

Chapter 7

Approximate Logic Synthesis Using Boolean Matrix Factorization



Soheil Hashemi, Hokchhay Tann, and Sherief Reda

7.1 Introduction

Synthesizing an approximate circuit from an arbitrary circuit, presumably accurate, is one of the key challenges in approximate computing. While recently many methodologies have proposed targeting approximate arithmetic building blocks (e.g., approximate adders, multipliers, and dividers) [2–5, 9], it is harder to generate approximate circuits for arbitrary circuits. Synthesis techniques in the literature operate either on gate-level [8, 11, 12, 16, 18, 19], RTL, and behavioral hardware descriptions [15], or high-level software descriptions [7].

This chapter describes our method for approximate logic synthesis using Boolean matrix factorization [17]. Non-negative matrix factorization (NNMF) factors an input matrix into two smaller matrices, such that in all three matrices all elements are non-negative [6]. The original input matrix is then approximated by matrix multiplication of the two factorized matrices. Furthermore, the non-negativity is an inherent constraint in many physical domains, such as data clustering and computer vision [21]. Recently, with advances in the applied mathematics, NNMF has been extended to Boolean matrices, where operations are performed using Boolean mathematics ($GF(2)$), such that multiplications are performed using logical AND, and additions are performed using logical OR (for Boolean semi-ring implementations) and logical XOR (for Boolean field implementations) [13, 14]. In our approach we first build the truth table of a given multi-input, multi-output logic circuit and then approximate its truth table using BMF. The approximate truth table is then synthesized into an approximate circuit.

S. Hashemi · H. Tann · S. Reda (✉)
Brown University, Providence, RI, USA
e-mail: soheil_hashemi@brown.edu; hokchhay_tann@brown.edu; sherief_reda@brown.edu

In order to scale our proposed BMF-based methodology to larger circuits, we introduce a decomposition methodology, effectively breaking down a large circuit to smaller subcircuits with reduced number of inputs and outputs. Furthermore, to efficiently explore the design space, we propose an algorithm to identify an optimal sequence of approximations for different subcircuits. In addition, to better account for binary representations, rather than using the standard L_2 norm, we introduce modified quality-of-results (QoR) functions. Our proposed methodology offers a wide range of smooth trade-offs between accuracy and design metrics. In order to demonstrate the trade-offs offered by our approach, we implement and test our methodology on a wide range of error tolerant applications. Our results highlight the benefits achieved by our methodology while introducing insignificant errors in the output.

The organization of this chapter is as follows. We discuss the details of our proposed method in Sect. 7.2. In this section, we discuss our idea for using BMF for approximate logic synthesis, as well as our methodology for approximating larger circuits. We report the results of our method’s performance and the trade-offs offered in Sect. 7.3. Finally, Sect. 7.4 summarizes our conclusions as well as possible directions for future expansion.

7.2 BMF-Based Approximation for Arbitrary Circuits

BMF is a factorization technique where a $k \times m$ matrix \mathbf{M} is factored into two Boolean: a $k \times f$ matrix \mathbf{B} and an $f \times m$ matrix \mathbf{C} , such that $\mathbf{M} \approx \mathbf{BC}$, where matrix operations are carried in Galois field of two elements, $GF(2)$. In such domain, multiplications can be performed using logical AND, while additions are performed using logical OR (for Boolean semi-ring implementations) and logical XOR (for Boolean field implementations) [13, 14]. BMF algorithms, essentially, compress the data representation in an approximate fashion and depending on the degree of factorization [21]. In the mathematical statistical community, the *factorization degree*, f , determines the number of “features” that are computed [13]. The factorization degree, f , represents a trade-off between quality of factorization and storage amount.

Figure 7.1 provides an example of a 3-input, 4-output arbitrary logic circuit. First, we construct the truth table of the original circuit. We also synthesize the circuit. Here we use 65 nm libraries and Synopsys DC as the logic synthesis tool, which gives an original circuit area of $22.3 \mu\text{m}^2$. The truth table is provided as input to the BMF factorization algorithm, along with the desired factorization degree. Here we choose a factorization degree $f = 2$. We then factor the matrix using ASSO the BMF algorithm [13, 14] into two matrices 16×2 and 2×4 . Thus, the approximate truth table has a Hamming distance of 6 from the original truth table, i.e., QoR changes only by 9.3%. The next step is to synthesize the two truth tables. The 16×2 truth table of the compressor circuit is used to create a 16×2 sum-of-product circuit with 4 inputs and 2 outputs using the established logic synthesis techniques [20].

