Design for SIMD MAC Unit Using 2-Bit Adders and Multipliers

[1]Sagar D, [2]Dr Kiran V

[1]PG Student , [2]Associate Professor
[1]Dept.of Electronics and Communication Engineering
[1]RV College Of Engineering Bengaluru, Karnataka, India

***Abstract*: The advent of digital circuits made it easy to realize many mathematical operations using the binary Boolean functions. The only problem was that, the mathematical operations were able to run at a significantly high speed with great accuracy for unsigned or signed integer numbers. Various architectures were able to accelerate the performance of mathematical operations on integers. But most of the real-world problems required operation with real numbers and hence either fixed point or more importantly floating-point arithmetic was necessary. It is possible to run different algorithms to execute operations on floating point numbers in an architecture designed for integer numbers. But the amount of CPU cycles required increases substantially with the complexity of problems involved, the precision required and accuracy as well.**

It is possible to develop architectures for fixed-point real numbers. But the reusability of the architecture is quite difficult for higher precision. For high accuracy or precision, the architecture hardware utilization increases exponentially. The conversion of fixed point to floating-point increases hardware as well. Hence, we have the IEEE 754 format for single/double precision floating-point architecture. But the amount of power spent and the hardware requirement is equally high. The time required to produce a stable output is quite high and the floating-point computation is the basic requirement in the overall algorithm of neural networks.

***keywords*: Mathematical operations, ALU, CPU,Vedic algorithm, Adders, Multipliers**

I. INTRODUCTION

The real-world signals we encounter are analog in nature. The primary communication with the external world is using analog signals. The use of digital signals arises due to various advantages among them being noise immunity and fast computation architectures and easy development of architectures. The architectures for digital computers or for any digital computation is for integer computation. The conventional procedure to compute floating-point arithmetic involved using the integer-based architecture with software support (algorithms). But the algorithms consumed large CPU cycles.

This reduced the throughput of the CPU. To increase the throughput, efficient algorithms have been developed, but the CPU cycle consumption reduction is quite feeble. The solution has always been to develop a complete floating-point unit separately operating on data or to develop floating-point architecture CPUs. The latter complexity is tremendous and hence, the former is preferred. The CPU off-loads floating-point arithmetic to this complete module. Hence, as a designer it is quite challenging to design a dedicated module for floating-point arithmetic which is highly efficient, low power, low area and is as fast as the ALU. The constraint of usage in real-time applications and further those involving human life require high precision facility for proper training. Although, this IEEE754 may be favorable for the case of training, the time and power consumed is significantly high. The computation of the same in a real-time application may not converge required results. Hence, the easier approach to achieve speed and power efficiency without much reduction in accuracy of the model is to use fixed-point arithmetic architectures. This reduces the hardware significantly and more importantly it's possible to obtain results for every clock cycle. This makes usage of this system architecture in real-time applications and possible optimization towards the application.

The IEEE 754 format for single/double precision floating-arithmetic facilitate an easy architecture based on the algorithm. The algorithm is a multi-cycle-based algorithm. The design takes multiple cycles to evaluate the output but consumes significantly lesser cycles compared to an integer-based architecture with software support. Today many processors and SoC use a floating-point compute unit separately. The architecture of this unit may involve multiple architectures supporting both fixed-point and IEEE standard floating-point as well. It has become necessary to use the design as a separate unit, so that it could be used by different modules or designs. In a modern SoC, there are different compute designs apart from the processor like DSP, Video encoder/decoder, security designs etc. Hence floating-point unit should be able to accommodate the necessary architectures to facilitate the usage by these other compute designs. It's possible to have both the IEEE 754 architecture and fixed-point compute unit together on a chip and software support can utilize both to achieve required application. But the fixed-point architecture reusability is extremely low. With the versatility of applications arising every day, it is required to have a new core logic which only processes for a AI application. These new lines of processors are called 'Neural processors. Their architecture consists of simple CPU core but the chip is majorly filled with compute units which accelerate the computations of the AI applications.

The major research happens to be in the amount of versatility that could be brought in these acceleration units. It is possible to modify the IEEE standard to user requirements and architecture. But this reduces the reusability of the design. This flexibility to modify the IEEE standard is helpful in easy design of architectures for DSP and other compute designs whereas the trade-off is the reusability and technical aspects like precision and accuracy and overall features of final product being developed.

