

Fault tolerant design for 8-bit Dadda multiplier for neural network applications

Raji Chandrasekharan¹, Sarappadi Narasimha Prasad²

¹School of Electronics and Communication Engineering, REVA University, Bangalore, India

²Department of Electrical and Electronics Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India

Article Info

Article history:

Received Jul 25, 2024

Revised Jan 31, 2025

Accepted Mar 3, 2025

Keywords:

Carry select adder

Dadda multiplier

Fault tolerant

Low power

Self-repairing full adder

ABSTRACT

As digital electronic systems continue to shrink in size, they face increased susceptibility to transient errors, especially in critical applications like neural networks, which are not inherently error-resilient. Multipliers, fundamental components of neural networks, must be both fault tolerant and efficient. However, traditional fault free designs consume excessive power and require substantial silicon real estate. Among existing multiplier architectures, the Dadda multiplier stands out for its speed and efficiency, but it lacks fault tolerance needed for robust neural network applications. Therefore, there is need to design a power efficient and fault free Dadda multiplier that can address these challenges without significantly increasing power consumption or hardware complexity. In this paper a solution involving a fault tolerant Dadda multiplier optimized for neural network applications is proposed. Because of its speed and efficiency when compared to other multipliers Dadda multiplier is used as the base architecture which is designed using carry select adder (CSA) in conjunction with binary to excess one converter to reduce power and complexity. To enhance fault tolerance, self-repairing full adder is used to implement the CSA. This allows the system to detect and correct errors, ensuring robust operation in the presence of transient faults. This combination achieves a power efficient, fault tolerant multiplier with a power consumption of 52.3 mW, reflecting a 3% reduction in power compared to existing designs.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Sarappadi Narasimha Prasad

Department of Electrical and Electronics Engineering, Manipal Institute of Technology Bengaluru,

Manipal Academy of Higher Education

Manipal, Karnataka, 576104, India

Email: sn.prasad@manipal.edu

1. INTRODUCTION

Fault tolerance can be defined as the property which helps a system to continue operating properly even if there are failures in some of the constituent components. Designs which are fault tolerant enable the systems to operate at a reduced level rather than failing completely if some part of the system fails. The systems must have the ability to detect the failures and repair them in as much less time as possible.

A significant portion of all hardware defects are transient faults caused by radiation or temperature impacts. For any system to be fault tolerant system, self-checking and self-repairing properties are required. The most commonly used methods to address faults are redundancy schemes: time redundancy, hardware redundancy and information redundancy. Protection from transient fault is proposed using Time redundancy in [1] in which, similar operations are carried out by the clone hardware, in addition to the primary hardware,

at different time intervals. Difference in time interval to the duplicate hardware is implemented by providing delayed clock. Detection of fault is done by comparing the outputs arriving at different intervals. By performing similar operations at different time intervals an area overhead and cost can be reduced. In Hardware redundancy method more than one hardware is used to produce different outputs. Outputs of the redundant and the original are compared to determine the faulty or fault free condition. The major disadvantage is the area overhead incurred [1]. To detect real time faults information redundancy could be used. Fault detection in combinational logic can be carried out by using error detection codes or predicted parity bits. The major drawback of area overhead in hardware redundancy could be overcome by using multiple parity groups [1]. Various approaches are used to attain fault tolerance but at the cost of power, area and delay. Thus, by incorporating fault tolerance and a low power design strategy for the fundamental building blocks, the problem of defective systems and power-hungry redundancy schemes will be reduced.

Adders and multipliers are the building blocks of digital circuits as well as neural networks. To make digital systems fault free it is imperative to use fault tolerant adders and multipliers. The time required for execution of the algorithms is dependent on the multiplication process which is where we require high speed multipliers. The comparative analysis of delay power performance of various multipliers given in [2] shows that there are several architectures to choose balancing power, delay and area. To implement fault free circuits, multiplier designs which are fault tolerant should be adopted. This can be achieved using fault tolerant adders to design multipliers. The Dadda multiplier, which is known for being faster and more efficient, is selected as the base multiplier architecture. To further enhance the Dadda multiplier, a carry select adder (CSA) is proposed. The CSA improves the speed of arithmetic operations by precomputing multiple sums in parallel. In this design, the CSA uses a binary to excess one converter, which replaces the conventional adders to reduce power consumption and area. This setup allows overall reduction in hardware complexity while maintaining performance. To achieve fault tolerance, self-repairing adder is employed in the CSA. This design approach helps handle single and double faults effectively thus contributing to the reliability of the multiplier and ensuring that errors can be self-corrected. One of the fastest adder in digital systems is CSA which performs better arithmetic operations. Using fault tolerant CSA Dadda Multiplier design is proposed here. The proposed CSA is built using self-repairing full adder which is fault tolerant. This Dadda multiplier can be used in neural network applications.

