Hardware Efficient Approximate Adder Design

P. Balasubramanian, Douglas Maskell
School of Computer Science and Engineering
Nanyang Technological University
50 Nanyang Avenue
Singapore 639798
(balasubramanian, asdouglas) @ntu.edu.sg

***Abstract*—This paper presents a new approximate adder architecture which when implemented on an FPGA consumes fewer logic resources compared to accurate adders of similar size and can achieve higher or comparable operating frequencies. For 32-bit addition, our approximate adder achieves a 25% reduction in the number of LUTs utilized compared to the accurate adder with no compromise on the speed performance. For 64-bit addition, our approximate adder achieves a 24% improvement in the maximum operating frequency, and a 25% reduction in the number of LUTs utilized compared to the accurate adder (post place and route on a Virtex-7 FPGA device). We also make comparisons with the FPGA-based implementations of some well-known gate-level approximate adders, and further provide insights into the error characteristics showing that the proposed approximate adder has a reduced error range.**

Keywords*—Approximate computing; Computer arithmetic; Adders; FPGA

I. Introduction

Approximate computing has gained considerable interest due to its ability to reduce the computational complexity, producing higher speed, reduced area, reduced power, and energy efficient digital designs, by trading-off the accuracy of computation. In many applications such as digital image /audio/video processing, data analytics, machine learning and artificial intelligence, etc. [1–4], approximately correct results which conform to a specified error bound are often acceptable. Approximate computing spans both computer hardware [5] and software [6]. In terms of hardware-level approximation, approximate circuits [7] and storage [8] represent the two broad facets of approximate computing, with research into approximate circuits being focused on approximate arithmetic circuits [9] [10], and approximate logic synthesis [11].

Approximate computer arithmetic has gained significant attention in the ASIC community in recent years, with several approximate adders and multipliers proposed in the literature. This is because binary addition and multiplication form the basis of most digital signal processing (DSP) operations. For example, more than 70% of the power consumed by a graphics processing unit is attributed to arithmetic operations [12] and about 80% of the power consumed by an fast Fourier transform processor is attributed to additions and multiplications [13]. DSP is pervasive in consumer and industrial electronics, and therefore the design of approximate adders and multipliers has been pursued vigorously by the VLSI research community.

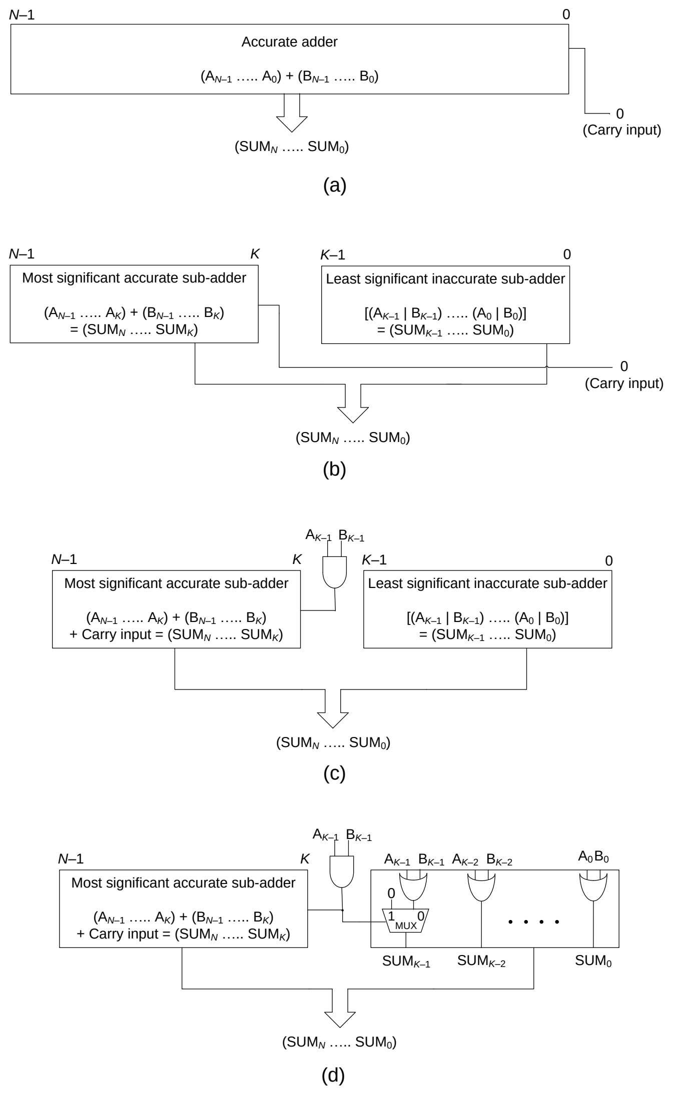
This work is supported by the Singapore Ministry of Education (MOE) Academic Research Fund Tier 2 under grant MOE2017-T2-1-002 and MOE Tier 1 under grant RG132/16.

