ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2023.0265

A POWER PREDICTION FOR A HIGH-SPEED VLSI ADDER CIRCUIT USING DEEP GENETIC MECHANISM

Maram Ashok[1], N. Jagadeeswari[2], K. Balaji[3] and M. Ramkumar[4]

[1]*Department of Computer Science and Engineering, Malla Reddy Institute of Engineering and Technology, India*
[2]*Department of Computer Science and Engineering, Thanthai Periyar Government Institute of Technology, India*
[3]*Department of Electronics and Communication Engineering, SSM College of Engineering, India*
[4]*Department of Computer Science and Business Systems, Knowledge Institute of Technology, India*

Abstract

Deep Submicron Architectures (DSAs) are utilised to design integrated circuits (ICs) that are both low power and high speed. In this paper, we present a new approach to Deep Genetic Approach (DGA), which is a hybrid error reduction LOA that is designed to increase the computation accuracy of the LOA. This approach is based on the use of OR and XOR to obtain an approximate adder for the inputs that have the (n-k-1)th least significant bit of both inputs to further reduce the output errors. The power consumption and latency of various implementations of the planned DGA are analysed. It has been discovered that using MTCMOS and GDI together leads to a large reduction in leakage power as well as an improvement in latency. In addition, the power delay product (PDP) of the CSA is one of the variables that can be significantly improved by adopting the way that was proposed.

Keywords:

VLSI, Adder, Prediction, Mechanism, Genetics, High-Speed

1. INTRODUCTION

The process of adders is typically taken care of by digital circuits rather than being done by hand in digital systems. It is required for the division step, as well as for the multiplication and subtraction steps [1]. The investigation of all digital systems begins with the process of addition. The greater the number of transistors that are present, the greater the amount of power that must be consumed [2]. One of the most important requirements for VLSI designs is to have a low overall power consumption. Deep submicron technologies, where they are utilised to do complicated criterion evaluations for the purpose of the future creation of integrated circuits (ICs) that are both low power and high speed [3].

As a direct result of recent developments in mobile computing and communication, there has been a noticeable increase in the amount of focus placed on the research and development of low-power VLSI systems [4]. In contrast to the rapid progress that has been made in the world of microelectronics, there has been very little innovation made in the field of battery technology. Designers are having more limits put upon them, such as those requiring fast speed, high performance, extremely small silicon areas, and low power consumption [5].

In recent years, a variety of additional logical approaches to the production of 1-bit adder cells have been found. These strategies have been made available to choose from as different options. In terms of the logical framework, there are primarily two different kinds of adder cells that can be used [5]. Both have their very own one-of-a-kind styles, with one being more static than the other and the other being more dynamic. When compared to dynamic adders, static logics such as C-CMOS, CPL, TGA, and TFA, as well as hybrid static logics, are typically more reliable, easier to work with, and demand a significantly lower amount of energy [6].

The dynamic mode has a variety of advantages over the static mode including higher switching speeds, zero power consumption from leakage, the lack of reasoning logic, complete oscillation voltage levels, and a smaller number of transistors. The dynamic mode also has several advantages while the static mode does not [7]. Several researchers in the field of computer science have advocated for the development of full dynamic-static hybrid adders. These adders use the most advantageous features of both static and dynamic structures and merge them into one [8].

In the next round of their investigation, the researchers turned their focus to a method known as hybrid logic, with the goal of improving the overall performance [9]. This makes use of the most effective features of a wide array of different schools of thought concerning reasoning [10]. Using this logic (hybrid) approach, you will be able to select the optimal circuits for each module, ensuring that the performance of your 1-bit adder cell is at its very best [11]. This allows to select the optimal circuits for each module. By methodically drawing each individual module, it is possible to minimise the needs for power, latency, and physical space of the complete 1-bit cell circuit to the absolute minimum.

2. PROPOSED ADDER

In this section, we discuss the proposed adder, which is a hybrid error reduction LOA. This LOA is designed to increase the computation accuracy of the LOA by employing an innovative hybrid error reduction approach. We are the ones that came up with the hybrid approach to error reduction. To explain the adder that was proposed, we will begin by presenting a few notations, and then we will proceed to provide some examples of how they are used.