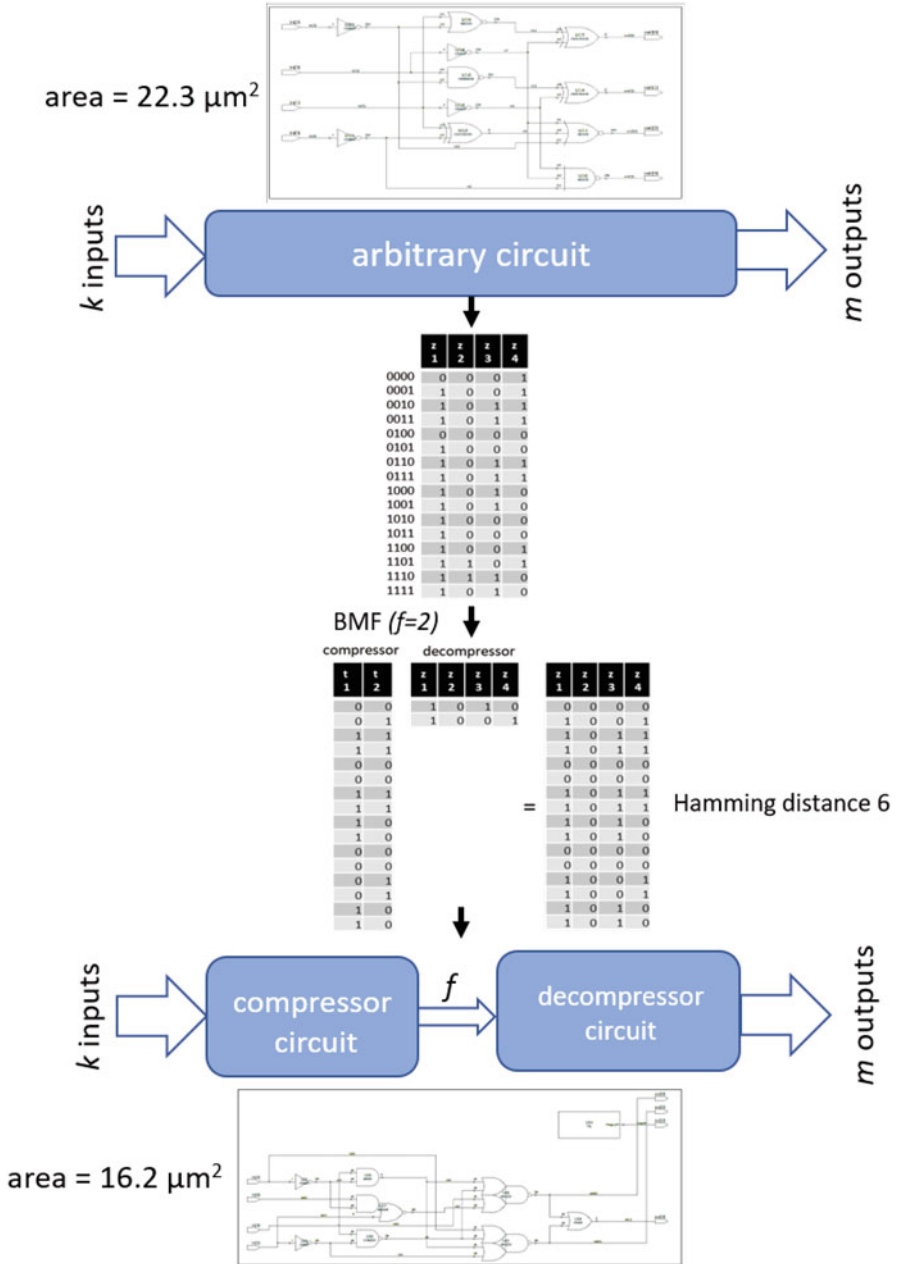


Fig. 7.1 An example of BMF-based approximations

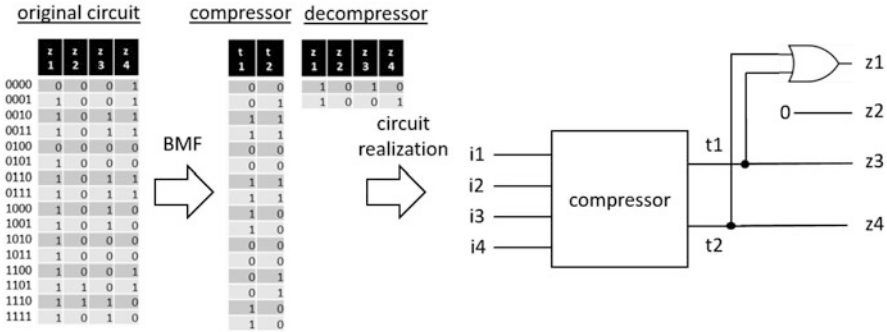


Fig. 7.2 The OR-based decompressor circuit for the example of Fig. 7.1

The 2×4 table for the decompressor is then used to synthesize a second circuit that ORes the outputs of the SOP circuit to produce the four outputs as illustrated in Fig. 7.2. The two circuits are then given to the logic synthesis tool to produce one final circuit from both of them. In this particular example, the synthesis tool generates an approximate circuit with area of $16.2 \mu\text{m}^2$. That is, we can reduce the area by 27.3%, while compromising the QoR by only 9.3%.

As demonstrated in the figure, our methodology provides a synthesis method that enables controlled trade-offs between accuracy and design costs based on the degree of approximation. Figure 7.3 illustrates our example with different factorization degrees, therefore resulting in different trade-offs. In this example and for $f = 3$, $f = 2$, and $f = 1$, our methodology results in approximate circuits with Hamming distance of 3, 6, and 13, respectively, and circuit areas of 19.1 , 16.2 , and $9.4 \mu\text{m}^2$, respectively. Figure 7.3 illustrates our example with factorization degrees $f = 3$, $f = 2$, and $f = 1$, which lead to approximate circuits of Hamming distance of 3, 6, and 13 respectively, with circuit areas of 19.1 , 16.2 , and $9.4 \mu\text{m}^2$.

There are two main challenges:

1. In typical NNMF application the index or location of the error does not bear any significance since NNMF algorithms utilize the L_2 norm as a metric for the quality of factorization. In the case of Boolean matrices, L_2 norm translates to Hamming distance. Since the binary outputs of many circuits have corresponding numerical association (e.g., 8-bit number), we need a metric for QoR with knowledge of output representation. Such QoR metrics are crucial in approximate hardware design.
2. As our technique introduces approximations by operating on circuit truth table, the scalability can be limited as the size of the truth table increases exponentially as a function of the number of inputs. Therefore, we need methodologies to enable the scaling of our method to larger circuits.

