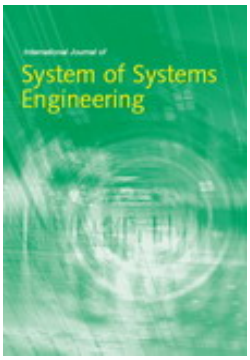


International Journal of System of Systems Engineering

ISSN online: 1748-068X - ISSN print: 1748-0671
https://www.inderscience.com/ijsse

Image analysis in healthcare systems using approximate multi-bit adders

M. Priyadharshni, Kishore Sanapala, Polinpapilinho F. Katina, K. Prasanna Kumar

DOI: 10.1504/IJSSE.2023.10053586

Article History:

Received:	21 February 2022
Last revised:	19 April 2022
Accepted:	21 April 2022
Published online:	16 February 2023

Image analysis in healthcare systems using approximate multi-bit adders

M. Priyadharshni

Department of Electronics and Communication Engineering,
Panimalar Engineering College,
Poonamalle, Chennai – 600123, India
Email: elangopriya43@gmail.com

Kishore Sanapala

Department of Electronics and Communication Engineering,
Marri Laxman Reddy Institute of Technology and Management,
Hyderabad-500043, India
Email: kishore.technova@gmail.com

Polinpapilinho F. Katina*

Department of Informatics and Engineering Systems,
University of South Carolina Upstate,
800 University Way,
Spartanburg, SC, 29303, Spartanburg
Email:pkatina@uscupstate.edu
*Corresponding author

K. Prasanna Kumar

Department of Electrical and Computer Engineering,
St. Peter's Engineering College,
Jawaharlal Nehru Technological University,
Hyderabad, Telangana, 500043, India
Email: prasannakumar458@gmail.com

Abstract: The expenses involved in designing computing systems, especially for the medical services framework, are a significant concern with technological advancements. However, the high cost of administrative waste tends to increase the complexity of digital system design. This paper suggests that high administrative costs can be reduced by reducing the complexity of the arithmetic systems used in the design. The major complexity of the arithmetic systems can be reduced by using approximate adders. In this paper, we have investigated approximate computing in full adders at the hardware phase. A hybrid multi-bit approximate adder (HMBAA) for multiple degrees of approximation is developed using approximate full adders. Hardware, error, and image evaluations for HMBAA are carried out in Synopsys using a 65 nm CMOS standard cell library, and MATLAB is used for error analysis.

When compared to the precise adder, the simulation results of the proposed design showed significant improvements and achieved 62%, 23%, and 74% savings in terms of area, delay, and power consumption, respectively.

Keywords: approximate full adders; ripple carry adders; image processing; error analysis; approximate computing; approximate multi-bit adders; image sharpening; structural analysis; hardware; error-tolerant applications.

Reference to this paper should be made as follows: Priyadharshni, M., Sanapala, K., Katina, P.F. and Kumar, K.P. (2023) 'Image analysis in healthcare systems using approximate multi-bit adders', *Int. J. System of Systems Engineering*, Vol. 13, No. 1, pp.30–49.

Biographical notes: M. Priyadharshni is an Assistant Professor in the Department of ECe at Panimalar Engineering College, (Chennai, India). She holds a BE from Anna University (Chennai, India), MTech from Veltech Dr. RR & Dr. SR Technical University (Chennai, India), and a PhD in Approximate Computing from Vellore Institute of Technology (Vellore, India). Her teaching and research area includes approximate computing, digital IC VLSI, low power VLSI, and image processing.

Kishore Sanapala is an Associate Professor in the ECE Department at Marri Laxman Reddy Institute of Technology and Management (Hyderabad, India). He holds a PhD in Electronics Engineering from VIT University (Vellore, India). He previously worked as an Assistant Professor and Research Associate with more than 8 years of experience in academia and industry. His teaching and research areas include subthreshold digital circuit design, energy-efficient memory design, and near-zero computational circuits for IoT devices, A.I. & Machine Learning. His profile includes more than 20 peer-reviewed journal papers and conference proceedings.

Polinpapilinho F. Katina is an Assistant Professor in the Department of Informatics and Engineering Systems at the University of South Carolina Upstate (Spartanburg, South Carolina). He has served in various capacities at the National Centers for System of Systems Engineering (Norfolk, Virginia) and Politecnico di Milano (Milan, Italy). He holds a BS in Engineering Technology, ME in Systems Engineering, and PhD in Engineering Management and Systems Engineering (Old Dominion University, Norfolk, Virginia). His teaching and research areas include complex system governance, emerging technologies (e.g., IoT), infranomics, and manufacturing systems. His profile includes more than 150 peer-reviewed journal papers, conference proceedings, and books.

K. Prasanna Kumar is the Head of the ECE Department at St. Peter's Engineering College (Hyderabad, India). He obtained his BTech in ECE from MVGR College of Engineering (Andhra Pradesh, India), MTech from Pydah College of Engineering and Technology (Andhra Pradesh, India), and a PhD from Koneru Lakshmaiah Education Foundation (Vijayawada, India). He previously worked at Koneru Lakshmaiah Education Foundation and Lendi Institute of Engineering and Technology as an Associate Professor. He has published several papers in reputable journals and participated in various conferences. He has over 13 years of teaching experience and handles several academic and administrative duties.

1 Introduction

In digital signal processing, the major complexity of the systems depends on arithmetic units such as adders, subtractors, and multipliers. Reduced complexity of the systems can be accomplished by reducing the hardware of the adders (Sanapala et al., 2021). It can be achieved by adopting approximation in the adders (Premson and Sakthivel, 2021). The major advantage of approximate computing is to design high-speed digital blocks for error-resilient applications with less complexity but with an inaccuracy trade-off. Approximate hardware or approximate software is used to perform simple computations in error-tolerant blocks of signal and image processing applications. Binary arithmetic plays a significant role in processing binary signals. As the adders are considered to be the basic data operator, it has grabbed the attention of many researchers.

Many adder blocks, at gate/circuit level, were implemented in the literature (Jiang et al., 2015; Mittal, 2016). The static CMOS is the most conventional design, but it occupies more area due to the huge transistor count (Weste and Harris, 2010). Complex pass transistor logic is the other conventional logic used for digital block design but suffers from poor logic swing and large area due to 32 transistors for 1-bit adder design (Zhang et al., 2003). Many hybrid designs at the circuit level were proposed that used the combination of the CMOS and pass transistors to overcome the drawbacks in terms of area complexity and energy consumption (Kishore and Sakthivel, 2019). This work addresses the issues of area and propagation delay of the adder blocks at the gate level using approximate computing for achieving better performance metrics.