Although a variety of approximate adders and multipliers have been proposed in the literature, they have mainly been designed and implemented in a full-custom or semi-custom ASIC design style at the circuit or gate level. Occasionally, FPGAs were used, but most times, just to perform functional verification of the approximate adders [14] [15] [24]. FPGAs contain specialized fast carry chain logic in the logic slice for building fast adders and embedded multiplier/DSP circuits which are highly optimized in terms of the design metrics. Hence, custom adders or multipliers when implemented on FPGAs may not be able to surpass the design metrics of accurate native FPGA adders. FPGA-based approximate adders have been examined in [16]. The underlying principle is to segment an adder and share some of the input bits between the two adder parts. Unfortunately, [16] was not compared to accurate LUT-based FPGA adders and compared only with the FPGA implementations of ASIC-oriented approximate adders.

The rest of this paper is organized as follows. Section 2 discusses the architectures of accurate FPGA adders, FPGA-based implementation of some gate-level (i.e. ASIC-based) approximate adders, and the FPGA-based implementation of our proposed approximate adder. Section 3 presents the FPGA-based implementation results of the accurate and approximate adders discussed. Section 4 describes the error characteristics, and Section 5 gives the conclusions.

II. Accurate and Approximate Adders

One of the more common approaches to approximate adder design involves breaking the carry chain by segmenting the accurate adder into multiple smaller parts [17] [18]. In this paper, we only consider approximate adders which are constructed by segmenting an N-bit dual-operand adder into two parts [19] [20], with K-bits allotted for the least significant (inaccurate) sub-adder, and $(N–K)$-bits allotted for the more significant (accurate) sub-adder. In an ASIC technology, the accurate sub-adder could be realized using a high-speed adder architecture such as a carry lookahead [20] [25], while the inaccurate sub-adder could be realized using discrete gates (with or without extra control logic), or even by using a second accurate sub-adder which is detached from the other accurate sub-adder. However, while this approach is likely to be beneficial for ASIC-based designs, it may not be beneficial for FPGA-based designs due to the FPGAs dedicated fast carry chain. In this work, we utilize the native accurate FPGA adder to construct the approximate adders, as shown in Fig 1.



N–1
0
Accurate adder
(A_{N–1} ….. A_0) + (B_{N–1} ….. B_0)
0
(Carry input)
(SUM_N ….. SUM_0)
(a)
N–1
K
K–1
0
Most significant accurate sub-adder
(A_{N–1} ….. A_K) + (B_{N–1} ….. B_K) = (SUM_N ….. SUM_K)
Least significant inaccurate sub-adder
[(A_{K–1} | B_{K–1}) ….. (A_0 | B_0)] = (SUM_{K–1} ….. SUM_0)
0
(Carry input)
(SUM_N ….. SUM_0)
(b)
A_{K–1} B_{K–1}
N–1
K
K–1
0
Most significant accurate sub-adder
(A_{N–1} ….. A_K) + (B_{N–1} ….. B_K) + Carry input = (SUM_N ….. SUM_K)
Least significant inaccurate sub-adder
[(A_{K–1} | B_{K–1}) ….. (A_0 | B_0)] = (SUM_{K–1} ….. SUM_0)
(SUM_N ….. SUM_0)
(c)
A_{K–1} B_{K–1}
N–1
K
A_{K–1} B_{K–1} A_{K–2} B_{K–2}
A_0 B_0
Most significant accurate sub-adder
(A_{N–1} ….. A_K) + (B_{N–1} ….. B_K) + Carry input = (SUM_N ….. SUM_K)
0
1
0
MUX
SUM_{K–1}
SUM_{K–2}
SUM_0
(SUM_N ….. SUM_0)
(d)

