International Journal of Research in Science and Technology
Volume 10, Issue 4: October - December, 2023
ISSN 2394 - 9554

COMPARISON OF POWER OPTIMIZED APPROXIMATE ADDERS FOR IMAGE PROCESSING AND MULTIMEDIA APPLICATIONS

Akshara R T[1] and Harshini A[2]
[1,2]Student, Department of ECE, PSG Institute of Technology and Applied Research, Neelambur - 641062

ABSTRACT
Despite occupying the lesser area, the basic half and full adders available to us offer limited benefits in complex applications. Certain fault tolerant applications require power efficient adders that operates at high speed. Hence approximate adders are employed in applications where there can be a considerable tradeoff with accuracy. In image processing and multimedia applications, the compromise in accuracy does not significantly impact the output. This is because the human senses are insensitive to such minute in accuracies. The circuits for more advanced applications perform efficiently with run time configurable approximate adders. This paper reviews and compares the various approximate adders.

1. INTRODUCTION
To enhance the speed and ensure efficiency in power consumption through approximation, the following strategies can be adopted: voltage-over-scaling and redesigning. In voltage-over-scaling, the supply voltage is varied to a larger extent to optimize the power consumption. The former strategy is applicable in various logic circuits, but it is less preferred since it causes timing errors. Redesigning the adders to approximate adders reduce the logic complexity of the circuit and the cost of implementation. Reducing the logic complexity of adders saves power and reduces the number of transistors, thereby reducing the area. In this way, the approximate adders consume less power than the conventional adders. The basic idea of approximate adders is to reduce the propagation delay caused by the carry. As the accuracies of MSBs impact the accuracy of the output, the higher-order bits are computed with more accurate adders while the lower-order bits are computed with approximate adders in most cases. Though the Almost Correct Adder (ACA) provides less accuracy it consumes more power[1,2,3]. The major reason why the conventional adders are less suitable for complex applications is their high power consumption and larger critical path delay. To address these drawbacks of accurate adders, segmented and speculative adders were proposed [2]. An alternative approach is to redesign the full adders into approximate adders. The following section discusses Ripple-Carry Adder (RCA) [2], Carry Look-Ahead Adder (CLA)[2], Speculative adders[2,5], Segmented Adders [2], Full Adders, and Approximate Adders [1].

2. BACKGROUND

2.1 Ripple Carry Adder (RCA):
In conventional RCA, to compute the sum of n-bits, n full adders must be employed. So the full adder in the $(i+1)^{th}$ stage has to wait for the carry generation and propagation of the full adder in the i^{th} stage. Thus the latency increases proportionally with increase in the number of bits or O(n). Delay of an n-bit RCA is n times the propagation delay of the full adder.

2.2 Carry Look-Ahead Adder (CLA):
Unlike RCA, CLAs doesn't have delay due to carry propagation, since the carry is generated parallelly by a carry generate circuit and therefore the delay is O(log(n)). In an n-bit CLA, the sum, propagate ($p_i = a_i+b_i$) and generate ($g_i = a_ib_i$) signals are produced by n SPGs. In spite of having a shorter delay CLAs occupy a larger area and consume more power. With increase in n, the area complexity increases further more. CLA has an area complexity of O (n log(n)), when fan-in and fan-out of the gates are fixed.

2.3 Speculative Adders:
An Approximation is achieved by decreasing the critical path and the hardware complexity of the conventional adders. The carry propagation chain which is the critical path of the adders is split into 2 or more shorter paths, instead of the carry propagation chain spanning the entire length of the adder. Here the carry is estimated at intermediate stages by using the previous less significant stages. This improves the energy-delay-area product to a greater extent when compared with the conventional adders.

2.3.1 Almost Correct Adder (ACA):
The speculative adder technique is employed in ACA. The length of the carry chain is much shorter than n, where n is the number of bits of the largest operand. The carry is predicted from k lower order bits($k<n$). This reduces the critical path delay to O(log k). An n-bit adder requires (n-k) k bit carry generators, which in turn increases the hardware complexity. However, this hardware overhead can be resolved by sharing a few components of the carry generators. An improvised version of the speculative adder, variable latency

ISSN 2394 - 9554

speculative adder (VLSA) integrates error detection and recovery circuit. VLSA operates at a speed of 1.5 times compared to CLA.