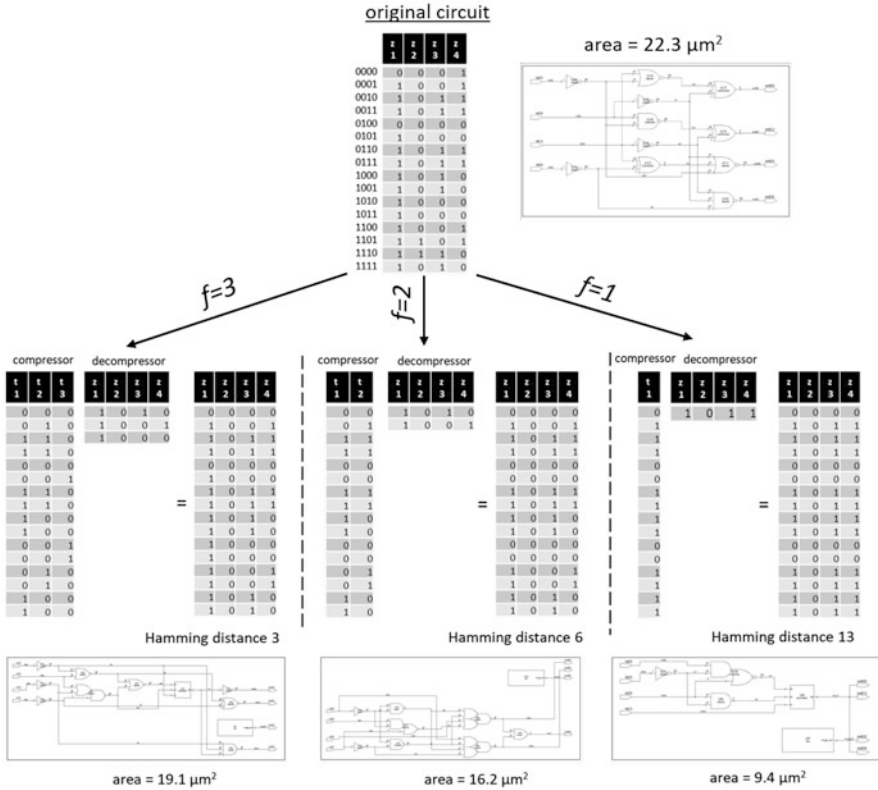


Fig. 7.3 Impact of factorization on accuracy and area

7.2.1 Circuit Factorization for Arbitrary QoR Metrics

In BMF algorithms, the objective is to minimize $\|\mathbf{M} - \mathbf{BC}\|_2$, which translates to Hamming distance in Boolean systems. In approximate circuit design, however, such metric does not provide a good representation of QoR in many cases. As an example, if the columns of an m -column matrix represent an m bit signal, minimizing the Hamming distance as the cost function can lead to significant errors in numerical value. For instance, a bit flip in the least significant bit will lead to a numerical error of 1, whereas a bit flip in the n th bit leads to an error of 2^{n-1} .

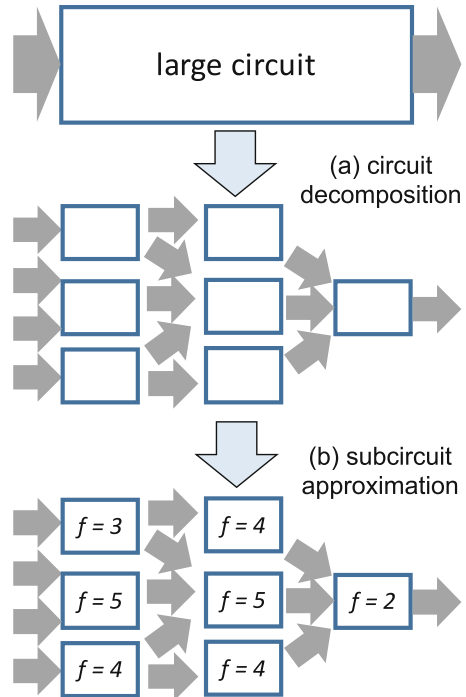
To account for the bit significance, we augment existing BMF algorithms with custom QoRs enabling weighted cost functions. Specifically, we propose to define the cost function as $\|(\mathbf{M} - \mathbf{BC})\mathbf{w}\|_2$, where \mathbf{w} is a constant weight vector, instead of $\|\mathbf{M} - \mathbf{BC}\|_2$ as the standard Hamming distance cost function. Here, if the numerical difference is the objective QoR, then \mathbf{w} will be defined to introduce bit significances based on powers-of-two (e.g., 8, 4, 2, 1), therefore, giving different numerical weights for different bit positions. In our experiments, we modify the

ASSO [14] algorithm as to penalize mismatches on higher bit indices more than lower significant bits. We will provide experimental results showcasing the benefits of such weighting schemes in contrast to uniform weights (Hamming distance) in Chap. 7.3.1.

7.2.2 *Scaling Up for Large Circuits*

As our methodology operates on the truth table of the input circuit, the size of the input matrix, i.e., the number of rows, grows exponentially as the number of primary inputs increases. Furthermore, BMF is a NP-hard problem, and the existing methodologies are based on heuristics [6, 13, 14]. Therefore, the applicability of our method can be limited as the complexity of the circuit increases. Therefore, we propose a circuit decomposition technique to scale the BMF algorithm for larger circuits. The overall idea of our method is to first decompose a large circuit into a number of subcircuits, such that each subcircuit has a maximum of k inputs and m outputs as illustrated in Fig. 7.4a and then each of the subcircuits is approximated as shown in Fig. 7.4b. The values for k and m are determined based on the afforded runtime of the factorization algorithm. While this approach closely resembles FPGA mapping algorithms, it is different from such algorithms as (1) such breakdowns

Fig. 7.4 Illustrated methodology for decomposing circuits



are performed purely to reduce the computational complexity, (2) the resulting approximate circuit can be synthesized using any target FPGA or ASIC technology, and (3) in contrast to FPGA mapping, the number of outputs can be larger than one. Therefore, instead of using classical k -cut algorithms, e.g., [1], we propose to use $k \times m$ -cut algorithms (e.g., KL algorithm [10]) to break down a large circuit into subcircuits with a maximum of k inputs and maximum of m outputs.

