Design, Simulation, and Physical Implementation of a 5-Bit Pipelined Brent-Kung Carry Look-Ahead Adder

Harry Jain 2024112005
VLSI Design: Course Project (Monsoon 2025)
IIIT Hyderabad

***Abstract*—This report details the complete design flow for a high-speed 5-bit Pipelined Carry Look-Ahead (CLA) Adder, implemented in a $180\,\mathrm{nm}$ technology node. The architecture utilizes the Brent-Kung (BK) prefix structure to achieve logarithmic delay. Pipelining is achieved using True Single-Phase Clock (TSPC) D-Flip-Flops (DFFs) at the input and output. Logic gates are realized using a mix of styles for optimal performance: Static CMOS for the Black Cell and P/G generation, and Complementary Pass-transistor Logic (CPL) for XOR gates. All blocks, where possible, are sized for minimum area using $W_p = 4\lambda$ and $W_n = 2\lambda$. Schematic NGSPICE simulations confirm correct functionality, reporting a worst-case combinational delay of 50.12522 ns. Comprehensive post-layout extraction and simulation results are presented, followed by structural Verilog HDL verification and planned FPGA implementation.**

***Index Terms*—Brent-Kung Adder, Carry Look-Ahead (CLA), TSPC D-Flip-Flop, VLSI Design, NGSPICE, MAGIC Layout, Static Timing Analysis.**

I. Proposed Adder Structure (Q1)

A. Why this method

Brent-Kung is a prefix adder which make the whole circuit into multiple repetative modules, which are easy to layout and to scale also the fan-out of each stage is max 2 in Kogge-Stone adder which is the source of this idea, in Brent-Kung the redudntant path in Kogge-Stone is removed hence we have a fanout of max 3 in 5-bit adder

B. The Brent-Kung Structure (Two-Phase Proof)

The BK structure is proven by demonstrating that it computes all required prefix values $G_{0:i}$ in a minimal number of computational stages using a specific **recursive decomposition**. The total number of prefix nodes (logic gates) is $O(n)$, and the depth (latency) is $O(\log_2 n)$.

The computation is broken into two phases:

1) Phase 1: Forward Tree (Reduction/Compute Phase): This phase calculates all group generates $G_{i:j}$ where i and j are powers of 2 (or $i = 0$) in a bottom-up, ascending manner. It requires $\log_2 n$ stages for an n-bit adder.

The calculation proceeds recursively. At stage m (where $m = 1, \ldots, \log_2 n$), the architecture computes:

$$(G_{k-2^m+1:k}, P_{k-2^m+1:k}) =$$

$$(G_{k-2^{m-1}+1:k}, P_{k-2^{m-1}+1:k}) \circ (G_{k-2^m+1:k-2^{m-1}}, P_{k-2^m+1:k-2^{m-1}})$$

This is done only for bit indices k that are a multiple of 2^m.

Key Result after Phase 1: After $\log_2 n$ stages, the total group generate $G_{1:n}$ is calculated at the n-th bit position, and importantly, the **full prefix generate $G_{0:2^m-1}$ is calculated for all $m = 1, \ldots, \log_2 n$ at the bit positions $2^m - 1$.**

$$\text{Phase 1 Output: } G_{0:2^m-1} \text{ for } m = 1, \ldots, \log_2 n \quad \text{(at positions } 2^m - 1)$$

2) Phase 2: Backward Tree (Broadcast/Distribution Phase): This phase computes the remaining required prefix carries $G_{0:i}$ by propagating the results from Phase 1 in a top-down, descending manner. This phase also requires $\log_2 n$ stages.

At stage m (where $m = \log_2 n, \ldots, 1$), the calculation for a position k is:

$$(G_{0:k}, P_{0:k}) = (G_{0:k-2^m}, P_{0:k-2^m}) \circ (G_{k-2^m+1:k}, P_{k-2^m+1:k})$$

This step uses the already computed $G_{0:k-2^m}$ from the previous stage (or from Phase 1's results) to update the current group.

Key Result after Phase 2: By distributing the previously calculated carries, all necessary prefix generates $G_{0:i}$ are computed for all indices $i = 1, 2, \ldots, n-1$. Since $c_i = G_{0:i}$, **all intermediate carries are now available**.

$$\text{Phase 2 Output: } G_{0:i} = c_i \quad \text{for all } i = 1, \ldots, n-1$$

C. Complexity Proof

The total complexity is determined by counting the required logic nodes (prefix cells) and the depth (stages).

- **Logic Gates (Nodes):**
  - In an n-bit BK adder, Phase 1 has $\sum_{m=1}^{\log_2 n} 2^{m-1} = n - 1$ nodes.
  - Phase 2 has $\sum_{m=1}^{\log_2 n} (2^{m-1} - 1) \approx n/2$ nodes.
  - **Total nodes:** $N_{\textbf{total}} \approx \frac{3}{2}n - \log_2 n - 1 = O(n)$. This linear gate count is a major advantage over the Kogge-Stone adder, which has $O(n \log n)$ gates.
- **Latency (Depth):**
  - Phase 1 requires $\log_2 n$ stages.
  - Phase 2 requires $\log_2 n$ stages.
  - Initial g_i, p_i generation and final s_i calculation each add one stage.

– **Total Depth:** $D_{\textbf{total}} = 2 \cdot \log_2 n + 2 = O(\log n)$.

The Brent-Kung adder is thus proven to be a **logarithmic depth** adder that uses a **linear number of logic gates**, representing an efficient trade-off between gate area and speed.

D. *Previous Ideas*

A more optimal way is to use Han-Calsomn adder which use Brent-Kung at the initial and final stages of the flow and uses Kogge-Stone in the intermediate stages. But for 5 bit adder this does have that big of an advantage hence i preffered Brent-Kung over it. Another i dea was to use Dynamic gates for AND and OR operation specifically Domino logic and CPL for XOR(which i implimented in my final Brent Kung adder) this way we have the fast operation speed of Domino, but the complexity in wiring made me to choose brent kung over it.

