Generalized Matrix Factorization Techniques for Approximate Logic Synthesis

Soheil Hashemi
School of Engineering
Brown University
Providence, RI 02912
soheil_hashemi@brown.edu

Sherief Reda
School of Engineering
Brown University
Providence, RI 02912
sherief_reda@brown.edu

***Abstract*—Approximate computing is an emerging computing paradigm, where computing accuracy is relaxed for improvements in hardware metrics, such as design area and power profile. In circuit design, a major challenge is to synthesize approximate circuits automatically from input exact circuits. In this work, we extend our previous work, BLASYS, for approximate logic synthesis based on matrix factorization, where an arbitrary input circuit can be approximated in a controlled fashion. Whereas our previous approach uses a semi-ring algebra for factorization, this work generalizes matrix-based circuit factorization to include both semi-ring and field algebra implementations. We also propose a new method for truth table folding to improve the factorization quality. These new approaches significantly widen the design space of possible approximate circuits, effectively offering improved trade-offs in terms of quality, area and power consumption. We evaluate our methodology on a number of representative circuits showcasing the benefits of our proposed methodology for approximate logic synthesis.**

I. Introduction

Many applications in domains such as signal processing, machine learning, computer vision, and computer graphics show inherent resilience to small errors in their outputs. This resilience can originate from different sources including, noise in input data, inherent approximate calculations, or user tolerance to variations in the outputs. In such domains, approximate computing proposes to exploit this error resilience by trading accuracy for benefits in hardware metrics.

In circuit design, a central challenge is to devise techniques for approximate circuit synthesis that can generate approximate circuits from input exact circuits, while enabling the designer to control the trade-off in the amount of inaccuracies introduced in the circuit and the associated design metrics. The last few years have seen a number of techniques that address the need for approximate synthesis techniques [3], [4], [8], [10], [11]. These techniques work across different design abstractions from gate-level to higher levels [2], [7], [9].

Recently, an approach, BLASYS, for approximate logic synthesis has been proposed based on matrix factorization [1]. In this approach, a truth table is first generated for the circuit. The truth table is then treated as a matrix, $\mathbf{A}$, that is factored into two smaller matrices $\mathbf{B}$ and $\mathbf{C}$, that when multiplied, $\mathbf{BC}$, using semi-ring Boolean algebra rules, leads to an approximate truth table. The truth table corresponding to $\mathbf{B}$ is synthesized as a *compressor* circuit and the matrix $\mathbf{C}$ is synthesized as a *decompressor* circuit that receives its inputs from the compressor circuit.

In this paper, we generalize our matrix-based approach for approximate circuit synthesis by considering both semi-ring and field algebra implementations. In particular, the contributions of this paper are as follows.

- We propose a generalized approach to circuit approximations using matrix factorization, where we use XOR-based field algebra for circuit approximations alongside the OR-based semi-ring algebra approach. In our approach, a decompressor circuit of ORs and XORs is used to combine the outputs of the compressor circuit to generate the final approximate circuit outputs.
- To improve the possibility of obtaining better approximations, we propose to fold the truth table to "balance" its size. This approach reduces the complexity of the compressor circuit at the expense of the decompressor circuits, which can lead to better approximations.
- We incorporate OR and XOR based factorizations and the degree of folding of the truth table into a design space exploration method to identify a larger space of possible approximate circuits designs.
- We evaluate our approach on a number of representative circuits, where we quantify the trade-off between accuracy and design metrics such as area and power. We show that our generalized approach improves BLASYS [1] by 3-12% depending on the target accuracy threshold.

The organization of this paper is as follows. In Section II, we describe the proposed approach, namely the XOR field-based circuit approximation, and truth table folding methods. Here, we also describe the integration of our approach in a circuit decomposition and design space exploration technique. Next, we provide our experimental results in Section III. Finally, the conclusions of this work are summarized in Section IV.

II. Proposed Methodology

Binary matrix factorization can be formulated as an optimization problem solving

$$\text{argmin}_{\mathbf{B},\mathbf{C}}|\mathbf{A} - \mathbf{BC}|, \tag{1}$$

where the elements of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ matrices are '0' or '1'. Here, the multiplications are carried out using the logical AND operation, and different algebra (such as Boolean and modulo-2) can be used for the addition [5], [6].