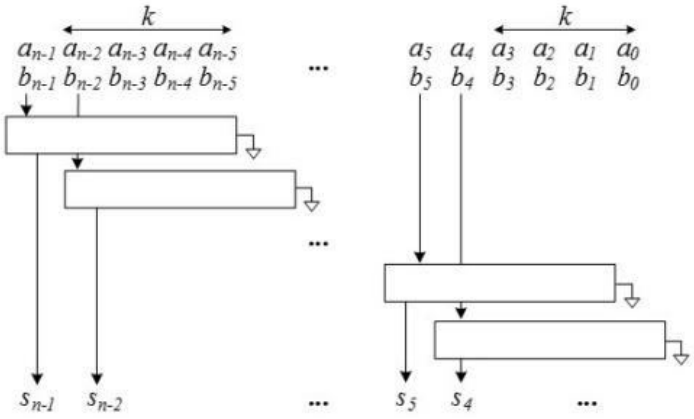

Figure 1: Carry propagation path of ACA

2.4 Segmented Adders:

In segmented adders several smaller adders operate parallelly. Therefore, the carry propagation chain divided into several shorter segments.

2.4.1 Equal Segmentation Adder (ESA):

A dynamic segmentation technique is integrated with error compensation to design a DSEC adder. By neglecting the error compensation mechanism an approximate design is obtained since our primary aim is to design approximate adders that consume less power. ESA is a simplified version of the DSEC adder. The circuit is made up of (n/k) sub-adders. The sub-adders obtained by bit slicing the data path. The size of the first sub adder is less than or equal to the size of other sub adders. Hence the hardware complexity is much lesser than an ACA. ESA has a delay of O(log(k)).

2.4.2 Error-Tolerant Adder Type-II (ETAII):

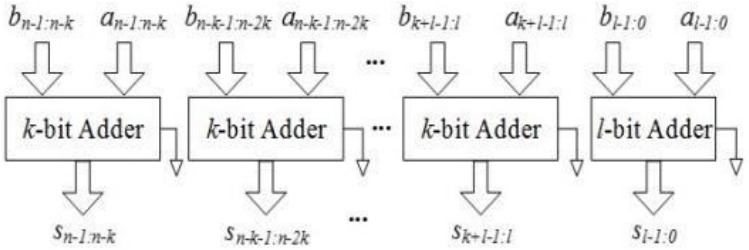

Figure 2: ESA: k-maximum length of the carry chain, l-length of the sub-adder

ETA II is quite different from ESA. The circuit is built with carry generators and sum generators. The carry propagation is not completely neglected instead it is divided into several smaller paths. The carry propagation in the shorter paths are computed simultaneously. The sum generator of the higher order bits waits for the carry signal to be propagated from the previous carry generator. This makes ETAII more accurate than ESA for the same values of k, because the carry bit is predicted with more accuracy. This does not increase the hardware complexity of ETAII and it remains similar to that of ESA. But the delay of ETAII is (O(log(2k)), larger than ESA.

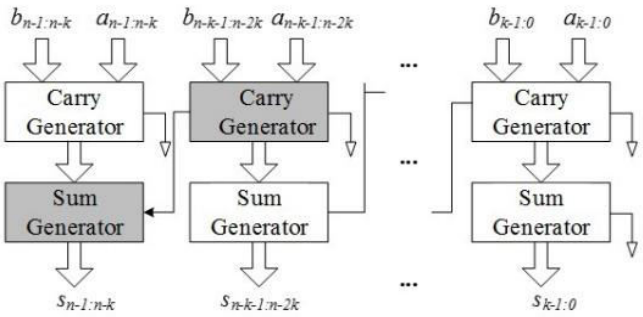

Figure 3: ETAII: The carry propagation happens between the shaded blocks.

