Author Accepted Manuscript

Title: Performance Evaluation of Digital Multipliers for Application in Neural Network Accelerators

Authors: Miloš Marjanović, Marko Andjelković, Danijel Danković, Jelena Nikolić, Fabian Vargas

Published in: *Proceedings of the IEEE International Conference on Microelectronics* (*MIEL* 2025), 2025

DOI: 10.1109/MIEL66332.2025.11260925

Please cite the final published version:
Miloš Marjanović, Marko Andjelković, Danijel Danković, Jelena Nikolić, Fabian Vargas, "Performance Evaluation of Digital Multipliers for Application in Neural Network Accelerators", *Proc. 2025 IEEE 34th International Conference on Microelectronics* (*MIEL* 2025), Niš, Serbia, October 2025, pp. 389-392, DOI: 10.1109/MIEL66332.2025.11260925

This work has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101160293 (AIDA4Edge).

Performance Evaluation of Digital Multipliers for Application in Neural Network Accelerators

Miloš Marjanović
Department of Microelectronics
Faculty of Electronic Engineering,
University of Niš
Niš, Serbia
milos.marjanovic@elfak.ni.ac.rs
ORCID: 0000-0002-2230-2646

Marko Andjelković
System Architectures Department
IHP - Leibniz Institute for High
Performance Microelectronics
Frankfurt (Oder), Germany
andjelkovic@ihp-microelectronics.com
ORCID: 0000-0001-6419-2062

Danijel Danković
Department of Microelectronics
Faculty of Electronic Engineering,
University of Niš
Niš, Serbia
danijel.dankovic@elfak.ni.ac.rs
ORCID: 0000-0002-0214-2606

Jelena Nikolić
Department of Telecommunications
Faculty of Electronic Engineering,
University of Niš
Niš, Serbia
jelena.nikolic@elfak.ni.ac.rs
ORCID: 0000-0002-3552-7211

Fabian Vargas
System Architectures Department
IHP - Leibniz Institute for High
Performance Microelectronics
Frankfurt (Oder), Germany
vargas@ihp-microelectronics.com
ORCID: 0000-0002-3871-6464

***Abstract*—This paper presents a performance analysis of three digital multiplier architectures—array, Wallace without carry-look-ahead (CLA), and Wallace with CLA logic—targeted for neural network (NN) accelerators. Simulations were performed using an IHP's 130 nm CMOS standard cell library in Cadence Spectre, focusing on propagation delay and power consumption across different supply voltages, temperatures and transistor dimensions. The Power Delay Product (PDP) was used to evaluate overall efficiency. Results show that the Wallace multiplier without CLA logic and the smallest cells offers the best trade-off in terms of delay and power, making it a strong candidate for low-power, high-performance applications.**

Keywords—array multiplier, Wallace multiplier, NN accelerators

I. Introduction

Machine Learning (ML) algorithms extract explicit knowledge structures from data through models. A neural network (NN) is a computational architecture inspired by the structure and functioning of the human brain, commonly employed in ML for processing and analyzing information. Within the framework of supervised learning, two fundamental categories are commonly distinguished: regression and classification. In this context, each training instance consists of an input paired with a corresponding output. Regression tasks involve predicting continuous outcomes, whereas classification tasks involve assigning inputs to one of a predefined set of discrete categories [1]. Starting from basic methods such as linear regression and extending to more advanced methods like deep neural networks, computations often involve numerous Multiply-Accumulate (MAC) operations. These operations are computationally expensive, especially for complex Artificial Intelligence (AI) systems, making specialized NN hardware accelerators, such as systolic arrays, essential for improving performance [2, 3]. The need to better understand and examine the power consumption of MAC units motivates our research, which could offer valuable insights into one of the key performance aspects in the context of deep neural networks. Systolic arrays consist of an array of processing elements (PEs), which must be carefully orchestrated to manage the flow of data between them, ensuring that they collaboratively process the input data before returning the result to memory. Fig. 1 illustrates the block diagram of a systolic array and its processing element. Each PE typically comprises a multiplier and an adder, where data is fetched from memory, multiplied in the PE, and the results are accumulated before being stored back in memory. This entire process is designed to be completed within a single clock cycle, enabling high-speed computation. This paper focuses on investigating the performance of various digital multipliers used as integral components of PEs in hardware architecture for ML.