The design of approximate adders mainly focuses on minimising the carry propagation lifetime and reducing the complexity of the circuit. The carry propagation lifetime is always based on the critical path length. In this paper, a set of approximate full adders is taken into account. The evaluations of these approximate full adders are carried out in terms of hardware and error. By utilising these approximate full adders, a few hybrid multi-bit approximate adders (HMBAAs) are proposed for the bit width $N = 20$ and varying width length at different levels of approximation. The proposed HMBAA are developed with approximate and exact adders for evaluation in terms of hardware and error metrics. The motivation behind this work is to estimate the tolerance range of approximation in the multi-bit adder. Further, the evaluation is extended to the image processing application. The hardware metrics considered for evaluating the approximate adders are power, area, gate count and delay (Townsend et al., 2003). For a better realistic estimation, the delay is evaluated through power delay product (PDP) and area delay product (ADP) metrics. In case of error estimation, metrics (Naseer et al., 2015) such as normalised error distance (NED), mean error distance, and error and pass rate is considered.

2 Existing methodologies

In redesigning the full adder, two methodologies are adopted, the Gate level and the transistor level approximation.

2.1 Gate level approximation.

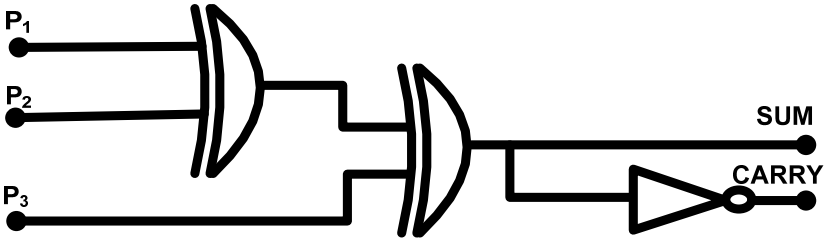
2.1.1 Inexact full adders (INXA)

Inexact full adders are developed with approximation in gate level with the minimal number of transistors. These three full adders (Almurib et al., 2016) are implemented with either approximating sum or carry.

INXA$_1$: In INXA$_1$ method, the approximation is performed only on the carry generation, where 'sum' remains exact. Figure 1 and equation (1) is a representation of the INXA$_1$ (Almurib et al., 2016) implementation. Table 1 shows the truth table (col 6 and 7). From the table, it is noted that the INXA$_1$ gives an accurate sum and two inaccurate in the carry.

$$\begin{aligned} \text{Sum} &= P_1 \oplus P_2 \oplus P_3 \\ \text{Carry} &= \overline{\text{Sum}} \end{aligned} \tag{1}$$

Figure 1 Logical expression of INXA$_1$



Source: Almurib et al. (2016)

Implementing the 1-bit INXA$_1$ needs 1 NOT gate and 2 XOR gates. For making a clear comparison, assume that one gate count is equal to one basic gate (Townsend et al., 2003), where the total number of gate count determines the adder area. Two 2-input gates (OR, AND) along with one 1-input NOT gate are used to realise the XOR gate (Oklobdzija, 2001). Then INXA1 is implemented with the four AND gates, two OR gates and two NOT gates. Therefore, nine basic gates are used to realise INXA$_1$.

INXA$_2$: The INXA$_2$ (Almurib et al., 2016) is achieved by approximating the sum. On the other hand, the carry is maintained as accurate. Here, the other XOR gate of non-approximate full adder is restored by the OR gate. Figure 2 and equation (2) is the representation of gate level implementation of INXA$_2$. Table 1 shows the truth table for the same (col 8 and 9), the carry maintains accurate results, and the sum consists of two errors among the 8 cases. For implementing the 1-bit INXA$_2$, one XOR gate, 5 two AND gates and 2 OR gates are required. If the XOR gate is restricted to two basic input gates, INXA$_2$ is developed by considering four AND gates, one NOT gate and three OR gates. 8 basic gates are utilised in INXA$_2$.

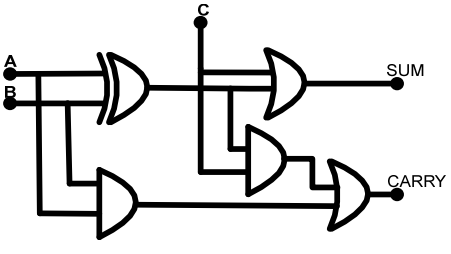
$$Sum = (P_1 \oplus P_2) + P_3$$

$$\text{Carry} = (P_1 \oplus P_2)P_3 + P_1P_2 \tag{2}$$

Table 1 Approximate full adders truth table

INPUTS			*OUTPUTS*		*INXA1*		*INXA2*		*INXA3*		*AFA1*		*AFA2*		*AFA3*		*AFA4*	
P1	*P2*	*P3*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*
0	0	0	0	0	0	0	0	0	**1×**	0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	**1×**	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	**1×**	1	0	1	**1×**
0	1	1	0	1	0	1	**1×**	1	0	1	0	**0×**	0	1	0	**0×**	0	1
1	0	0	1	0	1	0	1	0	1	0	1	**1×**	1	**0×**	1	0	1	**1×**
1	0	1	0	1	0	1	**1×**	1	0	1	0	1	0	**0×**	0	**0×**	0	1
1	1	0	0	1	0	**0×**	0	1	0	1	0	1	0	1	0	1	0	1

Figure 2 Logical expression of $INXA_2$



Source: Almurib et al. (2016)

$INXA_3$: In $INXA_3$ (Almurib et al., 2016), the carry remains exact whereas, the sum is approximated. Out of 8 combinations of sum generation, the $INXA_3$ generates only a couple of errors. Also, Table 1 (col 10 and 11) shows the truth table for the same. INXA3 is represented in Figure 3 and equation (3).

For implementing $INXA_3$, it requires two AND gates, one OR gate, one XOR gate and one NOT gate. The design of $INAX_3$ is implemented using one NOT gate, and with one XOR gate and also with two AND gate and one OR gate. Totally 8 basic gates are used in the implementation of $INXA_3$.

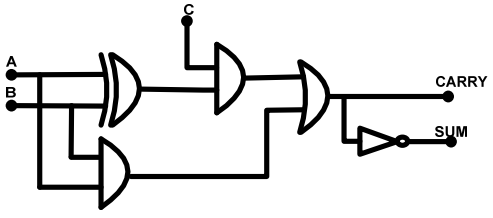
$$\text{Carry} = (P_1 \oplus P_2)P_3 + P_1P_2$$

$$Sum = \overline{Carry} \tag{3}$$

2.1.2 Approximate full adders (AFA)

The motivation behind AFA (Dutt et al., 2018) is to make the carry output independent of any generation circuit and to minimise the lifetime of carry, when it is realised in the multi-bit adders. In AFA's the approximation approach is applied only to the carry.