2. RELATED WORK

Many low power implementations of adders using pass transistor logic, transmission gate [3], [4] results in output degradation and consumes more power. Using pass transistor and branch-based logic, full adder design [5] can reduce the dynamic power consumed. Reduced power consumption can be obtained from hybrid full adder using gate diffusion method [6], [7]. The full adder mentioned in [8] which uses 16 transistors is a very promising design for the application outlined in this study since it has low power dissipation and good output fidelity. To keep the integrity, a hybrid 20-T Full adder that generates full swing at both the sum and carry could be used. Compact and low power architecture using shared adder [9] could be used in implementing CSA. The CSA adder is regarded as one of the quickest adders for solving arithmetic functions in majority of the digital processing systems [10]. The propagation of carry to the following step is the primary factor that restricts the speed of addition. Usage of CSA can be seen in a variety of processing systems to obtain sum by producing a number of carry from which one carry is chosen and hence optimizing the delay issue of carry propagation [11]. However, the major issue here is that the CSA uses numerous pairs of ripple carry adders (RCA) by seeing carry as input to construct the carry and the respective partial sum. The final sum and carry are chosen by a multiplexer (MUX), and the entire process shows that this approach is inefficient use of area. Therefore, a fundamental idea of applying a binary to excess-1 converter (BEC-1) is provided to advance the speed of addition [12].

Based on the capabilities, the multiplication procedure can help the system to reach a high data rate [13]. We need high speed multiplier because the majority of the algorithms' execution times depend on the multiplication operation. A huge volume of data must also be evaluated in a variety of applications of digital processing systems and very large-scale integration (VLSI) which calls for high performance processors [14]. Multipliers are essential in achieving improved performance outcomes for this goal. Therefore, the multiplier's operating speed is crucial and must be taken into account when developing circuits for digital systems specially for general-purpose processing [15]. There have been numerous attempts in recent years to create multipliers that offer high speed and low power. In essence, multipliers come in two flavors: serial multipliers and parallel multipliers. In general, parallel multipliers operate more quickly than serial multipliers [13]–[16]. Several multipliers like array multipliers, Dadda multipliers, Wallace, Booth multipliers are studied for their performance in [17]–[19]. Different algorithms and strategies have already been developed to achieve the least amount of delay, power consumption, and output through a digital

circuit. Merge delay transform [17], [18], genetic evolutionary algorithms are a few of the strategies now in use. All of these methods are implemented using complementary metal-oxide-semiconductor (CMOS) technology. Low power design approaches for other applications for VLSI implementation discussed in [20]–[22] is studied to see whether those can be applied for the application tried here. There are three steps in the operation of multipliers: creation of a partial product, its addition, and its final addition. The second stage of multiplication often involves the partial product reduction process, which can be bottleneck in terms of performance and power consumption. Parallel prefix adders, approximate adders, novel compressors are some of the methods reviewed in [23]. Dadda multiplier designed using compressors are discussed in [24] shows considerable prospects in reducing area and power and delay. A non-truncated design is used to design Wallace multiplier in [25] using approximate compressors which shows increased performance and less power consumption. Applications of approximate compressors for multipliers used for image processing discussed in [26] is prospective design as the accuracy metric is not compromised and has less power consumption. In multiplier design as discussed in [27] uses an and gate to generate a correction term to correct errors. Approximation method for multiplication using 4:2 compressors and full adders is discussed for image processing applications in [28]. Approximate computing assessment techniques for internet of things (IoT) and machine learning applications where requirement for accuracy is not paramount is discussed in [29]. Recursive based approximate multipliers for error resilient application is discussed in [30] which is a prospective contender in low power applications. In the existing architecture, it is found that different multipliers are build using full adder which are not fault free. In order to overcome this problem, Dadda multiplier is implemented using CSA which is fault tolerant with low power implementation.

3. PROPOSED DESIGN

3.1. Proposed fault tolerant carry select adder

The multiply accumulate unit (MAC) consists of adders, multipliers and accumulators where the multiplier is the heart of every MAC unit. The reduction and generation of partial product in multipliers has large contribution to power consumption. A regular CSA includes two RCA which produces sum and carry. One RCA generates ‘Carry 0’ and the second RCA generates ‘Carry 1’. Multiplexers are used for the selection of any one carry pin output. But the carry select adder-based Radix-multiplier consumes larger area and more number of transistors. To avoid more power consumption fault tolerant CSA using binary to excess one converter (BEC-1) is implemented. The constituent fault tolerant self-repairing full adder [8] block as shown in Figure 1 is used in CSA in turn making the CSA fault free. For the sake of brevity 4-bit adder design is considered here. Figure 2 represents the block diagram of fault tolerant CSA. Here carry 1 operates the same as that of regular CSA but carry 0 makes use of BEC-1 that adds 1 carry in the sum. Finally, MUX used to select the end result based on actual carry Cin. Equations (1) and (2) represent the sum and the carry out for the fault tolerant 4-bit carry select adder.