This research was supported in part by the German Research Foundation - DFG under Project No. 551672220 (RESIST), by the Horizon Europe Twinning project AIDA4Edge (Grant Agreement No. 101160293), and by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under Grant No. 451-03-137/2025-03/200102.

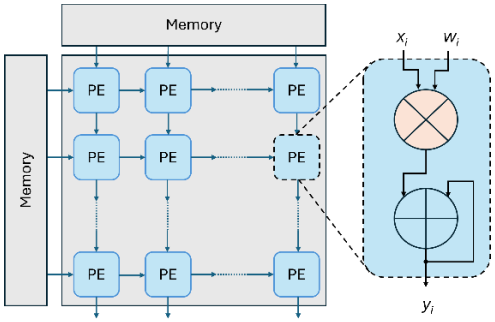

Fig. 1. Block diagram of the MAC systolic array and PE.

II. Digital Multipliers

Multiplication is a fundamental operation in digital systems, and various multiplier architectures have been developed to optimize performance in terms of latency, throughput, area, and design complexity. Latency refers to the real delay in computing the multiplication, specifically the time that elapses from when the inputs are stable until the result is available at the output [4]. Throughput, on the other hand, measures how many multiplications can be performed within a given period, reflecting the system's efficiency in executing operations [4]. In a typical (m×n) bit multiplication process, the operation begins by generating n partial products, each consisting of m bits. These partial products are then summed after appropriate shifting to yield an (m+n) bit result.

The formation of these partial products relies on logical AND operations between the bits of the multiplier and the multiplicand. Different multiplier designs, including the array multiplier and Wallace tree multiplier, are commonly employed in hardware implementations, particularly for VLSI circuits at the CMOS level [5-7]. The use of emerging technologies, such as Quantum-Cellular Automata (QCA), can enhance the performance of multipliers in computational systems [8-10]. Addressing this challenge requires the integration of QCA, which offers benefits like reduced power consumption and increased speed in processing operations. The choice of a specific multiplier is influenced by the targeted optimization in terms of speed, area, and overall complexity. This section provides an overview of some multiplier types, highlighting their underlying algorithms.

A. *Array Multiplier*

Array multipliers implement the multiplication of two numbers using the shift-and-add method, characterized by a highly regular and systematic structure. The architecture consists of an (m×n) array of AND gates that compute all the partial products, while the summation of these products is handled by a combination of [m×(n-2)] full adders (FAs) and n half adders (HAs). The logic diagram of the FA and HA circuits is shown in Fig. 2a. The alignment of partial products through shifting is efficiently managed via routing without the need for additional logic [11,12]. The principle of constructing an array multiplier is illustrated in Fig. 2b through the example of multiplying two 4-bit numbers. Partial product x_0w_0 represents the least significant bit of product- p_0. By adding the partial products x_1w_0 and x_0w_1 using an HA, the value s_1 is obtained, representing p_1, while the carry bit c_1 is processed further. Using an FA, the partial products x_2w_0 and x_1w_1 are summed, along with the carry bit c_1, to obtain the sum s_2, which is then added to the partial product x_0w_2 using an HA to produce the result s_3, corresponding to p_2. The carry bit c_2 is passed to the next FA. The block diagram of the complete 4×4 array multiplier, formed according to the described principle, is shown in Fig. 2c. Despite their structured design, array multipliers exhibit significant delay, particularly for large word lengths, leading to considerable power consumption. The delay in an array multiplier is primarily determined by the depth of the array, rather than the width of the partial products [13]. One of the key advantages of the array multiplier lies in its regular structure, which not only simplifies its layout but also contributes to its compact size [14]. However, despite these benefits, optimization is crucial to address the performance limitations in terms of delay and power consumption, especially for large-scale implementations.

B. *Wallace Multiplier*

The Wallace multiplier is an efficient parallel multiplication scheme that reduces the number of sequential addition stages, thus accelerating the partial product accumulation process. The procedure consists of three main steps: partial product generation (PPG), partial product grouping and reduction (PPGR), and final addition (FADD) [12, 15]. Fig. 3a illustrates the principle of operation of the Wallace multiplier using the example of multiplying two 4-bit numbers. In the first step, partial products are formed using AND gates and are then rearranged as shown in Fig. 3a. The products in the middle columns are added using HAs. In the second stage, the resulting arrangement with sums and carry bits from the first step is correctly placed, and the bits in columns are added using either HAs or FAs. The block diagram of a Wallace multiplier of two 4-bit numbers is given in Fig. 3b. This architecture achieves a delay of only three stages, compared to six stages in conventional array multipliers [12]. The number of reduction stages depends on the bit-width of the numbers being multiplied [15], with delay being proportional to the logarithm of that bit-width [14]. Over the years, various research groups have proposed modifications to further reduce the delay [16-18]. Although the Wallace tree structure is complex to layout, it is particularly suitable for applications where speed is the primary concern, rather than layout regularity [14].