E. *logic*

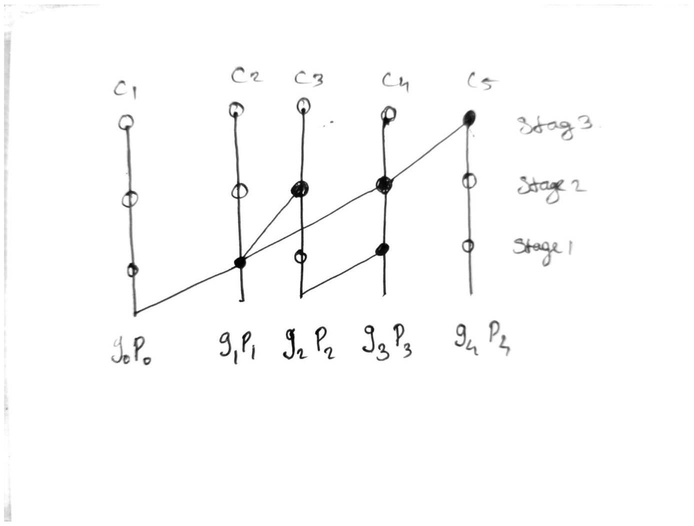
The core logic of the 5-bit adder is based on the **Brent-Kung (BK)** adder, a highly efficient prefix structure.

$$\text{DFF}_{\text{in}} \rightarrow \text{P/G Block} \rightarrow \text{CLA (BK)} \rightarrow \text{Sum Block} \rightarrow \text{DFF}_{\text{out}}$$

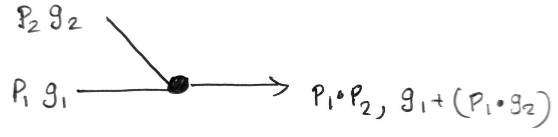
The adder consists of three main combinational stages:

1) **P/G Block:** Computes $P_i = a_i \oplus b_i$ and $G_i = a_i b_i$.
2) **CLA Block:** BK prefix network computing group $P_{i:j}$ and $G_{i:j}$.
3) **Sum Block:** Computes $S_i = P_i \oplus C_i$.

the logic followed by the BK prefix network computing group is as follows



where each black node does the following operations



$P_1 = A_1 \oplus B_1$

$G_1 = A_1 \cdot B_1$

II. Design Details and Sizing

The circuit is implemented in 180 nm technology with $V_{DD} = 1.8\,\text{V}$. All transistors use minimum sizing:

$$W_P = 4\lambda = 4 \times 0.09\,\mu\text{m} = 0.36\,\mu\text{m}$$

$$W_N = 2\lambda = 2 \times 0.09\,\mu\text{m} = 0.18\,\mu\text{m}$$

For my Ngspice simulation this proved to be able to handel the required load (20 λ /10 λ) by the minimum sized gates. But with magic extracted output it had issues post Propagation (ie the the Blackcell modules) but the logic is confirmed by Verilog simulation.

The adder uses 4 distinct modules
AND-made using static CMOS logic
BLACKCELL-made using static CMOS logic
XOR-made using CPL logic
D-FF-made using TSPC logic

