Hybrid Approximate Multiplier Architectures for Improved Power-Accuracy Trade-offs

Georgios Zervakis, Sotirios Xydis, Kostas Tsoumanis, Dimitrios Soudris and Kiamal Pekmestzi
National Technical University of Athens, 9 Heroon Polytechneiou, Athens Greece
{zervakis, sxydis, kostastsoumanis, dsoudris, pekmes}@microlab.ntua.gr

Abstract—**Approximate computing forms a promising design alternative for inherently error resilient applications, trading accuracy for power savings. In this paper, we exploit multi-level approximation, i.e. at the algorithmic, the logic and the circuit level, to design low power approximate arithmetic architectures for hardware multipliers. Motivated from the limited power savings that approximation techniques can achieve in isolation, we explore hybrid methods that apply simultaneously more than one techniques from different layers. We introduce the concept of perforation for approximate arithmetic circuit design and we explore the newly defined design space of hybrid designs showing that it leads to lower power consumption at every examined error range. To address the increased complexity of the target design space, we introduce an heuristic optimization technique and the corresponding design framework that automatically generates hybrid low-power approximate multipliers requiring a small number of design evaluations, i.e. synthesis, simulation, power and timing analysis. Through extensive experimentation, we show that the proposed techniques converge towards optimal solutions and deliver approximate designs that are always more efficient with respect to state-of-art approaches. Power savings of 11% are reported for small error bounds and more than 30% in case of more relaxed error constraints.**

I. Introduction

Power consumption is considered as a first class design concern of modern embedded electronic devices. Digital Signal Processing (DSP) units are widely used in portable devices to implement multimedia algorithms, e.g. image and video processing. Given that such applications produce outputs for human consumption, their exactness is highly relaxed due to limited human perception [1]. This inherent error resilience of these applications allow approximate calculations to be performed by relaxing the numerical exactness of such applications, thus significantly reducing their power dissipation.

Arithmetic units such as adders and multipliers constitute the main units of DSPs. Up to now, extensive research has been conducted on approximate adders [2]–[5]. In [5], the authors showed that the statistically longest carry chain in an n-bit adder is $\log_2 n$, and produced a fast approximate implementation limiting the carry propagation. An approximate adder design has been proposed in [4] comprising two partitions, an accurate and an inaccurate one. However, research activities on approximate multipliers are limited mainly due to their increased circuit complexity. Recently, [6] proposed a systematic method to compensate the error of approximate arithmetic circuits and applied it on a truncated multiplier, improving its error metrics compared with related truncation schemes. [7] proposed an imprecise 2x2 multiplier cell used as the basic block for constructing larger multiplier architectures. In [8], the authors presented approximate 4:2 compressors, by modifying the respective accurate truth table, which were then used to build two approximate multipliers outperforming [7]. Utilizing approximate adders that limit carry propagation, [9] proposed a fast low-power multiplier but with higher error than [8].

In this paper, we target approximate multiplier designs that, given the error bounds, push power gains to the limits. We introduce the adoption of perforation technique [10], originally used in software, for approximate multiplier design and then, we present a novel design framework that exploits multi-level approximation for designing power optimized hybrid approximate multiplier architectures. We show that even when applying state-of-the-art approximation techniques in isolation, limited power reductions are delivered. Motivated by this fact, we propose the exploration of hybrid approximation techniques that apply simultaneously more than one techniques. Through extensive experimentation, we prove that for any given error range, the usage of the proposed hybrid approximation techniques delivers, in all cases, approximate multiplier circuits with smaller power consumption than multiplier designs following the state-of-art approximation strategies. We define the design problem of finding the power-optimal hybrid approximate multiplier architecture subject to maximum error constraint and we show that its complexity is equivalent to the "nonlinear nonseparable bounded integer" KNAPSACK [11]. We devised an heuristic optimization procedure that exploits power-error analytical fitting models to produce optimized hybrid approximate multiplier designs without resorting to costly synthesis, simulation and power calculation steps. For small error values, the designs produced from our framework consume up to 11% less power than those with a single state-of-art approximation technique, while for relaxed error values they deliver more than 30% power savings.

The rest of the paper is organized as follows: In Section 2, we introduce the adaptation of perforation technique for approximate multiplier design and we provide an overview of approximation techniques. In Section 3, the problem of designing power efficient approximate arithmetic circuits subject to error constraints is defined and the proposed cross-layer design framework for optimized hybrid multiplier architectures is analysed. In Section 4, we experimentally evaluate our techniques proving their significance on moving towards more optimized solutions, exhibiting better power-error trade-offs in respect to state-of-art approximation approaches available in the literature. Finally, Section 5 summarizes our work.