In our previous work, BLASYS [1], we proposed to approximate a circuit by replacing the exact circuit with truth table $\mathbf{A}$

978-3-9819263-2-3/DATE19/©2019 EDAA

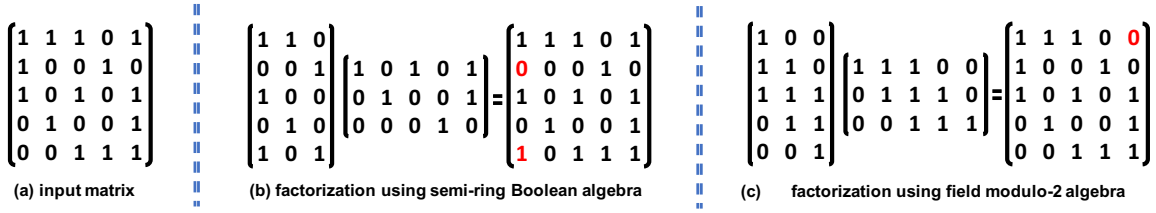

Fig. 1. Example of binary matrix factorization using different algebra. (a) input matrix, (b) factorization using Boolean algebra where addition is carried out using logical ORs, and (c) factorization using modulo-2 algebra, where the addition is carried out using logical XORs. The errors are highlighted in red.

with a reduced circuit representing $\mathbf{BC}$. While in our original paper we utilized Boolean matrix factorization (BMF), and hence an OR based decompressor, in this work we generalize and expand the scope of application of matrix factorization techniques for approximate logic synthesis.

A. Binary Matrix Factorization under Different Algebras

Binary matrix factorization can use different algebra when optimizing the factorized matrices to better approximate the original matrix. In the case of Boolean matrix factorization (BMF) proposed in our previous work [1], the algebra implements a semi-ring algebra, where the addition is carried out using logical OR, i.e., $1+1=1$. One potential shortcoming of using OR-based Boolean arithmetic is that ORing two bases from $\mathbf{B}$ with a '1' in the same location will lead to a '1', and this result will not change regardless of any additional bases that can be further ORed with the two. In the case of field modulo-2 algebra, however, the addition is carried out using logical XOR, i.e., $1+1=0$. Thus a '1' can be reduced back to '0' and therefore combining additional bases in modulo-2 implementation can offer more diversity in the results. Figure 1 shows an example of an input matrix as well as the factorized matrices using both Boolean and Modulo-2 arithmetic. As demonstrated in the figure, using different arithmetic can result in significantly different characteristics in the factorized matrices. In the specific case of Figure 1, modulo-2 algebra generates better quality of results. In this paper we advocate and demonstrate the benefits of the expanded search space offered by a modulo-2 based implementation when compared to the Boolean-only approach as proposed in previous work [1].

Interestingly, modulo-2 approximate logic synthesis closely resembles that of the Boolean based approach, where the only differences are (1) a modulo-2 approach is utilized for the matrix factorization, and (2) the decompressor circuit needs to be mapped to network of XOR gates instead of OR gates. Currently there are no modulo-2 matrix factorization algorithms and the complexity of the problem is unknown [6]. Note that the Boolean counterpart is proven to be NP-Hard, and therefore all existing algorithms are based on heuristics [5]. To enable our methodology using modulo-2 arithmetic, we propose a simple heuristic based on the methodologies used for the Boolean matrix factorization. More specifically, we use Asso [5], [6], a factorization heuristic based on Boolean algebra, for initial matrix factorization, and then we do an exhaustive search for the decompressor matrix to minimize the error assuming modulo-2 arithmetic. Note that this operation incurs a timing complexity of $O(m2^f)$, where m denotes the number of outputs (or columns of the decompressor matrix), and f denotes the factorization degree (or the number of rows in the decompressor matrix), as different columns of the decompressor circuit can be identified independently.

Finally, as different columns of the decompressor matrix represent different combinations of the compressor circuits, one can mix the OR-based and XOR-based methodologies, where some outputs are implemented using OR and other outputs are implemented using XORs, i.e., the decompressor circuit uses both OR and XOR gates. We refer to this approach as XOR/OR, as it chooses the best outcome of OR versus XOR results to implement. We will evaluate the XOR and OR/XOR methodologies in the experimental results highlighting the benefit of each in different circumstances.

B. Truth Table Folding

The number of rows in the truth table of a circuit grows exponentially as a function of the number of circuit inputs. If $2^n >> m$, this results in a tall-and-skinny matrix, which can lead to a lack of common bases between different outputs. As a result, the binary matrix factorization algorithms can truncate the least significant outputs, especially when the outputs represent a numerical value, reducing the decompressor circuit to wires. While still providing valid approximations, such occurrences can limit the benefits in hardware metrics.