$$Si = (Xi * Cin) + (Xxi * Cin) \quad (1)$$

$$Cout = (C4 * Cin) + (Xx3 * Cin) \quad (2)$$

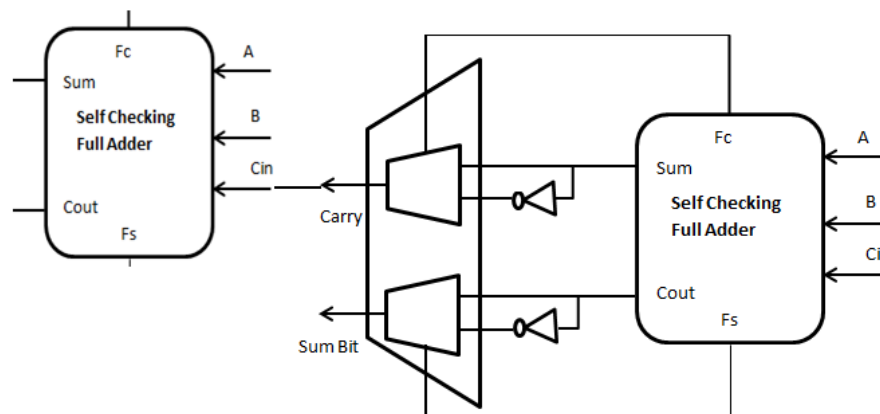


Figure 1. Block diagram of self-repairing full adder consisting of self-checking full adder which identifies fault and a decision-making MUX [8]

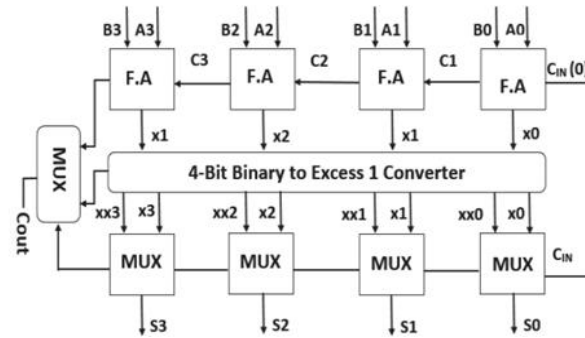


Figure 2. Block representation of 4-bit carry select adder using binary to excess 1 converter, full adder and 2:1 MUX [23]

3.2. Proposed Dadda multiplier

Dadda multiplier works similar to that of Wallace tree multiplier, but the difference is it is much faster and requires fewer gates. The proposed carry select adder is implemented in Dadda multiplier to obtain low power. Figure 3 represents the dot diagram of 8-bit Dadda multiplier where the height of each stage depends on the working principle of the final stage. The final stage includes two partial product rows. Each stage has a height of order 2, 3, 4, 6, 9, 13, 19, 28, 42, 63. The complete 8-bit dot diagram includes six stages. This stage includes 16 number of rows.

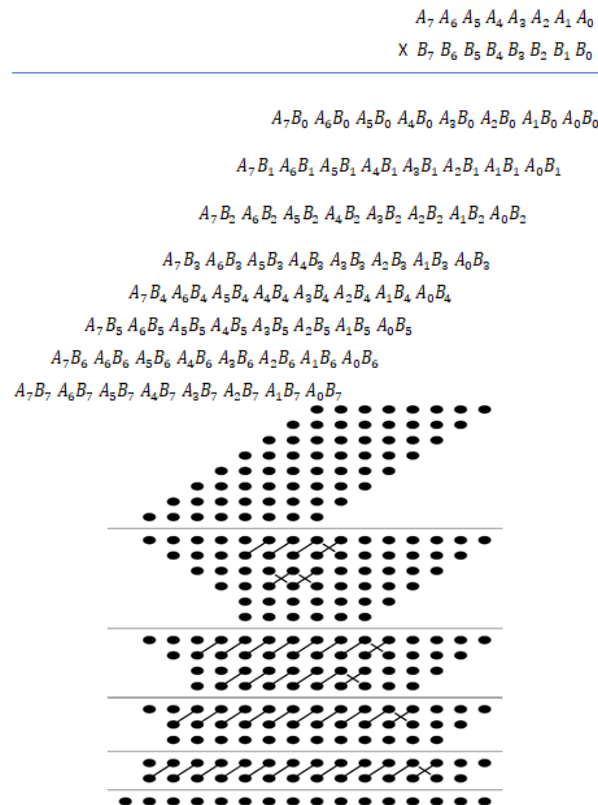


Figure 3. Dot diagram of Dadda multiplier for N=8

The first stage involves simple multiplication of multiplicand and multiplier which is termed as partial product stage. The first column in first row represents the least significant bit. 2nd row stage is being obtained from 3rd row stage, and 3rd row stage is obtained from 4th row stage with the help of (3, 2) and (2, 2) counters from the dot diagram. S indicated the number of stages used to implement the multiplier. In the

