Scalable Construction of Approximate Multipliers with Formally Guaranteed Worst-Case Error

Vojtech Mrazek, Zdenek Vasicek *Member, IEEE*, Lukas Sekanina, *Senior Member, IEEE*, Honglan Jiang, *Student Member, IEEE*, and Jie Han, *Senior Member, IEEE*

***Abstract*—Approximate computing exploits the fact that many applications are inherently error resilient. In order to reduce power consumption, approximate circuits such as multipliers have been employed in these applications. However, most current approximate multipliers are based on ad hoc circuit structures and, for automated circuit approximation methods, large efficient designs are difficult to find due to the increased search space. Moreover, existing design methods do not typically provide sufficient formal guarantees in terms of error if large approximate multipliers are constructed. To address these challenges, this paper introduces a general and efficient method for constructing large high-quality approximate multipliers with respect to the objectives formulated in terms of the power-delay product and a provable error bound. This is demonstrated by means of a comparative evaluation of approximate 16-bit multipliers constructed by the proposed method and other methods in the literature.**

I. Introduction

Energy efficiency is a major challenge for current computer systems. Among various techniques, approximate computing exploits the fact that many applications are inherently error resilient and energy requirements can be traded off for the quality of results [1]. Much attention has been paid to the design of approximate arithmetic circuits and in particular, approximate multipliers, as multiplication is a key operation in many applications.

Approximate implementations of multipliers are based on various design principles, see a recent review in [1]. The major weakness of the *manual circuit design* approach, which is clearly dominating in this area, lies in providing only a few different circuit implementations for a given bit width. Many interesting and useful design points thus remain unexplored. Hence, automated *search-based design* methods have been developed to provide many approximate designs showing high-quality tradeoffs between key design parameters [2].

We are primarily interested in approximate circuits belonging to the *Pareto set* which contains the so-called *non-dominated* solutions. Consider three objectives to be minimized, for example, the power-delay product (PDP), the worst case error and the area. Circuit C_1 *dominates* another circuit C_2 if: (1) C_1 is no worse than C_2 in all objectives, and (2) C_1 is strictly better than C_2 in at least one objective.

V. Mrazek, Z. Vasicek and L. Sekanina are with Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno, Czechia e-mail: {imrazek,vasicek,sekanina}@fit.vutbr.cz.

H. Jiang and J. Han are with University of Alberta, Edmonton AB, Canada e-mail {honglan, jhan8}@ualberta.ca.

Manuscript received XXXX; revised XXXX.

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic – INTER-COST project LTC18053.

Search-based methods, however, usually deliver circuits of limited complexity, which especially holds for approximate multipliers. In order to mitigate this issue, an efficient strategy is to compose complex approximate circuits using less complex but high-quality approximate design modules.

For an approximate circuit, another challenge is to find an efficient way to determine the quality (or error) of a design. While small designs can be perfectly evaluated by means of an exhaustive simulation, this problem is not tractable for complex circuits. A circuit simulation using a subset of all input combinations does not, in principle, guarantee an accurate result. Hence, various formal methods capable of determining the 'exact' error have been developed in recent years [3], [4], [5].

To address these challenges, this paper presents a general and efficient method for constructing high-quality non-dominated approximate multipliers with the aim to optimize the power-delay product (PDP) and design quality. A significant advantage of this method is the ability to analytically provide formal guarantees in terms of the worst case error for even complex multipliers. The proposed method exploits the fact that more than two thousands 8-bit approximate multipliers are available and they can directly be employed to construct large approximate multipliers showing various tradeoffs between the design objectives.

II. Evaluation of approximate circuits

Various error metrics have been developed to evaluate the quality of approximate circuits [2], for example, the worst-case error (WCE), sometimes denoted as the maximum error distance, the worst-case relative error (WCRE), the average-case error, also known as the mean absolute error (MAE), and the mean relative error (MRE).