II. Analyzing the Design Space of Approximation Techniques for Arithmetic Circuits

Approximation techniques can be applied over a set of different design levels (Fig. 1), i.e. (i) algorithmic-architectural, (ii) bit-structural and (iii) circuit. Lower layers are enclosed in the upper ones, since the application of approximate techniques in the former depends on decisions made in the latter. Algorithmic-architectural level approximation is achieved by modifying the algorithm accordingly, namely by substituting some instructions with simpler ones or even by avoiding to execute them [10]. In the bit-structural level, the correctness of some circuits' components is relaxed, thus, lowering their power consumption, but producing imprecise outputs [3], [4], [8]. Approximation techniques at circuit level can be applied once the circuit is implemented, i.e. the approximation is achieved by over-clocking or by voltage over-scaling [1], [2].

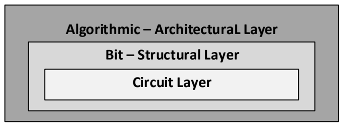


Fig. 1. The approximation layers.

The rest of the section analyses the approximation techniques associated with each of the aforementioned design levels.

Driver circuit design: Without loss of generality, we considered the circuit of a Dadda multiplier implemented with 4:2 compressor tree [12] and a carry look-ahead adder as the final adder (Fig. 2). Both Simple Partial Product (SPP) and Modified Booth Encoding (MBE) are considered in our analysis. All the designs discussed hereafter are synthesized using the Synopsys Design Compiler and the TSMC 65nm standard cell library at a 500Mhz frequency. The designs are simulated using Modelsim and their power consumption is calculated by Synopsys PrimeTime. For the error evaluation, the metrics of Error Distance (*ED*)[1], Mean Error Distance (*MED*)[2] and Normalized *MED* (*NMED*)[3] are considered as effective metrics for quantifying the accuracy of approximate circuits [13]. Finally, for the error calculation, an exhaustive simulation of all possible inputs is performed.

A. Adapting Perforation for Multiplier Design

At the algorithmic level, approximation is performed through the omission of instructions, i.e. the execution of a subset of the original operations. Targeting this method of approximation, [10] proposed the loop perforation technique which skips a number of loop iterations and results in executing less computations, thus saving time and power, but producing inaccurate outputs. Driven by the fact that hardware multiplication algorithms are inherently iterative, we propose the adaptation of perforation techniques for approximate multiplier design [14]. The result $A \times B$ of multiplying two n-bit numbers *A* and *B*, is obtained after summing all the partial products Ab_i, where b_i is the i^{th} bit of *B*, thus $A \times B = \sum_{i=0}^{n-1} Ab_i 2^i$. Inspired from [10], we apply the perforation technique on a hardware multiplier. A first approach would be to examine the removal of some bits from the partial products (remove some dots from stage 1 in Fig. 2) and to explore all the possible combinations. However, this approach leads to an explosion of the solution space. Therefore, we select and explore a more coarse grain method, the perforation on entire partial products, in which the generation of some products is omitted. If the j^{th}, k^{th} and m^{th} partial products are perforated, the approximate multiplication result is $A \times B = \sum_{\substack{i=0 \\ i \notin \{j,k,m\}}}^{n-1} Ab_i 2^i$. In Fig. 3 the accumulation tree of a Dadda 4:2 multiplier after perforating the 2^{nd} and 4^{th} partial products is illustrated. The minimum order of the perforated partial products is the order of perforation and their cardinality is the length of perforation. A perforated partial product is not inserted in the accumulation tree and, as a result, n full adders can be removed from the tree, reducing its power consumption while producing approximate results. The greater the length of perforation, the greater the power decreases and the error increases. Moreover, as the order of a perforated partial product increases, so does the imposed error, as it affects more significant bits. Trying to minimize the induced error when perforating with length more than one, we perforate on successive partial products. There is no upper bound on the power reduction that can be achieved using the perforation method (all the partial products can be perforated), however, perforating on many partial products makes the error enormous, rendering it infeasible. In Fig. 4a the variation of the power consumption respectfully to *NMED* when applying the perforation technique on the Dadda 4:2 multiplier is depicted. All the possible combinations of perforation that lead to $NMED < 10^{-3}$ are presented. The spikes in the graph are caused by the increase in the order of perforation while its length is one and, at these points, the power and error are greater than those at the previous adjacent points. For this reason, for the rest of the paper we consider an order of perforation as reference and vary its length in order to retain the monotony of the power-error curve.

