ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2020.0164

COMPARATIVE ANALYSIS OF VARIOUS APPROXIMATE FULL ADDERS UNDER RTL CODES

Chinna V. Gowdar[1], M.C. Parameshwara[2] and Savita Sonoli[3]
[1,3]Department of Electronics and Communication Engineering, Rao Bahadur Y Mahabaleshwarappa Engineering College, India
[2]Department of Electronics and Communication Engineering, Vemana Institute of Technology, India

Abstract

Approximate or inexact computing is a well-established paradigm for designing error-tolerant applications such as image and digital signal processing. It is an interesting area of research, especially in the computer arithmetic designs. One key feature of this technique is that it reduces accuracy but still provides meaningful results with low power and reduced circuit complexity. This paper presents a comparative analysis of state-of-the-art approximate 1-bit full adders (AFA) for inexact computation. The performance of these AFAs are compared in terms of the design metrics (DMs) such as power, delay, and area. For a fair comparison, all AFAs under consideration have been described in Verilog register-transfer-level (RTL) codes and synthesized using Cadence's RTL compiler. The synthesis is carried out using Cadence's 180 nm standard cell library.

Keywords:

Approximate Adder, Low Power, Approximate Computing, Full Adder, Inexact Adder

1. INTRODUCTION

The quest to minimize the gap between CMOS technology scaling and application computational workloads has made the designers to look into different techniques [1], [2] one such standard technique is an 'approximate computation' (AC). It exploits the error resilience of a system by trading-off the accuracy and performance. The common applications of an AC are in multimedia, wireless communications, data mining, recognition, neuromorphic systems etc. Processing multimedia applications such as image, audio, and video are error resilient. Since the human beings have limited perceptional capabilities, hence generating numerically approximate results is more sufficient rather than accurate. This relaxation on numerical exactness allows us to perform inexact or approximate computations. The handheld multimedia devices uses the 'Digital Signal Processing' (DSP) system as core to process the image and video [3].

A '1-bit full adder' (FA) being primitive unit of DSP block, plays an important role to determine the overall accuracy and performance of a DSP system. Thus, the state-of-the-art literature extensively focused on investigation of energy efficient DSP by approximating the FA to trade-off the accuracy and performance.

The various works proposed in the current technical literature have discussed and presented the design of energy efficient arithmetic blocks using AC. The energy efficiency can be achieved through AC by the reduction of logic complexity at different levels of design abstraction such as:

- Algorithmic level
- Architecture level
- Gate level
- Transistor level

At the algorithmic level, the energy efficiency is achieved by significant driven computation (SDC). In SDC, the computations are classified as significant and non-significant. The significant computations are performed using exact circuits and non-significant computations are performed using AC [4]-[6]. At architectural level, the reduction of logic complexity is achieved using the voltage over scaling (VOS) and algorithmic noise tolerance (ANT) [7]-[9]. The VOS is used to achieve low power by scaling down the supply voltage below its lower bound. This results in intermittent or time induced errors, whenever the critical delay path is excited. This degrades the signal-to-noise (SNR) performance of a system under consideration. To mitigate this problem an error control technique ANT is used. Thus, the VOS and ANT together are used to achieve low power with required accuracy. The reduction in logic complexity at gate level were investigated in [10]-[14]. And other works discussed extensively on reduction of logic complexity, at transistor level [15]-[20]. In both gate/circuit level, the logic complexity is reduced by eliminating some of the logic gates/transistors in the critical path and thus achieving the energy efficiency. In this paper, the state-of-the-art approximate-FAs (AFAs) have been compared in terms of DMs [21] such as power, delay, and area.

Rest of this paper is organized as follows. Section 2 presents related work on approximate FAs. Section 3 presents the approximate FAs. Section 4 presents the synthesis environment. The section 5 presents the results and discussion. Finally, section 6 concludes this work.