Figure 3 Logical expression of $INXA_3$



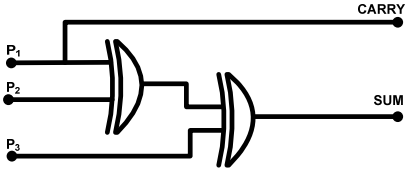
Source: Almurib et al. (2016)

AFA_1 and AFA_2: The carry output is given as one of the inputs in the AFA_1. Similarly, another input is taken as the output to carry in AFA2. AFA_1 and AFA_2 in the carry generates two errors among 8 cases. Figures 4 and 5 is the representation of gate level implementation of AFA_1 and AFA_2. The truth table for the same is represented in Table 1.

$$\text{Sum} = P_1 \oplus P_2 \oplus P_3$$
$$Carry = P_1 \tag{4}$$

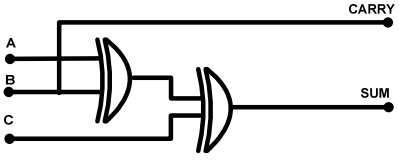
$$\text{Sum} = P_1 \oplus P_2 \oplus P_3$$
$$Carry = P_2 \tag{5}$$

Figure 4 Logical expression of AFA_1



Source: Dutt et al. (2018)

Figure 5 Logical expression of AFA_2



Source: Dutt et al. (2018)

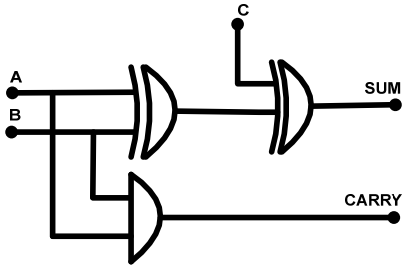
AFA_1 and AFA_2 are both realised with the help of two XOR gates. The total number of basic gates involved in the implementation of ofAFA_1 and $AFA_2 = 8$.

AFA_3: The carry output is considered as $P_1.P_2$ in the AFA_3 implementation. Table 1 shows the truth table of the AFA_3 (column 16 and 17) and Figure 6 shows the gate level implementation of the same.

The AFA_3 is implemented with the help of five AND gates, two NOT gates and two OR gates. Totally 9 basic gates are used to realise AFA_3.

$$\text{Sum} = P_1 \oplus P_2 \oplus P_3$$
$$\text{Carry} = P_1 P_2 \tag{6}$$

Figure 6 Logical expression of AFA_3

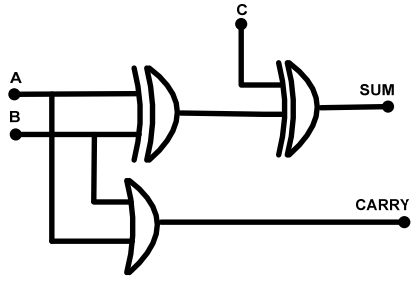


Source: Dutt et al. (2018)

AFA_4 The carry output in AFA_4 is considered as the summation of P_1 and P_2. Figure 7 is the representation of gate-level implementation of AFA_4. Table 1 shows the truth table for the same (column 18 and 19). The logical equation of AFA_4 is shown in equation (7). For implementing the AFA_4, one OR gate and two XOR gates are used. Totally 9 basic gates are utilised to realise AFA_4.

$$\text{Sum} = P_1 \oplus P_2 \oplus P_3$$
$$\text{Carry} = P_1 + P_2 \tag{7}$$

Figure 7 Logical expression of AFA_4



Source: Dutt et al. (2018)

2.1.3 Multiplexer based approximate full adder (MBAFA)

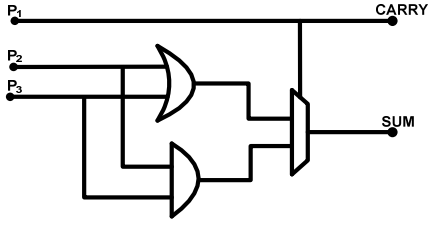
To reduce the gate count, Multiplexer based approximate full adder (MBAFA) (Jothin and Vasanthanayaki, 2018) is realised by using the pre-computation technique on the sum. Carry is designed as the fastest which is used to achieve high speed in multi-bit adders as shown in Figure 8. Table 2 shows the truth table of MBAFA (col 6 and 7). The logical expression of the MBAFA is shown below. The total gate count used for implementing MBAFA is 6 gates.

$$w = P_2 + P_3 \;\; v = P_2.P_3$$

$$\text{Sum} = \overline{P_1}.w + P_1.v$$

$$\text{Carry} = P_1 \tag{8}$$

Figure 8 Logical expression of MBAFA



Source: Jothin and Vasanthanayaki (2018)

Table 2 Approximate full adders truth table

INPUTS			*OUTPUTS*		*MBAFA*		*FTFA*	
P1	*P2*	*P3*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*
0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	**0×**	**1×**
0	1	0	1	0	1	0	1	0
0	1	1	0	1	**1×**	**0×**	0	1
1	0	0	1	0	**0×**	**1×**	1	0
1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	**0×**
1	1	1	1	1	1	1	**0×**	1

2.1.4 Fault-tolerant full adder

Fault tolerance full adder (FTFA) (Palanisamy et al., 2019) is designed by introducing two errors in the sum and two errors in the carry as shown in Figure 9. Carry of FTFA is equal to one of the inputs of the full adder. The total gate count utilised for implementing FTFA is 6 gates. The logical expression and truth table of FTFA are given below.

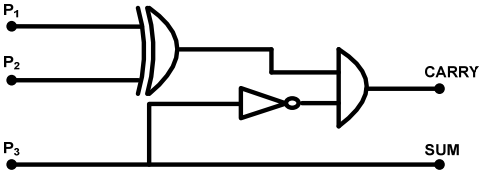
$$\text{Sum} = ((P_1 \oplus P_2)\overline{P_3})$$

$$\text{Carry} = P_1 \tag{9}$$

2.2 Transistor level approximation

From the exact full adder transistor level realisation, few transistors are removed to minimise the node capacitance, circuit complexity, and dynamic power dissipation. In this paper all the approximate full adders that have fewer errors available in the literature are designed at the gate level.

Figure 9 Logical expression of FTFA



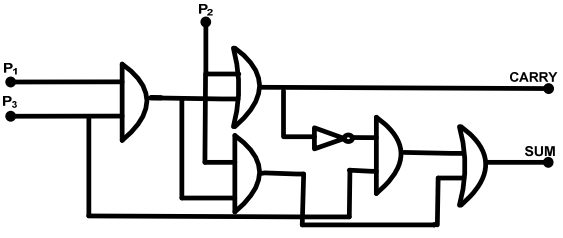
Source: Palanisamy et al. (2019)

2.2.1 Approximate mirror adders (AMA)

A Mirror Adder (MA) (Gupta et al., 2013) which is a combination of various approximate adders is formed by extracting few transistors from the depiction of the full adder. If some of the transistors are removed from the MA, there would be shorter delay by which faster discharging and charging of node capacitance is attained.