ISSN 2394 - 9554

ETAI, ETAII and ETAIIM are proposed in [3]. ETAI comprises an accurate part and inaccurate part to achieve application specific results. In ETAIIM carry parts are connected between the accurate MSB parts.

2.4.3 Accuracy Configurable Approximate Adder (ACAA):

Sometimes the same application might require accurate results at some scenarios and might be error tolerant in other scenarios. The requirement of such applications changes with respect to time and requirements of the user. This creates a necessity for a run time configurable circuit where the accuracy is configured by modifying the structure of the circuit. This creates a trade-off of accuracy, performance and power. An n-bit ACAA adder requires [(n/k)-1] 2k-bit sub-adders. Each sub-adder performs addition over 2k- consecutive bits with an overlap of k-bits. All the sub-adders operate parallelly, thereby reducing the delay to O(log(2k)). The half most significant sum bits of each sub-adder are considered as the partial sum.

ACAA consists of an inbuilt error detection and correction (EDC) circuit with a pipelined architecture to correct the errors generated. ACAA and ETAII share the same error characteristics since their carry propagation paths are similar.

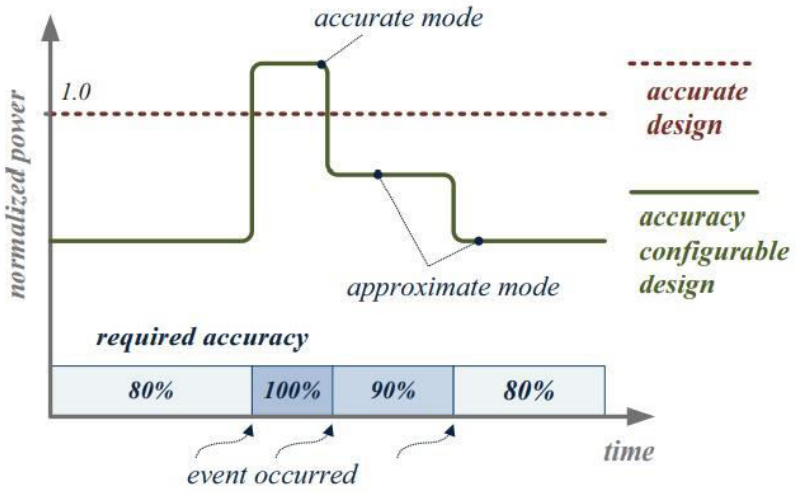

Figure 4: Power benefits of ACAA.

The unique feature of ACAA is that it operates in different modes based on the situation by adapting to change in accuracy. The advantage of ACAA over the previous approximate adders is that, the previous approximate adders do not have EDC circuits since they are particularly designed for error tolerant applications. This makes them incompatible for accurate computations. Though VLSI provides accurate results the error detection and correction mechanisms result in a large delay and area overhead.

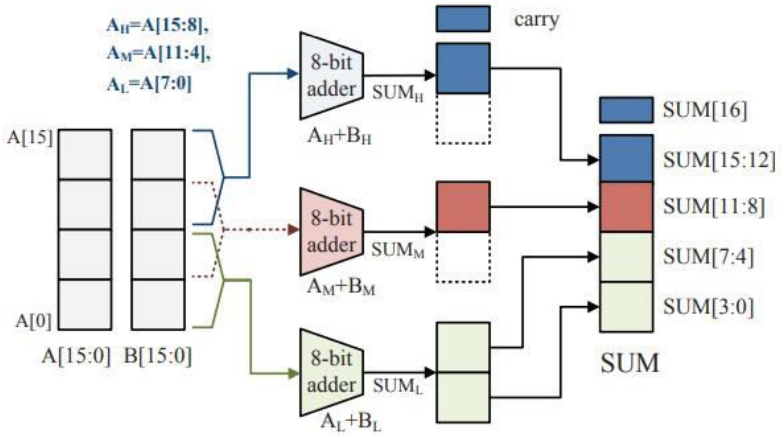

Figure 5: 16-bit ACAA.

