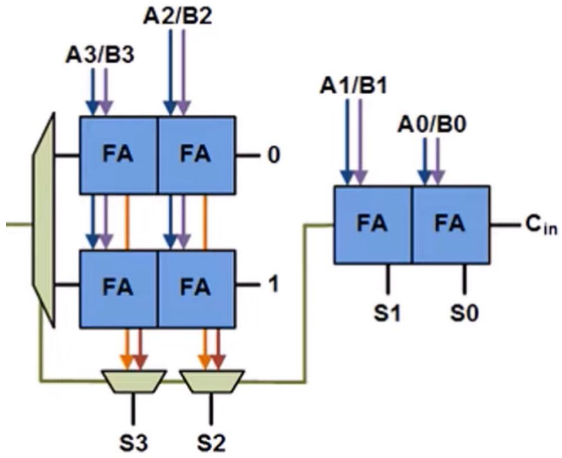
$$t_{CLA} = 4\, t_{gate}$$



Carry Select Adder

- This type of adders divides the ripple carry operation into two stages.
- The first stage is to calculate the middle output carry that could be either 0 or 1.
- The second stage does two parallel operations, one assuming that the carry is 0 and the other one assuming the carry is 1.
- This introduced type of parallelism helps to reduce the cascaded carry time delay.
- Multiplexers are then used to select the output of one of the two parallel stages based on the middle output carry.





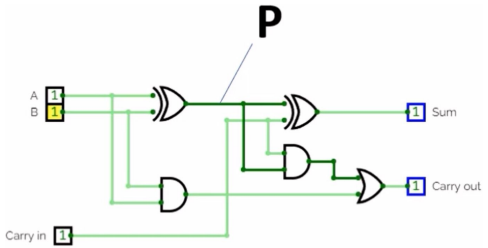
t_{mux} $5\,t_{gate}$

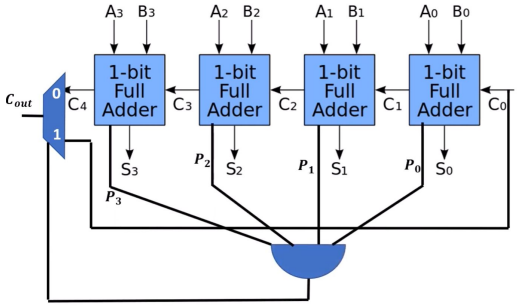
$t_{CSelectA} \sim 5t_{gate} + t_{mux}$

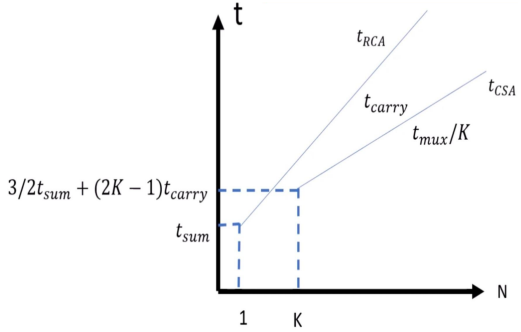
Carry Skip Adder

- Carry calculations are skipped based on the P signals values that are calculated using the XOR gates inside the full adder.

- Usually the reduced delay time of this type of adders appear as the number of bits get bigger, but for small number of bits, such as 4, the delay of this adder is almost close to that of the Ripple Carry Adder.







$$t_{CSA} = 3/2t_{sum} + Kt_{carry} + (\frac{N}{K} - 1)t_{mux} + (K\text{-}1)\, t_{carry}$$

$$t_{RCA} = t_{sum} + (n-1)t_{carry}$$

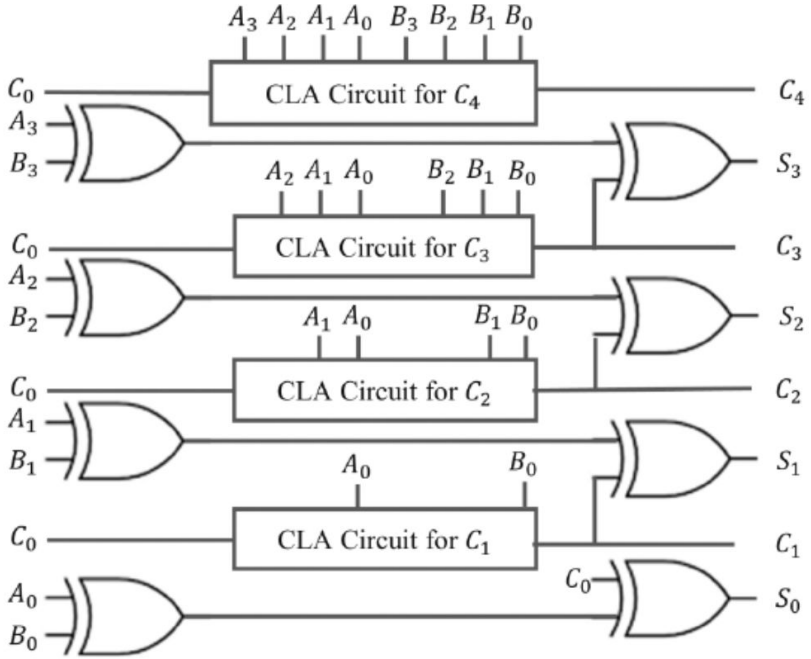
For n = 4 & k =4 $\rightarrow t_{CSA} = 3/2\, t_{sum} + 9t_{carry} \rightarrow t_{RCA} = t_{sum} + 3t_{carry}$

For n = 4 & k =2 $\rightarrow t_{CSA} = 3/2\, t_{sum} + t_{mux} + 3t_{carry} \rightarrow t_{RCA} = t_{sum} + t_{carry}$

Chosen Adder Architecture: Why this Architecture?

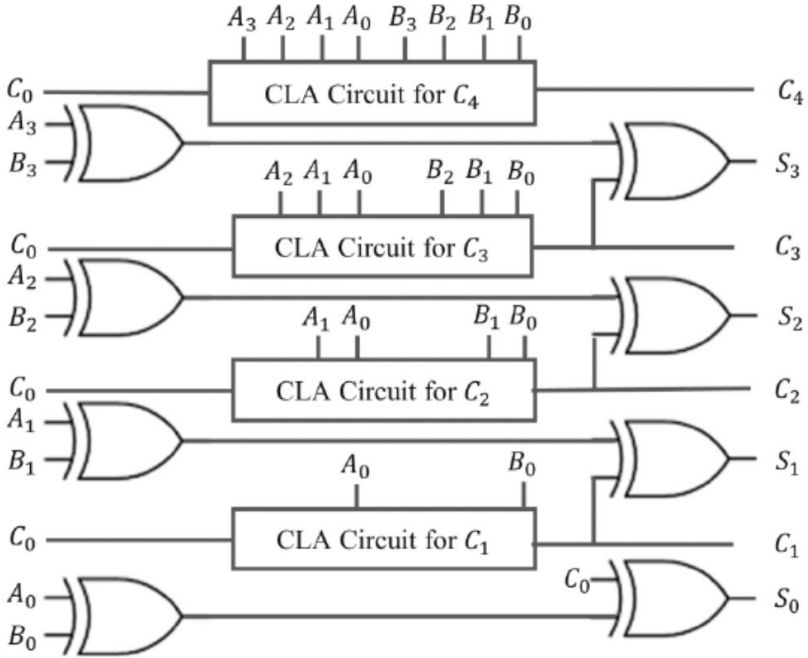
- From the time comparisons held between the different adders architectures, we found out that Carry Lookahead Adder is the fastest, with relatively doable topology for 4-bit inputs.

- The reported time delay simulations using similar technology, 45 nm, on this architecture are really promising.

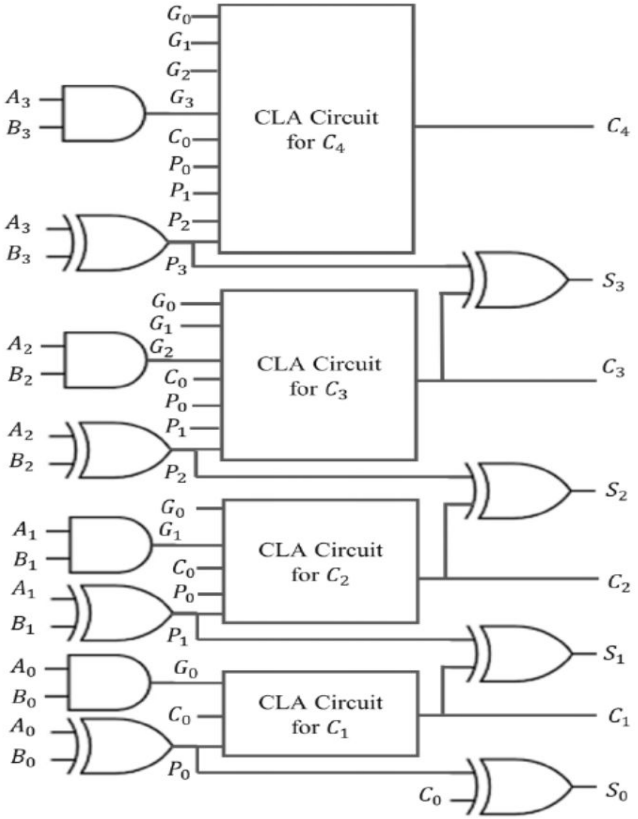


Proposed CLA Architecture

Chosen Adder Architecture



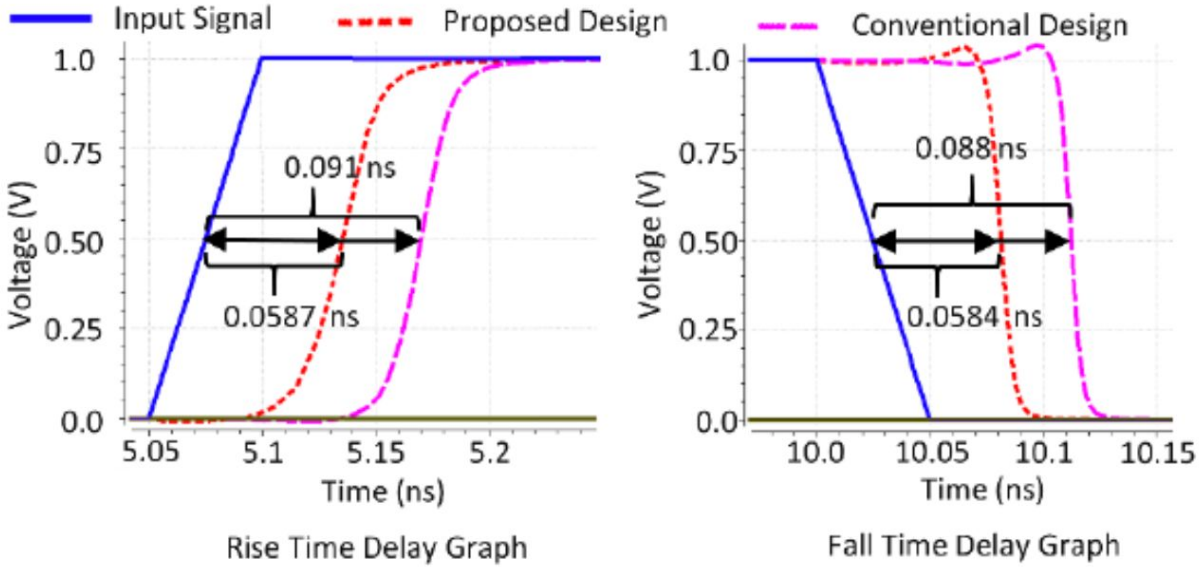
Proposed CLA Architecture



Conventional CLA Architecture

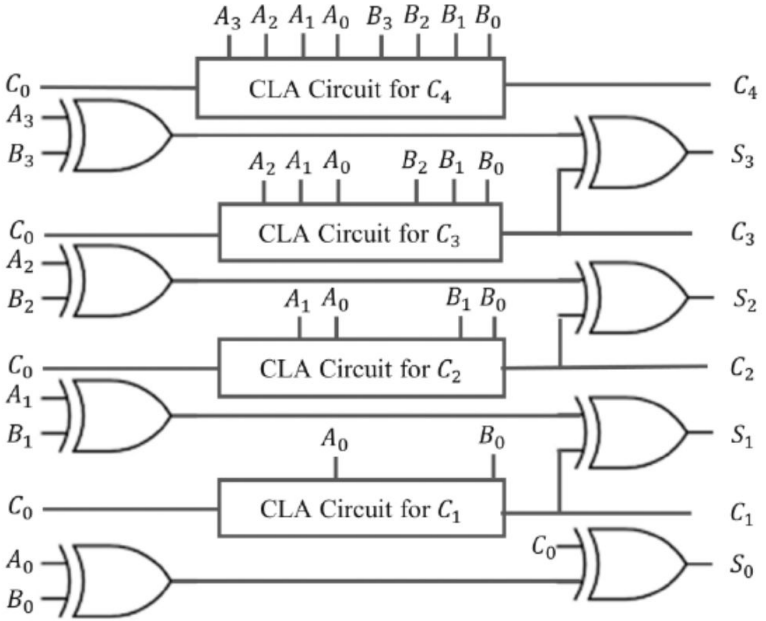
Proposed Vs Conventional Architectures: 45 nm Technology