AMA$_1$: It is implemented with the help of eight transistors resulting in removing a few transistors from the conventional MA. However, there should not be any open or short circuit after the extracting process in the AMA$_1$. The implementation of AMA$_1$ (Gupta et al., 2013) results in two errors in sum and one error in carry. Three AND gates, two OR gates and one NOT gate are used to implement AMA$_1$. Totally 6 basic gates are used in AMA$_1$. Figure 10 shows the representation of AMA$_1$.

Figure 10 Logical expression of AMA$_1$



Source: Gupta et al. (2013)

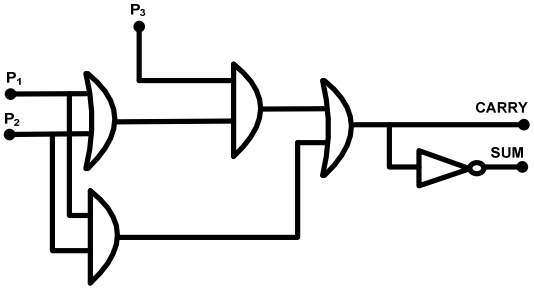
AMA$_2$: In the implementation of AMA$_2$ (Gupta et al., 2013), the carry is calculated first and then the sum is derived by inverting a carry. The outcome of the sum and the carry complimentary is found to be equal for six accurate instances among eight and two error instances. The output of the carry remains accurate. Table 3 shows the truth table for AMA$_2$ and Figure 11 shows the representation of the AMA$_2$.

The AMA$_2$ gate count is attained with the help of two OR gates, two AND gates and one NOT gate. 5 basic gates are utilised in AMA$_2$.

2.2.2 Approximate XNOR/XOR based adder (AXA)

AXA (Yang et al., 2013) is constructed with the help of an approximation of XOR/XNOR as shown in Figure 12. When compared with the exact full adders, AXA provides minimal transistor count and reduced power dissipation. However, there is a trade-off accuracy.

Figure 11 Logical expression of AMA_2



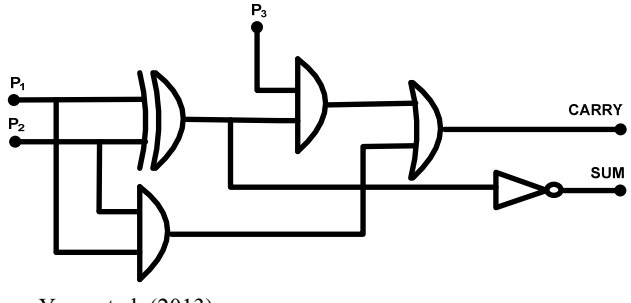
Source: Gupta et al. (2013)

AXA_1: The combinations of XNOR and pass transistor logic (PTL) are used in the realisation of AXA_1. The sum of AXA_1 is equal to XNOR, which results in four errors among eight. Whereas the result of XNOR gate is passed to the PTL, the result of PTL is equal to the output of the carry signal. For the implementation of AXA_1, two AND gates, one NOT gate and one OR gate are used. 8 basic gates are utilised in the realisation of AXA_1. The truth table for the same is shown in Table 3.

Table 3 Approximate full adders truth table

INPUTS			*OUTPUTS*		*AMA_1*		*AMA_2*		*AXA_1*		*AXA_2*	
P1	*P2*	*P3*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*	*Sum*	*Carry*
0	**0**	**0**	**0**	**0**	**0**	**0**	**1×**	**0**	**1×**	**0**	**0**	**0**
0	0	1	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	**0×**	**1×**	1	0	**0×**	0	**0×**	0
0	1	1	0	1	0	1	0	1	0	1	0	1
1	0	0	1	0	**0×**	0	1	0	**0×**	0	**0×**	0
1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1	**1×**	1	0	1
1	1	1	1	1	1	1	**0×**	1	1	1	1	1

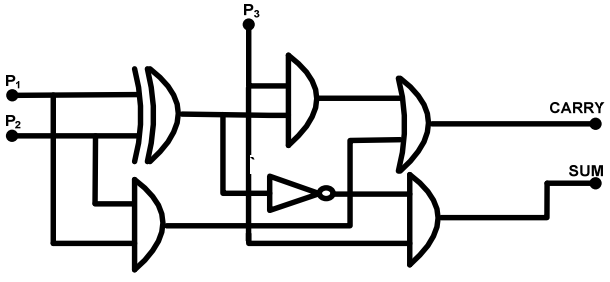
Figure 12 Logical expression of AXA_1



Source: Yang et al. (2013)

AXA_2: The construction of AXA_2 (Yang et al., 2013) is an add-on to AXA_1 implementation. This realisation leads to reduced errors in sum for which PTL configuration is utilised. The output provides two errors among 8 instances in the sum and the carry is maintained accurate. Implementation of the AXA_2 requires 9 basic gates (Figure 13 shows the representation of AXA_2 and Table 3 shows the truth table for the same).

Figure 13 Logical expression of AXA_2



Source: Yang et al. (2013)

3 Proposed hybrid multi-bit approximate adders

The primary design goal of the proposed design is to reduce carry propagation lifetime and circuit complexity. When the number of adders inputs is increased, the carry propagation delay increases, resulting in higher critical path time in multi-bit adders due to the long carry propagation paths. Among the multi-bit adders available in the literature, the Ripple carry adder remains the simplest adder in their interconnects but has a high carry propagation delay. The carry originates from the least significant bit (LSB) and propagates to the most significant bit (MSB) in the worst-case scenario of Ripple carry adder. The worst-case delay in the Ripple carry adder must be considered due to this carry rippling process. This remains a driving force behind the HMBAA.

The Ripple carry adder is divided into two blocks for LSB and MSB. LSB is constructed using an approximate adder, whereas the MSB remains exact. So the lifetime of carrying propagation is reduced by half. This paper proposes a few hybrid variants of 'Ripple Carry Adder (RCA)' using existing approximate full adders $n = 20$ (referred in section 2) with multiple levels of approximation. The HMBAA adder is developed for $N = 20$, 12 : 8, 8 : 12, and 10 : 10. For instance, in 12 : 8 HMBAA (HMBAA), the first 12 MSB adders are constructed with accurate adders and rest 8 are by approximate adders. Similarly, the adder is developed for other levels of approximation.

