VADER: Voltage-Driven Netlist Pruning for Cross-Layer Approximate Arithmetic Circuits

Georgios Zervakis, Konstantina Koliogeorgi, Dimitrios Anagnostos, Nikolaos Zompakis, and Kostas Siozios

***Abstract*— Leveraging the inherent error resilience of a large number of application domains, approximate computing is established as an efficient design alternative to improve their energy profile. In this brief, we design energy optimal cross-layer approximate arithmetic circuits by enabling the efficient application of voltage overscaling (VOS). Departing from the conventional approaches followed today, we introduce the voltage-driven functional approximation and present the VoltAge-Driven nEtlist pRuning (VADER) framework. VADER is an automated synthesis framework that can be seamlessly integrated in any hardware design flow and implements a voltage-driven gate-level netlist pruning. Experimental evaluation shows that VADER reduces the error of the VOS application by 52% on average and delivers on average designs with 34% higher energy savings compared to state-of-the-art approximate adders and multipliers.**

***Index Terms*— Approximate computing, arithmetic circuits, cross-layer approximation, netlist pruning, voltage overscaling (VOS).**

I. Introduction

Since the end of Dennard's scaling era, energy efficiency has become a primary design concern [1] and radical changes to conventional approaches are mandatory to sustain and improve our systems' efficiency. Approximate computing emerges as a promising design alternative and exploits the inherent error resilience of several application domains to trade accuracy for energy/performance gains [2]. It is shown that such applications spend, on average, 70% of their energy consumption in error tolerant computations [1]. Hardware-level approximate computing mainly targets arithmetic units (e.g., adders [2]–[7] and multipliers [8]–[13]) as they constitute the key components of a vast number of error-tolerant applications [1]. In hardware design, approximate computing can be applied in two distinct layers, i.e., the functional and the physical. In the physical layer, voltage overscaling (VOS) and overclocking are applied [14]. VOS is one of the most effective methods in reducing a circuit's energy consumption. Nevertheless, the exploitation of VOS in approximate computing works is still limited since: 1) quantifying the power-error characteristics of voltage overscaled circuits is a very complex task [15] and 2) high voltage decrease leads to unacceptable error values. In the functional layer, logic [5]–[10] and algorithmic [2]–[4], [11]–[13] approximations are applied. However, they mainly provide *ad hoc* solutions and cannot be reused in different applications and/or circuit architectures. In order to maximize the design efficiency, cross-layer approximation is also proposed, combining approximation techniques from both layers [16]. Such works apply functional approximation and translate the obtained delay gain to voltage scaling, delivering considerably higher energy savings. Yet, they use up most of the error margin in functional approximation and are then forced to apply conservative voltage decrease to adhere to the quality requirements, restricting the energy gains of aggressively decreasing the voltage value. Therefore, despite the high efficiency of cross-layer approximation, its exploitation is very limited and the efficient application of VOS remains an open issue.

In this brief, we design energy-efficient approximate arithmetic circuits by enabling effective VOS application and leveraging cross-layer approximation. We introduce the voltage-driven functional approximation that constrains the VOS error and achieves higher voltage decrease for the same error bound. On this basis, we propose VADER (VoltAge-Driven nEtlist pRuning), an automated synthesis framework for cross-layer approximate adders and multipliers. VADER leverages the effectiveness of voltage-driven functional approximation and applies VOS at the physical layer and netlist pruning at the functional one. VADER operates over the gate-level netlist and is build upon industry strength tools, seamlessly extending any hardware design flow. Experimental evaluation shows that the proposed voltage-driven functional approximation reduces the VOS error by 52% on average and that compared to state-of-the-art approximate adders and multipliers, VADER delivers on average 34% higher energy reduction.