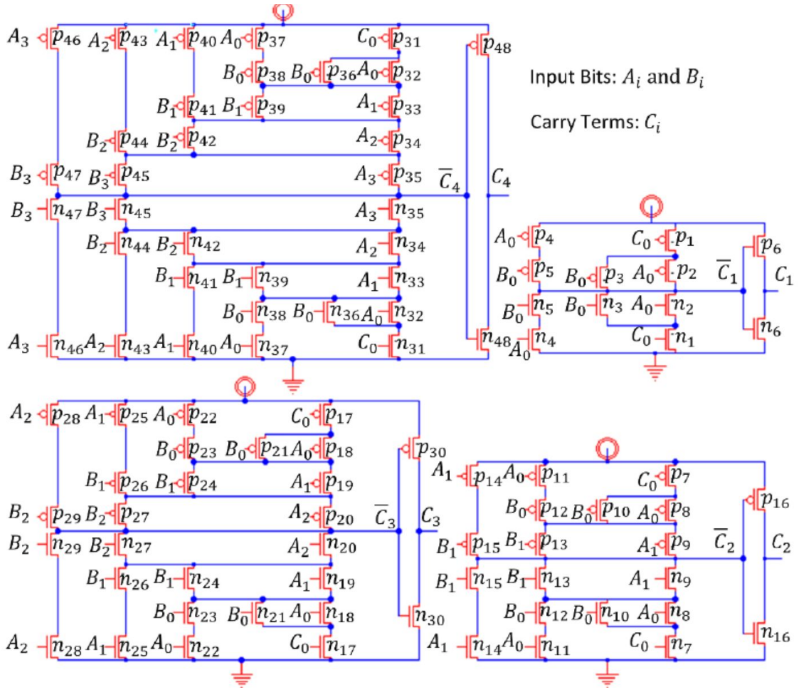
4-Bit CLA Adder Cell	Transistor Count	Propagation Delay (ns)	PDP (fJ)
Conventional	170	0.0895	0.703
Proposed	186	0.0586	0.438
Improvement	------	34.53%	37.696%



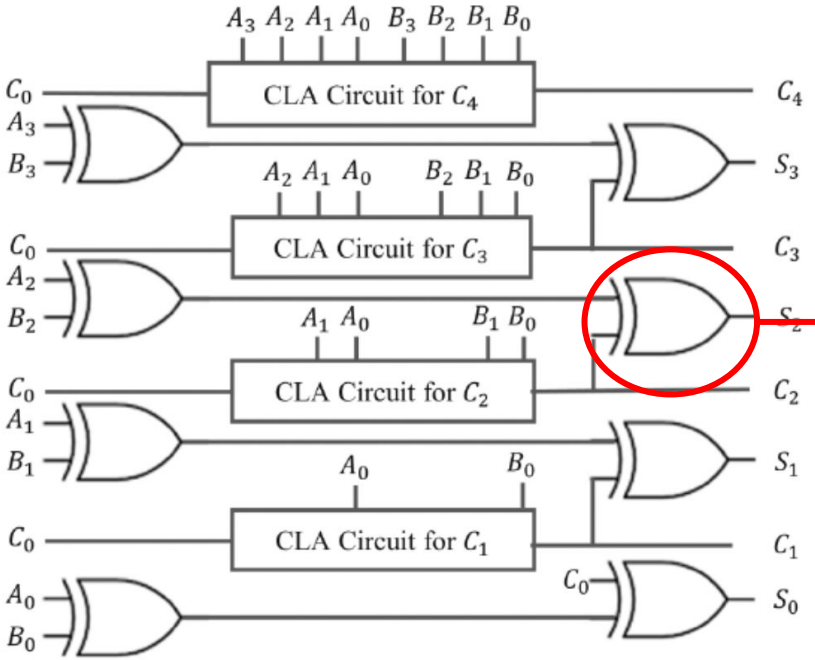
Hasan, Mehedi & Siddique, Abdul & Mondol, Abdal & Hossain, Mainul & Zaman, Hasan & Islam, Sharnali. (2020). High-Performance Design of a 4-Bit Carry Look-Ahead Adder in Static CMOS Logic. Indonesian Journal of Electrical Engineering and Informatics (IJEEI). 8. 666-673. 10.11591/ijeei.v8i4.2582.

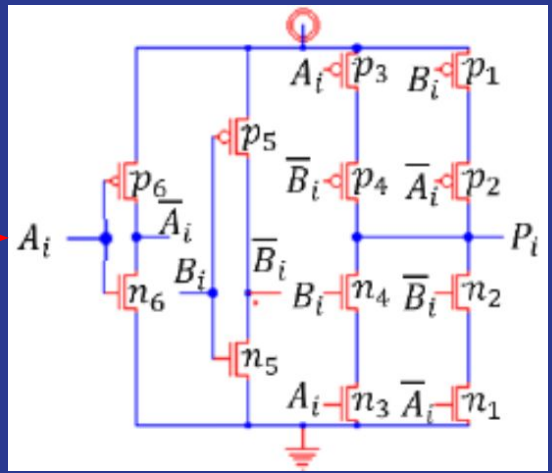
Chosen Adder Architecture





Hasan, Mehedi & Siddique, Abdul & Mondol, Abdal & Hossain, Mainul & Zaman, Hasan & Islam, Sharnali. (2020). High-Performance Design of a 4-Bit Carry Look-Ahead Adder in Static CMOS Logic. Indonesian Journal of Electrical Engineering and Informatics (IJEEI). 8. 666-673. 10.11591/ijeei.v8i4.2582.





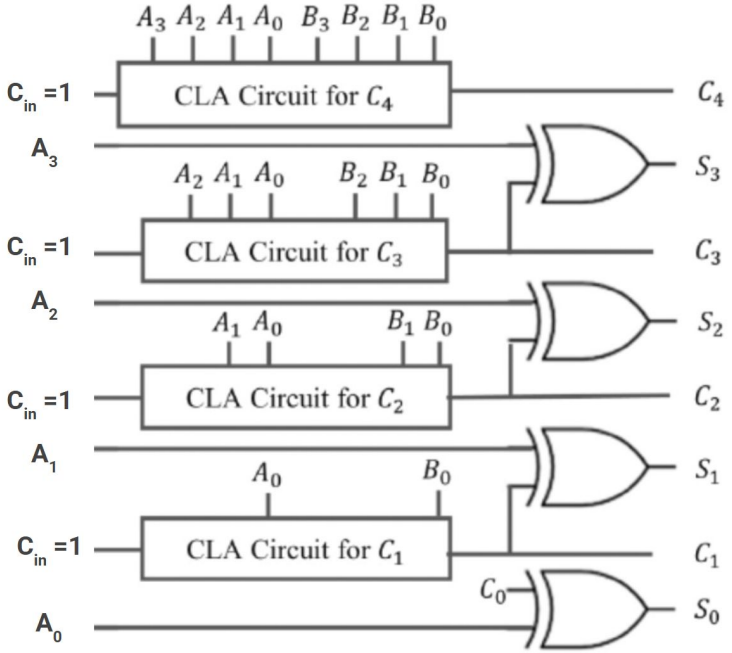
- XOR Gate that will be used to calculate the sum signals at the output

Increment Operation

- The same adder architecture will be used, but to decrease delay further, a simple change will be done.

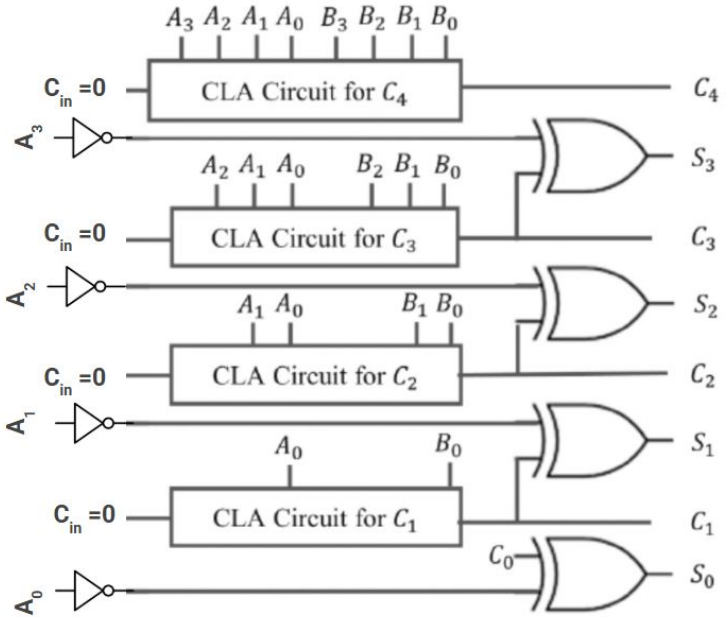
- Since increment of A is equivalent to A+1, which is equivalent to A+0 & C_{in} = 1.

- Since B = 0, we can eliminate the initial XOR operation (A XOR 0 = A), which means less hardware.



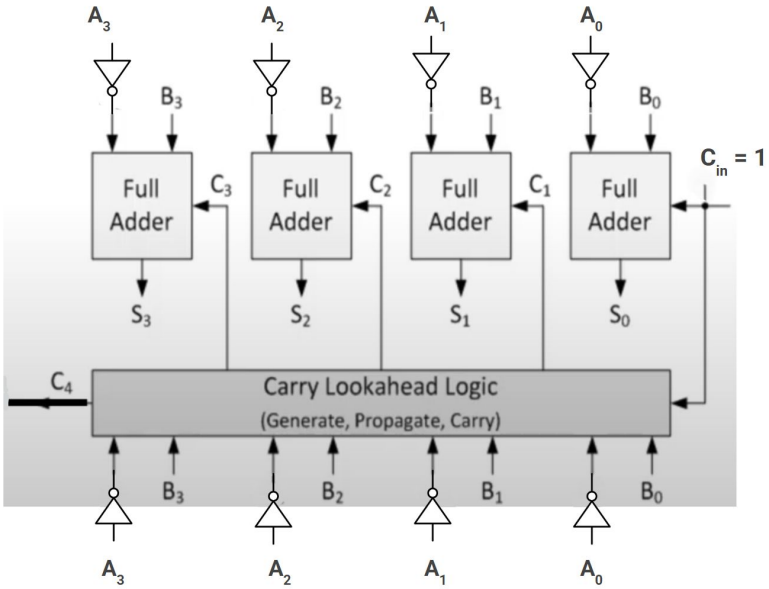
Decrement Operation

- The same adder architecture will be used, but to decrease delay further, a simple change will be done.
- Since decrement of A is equivalent to A-0001, which is equivalent to A+1110 & C_{in} = 1, which equals A+1111 & C_{in} = 0.
- Since B = 0, we can eliminate the initial XOR operation (A XOR 0 = A), which means less hardware.