B. Logic Approximation

A widely used technique is the logic approximation applied to the structural level. This method targets mainly the simplification of a component's logic complexity (i.e. alteration of the truth table). The simplification of a unit's complexity reduces its power consumption, but induces computational errors. In order to apply the logic approximation on a multiplier, we use the approximate 4:2 compressor[4] of [8]. By replacing the accurate compressors (the boxes containing four dots in Fig. 2) in a column of the Dadda 4:2 with imprecise ones, its power consumption decreases, however so does the accuracy of the multiplier; the more significant the column in which the compressors are replaced, the larger the error in the output. As a result, increasing the approximated columns decreases the power by increasing the error. Fig. 4b illustrates the variation of the power consumption with respect to *NMED* and confirms that the *NMED* increases and the power decreases with the increase of the approximated columns. At lower error values of Fig. 4b, the power decrease is faster than the error increase. This is explained by the architecture of the Dadda 4:2 tree. Moving from the right to the center of the tree, the number of the 4:2 compressors increases, while only the Least Significant

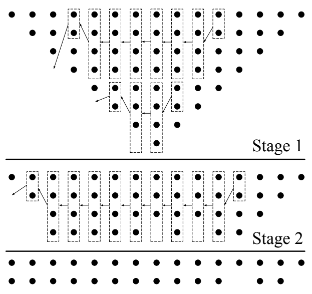


Fig. 2. Reduction strategy of an accurate 8-bit Dadda tree using 4:2 compressors. The boxes with four dots are the 4:2 compressors, while those with two are either full or half adders. The arrows represent the output carries.

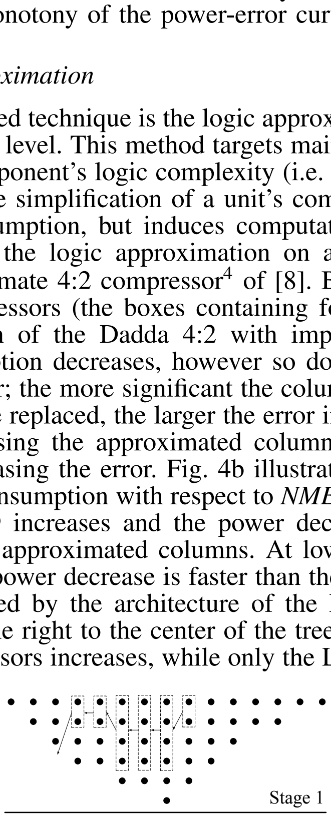


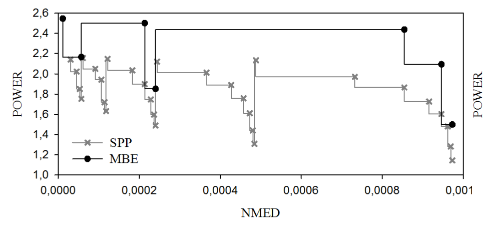
Fig. 3. The Dadda 4:2 accumulation tree after perforating the 2^{nd} and 4^{th} partial products.

[1]The *ED* metric is defined as the absolute distance of the accurate product *P* and the approximate one P', $ED = |P - P'|$.

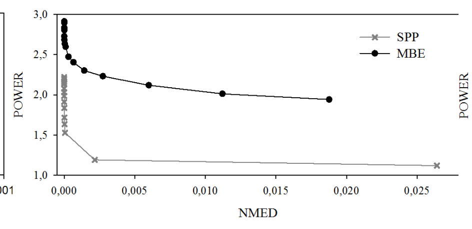
[2]The *MED* is the average of all *ED*s.

[3]$NMED = MED/P_{max}$, where $P_{max} = (2^n - 1)^2$ in the case of an n-bit multiplier [9].

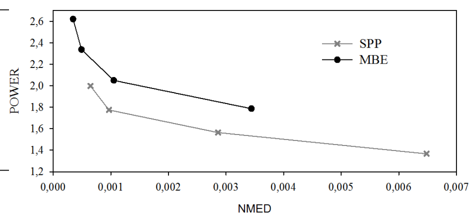
[4]We use the second proposed compressor. Its design as well as its differences from the accurate 4:2 compressor can be found in [8].



(a)



(b)



(c)

Fig. 4. The variation of the power consumption with respect to *NMED* of the approximate Dadda 4:2 multiplier when applying a) perforation, b) the approximate 4:2 compressors, and c) VOS. The power of the accurate multiplier with SPP and MBE is 2.24mW and 2.92mW, respectively.

Bits (LSB) are affected. As a result, the power reduction benefits more than the error increase. In contrast, moving from the center to the left, the number of the compressors in the columns decreases while the Most Significant Bits (MSB) are approximated. Thus, the error increase is more significant than the achieved power reduction. Finally, the power reduction attained by the application of approximate compressors is bounded. Even when all columns are approximated, the power consumption is 1.11 and 1.57 mW for the SPP and MBE designs respectively.