The critical path delay is reduced by eliminating the carry chain. The reduce in the critical path delay increases the performance by increasing the clock frequency and decreases power consumption by decreasing the operating voltage. The partial summation results are generated by the 3 sub-adders. The accuracy is increased by introducing a middle sub-adder. In the absence of middle sub-adder, if the 8^{th} carry bit is high, an error occurs. For random input patterns, the error rate is as high as 50.1%. The introduction of the middle sub-adders reduces the error rate to 5.5%

ISSN 2394 - 9554

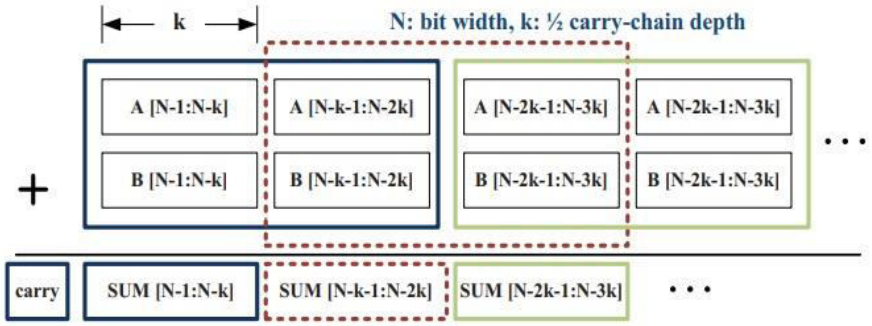

Figure 6: General implementation of ACAA. k is the bit-width of the sub-adders result.

Each divided sub-module of the adder produces a k-bit result whereas the last sub-module produces a 2k-bit result. In total the approximate adder consists of [(N/k)-1] sub-modules. The following equations describes the same.

SUM[N-ik-1:N-(i+1)k]=A[N-ik-1:N-(i+1)k] +B[N-ik-1:N-(i+1)k]

Where i=0,1,……(N/k)-2

The following equations represent the delay, area, power of the sub-adders respectively.

delay= $C_{delay}(\log_2 k +1)$

area= $C_{area}(N-2k)(\log_2 k +1)$

$Power_{dyn}= C_{power}(N-2k)(\log_2 k +1)^2$

where C_{delay} is constant for delay, C_{area} is constant for area and C_{power} is constant for power.

2.4.3.1 Rough estimation of power consumption of ACAA

i. At a fixed frequency, with voltage scaling, dynamic power consumption is proportional to capacitance. V_{dd}^2 and the capacitance is proportional to the area.

ii. Cell delay is proportional to $1/(V_{dd} -V_t)^{\beta}$, assuming $\beta = V_{dd}^2$ is approximately proportional to 1/cell delay.

iii. At a fixed frequency, cell delay x path depth is a constant, V_{dd}^2 is proportional to the path depth, which is $\log_2 k +1$.

iv. For dynamic power consumption, C_{power} is constant for given V_{dd}.

v. Static power consumption is approximated to be proportional to the area.

When a carry input, is to be propagated to the results, the output of each sub-adder, excluding the last sub-adder is incorrect. For a random input vector, the probability of obtaining a correct result is

$$P(N,k) = (1 - \frac{1}{2^k} \cdot \frac{2^k - 1}{2^{k+1}})^{\frac{N}{k}-2}$$

	k=2	k=3	k=4	k=5	k=6
min. clock period	0.5	0.65	0.75	0.83	0.89
area	0.87	1.05	1.12	1.15	1.12
dynamic power	0.44	0.68	0.84	0.95	1.00
pass rate	0.554	0.829	0.942	0.982	0.995

Figure 7: Estimated minimum clock cycle, area, dynamic power and pass rate for each value of k, when N=16 (Normalized to the conventional CLA 16-bit adder).

In the above table, it is observed that for smaller values of k, the clock period and dynamic power can be reduced, but this decreases the pass rate.