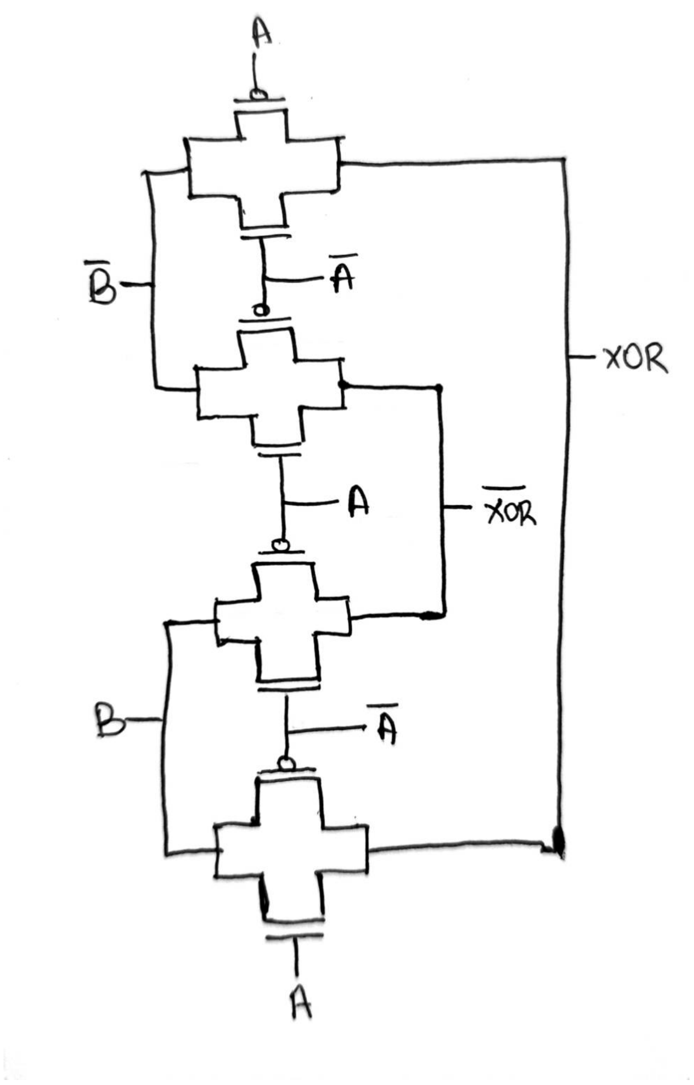


Fig. 1: XOR

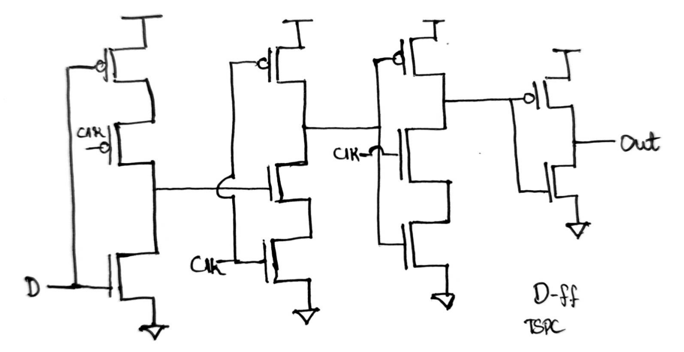


Fig. 3: D-FF

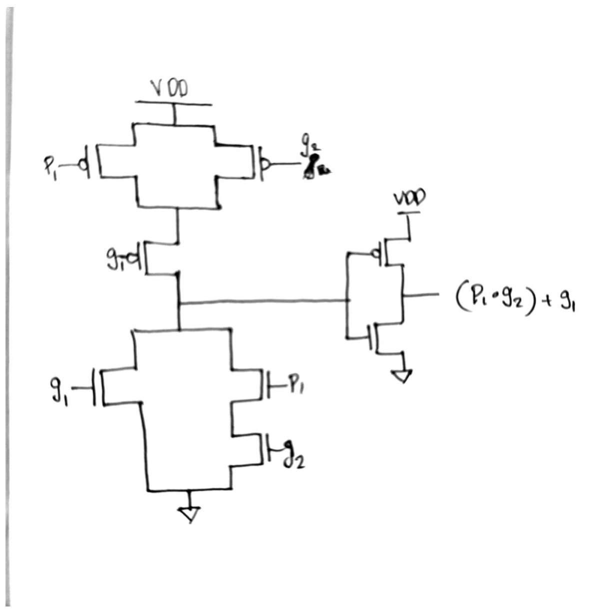


Fig. 4: G in BlackCELL

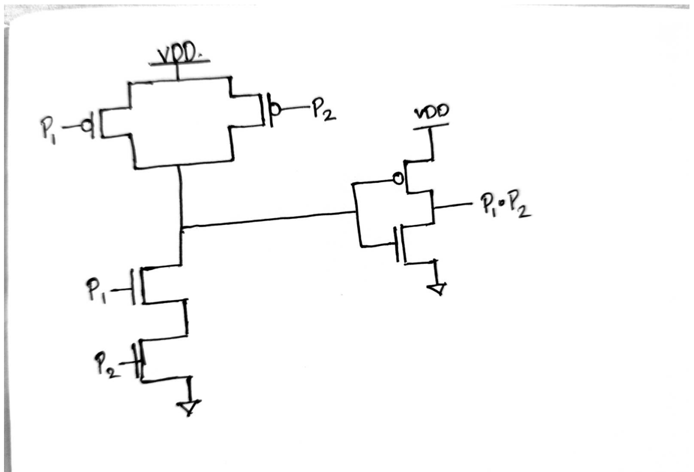


Fig. 2: AND

A. D-Flip-Flop (DFF)

The TSPC DFF is used for high-speed, single-phase clocking.

B. Adder Modules Stick Diagram

1) **Static CMOS Logic:** Used for AND, OR, Black Cell.

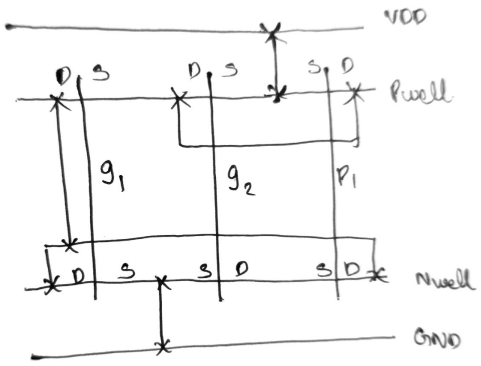


Fig. 5: G in BlackCELL

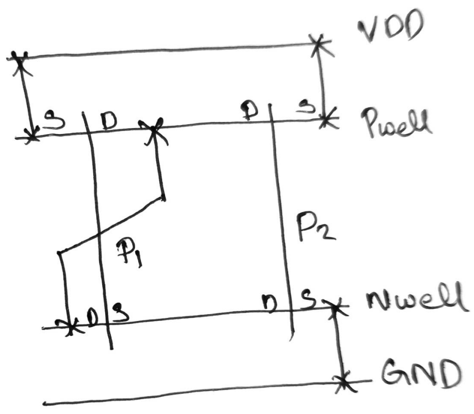


Fig. 6: P in BlackCELL and AND

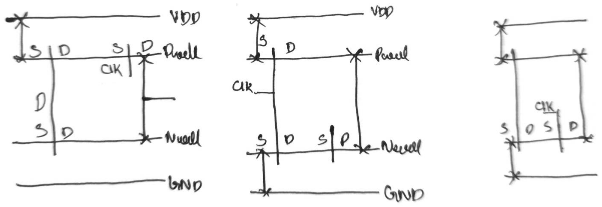


Fig. 7: TSPC D-FF

2) **CPL XOR:** Faster and smaller than static CMOS XOR.

III. NGSPICE Simulations

A. Block Functionality

Each block (AND2, BLACKCELL, XOR2, DFF) was simulated for correctness.

