Input-Aware Approximate Computing

Input-Aware Approximate Computing

Ali Piri[1], Sepide Saeedi[2], Mario Barbareschi[3], Bastien Deveautour[1], Stefano Di Carlo[2]
Ian O'Connor[1], Alessandro Savino[2], Marcello Traiola[4], Alberto Bosio[1]
[1]*Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, 69130 Ecully, France*
[2]*Politecnico di Torino, Dip. di Automatica e Informatica, Torino, Italy*
[3]*Dep. of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy*
[4]*University of Rennes, Inria, CNRS, IRISA, UMR6074*
Email: [1]{name.surname}@ec-lyon.fr, [2]{name.surname}@polito.it
[3]mario.barbareschi@unina.it, [4]marcello.traiola@inria.fr

***Abstract*—In the last decade, Approximate Computing (AxC) has been extensively employed to improve the energy efficiency of computing systems, at different abstraction levels. The main AxC goal is reducing the energy budget used to execute error-tolerant applications, at the cost of a controlled and intrinsically-tolerable quality degradation. An important amount of work has been done in proposing approximate versions of basic operations, using fewer resources. From a hardware standpoint, several approximate arithmetic operations have been proposed. Although effective, such approximate hardware operators are not tailored to a specific final application. Thus, their effectiveness will depend on the actual application using them. Taking into account the target application and the related input data distribution, the final energy efficiency can be pushed further. In this paper we showcase the advantage of considering the data distribution by designing an input-aware approximate multiplier specifically intended for a high pass FIR filter, where the input distribution pattern for one operand is not uniform. Experimental results show that we can significantly reduce the power consumption while keeping an error rate lower than state of the art approximate multipliers.**

***Index Terms*—Approximate Computing, Energy Efficiency, Embedded Systems, Arithmetic Circuits**

I. Introduction

The well-known Moore's law drove the wide world explosion of the integrated circuits market and the related exponential growth of the Information and Communication Technologies (ICT). Computers are now everywhere and managing all aspects of our life: from health monitoring, work, entertainment, home management, etc. Unfortunately, behind all the advantages, the dark face of ICT performances stems from the high energy consumption of digital circuits. Indeed, ICT devices and services are responsible for a substantial percentage of the total energy consumed in the world [1]. Even worse, the amount of required energy is expected to grow to almost 21% by 2030 [1]. A direct consequence is the huge research field dedicated to the design, development and fabrication of energy efficient digital circuits. The quest for emerging technologies and computing paradigms in the light of energy efficiency graal provided meaningful solutions so far [2], [3].

Among them, Approximate Computing (AxC) proved to be a very promising one [4]. The idea behind AxC is that several applications do not really need to be executed on a "precise" and thus "energy-expensive" hardware. AxC aims at reducing the precision of the hardware in order to save energy consumption. Interestingly, the reduced precision leads to applications providing less accurate, but still good enough results while reducing by orders of magnitude the required energy [5], [6]. Such applications are characterized to be intrinsically resilient to noise and errors affecting the computation (i.e., because of the less precise hardware). Indeed, the inherent resiliency property tightly depends on the application domain.

Well-known examples are algorithms dealing with noisy real-world input data (e.g., image processing, sensor data processing, speech recognition, etc.), or with outputs that require human interpretation, such as digital signal processing of images or audio; also data analytic, web search and wireless communications exhibit an equivalent property [7]–[9]. Other examples are iterative applications that process large amounts of information, sample data, stop the convergence procedure early, or apply heuristics [10]. Most of the proposed techniques try to define new methods to generate alternative versions of specific component (either hardware or software) with fewer resources. For example, there are several proposals of approximate arithmetic operations [11]–[13]. Such variants differ from speculative implementations because they do not focus on generating alternatives, rather on restoring the possible introduced error [14]–[16]. Other techniques generate variants by considering a high-level description of the application or its implementation at low-level [7]. Moreover, existing approaches target only implementations at a specific level of the computing stack, i.e. either software or hardware.

Even if the above existing techniques proven to be effective, they have been developed without considering the final application. In other words, they are not customized w.r.t the application and its workload (i.e., input data). We thus believe that there is the room to introduce a novel and promising approach to power efficiency based on the knowledge of the input data distribution and the target application. Indeed, there are several applications where the inputs do not follow a uniform distribution pattern. For instance, an image or signal processing application that is always working in similar environments with a limited range of inputs.