C. Voltage Over-Scaling

Voltage over-scaling (VOS) is applied at the circuit level. Unlike the other two techniques, VOS can be applied to every circuit without the need of special design or algorithm modification. VOS lowers the supply voltage below its nominal value. Decreasing the supply voltage reduces the circuit's power consumption, but produces errors caused by the number of paths that fail to meet the delay constraints [1], as qualitatively shown in Fig. 5. The number of paths that violate these constraints varies and depends on the circuit's architecture. However, the paths that cannot reach the delay constraints are the longest ones and usually those that affect the MSBs of the output. As a result, when errors are produced by the application of VOS, they are of great significance compared to the accurate output. We implemented the VOS technique on the Dadda 4:2 multiplier, by using the Synopsys Composite Current Source model (CCS) [15]. CCS models have been proven to deliver signoff-level accuracy to within 2% of HSPICE simulation and are designed to be scalable for voltage, temperature and process parameters. In addition, they offer better accuracy than the Non-Linear Delay and Power models [15]. Retaining the original path timing constraints, we use CCS model to scale the supply voltage from 1V (nominal) to 0.80V and measure its power consumption and error metrics. Fig. 4c shows the variation of the power consumption with respect to *NMED*. In average, VOS delivers around 95% accurate results. However, whenever an error occurs its magnitude is high. Finally, the power reduction is bounded by the fact that the voltage supply cannot take lower values than the threshold voltage.

D. The need for hybrid approximation techniques

Previous analysis indicated that the approximation methods described above exhibit a variety of characteristics and power optimization potentialities. More specifically:

- The VOS technique, despite delivering more than 95% exact results, requires high decrease of the supply voltage in order to attain high power reduction, inducing high error in the output.
- Logic approximation offers moderate, though considerable, power reduction for small error values when some LSB columns are approximated. However, approximating more than the half columns is profitless, because it results in much faster error increase than the power decreases. As a result, despite the fact that the approximate compressors technique delivers small error, the achieved power reduction is limited. Furthermore, the logic approximation shows worse behavior when the MBE technique is preferred for the partial product generation.
- Finally, the introduced perforation method delivers significant power reduction in exchange for a considerable error. More specifically, for the MBE designs, it offers better results than the approximate 4:2 compressors. However, for the SPP designs and at smaller error bounds, the approximate 4:2 compressors achieve lower power values.

Taking into consideration the limited power reduction offered by the aforementioned approximation techniques, when they applied in isolation, we have been highly motivated to examine the power optimization potential of hybrid designs generated by effective coordination of the aforementioned techniques.

III. Design Optimization Framework for Hybrid Approximate Multipliers

In this section we describe the proposed design framework that exploits multi-level approximation techniques to produce hybrid, low power approximate multipliers. The target design optimization problem can be formulated as follows:

$$\min_{\mathbf{x}\in \mathbf{D}}[\ Power(\mathbf{x})\ ] \tag{1}$$

subject to

$$[\ NMED(\mathbf{x})\ ]\leq[\ Max\_NMED\ ], \tag{2}$$

where the optimization goal is to find the configuration **x**, i.e. combination of approximation techniques that defines the overall design space **D**, that minimizes power consumption for a given error constraint.

The computational complexity of our optimization problem increases significantly due to the need for an exhaustive search of all possible configurations of each technique. To assess its complexity, our problem can be viewed as a generalization of the well-known NP-complete KNAPSACK problem, i.e. the

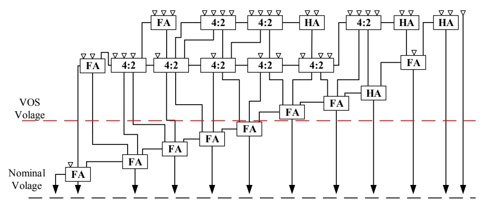


Fig. 5. Example of path violation when applying VOS.

"nonlinear nonseparable bounded integer" KNAPSACK [11]. We can re-formulate the combinatorial problem of Eq. (1)-(2), by considering the *maximization of power reduction* rather than the minimization of power. The problem can be re-formulated as follows: given a knapsack (multiplier), select an integer number of objects (each object maps to a specific configuration of an approximation technique from each layer), each one with distinct cost (multiplier error) and gain (power savings), such that their collection will maximize the multiplier's power savings under a given constraint on its output error. We consider that each object belongs to one of three categories (VOS, perforation and imprecise compressors) and that when applied to the multiplier (added to our knapsack), it has its own distinct effect on power and accuracy. In contrast to the conventional KNAPSACK, the cost/gain of each object is not constant and depends on the objects already placed in the knapsack. Thus, we use $\mathbf{x} =< x_1, x_2, x_3 >\in \mathcal{N}^3$ to denote the objects selected and we formulate:

$$\max_{\mathbf{x}\in \mathcal{D}}[\ Power\_Reduction(\mathbf{x})\ ] \tag{3}$$

subject to

$$[\ NMED(\mathbf{x})\ ]\leq[\ Max\_NMED\ ], \tag{4}$$

where the nonlinear $Power\_Reduction(x) \in \mathcal{R}$ denotes the total power reduction and the nonlinear $NMED(x) \in \mathcal{R}$ the total error. This formulation corresponds to the definition of the nonlinear KNAPSACK problem. We note that the functions $Power\_Reduction(x)$ and $NMED(x)$ are not separable, as they depend on the combination of x_i's. Overall, the increased complexity of this little-studied version of KNAPSACK problem mandates the use of heuristic algorithms in practice [11].