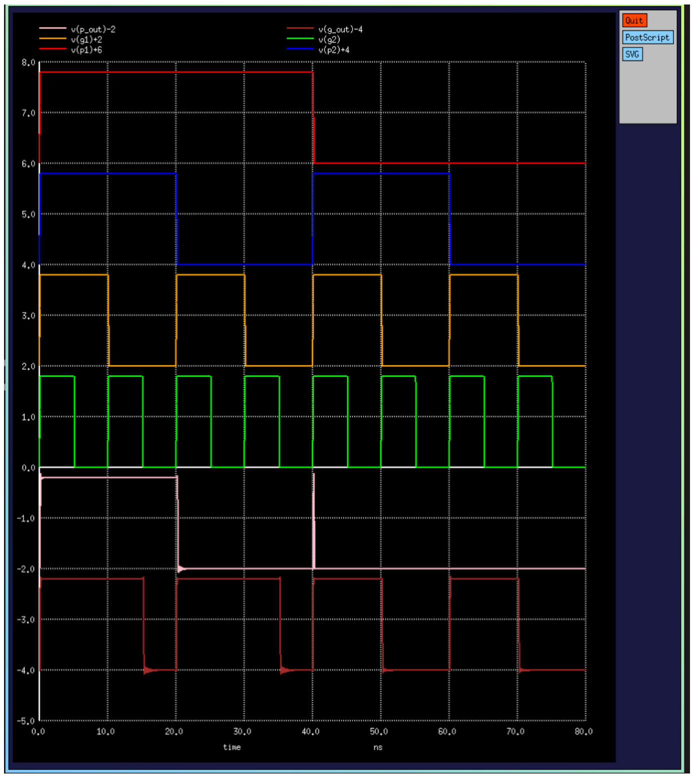


Fig. 8: G in BlackCELL

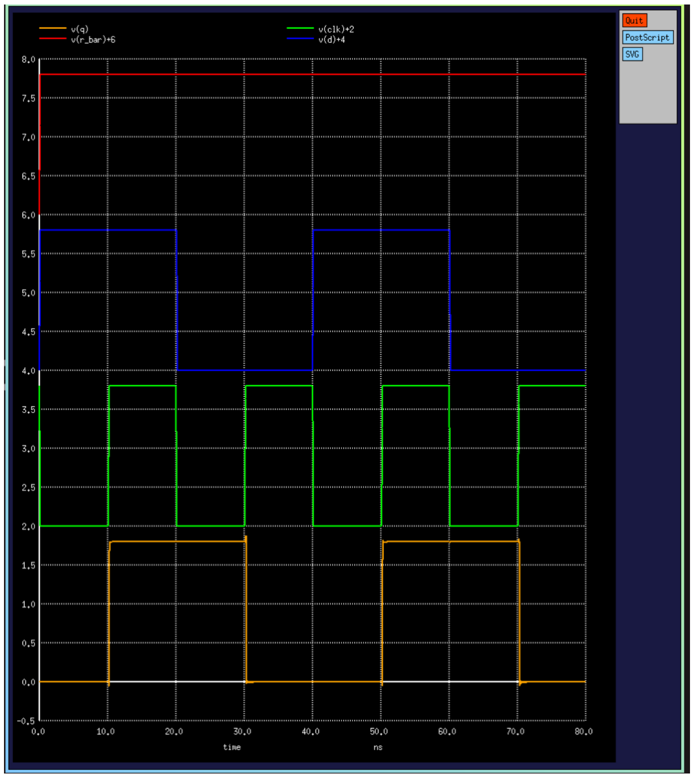


Fig. 9: D-FF

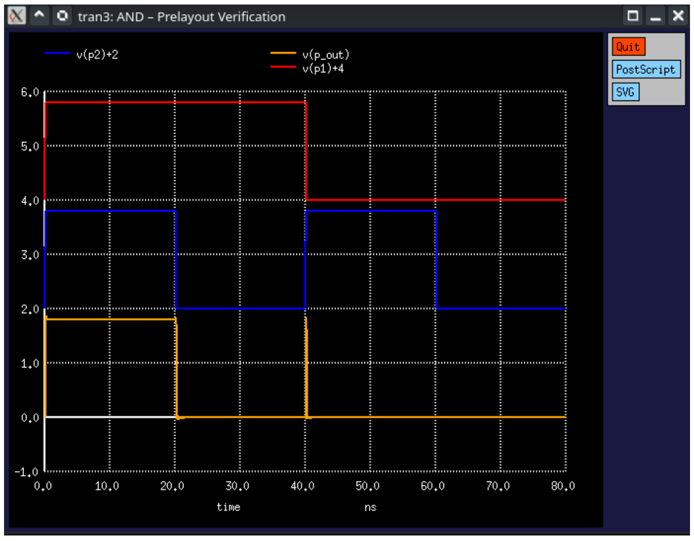


Fig. 11: AND

B. *DFF Timing (Q4)*

Table I summarizes the timing.

TABLE I: TSPC DFF Timing Characteristics (Schematic)

Parameter	Symbol	Value (ps)
Setup Time	t_{setup}	9.904805e-09
Hold Time	t_{hold}	1.019519e-08
Clock-to-Q	t_{cQ}	9.519459e-11

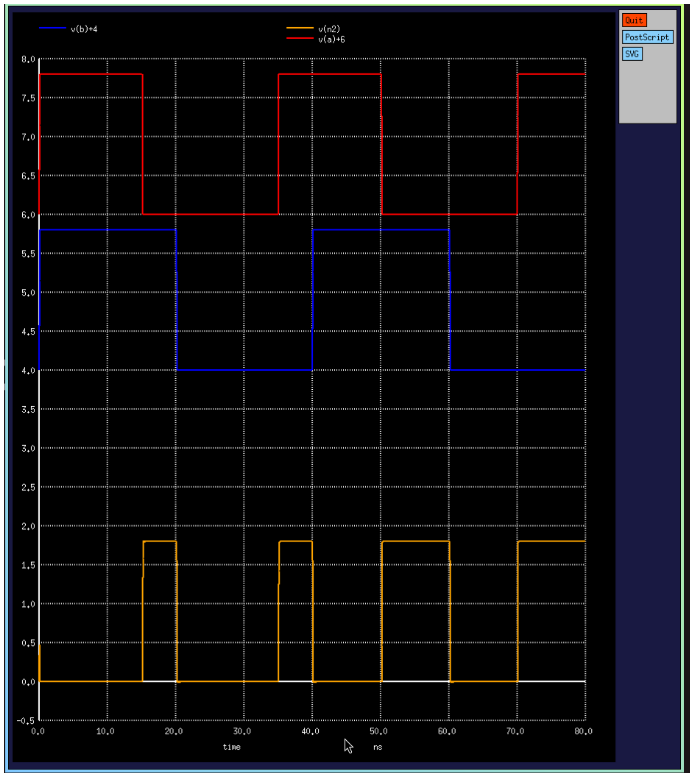


Fig. 10: XOR

C. *Combinational Delay (Q7)*

Table II shows delay values.