Let us resort to an example to clarify this point. We used the LeNet-5 [17] Convolution Neural Network (CNN) that is composed of 3 convolutional layers (CONV) followed by 2 fully connected (FC) layers, with a total of 61,470 parameters. To showcase the concept, we profiled the inference execution of LeNet-5 when the 10,000 MNIST test images are applied and we obtained the data distribution per bit and per layer shown in Figure 1. On the X-axis, we show the 32 bits from LSB (bit 0) up to the MSB (bit 31). On the Y-axis, we show the probability that the bit i is equal to logic '1'. As a first comment, it can be noted that the distribution is quite similar for the layers. This means that it is possible to design an approximate arithmetic circuit to be used in all layers (thus simplifying the overall design). A second comment is about the fact that from bit 0 to 6 the probability to have logic '1' is quite low (smaller the 20%). Interestingly, also the MSBs have low probability to be logic '1', especially for layer 0 and layer 4. These results confirm that it is possible to aggressively approximate the arithmetic circuits not only by working on the LSBs (as it is usually done) but also through the MSBs.

The main goal of this paper is to show that it is possible to take advantage of the data distribution to obtain a fine-tuned approximation and thus achieve better results (i.e., higher energy efficiency) with a lower impact on the application accuracy. In particular, by characterizing the input pattern distribution, we can design an approximate version of a given circuit exploiting the fact that some inputs with similar characteristics are elaborated more frequently than others. Therefore, in this work we present an input-aware approximate multiplier specifically designed for a case study design (high pass FIR filter), where the input distribution pattern for one operand is not uniform. The proposed design allows us to significantly reduce the power consumption while keeping an error rate lower than state of the art approximate multipliers.

The rest of the paper is organized as follows. Section II describes the state of the art of approximate multipliers. Section III details the application case study while Section IV presents the proposed Input-Aware Approximate multiplier tailored for the case study. Section V depicts the results and presents a comparison with state of the art multipliers. Finally, Section VI concludes the paper.

II. Related work

Generally, a conventional multiplication operates in three steps. In the first step, the partial products (PP) are generated by multiplying the multiplicand and the multiplier. In the second step, the PP tree is reduced by accumulation until only two rows remain. In the final step, the remaining two rows are summed by employing a carry propagation adder [18]. The approximation can be applied to each of these steps. Authors in [19] proposed an approximate 2x2 multiplier which is used to compose larger multipliers as shown in Figure 2. In this 2x2 multiplier accuracy reduction happens when both the inputs are "11" so instead of "1001" the output is "111". This way, the output is reduced to three bits and the circuit is simplified. Another approximation method is applied in the PP tree. For example [20] introduces an array multiplier where the least significant carry-save adders are removed from the circuit, both horizontally and vertically. Another simpler method has been applied in [21] where the LSBs from the inputs are truncated. In the PP perforation-based multiplier several consecutive rows of PPs are removed that are not necessarily from the LSBs. In [22] an approximate Wallace tree multiplier is presented which utilizes a carry-in prediction and a bit-width-aware approximate multiplication. In this design, the n-bit multiplier is implemented by four $n/2$ bit submultipliers, and the most significant submultiplier ($A_H B_H$) is divided again into four $n/4$ bit ones. The $n/4$ bit multipliers can have different accuracies and the three remaining less significant multipliers ($A_H B_L$, $A_L B_H$, and $A_L B_L$) are approximate.

A different approach relies on inaccurate counters and compressors in the PP reduction stage. An approximate 4x4 multiplier is presented in [23] that uses an inaccurate 4:2 counter for PP reduction as shown in Figure 3. This counter only results in a wrong answer when all the inputs are '1'. So instead of "100", the result is "10". If the input distribution is uniform the error rate for this 4x4 multiplier is 1/256. Larger multipliers can be built using this 4x4 approximate multiplier. In [24] a novel approximate adder is used to accumulate the PP tree by generating a sum and an error bit out of two adjacent inputs. Then an error recovery scheme is applied to accumulate the error bits in the final result either using only OR gates or approximate adders as well. Also, a truncated version of the same multiplier is presented in [25], [26] where the lower half of the PP is carved out of the circuit. In [27], the authors proposed a hybrid partial product-based approximate 4x4 multiplier that uses the methods of approximating in both the PP tree accumulation and removing insignificant PP bits.This 4x4 multiplier is used to build larger blocks of multipliers. They have altered the PP tree by turning the PP elements using propagate and generate function and removed the PPs with less probability of being logic '1'. They have also presented a novel half adder and full adder to accumulate the PP tree.