II. Related Work

Approximate solutions for arithmetic circuits are further divided into operation-specific techniques, as well as to general-purpose frameworks. Targeting approximate adders, logic approximation is applied in [5]–[7], while algorithmic approximation is used in [2]–[4]. The approximate cell replacement presented in [7] outperformed [5], [6], while the configurable segmentation scheme in [2] delivers more efficient circuits than those in [3] and [4]. Approximate multipliers are designed in [8]–[10] using logic approximation and cell replacement, while algorithmic approximations are proposed in [11]–[13]. A Pareto analysis is performed in [13] showing that [13] delivers more energy-error efficient solutions than those in [9]–[12]. However, [2]–[13] are operation specific and/or architecture dependent. Their application over differing architectures is arguable and their optimality has not been comprehensively evaluated. For example, the proposed logic approximations may deliver different error values when applied to different circuit architectures due to the different carry propagation. Similarly, the energy savings of the algorithmic approximations depend on the circuit architecture, and thus, a configuration that is optimal for an adder/multiplier architecture may be suboptimal for another one. All these limitations heavily increase the already increased complexity of hardware design, since the designer has to verify both functionality and optimality in addition to operating within the error bounds. To address this complexity, general-purpose approximation frameworks have been proposed. The "don't care" conditions are leveraged in [17] and [18] to generate approximate logic circuits. Logic minimization and probabilistic pruning are proposed in [14], while Schlachter *et al.* [19] extends the latter and presents the gate-level pruning framework. To avoid design complexity, these techniques focus only on functional approximation,

Manuscript received October 8, 2018; revised January 5, 2019; accepted February 6, 2019. Date of publication March 7, 2019; date of current version May 22, 2019. This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020" in the context of the project "Automated methodology for production and execution of data-centric multi-level approximate equivalent applications for heterogeneous computing platforms" (MIS 5005377). *(Corresponding author: Georgios Zervakis.)*

G. Zervakis, K. Koliogeorgi, D. Anagnostos, and N. Zompakis are with the Department of Electrical and Computer Engineering, National Technical University of Athens, Athens 15780, Greece (e-mail: zervakis@microlab.ntua.gr).

K. Siozios is with the Department of Physics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2900160

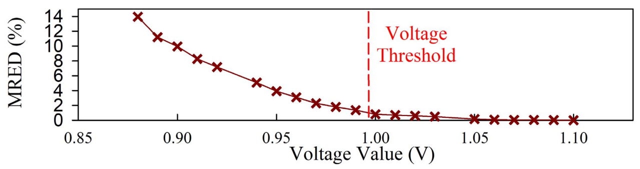

Fig. 1. VOS error versus voltage value graph for an 8-bit 4:2 Dadda multiplier.

neglecting the benefits originated by cross-layer approximation and the VOS application.

III. Voltage-Driven Functional Approximation

VOS is one of the most effective techniques in decreasing the energy consumption [20]. Decreasing the voltage value leads to significant energy savings, but the circuit becomes slower and output errors are generated due to the paths that fail to meet the time requirements. In this analysis, the mean relative error distance (MRED) [13] is used to evaluate the accuracy of the approximate circuits. MRED is the average relative error and is calculated by $(1/N)\sum_1^N |(O_{\text{Approx}} - O_{\text{Correct}})/O_{\text{Correct}}|$ [13], [19], where O_{Approx}, O_{Correct} are the approximate and accurate outputs, respectively, and N is the number of computations. For the rest of this brief, when referring to a circuit's error, we refer to its MRED value. A small voltage decrease leads to almost negligible error value, since very few paths violate the timing constraint. With further voltage reduction, the number of violating paths increases and the error value starts to slightly increase. After a voltage threshold, the number of violating paths increases significantly and the error increase becomes exponential [20], as illustrated in the example in Fig. 1. These large error values produced by VOS prohibit its wide adoption in approximate circuits.