Dividing a large circuit into smaller subcircuits of size $k \times m$ requires a change to the way we compute the QoR. More specifically, we can no longer evaluate the accuracy of a subcircuit in isolation, as errors in one component can propagate through the circuit leading to larger errors in the final outputs. Therefore, in our work instead of evaluating the QoR of a subcircuit individually, we evaluate the QoR of the entire approximate circuit, denoted by $Cir(s_i \rightarrow T_{s_i, f_i})$, where an accurate subcircuit, s_i , is substituted by its approximate version, T_{s_i, f_i} , with a factorization degree of f_i .

Since a large input circuit will have multiple subcircuits, the order and the degree to which the approximations are introduced to the circuit has to be carefully analyzed. We devise Algorithm 1 to gradually approximate the circuit. In our algorithm, first, the circuit is broken down into smaller subcircuits (Line 1). In the next stage (lines 3–9) and for each subcircuit, the set of *potential* approximate versions under various approximation degrees are profiled. Next, and starting from the accurate design, approximations are gradually added to the input design by exploring the neighbors of the current design (lines 14–22). Here, neighbors of a given subcircuit are defined as subcircuits for which the degrees of approximation only differ in one subcircuit. Here in lines 15–18, for each neighbor, first the degradations in overall QoR metrics are assessed. The subcircuit with the least degradation in QoR is then chosen to replace the current circuit for next iteration in lines 19–21. The process is repeated iteratively until the QoR gets lower than a predefined threshold.

7.3 Experimental Results

In this section we demonstrate, experimentally, the benefits achievable from using our BMF-based logic approximation (BLASYS) methodology. For our experiments we explore a broad range of applications ranging from arithmetic circuits (adder and multiplier) to more complicated datapaths. We report our results for circuits amenable for approximations, including a multiply-accumulate circuit (MAC), a butterfly network (BUT), a sum of absolute differences (SAD) circuit, and finite impulse response (FIR) circuit [16, 18]. In Table 7.1, we summarize the hardware characteristics of the evaluated accurate circuits, including the number of inputs and outputs, design area, power consumption, and critical path delay. To evaluate the design metrics, we used Synopsys design compiler with an industrial 65 nm technology library in typical processing corner.

Algorithm 1: BLASYS: Boolean level approximate circuit synthesis

Input : Accurate Circuit $ACir$, Error Threshold
Output: Approximate Circuit Cir

- 1 subcircuits=Decompose input circuit using $k \times m$ decomposition
- 2 // Factorization profiling Phase
- 3 **for** each subcircuit s_i with $m_i \leq m$ outputs **do**
- 4 **M**=Construct truth table of s_i
- 5 // profile for every possible factorization degree
- 6 **for** $f=1$ to m_i-1 **do**
- 7 **[B, C]** = BMF(**M**, f)
- 8 $T_{s_i, f}$ =Construct truth table of **BC**
- 9 **end**
- 10 **end**
- 11 // Circuit Space Exploration Phase
- 12 $Cir=ACir$;
- 13 Let $f_i = m_i$ for all subcircuits s_i
- 14 **while** $QoR(Cir) < threshold$ **do**
- 15 **for** each subcircuit s_i with $f_i > 1$ **do**
- 16 $Cir'=Cir(s_i \rightarrow T_{s_i, f_i-1})$
- 17 $\Delta err_i = QoR(Cir') - QoR(Cir)$
- 18 **end**
- 19 $b = \arg \min_i (\Delta err_i)$
- 20 $Cir = Cir(s_b \rightarrow T_{s_b, f_b-1})$
- 21 $f_b = f_b - 1$
- 22 **end**
- 23 Cir =Synthesize Best new Design
- 24 **return** Cir