To mitigate this issue and reduce the discrepancy between the dimensions of the input matrix, we propose to *fold* a tall-and-skinny input matrix. Specifically, as an example, one can reduce the number of rows by half, by dividing the input matrix into two equal sub-matrices and concatenating the two sub-matrices in a column-wise fashion. Figure 2 demonstrates the input matrix folding process for a *folding degree* of four. Effectively, folding an input truth table with n inputs and m outputs (therefore a truth table of size 2^n rows and m columns) by a folding degree of k generates a truth table with $2^{n-\log_2(k)}$ rows and mk columns for a circuit with $n-\log_2(k)$ inputs and $m \times k$ outputs.

In hardware domain, folding affects both the compressor and decompressor circuits. The compressor circuit uses the least significant $n-\log_2(k)$ inputs, and the most significant $\log_2(k)$ bits are used as inputs to a multiplexer tree to select the correct output from the available mk outputs of the decompressor as illustrated in Figure 3. We refer to the decompressor circuit with the multiplexing tree as the *extended decompressor circuit*. Folding essentially shifts the circuit complexity from the compressor circuit to the decompressor circuit. At the

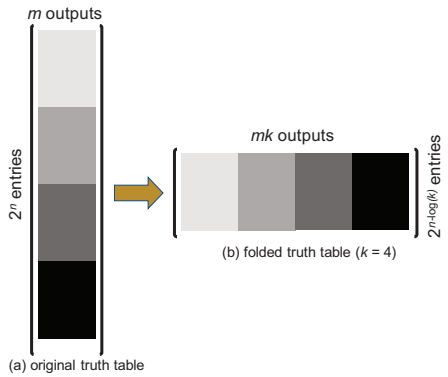

Fig. 2. The proposed truth table folding. (a) the input truth table, and (b) the 4-folded resulting truth table.

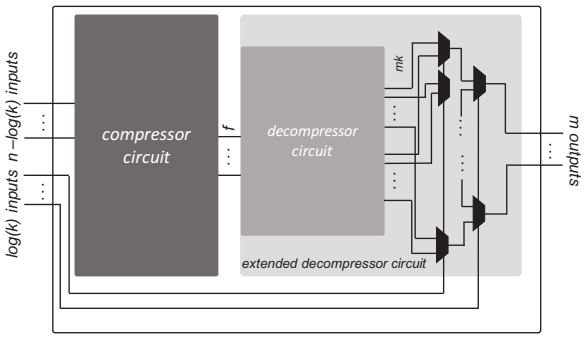

Fig. 3. Hardware realization of approximate circuit with truth table folding.

same time, folding increases the range of possible factorization degrees, f, by a factor of k potentially enabling a wider range of trade-offs. Finally, folding enables the binary matrix factorization algorithm to detect features not only among different output signals, but also among different segments of the same output. In Section III, we will demonstrate the benefits of matrix folding and demonstrate how the large design space enabled in this paper, can result in better trade-offs between accuracy and design metrics.

C. Design Space Exploration & Large Circuit Decomposition

Similar to our previous work [1], and since the truth table size of a circuit grows exponentially with the number of its inputs, we decompose any large circuit into sub-circuits, where each sub-circuit has a limited number of inputs (e.g., $n \leq 10$) and then approximate each sub-circuit individually using the proposed binary matrix decomposition method with mixed OR/XOR decompressor implementation and truth table folding. Furthermore, we utilize the same approach for design space exploration as BLASYS. After identifying the sub-circuits, we calculate the possible approximate realizations for each sub-circuit using various factorization degrees, OR/XOR implementations and folding degrees. We then greedily explore the space of generated approximate sub-circuits to identify a good approximation order. Due to space limits, we refer the readers to our previous work for more details [1].

III. Experimental Results

We evaluate the proposed generalized matrix factorization methodology, on eight different benchmarks, as summarized in Table I, where we report the accurate circuit characteristics. We evaluate two smaller benchmarks available in the LGSynth91 benchsuit, namely b1 and x2, two arithmetic circuits, as well as four larger circuits commonly used in DSP applications. For smaller circuit, b1 and x2, we generate the truth table and pass them to the factorization algorithm. For the larger benchmarks, however, we first decompose the circuit as described in Subsection II-C. For hardware metrics, all designs are implemented in Verilog and synthesized using Synopsys design compiler using an industrial 65 nm technology node at the typical processing corner.