The worst-case error of an n-bit approximate multiplier $\widetilde{M}$ is defined as the maximum difference between the outputs of $\widetilde{M}$ and a precise multiplier M

$$WCE_{\widetilde{M}} = \max_{\forall a,b} |\widetilde{M}(a,b) - M(a,b)|, \tag{1}$$

where $0 \leq a, b < 2^n$ and $M(a,b) = a \times b$. The worst-case error can be important in time-critical and dependable systems on one hand, but also in image and signal processing on the other, where low average error but excessive worst-case error can produce unacceptable results. The worst-case relative error is defined as

$$WCRE_{\widetilde{M}} = \max_{\forall a,b} \frac{|\widetilde{M}(a,b) - M(a,b)|}{M(a,b)}. \tag{2}$$

Test vectors are usually applied to estimate the error (e.g. 10^7 input vectors were used to evaluate 16-bit multipliers in [1]). Unfortunately, the accuracy of the simulation-based error evaluation varies with the number and quality of test vectors. This is especially noticeable in the case of WCE, where completely different results may be obtained for a different subset of input vectors.

Current computers only require a few minutes to exactly determine the quality of arithmetic circuits with 16-bit operands. For higher bit-widths, a more sophisticated, typically formal, approach has to be involved. The main advantage of the formal approach is that an 'exact' error or error bound can be obtained. Checking the worst-case error can be done using Boolean satisfiability (SAT) solvers as demonstrated in [6]. Determining the error probability using binary decision diagrams (BDDs) is a relatively straightforward task. For example, Ciesielsky et al. described a method based on BDDs that is able to establish the error probability even for large 64-bit adders [3]. Chandrasekharan et al. [4] employed BDDs to determine the worst-case arithmetic error. Vasicek et al. [5] proposed a method for determining the average-case arithmetic error.

III. Approximate Multipliers

Three stages can be identified in a multiplier: partial product generation, partial product reduction, and final addition. Four main methods are used for the design of approximate multipliers [1]: (1) Approximation in generating partial products based on a simpler structure. (2) Approximation in the partial product tree by ignoring some partial products (truncation), dividing the partial products into several modules and applying an approximation in the less significant modules, or composing complex approximate multipliers from simple approximate multipliers. (3) Using approximate adders, counters or compressors in the partial product tree to reduce partial products. (4) Using search-based methods to perform approximation on the gate level or in more complex cells. In the sequel, we briefly introduce the state-of-the-art approximate multipliers that provide the best trade-off between quality and other design parameters such as power, delay and area.

For *truncated multipliers* (TMs), the key idea is to remove k least significant bits of the input operands. As a result, a smaller $(n-k)$-bit multiplier is utilized instead of an accurate n-bit multiplier. A 5-bit approximate multiplier is implemented as a truncated carry-save adder array in Fig. 1(a). In general, an array multiplier consists of $n \times n$ cells (i.e. the carry-save adder array) used to reduce partial products, followed by a single n-bit merging adder (a ripple-carry adder in our example) for the final summation. Due to the truncation, the cells associated with k least significant bits of the first operand (i.e. the cells in the first k rows) and the cells associated with k least significant bits of the second operand (i.e. the k rightmost cells of each row) are omitted. As a result, $2k$ least significant bits of the final product are always zero.

The accuracy of a TM depends on the bit-width (n) and the number of truncated bits (k), where $0 \leq k < n$. The

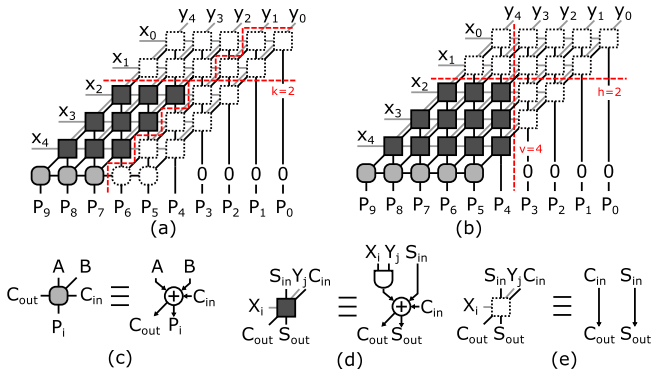

Fig. 1. Hardware architectures of two 5-bit approximate multipliers. a) Truncated multiplier TM(5,2) and b) broken-array multiplier BAM(5,4,2). Omitted cells are shown using dotted cells. Note the inputs of the cells situated in the first row that are not shown are implicitly connected to zero.