Next section describes the application used as case study and its characterization to obtain the input pattern distribution information.

III. Case study:FIR filter

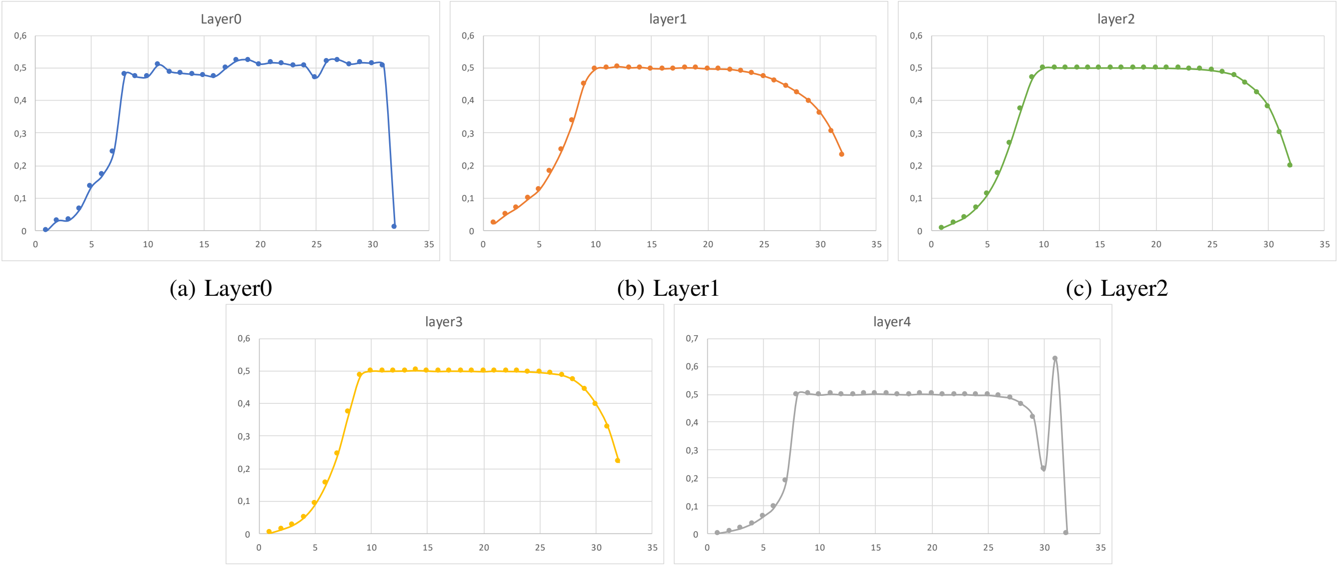
The case study is an finite impulse response (FIR) filter described by Eq. 1.

$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i] \qquad (1)$$

where:

- $x[n]$ is the input signal:
- $y[n]$ is the output signal;
- N is the filter order. In our case study $N = 52$;
- b_i is the value of i^{th} coefficient of the filter.

The FIR used as case study came from an audio application, an in particular the coefficients have been designed to implement a high pas filter (i.e., 500MHz). Coefficients as



(a) Layer0

(b) Layer1

(c) Layer2

(d) Layer3

(e) Layer4

Fig. 1: Data distribution at bit per layer.