4 Comparative analysis of proposed ripple carry adders with approximate full adders.

Ripple carry adder with different degrees of approximation are validated according to their error and hardware.

4.1 Hardware evaluation

The adders are designed using Verilog HDL and the simulation is done in verilog compiler simulator (VCS). Using the Synopsys design compiler (DC) using pre-layout CMOS standard cell library of 65 nm, the synthesised results of all the adders are obtained as shown in Table 4. For elaborative power analysis for these adders are carried out in DC using 1,00,000 random test stimulus vectors, which is stored in .saif. The delay, power, and area retrieved from the DC are used to perform a comparative analysis on the approximate adders. Further, the comparative analysis includes the area, delay product (ADP), power, delay product (PDP), and area, power product (APP).

$INXA_1$ achieves a reduction of 26% and 22% in area and power for the exact full adder. Similarly, $INXA_2$achieves the good delay value with a reasonable reduction in area and power. 38% and 47% reduction of area and power is achieved by $INXA_2$ with respect to the exact full adder. $INXA_3$ attains good results when compared to other approximate full adders. $INXA_3$ achieves 63% and 47% reduction in area and power with respect to the exact full adder.

The hardware achieved by the approximate full adders i.e., AFA_1 and AFA_3 are similar to the hardware gained byAFA_2 and AFA_4. The approximate full adders which are considered for the exploration achieves the reduction in area vary in the range from 19% to 66% and the reduction in power percentage varies with in the range from 10% to 53%.

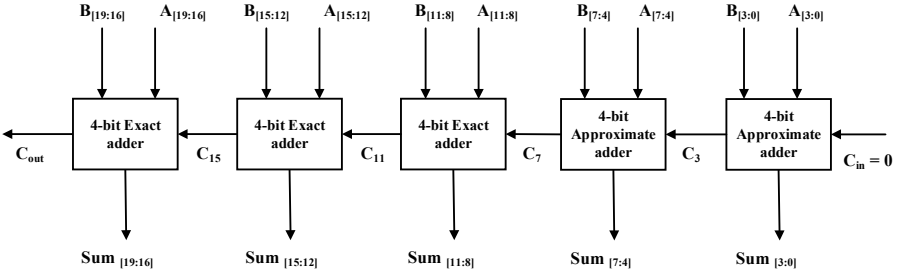
Table 4 Hardware evaluation of existing approximate full adders

Adders	*AREA* (μm^2)	*DELAY* (ns)	*POWER* (μW)
Exact full adder	36	0.13	0.086
INXA1 (Almurib et al., 2016)	27.6	0.14	0.0665
INXA2 (Almurib et al., 2016)	22	0.1	0.0455
INXA3 (Almurib et al., 2016)	13.2	0.1	0.033
AFA1 (Dutt et al., 2018)	26.4	0.13	0.069
AFA2 (Dutt et al., 2018)	26.4	0.13	0.069
AFA3 (Dutt et al., 2018)	32.4	0.13	0.0738
AFA4 (Dutt et al., 2018)	32.4	0.13	0.0742
MBAFA (Jothin and Vasanthanayaki, 2018)	13.6	0.1	0.022
FTFA (Palanisamy et al., 2019)	23.2	0.09	0.0498
AMA1 (Gupta et al., 2013)	20	0.09	0.0388
AMA2 (Gupta et al., 2013)	16.8	0.1	0.0287
AXA1 (Yang et al., 2013)	19.2	0.1	0.0334
AXA2 (Yang et al., 2013)	21.6	0.1	0.0402

Among these adders, the $INXA_3$ achieves the significant reduction in area, power and delay values when compared to other approximate full adders. MBAFA achieves the reduction of 62%, 23%, 74% in area, delay and power respectively. Similarly, FTFA shows the reduction of 35% in area, 30% in delay and 63% in power with respect to the exact full adder.

FTFA and AMA_1 achieve the low delay value among the existing approximate full adders. Figure 14 shows the comparison of approximate full adders in terms of ADP, PDP and APP values.

Figure 14 Block diagram of proposed hybrid multi-bit approximate adder for 12 : 8 with 12 MSB bits with an exact adder and the remaining 8 LSB bits with an approximate adder



Further, Table 5 elaborates on the proposed HMBAA with approximation at various levels. The more approximation in the adder results in less hardware with the sacrifices in their accuracy. At all the levels of approximation, MBAFA shows the best results. The AFA circuit is designed with the carry-free circuit (i.e., no circuit for carrying generation) because of that it achieves the shortest critical path in the multi-bit adders. From Figure 14, shows that AFA and MBAFA show a reasonable balance in their performance of ADP, PDP, and APP.

4.2 Error evaluation

The error evaluations (Naseer et al., 2015) of the approximate adders are based on mean error distance (MED), normalised error distance (NED), error rate (Wu et al., 2018) and pass rate (Liang et al., 2013). The error distance (ED), MED and NED are calculated as per the below expressions.

$$ED = R_{app} - R_{exact}$$

$$MED = \overline{ED}$$

$$NED = \frac{NED}{n}$$

where, n is the magnitude of the maximum output of an exact adder.

For performing the detailed error analysis of 1, 00,000 random test stimulus vectors are produced and recorded. This is done in MATLAB. The NED values for the Ripple Carry are shown in Table 7. Pass rate is defined as the number of exact outcomes to the total number of outcomes in an adder. Similarly, the error rate is the number of inexact outcomes to the total number of outcomes in an adder. Figure 15 is the representations of pass and error rate of all the adders.

Table 5 Hardware evaluation of proposed hybrid multi-bit approximate adder using approximate adders

Proposed hybrid multi-bit approximate adders	*Approximation at different levels for N = 20*	*Delay* (ns)	*Area* (μm^2)	*Power* (μw)
Ripple carry adder – Exact	20	0.19	623.2	0.3179
	10 : 10	0.19	623.2	0.3179
	8 : 12	0.19	623.2	0.3179
HMBAA-INXA_1 (Almurib et al., 2016)	12 : 8	0.19	623.2	0.3179
	20	0.3	383.2	0.202
	10 : 10	0.3	400.8	0.1782
	8 : 12	0.3	402.8	0.1696
HMBAA-INXA_2 (Almurib et al., 2016)	12 : 8	0.3	402.8	0.1874
	20	0.2	375.2	0.1804
	10 : 10	0.2	436	0.1976
	8 : 12	0.22	445.2	0.1947
HMBAA-INXA_3 (Almurib et al., 2016)	12 : 8	0.21	420.4	0.1901
	20	0.16	420.4	0.2106
	10 : 10	0.2	392.8	0.1752
	8 : 12	0.21	425.2	0.1787
HMBAA-AFA_1(Dutt et al., 2018)	12 : 8	0.2	386.8	0.1842
	20	0.09	151.2	0.0732
	10 : 10	0.15	334.4	0.1324
	8 : 12	0.17	321.2	0.1351
HMBAA-AFA_2 (Dutt et al., 2018)	12 : 8	0.13	383.2	0.1769
	20	0.09	151.2	0.0736
	10 : 10	0.15	334.4	0.135
	8 : 12	0.17	321.2	0.1352
HMBAA-AFA_3 (Dutt et al., 2018)	12 : 8	0.13	378	0.1851
	20	0.09	199.2	0.0096
	10 : 10	0.15	385.2	0.175
	8 : 12	0.17	370	0.1658
HMBAA-AFA_4 (Dutt et al., 2018)	12 : 8	0.13	402.8	0.1931
	20	0.1	210.8	0.0785
	10 : 10	0.15	400.8	0.1895
	8 : 12	0.18	361.6	0.1583