For the following test cases addition is analyzed and verified

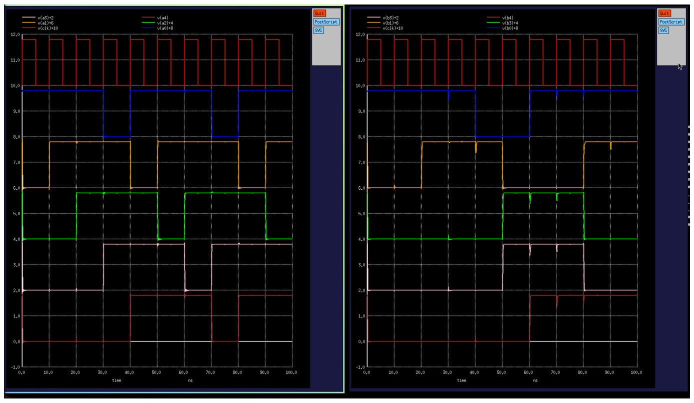


Fig. 12: INPUT

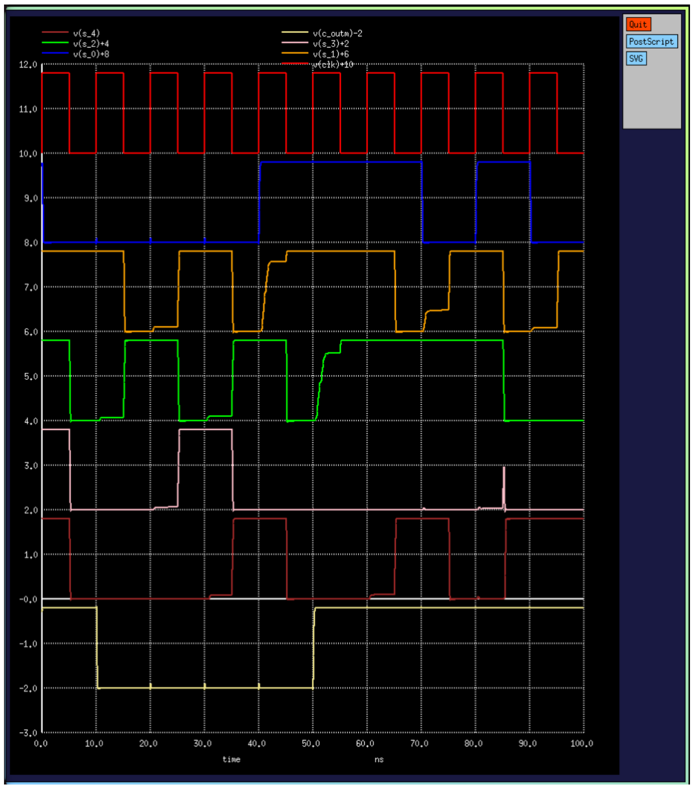


Fig. 13: OUTPUT

TABLE II: Combinational Delay Before Final DFF

Output	Delay (ns)
t_{s0}	40.12036
t_{s1}	25.13811
t_{s2}	15.13975
t_{s3}	25.14589
t_{s4}	35.14090
t_{Cout}	50.12022
Worst-Case	**50.12522**

The maximum clock speed for which a digital circuit design will operate correctly is determined by the Worst-Case Combinational Delay and the characteristics of the flip-flop (DFF), specifically its Setup Time (T_{setup}) and the Clock-to-Q Delay ($T_{\text{c-q}}$).

The general formula for the minimum clock period (T_{min}) is:

$$T_{\text{min}} = T_{\text{C-Q}} + T_{\text{Combinational\_Worst}} + T_{\text{setup}}$$

The critical path delay, or the Worst-Case Combinational Delay ($T_{\text{Combinational\_Worst}}$), is:

$$T_{\text{Combinational\_Worst}} = 50.12522 \text{ ns}$$

Substituting the numerical values (in nanoseconds):

$$T_{\text{min}} = 0.09519459 \text{ ns} + 50.12522 \text{ ns} + 9.904805 \text{ ns}$$
$$T_{\text{min}} = \mathbf{60.12521959} \text{ ns}$$

IV. Calculating the Maximum Clock Frequency (f_{MAX})

The maximum clock frequency is the inverse of the minimum clock period (T_{min}).

$$f_{\text{max}} = \frac{1}{T_{\text{min}}}$$

First, convert T_{min} to seconds: $T_{\text{min}} = 60.12521959 \times 10^{-9}$ s.

$$f_{\text{max}} = \frac{1}{60.12521959 \times 10^{-9} \text{ s}}$$

$$f_{\text{max}} \approx 16631955.89 \text{ Hz}$$

Expressed in Megahertz (MHz):

$$f_{\text{max}} \approx \mathbf{16.632} \text{ MHz}$$

The **maximum clock speed** for which my design operates correctly is approximately **16.632** MHz.

V. Floor Plan

Initial idea of layout. Here regular struchure include D-FF,XOR,AND,BlackCELL all of there have a horizontal pitch of 60λ and vertical pitch of 100 λ.