Fig. 1. Accurate and approximate N-bit adders
(a) *adder 1* (accurate), (b) *adder 2* [20], (c) *adder 3* [19], (d) *new_adder 4*

Fig 1a shows an accurate adder which can be implemented in the FPGA logic fabric (referred to as *adder 1*). Fig 1b shows an approximate adder (*adder 2*) formed by combining an (N–K)-bits accurate adder part having a carry input of 0 with a K-bits inaccurate adder part. The inaccurate sub-adder uses K 2-input OR functions. The OR functions perform logical disjunction of the corresponding augend and addend bits to produce the respective sum bits. Thus, bit-wise addition is performed in parallel with no connection between any two 2-input OR functions. The approximate adder of Fig 1c (*adder 3*) is a derivative of that shown in Fig 1b, except that the most significant bit pair of the inaccurate sub-adder is AND-ed and used as the carry input for the accurate sub-adder. The two OR-based approximate adders (*adder 2* and *adder 3*) are derived from [20] and [19], except that they propose using carry lookahead adders in the accurate adder part for high-speed and low power [25], which is realized using an ASIC technology. Fig 1d shows our proposed approximate adder (*new_adder 4*), which is a modified version of *adder 3* to primarily improve the error characteristics without sacrificing the performance.

III. Implementation Results

Accurate 32- and 64-bit adders, and approximate 32- and 64-bit adders corresponding to the architectures in Fig 1 were implemented on a Virtex-7 FPGA (part: xc7vx690tffg1761-2) using Xilinx Vivado 2015.1. The approximate adders were all implemented with $K = N/2$. All adders were implemented with a pair of registers on the adder inputs and a register after the adder output. For synthesis, the optimization goal was set as area (*Flow_Area Optimized_high*). For implementation the default strategy was used. All the adders were synthesized, and placed and routed. The adder type and size, FPGA resources used viz. LUTs and flip flops (FFs), the clock period and maximum clock frequency, the datapath delay (for logic and routing), and the error range of the adders are given in Table 1.

Table I. Design metrics (obtained post place and route) and the error range of accurate and approximate 32- and 64-bit adders

Adder Type	**LUTs/ FFs**	**Clock** (ns/MHz)	**Datapath** (ns) Logic/Routing	**Error Range**
Accurate 32-bit adder				
adder 1	32/97	1.41/709	0.987/0.380	0
Approximate 32-bit adders				
adder 2	24/97	1.41/709	0.871/0.362	– (2^K–1) to 0
adder 3	24/97	1.41/709	0.913/0.334	– ($2^{K–1}$–1) to $2^{K–1}$
new_adder 4	24/97	1.41/709	0.913/0.334	– ($2^{K–1}$–1) to 0
Accurate 64-bit adder				
adder 1	64/193	1.92/520	1.459/0.432	0
Approximate 64-bit adders				
adder 2	48/193	1.41/709	1.023/0.372	– (2^K–1) to 0
adder 3	48/193	1.55/645	1.023/0.509	– ($2^{K–1}$–1) to $2^{K–1}$
new_adder 4	48/193	1.55/645	1.023/0.509	– ($2^{K–1}$–1) to 0