2. RELATED WORK

In the current state-of-the-art literature many 1-bit AFAs have been proposed and discussed at both gate level and transistor level. The brief summary of these adders is as follows:

In [11] an AFA that approximates both *Sum* and C_{out} is presented. The gate level diagram of this AFA is shown in the Fig.1(a). It has smaller number of logic gates and shortest critical path as compared to the conventional exact FA. Three different gate level approximate FA were proposed in [14]. The gate level representation of these 1-bit AFAs are shown in Fig.1(b)-Fig.1(d), all these AFAs are derived by approximating both the Sum and C_{out}. All these adders are designed using smaller number of transistors, simulated and analyzed in 45 nm technology node. The authors claimed that these AFAs consume less power and have low delay.

Table.1. Truth Table of Approximate Full Adders

Adder			**Outputs**															
Inputs			**EFA**		**AFA1**		**AFA2**		**AFA3**		**AFA4**		**AFA5**		**AFA6**		**AFA7**	
A	***B***	C_{in}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}
0	0	0	0	0	1×	0✓	0✓	0✓	0✓	0✓	1×	0✓	1×	0✓	1×	0✓	0✓	0✓
0	0	1	1	0	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓
0	1	0	1	0	1✓	0✓	1✓	0✓	0×	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓
0	1	1	0	1	1×	0×	1×	1✓	1×	1✓	0✓	1✓	1×	1✓	0✓	1✓	0✓	1✓
1	0	0	1	0	0×	1×	0×	0✓	0×	0✓	1✓	0✓	0×	0✓	1✓	0✓	1✓	0✓
1	0	1	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓
1	1	0	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	1✓
1	1	1	1	1	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	0×	1✓	1✓	1✓
Reference			[12]															

Adder			**Outputs**															
Inputs			**EFA**		**AFA8**		**AFA9**		**AFA10**		**AFA11**		**AFA12**		**AFA13**		**AFA14**	
A	***B***	C_{in}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}
0	0	0	0	0	0✓	0✓	1×	0✓	1×	0✓	0✓	0✓	0✓	0✓	0✓	0✓	0✓	0✓
0	0	1	1	0	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	0×	0✓	1✓	1×	1✓	0✓
0	1	0	1	0	0×	1×	1✓	0✓	0×	1×	0×	0✓	0×	0✓	1✓	0✓	1✓	0✓
0	1	1	0	1	0✓	1✓	0✓	1✓	0✓	1✓	1×	0×	0✓	0×	0✓	1✓	1×	1✓
1	0	0	1	0	0×	0✓	1✓	0✓	1✓	0✓	0×	1×	1✓	1×	1✓	0✓	1✓	0✓
1	0	1	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	1✓	0✓	1✓	1×	1✓
1	1	0	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	1✓	0✓	0×	0✓	1✓
1	1	1	1	1	1✓	1✓	0×	1✓	0×	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓
Reference			[13,18]												[14]			

Adder			**Outputs**															
Inputs			**EFA**		**AFA15**		**AFA16**		**AFA17**		**AFA18**		**AFA19**		**AFA20**		**AFA21**	
A	***B***	C_{in}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}	***Sum***	C_{out}
0	0	0	0	0	1×	0✓	1×	0✓	1×	0✓	1×	0✓	0✓	0✓	1×	0✓	0✓	0✓
0	0	1	1	0	0×	1×	1✓	0✓	0×	1×	1✓	0✓	1✓	0✓	0×	1×	0×	1×
0	1	0	1	0	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓
0	1	1	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	0×	0✓	1✓	0✓	1✓
1	0	0	1	0	1✓	0✓	0×	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓
1	0	1	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	0×	0✓	1✓	0✓	1✓
1	1	0	0	1	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	1×	1✓	1×	0×
1	1	1	1	1	0×	1✓	0×	1✓	1✓	0×	0×	1✓	1✓	1✓	1✓	1✓	1✓	1✓
Reference			[10]										[11]		[15]		[16]	