Table 7.1 The list of benchmarks evaluated using the proposed NNMF methodology

Name	Function	I/O	Accurate design metrics		
			Area (um ²)	Power (uW)	Delay (ns)
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 all our experiments, we first decompose each circuit to a maximum of 10×10 (i.e., $k = 10$ and $m = 10$) subcircuits and then perform factorization. We use the ASSO algorithm for BMF [13, 14]. In addition, to evaluate the accuracy of the approximate applications, we utilize Monte Carlo simulations with one million randomly generated input test cases. In our results, the average relative error is defined as

$$\text{Average Relative Error} = \frac{1}{N} \sum_{i=1}^N \frac{|R_i - R'_i|}{R_i}, \quad (7.1)$$

and average absolute error as

$$\text{Average Absolute Error} = \frac{1}{N} \sum_{i=1}^N |R_i - R'_i|, \quad (7.2)$$

where N denotes the size of the test case, and R and R' denote accurate and approximate results, respectively.

In the next subsection we report our results highlighting the impact of arbitrary QoR functions in contrast to standard, uniformly weighted L_2 metric used in Boolean matrix factorization. Then, and in the second subsection, we summarize the benefits achieved when BLASYS utilized within our applications. Here, we also compare our work against state of the art.

7.3.1 Impact of QoR Metric

As previously described, we utilize a BMF algorithm, ASSO in this work, with a custom weighted cost function. Specifically, as the matrix columns represent the outputs of the circuit, in many cases a bit disparity on higher significance bits should generate higher penalties compared to errors on lower indices.

Figure 7.5 shows the accuracy versus design complexity trade-offs offered for the approximate `Mult8` design when comparing an off-the-shelf factorization algorithm using standard L_2 QoR with uniform bit weighting against the proposed weighted QoR. The trends in average relative error, normalized average absolute error, and the normalized Hamming distance are provided. In this plot, the solid lines denote the weighted QoR (WQoR) results, and the results for the original uniform QoR (UQoR) are shown with dashed lines.

As shown in the figure, for same circuit complexity, the proposed weighted QoR offers consistent improvements in all three accuracy metrics when compared against the uniform QoR. Therefore, for binary representations, significant benefits can be achieved by modifying the BMF algorithm to utilize a weighted QoR. In the next subsection, trade-offs offered by BLASYS are evaluated for all of our application circuits and using our heuristic design space exploration. We report the benefits in design area, power consumption, and critical path delay offered by our methodology.

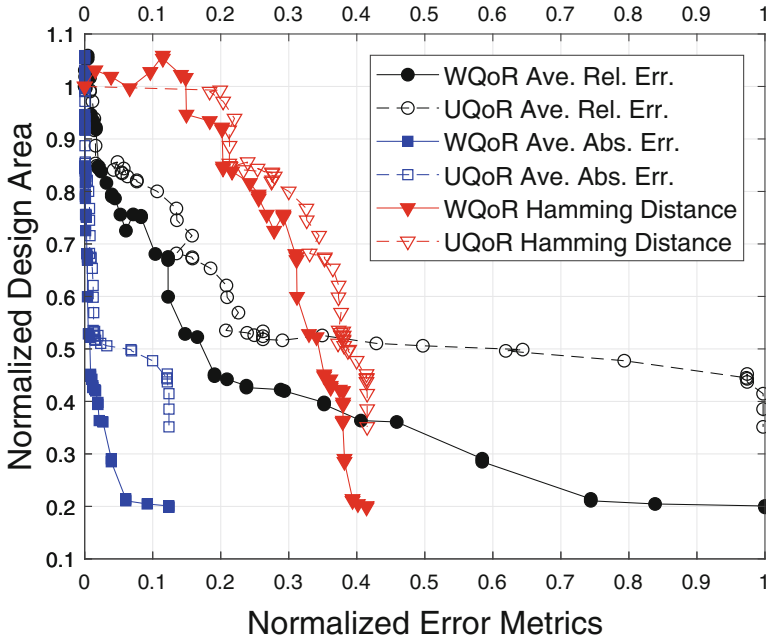


Fig. 7.5 Comparison of the trade-offs offered using the proposed weighted QoR vs. the original factorization algorithm

7.3.2 Circuit Results

As described in Sect. 7.2, in order to tackle larger circuits, our technique first decomposes a circuit into smaller subcircuits with lower number of input and output ports. For all subcircuits, the possible approximate subcircuits, using different factorization degrees, are then profiled. Next, the heuristic proposed in Algorithm 1 iteratively approximates the subcircuits while assessing the impact on the QoR of the whole circuit through simulation. For design area and power, these metrics for the whole circuit are defined as the sum of design metrics of the $k \times m$ -cut subcircuits. In our experiments we chose design area as our design metric, as the summation of the subcircuits design area values better represents the total area, while the power consumption shows more variations. In addition, note that for our design space exploration, our design area model only represents the portion of the subcircuits blocks being approximated, while register files and controls paths act as an area bias and they do not affect the relative gradients.