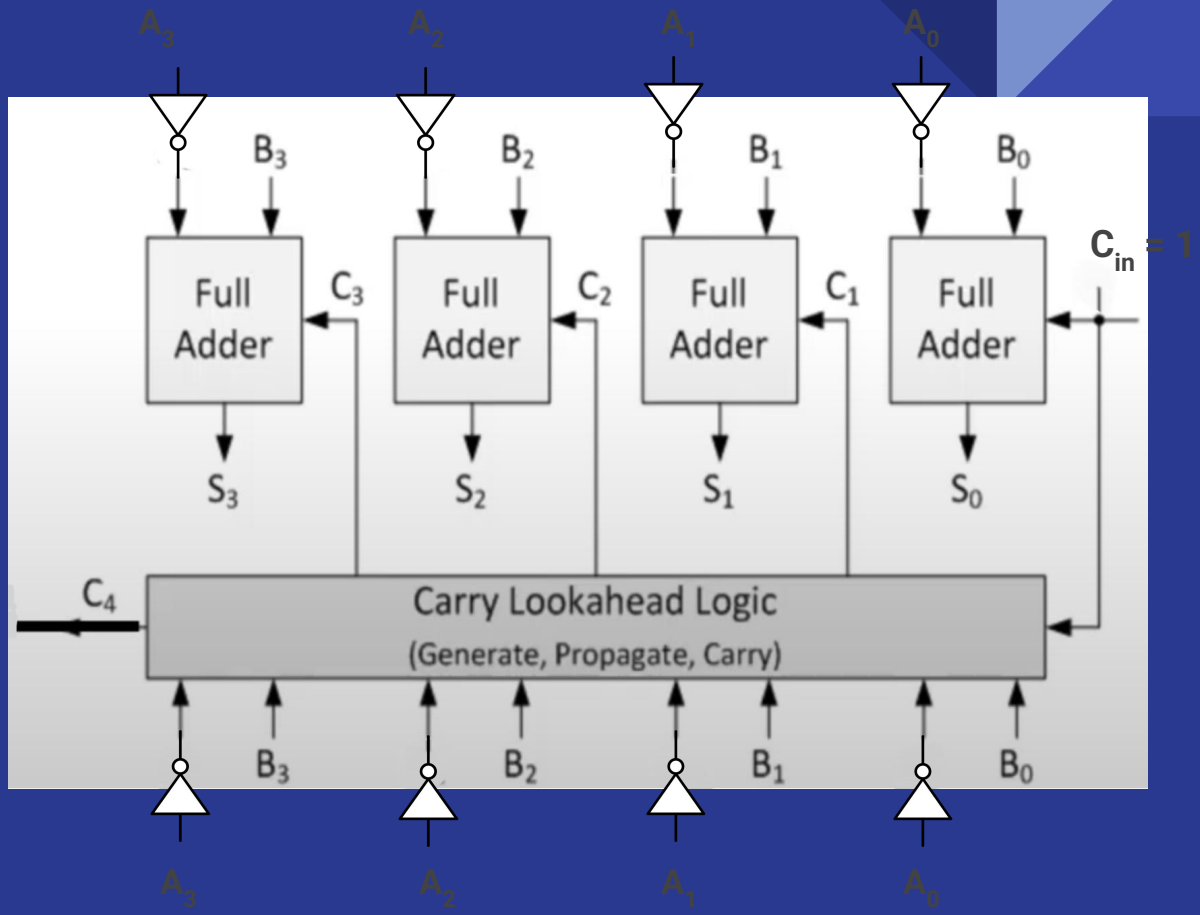
Subtraction Operation

- The subtrator will be of the same architecture of adder, but the addition process will be between the minuend and the 2's complement of the subtrahend.



- This actually benefits from the speed of the adder's proposed architecture, while keeping the overall design not redundant and area wasting.

- Note that:
  - B - A = B + A' + 1
-
  - The process of subtraction can be done using a control bit, one of the bits in the opcode that decides whether the inverse of A will be added to B or A will be added to B based on the operation selected.

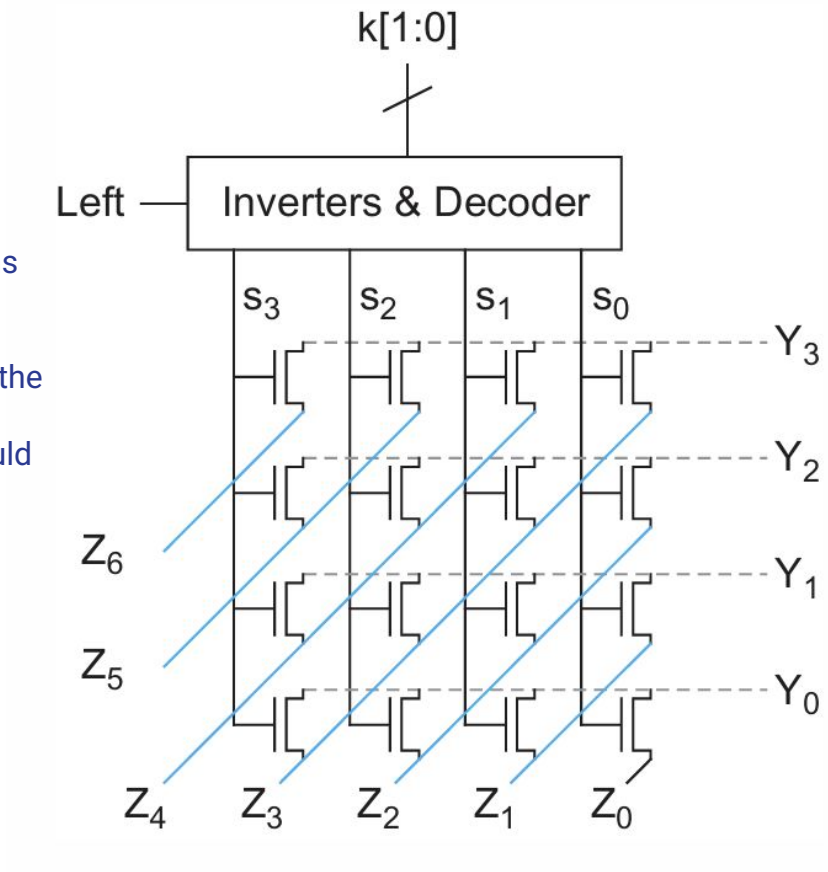

A3
A2
A1
A0
B3
B2
B1
B0
Full Adder
Full Adder
Full Adder
Full Adder
C3
C2
C1
Cin = 1
S3
S2
S1
S0
C4
Carry Lookahead Logic
(Generate, Propagate, Carry)
B3
B2
B1
B0
A3
A2
A1
A0

Different Shifter Architectures

Array Funnel Shifter

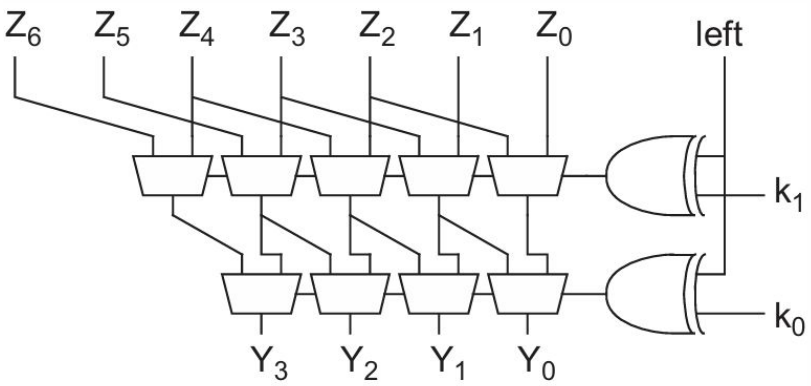
- an array of N N-input multiplexers accepting select signals (one multiplexer for each output bit).
- Implemented using NMOS pass transistor logic.
- High speed for small shifters in transistor-level designs
- A funnel shifter is a generalized barrel shifter, and it is called a funnel since bits from two registers are being poured into a "wide funnel," then an output window of the same width is extracted as a single register.
- The extra inputs may cause layout problems that should be accounted for.

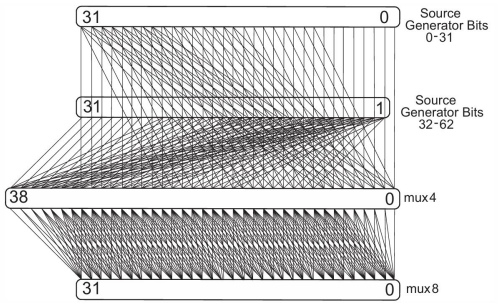
Shift Type	$Z_{2N-2:N}$	Z_{N-1}	$Z_{N-2:0}$	Offset
Logical Right	$A_{N-2:0}$	A_{N-1}	$A_{N-2:0}$	k
Arithmetic Right	0	A_{N-1}	$A_{N-2:0}$	k
Rotate Right	sign	A_{N-1}	$A_{N-2:0}$	k
Logical/Arithmetic Left	$A_{N-1:1}$	A_0	$A_{N-1:1}$	$\bar{k}$
Rotate Left	$A_{N-1:1}$	A_0	0	$\bar{k}$

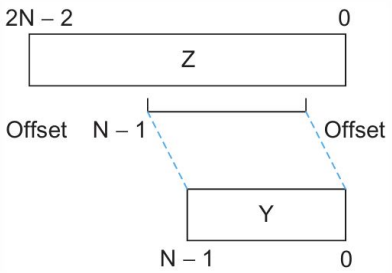


Logarithmic Funnel Shifter

- Uses a multi-stage, power-of-two shift structure
- Instead of using one giant array to jump directly to any position, it uses multiple smaller jumps in stages, which is much more hardware-efficient
- It also consumes less power due to the reduced fan in, and there is no longer a need for a complex array implementation
- However the multistage implementation increases the delay
- The source generator is the circuit block (often built with wiring and multiplexers) that creates that concatenated input word for the funnel shifter's first level of multiplexers.
- The source generator + first-level multiplexers takes much more silicon area in width than the rest of the datapath, making routing and timing awkward.
- This is a floorplan (32 bits) in which the source generator is folded to fit the datapath. Such folding also reduces wire lengths, saving energy. Instead of laying all input bits in a single, wide row, designers fold (bend) part of the circuit — into another row above or below the main datapath.

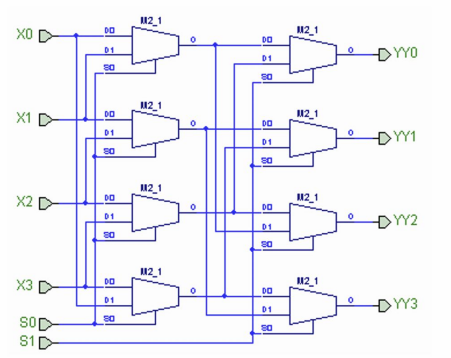


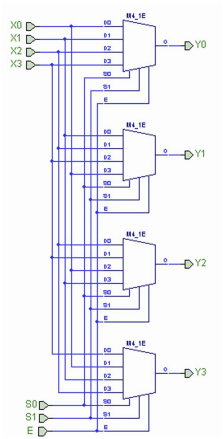


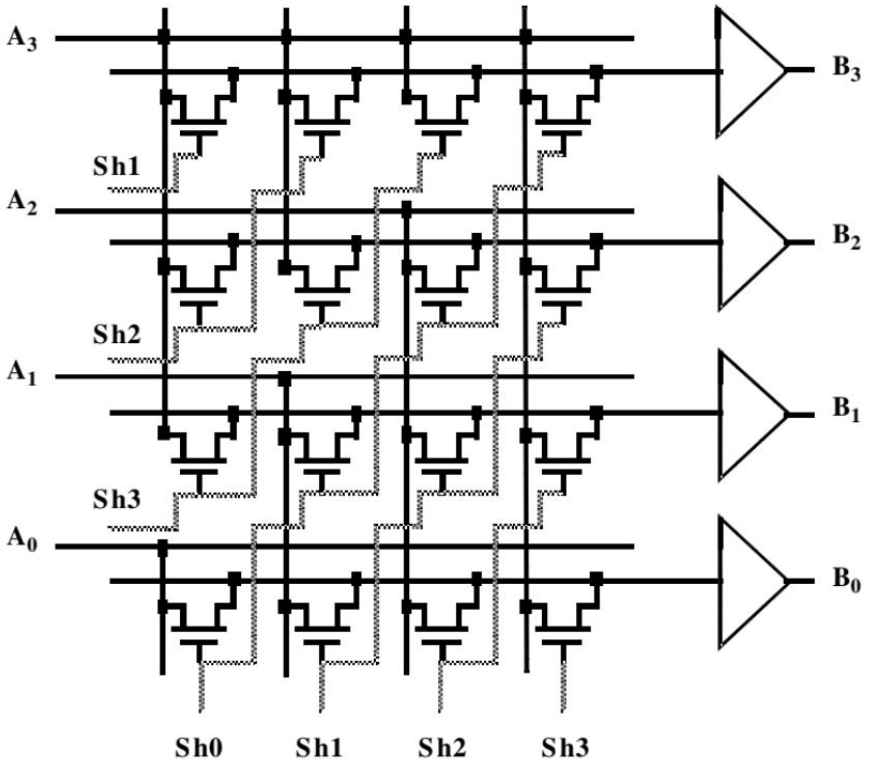


Array Barrel Shifter (without masking logic)

- A barrel shifter is designed primarily around rotation operations, treating shifts as modified rotations. Imagine it as a "barrel" where bits can roll around in a circle: the input data is rotated by the specified amount, and then extra logic (like masking) adjusts for non-rotation shifts.
- Here the multiplexers are implemented using NMOS pass transistor logic