maximum difference between the output of $\mathrm{TM}(n,k)$ and a precise multiplier is equal to

$$WCE_{\mathrm{TM}(n,k)} = (2^k - 1)(2^{n+1} - 2^k - 1). \quad (3)$$

In the *broken-array multiplier* (BAM), some of the carry-save adders are removed from an array multiplier [7]. The omitted cells are specified using two parameters: the horizontal break level (h) and vertical break level (v), where $0 \leq h < n$ and $h \leq v < 2n$. An example of a 5-bit BAM is shown in Fig. 1(b). In the case that the vertical break level is $2\times$ of the horizontal break level (i.e. $v = 2h$), a structure similar to TM with $k = v$ is obtained. As shown in our example, however, BAM preserves more carry-save adder cells. Since the reduction of carry-save adders can be done in both directions, the accuracy of $\mathrm{BAM}(n,h,v)$ depends on the three parameters. According to [7], the maximum difference is given by

$$WCE_{\mathrm{BAM}(n,h,v)} = (2^n - 1)\sum_{i=0}^{h-1} 2^i + 2^h \sum_{i=0}^{v-h-1} (2^{v-h} - 2^i). \quad (4)$$

The maximum relative error is $WCRE_{BAM} = 1$. The mean relative error significantly increases with increasing v [7].

Recently, a rich library of approximate 8-bit adders and 8-bit multipliers containing hundreds of alternative implementations was introduced [2]. The authors employed a general-purpose approximation method for combinational circuits based on a multi-objective genetic programming. The goal was to simultaneously minimize delay, power consumption and error to discover a set of approximate circuits along a Pareto front. The basic version of the library contains 471 annotated non-dominated 8-bit approximate multipliers that are available for download. Compared to other design methods, a search-based method explores a larger design space, so it is likely to produce approximate multipliers with better hardware characteristics.

Many other approximate multipliers have been proposed. A recent survey of existing implementations can be found, for example, in [1]. Unfortunately, the majority of these multipliers was optimized for a single quality parameter only. When multiple quality metrics such as the WCE, MAE, and MRE are considered, they typically show little advantage in the overall performance over a truncated design [1].

IV. Construction of Larger Approximate Multipliers

In order to avoid the time-consuming design loop inherently related with the search-based techniques, we propose to recursively construct complex approximate multipliers, using smaller multipliers.

We employ a divide-and-conquer strategy for synthesizing a $2n$-bit multiplier from four n-bit multipliers (see Fig. 2). The operands are divided into four n-bit chunks (each operand has a lower and higher part) that are independently processed using four multipliers whose outputs are reduced using two adders with one n-bit and one $2n$-bit operand each.

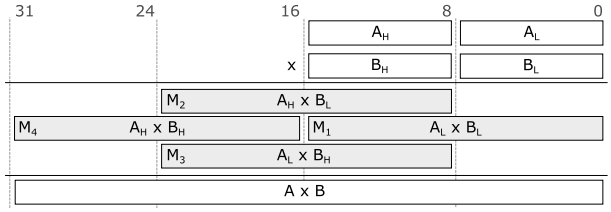

Fig. 2. Construction of a $2n \times 2n$ multiplier from four $n \times n$ multipliers denoted as M_1, M_2, M_3 and M_4. The principle is illustrated for $n = 8$.

The key advantage of this method is that each constructed multiplier M has the following properties. (1) In the case that all four n-bit multipliers (let us denote them M_1 to M_4) and adders are accurate, an accurate multiplier is obtained. (2) If accurate adders are employed and some of the multipliers are replaced with an approximate multiplier $\widetilde{M}$, the worst-case error is equal to

$$WCE_M = 2^{2n}WCE_{M4} + 2^nWCE_{M3} + 2^nWCE_{M2} + WCE_{M1}, \quad (5)$$

where $WCE_{M_i} = WCE_{\widetilde{M}}$ iff M_i is replaced with $\widetilde{M}$ and $WCE_{M_i} = 0$ otherwise. (3) If accurate adders are employed and some of the multipliers are replaced with different approximate multipliers, Eq. 5 gives us the upper bound since the approximate multipliers can have dependent inputs.