Table 5 Hardware evaluation of proposed hybrid multi-bit approximate adder using approximate adders (continued)

Proposed hybrid multi-bit approximate adders	*Approximation at different levels for N = 20*	*Delay* (ns)	*Area* (μm^2)	*Power* (μw)
HMBAA-MBAFA (Jothin and Vasanthanayaki, 2018)	12 : 8	0.13	393.6	0.1937
	20	0.08	193.6	0.0129
	10 : 10	0.25	332	0.0281
	8 : 12	0.2	283.6	0.0213
HMBAA-FTFA (Palanisamy et al., 2019)	12 : 8	0.3	344.8	0.0315
	20	0.08	112	0.0103
	10 : 10	0.25	283.6	0.0262
	8 : 12	0.2	226.4	0.0202
HMBAA- AMA_1 (Gupta et al., 2013)	12 : 8	0.3	305.2	0.0299
	20	0.16	317.6	0.1285
	10 : 10	0.18	430.8	0.1979
	8 : 12	0.21	425.6	0.1817
HMBAA- AMA_2 (Gupta et al., 2013)	12 : 8	0.18	436	0.1996
	20	0.17	361.6	0.1847
	10 : 10	0.2	392.8	0.1752
	8 : 12	0.21	425.2	0.1787
HMBAA- AXA_1 (Yang et al., 2013)	12 : 8	0.2	386.8	0.1841
	20	0.18	183.6	0.0645
	10 : 10	0.2	409.2	0.1813
	8 : 12	0.21	348	0.1464
HMBAA- AXA_2 (Yang et al., 2013)	12 : 8	0.18	469.6	0.2264
	20	0.17	442	0.2079
	10 : 10	0.2	455.6	0.2062
	8 : 12	0.2	462.4	0.1932
	12 : 8	0.2	418.8	0.1869

Even though the AMA_1 achieves a lesser NED value when compared to other approximate adders, it fails to meet the hardware requirements. An AFA_1 and AFA_2 adder achieves the trade-off between hardware and error.

5 Image application

The median filtering is broadly used in image processing applications. The filter is used to remove the noises by preserving the edges of an image in the pre-processing stages of an algorithm e.g., Edge detection. The median filter will replace each pixel value with the median value of the neighbouring pixels.

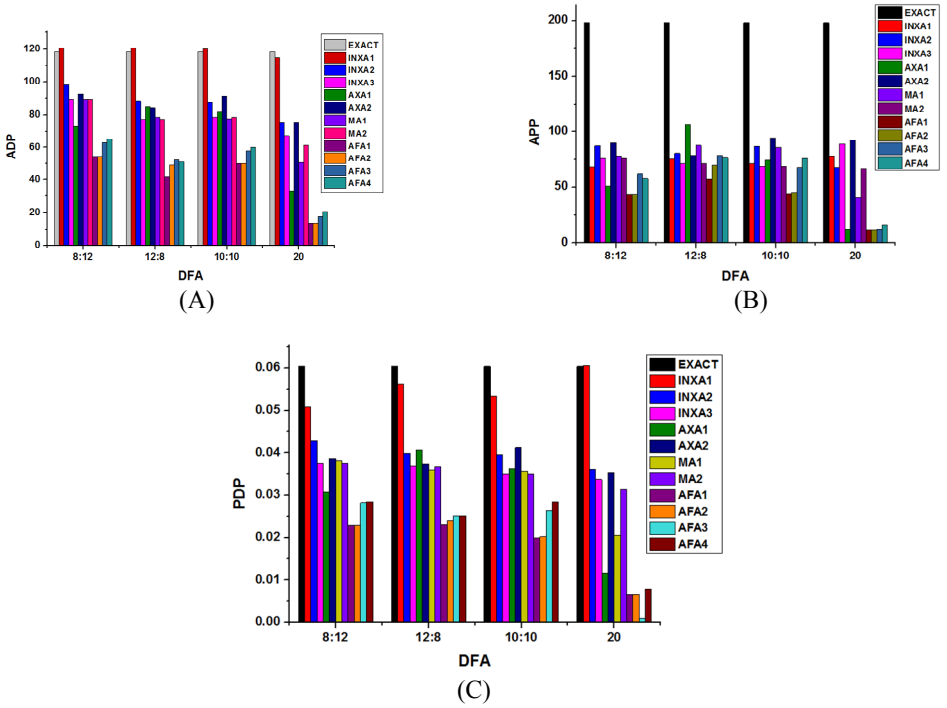
Table 6 NED measures of proposed hybrid multi-bit approximate adders

Proposed hybrid multi-bit approximate adders	*N = 12 : 8*	*N = 10 : 10*	*N = 8 : 12*	*N = 20*
HMBAA-INXA1 (Almurib et al., 2016)	4.24×10^{-4}	6.17×10^{-2}	1.8×10^{-3}	4.6×10^{-1}
HMBAA-INXA2 (Almurib et al., 2016)	4.104×10^{-2}	8.336×10^{-1}	5.68×10^{-2}	1.22×10^{-1}
HMBAA-INXA3 (Almurib et al., 2016)	3.44×10^{-4}	4.5×10^{-2}	1.6×10^{-3}	4.678×10^{-1}
HMBAA-AFA1 (Dutt et al., 2018)	4.21×10^{-3}	2.209×10^{-2}	2.209×10^{-2}	2.209×10^{-2}
HMBAA-AFA2 (Dutt et al., 2018)	4.21×10^{-3}	2.209×10^{-2}	2.209×10^{-2}	3.15×10^{-2}
HMBAA-AFA3 (Dutt et al., 2018)	2.23×10^{-3}	2.209×10^{-2}	4.128×10^{-2}	2.235×10^{-2}
HMBAA-AFA4 (Dutt et al., 2018)	6.689×10^{-2}	6.689×10^{-2}	4.128×10^{-2}	3.15×10^{-2}
HMBAA-MBAFA (Jothin et al., 2018)	1.8×10^{-3}	2.7×10^{-3}	1.09×10^{-2}	1.75×10^{-1}
HMBAA-FTFA (Palanisamy al., 2019)	1.8×10^{-3}	7.2×10^{-3}	2.4×10^{-2}	4.125
HMBAA-AMA1 (Gupta et al., 2013)	4.2×10^{-4}	2.97×10^{-2}	1.633×10^{-1}	1.57×10^{-2}
HMBAA-AMA2 (Gupta et al., 2013)	3.45×10^{-4}	4.497×10^{-2}	1.6×10^{-3}	4.678×10^{-1}
HMBAA-AXA1 (Yang et al., 2013)	2.27×10^{-4}	7.05×10^{-3}	1.34×10^{-3}	4.371×10^{-1}
HMBAA-AXA2 (Yang et al., 2013)	4.104×10^{-2}	5.56×10^{-2}	1.6×10^{-3}	3.07×10^{-2}