(S-2) stage, 6th row stage helps in deriving the 4th row stage. The 6th row can be obtained from 9th row stage. This can be (S-3)th stage. In the (S-4) stage, the 9th row stage is gained by 13th row and it can be obtained from partial product stage. The needed number of full adders in this Dadda implementation is N^2-4N+3 , and required number of half adders is $N-1$. Figure 3 shows the total number of reduced stages to design Dadda architecture. For clarity 4-bit multiplier operation is shown here. 8-bit operation can be obtained by using two of such 4-bit blocks. Figure 4 represents the block diagram of 4-bit Dadda multiplier.

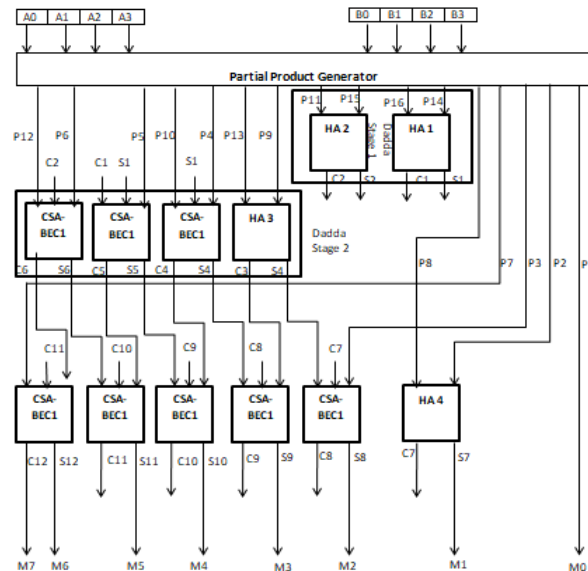


Figure 4. Block diagram representation of Dadda multiplier using carry select adder implemented using the design mentioned in Figure 2

4. RESULTS AND DISCUSSION

The proposed Dadda multiplier is designed using ISE Xilinx software and the waveform is simulated using Xilinx simulator. Figure 5 represents the simulation results of fault tolerant self-repairing full adder. The faults occurred in the full-added circuit is repaired by the self-repairing process.

Figure 6 represent schematic diagram of 4-bit fault tolerant carry select adder using excess 1 convertor where there are 4 bits of input 'A' and 4 bits of input 'B' along with C_{in} and $C_{in} 0$. It produces an output of sum $S0 - S3$ and carryout $Cout$. Figure 7 represents RTL schematics view of 4-bit fault tolerant carry select adder obtained after synthesizing the code. Figure 8 shows simulation waveform of 4-bit fault tolerant carry select adder. The inputs are $A_0 - A_3$ and $B_0 - B_3$. Outputs Sum is represented as $S0 - S3$ while the carry is represented as $Cout$.