Clearly, the exact value of $WCE_{\widetilde{M}}$ can easily be obtained for a reasonable n (in our case $n = 8$) using exhaustive simulation. The question is, how to efficiently determine which multiplier M_i should be replaced with an approximate multiplier and what approximate multipliers should be used to obtain the best trade-offs without compiling and synthesizing all the possible design points. In general, $(n_a + n_e)^4$ possible solutions exist provided that n_a denotes the number of non-dominated approximate n-bit multipliers and n_e is the number of exact n-bit multipliers. As more than 2,200 different non-dominated 8-bit multipliers are available at the extended version of EvoApprox8 library and more than 60 different implementations can be obtained using the approaches proposed in the literature, it is practically infeasible to synthesize all possible implementations (more than 1.4×10^{11} potential solutions exist).

Hence, we propose the following two strategies. In the *first strategy*, a single approximate multiplier $\widetilde{M}$ is chosen and M_1 to M_4 are successively replaced with $\widetilde{M}$ (see A1 – A4 in Table I). The advantage of this approach is that it produces a relatively small number of implementations that can easily be synthesized and evaluated. Due to its simplicity, a similar scenario (i.e., some submodules are replaced with a single approximate circuit) is typically employed in quality configurable multipliers such as AWTM [8] or lpAClib [9].

TABLE I
Proposed Architectures of $2n$-bit Approximate Multipliers

Architecture	M_1	M_2	M_3	M_4
A1	$\widetilde{M}$	*accurate*	*accurate*	*accurate*
A2	$\widetilde{M}$	$\widetilde{M}$	*accurate*	*accurate*
A3	$\widetilde{M}$	$\widetilde{M}$	$\widetilde{M}$	*accurate*
A4	$\widetilde{M}$	$\widetilde{M}$	$\widetilde{M}$	$\widetilde{M}$
A5	$\widetilde{M}_1$	$\widetilde{M}_2$	$\widetilde{M}_2$	$\widetilde{M}_3$
A6	$\widetilde{M}_1$	$\widetilde{M}_2$	$\widetilde{M}_3$	$\widetilde{M}_4$

* Note that $\widetilde{M}_i$ may represent approximate or accurate multiplier.

Eq. 5 suggests that much better results could be potentially obtained when up to four different approximate 8-bit multipliers are utilized in the 16-bit multiplier. Intuitively, M_4 should produce the lowest error since it has the largest impact on the quality of the obtained multiplier. In order to reduce a huge number of design alternatives that have to be synthesized, the *second strategy* is proposed. In its first phase, we filter out all approximate 16-bit multipliers that are dominated by some other multipliers. This can be done relatively quickly and without using a professional design tool if pre-computed circuit parameters (WCE calculated according to Eq. 5, power consumption calculated as the sum of power consumption of all 8-bit multipliers M_i, and estimated area calculated as the sum of the multipliers' areas) are used instead of 'exact' values. In the second phase, only the circuits identified in the first phase are synthesized and thoroughly evaluated.

In the multiplier denoted as A6 in Table I, four different approximate multiplier modules are utilized. For example, if there are 258 different 8-bit approximate multipliers, there exist $258^4 = 4,430,766,096$ different 16-bit approximate multipliers using the A6 architecture. However, most of the 16-bit approximate multipliers are dominated by others and can be filtered out. As a result, only 6,753 multipliers remain for further evaluation. To significantly reduce the number of 16-bit multipliers that have to be investigated in the first phase, we also consider the architecture A5 that contains only three different approximate 8-bit multipliers (see Table I).

V. Experiments and Simulation Results

Although the proposed method is applicable to an arbitrary bit-width, the construction of 16-bit multipliers is investigated because the 16-bit multipliers can be evaluated by simulation that enables to thoroughly evaluate the proposed method and compare the obtained circuits with existing designs.

A. Evaluation of 8-bit approximate multipliers

Firstly, we implemented all relevant 8-bit multipliers in VHDL and synthesized them together with multipliers from the EvoApprox8b library. Synopsys Design Compiler with 45nm PDK was employed for synthesis. Each VHDL model was converted to an equivalent C code that was utilized for