Table 7 PSNR and SSIM values obtained from image application using HMBAA with 12 : 8

Proposed hybrid multi-bit approximate adders	*PSNR*	*SSIM*
HMBAA -INXA1 (Almurib et al., 2016)	7.56	0.03
HMBAA-INXA2 (Almurib et al., 2016)	45.94	0.9878
HMBAA-INXA3 (Almurib et al., 2016)	8.12	0.5895
HMBAA-AFA1 (Dutt et al., 2018)	45.97	0.9878
HMBAA-AFA2 (Dutt et al., 2018)	46.038	0.9878
HMBAA-AFA3 (Dutt et al., 2018)	45.97	0.9878
HMBAA-AFA4 (Dutt et al., 2018)	46.01	0.9878
HMBAA-MBAFA (Jothin et al., 2018)	45.96	0.9878
HMBAA-FTFA (Palanisamy et al., 2019)	5.61	0.0047
HMBAA-AMA1 (Gupta et al., 2013)	5.32	0.6307
HMBAA-AMA2 (Gupta et al., 2013)	45.97	0.9878
HMBAA-AXA1 (Yang et al., 2013)	45.96	0.9878
HMBAA-AXA2 (Yang et al., 2013)	8.41	0.5895

Figure 15 Comparison of compound metrics for approximate full adders in terms of: (A) ADP; (B) APP and (C) PDP (see online version for colours)

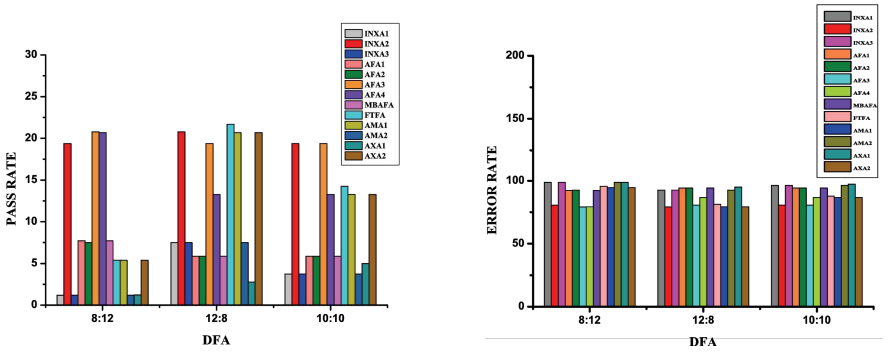


(A) (B)

(C)

The neighbouring pixels values are selected by the window process. The Median filter is coded in MATLAB. Proposed HMBAA for $N = 20$ with 12 MSB bits are accumulated using exact adder and rest using approximate full adders. The corresponding NED values are computed as shown in Table 6. This HMBAA adders for $N = 12 : 8$ are utilised to perform the median filters. 8-bit data type of an input image is mapped with 20-bit data type to perform the 20-bit addition in the filter. The output images of median filters are shown in Figure 16.

Figure 16 Comparison of pass and error rate evaluation for approximate full adders: (A) pass rate and (B) error rate (see online version for colours)



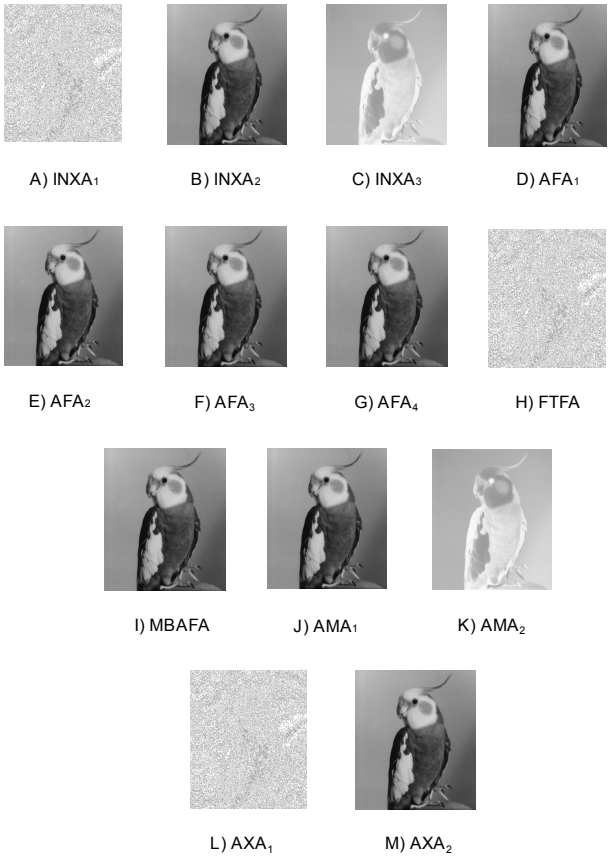
(A) (B)

Peak signal to noise ratio (PSNR) and structural similarity index (SSIM) are the metrics used to evaluate the efficiency of approximate full adders in image applications. PSNR is defined as the ratio between mean square error (MSE) to the maximum value of the pixel in the image.

MSE is evaluated as the difference between the input images to the resulting images. The SSIM (Bovik, 2009) is a quality metric used to estimate image quality degradation caused by noise. PSNR and SSIM values are calculated between the input images to the processed images and they are tabulated in Table 7. The HMBAA which is having less approximation in the LSB segment achieves PSNR and SSIM values which is similar to exact adder.

From Figure 17, it is noted that the output images obtained by using approximate full adders (AFA), MBAFA, INXA achieves similar images. These results indicate that these approximate full adders achieve similar results as similar as the exact adders with the improvement in their hardware complexity.