ISSN 2394 - 9554

2.4.3.2 Error detection and correction for accurate computation

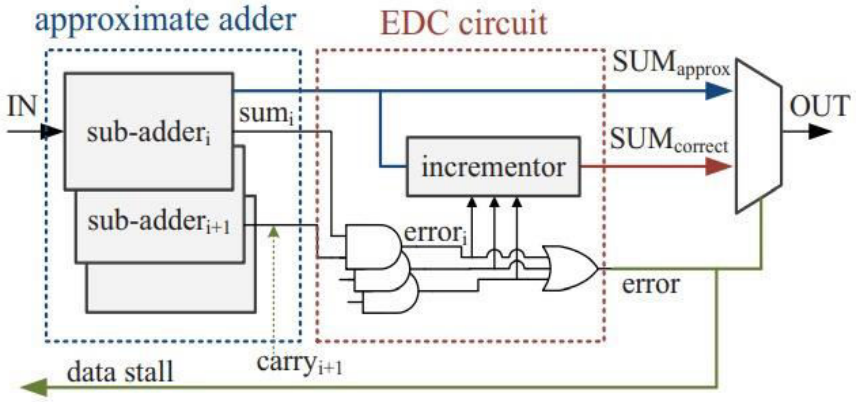

Figure 8: Error detection and correction with ACAA

The propagation of carry bit, between the sub-adders results in an incorrect output, however this error can be detected and corrected with a small overhead. The output of the sub-adders and the carry-in signal of the previous sub-adder are checked for errors. Error detection is implemented with AND gates and error correction is implemented by adding 1 to the inaccurate output. In general an error correction is done with the help of an incrementor circuit.

2.4.3.3 Accuracy Configuration

Through pipelined architecture, the previous approximate adders, have lesser throughput than the conventional adders. By combining ACAA with a pipelined architecture we can obtain both accurate results as well as the same throughput of that of the conventional adders.

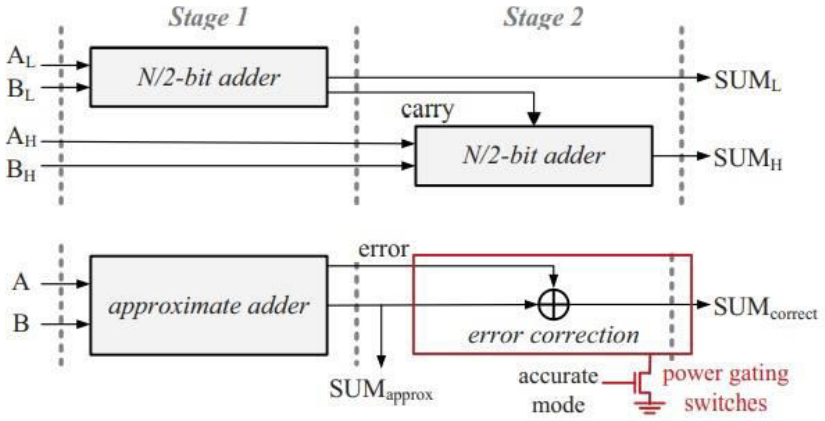

Figure 9: Pipelined implementation of conventional adder (above), Pipelined implementation of approximate adder (below).

In the first pipelined stage approximate additions are computed, in the second stage error correction technique is implemented. Pipelined approach doesn't improve the clock frequency since the clock period achieved in ACAA is same as that of the conventional adder. The configuration of accuracy however provides power benefits. The performance can be further improved by reducing the depth of carry chain of the sub-adders. In order to improve the accuracy of results in the pipelined implementations, the error correction stages should be extended to multiple stages.

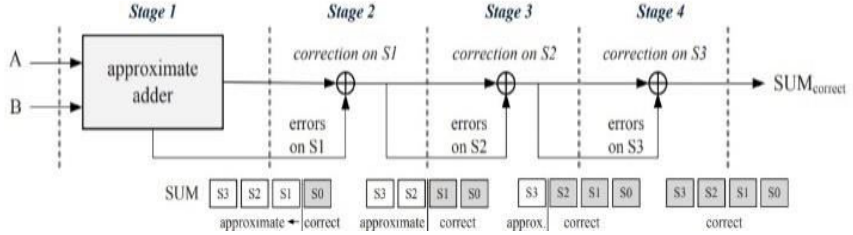

Figure 10: Accuracy configurable implementation for pipelined architecture.