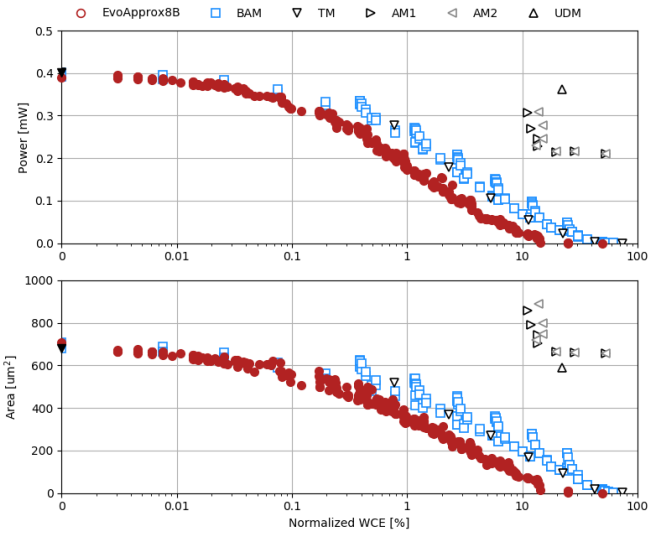

Fig. 3. Measurements of various 8-bit multipliers synthesized using 45nm technology. The multipliers forming a Pareto set considering power, area and WCE are highlighted using filled markers. Normalized WCE is calculated as $\text{NWCE} = \text{WCE}/2^{2n}$, where n is equal to 8. Measurements of five different exact multipliers are provided as the points at WCE=0.

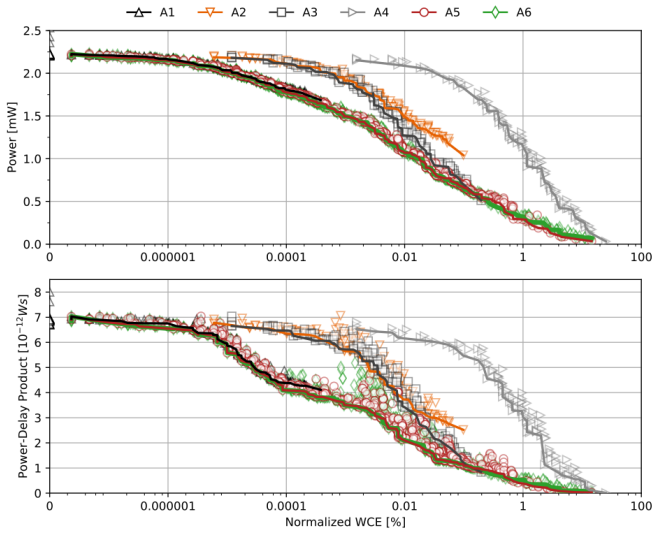

Fig. 4. Measurements of 16-bit approximate multipliers consisting of four 8-bit multipliers using the proposed approaches A1–A6. The implementations lying on the Pareto set, determined for each method and each plot separately, are shown using a line.

quality analysis. Exact values of WCE, WCRE, ER, MAE and MRE were determined by using all 2^{16} input vectors. In this phase, more than 2,210 implementations were synthesized and analyzed. The obtained results are shown in Fig. 3. The naming of the multipliers corresponds with [1]. Only non-dominated solutions are shown for each architecture for clarity. The results show the advantages of TM, BAM and EvoApprox8b over the other designs when WCE and circuit area or power consumption are considered. Considering WCE, MAE and MRE, EvoApprox8b designs outperform the other ones. In addition to that, they fill the missing spaces that are unreachable by truncation. Fig. 3 also contains the measurements of an accurate multiplier implemented using the star operator in VHDL (see the triangle at WCE=0). Interestingly, many approximate multipliers (some instances of AM1, AM2, UDM) require a larger area compared to the accurate multipliers.

B. Synthesis of 16-bit approximate multipliers

From the previous results, 258 non-dominating design points were identified (by considering WCE, power-delay product (PDP) and area only) and employed as $\widetilde{M}$ or $\widetilde{M}_i$. This yields $4 \times 258 = 1,032$ different implementations of A1–A4. In addition to that, 4,449 different implementations of A5 and 6,753 different implementations of A6 were produced in the first phase. Then, the 16-bit implementations were synthesized using Synopsys Design Compiler and analyzed using a simulator with 2^{32} input vectors to obtain exact values of WCE, WCRE, MAE and MRE. The accurate carry lookahead adder (CLA) was employed in all designs. The runtimes for filtration, synthesis and analysis are given in Table II. After synthesis and analysis, 192 (resp. 158, 158, 142, 810, and 1,257) non-dominated solutions[1] were identified for A1 (resp. A2, A3, A4, A5 and A6). Measurements of the non-dominated designs are shown in Fig. 4. For illustration, we also included dominated solutions.