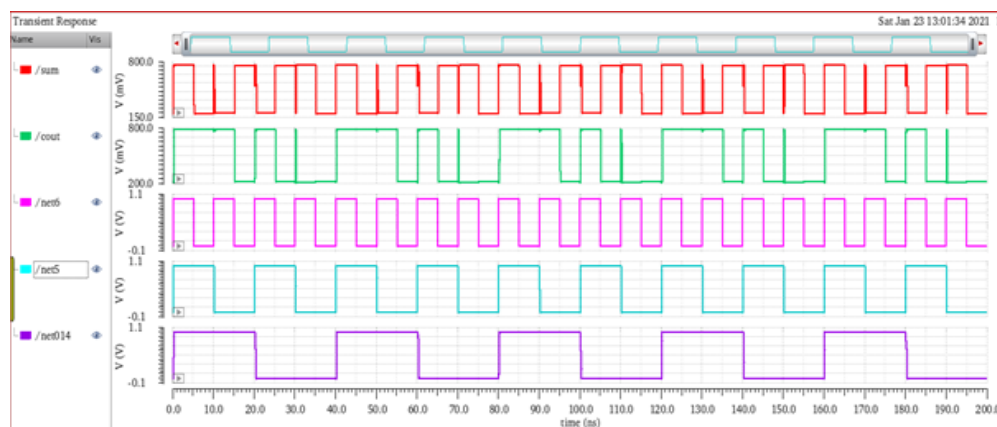


Figure 5. Simulation results of self-repairing full adder

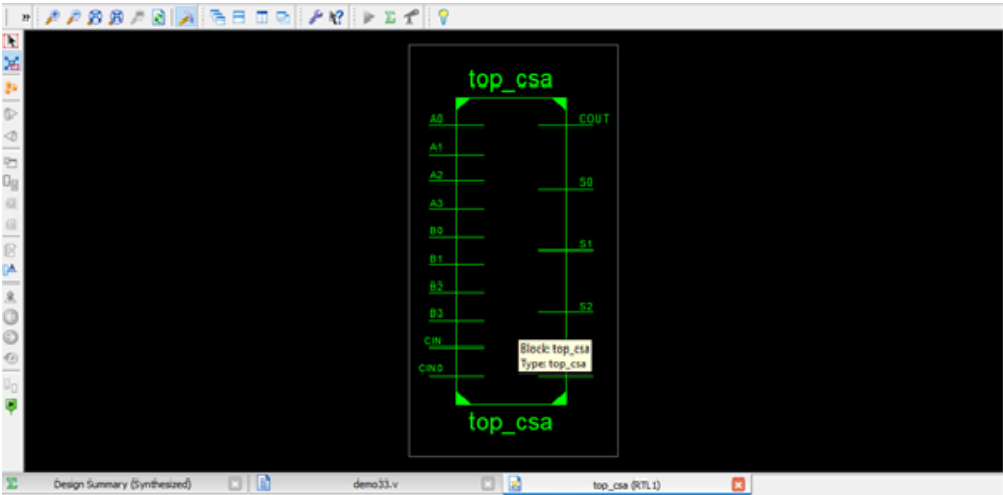


Figure 6. Top level schematic diagram of 4-bit fault tolerant carry select adder

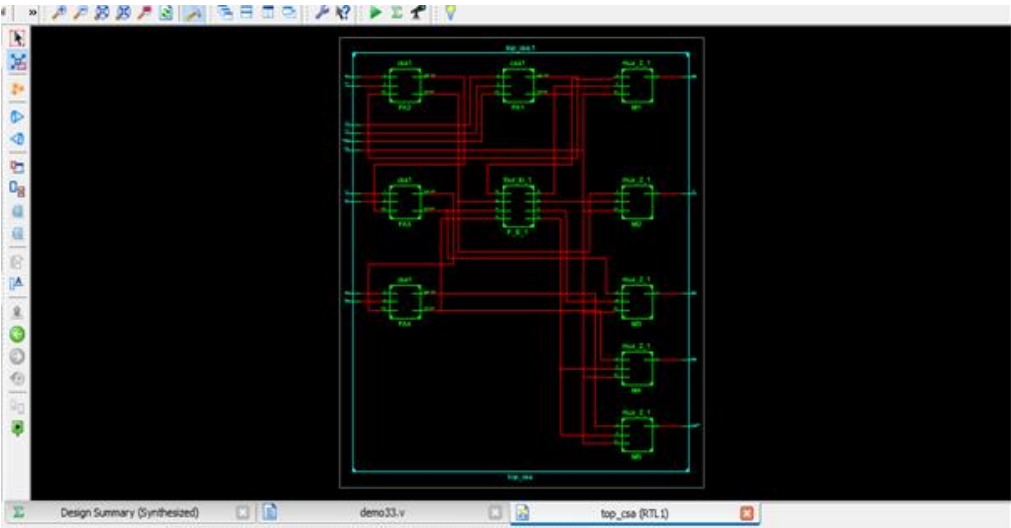


Figure 7. RTL Schematic view of 4-bit fault tolerant carry select adder

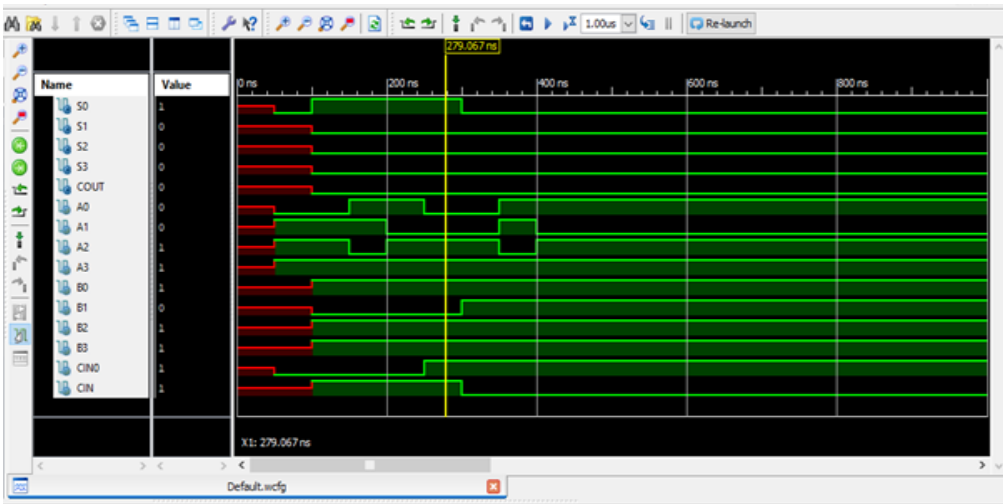


Figure 8. Simulation result of 4-bit fault tolerant carry select adder

8-bit Dadda multiplier can be made by extending the 4-bit design discussed previously. The 8-bit Dadda multiplier which is implemented using fault tolerant CSA with BEC. It includes 8 bits input of A and B and produces a final output of 16-bit.

Figure 9 shows the RTL Schematic view and Figure 10 represents simulated waveform of 8-bit Dadda multiplier. For different values inputs the results are verified. Table 1 shows comparative result for power and delay of 8-bit Wallace, Array and Dadda multiplier. The comparison results show that among all the multipliers Dadda multiplier with CSA using excess one converter in its carry propagation path is more efficient. The power is reduced by 3% when compared to Wallace Tree multiplier and the delay is reduced by 2.8%.

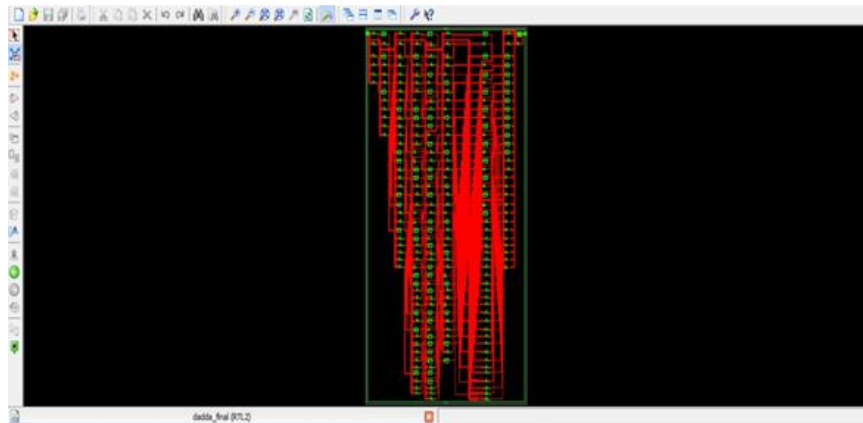


Figure 9. RTL schematic view of 8-bit Dadda multiplier

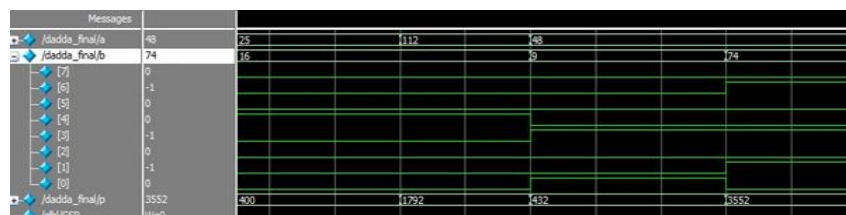


Figure 10. Simulation result of 8-bit Dadda multiplier

Table 1. Comparison of different multipliers based on delay and power dissipation