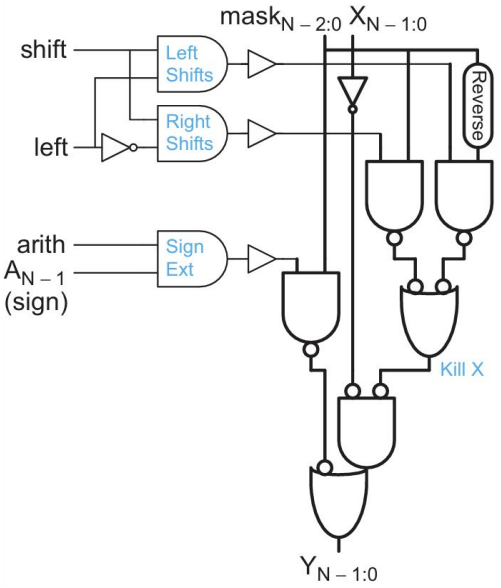


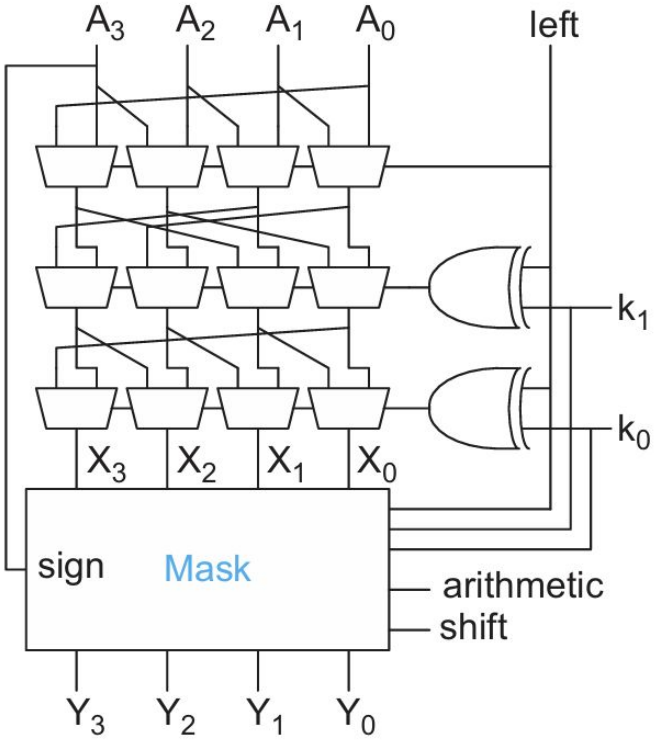




Logarithmic Barrel Shifter

- Similar to an Array Barrel shifter but with added masking logic and the multiplexer network implemented in multiple stages
- If only certain shifts are supported, the unit can be simplified, and if only rotates are supported, the masking unit can be eliminated, saving substantial hardware, power, and delay.
- The fastest shifters are implemented as full crossbars or arrays. These incur the least delay, with the output always a single gate delay behind the input to be shifted.





Multiplexer Implementation

- The 2-1 MUX implemented using static CMOS, providing:
- Full rail-to-rail voltage swing without signal degradation
- Better noise margins
- Easier cascading and logic integration
- No static power consumption
- A 4x1 MUX can easily be implemented using three 2x1 MUXes as shown in the configuration on the right

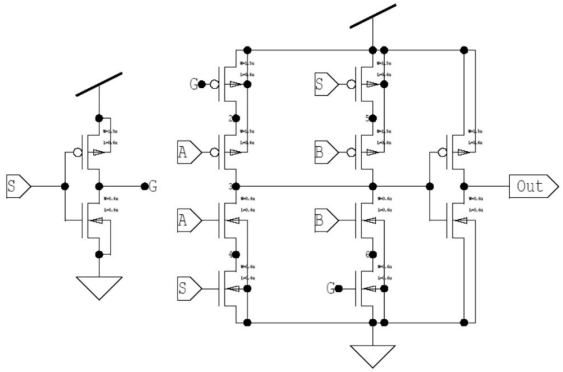


Fig. 3. A schematic of 2:1 MUX cell using optimized static CMOS logic

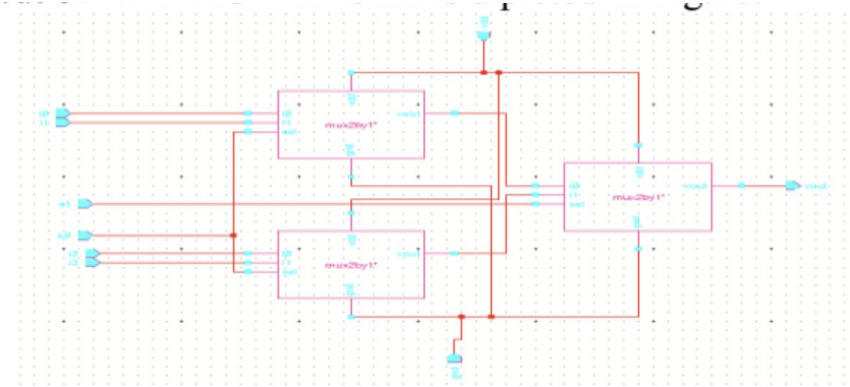


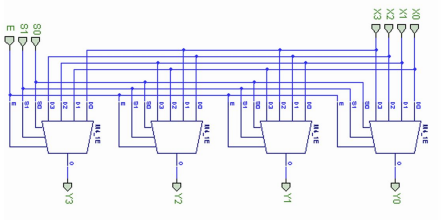
Fig.8. Schematic of 4X1 MUX

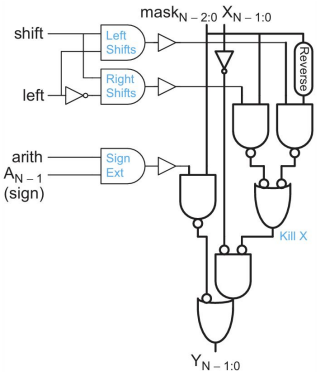
Neeraja B, Ramya K, Design and Analysis of an Energy Efficient 4-bit Barrel Shifter Circuits in 45nm Technology

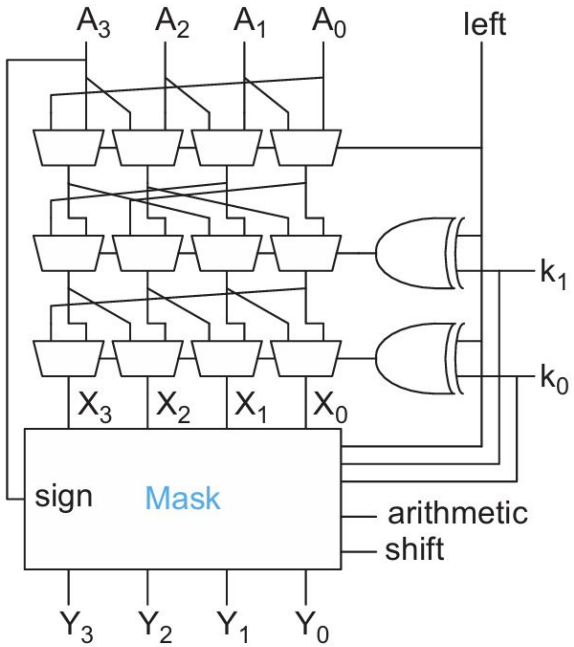
Abhijit Asati and Chandra Shekhar, A Purely MUX Based High Speed Barrel Shifter VLSI Implementation Using Three Different Logic Design Styles

Proposed Shifter Architecture

- We decided to use a barrel shifter since a funnel shifter has a higher fan in and causes layout difficulties. And we use a similar masking logic as in the logarithmic barrel shifter to implement all the shift operations.
- A cascade of parallel 2×1 multiplexers can be used instead, which allows a large reduction in gate count, now growing only with *n* log(*n)*; however the propagation delay is larger, growing with log *n* (instead of being constant as with the array shifter).
- Built from $\log_2(N)$ stages where stage i either shifts by 2^i bits or passes the data. The data must traverse those stages, so the critical path (delay) grows roughly proportional to $\log_2(N)$ (i.e. O(log N)). This implementation uses much less hardware (≈ $N \cdot \log_2 N$ muxes).
- Use a crossbar if you need the lowest possible latency and can afford the large area/ wiring cost (or for small widths where N^2 is acceptable).
- Therefore we replace the two levels at the bottom with a cross-bar like implementation.





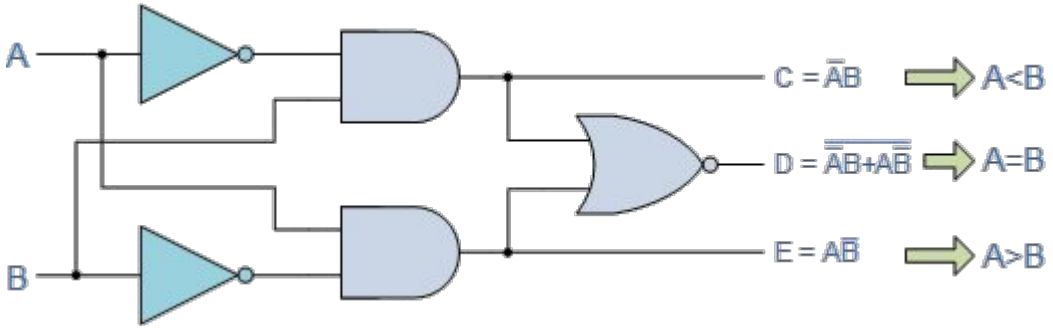


N. Weste, D. Harris ,*CMOS VLSI Design : A Circuits and Systems Perspective*

Different Magnitude Comparators Architectures

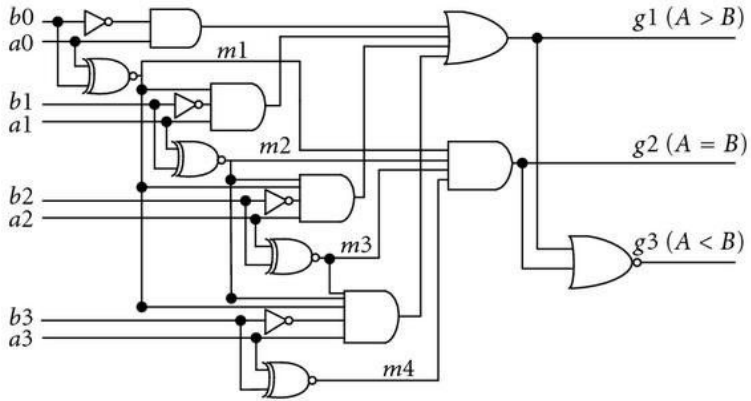
Types of Comparators

- Analog Comparators
  - Voltage Comparators
  - Current Comparators
  - Window Comparators
  - Time Delay Comparators
- Digital Comparators
  - Identity comparator
  - **Magnitude comparator**

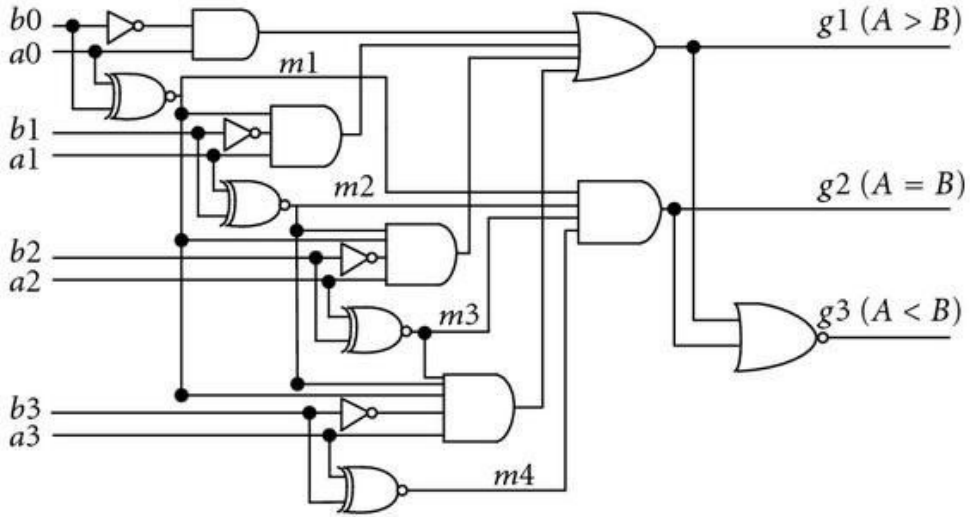


Types of Comparators

- In our application we are using a 4 bit digital comparator
- We have two choices identity comparator or magnitude comparator
- We will be using a magnitude comparator because it performs all of the functions that we need not just equality (identity comparatorsity checks if A = B only)
- Needed functions:
  - A = B
  - A > B
  - A < B
  - A ≤ B



Conventional Comparator



Architecture Comparison

Architecture / Logic Style	Structural Approach	Critical Limitation	Area / Transistor Count	Suitability with chosen frequency
Iterative (Serial)	Sequential/Ripple	Propagation delay scales linearly (low speed due to the accumulation of sequential delay)	Lowest complexity per bit, but overall inefficiency	Unsuitable
Parallel	Combinational (Fixed Depth)	High resistance due to deep NMOS/PMOS transistor stacking (high fan-in)	High; large gate size required to overcome stacking resistance	Optimal