II. METHODOLOGY

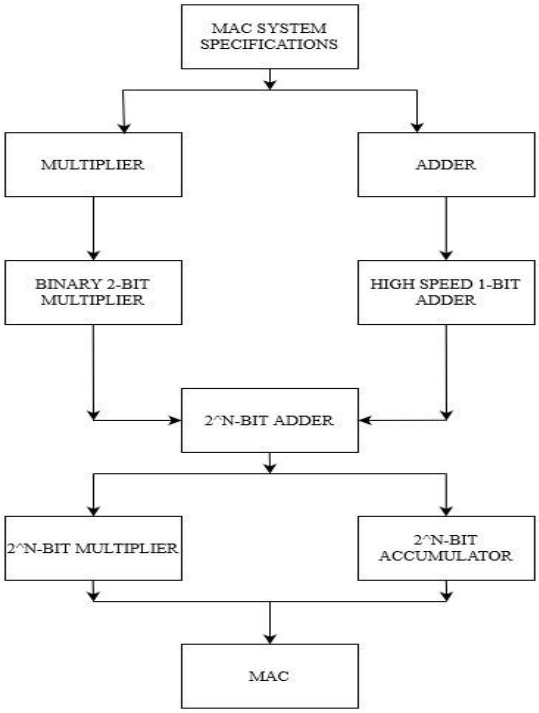


Fig 1: Flowchart for MAC design

The SIMD MAC architecture requires a basic MAC unit. The high-speed multiplier is designed based on Vedic algorithms, which 2-bit binary multipliers and higher order adders. The high-speed adder is designed using minimal transistors. This is necessary as the same adder is used in multiplier design. The 2-bit binary multiplier is designed using and gates and half-adders. The 2-bit binary multiplier is used in the design of a 4-bit vedic multiplier. Higher order multipliers are designed using the high-speed higher order adders. The basic cells required are inverter, AND gate, XOR gate. These basic cells are optimized for minimum delay, minimum area and equal rise and fall time. The half adder, 1-bit full adder are constructed using these basic cells. The 2-bit binary multiplier and higher order adders using 'RCA' architecture are constructed. The 2-bit binary multiplier and higher order adders are used to design higher order multiplier using vedic algorithm.

III. SCHEMATIC OF TG ADDER

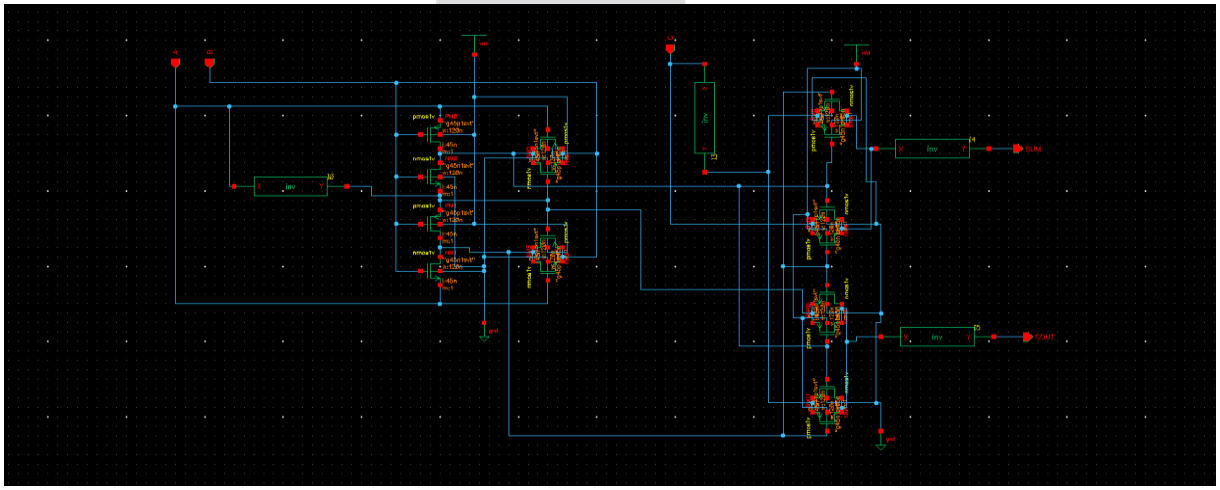


Fig 2: TG adder schematic

The Fig 2, represents the transmission gate 1-bit adder schematic. It uses a total of 24 transistors. The TG adder is sized for minimum delay. This minimum delay is obtained by converging the function for equal rise and fall time.