In this brief, we enable the exploitation of the full potential of VOS by introducing the voltage-driven functional approximation. The main idea is that by moving the aforementioned voltage threshold toward lower voltage values, we will be able to apply higher voltage decreases for the same error bounds and, thus, increase the energy savings and satisfy the error constraints. This can be achieved by making the circuit faster, or else, by reducing the number of paths that violate the timing constraint, i.e., the paths that are susceptible to producing errors due to VOS. Therefore, we propose to strategically apply functional approximation at the circuit paths that violate the timing constraint, in order to reduce their delay and, consequently, decrease the error generated due to VOS. Leveraging the error compensation achieved by applying synergetic approximation techniques, we can decrease the error rate of a voltage overscaled path by applying directed functional approximation in addition to VOS in that path. In Fig. 2, a representative example is depicted, considering a small path of three NAND gates. The error rate reported in Fig. 2(a) and (b) is calculated through voltage-aware circuit simulations using a randomly generated data set of 10^4 inputs. Decreasing the voltage supply from 1.1 to 0.88 V [Fig. 2(a)], results in an error rate of 9.5%. In Fig. 2(b), we apply functional approximation and replace the first NAND with a constant "1." Then, decreasing the voltage value to 0.88 V results in a (combined) error rate of 6.2%, i.e., 1.5× smaller. Since VOS mainly affects the paths that drive the most significant bits, applying functional approximation in these paths may lead to large error values. The key point is to apply the suitable functional approximations that deliver the lowest error value but still manage to adequately decrease the path delays. Thus, the functional approximation must be applied in a very disciplined manner and low error rate approximations must be used.

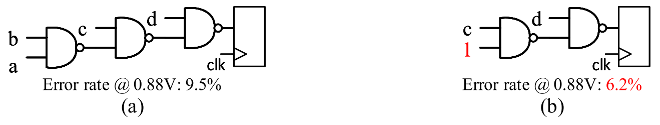

Fig. 2. Error rate of applying (a) only VOS and (b) netlist pruning and VOS.

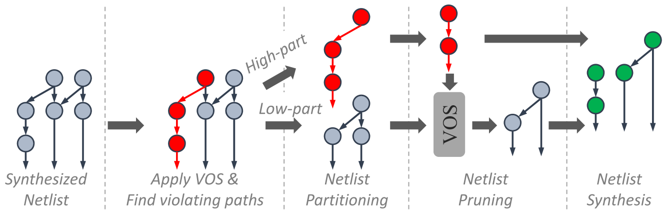

Fig. 3. Proposed VADER framework flow diagram.

To conclude, voltage-driven functional approximation applies targeted and light approximations at the VOS-affected paths to constrain the VOS error and benefit from the aggressive voltage decrease. Voltage-driven approximation should implement the following steps.

1) Identify the paths affected by VOS.
2) For each path identify, the functional approximation candidates, i.e., possible approximations with low error rate.
3) Apply the bare minimum set of approximations in order to keep the induced functional error to its lowest value.
4) Find the lowest voltage value that satisfies the error bound.

IV. VADER Framework

In this section, VADER, the proposed synthesis framework for approximate arithmetic circuits is described. VADER enables effective exploitation of cross-layer approximation by implementing the proposed voltage-driven functional approximation. The proposed framework applies the VOS and gate-level netlist pruning approximation techniques in the physical and functional layers, respectively. VADER operates on the circuit's synthesized netlist and is build upon industry strength tools. As a result, it can be applied to any circuit and can be seamlessly integrated to any typical hardware design flow.

An abstract overview of the proposed framework is illustrated in Fig. 3. The circuit is synthesized at the desired clock period and voltage value and the highest voltage value that violates the error constraint is identified. Next, VADER partitions the circuit into two distinct parts. The first one, named high part, contains all the circuit paths that violate the timing constraint (i.e., paths that may produce errors due to VOS), and the second one, named low part, contains the circuit's remaining paths. Then, VADER applies the voltage-driven functional approximation, by applying light pruning to the high part and aggressive pruning to the low part. Finally, VADER synthesizes the pruned netlist to remove any floating cells and leverage the optimizations performed by the synthesis tools. The output of our framework is the synthesized pruned netlist and the respective overscaled voltage value. VADER's flow is described in detail hereafter:

A. Netlist Partitioning

First, the circuit is synthesized at the desired clock period and voltage value to produce its gate-level netlist. Then, a binary search is invoked to efficiently find the highest voltage value at which the error bound is violated. This voltage value will be used next to guide the netlist partitioning and pruning. In every step of the binary search, the circuit is simulated to measure its error at the voltage value under investigation. This simulation is performed using the Synopsys CCS model and the VOSsim tool [15] that enables very fast VOS-aware simulation at gate level. Note that since the VOS error is monotone with respect to the voltage decrease, invoking a binary search minimizes the number of required voltage-aware simulations. Next, using the CCS model, VADER performs a static time analysis of the synthesized netlist (at the previously computed voltage value) and extracts the circuit paths that violate the timing requirements. These paths constitute the circuit's high part, while the remaining paths constitute the circuit's low part. In the low part, we subtract all the subpaths that also belong in the high part so that the two parts are mutually exclusive. As a result, there are no wires crossing the two parts and pruning low part wires does not affect the high part. Finally, an advantage of performing VOS-aware simulations at gate level is that VADER also computes the circuit's value change dump (VCD) file at the desired voltage value, thus allowing immediate comparison with the respective accurate one and straightforward calculation of the VOS error frequency (VERF) of every path in the high part (i.e., how often the path is activated when VOS errors occur).

B. Voltage-Driven Netlist Pruning

After partitioning the circuit, the next step of VADER is to apply the functional approximation and prune the netlist accordingly. The pruning procedure is applied to both the high and low parts but with different objectives. The objective of the high part pruning is to minimize the number of violating paths, while the goal of the low part pruning is to maximize the number of pruned wires. VADER implements the netlist pruning by replacing a wire with a "0" or "1."

VADER uses a verilog python parser to read the gate-level netlist and produces its directed acyclic graph. The graph nodes represent the netlist gates and the graph edges the wires. Using the python parser, we can evaluate at high level, and thus very fast, the error induced by pruning a wire without performing gate-level simulations. Moreover, using the output of the static time analysis, we annotate each graph node with the delay of the respective cell, and thus, when pruning a wire (edge), we can accurately estimate the obtained delay reduction.

1) High Part Pruning: The goal of the high part pruning is to apply those approximations that decrease the VOS error. VADER achieves this by pruning the respective wires that reduce the delay of the paths with the highest VERF while respecting the circuit's error bound. The high part paths are sorted by decreasing VERF (in order to approximate first the paths that produce VOS errors more frequently) and then a pruning loop procedure is invoked. The iterative Algorithm 1 is used to prune the high part wires. For each path in the high part, the algorithm evaluates its delay and if the time requirement is violated, a pruning process is invoked. In the pruning process, VADER has to decide which wires to prune and by which value. To identify the optimal pruning, VADER subsequently and separately sets every wire in the path first to "0" and then to "1" and the followings are calculated: 1) the resulting error; 2) the error rate; and 3) the path's delay reduction. Then, the algorithm finds all the pairs (*wire*, *bv*), with $bv \in \{0, 1\}$, that respect the error bound and decrease the path delay. Among these pairs, the one that minimizes the product *error* × *error rate* is selected and the respective *wire* is pruned by being replaced by the binary value *bv*. Algorithm 1 terminates when all the paths meet the time requirement or when no further approximations can be made without violating the error bound. To ensure that adequate error margin remains for approximating the low part, we set the high part error bound (in Algorithm 1) equal to 70% of the total error bound.

2) Low Part Pruning: After pruning the high part, VADER reperforms a binary search to find the lowest voltage value that satisfies the error bound. Ideally, this would be the voltage value extracted by the first binary search (or smaller). However, this cannot be known *a priori* and a second binary search is required to find the lowest value that satisfies the error constraint. After this step, we obtain: 1) the pruned high part of the netlist; 2) the voltage value that will be used to apply VOS; and 3) the error value of applying both high part netlist pruning and VOS. Based on the VADER's netlist partitioning, approximations in the low part do not affect the high part. Therefore, the circuit's total error (E_C) is given by $E_C = E_{H\&V} + E_L$, where E_L is the error of the low part and $E_{H\&V}$ is the combined error of the high part and VOS. Hence, if E_B is the circuit's error bound, the available error margin for pruning the low part is given by $E_B - E_{H\&V}$, where $E_{H\&V}$ is obtained from the second binary search. A similar loop procedure to Algorithm 1 is used in order to prune the wires of the low part. The high part pruning procedure is path-centric and Algorithm 1 performs per path approximations targeting to reduce the delay of each violating path. On the other hand, in the low part pruning, the goal is to maximize the number of pruned wires regardless of the paths they belong to. This pruning procedure, in every iteration, evaluates the error of pruning each wire in the low part and prunes the one that induces the least error. The low part pruning procedure terminates when no further approximation can be made without violating the low part error margin.