From Table 1, we see that all of the LUT-based 32-bit adders (both accurate and approximate) are constrained by the clock network delay (not the logic fabric), and hence the maximum operating frequency is identical (709MHz). The datapath delay in the logic fabric (i.e. combined logic delay and logic routing delay) for the accurate and approximate 32-bit adders are: 1.367ns for *adder 1*; 1.233ns for *adder 2*; 1.247ns for *adder 3*; and 1.247ns for the *new_adder 4*. All approximate adders have a reduced datapath delay, which when the adder is used with other circuits in the logic fabric will likely result in a smaller total worst-case delay (and thus a higher operating frequency) than is possible with the LUT-based accurate 32-bit adder (*adder 1*). The *adder 3* and *new_adder 4* realizations have a slightly larger logic delay than *adder 2* due to the additional AND function in the accurate sub-adder carry chain.

For the 64-bit LUT-based adders, the inaccurate *adder 2* is still constrained by the clock network delay, and has a maximum frequency of 709MHz. All LUT-based approximate adders have a significantly higher maximum operating frequency than the LUT-based 64-bit accurate adder. *adder 2* has a 36.3% increase in speed, and *adder 3* and *new_adder 4* report a 24% increase in speed (due to the additional AND function in the accurate sub-adder carry chain). The datapath delay encountered in the logic fabric for the accurate and approximate 64-bit adders are: 1.891ns for *adder 1*; 1.395ns for *adder 2*; 1.532ns for *adder 3*; and 1.532ns for the *new_adder 4*.

Again, all approximate adders have a reduced datapath delay than the accurate adder.

The resource usage for the various adder implementations is shown in Table 1. The number of LUTs required for *new_adder 4* is the same as for *adder 2* and *adder 3* (irrespective of the addition size), due to the bi-partitioning of the accurate and inaccurate adder parts, as shown in Fig 1. The *adder 2*, *adder 3* and *new_adder 4* implementations require 25% fewer LUTs than the accurate *adder 1* implementation, for both 32- and 64-bit implementations. The FF utilization for the LUT-based adders reflects the size of the registers on the adder inputs and outputs.

In [16], a LUT-based approximate adder named FAU was implemented using the same version of Xilinx Vivado and targeting the same Virtex-7 FPGA device. For 32-bit addition, FAU [16] using a 16-16 input partition, with 1 or 3 shared inputs between adjacent LUTs, consumes around 50 LUTs. In contrast, as seen in Table 1, the accurate *adder 1* consumes only 32 LUTs while *adder 2*, *adder 3*, and *new_adder 4* consume only 24 LUTs. The maximum frequency reported for FAU is 328.19MHz for 32-bit addition and 281.06MHz for 64-bit addition. In contrast, the accurate *adder 1* and the approximate *adder 2*, *adder 3* and *new_adder 4* exhibit a 116% increase in the maximum frequency for 32-bit addition. For 64-bit addition, the accurate *adder 1* reports an 85% increase in the maximum frequency, and the approximate *adder 2*, *adder 3* and *new_adder 4* report 152.3%, 129.5% and 129.5% improvements in the maximum frequency compared to FAU.

In [16], FPGA-based implementations of the approximate adders given in [21–24] were considered and it was concluded that they all consumed more resources than the FAU. This is quite understandable given that [21–24] present ASIC-based approximate adder designs and so implementing the same on FPGAs would require more resources. This is because [21] incorporates custom control logic to produce the approximate sum bits, [22] includes carry prediction and control units to permit graceful degradation in the accuracy of results, and [23] and [24] include extra error detection and/or correction circuits to adjust the accuracy of results based on need. Additionally, [22–24] are dynamic designs which can either produce the accurate result or an approximate result subject to a specified accuracy. However, the provision of error detection and correction circuits in dynamic designs may require multiple clock cycles to produce the desired result. It was observed in [26] that although the extent of approximate computation could be varied on the fly, the power savings for the dynamic approximation computing hardware is comparable with that of static approximation computing hardware. This is mainly due to the overhead associated with the dynamic approximation logic in the former. Additionally, FAU is not power efficient compared to the FPGA-based implementation of approximate adders given in [22, 23].