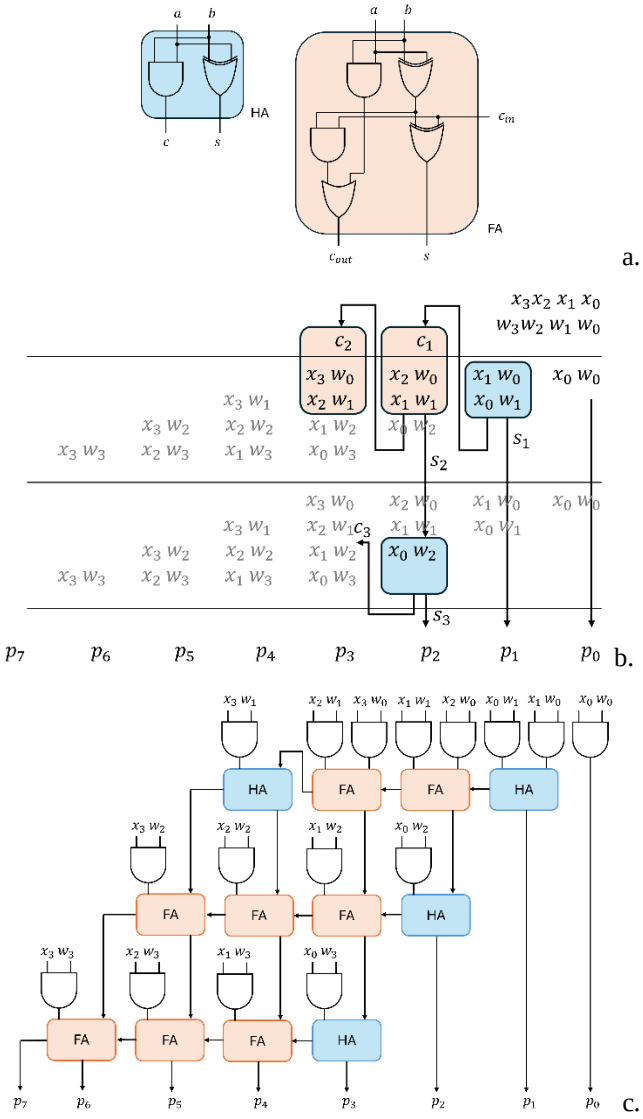

Fig. 2. a. Logic diagram of the half adder and full adder, b. Principle of constructing an array multiplier, c. Block diagram of 4 × 4 array multiplier.

The carry look-ahead (CLA) algorithm accelerates the addition process by computing carry signals in advance, based on the input bits [19]. In a modified full adder (FADD) circuit, the integration of CLA logic enables faster carry calculation, although it increases hardware complexity. The fundamental concept involves predicting the carry bit by analyzing the truth table of a full adder, focusing on the carry-out (see Table I). It can be observed that when both input bits are '1', the carry-out will be '1' regardless of the carry-in. Additionally, when the XOR of the input bits is '1' and the carry-in is also '1', the carry-out is again '1'. From this, the carry-out equation can be expressed as [19]:

$$c_{out} = ab + (a \oplus b)c_{in}, \quad (1)$$

where ab is named the carry generator (G), as it is independent of the carry-in, and ($a \oplus b$) is the carry propagator (P). This leads to a generalized form:

$$c_i = G_i + P_i c_{i-1}, \quad (2)$$

for i = 1, 2, ..., n, which is used to create CLA logic for Wallace multiplier (Fig. 4). While CLA logic provides speed improvements, it becomes increasingly complex for adders larger than 4 bits. As a result, CLA is typically implemented in 4-bit modules and used hierarchically to construct wider adders, such as those found in multiplier circuits [20].