Adder			Outputs															
Inputs			EFA		AFA22		AFA23		AFA24		AFA25		AFA26		AFA27		AFA28	
A	*B*	C_{in}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}	*Sum*	C_{out}
0	0	0	0	0	0✓	0✓	0✓	0✓	1×	0✓	0✓	0✓	0✓	0✓	0✓	0✓	1×	0✓
0	0	1	1	0	0×	0✓	1✓	0✓	1✓	0✓	1✓	0✓	1✓	0✓	0×	0✓	1✓	0✓
0	1	0	1	0	1✓	0✓	0×	1×	0×	0✓	0×	0✓	1✓	0✓	0×	0✓	1✓	0✓
0	1	1	0	1	1×	0×	1×	0×	0✓	1✓	0✓	1✓	1×	1✓	0✓	1✓	0✓	1✓
1	0	0	1	0	0×	1×	0×	1×	0×	0✓	0×	0✓	1✓	0✓	0×	0✓	1✓	0✓
1	0	1	0	1	0✓	1✓	1×	0×	0✓	1✓	0✓	1✓	1×	1✓	0✓	1✓	0✓	1✓
1	1	0	0	1	1×	1✓	0✓	1✓	1×	1✓	0✓	1✓	1×	1✓	0✓	1✓	0✓	1✓
1	1	1	1	1	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	1✓	0×	1✓
Reference					[13] [18]		[19]						[20]				[17] [20]	

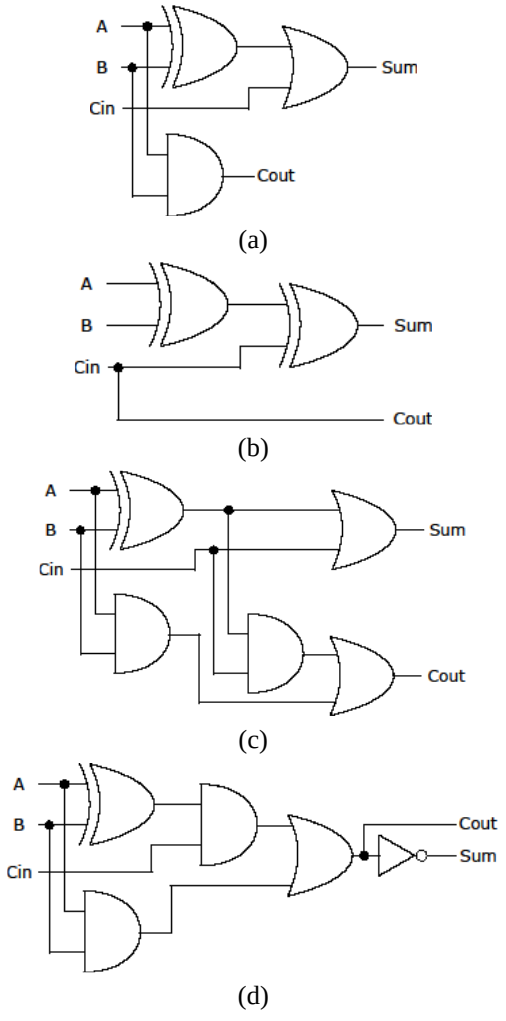
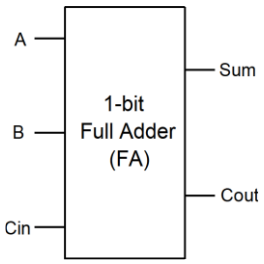


Fig.1. Gate level approximate full adders reported in [11] [14]

In [12], a total 7 gate level AFAs have been proposed and used in the design and analysis of approximate multipliers. All the proposed approximate multipliers have been simulated and synthesized using Cadence's RTL compiler using TSMC 180 nm standard cell library. Among the 7 proposed AFAs, the authors found that the *Sum* and C_{out} approximated adders are more power efficient. A carry based approximate FAs were reported in [10]. In this work the authors claimed that the carry based approximate adders are found to be more efficient than conventional approximate adders. The other works [15] [16] also discussed on gate level approximate adders and discussed their merits in terms of power and delay. The transistor level approximate adders have been extensively studied in the works [13] [14] [18] [20]. A 5 different approximate adders have been derived by approximating the CMOS mirror adder [13] [18]. The approximation is carried out by removing some of the transistors in the critical path. This resulted in reduced delay and low power.