Figure 7.6 shows the trade-offs offered by our method, BLASYS, for our six benchmarks. In this figure we plot the normalized design metric as a function of average relative error, shown in black and using the bottom x axis, and average normalized absolute error, shown in red and using the top x axis. In the case of average absolute error, as the range can differ significantly based on output bit-

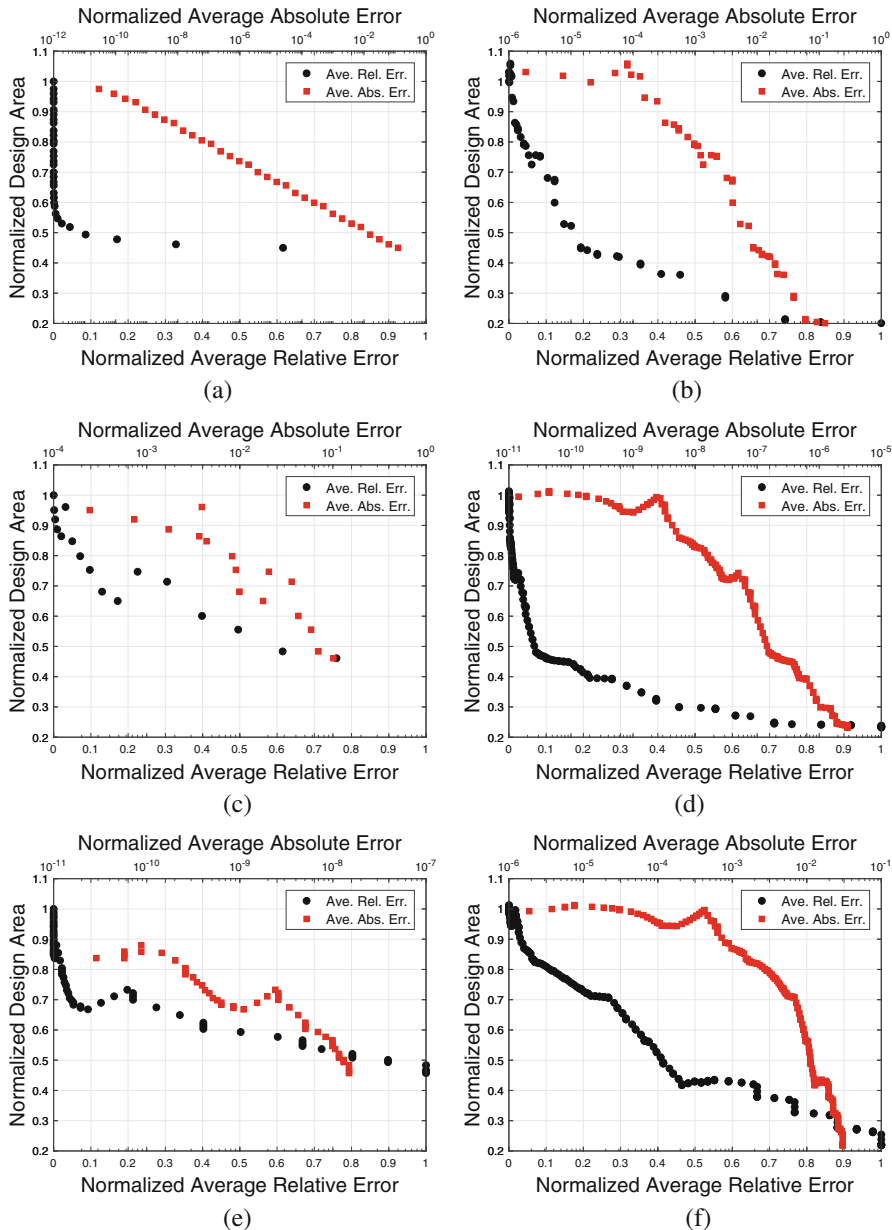


Fig. 7.6 The trade-offs offered for each application. (a) Adder32, (b) Mult8, (c) BUT, (d) MAC, (e) SAD, and (f) FIR

width, we normalize the values to the highest possible output to better illustrate its significance. Note that the average absolute error is in log scale to better demonstrate the trend.