Finally, after pruning the high and low parts, the generated netlist is synthesized in order to remove any floating cells as well as to leverage the high optimization efficiency of the circuit synthesis tools.

Algorithm 1 High Part Netlist Pruning

```
Input: 1. netlist, 2. error bound, 3. high-part sorted for descending VERF
 1: while (netlist changed)
 2:     for all path in high-part
 3:         evaluate path delay
 4:         if (path delay is met) go to next path
 5:         for all wire in path and bv in $\{0, 1\}$
 6:             set wire to bv
 7:             evaluate: error, error rate, and path delay
 8:         for all (wire, bv) that decrease the path delay
 9:             find (wire, bv) with minimum product error $\times$ error rate
10:         if (error bound is not violated) prune wire
```

C. VADER Implementation

One of the main objectives of VADER is its usability and seamless integration in existing design flows. VADER comprises only of bash, tcl, and python scripts and does not require any modification of the circuits hardware description. Moreover, it is implemented over widely used, industry strength tools, i.e., Synopsys Design Compiler, PrimeTime, and Mentor Questasim. The gate-level voltage-aware timing simulation is performed by VOSsim [15] which also operates over the same tools. However, both VADER and VOSsim are independent of these tools and can be adapted to any tool with similar functionality (by changing the tool specific commands in our scripts). Therefore, VADER can be out-of-the-box integrated into any typical hardware design flow with zero user-overhead. Moreover, to minimize VADER's execution time, the majority of the evaluations is performed at high level and over the circuit's graph. Python's bitwise operators are used to emulate the functionality of each graph's node and pruning is performed by removing the respective edge and replacing the input of the nodes that it connects with "0" or "1". Finally, voltage-aware simulations are required only during the two binary searches. Hence, up to $2\lfloor \log_2(M) + 1 \rfloor$ simulations are required, where M is the VOS range. In this brief, up to 20% voltage reduction is examined, and thus, 10 voltage-aware simulations are performed at most.

V. Experimental Evaluation

In this section, we experimentally evaluate the proposed VADER framework by comparing it against state-of-the-art approximate computing works. Moreover, we examine the efficacy of the voltage-driven functional approximation (implemented in VADER) in decreasing the VOS [21] error. The ripple carry adder (RCA) [7] and the 4:2 Dadda multiplier [8] are used as our driving circuits. Two bitwidths, 8-bit and 16-bit, are considered for every circuit. The efficiency of VADER in image processing is also examined, by using the adders and multipliers produced by VADER in the Sobel filter [1] and in image multiplication [22], respectively. VADER is compared against operation-specific techniques that apply logic approximation [7], [8] and algorithmic approximation [2], [13]. InXA [7] and GeAr [2] build approximate adders, while ACMP [8] and ROPE [13] design approximate multipliers. VADER is also compared against the GLP framework [19] that is applied to both adders and multipliers. MRED [13] and mean structural similarity (MSSIM) [22] are used to evaluate the accuracy for the arithmetic units and image processing applications, respectively. The error and energy consumption of all approximate designs are calculated through exhaustive simulation. Two different randomly generated input data sets are used in our evaluation. The first one (10^4 inputs) is used during VADER's evaluations and the second one (10^6 inputs) is used to compute the error and energy metrics of the examined approximate arithmetic circuits. For the Sobel and Image Multiplication, ten 8-bit 256 pixel images are used. All the designs are synthesized and simulated at the critical path delay of the respective accurate one. The Nangate 45-nm standard cell library is used and the nominal voltage value is 1.1 V. The experiments run on a dual Xeon Gold 6138 server with 128-GB RAM.