3. APPROXIMATE FULL ADDERS

The state-of-the-art AFAs are derived based on approximation of the Sum and Carry outputs of exact FA. The block diagram and truth table of exact FA is shown in Fig.2(a) and Fig.2(b), respectively.



(a)

A	*B*	C_{in}	*Sum*	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b)

Fig.2. FA (a) Block diagram (b) Truth Table

The *Sum* and C_{out} of exact FA are expressed as follows

$$Sum = A \oplus B \oplus C_{in} \quad (1)$$

$$C_{out} = A.B + B \cdot C_{in} + A \cdot C_{in} \quad (2)$$

The state-of-the-art AFAs are derived either by approximating the *Sum* or C_{out} or both outputs of the exact full adder (EFA). Depending on the output that is being approximated, in this work we classify the AFAs into 3 different types.

3.1 APPROXIMATED SUM ADDERS (TYPE1)

These AFAs are derived by approximating the Sum alone and retaining the exact C_{out} as in Eq.(2). These type approximate adders are herein referred to as 'Type1'. From the Table.1 it is found that the approximate adders AFA2-AFA7 [12], AFA9 [13] [18], AFA14 [14], AFA16, AFA18 [10], AFA24, AFA25 [19], and AFA27-AFA29 [17] [20], all these adders fall under Type1 category. The K-map simplified Sum expressions for Type1 AFAs are listed in Table.2.

Table.2. K-Map Simplified approximated Sum

AFA	*Sum*
AFA2	$\bar{A} \cdot (B + C_{in}) + B \cdot C_{in}$
AFA3	$(\bar{A} + B) \cdot C_{in}$
AFA4	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in} + A \cdot B \cdot C_{in}$
AFA5	$\bar{A} + B \cdot C_{in}$
AFA6	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in}$
AFA7	$\bar{B} \cdot \bar{C}_{in} + \bar{A} \cdot \bar{C}_{in} + \bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{B} \cdot C_{in}$
AFA9	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in}$
AFA14	$C_{in} + \bar{A} \cdot B + A \cdot \bar{B}$
AFA16	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in}$
AFA18	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in}$
AFA24	$\bar{A} \cdot \bar{B} + A \cdot B$
AFA25	$\bar{A} \cdot \bar{B} \cdot C_{in} + A \cdot B \cdot C_{in}$
AFA26	$A + B + C_{in}$
AFA27	$A \cdot B \cdot C$
AFA28	$\bar{A} \cdot \bar{B} + \bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in}$

3.2 APPROXIMATED C_{OUT} ADDERS (TYPE 2)

The Type 2 AFAs are derived by approximating the Cout alone and retaining the exact Sum as in Eq.(1). From the Table.1 it is found that the only approximate adder that fall under this category is AFA13 [14]. The K-map simplified Sum expression for Type 2 AFA is listed in Eq.(3).

$$C_{out} = C_{in} \quad (3)$$

3.3 APPROXIMATED *Sum* AND C_{OUT} ADDERS (TYPE3)

The Type3 AFAs are derived by approximating both Sum as well as C_{out} outputs. From the Table.1 it is found that the approximate adders namely AFA1 [12], AFA8, AFA10-AFA12 [13, 18], AFA15, AFA17 [10], AFA19 [17], AFA20 [15], AFA21 [16], AFA22 [13], and AFA23 [18] fall under Type3.

The K-map simplified *Sum* expression for Type3 AFAs are tabulated in Table.3.