Using the notation of $A_{n\text{-}1:0}$, $B_{n\text{-}1:0}$, and $S_{n\text{-}1:0}$ for the adder two n-bit inputs and one n-bit output, as well as $S'_{n\text{-}1:0}$ for the n-bit intermediate estimated output before error reduction, we can explain the following. A_i, B_i, S_i, and S'_i denote all the i^{th} least significant bits in the ratio $A_{n\text{-}1:0}$, $B_{n\text{-}1:0}$, $S_{n\text{-}1:0}$, and $S'_{n\text{-}1:0}$, respectively.

2.1.1 Operation of the Proposed Adder:

The adder that has been described here is shown in Fig.1 in the form of a schematic that explains how it operates when given 16-bit inputs. To complete the task of adding n bits, two distinct types of additions are required.

We make use of a k-bit accurate adder, such as an RCA or CLA, to achieve an exact value for the k bits that are thought to be the most significant. This allows us to obtain an exact value for the *k* bits. On the other hand, we make use of OR and XOR to obtain an approximation for the *n* bits that are thought of as being the most significant to obtain an approximation for those bits, $k < n$.

A 16-bit input is divided into two parts, one of which is an exact 8 bits and the other of which is an approximate 8 bits, as shown in Fig.1. These two parts are then combined to form the entire input. It is vital that the bit widths of two components do not have to be the same for there to be compatibility between them. Accurate addition is achieved through the utilisation of k most significant bits (MSBs), and the carry-in signal (C_{in}) is forecasted through the utilisation of an AND operation of two $(n\text{-}k\text{-}1)^{th}$ inputs (i.e., $C_{in} = A_{n\text{-}k\text{-}1}$ AND $B_{n\text{-}k\text{-}1}$). Both operations are accomplished through the utilisation of an AND operation.

The approximation component, just like the LOA, uses the OR function to add the lower-order input bits of two operands. However, the bit operation of the $(n\text{-}k\text{-}1)^{th}$ input is substituted with the XOR operation in the proposed adder. This is because the OR function can only add bits that are in the same bit position. This is since the OR function is unable to add bits that are not arranged in the same order as the rest of the bits in the inputs.

This results in the creation of a half adder for the inputs that have the $(n\text{-}k\text{-}1)^{th}$ least significant bit; hence, the length of accurate addition is increased by one bit, and its accuracy is improved in compared to that of the LOA. To be more explicit, the OR gate is changed to an XOR gate at the $(n\text{-}k\text{-}1)^{th}$ least significant bit position. This is done so that more bits can be processed simultaneously. It is guaranteed that the bits in positions $(n\text{-}1)$ through $(n\text{-}k\text{-}1)$ provide the correct result.

The adder produces an output that is close to 01011010 in response to the input that is depicted in Fig.1, whereas the LOA produces an output that is 11011010. The error distance, which is defined as $|S_{appr}\text{-}S_{acc}|$, where S_{appr} and S_{acc} are the approximate and correct outputs, respectively, for the given input, decreases from 1110000 to 10000, which demonstrates that the output of the proposed adder is closer to the correct summation of 01101010 than the output of the LOA. This is the case because S_{appr} is the approximate output.

	Precise Part		**Approx Part**
	MSB		LSB
$A_{n\text{-}1:0}$	10110111	c_{in} 1	1010010
$B_{n\text{-}1:0}$	00101001	1	0011000
…	…	↓	…
$S_{n\text{-}1:0}$	11100001	0	1011010

Fig.1. Operation of Proposed Adder

2.2 PROPOSED HYBRID ERROR REDUCTION SCHEME

An error reduction is carried out by the proposed approximate adder in the event when the $(n\text{-}k\text{-}2)^{th}$ bits of both inputs are set to 1 to further reduce the output errors. This takes place when $A_{n\text{-}k\text{-}1} = 1$ and $B_{n\text{-}k\text{-}1} = 1$. If this is not the case, as shown in Figure 3, it does not carry out any error reduction. The fact that our adder includes logic for error reduction does not, however, result in any modifications to the outputs that are produced by the adder. To provide further clarity, the $(n\text{-}k\text{-}1)^{th}$ input determines the manner in which the proposed hybrid error reduction is applied to the data in order to get the desired results.