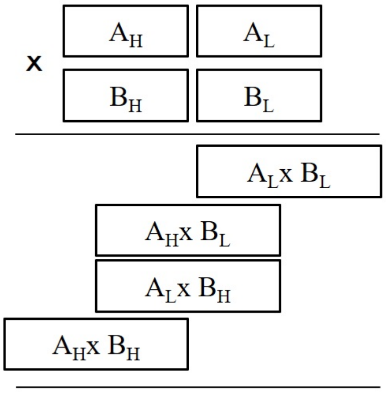


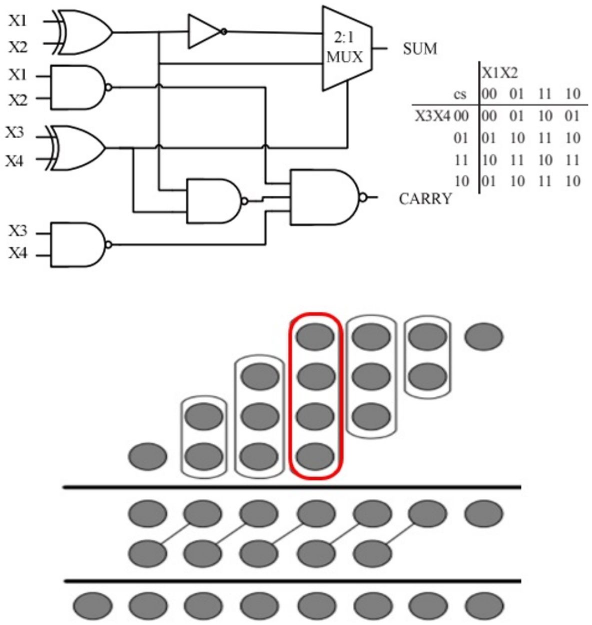
Fig. 2: building *NxN* multiplier using *N/2xN/2* submultiplier

well as input data (i.e., $x[n]$) are encoded as signed 32 bit fixed point with one bit (i.e., the MSB) for the integer part and the remaining 15 bits for the fractional part. Fig. 4 shows the architecture of the filter. On this architecture we intend to approximate the multipliers (i.e., $b_i \cdot x[n-i]$). First, we profiled the coefficients since they are the constant inputs for the multiplier and we reported the data distribution in Table I. The next section will detail the table and how that is used to design the approximate multiplier.

IV. Input Aware Approximate Multiplier

As already mentioned, the proposed multiplier intends to exploit the peculiar input distribution for the FIR high pass filter. In particular, for one of the multiplication operands, five bits are set to logic '1' 100% of the time (B_{10} to B_{14}); moreover, the MSB has a 98% probability to be '1'. The complete input distribution pattern is reported in Table I.

The inputs in this filter are fixed-point 16-bit binary numbers with one bit for the integer part and 15 bits for the fraction.



	cs	X1X2 00	01	11	10
X3X4	00	00	01	10	01
	01	01	10	11	10
	11	10	11	10	11
	10	01	10	11	10

Fig. 3: Structure inexact 4:2 counter and the 4x4 multiplier [23]

The multiplication for the fixed point numbers is the same as the unsigned multiplier. However, since there are 5 bits always set to '1' (B_{10} to B_{14}), there is no need to calculate the relative PPs. Instead, we can directly use the other operand's bit values. The resulting partial product tree for this multiplier is sketched in Figure 5.

If we consider input A as the one with a non-uniform distribution pattern, the squares are representing the AND gates generating the PP $a_i b_j$. The circles represent the related

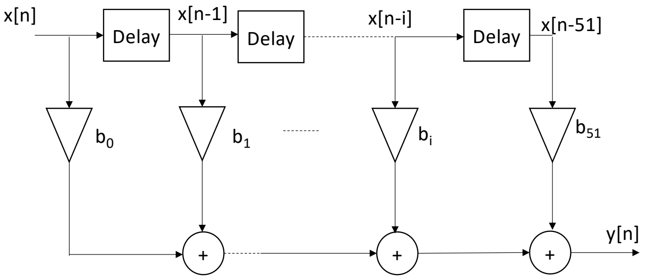


Fig. 4: FIR architecture

TABLE I: Input distribution of the Hamming high pass filter expressed as probability to be '1' for each bit B_i.

B_{15}	B_{14}	B_{13}	B_{12}	B_{11}	B_{10}	B_9	B_8
0.98	1.00	1.00	1.00	1.00	1.00	0.67	0.76
B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
0.59	0.57	0.41	0.49	0.39	0.45	0.55	0.63

input bits of b because inputs $a[14 : 10]$ are always '1' in this application. In this way, the number of the AND gates needed to generate the PP tree is reduced by 80% without any accuracy loss.

To approximate the multiplier, we have adopted the 4x4 multiplier design proposed in [23], introduced in Section II and shown in Figure 3. We used them to build the 8x8 multiplier illustrated in Figure 6. In particular, the least significant sub-multiplier ($A_L B_L$) is removed and an accurate 4x4 Wallace multiplier is used for the most significant part ($A_H B_H$) to maintain acceptable accuracy.