$$SUM = \overline{C}(A \oplus B) + C(\overline{A \oplus B})$$
$$CARYY = (\overline{A \oplus B})B + (A \oplus B)C$$

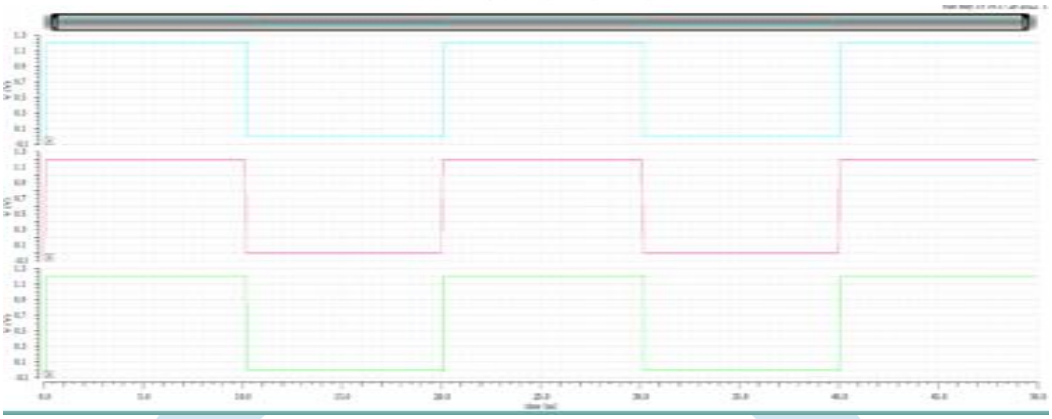


Fig 3: TG adder transient response

The Fig 3, represents the transient response of the TG adder. The delay for input to sum is **56.73ps**, the input to carry delay is **51.05ps**.

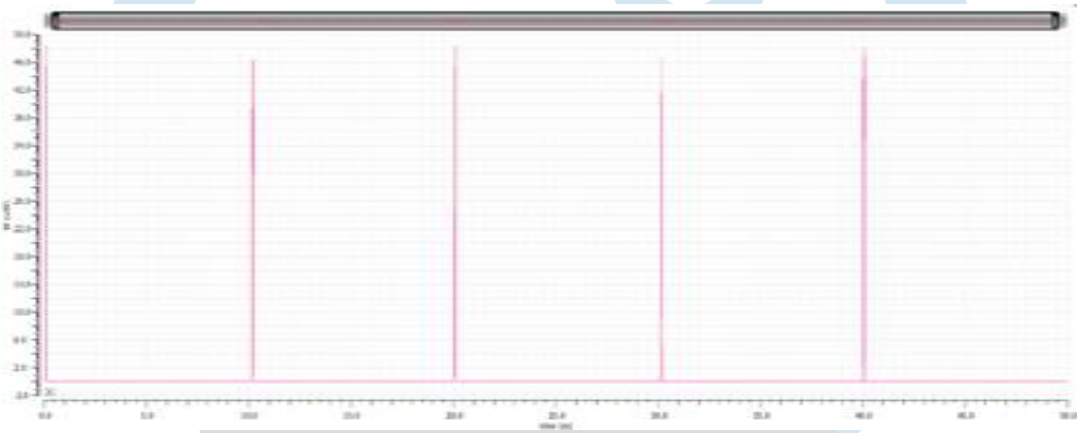


Fig 4: TG adder power consumption

The Fig 4, represents the power consumption of the TG adder. The total power consumption is **207.4nW**.