Multipliers (8 by 8)	Delay (ns)	Power (mW)
Array multiplier	3.02	5.16
Wallace multiplier	2.81	5.39
Dadda multiplier	2.73	5.23

5. CONCLUSION

Neural networks are not inherently fault tolerant. It has to be made fault free. Implementations of fault free carry select adder and multiplier can result in the constituents of a neural network fault free. The Self-Repairing full adder design can identify and repair both single and double faults at a time. The aim of fault-tolerant circuits is to reduce the probability of failures. The constituent self-repairing full adder which is used for CSA is fault tolerant in nature. Since this is used for building CSA, in turn making it fault free. 8-bit Dadda multiplier which has low power dissipation is implemented using this CSA. The desired output waveforms are obtained. In this paper, an implementation of the Dadda multiplier is carried out. The Power consumed by the proposed Dadda multiplier is 5.23 mW. The power comparison results show that the proposed design reduces the power by 3% than array multiplier and wallace multiplier. CSA with BEC that is fault tolerant is designed and implemented in the proposed multiplier. From all the comparison results it is evident that the Dadda multiplier using fault tolerant CSA with BEC more efficient. The proposed 8-bit Dadda tree multiplier which uses the fault-tolerant CSA using BEC-1 reduces power significantly more than several other existing multipliers. These multipliers, though, are also fairly quick compared to other multipliers.