The outputs of the 4x4 sub-multipliers are accumulated using accurate adders. Then, to implement the 16x16 multiplier, we used the approximate 8x8 multiplier for the two least significant parts of the multiplier ($A_L B_L$ and $A_L B_H$). However, for the two more significant parts ($A_H B_H$ and $A_H B_L$) we have adopted an input-aware accurate Wallace multiplier with the related AND gates removed. In this design, the related b input bits are directly used as the PP since $a[14 : 10]$ are always at '1'. The design is sketched in Figure 7a. The final accumulation step is done with exact adders as shown in Figure 7b.

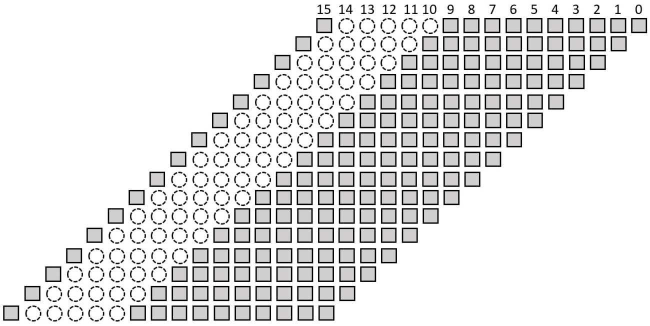


Fig. 5: partial product tree for the Input aware multiplier with five columns of AND gates removed

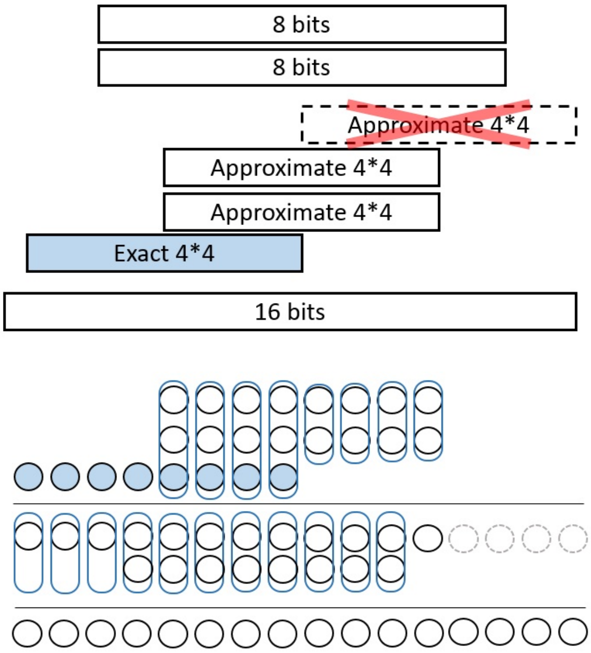
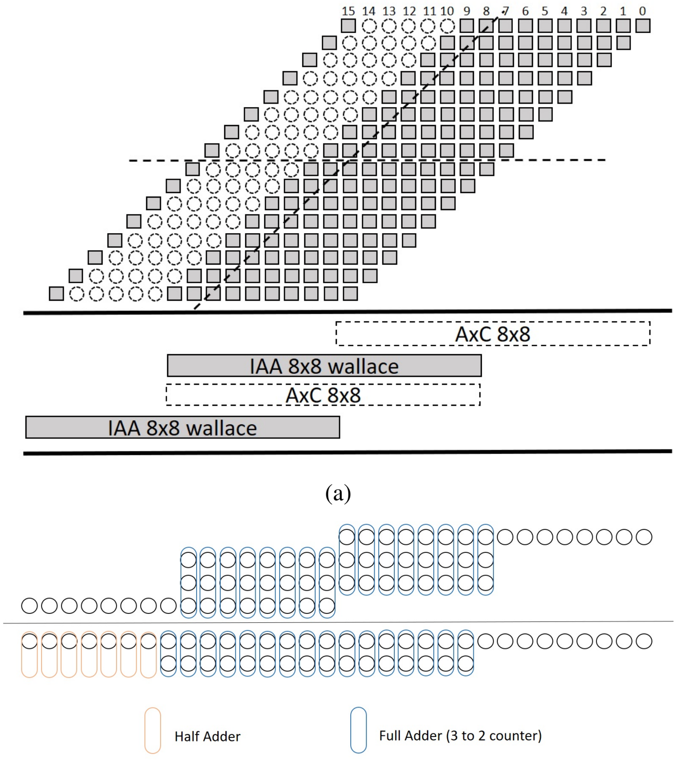


Fig. 6: Structure of the 8x8 multiplier and the accumulation step.