Figure 17 Output images obtained after applying the median filter implemented with HMBAA



From the overall observations, it is inferred that the HMBAA which is having less approximation in the MSB achieves a reasonable value in their hardware and error. The HMBAA implemented with approximate full adder such as $INXA_2$, AXA_2, AMA_1, MBAFA, and the AFA achieves good PSNR and SSIM values.

6 Conclusion

There's a dire need to extend testing and treatment for COVID-19 to all inhabitants who require it, paying little heed to medical coverage status. There is a potential decrease in the administrative section to keep up with billing and patient records – these issues tend to increase the cost. To reduce the medical cost, we can lean toward minimising the complexity of the circuits for printing, using less area, delay, and price. The optimisation of the adders can achieve the reduced complexity of the circuits. The optimisation of adders can be achieved by adopting approximate adders. Approximate full adders are utilised in these error-resilient applications to minimise the complexity of the circuits that come along with an acceptable amount of output degradation. The approximations are induced on the full adders at the transistor level and also at the gate level. Based on the results in approximate full adders, the probability of invalid output is less at the RTL level than at the transistor level. Also, HMBAAs are developed with the help of an approximate full adder which has various levels of approximation. Hardware utilisation, error, and image metrics are evaluated. Based on our evaluations, we can realise that MSB with least approximations provide good results in error resilient application among different degrees of approximation. Among the 1-bit approximate full adders considered for evaluation, MBAFA, $AFA_{1,}$ and AFA_2 show a trade-off between accuracy and area. Further, the efficiency of these adders is shown in image processing applications with significant improvement in PSNR and SSIM values.

References

Almurib, H.A., Kumar, T.N. and Lombardi, F. (2016) 'Inexact designs for approximate low power addition by cell replacement', *Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE),* Dresden, Germany, pp.660–665.

Bovik, A.C. (2009) *The Essential Guide to Image Processing*, Academic Press, Burlington, MA 01803, USA.

Dutt, S., Nandi, S. and Trivedi, G. (2018) 'Analysis and design of adders for approximate computing', *ACM Transactions on Embedded Computing Systems*, Vol. 17, No. 2, pp.1–28, https://doi.org/10.1145/3131274

Gupta, V., Mohapatra, D., Raghunathan, A. and Roy, K. (2013) 'Low-power digital signal processing using approximate adders', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 32, No. 1, pp.124–137, https://doi.org/10.1109/TCAD.2012.2217962, https://doi.org/10.1145/2742060.2743760

Jiang, H., Han, J. and Lombardi, F. (2015) 'A comparative review and evaluation of approximate adders', *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, ACM, Pittsburgh Pennsylvania USA, pp.343–348.

Jothin, R. and Vasanthanayaki, C. (2018) 'High performance error tolerant adders for image processing applications', *IETE Journal of Research*, Vol. 67, No. 2, pp.1–12, https://doi.org/10.1080/03772063.2018.1535920

Kishore, S. and Sakthivel, R. (2019) 'Ultra-low-voltage GDI-based hybrid full adder design for area and energy efficient arithmetic applications', *IET Circuits, Devices and Systems*, Vol. 13, No. 4, pp.465–470, https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-cds.2018.5559

Liang, J., Han, J. and Lombardi, F. (2013) 'New metrics for the reliability of approximate and probabilistic adders', *IEEE Transactions on Computers*, Vol. 62, No. 9, pp.1760–1771, https://doi.org/10.1109/TC.2012.146

Mittal, S. (2016) 'A survey of techniques for approximate computing', *ACM Computing Surveys*, Vol. 48, No. 4, pp.1–33, https://doi.org/10.1145/2893356

Naseer, A.A., Ashraf, R.A., Dechev, D. and DeMara, R.F. (2015) 'Designing energy-efficient approximate adders using parallel genetic algorithms', *Proceeding of the IEEE Southeast Conference,* USA, pp.1–7, https://doi.org/10.1109/SECON.2015.7132970

Oklobdzija, V. (2001) *The Computer Engineering Handbook*, 2nd ed., CRC Press, Boca Raton.

Palanisamy, G., Natarajan, V.K. and Sundaram, K. (2019) 'Area-efficient parallel adder with faithful approximation for image and signal processing applications', *IET Image Processing*, Vol. 13, No. 13, pp.2587–2594, https://doi.org/10.1049/iet-ipr.2019.0580

Premson, Y. and Sakthivel, R. (2021) 'VLSI system architecture optimisation for DLMS adaptive filter using PPG based multiplier', *International Journal of System of Systems Engineering*, Vol. 11, Nos. 3–4, p.352, https://doi.org/10.1504/IJSSE.2021.121463

Sanapala, K., Satyanarayana, S.V.V. and Sakthivel, R. (2021) 'Near-zero computing using NCFET for ioT applications', *International Journal of Intelligent Enterprise*, Vol. 8, Nos. 2–3, p.288, https://doi.org/10.1504/Ijie.2021.114514

Townsend, W.J., Swartzlander, E.E. and Abraham, J.A. (2003) 'A comparison of dadda and Wallace multiplier delays', *Proceedings of the Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, International Society for Optics and Photonics*, Vol. 5205, pp.552–561, http://dx.doi.org/10.1117/12.507012

Weste, N.H. and Harris, D. (2010) *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed., Addison-Wesley, Boston, USA.

Wu, Y., Li, Y., Ge, X., Gao, Y. and Qian, W. (2018) 'An efficient method for calculating the error statistics of block-based approximate adders', *IEEE Transactions on Computers*, Vol. 68, No. 1, pp.21–38, https://doi.org/10.1109/TC.2018.2859960

Yang, Z., Jain, A., Liang, J., Han, J. and Lombardi, F. (2013) 'Approximate XOR/XNOR based adders for inexact computing', *Proceedings of the 13th IEEE Conference on Nanotechnology (IEEE NANO)*, pp.690–693, https://doi.org/10.1109/NANO.2013.6720793

Zhang, M., Gu, J. and Chang, C.H. (2003) 'A novel hybrid pass logic with static CMOS output drive full-adder cell', *Proceedings of the IEEE ISCAS,* Bangkok, Thailand, pp.317–320, https://doi.org/10.1109/ISCAS.2003.1206266

Bibliography

Parhami, B. (2009) *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Inc., New York.

Priyadharshni, M. and Kumaravel, S. (2019) 'A comparative exploration about approximate full adders for error tolerant applications', in Rajaram, S., Balamurugan, N., Gracia Nirmala Rani, D. and Singh, V. (Eds.): *VLSI Design and Test. VDAT. 2018*, *Communications in Computer and Information Science*, Vol. 892, Springer, Singapore, https://doi.org/10.1007/978-981-13-5950-7_6