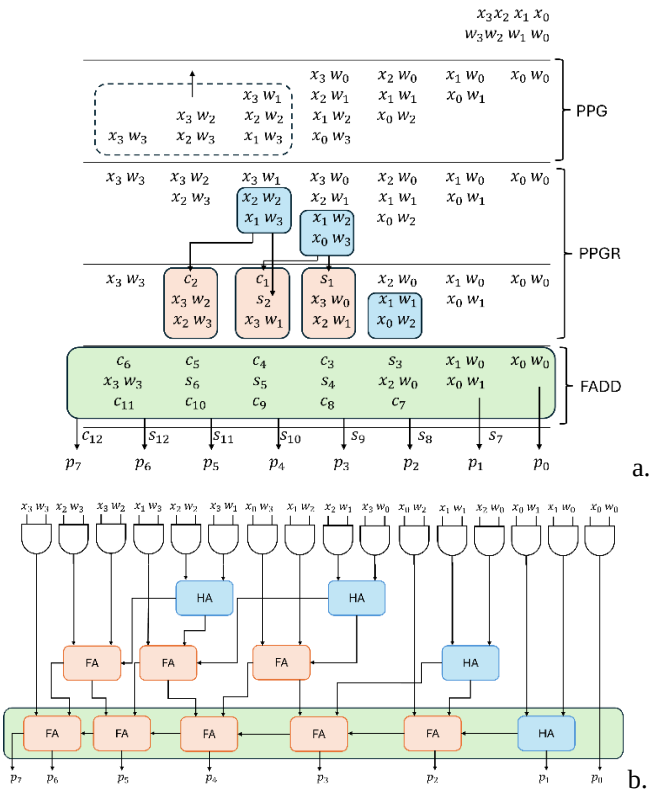

Fig. 3. a. Principle of constructing the Wallace multiplier, b. Block diagram of 4 × 4 Wallace multiplier.

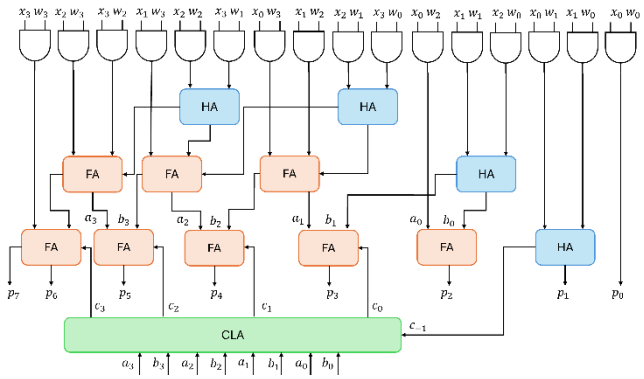

Fig. 4. Block diagram of 4×4 Wallace multiplier with carry look-ahead logic.

III. Results and Discussion

To analyze the performance of the described digital multipliers, SPICE simulations were conducted using Cadence Spectre and IHP's 130 nm bulk CMOS technology. The circuit schematics were created using a standard cell library, subcircuits of HAs and FAs were implemented and interconnected to form array and Wallace multiplier structures (with and without CLA logic). Based on the results of transient simulations, key performance metrics such as propagation delay and power consumption were extracted. Fig. 5 illustrates the output waveform of the array multiplier for the input values X = 1011 and W = 0110, resulting in a product P = 01000010. However, due to propagation delays through the logic gates, not all product bits are generated simultaneously. The simulation demonstrates that the least significant bit of input X is set to '1' and maintained for 1 ns, while the product bit p_1 transitions to '1' after 150 ps, and bit p_6 does not reach its final value of '1' until 870 ps. The selected input vectors are close to the worst-case scenario, inducing a state change in the most significant bit (p_6), and since bits p_6 and p_7 are computed within the same FA, the results would not differ significantly even with greater delays in the worst case.

TABLE I. The Truth Table of a Full Adder

a	*b*	c_{in}	c_{out}	s_{out}
0	0	0	**0**	0
0	0	1	**0**	1
0	1	0	**0**	1
0	1	1	**1**	0
1	0	0	**0**	1
1	0	1	**1**	0
1	1	0	**1**	0
1	1	1	**1**	1