ISSN 2394 - 9554

In the above circuit, where k=N/8, to achieve a 100% accurate result 4 pipelined stages are essential. In the pipelined adder the accuracy of the result generated at each stage is different. The increase in number of pipeline stages increases the accuracy of the output. The pipelined architecture employs a power gating technique.

2.5 Approximate Adders

The basic building block of digital circuits are full adders. In order to explore a low power and high-speed circuits, we move towards approximate adders.

The existing full adders are approximated based on the idea of reduction in logic complexity. Those two techniques are:

- Approximation of sum only
- Approximation of both sum and carry out.

Compromising the accuracy of the arithmetic circuit provides improvement in metrics such as lesser delay and reduced circuit complexity. The approximate adders experience a tradeoff in area and probability of error. The reduction of logic complexity at bit level in the approximate adders offers power savings.

2.5.1 Approximate adder 1 (AA1) :

In this type of adder both sum and carry out (C_{out}) are approximated.

Sum = (ABCin + Cout'Cin)

Cout = ACin + B

2.5.2 Approximate adder 2 (AA2) :

In this type of adder only sum is approximated.

Sum =Cout'

Cout = AB + BCin + ACin

2.5.3 Approximate adder 3 (AA3) :

In this type of adder both sum and carry out (C_{out}) are approximated.

Sum = Cout'

Cout = ACin + B

2.5.4 Approximate adder 4 (AA4) :

In this type of adder both sum and carry out (C_{out}) are approximated.

Sum = ABCin+Cout' Cin'

Cout = A

2.5.5 Approximate adder 5(AA5) :

In this type of adder both sum and carry out (C_{out}) are approximated.

Sum = B

Cout = A

2.5.6 Approximate adder 6(AA6) :

In this type of adder both sum and carry out (C_{out}) are approximated.

Sum = (A' + BCin')

Cout = A

2.5.7 Approximate adder 7(AA7) :

In this type of adder only sum is approximated.

Sum = (A' (B+Cin) + BCin)

Cout = AB + BCin + ACin

2.5.8 Approximate adder 8(AA8) :

In this type of adder only sum is approximated.

ISSN 2394 - 9554

Sum = (A' + B Cin)

Cout = AB + BCin + ACin

2.5.9 Approximate adder 9(AA9) :
In this type of adder only sum is approximated.

Sum = A' B' + B' C' $_{in}$ + ABC_{in} + A'BC'_{in}

Cout = AB + BCin + ACin

2.5.10 Approximate adder 10(AA10) :
In this type of adder only sum is approximated.

Sum = A' + BC_{in}

Cout = AB + BCin + ACin

2.5.11 Approximate adder 11(AA11) :
In this type of adder only sum is approximated.

Sum = A' (B' + C' $_{in}$) + A'BC' $_{in}$

Cout = AB + BCin + ACin

2.5.12 Approximate adder 12(AA12) :
In this type of adder only sum is approximated.

Sum = AB + BC'$_{in}$ + A'B' C_{in} + AB' C' $_{in}$

Cout = AB + BCin + ACin

Inputs			Outputs											
			FA		AA1		AA2		AA3		AA4		AA5	
A	B	C_{in}	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	1	1	0	1	0	1	0	1	0	1	0	0×	0
0	1	0	1	0	0×	1×	1	0	0×	1×	0×	0	1	0
0	1	1	0	1	0	1	0	1	0	1	1×	0×	1×	0×
1	0	0	1	0	0×	0	1	0	1	0	0×	1×	0×	1×
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	1×	1
1	1	1	1	1	1	1	0×	1	0	1	1	1	1	1

Note: × denotes incorrect output.

Table 1: Truth table for AA1-AA5.