In order to tackle the high computational cost, we developed an heuristic optimization algorithm that approximates the optimal solutions after a small number of synthesis/simulation runs. The proposed heuristic exploits/devises problem specific models to capture the behavior of the parameter space, i.e. configurations of approximation techniques, which are subsequently used during iterative optimization. The goal of the regressive models is to form an estimation proxy of the error and power to be further used during optimization search and not to be an accurate calculation of the error/power.

Error proxy of hybrid multipliers: Let us assume E_P, E_C, E_V and O_P, O_C, O_V to be the error functions and the multiplier output after applying the perforation, the approximate compressors, and the VOS techniques. O_A refers to the output of an accurate design, while E_H refers to the error of the hybrid design when the three approximation techniques are applied. The perforation of some of the partial products results in skipping their generation and thus, they are not inserted in the accumulation tree. As a result a new, smaller tree that performs correct calculations is used to accumulate the remaining partial products. Hence, $O_A = O_P + E_P$. Applying logic approximation to the new tree, i.e. replacing its exact 4:2 compressors with the approximate ones in some of its columns, induces a new error to the accurate output of the new tree, i.e. $O_A = (O_C + E_C) + E_P$. As mentioned in Section II-C, VOS produces errors that mainly affect the critical paths, which in multipliers are formed at the MSBs, while logic approximation and perforation mainly affect the LSBs. Under this heuristic assumption that VOS generates errors in higher bit-rankings, we can write $O_A = (O_V + E_V) + E_C + E_P$. Hence, the final error E_H of the hybrid design with the three techniques, can be estimated by $E_H = O_A - O_V = E_P + E_C + E_V$.

Power proxy of hybrid multipliers: Assume that RP_P, RP_C and RP_V are the respective relative power (*RP*) functions of applying the perforation, the approximate compressors, and VOS, e.g. $RP_P = \frac{P_P}{P_A}$. Similarly, for the *RP*, the perforation of some of the partial products reduces the power of the tree by a percentage that depends on the length of the perforation. The application of logic approximation on the new tree further reduces its power by a percentage that depends on the number of the compressors that are replaced, i.e. for λ approximated compressors over N compressors in total, $P_C = P_P \times \frac{\lambda}{N} \Rightarrow RP_H = \frac{P_C}{P_A} = RP_P \times \frac{\lambda}{N}$. Finally, applying VOS on a circuit decreases its power consumption by a factor k, $k = (V_1/V_0)^2$ where V_0 is the nominal voltage and V_1 is the decreased voltage, that depends only on the percentage of its supply voltage decrease. Thus, applying VOS on the previous approximated tree with perforation and logic approximation will deliver a relative power for the hybrid design, i.e. $P_H = k\times P_C \Rightarrow RP_H = k\times\frac{\lambda}{N}\times RP_P = RP_V\times RP_C\times RP_P$.

Validation of error and power proxies: In order to evaluate the accuracy of our regressive estimators, we calculate through synthesis and simulations the error and power of every hybrid model and compare it with the values obtained from the proposed estimators. The resulting mean square error (*MSE*) for the error estimator E_H is 1.2×10^{-9} and that of the *RP* estimator RP_H is 6.7×10^{-4}. Taking into account that the error of all hybrid designs is greater than 3.8×10^{-6} and their *RP* is greater than 2×10^{-1}, the proposed estimators are considered to offer satisfactory accuracy.

Design exploration and optimization: In Fig. 6 the proposed framework is presented. The first step of the framework is to calculate the error and the *RP* functions of each approximation technique. We synthesize and simulate the accurate multiplier and four different configurations for each (single) technique and measure their power consumption and error. Using the points created, a regression curve is calculated to estimate each of the aforementioned functions. Since each of the examined approximation techniques preserves monotony and convexity regarding to the error and power values, the four points sampling is considered enough to produce accurate regression. Table I depicts the resulted regression lines and their respective *MSEs* for a 16-bit Dadda 4:2 multiplier with SPP. The *MSE* of each regression line is very low, verifying that the error and *RP* of each technique can be estimated precisely with only four points.

In Table II the accuracy of the proposed error/power estimators using as inputs the resulted regression lines from Table I are evaluated in comparison with: i) the same estimators trained with the overall design space, and ii) with well known regression techniques, i.e. Quadratic and Linear Regression, Regression Trees and a Neural Network with 10 neurons, trained with 13 points. As expected, the proposed estimators, have larger *MSE* when they use the regression lines than when they use the real measured power and error of the single technique approximate multipliers. Nevertheless, their *MSE* is still very small. Moreover, estimators using the regression lines require only 13 synthesis/simulation runs for training. In comparison with the rest estimators, trained with 13 hybrid