IV. SCHEMATIC OF CPL ADDER

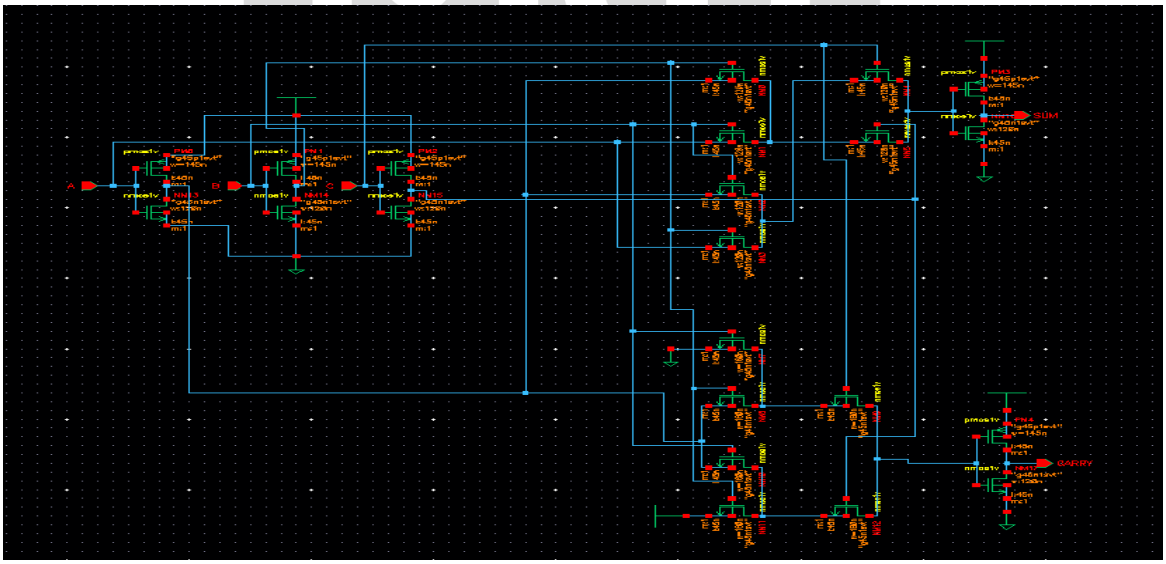


Fig 5: CPL adder schematic

The Fig 5, represents the Complimentary Pass Transistor (CPL) 1-bit adder schematic. It uses a total of 22 transistors. The CPL adder is sized for minimum delay. This minimum delay is obtained by converging the function for equal rise and fall time.

$$SUM = \overline{C}(A \oplus B) + C(\overline{A \oplus B})$$
$$CARYY = (\overline{A \oplus B})B + (A \oplus B)C$$

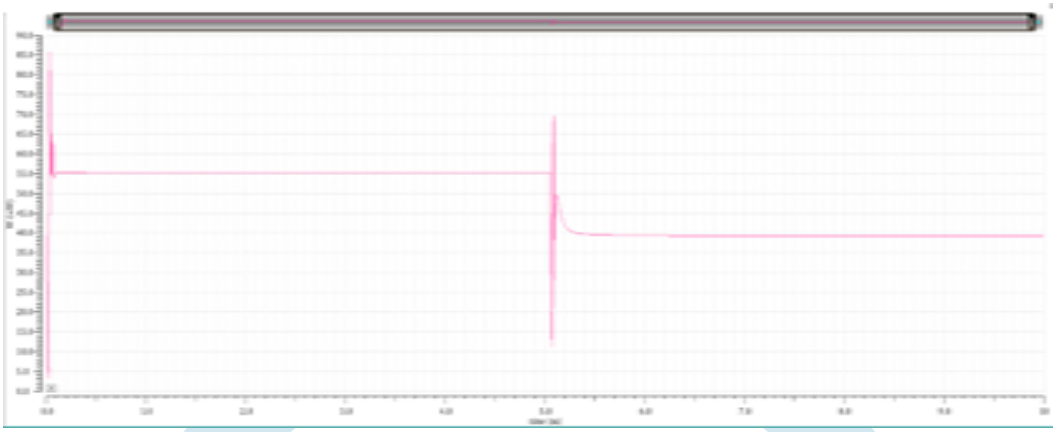


Fig 6: CPL adder power consumption

$$P_{total} = P_{dynamic} + P_{static}$$

The Fig 6, represents the power consumption of the CPL adder. The total power consumption is **47.42µW**.