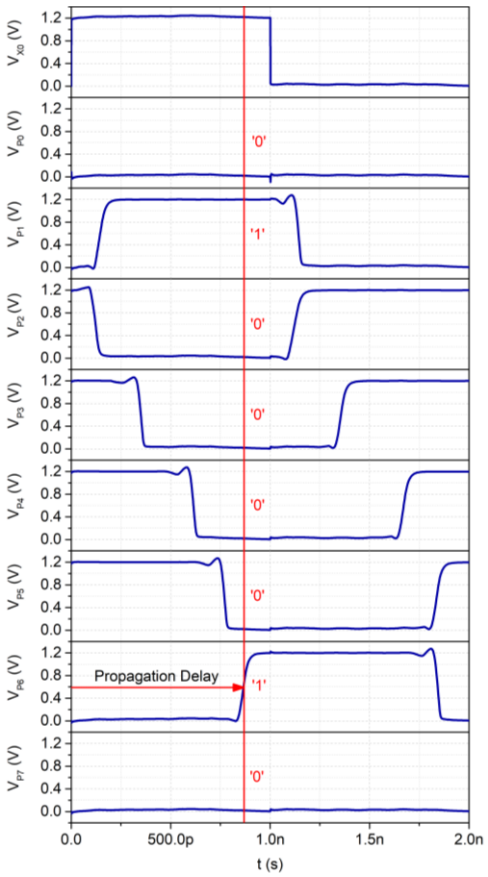

Fig. 5. Output waveform of the array multiplier.

Fig. 6 shows the dependence of propagation delay in different multiplier architectures on the supply voltage (V_{DD})

across various temperatures. The minimum delay is observed at the nominal supply voltage of 1.2 V, while scaling the voltage down to 0.8 V results in a significant increase in delay. Among the analyzed architectures, only the Wallace multiplier with CLA logic remains operational at the lowest supply voltage of 0.8 V, whereas the other two multipliers function correctly only down to 0.9 V. At the nominal supply voltage and room temperature (27 °C), the Wallace multiplier with CLA logic exhibits the shortest delay (585 ps), while the array multiplier shows the longest (865 ps). When the supply voltage is reduced to 0.9 V, the delay in the Wallace multiplier with CLA logic increases by 96.7%, compared to a 98.3% increase in the array multiplier. Although temperature variations affect delay, this influence is less pronounced than that of supply voltage scaling. This phenomenon can be explained by the degradation of carrier mobility with increasing temperature, which alters the circuit's timing behavior. At 1.2 V, the delay in the Wallace-CLA multiplier rises from 557 ps at −40 °C to 602 ps at 125 °C, representing an 8% increase, while the array multiplier shows a 7.4% increase over the same range. Furthermore, with lower supply voltages, the impact of temperature on delay becomes more significant, showing a counterintuitive trend where increasing temperature can reduce delay by up to 20%.

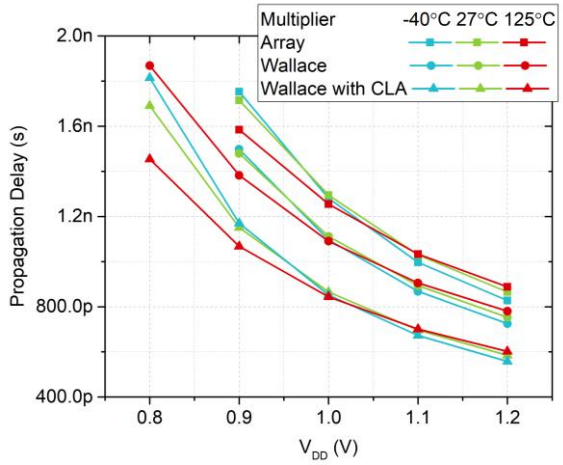

Fig. 6. Propagation delay in different multipliers vs. supply voltage across various temperatures.

Fig. 7 presents power consumption as a function of supply voltage for different multiplier architectures across various temperatures. It can be observed that power consumption increases linearly with the supply voltage and is consistently higher at elevated temperatures for all architectures. This increase in power is primarily attributed to the thermal voltage dependency of leakage current. Due to its higher complexity, the Wallace multiplier with CLA logic exhibits the highest power consumption, 52 µW at $V_{DD} = 0.8$ V and room temperature, rising to 161 µW at the nominal V_{DD} of 1.2 V. Over the temperature range from −40 °C to 125 °C, its power consumption increases by approximately 3% at the nominal V_{DD}. The lowest power consumption is observed in the Wallace multiplier without CLA logic, with 30 µW at $V_{DD} = 0.8$ V and room temperature, increasing to 96.8 µW at $V_{DD} = 1.2$ V. In this case, the temperature-induced increase in power consumption is around 2.5% over the same temperature range.