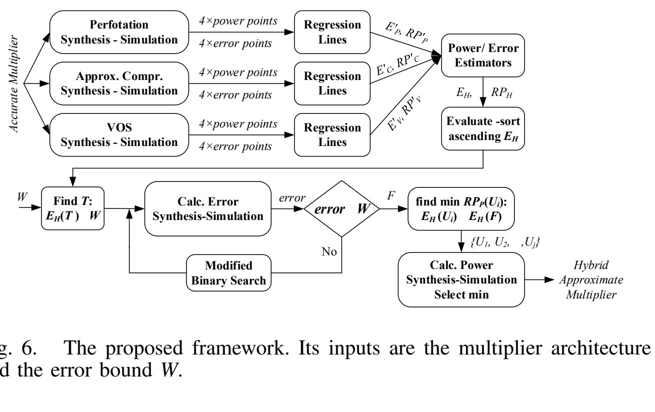

Accurate Multiplier
Perfotation Synthesis - Simulation
Approx. Compr. Synthesis - Simulation
VOS Synthesis - Simulation
4×power points
4×error points
Regression Lines
E'P, RP'P
E'C, RP'C
E'V, RP'V
Power/ Error Estimators
EH, RPH
Evaluate -sort ascending EH
W
Find T: EH(T) W
Calc. Error Synthesis-Simulation
error
error W
F
No
Modified Binary Search
find min RPP(Ui): EH(Ui) EH(F)
{U1, U2, ,Uj}
Calc. Power Synthesis-Simulation Select min
Hybrid Approximate Multiplier

Fig. 6. The proposed framework. Its inputs are the multiplier architecture and the error bound *W*.

TABLE I. REGRESSION LINES OF A 16-BIT DADDA 4:2 MULTIPLIER

Regression Lines	MSE
$E'_P(p)^a = 2 \times 10^{-6} e^{0.79p}$	1.3×10^{-10}
$RP'_P(p) = -0.057p + 1.003$	7.5×10^{-5}
$E'_V(v)^b = 2 \times 10^{-4} e^{0.16v}$	7.4×10^{-7}
$RP'_V(v) = -0.019v + 0.9826$	1.6×10^{-5}
$E'_C(c)^c = 6 \times 10^{-11} e^{0.74c}$	5.5×10^{-8}
$RP'_C(c) = -0.0015c^2 + 0.013c + 0.97$	1.4×10^{-3}

[a] p is the length of the perforation.
[b] v is the percentage decrease of the voltage vupply.
[c] c is the number of columns where logic approximation is applied.

TABLE II. COMPARISON OF THE ERROR AND *RP* ESTIMATORS

Estimator	Error-MSE	RP-MSE
Proposed Estimators using measured power/error	1.2×10^{-9}	6.7×10^{-4}
Proposed Estimators using regression lines	5.9×10^{-7}	1.0×10^{-3}
Quadratic regression	1.3	6.1
Linear regression	7.2×10^{-1}	1.0×10^{-1}
Regression Trees	2.4×10^{-1}	1.1
Neural Network	6.5×10^{-1}	5.6×10^{-1}

designs, it is shown that these generalized regression models attain worse accuracy than the proposed estimators using the regression lines.

Algorithm 1 is an heuristic optimizer used to find the quasi-optimal configuration of approximation techniques for the specified error bound. This algorithm: i) uses the proposed estimators and the calculated regression lines, ii) analytically, thus, very fast, evaluates the RP_H and E_H of all the hybrids designs, and iii) sorts them in ascending error. The first hybrid design with estimated error less than or equal to the error bound is extracted. Then, it is synthesized and simulated in order to evaluate its "real" error. In the case that its "real" error is not less than the upper bound, an iterative binary search procedure is invoked, finding the first hybrid design F with "real" error less than or equal to the error bound. Then, among the hybrid designs with estimated error less than or equal to the estimated error of F, those with the least estimated *RP* (if more than one) are synthesized and simulated. Finally, among the synthesized and simulated hybrid designs (with "real" error less than the error bound), the one with the the minimum "real" power is the output of the algorithm/framework.

The proposed framework does not depend on the multiplier's architecture nor the multiplier's width. The perforation and VOS techniques are directly applied on any architecture while the 4:2 compressors can replace any two successive full adders of the accumulation tree. The multiplier's width affects only the number of different configurations, and the only interaction between the framework and the input architecture is during the performed syntheses-simulations. In addition, the proposed optimization is scalable since searching is performed over analytical regression models. The sampling for training the models could be amenable to scalability issues. However, the error and power functions of each technique are strictly monotone retaining convexity independently from the bitwidth, thus, accurate regression can still be performed with a limited number of samples. As a result, the proposed framework can be applied to any multiplier and can even be used to explore the most efficient architecture for any specified error bound.