The adder that is being considered employs the $(n\text{-}k\text{-}1)^{th}$ bit of the output stream as the factor that decides which of the two distinct error correction strategies to use. The reason that $S_{n\text{-}k\text{-}1:n\text{-}k\text{-}2} = 10$ is because the reduction logic sets the $(n\text{-}k\text{-}1)^{th}$ and $(n\text{-}k\text{-}2)^{th}$ outputs to 1 and 0, respectively, if both of the inputs are 1 or 0. In the case that this does not take place, it will cause all of the switches beginning with $S_{n\text{-}k\text{-}3}$ and ending with S0 to show the value 1.

The 01011010 that is produced because of executing a standard addition on the sample inputs as the intermediate approximation output. This is because it is the result of the addition being performed on the sample inputs. When both of the $(n\text{-}k\text{-}2)^{th}$ input bits are 1, the adder proceeds to the $(n\text{-}k\text{-}1)^{th}$ output bit for another 1 check. This occurs when both of the $(n\text{-}k\text{-}2)^{th}$ input bits are 1. This takes place when both of the bits located at the $(n\text{-}k\text{-}2)^{th}$ input position are set to 1.

When an XOR operation is done at the $(n\text{-}k\text{-}1)^{th}$ bit location, the intermediate approximation output $S'_{n\text{-}k\text{-}1} = 0$ demonstrates that the associated input bits are identical. Additionally, the final adder outputs at the $(n\text{-}k\text{-}1)$ and $(n\text{-}k\text{-}2)$ bit positions are 1 and 0, respectively.

If the intermediate approximate output $S'_{n\text{-}k\text{-}1} = 1$, then all the other lower-order bits are also set to 1, which ultimately results in a value that is extremely close to 11111111. It is anticipated, given this input, that a carry will start in the $(n\text{-}k\text{-}2)^{th}$ LSB and make its way to the precise part via the $(n\text{-}k\text{-}1)^{th}$ LSB.

The carry prediction is produced by performing an AND operation with only the $(n\text{-}k\text{-}1)^{th}$ LSB inputs, as shown in Fig.1, the carry does not actually propagate to the precise adder ($C_{in} = 0$). This is because the AND operation only uses the $(n\text{-}k\text{-}1)^{th}$ LSB inputs. This consequence demonstrates that the actual sum will always be more than the increment that was intended for it, as can be observed.

Raising each of the approximation part outputs to 1 brings the approximate result closer to the correct sum. There is a chance that the error distance could be reduced by a factor.

2.3 IMPLEMENTATION OF THE PROPOSED ADDER

In order to generate the intermediate approximation outputs $S'_{n\text{-}k\text{-}1:0}$ in the process of approximate addition, the XOR gate for the $(n\text{-}k\text{-}1)^{th}$ LSB and the OR gates for the other lower-order LSBs are utilised. This is done in conjunction with each other. The outcome of the AND operation carried out on the least significant bit located at position $(n\text{-}k\text{-}2)^{th}$ will indicate whether error reduction should be carried out.

Calculating the final approximate outputs $S'_{n\text{-}k\text{-}1:0}$ involves using the intermediate approximate outputs $S'_{n\text{-}k\text{-}1:0}$ as inputs to INV, AND, OR, and NAND gates. This allows the final approximate outputs to be determined. This enables the calculation to be as accurate as it possibly can be. If neither of the $(n\text{-}k\text{-}2)^{th}$ inputs is 1, then the intermediate approximation outputs will bypass the reduction logic and go directly to the final results.

This occurs when neither of the inputs is 1. This happens if neither of the $(n\text{-}k\text{-}2)^{th}$ inputs is a 1. Compute the critical route delay, also referred to as tapproximate, of the approximation component by adding together the delays of the inverter, the two-input XOR gate, the NAND gate, and the AND gate. This is a straightforward method. This computation utilises the following formula:

$$t_{appr} = t_{INV} + t_{XOR} + t_{NAND} + t_{AND}, \quad (1)$$

3. DEEP GENETIC ALGORITHM

SC_1, SC_2,...SC_n are chromosomes that can be found in the search section of the GA module. They are positioned in this location because they have the potential to provide solutions to the generation difficulty. Each gene imparts a unique set of characteristics and capabilities upon the organism. The genes that are situated on each chromosome are denoted by the integers SC_iG_1 to SC_nG_5, where i refers to the chromosomal index and n refers to the total number of chromosomes that are present in the population.