Compared to the LUT-based approximate adder FAU [16], all of the LUT-based adder implementations discussed here (i.e. both accurate and inaccurate) have a smaller LUT utilization and a significantly higher operating frequency. An inaccurate adder is unlikely to be used in an FPGA if it is unable to outperform a native accurate adder in at least some of speed or resource utilization or power.

IV. Error Analysis

We now examine the error characteristics of the approximate adders described in Section 2. Many error metrics have been proposed for comparison purposes, including mean error (ME), mean absolute error (MAE), mean error distance[1] (MED), root mean squared error (RMSE), etc. We use RMSE as an error metric, rather than MED, as RMSE gives a relatively higher weight to larger errors, as these are more likely to impact an application using inaccurate arithmetic. The error metrics are defined as:

$$\mathrm{MAE} = \mathrm{MED} = \frac{1}{2^{2N}} \sum_{j=0}^{2^{2N}-1} |e_j| = \sum_{\delta} |e_\delta| \cdot P_\delta \tag{1}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{2^{2N}} \sum_{j=0}^{2^{2N}-1} e_j^2} = \sqrt{\sum_{\delta} e_\delta^2 \cdot P_\delta} \tag{2}$$

where N is the adder bit-width, e is the error (the difference between the inaccurate and the accurate adder output), P is the probability of an error value occurring, while δ is the set of all error values.

When an adder is partitioned into an (N–K)-bit accurate adder and a K-bit inaccurate adder, the errors will range somewhere between -2^K and $+2^K$. For the three inaccurate adder types presented in Fig. 1, the error range is shown as the error bars in Fig. 2. Fig. 2 also shows the error distribution (in terms of the *% Occurrence* of a particular error value) for a 12-bit implementation using a 6-bit inaccurate sub-adder, assuming a uniform input distribution.

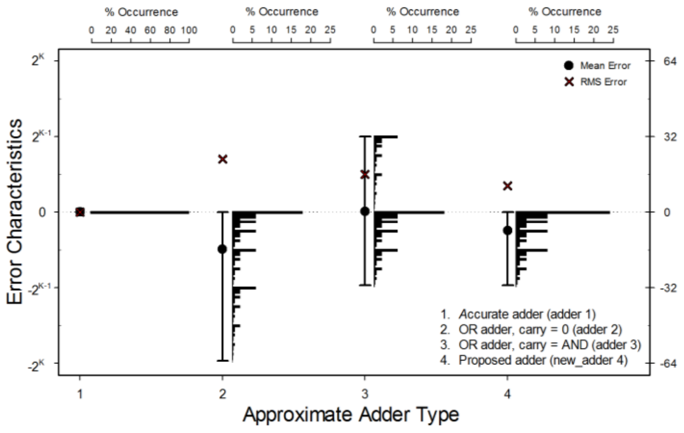


Fig. 2. The error range and distribution of the 4 adders

For the OR-based adder with carry input = 0 (*adder 2*), the error range is between $-(2^K-1)$ and 0, as shown in Fig. 2. The error characteristics for the 6-bit *adder 2* are: Minimum (Min) error = -63; Maximum (Max) error = 0; MED = -ME = 15.75; and the RMSE is 22.45. The OR-based adder where the carry

[1] MED and MAE are identical, with MED being used in the approximate computing literature rather than the more common MAE.

is the AND of the MSBs of the inaccurate sub-adder (*adder 3*) has an error distribution which is better distributed about 0, with an error range between $-(2^{K-1}-1)$ and 2^{K-1}. The error characteristics for the 6-bit *adder 3* are: Min error = -31; Max error = 32; ME = 0.25; MED = 11.88; and the RMSE is 16.0. The proposed adder (i.e., *new_adder 4*) has half of the error distribution of the other inaccurate adders, which has a range of $-(2^{K-1}-1)$ to zero. The error characteristics for the 6-bit *new_adder 4* are: Min error = -31; Max error = 0; MED = -ME = 7.75; and the RMSE is 11.14.