Table.3. K-Map Simplified approximated *Sum* and C_{out}

AFA	*Sum*	C_{out}
AFA1	$\bar{A} + B \cdot C_{in}$	A
AFA8	$\bar{A} \cdot \bar{B} \cdot C_{in} + A \cdot B \cdot C_{in}$	$B + A \cdot C_{in}$
AFA10	$\bar{A} \cdot \bar{B} + \bar{B} \cdot \bar{C}_{in}$	$B + A \cdot C_{in}$
AFA11	$(\bar{A} + B) \cdot C_{in}$	A
AFA12	A	A
AFA15	$\bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in}$	$C_{in} + A \cdot B$
AFA17	$\bar{A} \cdot \bar{C}_{in} + \bar{B} \cdot \bar{C}_{in} + A \cdot B \cdot C_{in}$	$\bar{A} \cdot C_{in} + \bar{B} \cdot C_{in} + A \cdot B \cdot \bar{C}_{in}$
AFA19	$C_{in} + \bar{A} \cdot B + A \cdot \bar{B}$	$A \cdot B$
AFA20	$\bar{C}_{in} + A \cdot B$	$C_{in} + A \cdot B$
AFA21	$B \cdot \bar{C}_{in} + A \cdot \bar{C}_{in} + A \cdot B$	C_{in}
AFA22	B	A
AFA23	C_{in}	$B \cdot \bar{C}_{in} + A \cdot \bar{C}_{in} + A \cdot B$

4. SYNTHESIS ENVIRONMENT

The synthesis environment used to extract the DMs is shown in Fig.3. The Verilog RTL code and TSMC 180 nm standard cell library are used an inputs to the synthesis tool. With these inputs the Cadence's RTL compiler (RC) generates gate level netlist to extract the required DMs. For a fair comparison, all the AFAs under consideration have been described using Verilog RTL codes and synthesized using supply voltage, V_{dd} = 1.8 V.

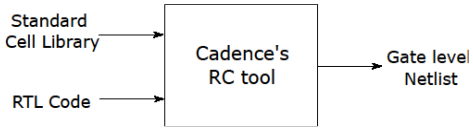


Fig.3. Synthesis environment used to extract the DMs

RTL is an acronym for 'register transfer level'. Any synthesizable HDL code is called RTL code. An example of RTL code that describes an AFA1 is illustrated below. In this example the continuous assignment statements are used to describe RTL code.

module AFA1(*A*, *B*, C_{in}, *Sum*, C_{out});
input *A*, *B*, C_{in};
output *Sum*, C_{out};
assign Sum = ~*A*|(*B* and C_{in});
assign Cout = *A*;
endmodule

5. RESULTS AND DISCUSSION

This section presents the performance of state-of-the-art AFAs. The DMs that have been extracted for Type1, Type 2 and Type3 AFAs (using the synthesis environment shown in Fig.3) are tabulated in the Table.4 - Table.6 respectively. The maximum and minimum values in each column are highlighted using bold font.

From the Table.4, among the Type1 AFAs, the following inferences can be drawn:

- The AFA6, AFA9, AFA18, and AFA28 are found to be power and area efficient. These adders are having same power (2647.961nW) and area (36.59µm^2). This can be attributed to their underlying architecture.
- The AFA3, AFA5, and AFA27 are found to be having low delay this is due to shortest critical path. The delay of these adders are equal and found to be 169ps.
- The AFA2 is having a leakage power of 0.101nW, which is the lowest among other Type-1 AFAs.

Considering the Table.5, among Type 2 AFAs, it is noticed that:

- AFA13 is the only adder that falls under this category, it has leakage power = 0.314nW, total power = 3869.344nW, delay = 371ps, and area = 49.90µm^2.

Considering Table.6, among the Type3 AFAs, the following observations can be made:

- The AFA10 has the low leakage power (0.034nW).
- The AFA1 is found to be power (1077.562nW), delay (100ps) and area (16.63µm^2) efficient as compared to other AFAs.

Note: Among AFA12 and AFA22 are not considered for comparison as their area and delay are found to be same and equal to 0. Therefore it is not fair to compare these adders against other AFAs.