Sigmoid gates are responsible for controlling the read and write operations that are performed by the memory cell. The LSTM network incorporates data from a wide variety of sources into its decision-making process at each stage. The present input, denoted by x_t, the previous hidden state of all LSTM units, denoted by h_{t-1}, and the previous memory cell state, denoted by c_{t-1}, are all examples of these sources. At each time step t, these gates have their inputs x_t, h_{t-1}, and c_{t-1} updated in the following manner:

$$it=\sigma(Wxixt+Whiht-1+bi) \quad (3)$$

$$ft=\sigma(Wxfxt+Whfht-1+bf) \quad (4)$$

$$ot=\sigma(Woixt+Whoht-1+bo) \quad (5)$$

$$gt=\phi(Wxcxt+Whcht-1+bc) \quad (6)$$

$$ct=ft\odot ct-1+it\odot gt \quad (7)$$

$$ht=ot\odot\phi(ct) \quad (8)$$

where

W - weight matrices,

b - bias vectors,

ϕ - hyperbolic tangent,

σ - sigmoid activation function and

$\odot$ - dot products.

$$\phi(x)=(\exp(x)-\exp(-x)).$$

$$\sigma(x)=1/(1+\exp(-x)).$$

Input gate $i(t)$, where t is a time value between 0 and 1, receives data that has been sigmoid modulated now that time value $t = 0$. At this stage, the forget gates of each LSTM unit have been calibrated to the value 0, as shown by $f(t)$, which stands for forget function at time. As the duration of each time step continues to increase, the forget gate will gradually start to store only the information that is of the importance. Additionally, it will initiate the process of removing data that is deemed to be irrelevant.

The states of the memory cells $c(t)$ and the output gate $o(t)$ gradually take in the valuable background information to develop a dense representation $h(t)$ of the output data. This is done to complete the process. The softmax function, together with its parameters Ws and bs, was applied to the hidden output, which may have one of K possible outcomes ($h_t=\{h_{tk}\}K_k=0$, $h_t \in R_K$), to offer a prediction for the next word. This was done so that the user could know what word would follow next.

$$F(p_{ti};W_s,b_s)=exp(W_{shti}+b_s)\sum exp(W_{shtj}+b_s) \quad (9)$$

where pti - estimated probability.

In this study, we utilised the same criteria to evaluate the various similarity measures that were available to us to facilitate comparisons between them. A random selection will be made among the chromosomes that will prove to be the most advantageous to the newly formed population. In the meantime, a new set of chromosomes that are being generated to offer to the requirements of the growing population is being produced.

$$F(x)=\max(Simsm(GTi)) \quad (10)$$

where

sm - similarity metrics and

GT_i - generated power value.

4. RESULTS AND DISCUSSION

The power consumption and latency of various implementations of the planned DGA, including MTCMOS, GDI, and GDI-MTCMOS are analysed. Utilising GDI logic results in a sizeable reduction in the total number of transistors that are required, as demonstrated in the Table.1-Table.3 for the various bit sizes.

When compared to the other possible architectures, the combination of GDI and MTCMOS produces the best results in terms of power consumption, performance, and area. This is the case regardless of which potential architectures are considered. The number of transistors that are necessary for the method that has been proposed is 65% smaller when compared to the utilisation of traditional CMOS logic.

Table.1. Power Prediction of DGA with n = 1:16 and k = 8

n	Area (μm^2)	Power (μW)	Delay (ns)
2	98.19	24.70	1.15
4	89.69	22.27	1.10
6	86.45	23.18	1.15
8	90.40	24.09	1.17
10	109.83	26.62	1.10
12	115.30	28.34	1.15
14	117.93	30.07	1.28
16	105.89	26.93	1.15

Table.1. Power Prediction of DGA with n = 1:16 and k = 8

n	EDP (fJ·s) ($\times 10^{-8}$)	PDP (fJ)	ER (%)	MRED	MED	NMED ($\times 10^{-3}$)
2	3.21	28.14	91.09	4.43	112.26	1.71
4	2.65	24.30	91.09	5.14	191.93	2.93
6	3.02	26.42	100.35	4.43	116.41	1.77
8	3.24	27.94	100.06	4.43	96.18	1.47

10	3.17	29.05	91.09	5.14	179.28	2.73
12	3.68	32.29	87.72	4.43	89.59	1.37
14	4.77	37.86	87.72	3.75	82.50	1.26
16	3.49	30.67	85.47	4.43	85.84	1.31