In Fig. 4, we evaluate the proposed voltage-driven functional approximation, and we examine the effectiveness of VADER to decrease the error produced by the VOS application [21]. Specifically, the normalized error of VADER with respect to VOS-only application (ratio of VADER error over VOS error) for varying voltage decreases is presented. In order to evaluate the efficacy of the voltage-driven functional approximation, for each examined voltage value, we apply pruning only in the high part. As shown in Fig. 4, in almost all the examined voltage values, VADER decreases the VOS [21] error significantly. On average, for all the examined designs and voltage values, VADER reduces the VOS error by 52% enabling us to apply significantly higher voltage decreases (and thus higher energy savings) for the same error bounds. For the large designs (16-bit adder and multiplier), VADER achieves higher error reduction, i.e., 68% on average, while for the smaller ones (8-bit designs), the error decrease is 35% on average. Similarly, for higher voltage decreases, VADER delivers, mainly, higher error improvements. For larger circuits or lower voltage values, the number of paths that violate the timing constraint increases significantly. As a result, the number of wires that can be pruned increases, and hence, VADER has more flexibility and can apply less significant approximations in the high part. If there are only a few wires, pruning any of these wires will induce considerable error, and thus, VADER will not be able to decrease significantly the delay of the violating paths. However, even in the case of a very small circuit as the 8-bit RCA, VADER delivers an average 32% error reduction. In total, VADER fails to reduce the VOS error merely in four cases, and only due to the voltage values being higher than the voltage threshold of the respective design. As a result, a few paths violate the time constraint and VADER cannot prune any of them without increasing the error. Note, however, that at these cases, the error is almost negligible. Finally, by applying voltage-driven functional approximation in addition to VOS, not only is the VOS-only error decreased but also the energy consumption is reduced. On average, for all the designs and voltage values in Fig. 4, VADER reduces energy consumption compared to VOS-only [21] by 37%. The energy reduction ranges from 0% (the four cases that VADER could not apply functional approximation in addition to VOS) up to 65%. Note that this comparison refers to isovoltage conditions. In addition, since VADER also features lower error than

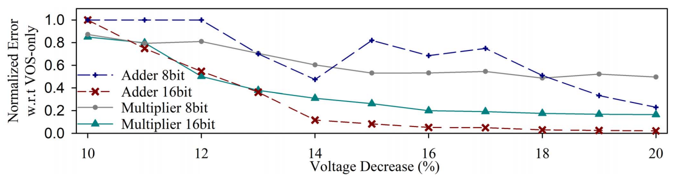

Fig. 4. Error reduction delivered by applying the voltage-driven functional approximation (VADER) with respect to VOS-only application.

VOS, the energy savings of VADER under isoerror conditions will be higher.