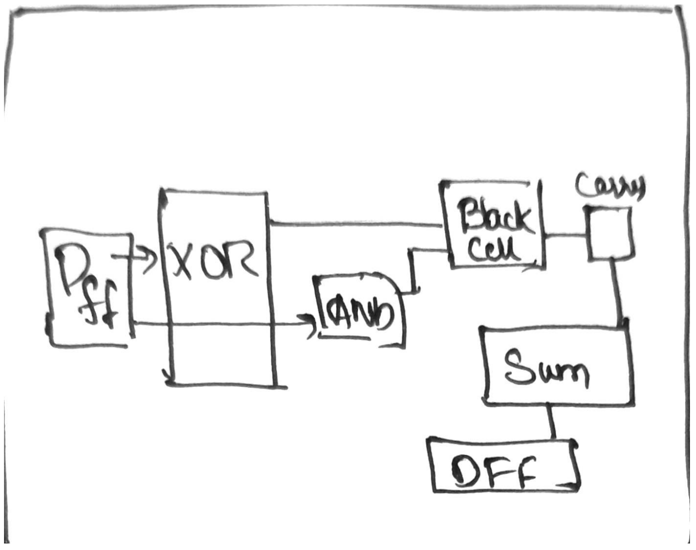


Fig. 14: initial idea

Here are the final magic layout

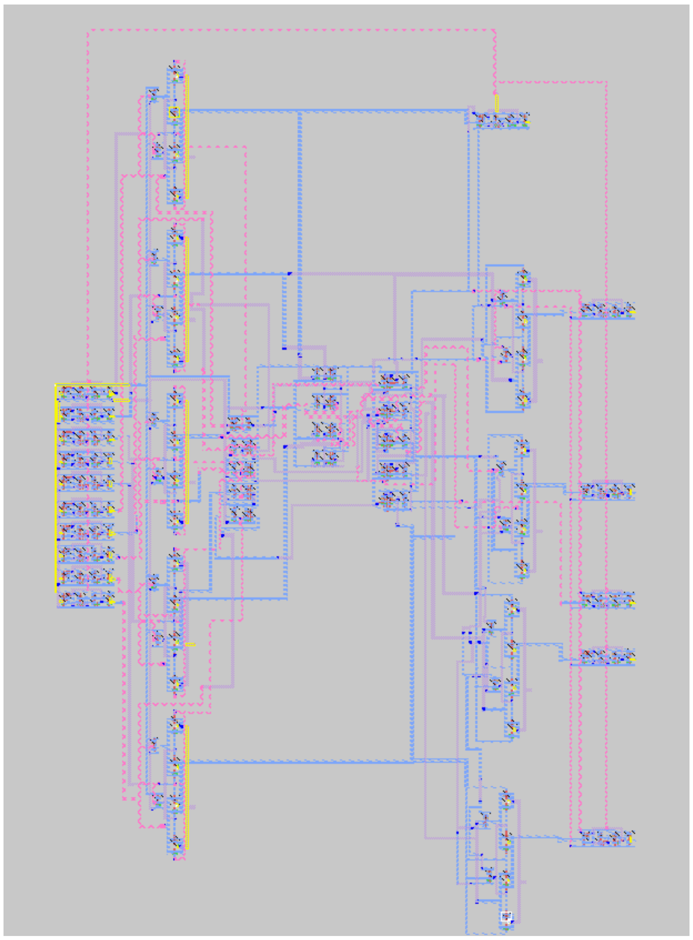


Fig. 15: magic layout

At the time of this report my complete magic layout is not working properly, the propagation modules are giving proper output, but the combined sum is not working properly, whereas each of my modules are giving proper output for the inputs received.