TABLE I
The list of benchmarks evaluated using the proposed methodology.

Name	Function	I/O	Accurate Design Metrics		
			Area (um^2)	Power (uW)	Delay (ns)
b1	general logic	3/4	9	0.8	0.08
x2	general logic	10/7	41.04	3.3	0.18
Adder32	32-bit adder	64/33	320.8	81.1	3.23
Mult8	8-bit multiplier	16/16	1731.6	263.5	2.03
BUT	butterfly structure	16/18	297.4	80.6	1.79
MAC	multiply and accumulate with 32-bit accumulator	48/33	6013.1	470.5	2.36
SAD	sum of absolute difference	48/33	1446.5	195.1	2.43
FIR	4-tap FIR filter	64/16	8568.0	466.3	1.56

For design accuracy, we report the normalized Hamming distance (HD), which is defined as

$$\text{Normalized HD} = \frac{|\mathbf{A} - \mathbf{BC}|}{Nm}, \tag{2}$$

and average relative error (AVE) defined as

$$\text{AVE} = \frac{1}{N}\Sigma_{i=1}^{N}\frac{|R_i - R_i'|}{R_i}, \tag{3}$$

for logical and binary numerical outputs, respectively. Here, N represents the size of the test data while R_i and R_i', represent the accurate and approximate numerical results. Furthermore, for smaller circuits, we define the accuracy over all possible inputs, while for larger networks, we utilize a randomly generated $N = 100,000$ instance test set. Finally, in larger circuits, we report the total hardware cost as the summation of the smaller subcircuits.

We first briefly present the possible benefits of the expanded design space. For this study we focus on the smaller benchmarks, as they eliminate the need for an exploration methodology and therefore better demonstrate the isolated impact of the proposed XOR-based methodology and the proposed folding approach. Figure 4 shows the design space offered by the proposed methodology. Here, we plot the accuracy versus the design area utilization for the x2 benchmark. As shown in the figure, approximate logic synthesis using binary matrix factorization offers a wide and smooth range of trade-offs between design metrics and accuracy, where using OR/XOR algebra and truth table folding further expand the possible trade-offs. In the case of this benchmark, the new design points enable new optimal trade-offs with significant improvement in area savings. As shown in the figure, the design points enabled by this paper completely dominate the OR-based approach.

TABLE II
THE DESIGN SPACE EXPLORATION RESULTS FOR OUR EIGHT BENCHMARKS. HD STANDS FOR HAMMING DISTANCE. NOTE THAT FOR 5% AND 10% ACCURACY THRESHOLDS SMALLER CIRCUITS (B1, AND X2) ARE EXCLUDED FROM THE AVERAGE VALUES.

Benchmark	Error Metric	5% Accuracy Threshold			10% Accuracy Threshold			25% Accuracy Threshold		
		OR [1]	OR/XOR	Folding	OR [1]	OR/XOR	Folding	OR [1]	OR/XOR	Folding
b1	normalized HD	NA	NA	NA	36.0%	36.0%	36.0%	36.0%	36.0%	36.0%
x2	normalized HD	NA	NA	NA	NA	NA	12.3%	43.0%	48.3%	86.4%
Adder32	avg. rel. error	48.1%	48.1%	48.1%	50.6%	50.6%	50.6%	52.2%	52.2	52.2%
Mult8	avg. rel. error	21.3%	24.4%	29.0%	27.5%	38.2%	41.7%	57.4%	60.6%	65.6%
BUT	avg rel. error	7.9%	7.9%	7.9%	24.7%	24.7%	24.7%	31.9%	31.9%	31.9%
MAC	avg rel. error	41.4%	45.2%	45.2%	54.0%	56.7%	56.8%	60.5%	63.6%	63.6%
SAD	avg rel. error	30.2%	30.2%	30.2%	33.2%	33.2%	33.2%	33.2%	33.2%	33.2%
FIR	avg rel. error	13.6%	19.9%	20.1%	18.6%	23.6%	23.6%	29.0%	32.3%	35.8%
Average area savings		27.9%	29.3%	30.1%	34.8%	37.8%	38.4%	42.9%	44.8%	50.6%

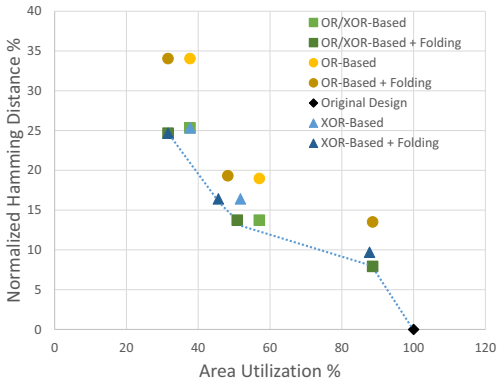

Fig. 4. The trade-offs offered by XOR and OR/XOR algebras as well as truth table folding for x2 benchmark.