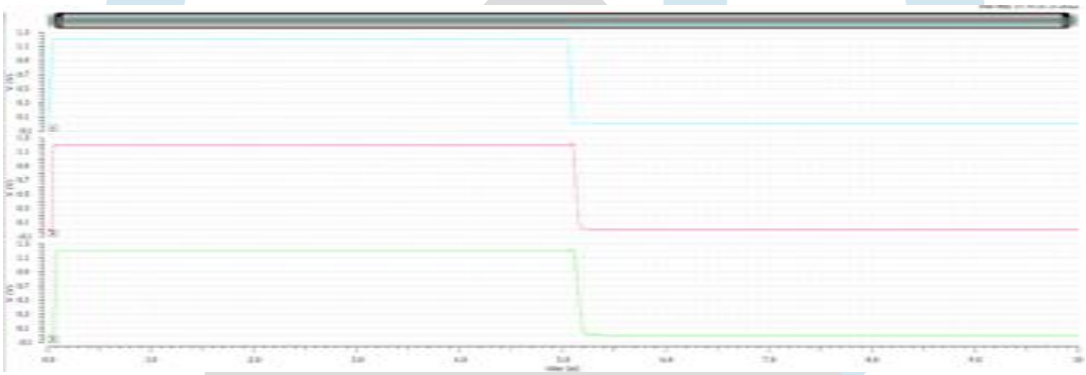
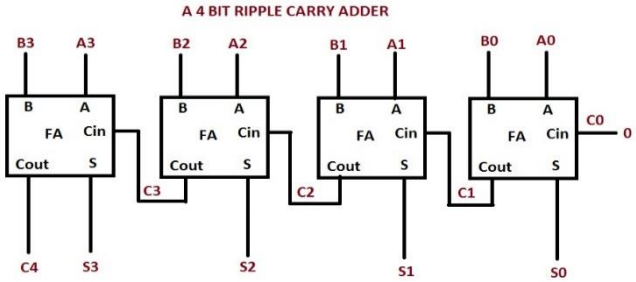


Fig 7: CPL adder transient response

The Fig 7, represents the transient response of the CPL adder. The delay from the input to the sum is **78ps** while the delay from the input to the carry is **57ps**.

V. SCHEMATIC OF 4 BIT RIPPLE CARRY ADDER



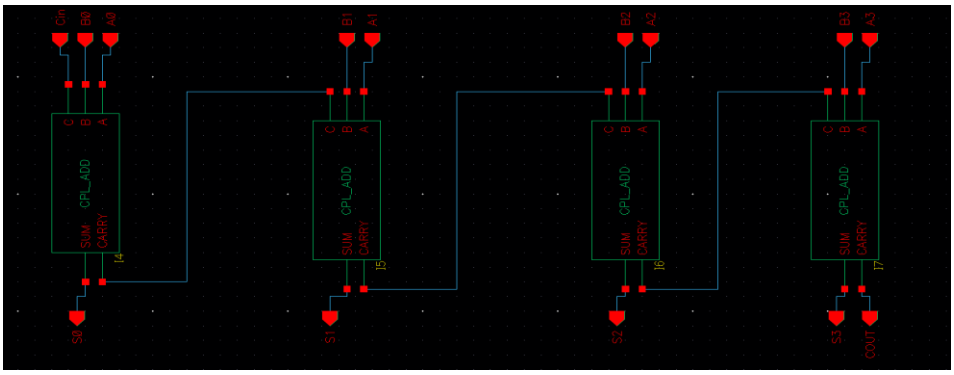


Fig 8: 4-bit ripple carry adder

The Fig 8, represents the schematic of 4-bit ripple carry adder either using CPL adder or TG adder. The adder is sized for minimum delay. This minimum delay is obtained by converging the function for equal rise and fall time.

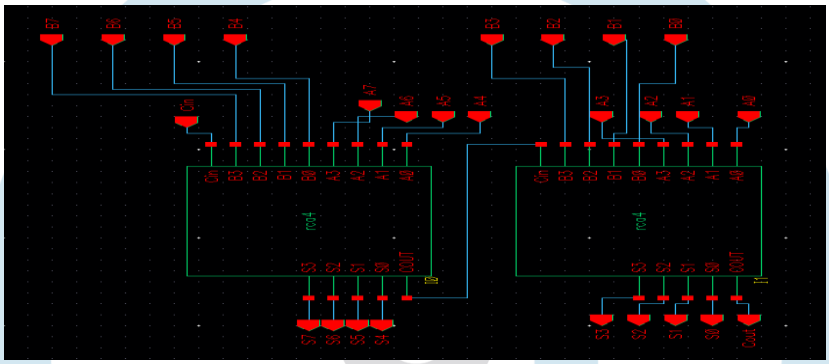


Fig 9: 8-bit Ripple Carry Adder (RCA) schematic

The Fig 9, represents the schematic of 8-bit ripple carry adder using 4-bit ripple carry adder. The adder is sized for minimum delay. This minimum delay is obtained by converging the function for equal rise and fall time.