following are the modules

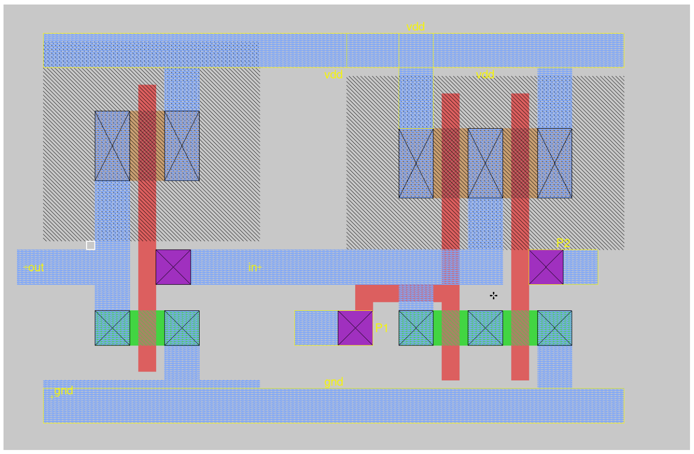


Fig. 17: AND module of black cell and general use magic layout

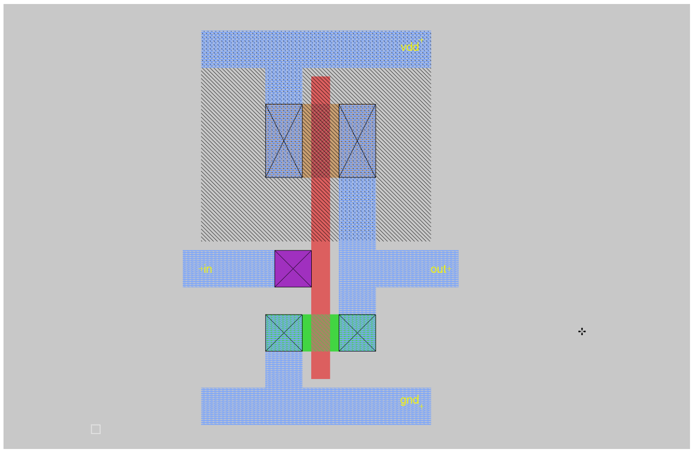


Fig. 18: Inverter magic layout

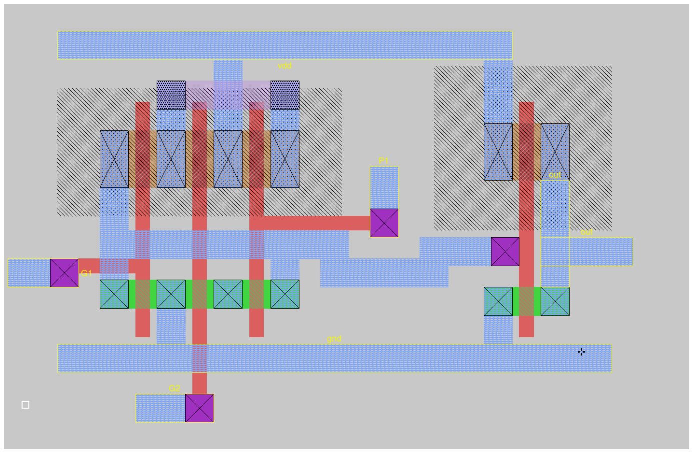


Fig. 16: Generate module of black cell magic layout

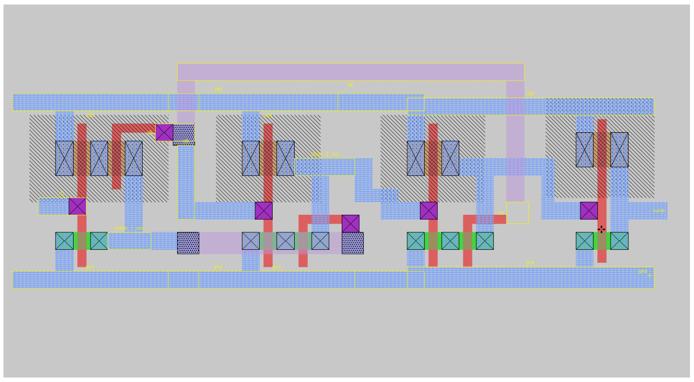


Fig. 19: TSMC D-FF magic layout

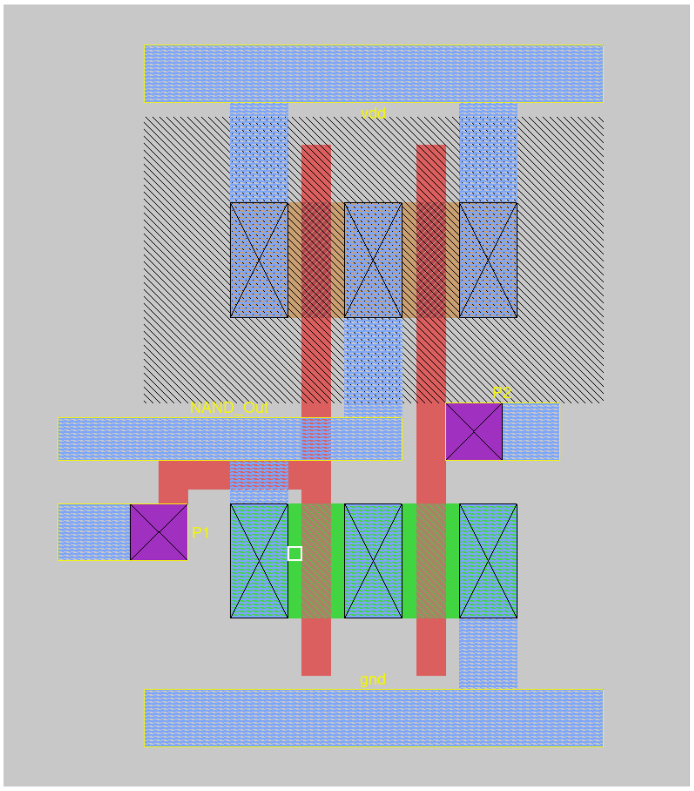
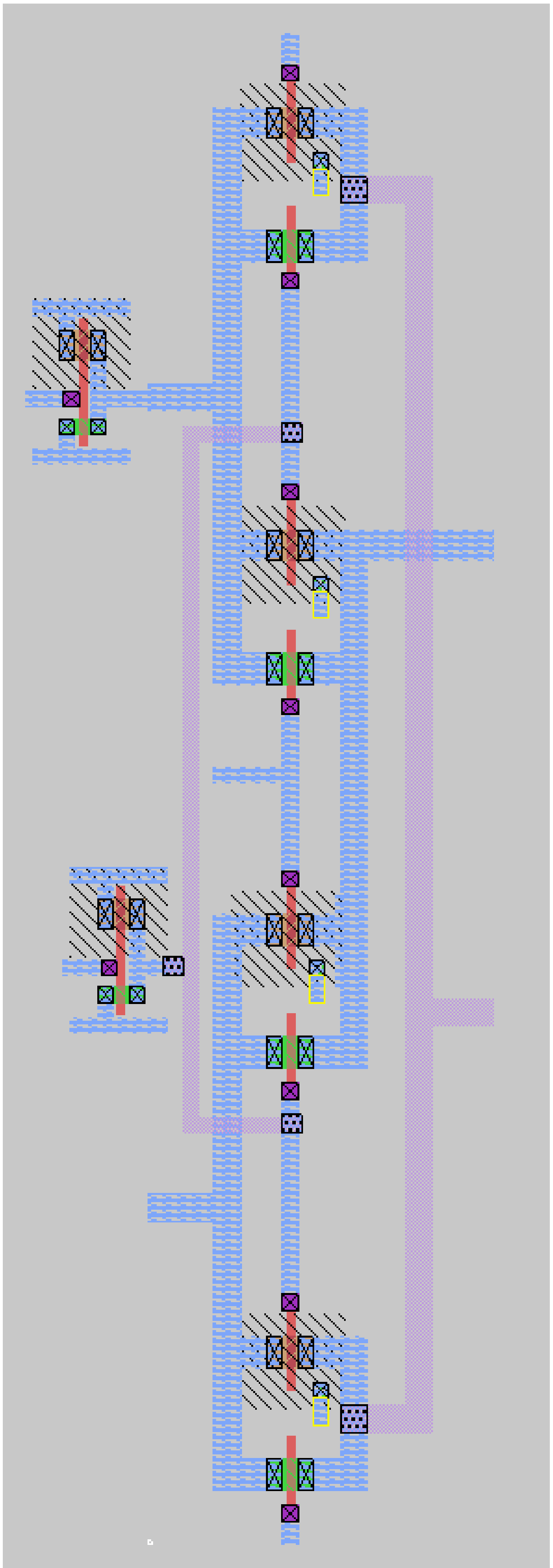


Fig. 20: NAND magic layout



VI. Verilog

Verilog code for the exact method of adding is made and verified.