Possible Architectures

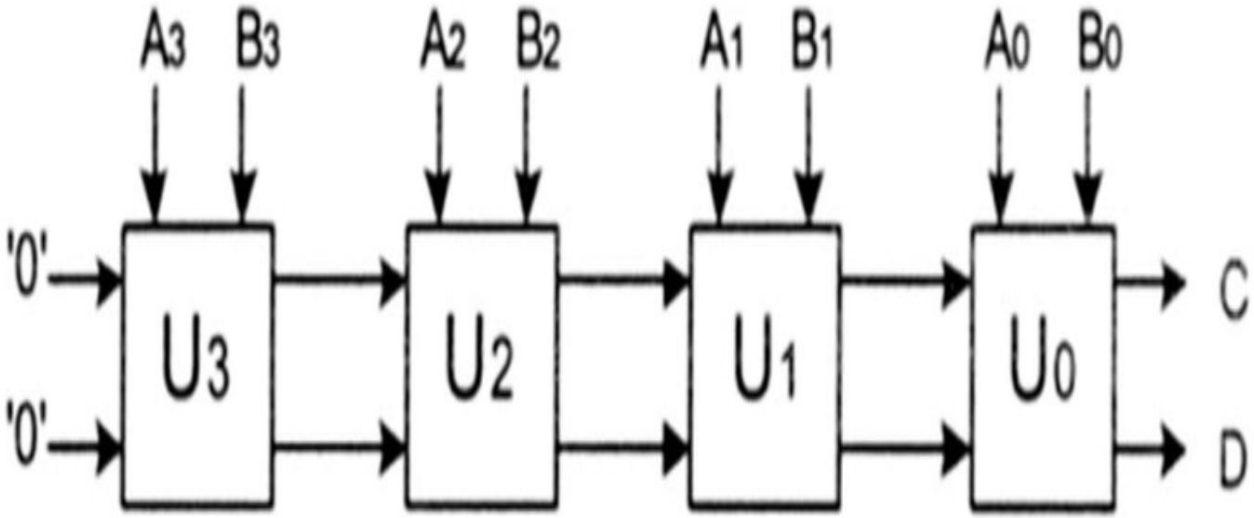
Architecture	Family	Latency	Area/gates count	Dynamic Power	Scalability	Additional Comments
Bitwise cascade (ripple)	Iterative (Serial)	Highest: accumulated delay	Low: minimal gates per stage	Lowest	Poor: grows linearly	Poor speed, Low complexity
Proposed Architecture	Parallel	Low (speed limited by fan-in stacking)	Moderate	Moderate: high fan-in/fan-out gates (total capacitance)	Good: scales to 8–16b with balanced trees	Fast for 4-bit without over-engineering
Parallel-prefix	Parallel (advanced)	Very low	Highest: prefix network overhead	Highest: more wires/activities	Excellent: shines at ≥16–32b	Overkill for 4-bit; routing and control complexity in 65nm

Architecture Comparison

Table 3: **Performance comparison of two-bit digital comparator**

Parameters	Conventional	Transmission gate logic [5]	Proposed transmission gate logic	Half adder logic technique	X–E logic
Power consumption (μW)	0.62	0.47	0.28	0.12	0.33
Number of transistor	54	74	30	20	24
Propagation delay (n-sec)	8.40	8.49	7.63	6.87	7.96
Power delay Product (μ-nJ)	5.20	3.99	2.13	0.82	2.62

Iterative (Serial/Ripple) Structure

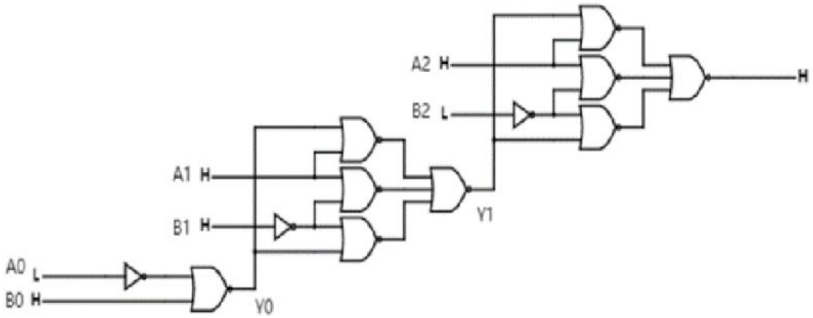


Iterative (Serial/Ripple) Structure: how it works

Component/Stage	Function	Logic/ Condition	Output Propagation
Overall Structure	Breaks the 4-bit comparison into four separate 1-bit stages	Decision "ripples" from the MSB to the LSB	The final result is the output of the LSB stage
Bit Comparison	Compares the individual bits, A_i and B_i	Generates the bit equivalent signal (X_i): $X_i = A_i$ XNOR B_i	$X_i = 1$ if $A_i = B_i$; $X_i = 0$ if $A_i \neq B_i$
Cascading Logic (If $A_i = B_i$)	Passes the result from the previous, more significant stage	$A_i = B_i$ (or $X_i = 1$)	The previous stage's result ($A > B$, $A < B$, or $A = B$) is passed through to the next cell (i-1) via an enable signal (X_i)
Cascading Logic (If $A_i \neq B_i$)	Immediately determines the overall magnitude result	$A_i \neq B_i$ (or $X_i = 0$)	The comparison decision is immediately made at stage i and the result is propagated forward (overwriting previous stages' "equal" results)

Iterative (Serial/Ripple) Structure: How it Works

- The comparison logic for the Least Significant Bit (LSB) group (A0, B0) produces the result V0.
- This result V0 is used as a critical input to the logic group calculating bit 1 (A1, B1).
- This entire group's result is Y1.
- The result Y1 is then used as a critical input to the logic group calculating bit 2 (A2 ,B2).
- This group produces the final output H

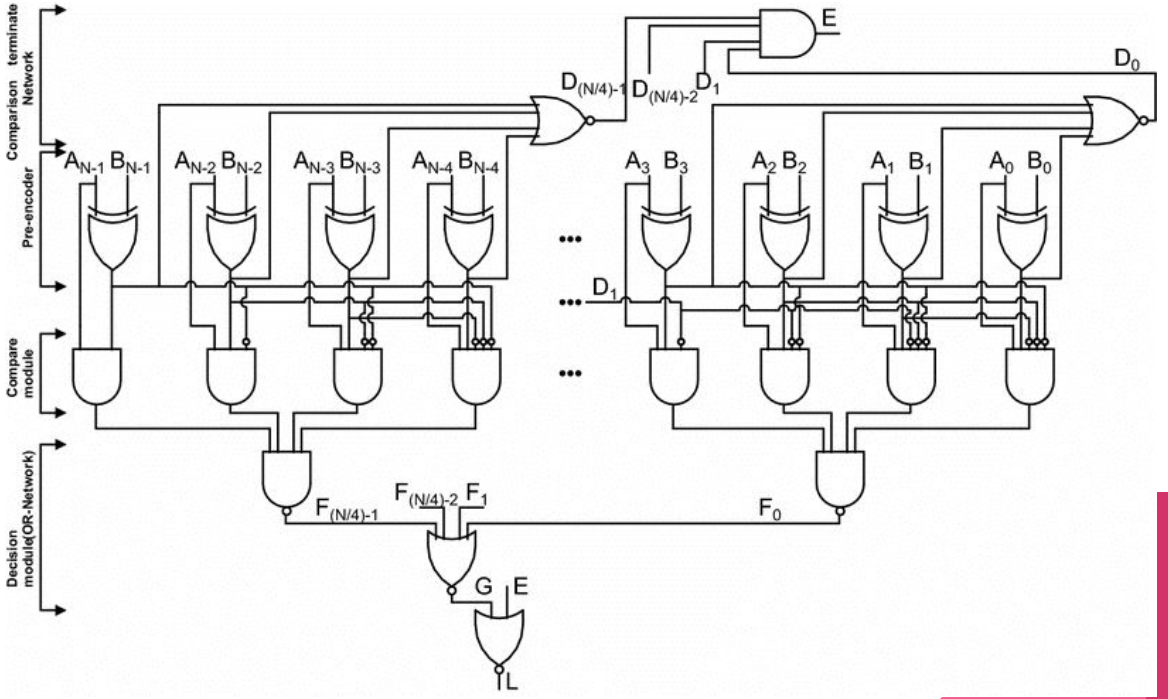


Parallel (Combinational) Structure

- Parallel structure is the most suitable for our application.
- Using the serial structure gives us high delay which is highly undesirable for our high performance application.
- The parallel structure family has some architectures underneath it:

    1. Parallel-Prefix
    2. XNOR with Parallel Priority Logic

Parallel Prefix



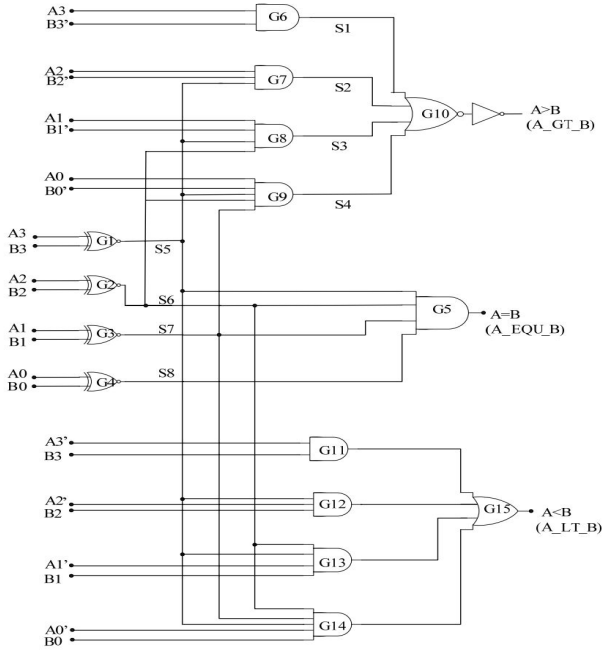
B. P. A. N. and D. V., "Proposed parallel binary comparator using conventional EX-OR gates," ResearchGate, fig. 6. [Online]. Available: https://www.researchgate.net/figure/Proposed-parallel-binary-comparator-using-conventional-EX-OR-gates-PA1_fig6_317974073

Parallel Prefix: how it works

	Function	Explanation
Pre-encoder network	Detects if each bit pair is equal or not	It uses XNOR or XOR to find if each bit of A and B are the same. Example: if A3 = 1, B3 = 1 then this bit is equal
Compare module	Finds whether A>B or A<B within small bit groups (e.g., 4 bits)	It checks within small “blocks” of bits (like A3–A0 vs B3–B0) which group is larger, using AND/OR logic
Decision module (OR-network)	Combines the results of each group to get the final comparison result	It takes the outputs from all groups and decides overall A>B, A<B, or A=B using OR and AND logic

S. K. S, R. P, and P. R. L., "Area-Efficient Parallel-Prefix Binary Comparator," in *Lecture Notes in Electrical Engineering*, vol 994. Singapore: Springer, 2023, pp. 785-794. [Online]. Available: https://www.researchgate.net/publication/339473181_Area-Efficient_Parallel-Prefix_Binary_Comparator

Proposed Architecture



Logic diagram for a 4-bit comparator

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

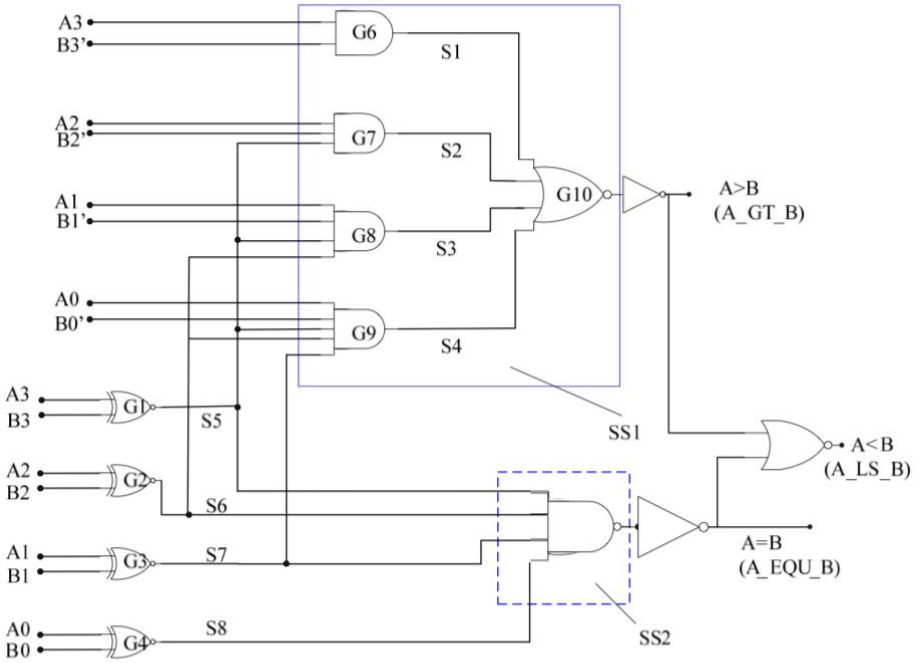
- We recommend using the XNOR equality tree + priority logic comparator architecture for our 4-bit high-performance ALU that uses 65nm technology, 1.0GHz and 1.8V.