Algorithm 1 Returns quasi-optimal hybrid designs for specified error bounds

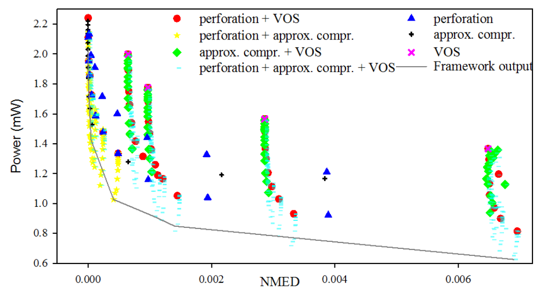
```
INPUT:  Estimator Functions: $E_H(p,c,v), RP_H(p,c,v)$
        perforation values: $p \in [p_{min}, p_{max}]$
        logic approximation values: $l \in [l_{min}, l_{max}]$
        VOS values: $v \in [v_{min}, v_{max}]$
        Error Bound: $MaxError$
    # Estimate the error and RP of every hybrid design
 1: for all $p, c, v$ do
 2:     AllHybrids←add(new Hybrid($p, c, v, E_H(p,c,v), RP_H(p,c,v)$))
 3: end for
    # Sort the hybrid designs for ascending error
 4: SortedError ← sort(AllHybrids, error)
    # Find the first hybrid design with estimated error ≤ MaxError
 5: f ← search(SortedError, MaxError)
    # Among the hybrid designs with estimated error ≤ MaxError,
    # find the first design F with "real" error ≤ MaxError
 6: SimT ← synth&sim(SortedError[f])
 7: if SimT.Error > MaxError then
 8:     imin ← 0, imax ← f
 9:     while imin < imax do
10:         imid ← (imin+imax)/2
11:         SimR ← synth&sim(SortedError[imid])
12:         if SimR.Error ≤ MaxError then
13:             imin ← imid + 1
14:         else
15:             imax ← imid - 1
16:         end if
17:     end while
18:     f ← imid
19: end if
    # Among the designs with estimated error ≤ estimated error of F,
    # find those with the minimum RP and synthesize-simulate them
20: SortedPower ← sort(SortedError[0:f], power)
21: for i←0 to f do
22:     if SortedPower[i].RP = SortedPower[0].RP then
23:         powerCandidates.add(synth&sim(SortedPower[i]))
24:     end if
25: end for
    # return the design with the minimum "real" power
26: return min (powerCandidates, power)
```

IV. EXPERIMENTAL EVALUATION

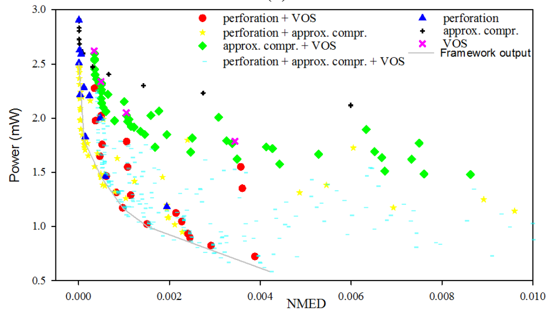
In this section we experimentally evaluate the efficiency of the proposed techniques, by providing comparative results against an exhaustive full-search exploration of all the possible hybrid designs, thus, proving the necessity and optimization potentials exposed by the adoption of hybrid approximation solutions. In full-search, more than 7000 hybrid designs in total (including both the SPP and MBE designs) have been evaluated, which translate to more than 3 weeks of experiments on an intel i7-2600k workstation with 16GB of RAM.

Fig. 7 depicts the power - error (*NMED*) results of: i) all the explored hybrid designs, and ii) all the designs with only one approximate technique, that feature $NMED < 10^{-2}$. Each point in the graph represents a design with different configuration for each approximate method, grouped through the use of different colors. The points with the same color correspond to the application of the same approximate techniques, e.g. the yellow point constitute the application of both the perforation and the approximate 4:2 compressors on the multiplier. Different yellow points represent approximate hybrid designs with different configuration for either the perforation or the number of columns in which the imprecise compressors are used.

The grey line is the output of the proposed framework. As shown in Fig. 7, for both the SPP and the MBE designs, the application of hybrid approximate techniques exhibits higher quality design solutions. For the same error value, the lowest power is achieved by a hybrid design and there is no error region in which a single approximate technique achieves a (local) minimum. The hybrid designs produced by the framework are close to the optimal ones (Pareto points of Fig. 7) that resulted from the conducted exploration, and are always below (or equal to) the approximate design with only one technique. For example, for the SPP designs and for error bound $NMED = 3 \times 10^{-5}$ the hybrid multiplier produced from the framework, consumes 11% less power than the single-technique design with the least power. For $NMED = 3 \times 10^{-4}$ and $NMED = 7 \times 10^{-3}$ the power savings are 30% and



(a)



(b)

Fig. 7. The power and error of all the single technique and hybrid approximate designs with $NMED < 0.01$ when a) SPP and b) MBE is preferred.

48% respectively. When the MBE is preferred, for error bound $NMED \geq 5\times10^{-4}$ the power savings are more than 41%. For the error bounds examined in Fig. 7 the proposed framework required 13 syntheses-simulations to calculate the regression lines and then an average of 4.71 syntheses-simulations to produce its output. Finally, for low error values, hybrid models with two approximate techniques exhibit the lowest power values, while for larger ones, the hybrid designs that apply all three techniques attain the minimum power consumption. In conclusion, for small error values, the hybrid designs achieve lower power values than those with only one technique and for larger error values, they can offer further power reduction.