Table.4. Design Metrics of Type1 AFAs

AFA	Power (nW)			Delay (ps)	Area (µm^2)
	Leakage	Dynamic	Total		
AFA2	**0.101**	3525.857	3525.957	214	53.22
AFA3	0.153	3536.112	3536.265	**169**	56.55
AFA4	0.319	**5101.063**	**5101.381**	**383**	**76.50**
AFA5	0.189	2921.106	2921.295	**169**	46.57
AFA6	0.119	**2647.841**	**2647.961**	271	**36.59**
AFA7	0.135	4505.181	4505.316	379	69.85
AFA9	0.119	**2647.841**	**2647.961**	271	**36.59**
AFA14	**0.487**	4497.339	4497.826	332	69.85
AFA16	0.217	3691.747	3691.964	334	49.90
AFA18	0.119	**2647.841**	**2647.961**	271	**36.59**
AFA24	0.278	3153.634	3153.912	279	49.90
AFA25	0.289	3544.190	3544.479	280	59.87
AFA26	0.341	3094.364	3094.705	331	49.90
AFA27	0.179	2871.688	2871.866	**169**	46.57
AFA28	0.119	**2647.841**	**2647.961**	271	**36.59**

Table.5. Design Metrics of Type 2 AFAs

AFA	Power (nW)			Delay (ps)	Area (µm^2)
	Leakage	Dynamic	Total		
AFA13	0.314	3869.030	3869.344	371	49.90

Table.6. Design Metrics of Type3 AFAs

AFA	Power (nW)			Delay (ps)	Area (µm^2)
	Leakage	Dynamic	Total		
AFA1	0.082	**1077.481**	**1077.562**	**100**	**16.63**
AFA8	0.195	3752.307	3752.502	249	56.55
AFA10	**0.034**	1510.208	1510.242	191	19.95
AFA11	0.046	1692.486	1692.532	135	26.61
AFA15	0.034	1680.161	1680.195	191	19.95
AFA17	0.119	3090.067	3090.186	**292**	39.92
AFA19	0.137	2891.337	2891.474	254	46.57
AFA20	0.139	2223.611	2223.749	148	33.26
AFA21	0.099	1966.927	1967.026	159	33.26
AFA23	0.099	1966.927	1967.026	159	33.26

6. CONCLUSION

This paper discussed and compared a total 28 AFAs that were reported in the state-of-the-art technical literature. The comparison of these adders have been performed in terms of design metrics (DMs) such as total power, delay, and area. Based on the type approximation used, we have classified the AFAs into three different categories namely Type-1, Type-2, and Type-3. Among these, the AFA1 is found to be power and area efficient with low delay. It has a total power = 1077.562nW, delay = 100ps, and area=16.63µm^2. The AFA10 has the lowest leakage power as compared to any other AFA under consideration. It has the leakage power of 0.034nW. This comparison work assists the designers to trade-off the various AFAs in terms of power, area, and delay DMs. This study also helps in choosing a right AFA for particular application. There is a wide scope for further research on these AFAs at image/video/audio application level.

REFERENCES

[1] G. Zervakis, K. Koliogeorgi, D. Anagnostos, N. Zompakis and K. Siozios, "VADER: Voltage-Driven Netlist Pruning for Cross-Layer Approximate Arithmetic Circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 27, No. 6, pp. 1460-1464, 2019.

[2] A. Yazdanbakhsh, D. Mahajan, P. Lotfi Kamran and H. Esmaeilzadeh, "AxBench: A Benchmark Suite for Approximate Computing Across the System Stack", Technical Report, Department of Computer Science, Georgia Institute of Technology, pp. 1-14, 2016.

[3] H.P. Wong, "The End of the Road for 2D Scaling of Silicon CMOS and the Future of Device Technology", *Proceedings of 76th International Conference on Device Research*, pp. 1-2, 2018.