Fig. 8 illustrates the propagation delay of various multiplier architectures as a function of supply voltage for different driving strengths. In our simulations, standard cells with driving strengths of x0, x1, and x2 were used for the implementation of subcircuits such as HAs, FAs, and CLA logic. In the 130-nm standard cell library under investigation, x2 cells have twice the channel width of x1 cells, while x1 cells have twice the channel width of standard x0 cells. The PMOS transistors in x0 cells have a channel width of 670 nm, while NMOS transistors have a width of 440 nm. Increasing the channel width reduces the channel resistance but simultaneously increases the transistor capacitance. Therefore, finding an optimal channel width is essential to minimize propagation delay. In this case, however, no custom transistor sizing was performed—only predefined standard cells were used. The results show that the lowest delay for the Wallace multiplier with CLA logic is achieved using x0 cells at the nominal supply voltage, with delay increasing as the supply voltage is scaled down. The use of x1 cells results in up to a 2% increase in delay, while x2 cells lead to an increase of up to 10% for the same multiplier architecture. For the other multiplier architectures analyzed, the absolute propagation delays are higher, but the relative increase due to cell sizing remains below 10%. For instance, in the case of the array multiplier, using either x0 or x2 cells results in lower delay compared to x1 cells, where the specific sizing leads to a 3.4% increase in delay.

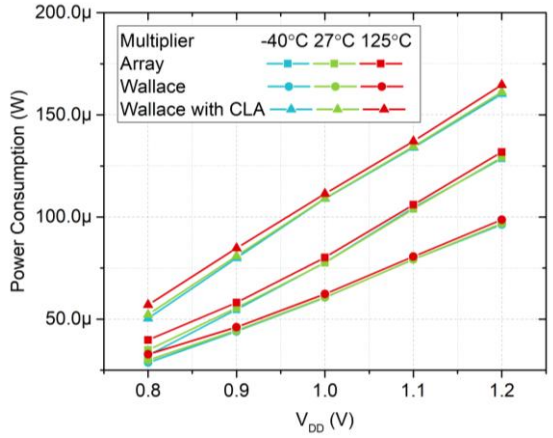

Fig. 7. Power consumption in different multipliers vs. supply voltage across various temperatures.

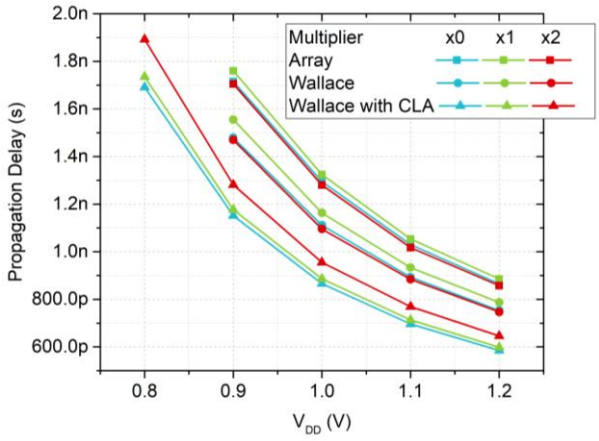

Fig. 8. Propagation delay in different multipliers vs. supply voltage for different driving strengths.

Fig. 9 shows the power consumption of different multiplier architectures as a function of supply voltage for various driving strengths. It is evident that power consumption increases with the supply voltage for all architectures.

Implementations using x0 standard cells consistently exhibit lower power consumption compared to those using x2 cells—up to three times lower. Among the analyzed designs, the Wallace multiplier without CLA logic implemented with x0 cells demonstrates the lowest power consumption. Additionally, the Wallace multiplier with x1 cells consumes less power than the Wallace multiplier with CLA logic implemented using x0 cells. The highest power consumption is observed in both the array multiplier and the Wallace multiplier with CLA logic using x2 cells, which show nearly identical values.

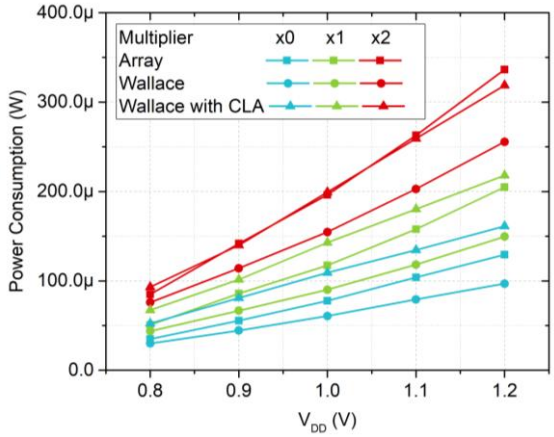

Fig. 9. Power consumption in different multipliers vs. supply voltage for different driving strengths.