Table.3. Delay Prediction of DGA with n = 1:32 and k = 1:16

k	***n***							
	4	**8**	**12**	**16**	**20**	**24**	**28**	**32**
2	5.84	6.55	5.84	5.83	6.55	5.83	5.12	5.83
3	5.18	5.88	5.18	5.18	5.88	5.18	4.47	5.18
4	4.43	5.14	4.43	4.43	5.14	4.43	3.75	4.43
8	3.86	4.59	3.86	3.86	4.59	3.86	3.09	3.86
12	2.93	3.66	2.93	2.93	3.66	2.93	2.09	2.93
14	2.08	2.81	2.08	2.08	2.81	2.08	1.35	2.08
16	1.43	2.16	1.43	1.43	2.16	1.43	0.63	1.43

The Table.1-Table.3 employs the GDI methodology to generate RCA blocks and multiplexers, this method requires a noticeably smaller amount of physical space than the traditional CMOS design. The strategy that CSA took with parallel computing and the incorporation of GDI logic both contribute to the reduction of wait times to a significant degree.

MTCMOS logic is incorporated throughout the entirety of the design to turn it into a functioning product. Implementing a CSA using both GDI and MTCMOS techniques allows for the possibility of achieving the lowest possible power dissipation. This goal can be accomplished. The use of the MTCMOS approach for accomplishing this goal is becoming increasingly popular, and this is in line with that trend.

5. CONCLUSION

In this paper, DGA can achieve both a low amount of power consumption and a high level of performance. The power delay product (PDP) of the CSA is one of the variables that can be significantly improved by adopting the way that was proposed. It has been discovered that using MTCMOS and GDI together leads to a large reduction in leakage power as well as an improvement in latency. This is one of the benefits of using this combination. Chip density can be increased with the MTGDI method because this method cuts down on the total number of transistors required for the design.

REFERENCES

[1] Y. Shen and V. Chen, "Class-E Power Amplifiers Incorporating Fingerprint Augmentation with Combinatorial Security Primitives for Machine-Learningbased Authentication in 65 nm CMOS", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 69, No. 5, pp. 1896-1909, 2022.

[2] R.M. Weng, C.Y. Liu and P.C. Lin, "A Low-Power FullBand Low- Noise Amplifier for Ultra-Wideband Receivers", *IEEE Transactions Microwave Theory and Techniques*, Vol. 58, No. 8, pp. 2077-2083, 2010.

[3] P. Kumar and C.S. Thakur, "Hybrid Architecture based on Two-Dimensional Memristor Crossbar Array and CMOS Integrated Circuit for Edge Computing", *NPJ 2D Materials and Applications*, Vol. 6, No. 1, pp. 1-8, 2022.

[4] C. Xu, "SNS's not a Synthesizer: a Deep-Learning-based Synthesis Predictor", *Proceedings of Annual International Symposium on Computer Architecture*, pp. 847-859, 2022.

[5] Behzad Razavi, "*RF Microelectronics*", New York: Prentice Hall, 1998.

[6] V. Govindaraj and B. Arunadevi, "Machine Learning Based Power Estimation for CMOS VLSI Circuits", *Applied Artificial Intelligence*, Vol. 35, No. 13, pp. 1043-1055, 2021.

[7] S. Bavikadi and S.M. Pudukotai Dinakarrao, "A Review of In-Memory Computing Architectures for Machine Learning Applications", *Proceedings of International Symposium on Great Lakes on VLSI*, pp. 89-94, 2020.

[8] Paul R. Gray, et al., "*Analysis and Design of Analog Integrated Circuits*", Wiley, 1993.

[9] A. Keshavarzi, C. F. Hawkins, K. Roy and V. De, "Effectiveness of reverse body bias for low power CMOS circuits," *Proceedings of NASA Symposium on VLSI Design*, pp. 2.3.1–2.3.9, 1999.

[10] Kyung Ki Kim and Yong-Bin Kim, "A Novel Adaptive Design Methodology for Minimum Leakage Power Considering PVT Variations on Nanoscale VLSI Systems" *IEEE Transactions on Very Large Scale Integrated Systems*, Vol. 17, No. 4, pp. 517-528, 2009.

[11] Cassondra Neau and Kaushik Roy, "Optimal Body Bias Selection for Leakage Improvement and Process Compensation Over Different Technology Generations", *IEEE International Symposium on Low –Power Electronics and Design*, pp.116-121, 2003.