ACKNOWLEDGEMENTS

The authors acknowledge the support from REVA University for the facilities provided to carry out the research.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Raji Chandrasekharan	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓			
Sarappadi Narasimha Prasad		✓				✓			✓	✓		✓	✓	

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.




REFERENCES

- [1] D. Miljković, "Fault detection methods: a literature survey," in *MIPRO 2011 - 34th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings*, 2011, pp. 750–755.
- [2] M. Ali, M. Welzl, S. Hessler, and S. Hellebrand, "A fault tolerant mechanism for handling permanent and transient failures in a network on chip," in *Proceedings - International Conference on Information Technology-New Generations, ITNG 2007*, 2007, pp. 1027–1032. doi: 10.1109/ITNG.2007.5.
- [3] S. Vaidya and D. Dandekar, "Delay-power performance comparison of multipliers in VLSI circuit design," *International journal of Computer Networks & Communications*, vol. 2, no. 4, pp. 47–56, 2010, doi: 10.5121/ijcnc.2010.2405.
- [4] M. Guduri, V. K. Agarwal, and A. Islam, "Which is the best 10T static CMOS full adder for ultralowpower applications," *Journal of VLSI Design Tools & Technology*, vol. 5, no. 1, pp. 45–50, 2015.
- [5] T. V. Rao and A. Srinivasulu, "16-BIT RCA implementation using current sink restorer structure," *International Journal of Design, Analysis and Tools for Circuits and Systems*, vol. 4, no. 1, 2013.
- [6] M. Hasan, H. U. Zaman, M. Hossain, P. Biswas, and S. Islam, "Gate diffusion input technique based full swing and scalable 1-bit hybrid full adder for high performance applications," *Engineering Science and Technology, an International Journal*, vol. 23, no. 6, pp. 1364–1373, 2020, doi: 10.1016/j.jestech.2020.05.008.
- [7] O. A. Badry and M. A. Abdelghany, "Low power 1-Bit full adder using full-swing gate diffusion input technique," in *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*, Feb. 2018, pp. 205–208. doi: 10.1109/ITCE.2018.8316625.
- [8] C. Raji and S. N. Prasad, "A hybrid design for low-power fault tolerant OneBit full adder for neural network applications," *Lecture Notes in Electrical Engineering*, vol. 836, pp. 277–293, 2022, doi: 10.1007/978-981-16-8542-2_22.
- [9] B. Amelifard, F. Fallah, and M. Pedram, "Closing the gap between carry select adder and ripple carry adder: a new class of low-power high-performance adders," in *Sixth International Symposium on Quality of Electronic Design (ISQED'05)*, 2005, pp. 148–152. doi: 10.1109/ISQED.2005.131.
- [10] N. Sharma and R. Sindal, "Modified booth multiplier using wallace structure and efficient carry select adder," *International Journal of Computer Applications*, vol. 68, no. 13, pp. 39–42, 2013, doi: 10.5120/11643-7130.
- [11] N. Ravi, A. Satish, T. J. Prasad, and T. S. Rao, "A new design for array multiplier with trade off in power and area," *arXiv preprint arXiv:1111.7258*, 2011.
- [12] K. Priya. K and K. S. N. Raju, "Carry select adder using BEC and RCA," *Ijarccce*, pp. 8211–8214, 2014, doi: 10.17148/ijarccce.2014.31030.
- [13] M. Vlăduțiu, *Computer arithmetic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-18315-7.
- [14] N. Bano, "VLSI design of low power Booth multiplier," *International Journal of Scientific & Engineering Research*, vol. 3, no. 2, pp. 2–4, 2012.