From the preceding analysis, it is evident that designing efficient multipliers requires a trade-off between propagation delay, power consumption, and chip area. To quantitatively compare the analyzed multiplier architectures, the Power Delay Product (PDP) metric is introduced, where a lower PDP indicates a more efficient design. The calculated PDP values for all considered architectures at the nominal supply voltage of 1.2 V and room temperature (27 °C) are presented in Fig. 10. The lowest PDP value ($0.73 \cdot 10^{-13}$ J) was obtained for the Wallace multiplier without CLA logic implemented with x0 standard cells. A comparable performance is achieved by the Wallace multiplier with CLA logic (PDP = $0.94 \cdot 10^{-13}$J), although at the cost of significantly increased area (Table II). The least favorable solutions in terms of PDP are those implemented with x2 cells, as they yield the highest PDP values. For reference, the dashed lines in the Fig. 10 indicate PDP values for 4×4 array and Wallace multipliers implemented in 90 nm technology [21] amounting to $1.687 \cdot 10^{-13}$ J and $1.473 \cdot 10^{-13}$ J, respectively. It can be concluded that the array multiplier with x0 cells in 130 nm technology outperforms the referenced design in the literature [18], unlike the versions implemented with x1 and x2 cells. On the other hand, the examined Wallace multiplier designs with and without CLA logic using x0 and x1 cells demonstrate better performance than the 90 nm reference design [21], while designs using x2 cells do not.

TABLE II. ESTIMATED TRANSISTOR COUNTS FOR VARIOUS MULTIPLIER IMPLEMENTATIONS

Multiplier	***Number of transistors***
Array	544
Wallace	544
Wallace with CLA logic	736

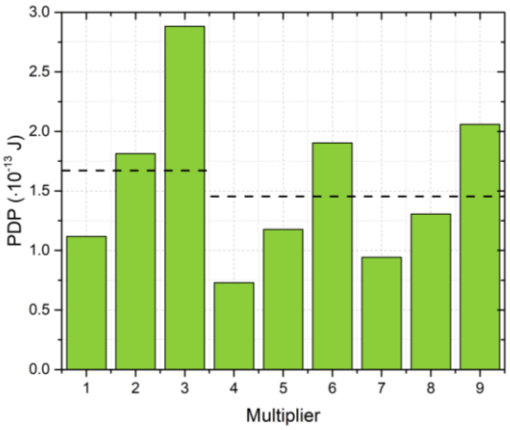

Fig. 10. Power Delay Product of the multipliers: 1 – Array x0, 2 – array x1, 3 – array x2, 4 – Wallace x0, 5 – Wallace x1, 6 – Wallace x2, 7 – Wallace with CLA logic x0, 8 – Wallace with CLA logic x1, 9 – Wallace with CLA logic x2.

IV. CONCLUSION

In this study, we have provided a detailed evaluation of array and Wallace-based multiplier architectures under realistic operating conditions using SPICE simulations in an IHP's 130 nm CMOS process.

Our analysis has covered a supply voltage range from 0.8 V to 1.2 V and included variations in temperature ranging from −40 °C to 125 °C, and driving strength (x0, x1, x2). The results have confirmed that propagation delay increases as supply voltage decreases, while power consumption rises linearly with voltage. Temperature increases have resulted in higher delay and power consumption at nominal voltage. Furthermore, increasing the driving strength has reduced channel resistance but increased capacitance, resulting in a non-linear impact on delay and highlighting the importance of careful transistor sizing. The Power Delay Product metric has been used to quantify overall efficiency. The Wallace multiplier without CLA logic, implemented with x0 cells, has achieved the lowest PDP, outperforming other designs and even surpassing previously published results for similar multipliers in 90 nm technology. Wallace architectures with CLA logic using x0 or x1 cells have demonstrated competitive performance as well. These findings emphasize the importance of architectural choice and cell sizing in balancing performance, power, and area for energy-efficient NN accelerators design. Future research could focus on exploring approximate multiplier designs, particularly considering the energy-error trade-offs for NN inference. Additionally, a detailed analysis of the area overhead, layout regularity, and routing complexity of multipliers should be included, along with layout estimates and the implications on overall system design. Finally, fault analysis, including stuck-at faults and single-event transients (SETs), would be crucial for evaluating reliability and ensuring robustness in practical applications.

REFERENCES

[1] F. Chollet, Deep Learning with Python, 2nd Edition, Manning Publications, 2021.