Next, we compare VADER against the accurate design and state-of-the-art functional approximate ones. Fig. 5 depicts the relative energy consumption of the examined approximate designs compared to the respective accurate one. First, Fig. 5(a), (b), (d), and (e) evaluates the effectiveness of VADER in producing approximate multipliers and adders. Next, the approximate multipliers and adders are used in the image multiplication and sobel benchmarks, respectively. All the designs that feature average MSSIM greater than 0.9 are depicted in Fig. 5(c) and (f). Regarding the approximate adders and multipliers, varying error bounds are examined that range from 1% up to 5%. The designs produced by VADER exhibit MRED less or equal to the error bound while the distribution of their RED is right-skewed featuring low dispersion. On average, for the examined designs and error bounds, the median absolute deviation of RED is 1.1% with a maximum value of 2.8% for 5% error bound. The approximate adders and multipliers produced by VADER apply both functional approximation and VOS. Regarding VOS, the designs shown in Fig. 5(a), (b), (d), and (e) have undergone on average 16% voltage reduction (ranging from 6% up to 20%). Similarly, regarding functional approximation VADER prunes, on average, 35% of wires, ranging from 6% for the 8-bit adder and 1% error bound to 64% for the 16-bit multiplier and 5% error bound. This high pruning rate is translated to significant area savings. Compared to the accurate circuits, VADER delivers 49% area reduction on average, ranging from 10% up to 82%. Regarding the energy efficiency, i.e., our main target, VADER delivers on average 62% energy reduction compared to the accurate designs that ranges from 24% for the 8-bit RCA [Fig. 5(e)] with 1% error bound up to 90% for the 16-bit Dada multiplier [Fig. 5(a)] with 5% error bound. As shown in Fig. 5, VADER delivers higher energy savings for larger circuits and error bounds. This comes in compliance with Fig. 4, where VADER is more efficient when the number of violating paths increases. Compared to state-of-the-art approximate adders [2], [7], [19] [Fig. 5(d) and (e)], the designs produced by VADER deliver on average 40% higher energy reduction. Similarly, compared to the existing approximate multipliers [8], [13], [19] [Fig. 5(a) and (b)], VADER delivers 29% higher energy savings on average. As illustrated in Fig. 5, VADER is always the most energy efficient (from 9% up to 66%) regardless of the operation type, circuit size, and error bound. Finally, the GLP framework [19] also applies netlist pruning following, however, a conventional approach widely used in approximate computing, i.e., approximate the paths that affect the LSBs in order to constrain error value. Compared to GLP, VADER delivers 28% higher energy reduction on average that ranges from 14% up to 39%. This significant energy gain highlights the high efficiency of the proposed voltage-driven functional approximation. In image processing [Fig. 5(c) and (f)], VADER achieves significant energy reduction and very high MSSIM values. Compared to the accurate design, in image multiplication, VADER attains from 31% up to 60% energy reduction for 0.985 and 0.915 average MSSIMs, respectively. For the Sobel filter, the respective values are 48% up to 69% lower energy for 0.991 and 0.906 average MSSIM. Compared to the state-of-the-art techniques, in both benchmarks, the designs of

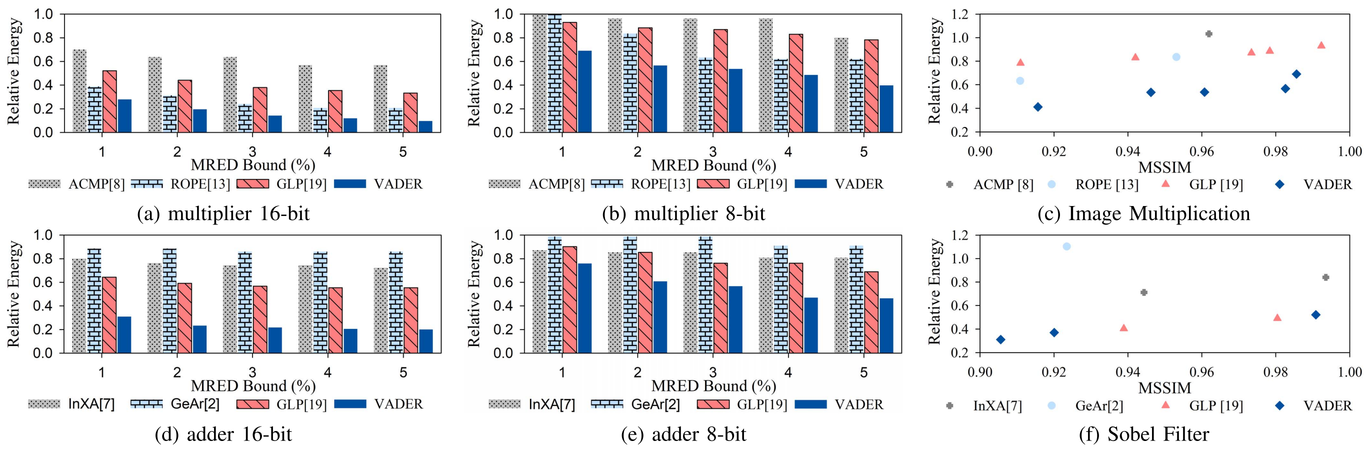

Fig. 5. Comparison of VADER against the state-of-the-art approximate works InXA [7], GeAr [2], ACMP [8], ROPE [13], and GPL [19]. (a) Multiplier 16-bit. (b) Multiplier 8-bit. (c) Image Multiplication. (d) Adder 16-bit. (e) Adder 8-bit. (f) Sobel Filter.