'Inputs			Outputs													
			FA		AA6		AA7		AA8		AA9		AA10		AA11		AA12	
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	1x	0	0	0	0	0	1x	0	1x	0	1x	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	0x	0	1	0	1	0	1	0	1	0
0	1	1	0	1	1x	0x	1x	1	1x	1	0	1	1x	1	0	1	0	1
1	0	0	1	0	0x	1x	0x	0	0x	0	1	0	0x	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	1x	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0x	1	1	1

Table 2: Truth table for AA6-AA12.

ISSN 2394 - 9554

Adder	Power (Nanowatts)	Delay (Ps)	Area (μm^2)	Power (Nanowatts)	Delay (Ps)	Area (μm^2)	Power (Nanowatts)	Delay (Ps)	Area (μm^2)	Number of errors
Technology	@180nm			@90nm			@45nm			
FA	4443	491	70	1354	179	29	573	75	12	0
AA1	1547	383	47	703	135	21	339	57	9	3
AA2	1078	427	39	527	179	16	203	69	7	2
AA3	565	200	26	268	88	11	121	36	5	4
AA4	707	146	29	341	67	13	163	28	6	5
AA5	154	67	13	68	31	6	31	14	2	6
AA6	350	153	17	165	57	8	78	24	3	5
AA7	1509	253	53	621	119	25	304	49	11	2
AA8	1499	229	40	603	102	17	275	37	8	3
AA9	2443	412	77	1014	183	31	479	78	15	1
AA10	1141	222	47	410	62	21	127	28	4	3
AA11	1069	422	37	389	74	9	104	34	3	2
AA12	2171	386	70	599	96	29	139	53	12	1

Table 3: Comparison of AAs

3. DISCUSSION AND CONCLUSION

In this paper conventional adders and approximate adders are reviewed. Compared to ACA, ESA has lesser hardware overhead and it is less power consuming. However, ETAII predicts the carry bit with more information and hence it is more accurate than ESA. ACA and ETAII have same delays but, the delay of ETAII increases with the increase in the value of k. Applications that tolerate errors find ETAII to be efficient but in applications that require fast computations ESA can be preferred. A smaller delay doesn't always imply lesser power consumption and hence the power delay product is used as a metric to measure the efficiency of an approximate adder. The lesser the power delay product of the adder, the higher is the efficiency. ETAII and ACAA are equally accurate, but in applications that doesn't need runtime accuracy configuration ETAII will serve as a better choice due to its lower power delay product. Among the approximate adders AA1-AA5, AA5 has the maximum logic approximation both in Sum and Cout. Thus, it has the highest performance with the tradeoff in accuracy. Further approximation in accuracy is also possible. AA2 can be used in the applications demanding a higher accuracy. The approximate adders AA6-AA12 have been designed based on the fact that the accuracy of Cout has a greater significance than that of the sum for an adder. This is because an error that occurs in carrying operation propagates through every subsequent stage, thereby amplifying the error. Hence only in AA6 approximation was done in Sum and Cout. This makes AA6 the adder with high probability of error. In AA9 and AA12, less probability of error is observed. AA1-AA5 report lesser performance in terms of area, delay and power than AA6. Not all the approximate adders operate in low power and high speed. Only the adder AA6 observes lesser power and higher speed. AA12 is observed to be a competent adder with higher performance and only one error.

4. REFRENCES

[1] G.Anusha, P.Deepa, Design of Approximate adders and Multipliers for error tolerant image processing, Elsevier, 2019.

[2] H.onglan Jiang, J.ie Han, F.abrizo Lombardi, A comparative review and evaluation of approximate adders, in: proceedings of GLSVLSI, 2015, pp. 343–348.

[3] A.B. Kahng, S. Knag, Accuracy-Configurable adder for approximate arithmetic designs, in: Proceedings of the 49th ACM Annual Design Automation Conference, 2012, pp. 820–825.

[4] V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan, K. Roy, IMPACT imprecise adders for low-power approximate computing, in: Proc. IEEE/ACM international Symposium on Low-power Electron Design, 2011, pp. 409–414.

[5] S.-L. Lu. Speeding up processing with approximation circuits. Computer, 37(3):67–73, 2004.