In Table III, for two different error bounds, the hybrid designs produced from the proposed framework are directly compared with the approximate designs that apply the state-of-art techniques of [8] and [2]. For maximum error 3.0×10^{-5}, among the single-technique designs, logic approximation on 17 columns offers lower area and power than perforation (with length 3), while VOS is not applicable as it cannot offer error less than this bound. The output of our framework is a hybrid design that applies perforation with length 3 and logic approximation on 16 columns but does not apply VOS. This hybrid design attains 15% and 18% lower area than logic approximation and perforation and its power consumption is 11% and 22% lower, respectively. For higher error bound (4.0×10^{-3}), among the single-technique designs, VOS with 0.82V supply voltage achieves lower power than logic approximation on 25 columns, while perforation with length 10 delivers the lowest area and power. The hybrid design resulting from the framework applies perforation with length 7, logic approximation on 17 columns and VOS with 0.85V. Comparing this hybrid design with VOS and logic approximation, it achieves 53% and 29% lower area and 36% and 38% lower power respectively. Finally, compared with perforation, the hybrid design delivers 22% lower power but 7% more area.

V. Conclusion

In this paper, we addressed the problem of power-optimal approximate arithmetic multipliers, that for given error bounds push power gains to the limits. We first introduced the application of perforation as an efficient technique in the algorithmic level for creating approximate multipliers. We then explored hybrid techniques that apply more than one methods. We conducted a full exploration over all the hybrid and single-technique designs and deduced that the former achieve higher power savings for same error values. We presented a design optimization framework that requires only a small number of synthesis and simulation runs to provide quasi-optimal solutions. The proposed framework can be applied to any multiplier architecture and delivers hybrid designs achieving significant power reduction compared with the state-of-art arithmetic approximation techniques.

TABLE III. Evaluation of the Hybrid Designs at 500MHz

Error Bound	Design	Area (um^2)	Power (mW)
3.0×10^{-5}	Hybrid (3,16,1.00)	2340	1.46
	Logic Approx. (17)	2737	1.64
	Product Perforation (3)	2842	1.86
	VOS	-	-
4.0×10^{-3}	Hybrid (7,17,0.85)	1615	0.72
	Logic Approx. (25)	2261	1.17
	Product Perforation (10)	1507	0.92
	VOS (0.82)	3401	1.12

Acknowledgements

This research has been partially supported by the E.C. funded program AEGLE under H2020 Grant Agreement No: 644906

References

[1] R. Venkatesan *et al.*, "Macaco: Modeling and analysis of circuits for approximate computing," in *ICCAD*, Nov. 2011, pp. 667–673.

[2] Y. Liu *et al.*, "Computation error analysis in digital signal processing systems with overscaled supply voltage," *IEEE Trans. VLSI Syst.*, vol. 18, no. 4, pp. 517–526, Apr. 2010.

[3] V. Gupta *et al.*, "Impact: Imprecise adders for low-power approximate computing," in *ISLPED*, Aug. 2011, pp. 409–414.

[4] N. Zhu *et al.*, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. VLSI Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[5] A. K. Verma *et al.*, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Design, Automation and Test in Europe*, Mar. 2008, pp. 1250–1255.

[6] B. Shao and P. Li, "A model for array-based approximate arithmetic computing with application to multiplier and squarer design," in *ISLPED*, 2014, pp. 9–14.

[7] P. Kulkarni *et al.*, "Trading accuracy for power with an underdesigned multiplier architecture," in *24th Int. Conf. on VLSI Design*, Jan. 2011, pp. 346–351.

[8] A. Momeni *et al.*, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. PP, Feb. 2014.

[9] C. Liu *et al.*, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Design, Automation and Test in Europe*, Mar. 2014.

[10] S. Sidiroglou *et al.*, "Managing performance vs. accuracy trade-offs with loop perforation," in *Foundations of software engineering (ESEC/FSE)*, Sep. 2011, pp. 124–134.

[11] K. Bretthauer and B. Shetty, "The nonlinear knapsack problem - algorithms and applications," *European Journal of Operational Research*, vol. 138, no. 3, pp. 459–472, May 2002.

[12] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. NY: Oxford University Press, 2000.

[13] J. Liang *et al.*, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Jun. 2012.

[14] G. Zervakis *et al.*, "Approximate multiplier architectures through partial product perforation: Power-area tradeoffs analysis," in *Proc. of the 25th Edition on Great Lakes Symposium on VLSI*, 2015, pp. 229–232.

[15] G. Mekhtarian, *Composite Current Source (CCS) Modeling Technology Backgrounder*. Synopsys, Inc., Nov. 2005.