A. verification

the verilog code is test againt a couston test bench and the output result and the gtkwave plots are attached

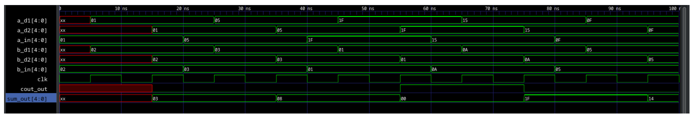
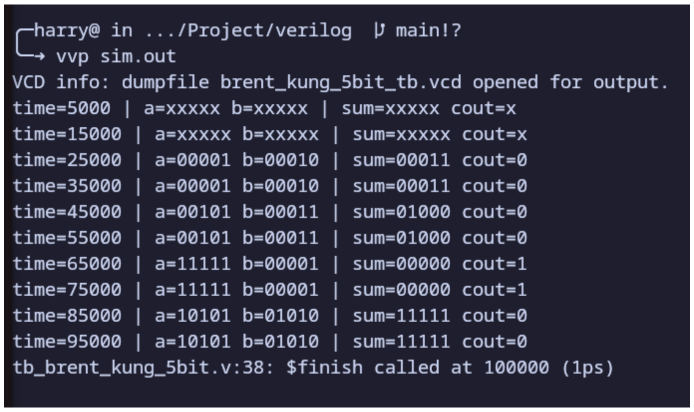


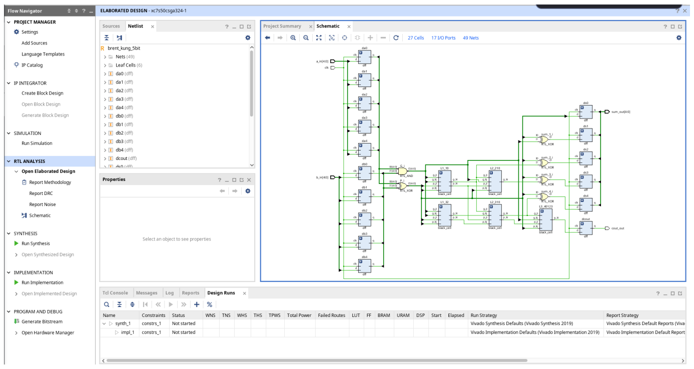
Fig. 22: GTKwave out



```
╭─harry@ in .../Project/verilog  main!?
╰─→ vvp sim.out
VCD info: dumpfile brent_kung_5bit_tb.vcd opened for output.
time=5000 | a=xxxxx b=xxxxx | sum=xxxxx cout=x
time=15000 | a=xxxxx b=xxxxx | sum=xxxxx cout=x
time=25000 | a=00001 b=00010 | sum=00011 cout=0
time=35000 | a=00001 b=00010 | sum=00011 cout=0
time=45000 | a=00101 b=00011 | sum=01000 cout=0
time=55000 | a=00101 b=00011 | sum=01000 cout=0
time=65000 | a=11111 b=00001 | sum=00000 cout=1
time=75000 | a=11111 b=00001 | sum=00000 cout=1
time=85000 | a=10101 b=01010 | sum=11111 cout=0
time=95000 | a=10101 b=01010 | sum=11111 cout=0
tb_brent_kung_5bit.v:38: $finish called at 100000 (1ps)
```

Fig. 23: terminal out

Vivado generated schematics is attached



While trying to impliment the oscilloscope out i was facing some diffuculty but i was able to generate bit stream and run the the verilog on the FPGA and test it using the mapped switches and LEDs.

VII. Conclusion

A 5-bit pipelined Brent-Kung CLA adder was successfully designed and verified. The critical path delay is $\mathbf{50.12522}$ ns.

References

[1] A. Sharma, M. I. Hasan, and D. Tiwari, ”FPGA Implementation of Floating-Point Multiplier Employing Carry-Lookahead Adders,” in *2025 3rd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, 2025.

[2] A. Alghamdi and F. Gebali, ”Performance Analysis of 64-bit Carry Lookahead Adders Using Conventional and Hierarchical Structure Styles,” in *2015 International Conference on Advances in Computing, Communications and Control (ICAC3)*, 2015, pp. 80–83.

[3] P. Annapurna Bai and B. Thirupathaiah, ”Design of Efficient Han-Carlson Adder,” *International Journal of VLSI System Design and Communication Systems*, vol. 4, no. 12, pp. 1439–1443, Nov. 2016.

[4] R. P. P. Singh, P. Kumar, and B. Singh, ”Performance Analysis Of Fast Adders Using VHDL,” in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, 2009, pp. 189–193.

[5] R. Barik and S. Mangaraj, ”Design and Implementation of 64-bit Carry Lookahead Adders Using Fixed and Variable Stage Structure,” in *2018 4th International Conference on Devices, Circuits and Systems (ICDCS)*, 2018, pp. 143–147.

[6] T. Han and D. A. Carlson, ”Fast Area-Efficient VLSI Adders,” in *Proceedings 8th Symposium on Computer Arithmetic*, May 1987, pp. 49–56.

[7] M. R. Ravula, A. Potharaju, and R. P. Vidyadhar, ”Designing Carry Look Ahead Adder to Enrich Performance using One Bit Hybrid Full Adder,” in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, 2022, pp. 86–89.

[8] R. P. Brent and H. T. Kung, ”A Regular Layout for Parallel Adders,” *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, Mar. 1982.

[9] V. Sidharthan, ”Comparative Analysis of Ladner-Fischer Adder and Han-Carlson Adder Parallel-Prefix Adder for Their Area, Delay and Power Consumption,” *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 4, no. 1, pp. 920–923, Jan.–Feb. 2018.

[10] Abhilash and S. Hussain, ”Comparative Analysis of Parallel Prefix Adders,” in *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, 2024.

[11] M. Bharathi et al., ”A Comparative Analysis of 8-Bit Parallel Prefix Adder Architectures,” in *2024 5th International Conference on Smart Electronics and Communication (ICOSEC)*, 2024.

[12] J. Samanta, M. Halder, and B. P. De, ”Performance Analysis of High Speed Low Power Carry Look-Ahead Adder Using Different Logic Styles,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 6, pp. 330–336, Jan. 2013.

[13] S. Yamtim and S. Tooprakai, ”Efficient Processing Time with Carry-lookahead Adder Memristor Rationed Logic,” in *2023 9th International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, 2023.