Since the probability of being '1' for the MSB in the input for this high pass filter is 98%, the final column of the PP tree is also a good candidate for removing the related AND gates. Although it is very unlikely that the mentioned bit to be '0', when this happens the Error Distance (ED) can be very high for this approximate method. We have also included this design in our comparison and the results are presented in the next section.

V. Experimental Results

For the circuit evaluation, we compared our 16-bit input-aware approximate multipliers with some of the multipliers from the EvoApproxLib [28] library. Besides the presented IAA multiplier, we have also simulated an accurate version for this multiplier where the PP tree (IAM_16) is missing the respective AND gates. Since the probability of being '1' for the MSB in the input a is 98%, another version of the multiplier is also implemented and the last column of the AND gates is removed as well (IAM_16_V2). Three Wallace multipliers are also implemented and included in the comparison: (i) the conventional Wallace tree, (ii) an input aware Wallace tree with 5 columns on AND gates removed, and (iii) the one with the MSB column removed as well. We have calculated the Power consumption, delay, and area of these 16-bit multipliers by synthesizing them using the Synopsys Design Compiler (DC) at 45nm. The designs were done in Verilog at the gate level. These results are presented in Table II.



(a)

(b)

Fig. 7: Structure of the 16x16 multiplier and the accumulation step.

In order to assess the quality of the approximation, the evaluation employed the following error metrics:

- The error distance (ED): is the arithmetic difference between the accurate result and the result from the multiplier for a given input.
- The mean error distance (MED): is the mean of all possible EDs.
- The error rate (ER): is the probability of producing an incorrect result.
- The normalized mean error distance (NMED): is the normalization of MED by the maximum output of the accurate design.
- the mean relative error distance (MRED), which is the average value of all possible REDs.

TABLE II: Power, Area and Delay for the synthesised multipliers

	Power (mW)	Area(μM^2)	Delay
IAAxC	2.9126	4731.95	3.07
IAM	3.3442	4968.95	3.31
IAM_V2	3.2061	4704.26	3.31
wallace	3.6962	5365.98	3.23
IA_Wallace	3.5187	4962.85	3.2
IA_Wallace_V2	3.3989	4753.54	3.27
mul16u_5FA [28]	3.3331	5663.51	2.74
mul16u_AQ1 [28]	3.5449	6019.24	2.82
mul16u_BMC [28]	3.4873	5963.40	2.86
mul16u_CK3 [28]	1.9708	3927.57	2.99
mul16u_DAE [28]	2.9774	5250.06	2.61
mul16u_F6B [28]	2.6541	4749.79	2.69

For the error analysis, we have simulated the multipliers using the dataset from the hamming high pass filter as one of the operands and uniform random input for the other. As mentioned before, the inputs for the multiplier of this application are fixed-point binary numbers where MSB is the only bit on the left side of the point and the remaining 15 bits are in the fractional part. The multiplication process for the fixed point binary numbers is the same as the unsigned multiplication. This means that we can ignore the binary point of a and b, perform the multiplication, and put the binary point to the left of the 30th bit of the output to obtain the correct multiplication result. In Table III the error rate for the simulated multipliers is reported.

TABLE III: Accuracy comparison

	ER%	ED	MRED	NMED
IAAxC	92.16%	1.292E-05	1.113E-05	3.009E-15
IAM	0.00%	0	0	0
IAM_V2	1.96%	0.03519	0.00990	8.193E-12
wallace	0.00%	0	0	0
IA_Wallace	0.00%	0	0	0
IA_Wallace_V2	1.96%	0.03519	0.0099	8.19378E-12
mul16u_5FA [28]	100.00%	2.472E-08	2.901E-08	5.75708E-18
mul16u_AQ1 [28]	0.00%	1.059E-09	1.489E-09	2.466E-19
mul16u_BMC [28]	0.00%	0	0	0
mul16u_CK3 [28]	100.00%	3.208E-05	3.314E-05	7.47E-15
mul16u_DAE [28]	100.00%	1.843E-07	1.967E-07	4.291E-17
mul16u_F6B [28]	100.00%	2.356E-06	2.387E-06	5.486E-16