[4] D. Mohapatra, G. Karakonstantis and K. Roy, "Significance Driven Computation: A Voltage-Scalable, Variation-Aware, Quality-Tuning Motion Estimator", *Proceedings of IEEE/ACM International Symposium on Low Power Electronics Design*, pp. 195-200, 2009.

[5] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, S. Das and A. Yakovlev, ''Significance-Driven Logic Compression for Energy-Efficient Multiplier Design", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 8, No. 3, pp. 417-430, 2018.

[6] N. Banerjee, G. Karakonstantis and K. Roy, "Process Variation Tolerant Low Power DCT Architecture", *Proceedings of International Conference on Design, Automation and Test*, pp. 1-6, 2007.

[7] R. Hegde and N.R. Shanbhag, "A Voltage Overscaled Low-Power Digital Filter IC", *IEEE Journal of Solid-State Circuits*, Vol. 39, No. 2, pp. 388-391, 2004.

[8] R. Hegde and N.R. Shanbhag, "Soft Digital Signal Processing", *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, Vol. 9, No. 6, pp. 813-823, 2001.

[9] H. Afzali Kusha, M. Vaeztourshizi, M. Kamal and M. Pedram, "Design Exploration of Energy-Efficient Accuracy-Configurable Dadda Multipliers with Improved Lifetime Based on Voltage Overscaling", *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, Vol. 28, No. 5, pp. 1207-1220, 2020.

[10] M. Ramasamy, G. Narmadha and S. Deivasigamani, "Carry based Approximate Full Adder for Low Power Approximate Computing", *Proceedings of 7th International Conference on Smart Computing and Communication*, pp. 1-4,2019.

[11] D. Shin and S.K. Gupta, "A Re-Design Technique for Datapath Modules in Error Tolerant Applications", *Proceedings of 17th Asian Test Symposium*, pp. 431-437, 2008.

[12] G. Anusha and P. Deepa, "Design of Approximate Adders and Multipliers for Error Tolerant Image Processing", *Journal of Microprocessors and Microsystems*, Vol. 72, No. 2, pp. 1-7, 2019.

[13] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, "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, 2013.

[14] H.A.F. Almurib, T.N. Kumar and F. Lombardi, "Inexact Designs for Approximate Low Power Addition by Cell Replacement", *Proceedings of International Conference on Design, Automation and Test*, pp. 660-665, 2016.

[15] H. Waris, C. Wang and W. Liu, "High-Performance Approximate Half and Full Adder Cells using NAND Logic Gate", *IEICE Electronics Express*, Vol. 55, No. 3, pp. 1-3, 2019.

[16] T. Zhang, W. Liu, E. McLarnon, M. O'Neill and F. Lombardi, "Design of Majority Logic (ML) Based Approximate Full Adders", *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 1-5, 2018.

[17] C. Labrado, H. Thapliyal and F. Lombardi, "Design of Majority Logic Based Approximate Arithmetic Circuits", *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 2122-2125, 2017.

[18] V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan and K. Roy, "IMPACT: IMPrecise Adders for Low-Power Approximate Computing", *Proceedings of IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 409-414, 2011.

[19] Z. Yang, A. Jain, J. Liang, J. Han and F. Lombardi, "Approximate XOR/XNOR-Based Adders for Inexact Computing", *Proceedings of 13th IEEE International Conference on Nanotechnology*, pp. 690-693, 2013.

[20] Z. Zareei, K. Navi and P. Keshavarziyan, "Low-Power, High-Speed 1-Bit Inexact Full Adder Cell Designs Applicable to Low-Energy Image Processing", *International Journal of Electronics*, Vol. 105, No. 3, pp.375-384, 2018.

[21] M.C. Parameshwara and H.C. Srinivasaiah, "Low-Power Hybrid 1-Bit Full Adder Circuit for Energy Efficient Arithmetic Applications", *Journal of Circuits, Systems and Computers*, Vol. 26, No. 1, pp. 1-15, 2017.