The smaller RMSE of the proposed *new_adder 4* (compared to *adder 2* and *adder 3*) indicates that it is a much better implementation from an adder error perspective. Fig. 2 also shows that most of the errors fall within the range between $-(2^{K-3}-1)$ to zero, which contribute to the reduced RMSE.

The proposed 32-bit *new_adder 4* implemented as a K-bit inaccurate adder ($K = N/2 = 16$) with an MED of 8191.75 has a better MED and LUT utilisation than the adders in [21–24], as presented in Table 1 of [16]. However, when compared to the FAU adder [16], the proposed adder (with 16-bit inaccurate part) has a significantly worse MED, but with a significantly reduced LUT utilisation and a higher operating frequency. Changing the size of the inaccurate adder significantly impacts the MED, as can be seen in Table 3. In Table 3, p signifies the number of shared input bits between adjacent LUTs in the FAU [16]. Comparing FAU ($p = 1$) with the proposed adder ($K = 14$), (and the $p = 3$ FAU with the proposed $K = 10$ adder), shows that the proposed adder outperforms FAU on all metrics, as shown in Table 3.

TABLE III. COMPARISON WITH THE FAU ADDER OF [16]

Adder Type	**Adder Config.**	**LUTs**	**Clock** (MHz)	**MED**
Approximate 32-bit adders				
FAU [16]	$p = 1$ [16]	50	321	2731.67
FAU [16]	$p = 3$ [16]	52	328	171.67
new_adder 4	$K = 10$	27	709	127.75
new_adder 4	$K = 14$	25	709	2047.75
new_adder 4	$K = 16$	24	709	8191.75

V. CONCLUSIONS

A new approximate adder architecture (*new_adder 4*) that is suitable for FPGA and ASIC-style implementations was presented. Comparisons were made with the implementations of some well-known approximate adders given in [20], [19] (i.e., *adder 2* and *adder 3*), the accurate adder (*adder 1*), and an FPGA-based approximate adder (FAU) [16]. It was noted that the *new_adder 4* is hardware efficient and is comparable to *adder 2* and *adder 3* in terms of the design metrics for FPGA implementations while featuring a reduced error range. Comparing the FPGA implementations of the accurate *adder 1* and the proposed *new_adder 4* (Table 1) shows that the proposed adder requires 25% less LUTs for 32-bit addition while providing the same speed performance. For the FPGA 64-bit addition, *new_adder 4* reports a 24% improvement in the maximum frequency and requires 25% fewer LUTs compared to the accurate *adder 1*.

Compared to $p = 1$ FAU, the $K = 16$ *new_adder 4* uses 52% fewer LUTs for the 32-bit adder (the results for the 64-bit adder are not provided in [16]). In terms of the maximum operating frequency, *new_adder 4* outperforms FAU by 116% and 129.5% for 32- and 64-bit additions, respectively. Implementing the proposed adder so that it achieves a similar MED to FAU results in it outperforming FAU on all metrics.

Future work will consider utilizing the proposed approximate adder in computation units used in practical DSP applications.

REFERENCES

[1] M.A. Breuer, "Multi-media applications and imprecise computation," *Proc. 8th Euromicro Conference on Digital System Design*, pp. 2-7, 2005.

[2] H.-Y. Cheong, I.S. Chong, A. Ortega, "Computation error tolerance in motion estimation algorithms," *Proc. International Conference on Image Processing*, pp. 3289-3292, 2006.

[3] R. Nair, "Big data needs approximate computing: technical perspective," *Communications of the ACM*, vol. 58, pp. 104, 2015.

[4] M. Shafique, R. Hafiz, M.U. Javed, S. Abbas, L. Sekanina, Z. Vasicek, V. Mrazek, "Adaptive and energy-efficient architectures for machine learning: challenges, opportunities, and research roadmap," *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 627-632, 2017.