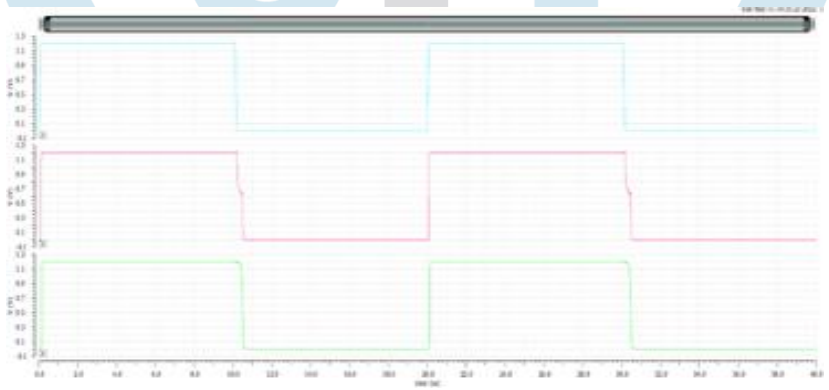


Fig 10: 8-bit RCA transient response

The Fig 10, represents the transient response of the 8-bit RCA. The delay from input to the sum is **337.6ps**, while the delay from the input to the carry is **323.4ps**.

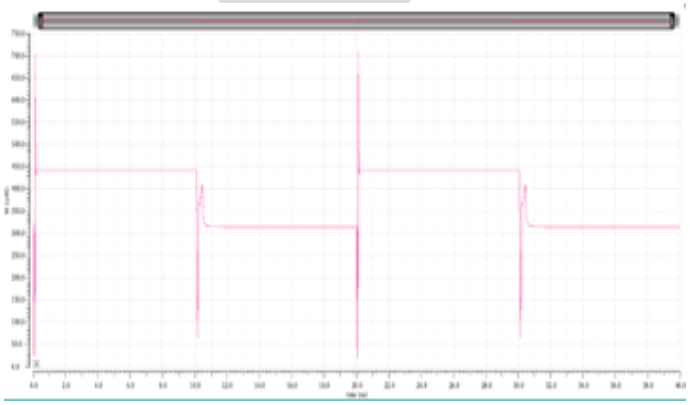


Fig 11: 8-bit RCA power consumption

Fig 11, represents the power consumption of the 8-bit RCA. The total power consumption is **379µW**.

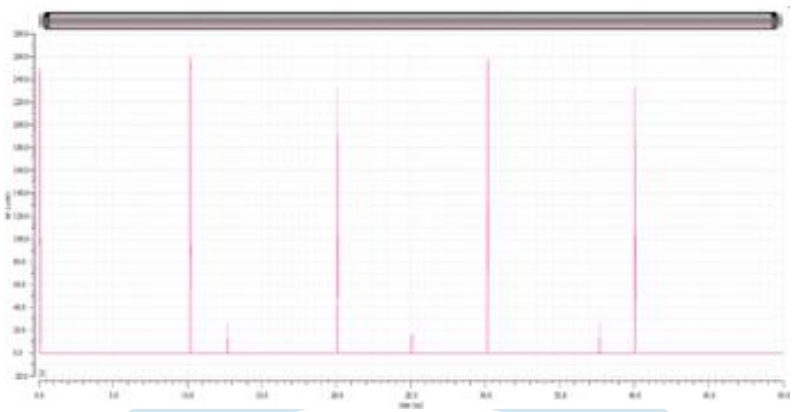


Fig 15: 2-bit binary multiplier power consumption

$$P_0=A_0+B_0$$
$$P_1=Sum(A_0B_1+A_1B_0)$$
$$P_2=Sum(A_1B_1+C_1)$$
$$P_3=C_3$$

The Fig 15, represents the power consumption of the 2-bit binary multiplier. The total power consumption is **1.1µW**.

The analysis of the designed basic cells for SIMD MAC unit provides required speed for high throughput of the system and maintains a decent power budget. The overall delay computed from the critical path is about **1ns** which is less than **3.6ns** in [1]. Hence, full custom design with much smaller nodes can provide improved performance with better power performance as well.