The results validate our assumption regarding the quality of approximate multipliers using the proposed architecture A6 compared to the basic construction mechanisms A1–A4. Interestingly, there is no significant difference between the multipliers using A5 and A6. Considering this fact, A5 is a very efficient architecture for constructing multipliers of higher bit-widths. Compared to A6, it requires approximately 2x smaller computational power as shown in Table II. Although only exact WCE is plotted due to the limited space, this observation is valid for MAE and MRE too (see Fig. 5). The remaining results can be found at our website[2].

For some instances, Eq. 5 provides the upper bound and the exact WCE may be lower than the estimated one. Although this can not have a negative impact on a real application, the knowledge of the exact and estimated WCE offers an opportunity for a more detailed analysis. Considering 6,753 different implementations of A6, the non-zero difference occurs in 85%. The mean (resp. median, maximum) difference is 2.8% (resp. 2.2%, 16.6%). Difference greater than 5% occurs in 13%.

TABLE II
THE NUMBER OF NON-DOMINATED MULTIPLIERS (n_{dom}) IDENTIFIED IN THE FIRST / SECOND PHASE AND THE CORRESPONDING RUNTIME (FILTRATION t_{filt}, SYNTHESIS t_{syn}, AND ANALYSIS t_{ev}) IN MINUTES

Arch.	First phase (filtration)			Second phase			Total runtime
	n_{total}	n_{dom}	t_{filt}	n_{dom}	t_{syn}	t_{ev}	
A1	258	258	-	192	10	42	52
A2	258	258	-	158	10	42	52
A3	258	258	-	158	10	42	52
A4	258	258	-	142	10	42	52
A5	$1.7 \cdot 10^7$	4,449	2	810	180	720	902
A6	$4.4 \cdot 10^9$	6,753	469	1,257	273	1,093	1,846

[1]Power, area, delay, WCE and MRE were considered.