- This is because:
  - It gives the best balance of fixed, low latency and moderate area
  - Parallel prefix is unnecessary overhead at 4 bits
  - Serial ripple risks timing variability and worst-case delay that can bottleneck the ALU

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

- In order to decrease the area we can decrease the number of gates.
- We can do A<B using A>B and A=B.
- We can easily use a 2-input NOR gate to find if A<B.
- The area of the 2-input NOR gate is much less than that of gates 11-15.
- Another main benefit is that when we reduce the number of gates the power dissipation also decreases



Optimized logic diagram for a 4-bit comparator

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

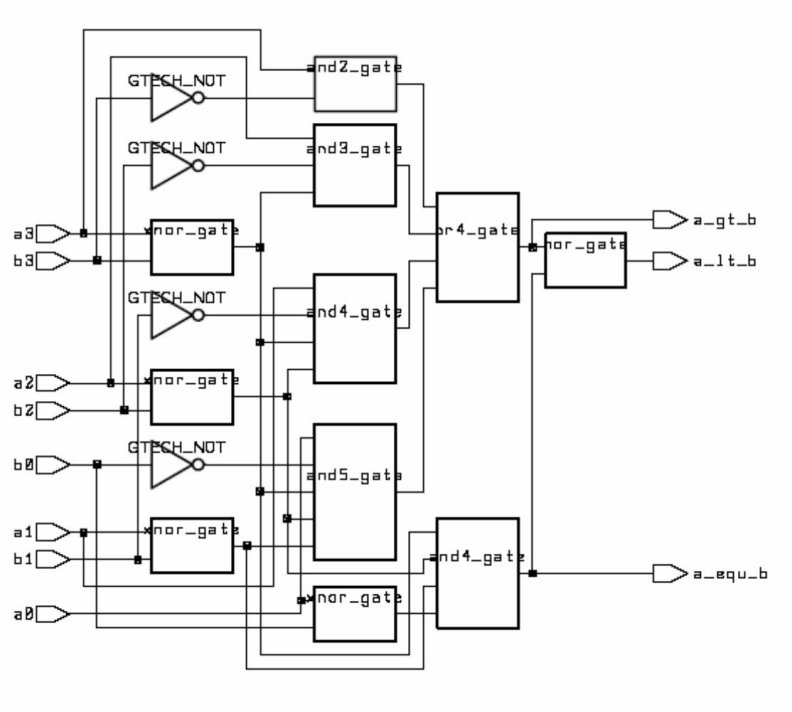


Figure 2.6 Logic simulation diagram

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture: How it Works

	Gates	Description
Bitwise Equality Detection	4 XNOR gates (G1–G4)	Each gate compares one bit pair (Ai, Bi). The output (S5-S8) is 1 only when the two input bits are equal
Priority (A > B) Network	Gated AND-OR logic: G6, G7, G8, G9 (AND gates) and G10 (OR gate made from NOR + NOT gate)	This is the priority logic. It detects the first (most-significant) bit where A[i]=1 and B[i]=0, only if all higher bits are equal (using S5-S7 as "enables"). G10 combines these conditions
Equality (A = B) Network	Multi-input AND gate (a 4-input NAND + NOT gate)	It takes all the bitwise equality signals (S5-S8) and outputs 1 only if all of them are 1 (all are equal)
Less-Than (A < B) Logic	2-input NOR gate	Generated by combining A > B and A = B outputs. It implements the logic: "If A is NOT greater than B OR A is NOT equal to B", then A must be less than B

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Table 2.4 Truth table of 4-Bit Comparator

COMPARING INPUTS				OUTPUT		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**A > B**	**A < B**	**A = B**
A3 > B3	X	X	X	H	L	L
A3 < B3	X	X	X	L	H	L
A3 = B3	A2 >B2	X	X	H	L	L
A3 = B3	A2 < B2	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H
H = High Voltage Level, L = Low Voltage, Level, X = Don’t Care						

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

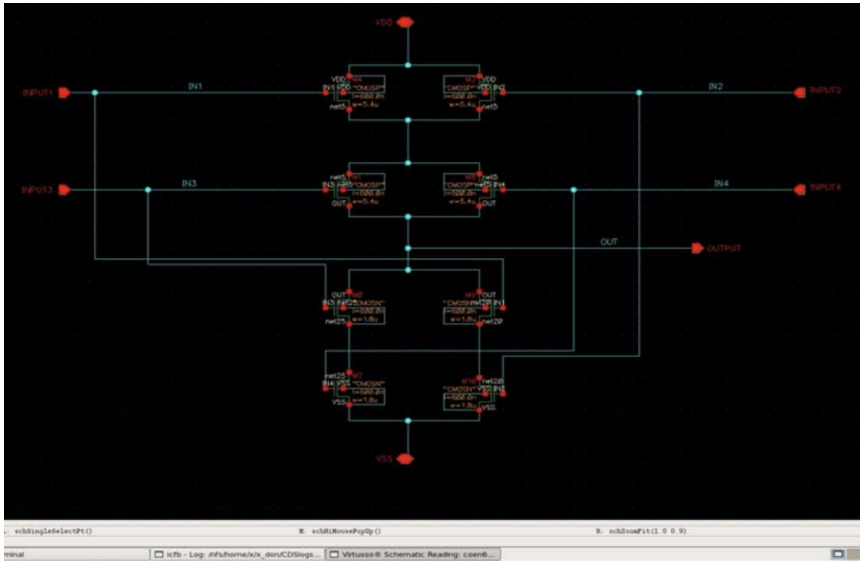


Figure 3.1 Schematic of XNOR

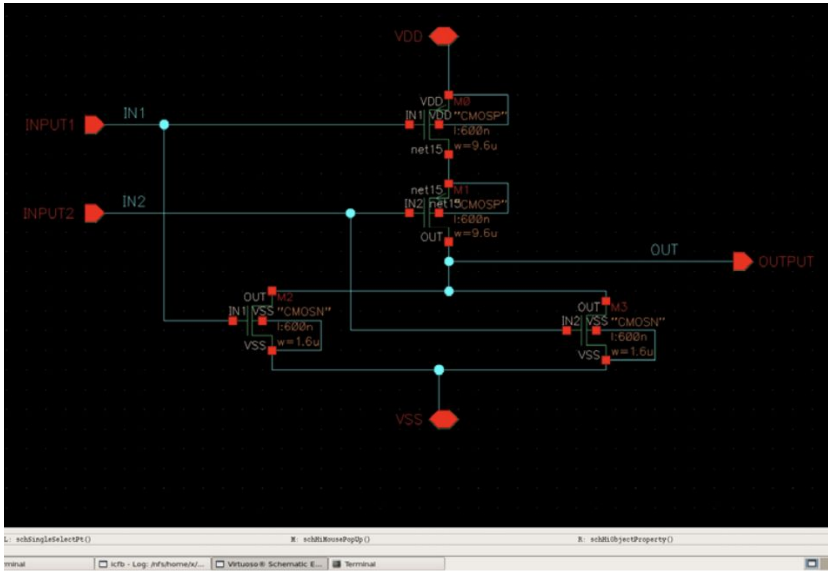


Figure 3.3 Schematic of NOR

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

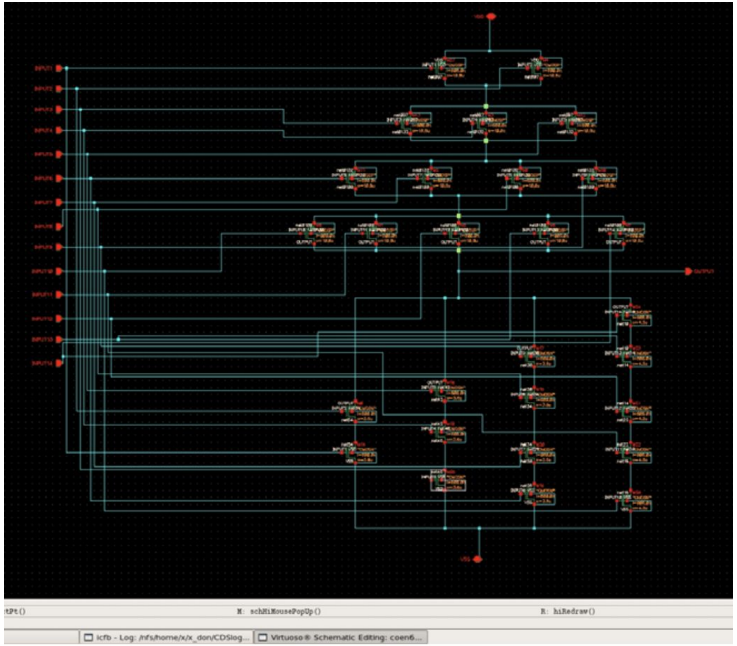


Figure 3.5 Schematic of SS1

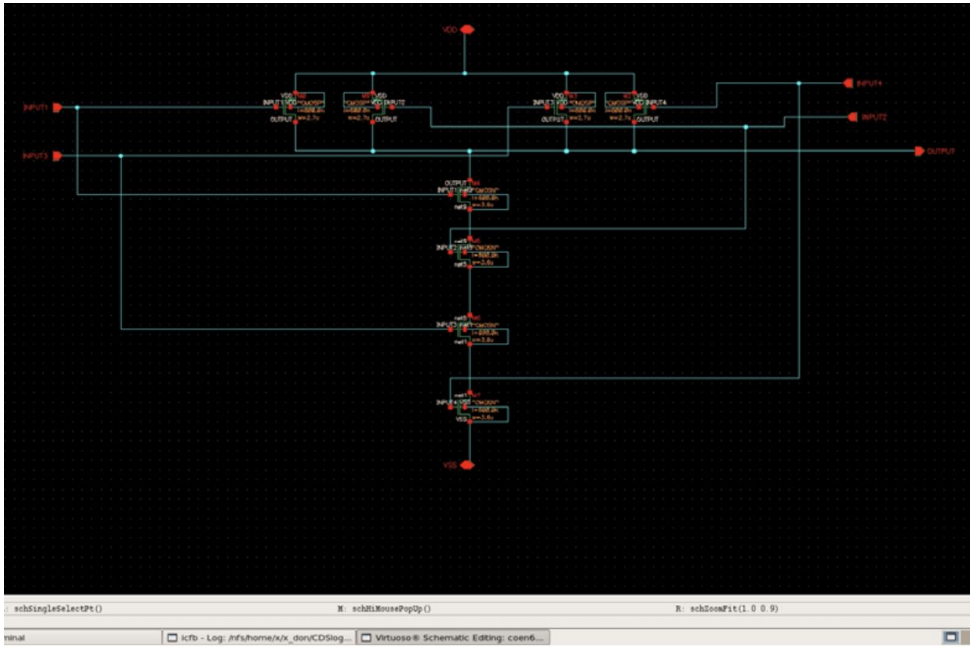


Figure 3.7 Schematic of SS2

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Proposed Architecture

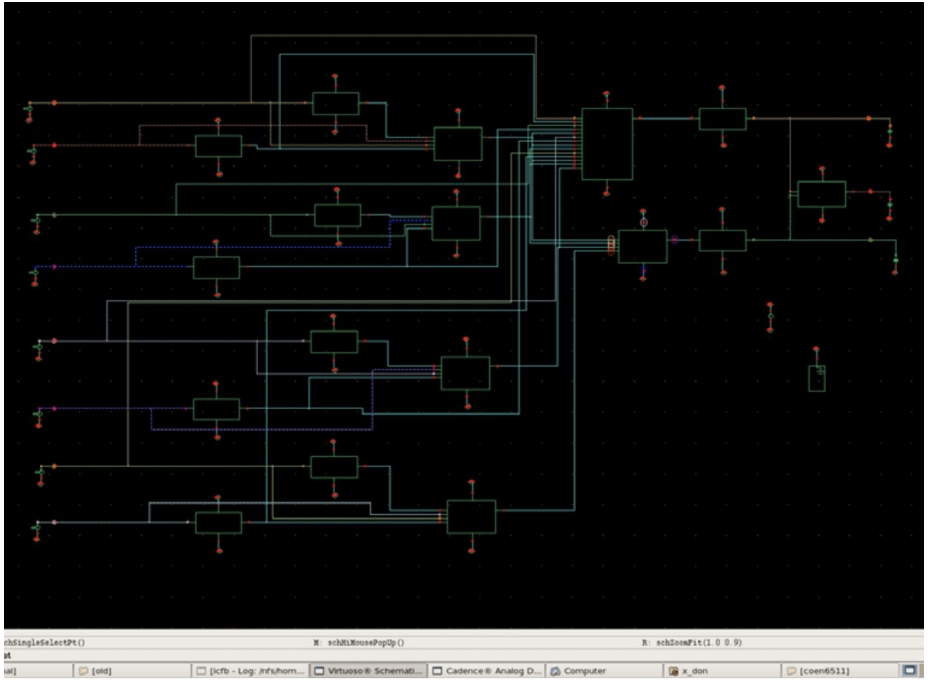


Figure 3.9 Testbench schematic of whole 4-bit comparator

M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

Delay Calculations

Going from 500nm technology to 65nm technology (ideally)

$$K\ (scaling\ factor) = \frac{Old\ size}{New\ size} = \frac{500\ nm}{65\ nm} \approx 7.7$$

$$TP_{HL\ new} = \frac{TP_{HL\ old}}{K} = \frac{1102.7\ ps}{7.7} \approx 143\ ps$$