[5] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, J. Henkel, "Cross-layer approximate computing: from logic to architectures," *Proc. 53rd ACM/EDAC/IEEE Design Automation Conference*, pp. 1-6, 2016.

[6] A. Sampson, W. Dietl, D. Fortuna, D. Gnanapragasam, L. Ceze, D. Grossman, "EnerJ: approximate data types for safe and general low-power computation," *ACM SIGPLAN Notices*, vol. 46, pp. 164-174, 2011.

[7] K. Roy, A. Raghunathan, "Approximate computing: an energy-efficient computing technique for error resilient applications," *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 473-475, 2015.

[8] A. Sampson, J. Nelson, K. Strauss, L. Ceze, "Approximate storage in solid-state memories," *ACM Transactions on Computer Systems*, vol. 32, pp. 1-9, 2014.

[9] Z. Yang, A. Jain, J. Liang, J. Han, F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," *Proc. IEEE International Conference on Nanotechnology*, pp. 690-693, 2013.

[10] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, J. Han, "A comparative evaluation of approximate multipliers," *Proc. IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 191-196, 2016.

[11] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," *Proc. Design Automation Conference*, pp. 796-801, 2012.

[12] H. Zhang, M. Putic, J. Lach, "Low power GPGPU computation with imprecise hardware," *Proc. 51st Annual Design Automation Conference*, pp. 1-6, 2014.

[13] L. Wanhammar, *DSP Integrated Circuits*. Academic Press, USA, 1999.

[14] K. Shi, G.A. Constantinides, "Evaluation of design trade-offs for adders in approximate datapath," *Proc. HiPEAC Workshop on Approximate Computing*, pages 6, 2015.

[15] A. Raha, H. Jayakumar, V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Transactions on VLSI Systems*, vol. 24, no. 3, pp. 846-857, 2016.

[16] J. Echavarria, S. Wildermann, A. Becher, J. Teich, D. Ziener, "FAU: fast and error-optimized approximate adder units on LUT-based FPGAs," *Proc. International Conference on Field-Programmable Technology*, pp. 213-216, 2016.

[17] N. Zhu, W.L. Goh, W. Zhang, K.S. Yeo, Z.H. Kong, "Design of low-power high-speed truncation-error-tolerant adders and its application in digital signal processing," *IEEE Transactions on VLSI Systems*, vol. 18, pp. 1225-1229, 2010.

[18] S. Mahazir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, "Probabilistic error modeling for approximate adders," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 515-530, 2017.

[19] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, pp. 850-862, 2010.

[20] P. Albicocco, G.C. Cardarilli, A. Nannarelli, M. Petricca, M. Re, "Imprecise arithmetic for low power image processing," *Proc. 46th Asilomar Conference on Signals, Systems and Computers*, pp. 983-987, 2012.

[21] N. Zhu, W.L. Goh, K.S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," *Proc. 12th International Symposium on Integrated Circuits*, pp. 69-72, 2009.

[22] R. Ye, T. Wang, F. Yuan, R. Kumar, Q. Xu, "On reconfiguration-oriented approximate adder design and its application," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 48-54, 2013.

[23] A.K. Verma, P. Brisk, P. Ienne, "Variable latency speculative addition: a new paradigm for arithmetic circuit design," *Proc. Design, Automation and Test in Europe*, pp. 1250-1255, 2008.

[24] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, "A low latency generic accuracy configurable adder," *Proc. 52nd ACM/EDAC/IEEE Design Automation Conference*, pp. 1-6, 2015.

[25] P. Balasubramanian, C. Dang, D.L. Maskell, K. Prasad, "Approximate ripple carry and carry lookahead adders – a comparative analysis," *Proc. IEEE 30th International Conference on Microelectronics*, pp. 299-304, 2017.

[26] A. Raha, H. Jayakumar, V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Transactions on VLSI Systems*, vol. 24, pp. 846-857, 2016.