I. RESULTS

The analysis of the designed basic cells for SIMD MAC unit provides required speed for high throughput of the system and maintains a decent power budget. The overall delay computed from the critical path is about 1ns which is less than 3.6ns. Hence, full custom design with much smaller nodes can provide improved performance with better power performance as well.

Table 1: Comparison of Power & Delay

INSTANCE	DELAY(ps)	POWER(nW)
Inverter	Tr = 13.04 Tf = 13.12	16.2
AND gate	27.76	84.42
XOR gate	30.14	77
TG Adder	Tsum = 56.73 Tcarry = 51.05	207.4
CPL Adder	Tsum = 78 Tcarry = 57	0.474
Ripple Carry Adder 8-bit	Tsum = 337.6 Tcarry = 323.4	0.379
2-bit binary multiplier	68.05	0.11

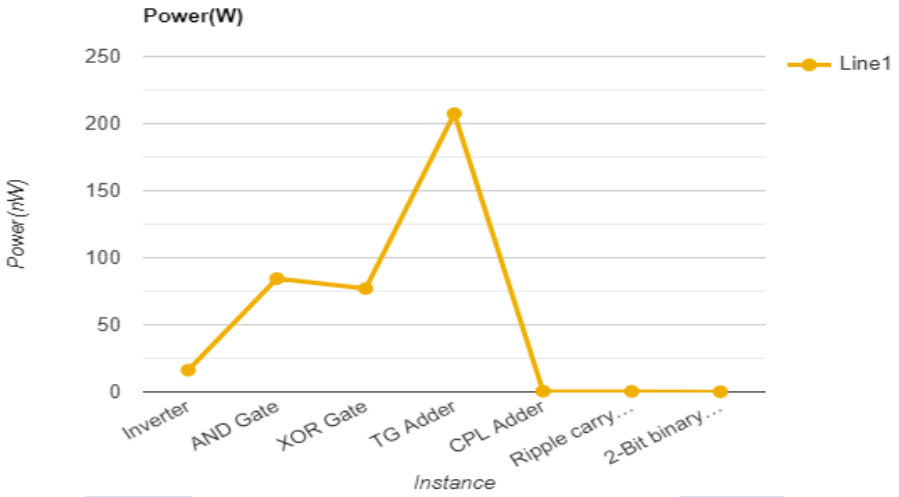


Fig 16: Graph between Power vs Instance

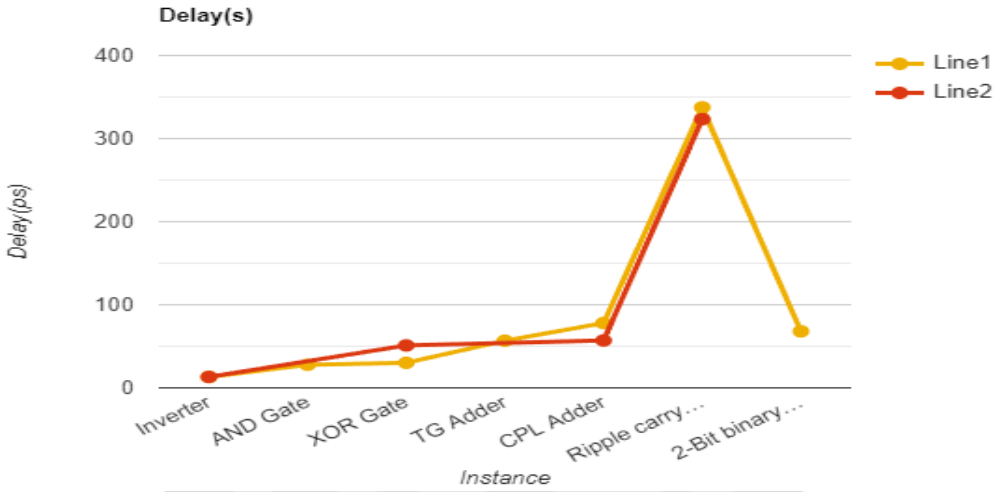


Fig 17: Graph between Delay vs Instance

REFERENCES

1. S. Lee, D. Kim, D. Nguyen and J. Lee, "Double MAC on a DSP: Boosting the Performance of Convolutional Neural Networks on FPGAs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 5, pp. 888-897, May 2019, doi: 10.1109/TCAD.2018.2824280
2. K. Khalil, B. Dey, A. Kumar and M. Bayoumi, "A Reversible-Logic based Architecture for Convolutional Neural Network (CNN)," 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021, pp. 1070-1073, doi: 10.1109/MWSCAS47672.2021.9531842
3. Y. Yu, C. Wu, T. Zhao, K. Wang and L. He, "OPU: An FPGA-Based Overlay Processor for Convolutional Neural Networks," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 28, no. 1, pp. 35-47, Jan. 2020,doi:10.1109/TVLSI.2019.2939726
4. M. Kang, S. Lim, S. Gonugondla and N. R. Shanbhag, "An In-Memory VLSI Architecture for Convolutional Neural Networks," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 3, pp. 494-505, Sept. 018, doi: 10.1109/JETCAS.2018.2829522
5. K. A. Shahan and J. Sheeba Rani, "FPGA based convolution and memory architecture for Convolutional Neural Network," 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID), 2020, pp. 183-188, doi: 10.1109/VLSID49098.2020.00049.
6. Y. Chou, J. -W. Hsu, Y. -W. Chang and T. -C. Chen, "VLSI Structure-aware Placement for Convolutional Neural Network Accelerator Units," 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 1117-1122, doi: 10.1109/DAC18074.2021.9586294