[2]http://www.fit.vutbr.cz/research/groups/ehw/approxlib/

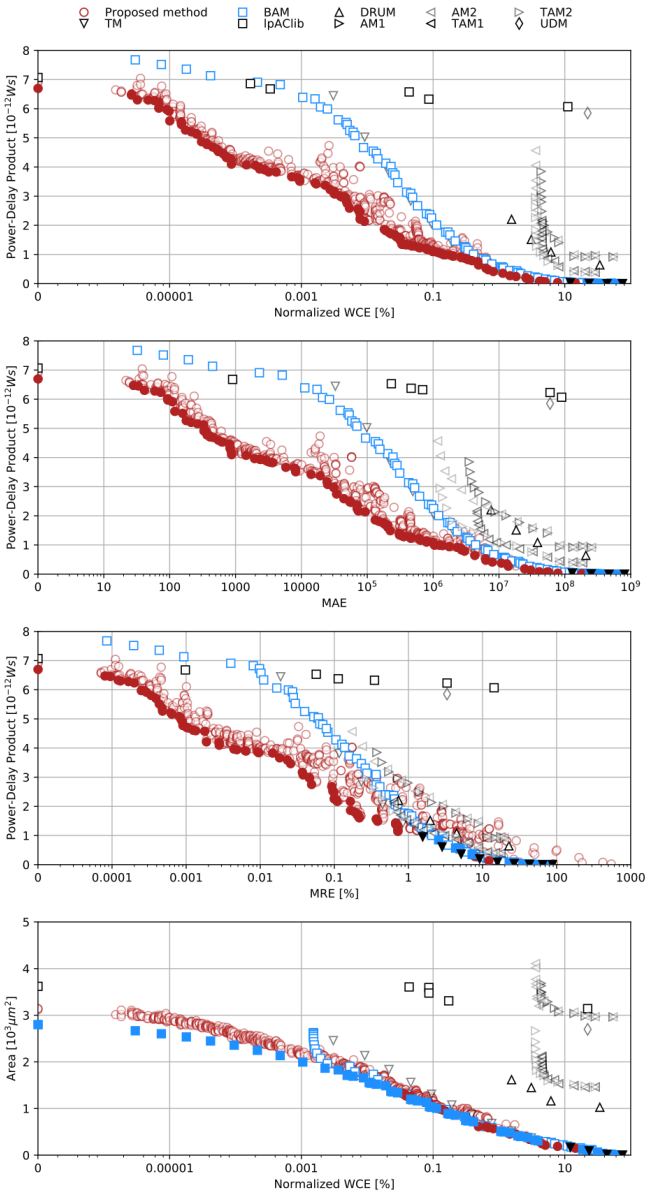

Fig. 5. Comparison of the proposed 16-bit approximate multipliers (using the A5 architecture) with multipliers from the literature across various quality indicators. The multipliers on a Pareto set, determined for each plot separately, are highlighted using filled markers. Parameters of exact multipliers are provided as the points at WCE=0.

C. Comparison with the state-of-the-art approaches

The detailed comparison of the proposed method with the state-of-the-art designs is shown in Fig. 5. For evaluation with the typical error metrics, we reimplemented the approximate multipliers that are believed to provide the best results according to the latest review [1]. In total, 244 different 16-bit designs were created, synthesized and evaluated. For configurable architectures (AM, BAM, TAM, ACM, and lpAClib), all meaningful configurations were considered. The error metrics are evaluated accurately using all test vectors for all considered designs.

The results are mostly consistent with the review [1], although some multipliers exhibit lower quality, see e.g. TAM1. The differences are probably caused by the fact that a different technology node generation is considered with a different cell library. We also determine the errors exactly, whereas only a small fraction (0.2%) of all possible input combinations is employed in the review to assess the quality of the approximate multipliers. While the quality of particular designs varies depending on the chosen error criteria, the truncated multipliers (TM and BAM) exhibit stable performance and achieve excellent design tradeoffs.

The multipliers constructed using the proposed method provide the best tradeoffs except for the Area vs. WCE result, whereas BAMs occupy smaller area. Despite the fact that only the worst-case error, power and area were considered to obtain non-dominated designs, the obtained multipliers perform well even under MAE and MRE. In fact, it is shown that MAE strongly correlates with WCE.

In order to construct a database of 810 annotated 16-bit multipliers, 15 hours (including quality evaluation taking more than 80% of the total runtime) were required on an eight-core Intel Xeon CPU @ 2.4GHz

VI. Conclusions

In this paper, a scalable recursive method for the construction of large approximate multipliers with guaranteed worst-case error was proposed. We demonstrated how to relatively quickly construct a high-quality Pareto set of non-dominated $2n$-bit approximate multipliers provided that we have a reasonable database of n-bit approximate multipliers.

We show that it is sufficient to construct a $2n$-bit multiplier using three different n-bit multipliers (in the architecture A5) without sacrificing much quality of the obtained $2n$-bit multipliers. This method enables to reduce the design time to nearly one half of the A6 architecture, in which four different 8-bit multipliers are selected. The constructed designs show worst case errors limited by a maximum error bound that can be analytically obtained due to the proposed design approach.

References

[1] H. Jiang, C. Liu *et al.*, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 60:1–60:34, Aug. 2017.

[2] V. Mrazek, R. Hrbacek *et al.*, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. of DATE'17*, 2017, pp. 258–261.

[3] C. Yu and M. Ciesielski, "Analyzing imprecise adders using BDDs – a case study," in *Proc. of ISVLSI'16*. IEEE, 2016, pp. 152–157.

[4] A. Chandrasekharan, M. Soeken *et al.*, "Approximation-aware rewriting of AIGs for error tolerant applications," in *Proc. of ICCAD'16*. ACM, 2016, pp. 83:1–83:8.

[5] Z. Vasicek, V. Mrazek, and L. Sekanina, "Towards low power approximate DCT architecture for HEVC standard," in *Proc. of DATE'17*. EDAA, 2017, pp. 1576–1581.

[6] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *Proc. of ICCAD'11*. ACM, 2011, pp. 667–673.

[7] 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, April 2010.

[8] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *15th Int. Symp. on Quality Electronic Design*, March 2014, pp. 263–269.

[9] M. Shafique, W. Ahmad *et al.*, "A low latency generic accuracy configurable adder," in *Proc. of DAC'15*. ACM, 2015, pp. 86:1–86:6.