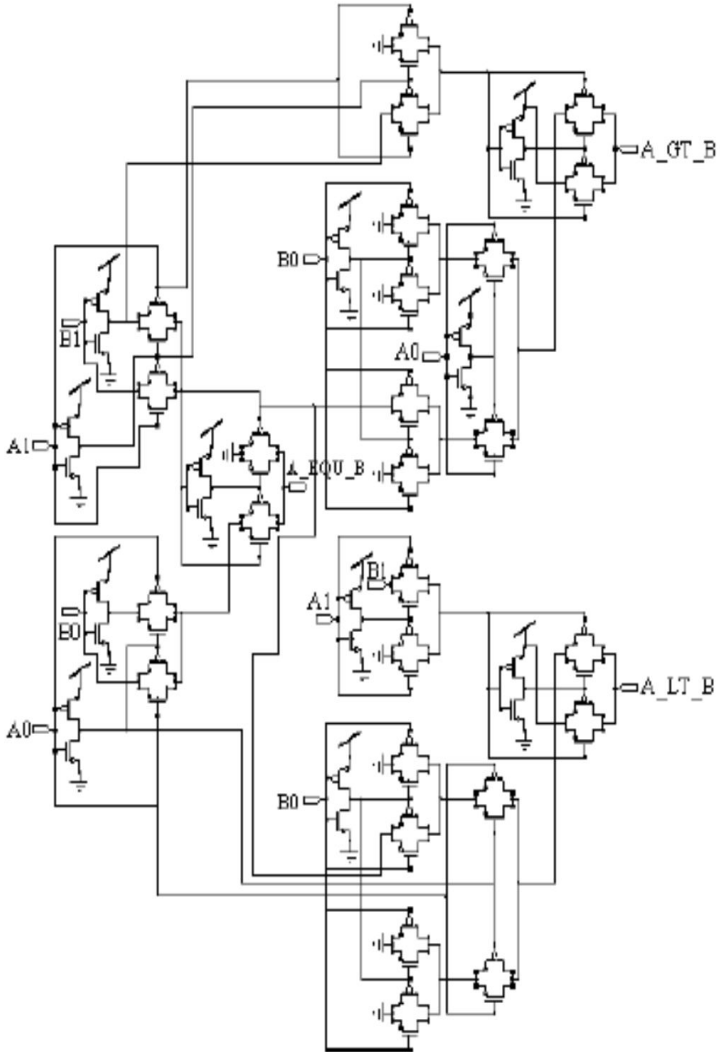
$$TP_{LH\ new} = \frac{TP_{LH\ old}}{K} = \frac{992.33\ ps}{7.7} \approx 128.9\ ps$$

$$T_{rise\ new} = \frac{T_{rise\ old}}{K} = \frac{349.5\ ps}{7.7} \approx 45.4\ ps$$

$$T_{fall\ new} = \frac{T_{fall\ old}}{K} = \frac{340.5\ ps}{7.7} \approx 44.2\ ps$$

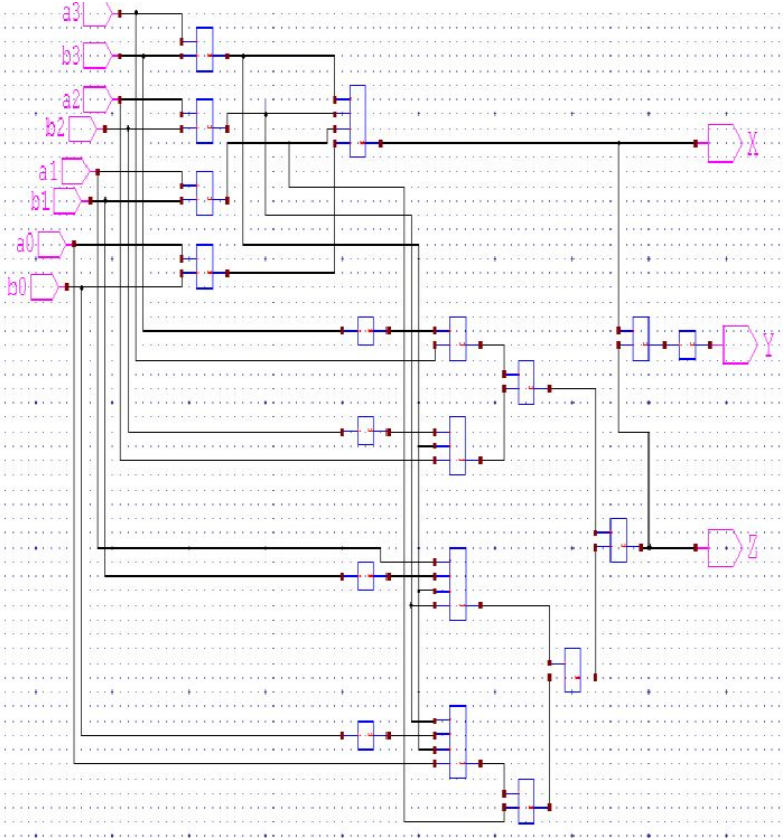
Proposed Architecture using PTL

- The Parallel Comparator, realized in Pass Transistor Logic (PTL)
- It is functionally an XNOR equality tree combined with priority logic
- It was implemented in PTL to achieve area efficiency and reduced delay
- While this makes it well-suited for a 4-bit ALU, static CMOS remains the safer, more scalable baseline due to PTL's inherent trade-offs in robustness and full voltage swing
- PTL doesn’t give us a strong 0 or 1



Proposed Architecture using TGL

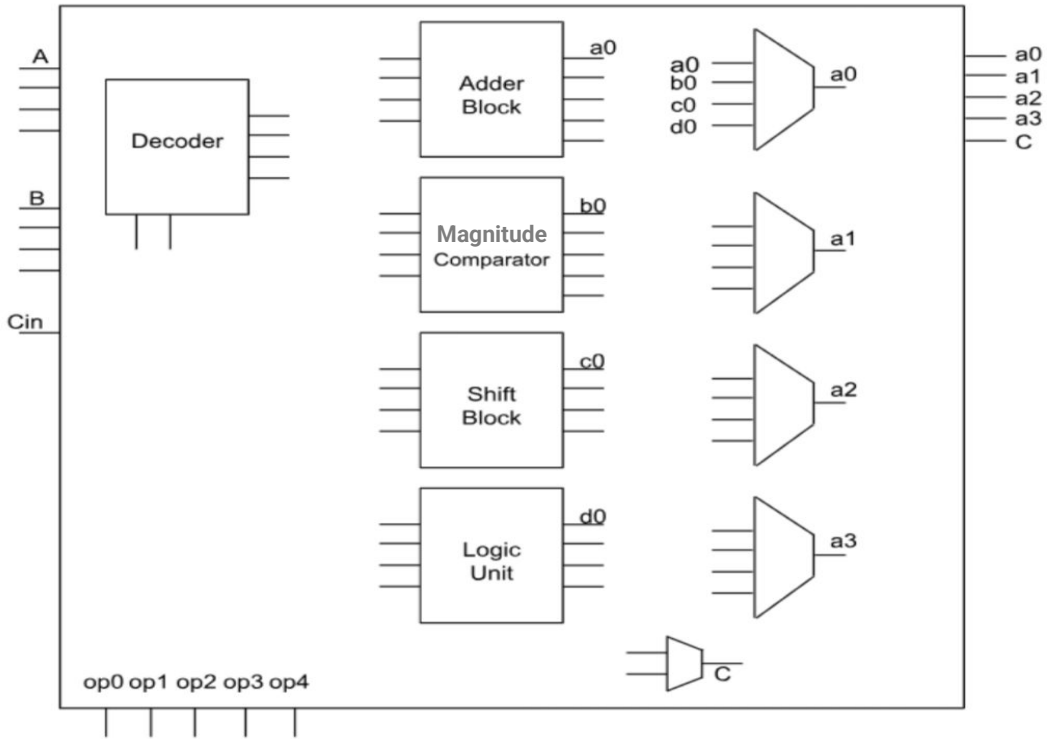
- The Parallel Comparator is realized in Transmission Gate Logic (TGL)
- Functionally, it uses an XNOR equality tree combined with priority logic
- It was implemented in TGL to achieve power efficiency
- Since, in a high performance application we care most about speed not power, CMOS is still preferred



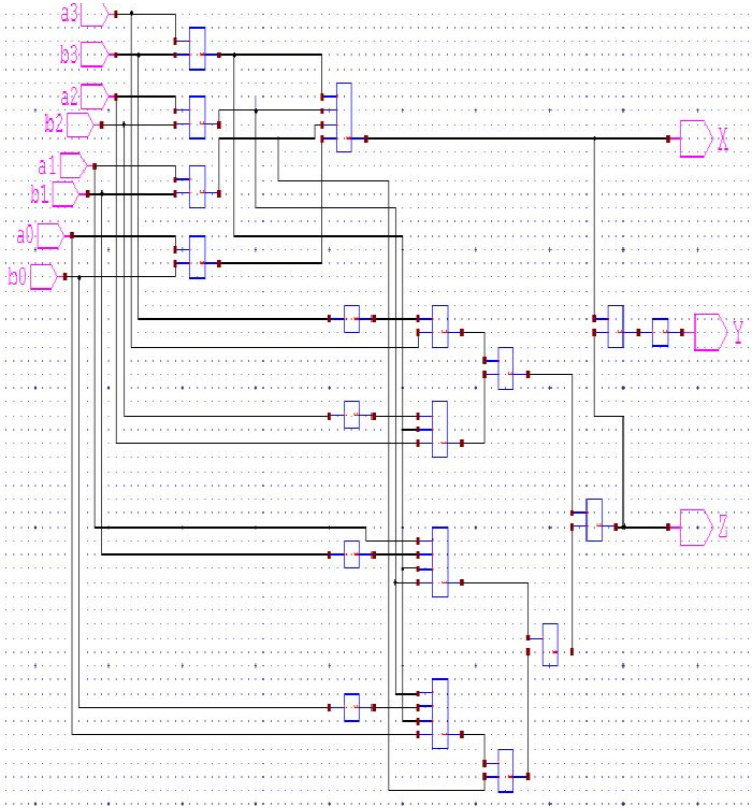
Additional Operation

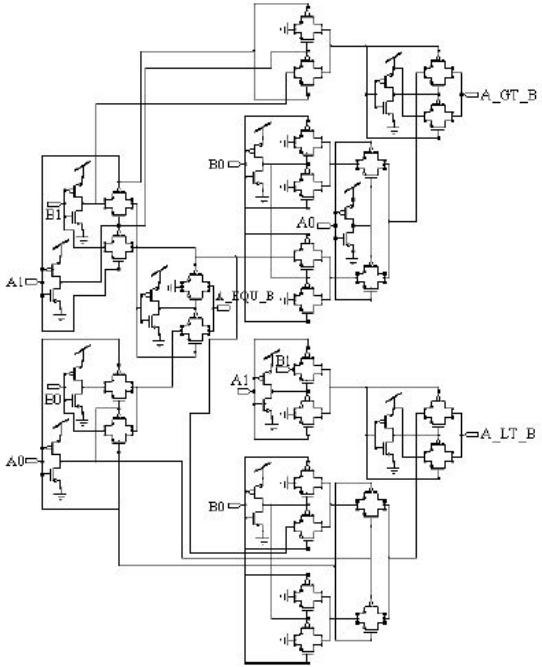
- After choosing the most appropriate architecture for our magnitude comparator there was only one thing left to do; add a block to check if A is less than or equal to B
- To do this we will be using a 2-input OR gate
- The first input is A<B and the second input is A = B
- When either one of the inputs is true (logic ‘1’) the output will be 1 (A is less than or equal to B)
- Else, if both are not true (logic ‘0’) then A is greater than B