Our Input Aware approximate multiplier (IAAxC) has very low power consumption compared to the other multipliers and showed better accuracy if compared to the ones with less power consumption. If we only compare the area for the different versions of the Wallace multiplier we can see it is reduced by 7.5% since we have removed the AND gates respective to the non-uniform input distribution pattern. Meanwhile, there is no accuracy loss in Input aware versions of the Wallace and segmented multiplier (IA_Wallace and IAM). In the second version of these multipliers (IAM_V2 and IA_Wallace_V2), where the final column of AND gates in the PP tree is removed as well, the probability of the MSB in a to be '0' is very low (1.98%), it results in a big error distance. However, in many applications, a low Error Rate is more important than low error distances.

VI. Conclusions

In this work, we have proposed the concept of Input Aware approximation which is a promising approach for having more efficient computers. Given the chance of knowing the data set the system is dealing with, we can design and approximate it knowing the fact that the input is following a specific behavior. The example we have adopted in this work is only one of the many applications with such a working environment where the input distribution is not uniform. For future works, we are planning to use the IAA approach for a larger number of applications with more possibility of approximating. There are some works in progress for automating the approximation where the computer will decide which part of a design and with which method should it be approximated. Input distribution patterns also can be included in this process of decision-making.

Acknowledgment

This work has received funding from the APROPOS project in the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956090.

References

[1] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, vol. 561, no. 7722, pp. 163–166, Sep. 2018. [Online]. Available: https://doi.org/10.1038/d41586-018-06610-y

[2] S. Rai, M. Liu, A. Gebregiorgis, D. Bhattacharjee, K. Chakrabarty, S. Hamdioui, A. Chattopadhyay, J. Trommer, and A. Kumar, "Perspectives on emerging computation-in-memory paradigms," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1925–1934.

[3] I. O'Connor, A. Poittevin, S. Le Beux, A. Bosio, Z. Stanojevic, O. Baumgartner, C. Mukherjee, C. Maneux, J. Trommer, T. Mikolajick, and G. Larrieu, "Analysis of energy-delay-product of a 3d vertical nanowire fet technology," in *2021 Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EuroSOI-ULIS)*, 2021, pp. 1–4.

[4] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing [point of view," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 394–399, 2020.

[5] S. Barone, M. Traiola, M. Barbareschi, and A. Bosio, "Multi-objective application-driven approximate design method," *IEEE Access*, vol. 9, pp. 86 975–86 993, 2021.

[6] M. Barbareschi, S. Barone, A. Bosio, J. Han, and M. Traiola, "A genetic-algorithm-based approach to the design of dct hardware accelerators," *J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 3, jan 2022. [Online]. Available: https://doi.org/10.1145/3501772

[7] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze, and M. Oskin, "Accept: A programmer-guided compiler framework for practical approximate computing," *University of Washington Technical Report UW-CSE-15-01*, vol. 1, no. 2, 2015.

[8] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[9] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 26, no. 1, pp. 97–107, 2014.

[10] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," *Commun. ACM*, vol. 58, no. 1, p. 105–115, Dec. 2014. [Online]. Available: https://doi.org/10.1145/2589750

[11] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*. IEEE Press, 2011, pp. 409–414.

[12] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 1, pp. 124–137, 2013.

[13] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *Computers, IEEE Transactions on*, vol. 62, no. 9, pp. 1760–1771, 2013.

[14] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, April 2009, pp. 664–669.

[15] ——, "Modular inversion based on digit-level speculative addition," *Electronics Letters*, vol. 49, no. 25, pp. 1609–1610, December 2013.

[16] ——, "Variable-latency signed addition on fpgas," in *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*, Sept 2015, pp. 1–6.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[18] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 4, pp. 1–34, 2017.

[19] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th Internatioal Conference on VLSI Design*. IEEE, 2011, pp. 346–351.

[20] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and 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, no. 4, pp. 850–862, 2009.

[21] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105–3117, 2016.

[22] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Fifteenth International Symposium on Quality Electronic Design*. IEEE, 2014, pp. 263–269.

[23] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 33–38.

[24] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[25] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189–202, 2018.

[26] C. Liu, "Design and analysis of approximate adders and multipliers," 2014.

[27] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, "Hybrid partial product-based high-performance approximate recursive multipliers," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[28] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 258–261.