VADER are always at the Pareto front, constituting the most efficient ones. For average MSSIM bound of 0.9, VADER features 72%, 56%, and 23% less energy consumption compared to [2], [7], and [19], respectively [Fig. 5(f)]. In Fig. 5(c), the respective values are 35%, 60%, and 48% compared to [8], [13], and [19].

Regarding time complexity, VADER requires a few minutes for the 8-bit adder and up to 4 h for the 16-bit multiplier. Although the time required by VADER is considerable, it is comparable, and even significantly smaller, to existing approximate design frameworks. For example, in our evaluation [19] required 13 h for the 16-bit multiplier, while Miao *et al.* [18] reports requiring more than 20 h for 8-bit multipliers. In every iteration of Algorithm 1, VADER estimates if a path is affected by VOS based on its delay. If the delay is higher than the timing constraint, VADER assumes that the path produces VOS errors. Hence, in each loop, VADER does not need to perform circuit syntheses and voltage-aware simulations. Moreover, all the functional error evaluations are performed very fast at high-level using python and the CCS model is used for the paths' delay estimations.

VI. Conclusion

In this brief, we introduce the voltage-driven functional approximation technique and we demonstrate its high effectiveness in decreasing the error produced by the VOS application. Furthermore, we propose VADER, an automated synthesis framework for approximate cross-layer arithmetic circuits, which implements the voltage-driven functional approximation and enables the efficient VOS application and cross-layer approximation automation. We show that VADER delivers very energy-efficient approximate circuits and it significantly outperforms existing state-of-the-art approximate adders and multipliers.

References

[1] A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, and H. Esmaeilzadeh, "AxBench: A benchmark suite for approximate computing across the system stack," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-CS-16-01, 2016.

[2] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. IEEE 52nd ACM/EDAC Design Automat. Conf.*, Jun. 2015, pp. 1–6.

[3] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[4] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2013, pp. 48–54.

[5] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in *Proc. IEEE 13th Int. Conf. Nanotechnol.*, Aug. 2013, pp. 690–693.

[6] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[7] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhib.*, Mar. 2016, pp. 660–665.

[8] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[9] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

[10] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[11] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2015, pp. 418–425.

[12] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "Roba multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[13] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Walking through the energy-error Pareto frontier of approximate multipliers," *IEEE Micro*, vol. 38, no. 4, pp. 40–49, Jul. 2018.

[14] A. Lingamneni, C. Enz, K. Palem, and C. Piguet, "Synthesizing parsimonious inexact circuits through probabilistic design techniques," *IEEE ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, p. 93, May 2013.

[15] G. Zervakis, F. Ntouskas, S. Xydis, D. Soudris, and K. Pekmestzi, "Vossim: A framework for enabling fast voltage overscaling simulation for approximate computing circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 6, pp. 1204–1208, Jun. 2018.

[16] S. Lee, L. K. John, and A. Gerstlauer, "High-level synthesis of approximate hardware under joint precision and voltage scaling," in *Proc. IEEE Design Autom. Test Eur. Conf. Exhib.*, Mar. 2017, pp. 187–192.

[17] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *Proc. IEEE DAC Design Autom. Conf.*, Jun. 2012, pp. 796–801.

[18] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2014, pp. 504–510.

[19] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1694–1702, May 2017.

[20] Y. Liu, T. Zhang, and K. K. Parhi, "Computation error analysis in digital signal processing systems with overscaled supply voltage," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 4, pp. 517–526, Apr. 2010.

[21] G. Karakonstantis and K. Roy, "Voltage over-scaling: A cross-layer design perspective for energy efficient systems," in *Proc. IEEE 20th Eur. Conf. Circuit Theory Design*, Aug. 2011, pp. 548–551.

[22] M. H. Moaiyeri, F. Sabetzadeh, and S. Angizi, "An efficient majority-based compressor for approximate computing in the nano era," *Microsyst. Technol.*, vol. 24, no. 3, pp. 1589–1601, Mar. 2018.