Final Overall Architecture

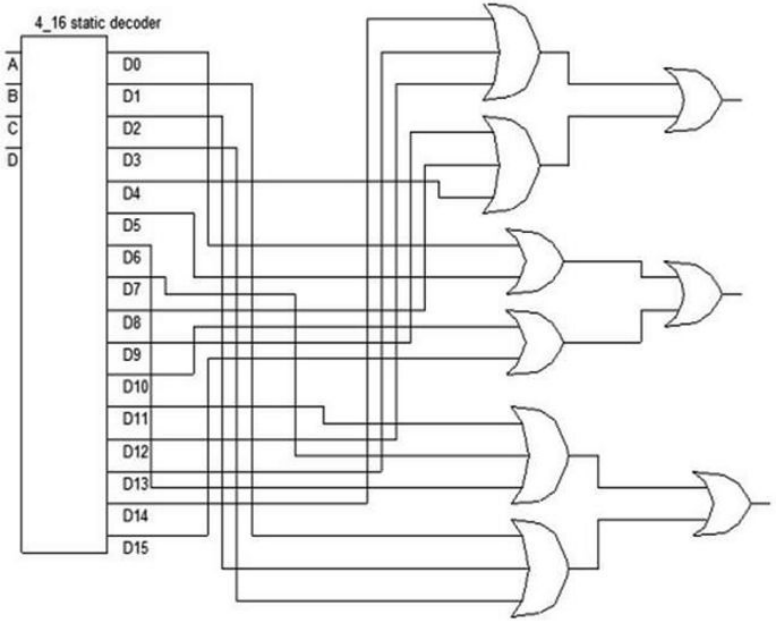


TGL Comparator

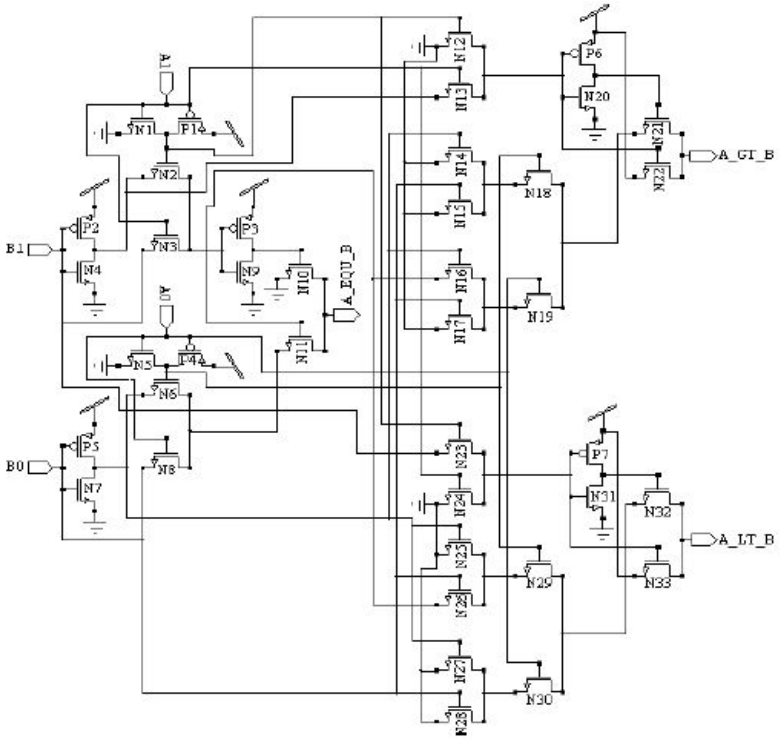




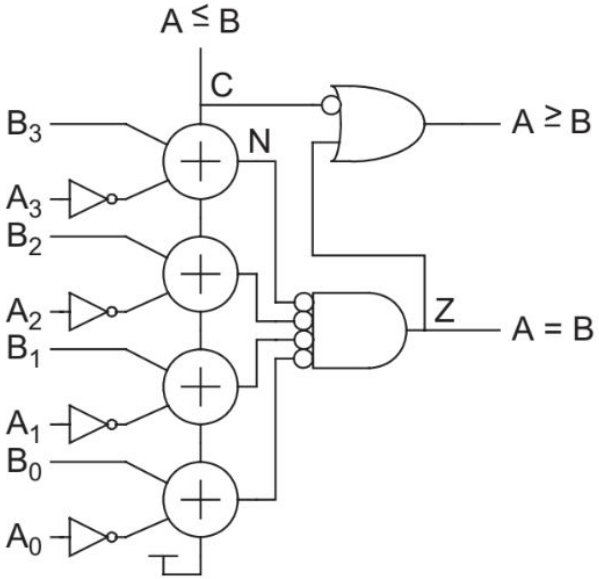
GDI Comparator



PTL Comparator



Comparator built from a carry-ripple adder and two's complement



Relation	Unsigned Comparison	Signed Comparison
$A = B$	Z	Z
$A \neq B$	$\overline{Z}$	$\overline{Z}$
$A < B$	$C \cdot \overline{Z}$	$\overline{S} \cdot \overline{Z}$
$A > B$	C	S
$A \leq B$	C	$\overline{S}$
$A \geq B$	$\overline{C} + Z$	$S + Z$

References

[1] M. Hasan, M. S. Uddin, and M. S. Hossain, "High-Performance Design of a 4-Bit Carry Look-Ahead Adder in Static CMOS Logic," in *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, 2020, pp. 268-273.

[2] Aragio Solutions, "65nm G (TSMC) 2.5V / 2.5V LVDS I/O Library," Product Brief, Rev. 1a. [Online]. Available: https://www.aragio.com/assets/PDFfile/resources/TSMC/65nm/rgo_tsmc65_25v25_lvds_product_brief_rev_1a.pdf

[3] S. V. D. A. Prasad, G. S. P. Varma, B. R. K. Reddy, and S. S. S. N. Raju, "A High Speed 4-Bit Magnitude Comparator in Digital VLSI Circuit Design," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 19, pp. 129-132, Feb. 2019.

[4] S. Dutta, S. Kumar, S. Kundu, and A. K. Saha, "A High-Speed, Low-Power, and Area-Efficient 4-Bit Magnitude Comparator," *Electronics*, vol. 11, no. 7, p. 1001, Mar. 2022.

[5] R. P. Dasari *et al.*, "A high speed 4-bit carry look ahead adder," in *2009 WRI Global Congress on Intelligent Systems*, 2009, vol. 3, pp. 491-494.

[6] TutorialsPoint.com, "Digital Electronics - Comparators." [Online]. Available: https://www.tutorialspoint.com/digital-electronics/digital-electronics-comparators.htm. [Accessed: Oct. 26, 2025].

[7] H. H. K. H. *et al.*, "A 1.2-V 1-GHz 32-bit integer execution unit in 65-nm SOI CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 11, pp. 2046-2053, Nov. 2004.

References

[8] J. Goll and R. Zimmermann, "A Comparator With Reduced Delay Time," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 6, pp. 484-488, June 2009.

[9] K. K. Pradhan *et al.*, "Efficient subthreshold leakage current optimization," in *18th International Conference on VLSI Design*, 2005, pp. 235-240.

[10] S. S. S. R. S. *et al.*, "A Novel 4-bit Binary-Coded Decimal (BCD) Adder," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 1, pp. 2221-2224, May 2019.

[11] N. K. G., "Design of Static CMOS Barrel Shifters," M.S. thesis, Dept. Elect. Comput. Eng., Louisiana State University, Baton Rouge, LA, USA, 2003.

[12] P. R. P. D. S., "Low Power and High Speed 4-bit Magnitude Comparator using Pass Transistor Logic," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 1152-1156.

[13] World of Computing, "Magnitude Comparator in Digital Electronics." [Online]. Available: https://worldofcomputing.net/digital-electronics/magnitude-comparator.html. [Accessed: Oct. 26, 2025].

[14] R. S, A. M, and S. S., "A 32-bit parallel binary comparator using 2-bit comparator blocks," *International Journal of Applied Engineering Research*, vol. 12, no. 15, pp. 5092-5096, 2017.

References

[15] M. A. A, A. A, and M. S. A. A., "Design and Implementation of a 4-Bit ALU," Course Project Report, COEN 6511, Concordia University, Dec. 2004. [Online]. Available: https://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

[16] S. K. S, R. P, and P. R. L., "Area-Efficient Parallel-Prefix Binary Comparator," in *Lecture Notes in Electrical Engineering*, vol 994. Singapore: Springer, 2023, pp. 785-794. [Online]. Available: https://www.researchgate.net/publication/339473181_Area-Efficient_Parallel-Prefix_Binary_Comparator

[17] R. P, A. B, and B. G., "Design and analysis of 1-bit magnitude comparator using different logic styles," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 4, no. 7, pp. 3122-3126, Jul. 2013.

[18] M. P, C. M, and P. B, "Design of High Speed 4-Bit Magnitude Comparator Using 90nm CMOS Technology," *International Journal of Research in Engineering and Technology (IJRET)*, vol. 3, no. 7, pp. 367-370, Jul. 2014.

[19] K. V. S. R. P. Varma and M. V. D. Prasad, "A Novel Low Power Comparator using Transmission Gate," in *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015, pp. 1111-1114.

[20] Electronics-Tutorials.ws, "Magnitude Comparator." [Online]. Available: https://www.electronics-tutorials.ws/combination/comb_8.html. [Accessed: Oct. 26, 2025].

[21] S. G. P. K. S, S. S. N. V, N. B, and S. A. G, "DESIGN AND COMPARISION OF LOW POWER HIGH SPEED 4 BIT -ALU," in *2021 2nd International Conference on Communication, Computation and Power (ICCCP)*, 2021.

References

[22] M. V. B, N. B. B, P. A, R. C, and G. B, "A fast ALU design in CMOS for low voltage operation," *VLSI Design*, vol. 14, no. 4, pp. 315-327, 2002.

[23] K. S. D, M. P. N, and V. R, "Design and Implementation of 4-bit Arithmetic Logic Unit," *International Journal of VLSI design & Communication Systems (IJVDCS)*, vol. 4, no. 3, pp. 133-140, June 2013.

[24] A. V *et al.*, "Implementation of High Performance 4-Bit ALU using Dual Mode Pass Transistor Logic," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, pp. 1600-1605.

[25] M. I. Younis and K. Z. Zamli, “Schematic diagram for the 4-bit magnitude comparator,” ResearchGate, fig. 3. [Online]. Available: https://www.researchgate.net/figure/Schematic-diagram-for-the-4-bit-magnitude-comparator_fig3_41449498

[26] R. Hashemian, “Highly parallel increment/decrement using CMOS technology,” in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, 1990, pp. 932–935. doi:10.1109/MWSCAS.1990.140858

[27] [1] K. Prabhala and P. S. Raju, “A Hierarchical Design of 128 Bit Carry Lookahead Adder in 65 nm CMOS Technology,” *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 3, Jan. 2020.

References

[28] N. Weste, D. Harris ,*CMOS VLSI Design : A Circuits and Systems Perspective*

[29] DHANANJAY JADHAV, MITHILESH MULEY, MANGESH ASHTANKAR ,*VLSI DESIGN OF BARREL SHIFTER USING COMPLEMENTARY AND PSEUDO NMOS LOGIC*, Proceedings of 4th IRF International Conference, Pune, 16th March-2014, ISBN: 978-93-82702-66-5

[30] Neeraja B, Ramya K, *Design and Analysis of an Energy Efficient 4-bit Barrel Shifter Circuits in 45nm Technology*, International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958 (Online), Volume-9 Issue-4, April, 2020

[31] James Morizio, ECE 261, Shifters

[32] Priyanka Mandal, Siddhant Malani, Yogesh Gudepkar, Suparas Singhi and P.M.Palsodkar ,*VLSI Implementation of a Barrel Shifter*, Proceedings of SPIT-IEEE Colloquium and International Conference, Mumbai, India, Vol 2150

[33] B. P. A. N. and D. V., “Proposed parallel binary comparator using conventional EX-OR gates,” ResearchGate, fig. 6. [Online]. Available: https://www.researchgate.net/figure/Proposed-parallel-binary-comparator-using-conventional-EX-OR-gates-PA1_fig6_317974073