Table II summarizes the savings offered by the proposed methodology for three different accuracy thresholds, where we compare our expanded approach with OR/XOR factorization, and truth table folding versus BLASYS [1], which is an OR-based implementation. Here, we report the best total design area savings, computed as the sum of the smaller subcircuits, under the specified accuracy threshold. Note that all approximate variant of benchmarks b1 and x2, due to their smaller size, fail to meet the tighter 5% accuracy threshold (also for x2 in the case of OR-based and OR/XOR-based methods for 10%) and therefore do not offer any benefits. As evident from the table, using different algebra results in benefits in hardware metrics for many of the benchmarks. As an example, in the case of the mult8 benchmark circuit, and for an accuracy threshold of 25%, enabling the OR/XOR implementations, introduce an additional 3.2% savings in total area. On the other hand for some benchmarks, such as b1 and adder32, the new design space does not provide any improvements, since the OR-based Boolean implementation provides the best design points. Therefore, as previously discussed, the introduced methodologies prominently expand the design space exploration and therefore should be considered alongside the Boolean design points.

Furthermore, by enabling truth table folding, additional savings are achieved in many of the applications. Specifically, extra savings of 5% can be offered in the case of mult8 benchmark. Similar to enabling different algebra, however, no benefits are guaranteed and therefore a design space exploration considering all the data points is necessary. Compared to BLASYS [1], methodologies proposed, on average introduce additional area savings of 2.2%, 5.1%, and 7.7% for 5%, 10%, and 25% accuracy thresholds, respectively. Finally, in power and delay, and for the case of a 5% accuracy threshold, our proposed methodology results in average savings of 37.65% and 25.46% across the benchmarks, respectively.

IV. CONCLUSIONS

In this paper we expand our work in approximate circuit synthesis by generalizing matrix factorization techniques to incorporate field (XOR) and semi-ring (OR) algebra implementations. This led to a wider range of possible approximate circuit realizations that can be explored to identify the best trade-offs. We also proposed a truth table folding technique that shifts circuit complexity between the compressor and decompressor circuits, enabling a wider range of possible circuit approximations. We implemented and evaluated our approach on a large range of circuits using a number of error metrics such as numerical differences and Hamming distances, and we have demonstrated that the expanded design space improves upon our previous approach by 3-12% depending on the target accuracy threshold.

Acknowledgments: This work is partially supported by NSF grant #1814920.

REFERENCES

[1] S. Hashemi et al. BLASYS: approximate logic synthesis using boolean matrix factorization. In *DAC*, pages 55:1–6, 2018.
[2] S. Lee et al. High-level synthesis of approximate hardware under joint precision and voltage scaling. In *DATE*, 2017.
[3] C. Li et al. Joint precision optimization and high level synthesis for approximate computing. In *DAC*, pages 104:1–6, 2015.
[4] Jin Miao et al. Approximate logic synthesis under general error magnitude and frequency constraints. In *ICCAD*, pages 779–786, 2013.
[5] P. Miettinen et al. Model order selection for boolean matrix factorization. In *international conference on Knowledge discovery and data mining*, pages 51–59, 2011.
[6] P. Miettinen et al. Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Transactions on Knowledge Discovery from Data*, 8(4):18:1–31, 2014.
[7] K. Nepal et al. Automated high-level generation of low-power approximate computing circuits. *IEEE TETC*, pages 1–13, 2016.
[8] Ashish Ranjan et al. Aslan: Synthesis of approximate sequential circuits. In *DATE*, pages 1–6, 2014.
[9] M. Shafique et al. Cross-layer approximate computing: from logic to architectures. In *DAC*, pages 99:1–6, 2016.
[10] S. Venkataramani et al. Salsa: Systematic logic synthesis of approximate circuits. In *DAC*, pages 796–801, 2012.
[11] Swagath Venkataramani et al. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *DATE*, pages 1367–1372, 2013.