7. G. Shu, W. Liu, X. Zheng and J. Li, "IF-CNN: Image-Aware Inference Framework for CNN With the Collaboration of Mobile Devices and Cloud," in IEEE Access, vol. 6, pp. 68621-68633, 2018, doi: 10.1109/ACCESS.2018.2880196.

8. F. U. D. Farrukh, T. Xie, C. Zhang and Z. Wang, "A Solution to Optimize Multi-Operand Adders in CNN Architecture on FPGA," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1-4, doi: 10.1109/ISCAS.2019.8702777.
9. M. Palaria, S. Sanjeet, B. D. Sahoo and M. Fujita, "Adder-Only Convolutional Neural Network with Binary Input Image," 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS), 2019, pp. 319-322, doi: 10.1109/MWSCAS.2019.8885354.
10. D. Manatunga, H. Kim and S. Mukhopadhyay, "SP-CNN: A Scalable and Programmable CNN-Based Accelerator," in IEEE Micro, vol. 35, no. 5, pp. 42-50, Sept.-Oct. 2015, doi: 10.1109/MM.2015.121.
11. Y. -X. Kuo and Y. -K. Lai, "An Efficient Accelerator for Deep Convolutional Neural Networks," 2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), 2020, pp. 1-2, doi: 10.1109/ICCE-Taiwan49838.2020.9258103.
12. T. Zebin, P. J. Scully, N. Peek, A. J. Casson and K. B. Ozanyan, "Design and Implementation of a Convolutional Neural Network on an Edge Computing Smartphone for Human Activity Recognition," in IEEE Access, vol. 7, pp. 133509-133520, 2019, doi: 10.1109/ACCESS.2019.2941836.
13. B. Misganaw and M. Vidyasagar, "Exploiting Ordinal Class Structure in Multiclass Classification: Application to Ovarian Cancer," in IEEE Life Sciences Letters, vol. 1, no. 1, pp. 15-18, June 2015, doi: 10.1109/LLS.2015.2451291.
14. Z. Nagy and P. Szolgay, "Configurable multi-layer CNN-UM emulator on FPGA using distributed arithmetic," 9th International Conference on Electronics, Circuits and Systems, 2002, pp. 1251-1254 vol.3, doi: 10.1109/ICECS.2002
15. C. Guo, L. Zhang, X. Zhou, W. Qian and C. Zhuo, "A Reconfigurable Approximate Multiplier for Quantized CNN Applications," 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), 2020, pp. 235-240, doi: 10.1109/ASP-DAC47756.2020.9045176.
16. F. U. D. Farrukh et al., "Power Efficient Tiny Yolo CNN Using Reduced Hardware Resources Based on Booth Multiplier and WALLACE Tree Adders," in IEEE Open Journal of Circuits and Systems, vol. 1, pp. 76-87, 2020, doi: 10.1109/OJCAS.2020.3007334.