- [15] W. J. Townsend, E. E. Swartzlander, Jr., and J. A. Abraham, "A comparison of Dadda and Wallace multiplier delays," *Advanced Signal Processing Algorithms, Architectures, and Implementations XIII*, vol. 5205, p. 552, 2003, doi: 10.1117/12.507012.
- [16] P. R. Cappello and K. Steiglitz, "A VLSI layout for a pipelined dadda multiplier," *Computer Arithmetic: Volume II*, pp. 197–214, 2015, doi: 10.1142/9789814641470.
- [17] V. Kaushik, "Comparative analysis of proposed parallel digital multiplier with Dadda and other popular multipliers," *IJIRST – International Journal for Innovative Research in Science & Technology*, no. September, 2017.
- [18] Z. Shabbir, A. R. Ghuman, and S. M. Chaudhry, "A reduced-sp-D3Lsum adder-based high frequency 4×4 bit multiplier using Dadda algorithm," *Circuits, Systems, and Signal Processing*, vol. 35, no. 9, pp. 3113–3134, Sep. 2016, doi: 10.1007/s00034-015-0201-7.
- [19] R. A. Javali, R. J. Nayak, A. M. Mhetar, and M. C. Lakkannavar, "Design of high speed carry save adder using carry lookahead adder," in *Proceedings of International Conference on Circuits, Communication, Control and Computing, I4C 2014*, 2014, pp. 33–36. doi: 10.1109/CIMCA.2014.7057751.
- [20] R. A. Vural, B. Erkmen, U. Bozkurt, and T. Yildirim, "CMOS differential amplifier area optimization with evolutionary algorithms," *Lecture Notes in Engineering and Computer Science*, vol. 2, pp. 666–670, 2013.
- [21] S. López, Ó. Garnica, I. Hidalgo, J. Lanchares, and R. Hermida, "Power-consumption reduction in asynchronous circuits using delay path unequalization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2799, pp. 151–160, 2003, doi: 10.1007/978-3-540-39762-5_17.
- [22] M. Munawar, Z. Shabbir, and M. Akram, "Area, delay, and energy-efficient full Dadda multiplier," *Journal of Circuits, Systems and Computers*, vol. 32, no. 15, 2023, doi: 10.1142/S0218126623502584.
- [23] A. Sebastian, F. Jose, K. Gopakumar, and P. Thiagarajan, "Design and implementation of an efficient Dadda multiplier using novel compressors and fast adder," in *3rd International Symposium on Devices, Circuits and Systems, ISDCS 2020 - Proceedings*, 2020, pp. 1–4. doi: 10.1109/ISDCS49393.2020.9263014.
- [24] A. Gorantla and P. Deepa, "Design of approximate compressors for multiplication," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, 2017, doi: 10.1145/3007649.
- [25] M. Vidhyalakshmi and S. Padmapriya, "High throughput and less error approximate compressor design for partial product reduction in multipliers," in *International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering, ICSSEEC 2024 - Proceedings*, 2024, pp. 19–22. doi: 10.1109/ICSSEEC61126.2024.10649474.
- [26] G. Harish, R. Mathumitha, R. S. Dharini, P. A. Valli, and N. Mohan, "Low power, high accuracy approximate multiplier for error-resilient image processing application," in *8th IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics, DISCOVER 2024 - Proceedings*, 2024, pp. 159–163. doi: 10.1109/DISCOVER62353.2024.10750693.
- [27] U. Anil Kumar, S. K. Chatterjee, and S. E. Ahmed, "Low-power compressor-based approximate multipliers with error correcting module," *IEEE Embedded Systems Letters*, vol. 14, no. 2, pp. 59–62, 2022, doi: 10.1109/LES.2021.3113005.
- [28] B. Rashidi, "Efficient and low-cost approximate multipliers for image processing applications," *Integration*, vol. 94, 2024, doi: 10.1016/j.vlsi.2023.102084.
- [29] A. M. Dalloo, A. J. Humaidi, A. K. Al Mhdawi, and H. Al-Raweshidy, "Approximate computing: concepts, architectures, challenges, applications, and future directions," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3467375.
- [30] E. J. Rao and P. Samundiswary, "High-speed and low-power recursive rounding based approximate multipliers for error-resilience applications," *Wireless Personal Communications*, vol. 136, no. 2, pp. 773–791, 2024, doi: 10.1007/s11277-024-11283-0.

BIOGRAPHIES OF AUTHORS



Raji Chandrasekharan    received the B.Tech. degree in Electrical & Electronics Engineering from Mahatma Gandhi University, India, in 1999 and the M.Tech. degree in VLSI Design and Embedded Systems from Viswesvaraya Technological University, Karnataka, India in 2009. Currently, she is pursuing her research and is working as an assistant professor at the School of Electronics and Communication Engineering, REVA University. Her research interests are digital and analog VLSI and artificial neural networks. She enjoys teaching circuit theory and microelectronics. She can be contacted at email: raji.c@reva.edu.in.



Sarappadi Narasimha Prasad    has completed his Bachelor of Engineering from Mangalore University in 1995, master's from Visvesvaraya Technological University in 2005 and Ph.D. from Jain University in 2016. Dr S. N. Prasad has 25 years of academic and research experience in technical education. In his career, he has taken several administrative and academic assignments at university level such as professor-electronics and communication engineering, special officer (evaluation), assistant registrar (evaluation), assistant director-electronics and communication engineering. He is guiding 7 research scholars at present, out of which 2 scholars have already completed. He has more than 70 publications in various indexed journals and international conferences. He has chaired many International Conferences and played a part in organizing an international conference in 2014 and 2018. He is a member of IETE, IEEE and ISTE chapters. He can be contacted at email: sn.prasad@manipal.edu.