[2] U. S. Solangi, M. Ibtesam, M. A. Ansari, J. Kim, and S. Park, "Test architecture for systolic array of edge-based AI accelerator," IEEE Access, 2021, vol. 9, pp. 96700-96710.

[3] H.-C. Lu, L.-Y. Su, and S.-H. Huang, "Highly fault-tolerant systolic-array-based matrix multiplication," Electronics, vol. 13, no. 9, p. 1780, 2024.

[4] S. Shanthala, C. Raj, and S. Y. Kulkarni, “Design and VLSI implementation of pipelined multiply accumulate unit,” Proc. of the Second International Conference on Emerging Trends in Engineering & Technology, Nagpur, India, 2009, pp. 381-386.

[5] K. N. Singh, and H. Tarunkumar, “A review on various multipliers designs in VLSI,” Proc. of Annual IEEE India Conference (INDICON), New Delhi, India, 2015, pp. 1-4.

[6] S. Vaidya, and D. Dandekar, “Delay-power performance comparison of multipliers in VLSI circuit design,” Int. J. Comput. Netw. Commun., vol. 2, no. 4, pp. 47-56, 2010.

[7] R. Długosz, K. Kubiak, T. Talaśka, and I. Zbierska-Piątek, “Parallel matrix multiplication circuits for use in kalman filtering,” FU: Elec. Energ., vol. 32, no 4, pp. 479-501, 2019.

[8] R. Faraji, and A. Rezai, “Design of a multilayer reversible ALU in QCA technology,” J. Supercomput., vol. 80, pp. 17135-17158, 2024.

[9] R. Faraji, and A. Rezai, “A novel reversible multilayer full adder circuit design in QCA technology,” FU: Elec. Energ., vol. 37, no. 3, pp. 437-453, 2024.

[10] R. Faraji, and A. Rezai, “Design of new single-bit multilayer ALU in QCA technology,” Integration, 2025, vol. 103.

[11] M.B. Damle, Dr. S.S Limaye, M.G. Sonwani, “Comparative analysis of array multiplier using different logic styles,” IOSR JEN, vol. 3, no. 5, pp. 16-22, 2013.

[12] P. G. Parate, P. S. Patil, and S. Subbaraman, “ASIC Implementation of 4 Bit Multipliers,” Proc. of the First International Conference on Emerging Trends in Engineering and Technology, Nagpur, India, 2008, pp. 408-413.

[13] A. K. Singh, B. P. De, and S. Maity, “Design and comparison of multipliers using different logic styles,” Int. J. Soft Comput. Eng., vol. 2, no. 2, pp. 374-379, 2012.

[14] N. Goel, and L. Garg, “Comparative analysis of 4-bit CMOS multipliers,” Proc. of International Conference on VLSI, Communication & Instrumentation, 2011, pp. 33-36.

[15] S. Khan, S. Kakde, and Y. Suryawanshi, “VLSI implementation of reduced complexity Wallace multiplier using energy efficient CMOS full adder,” Proc. of IEEE International Conference on Computational Intelligence and Computing Research, Enathi, India, 2013, pp. 1-4.

[16] C. Kalamani, V. P. Krishnammal, V. R. Balaji, and C. N. Marimuthu, “Energy efficient Wallace multiplier using symmetric stacking counter circuit,” Meas.: Sensors, vol. 35, p. 101267, 2024.

[17] K. Juneja, A. Jangra, and D. Khurana, “Design of a quaternary component and Wallace tree integrated Baugh-Wooley multiplier,” Int. J. Netw. Distrib. Comput., vol. 13, p. 3, 2025.

[18] I. Rizos, G. Papatheodorou, and A. Efthymiou, “Designing approximate reduced complexity Wallace multipliers,” Electronics, vol. 14, p. 333, 2025.

[19] A. M. Bhavani et al. “Design and implementation of Wallace tree multiplier using parallel prefix adders,” Int. J. Innov. Res. Technol., vol. 9, p. 1, 2022.

[20] D. Ravi, “Design and implementation of Wallace tree multiplier using higher order compressors,” Int. J. VLSI System Design Commun. Syst., vol. 4, no. 6, pp. 442-448, 2016.

[21] I. Hussain, C. K. Pandey and S. Chaudhury, “Design and analysis of high performance multiplier circuit,” Proc. of Devices for Integrated Circuit (DevIC), Kalyani, India, 2019, pp. 245-247.