As evident from the figure, using our methodology, in all applications a wide range of fine-grains trade-offs are available. Intuitively, at each iteration, the proposed design space exploration heuristic tries to find the optimum design point, effectively the design with minimum design area, for a given degree of approximation. The degree of approximation is then incremented by one in each iteration. This insight explains the smooth trade-offs for larger circuits, whereas the characteristics of smaller circuits can change significantly in one iteration. Furthermore, in some cases, we observed an increase in the area utilization as more approximations are introduced. This phenomenon is explained by an overall increase in the total number of literals in the intermediate signals. In other words, while BLASYS compresses the outputs into f signals, these f signals can have higher complexity compared to their accurate counterparts. Our design space exploration methodology, however, can navigate through these temporary increases to find a minimal design point offering significant savings in design metrics. Based on the circuit, benefits of approximately 25–60% can be achieved for average relative errors of 20%.

Utilizing the design space exploration methodology, the results of our six evaluated applications, and for two accuracy thresholds of 5% and 25% are summarized in Table 7.2. For both error thresholds, our methodology delivers significant benefits in all design metrics, as demonstrated in the table. In our experiments and for a tight threshold of 5%, benefits of approximately 8–47% can be achieved in the case of design area, while the same benefits for a threshold of 25% range from 26 to 65%.

Table 7.2 The hardware characteristics of the approximate test cases for two accuracy thresholds, namely 5% and 25%

Design	Accuracy threshold (%)	Area savings (%)	Power savings (%)	Delay reduction (%)
Adder32	5	44.78	63.79	12.07
	25	48.15	69.35	17.03
Mult8	5	28.77	26.87	12.32
	25	63.18	68.93	41.38
BUT	5	7.87	11.25	2.23
	25	26.39	35.14	5.03
MAC	5	47.55	55.58	64.41
	25	65.86	75.13	69.07
SAD	5	32.80	41.47	69.14
	25	38.08	55.07	77.37
FIR	5	19.52	22.26	12.18
	25	34.00	33.84	16.03

7.4 Conclusion

In this chapter we described a new direction for approximate logic synthesis using Boolean matrix factorization. Our proposed methodology leads to a systematic approach to trade-off accuracy with circuit complexity. To scale our approach to handle large circuits, we propose a circuit decomposition method that breaks down a large circuit into smaller circuits that are amenable for BMF. Our algorithm leads to a smooth trade-off between the complexity of entire large circuits with accuracy. We also described how to incorporate different QoR metrics into the circuit factorization. Our experimental results show robust trade-off between accuracy and circuit area. Our technique is developed into BLASYS, which is a fork of the popular logic synthesis tool YOSYS, and it is available to download at <http://github.com/scale-lab/BLASYS>. Our approach opens many doors for investigation in logic synthesis, including improved techniques for BMF and improved $k \times m$ circuit decomposition.

References

1. Cong J, Ding Y (1994) Flowmap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Trans CAD Integr Circuits Syst* 13:1–12
2. Hashemi S, Bahar RI, Reda S (2015) Drum: a dynamic range unbiased multiplier for approximate applications. In: *Proceedings of the IEEE/ACM international conference on computer-aided design, ICCAD '15*. IEEE Press, Piscataway, pp 418–425
3. Hu J, Qian W (2015) A new approximate adder with low relative error and correct sign calculation. In: *2015 design, automation test in Europe conference exhibition (DATE)*, pp 1449–1454
4. Imani M, Peroni D, Rosing T (2017) CFPU: configurable floating point multiplier for energy-efficient computing. In: *Proceedings of the 54th annual design automation conference 2017, DAC '17*. ACM, New York, pp 76:1–76:6.
5. Kahng AB, Kang S (2012) Accuracy-configurable adder for approximate arithmetic designs. In: *DAC design automation conference 2012*, pp 820–825
6. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788–791
7. Lee S, John LK, Gerstlauer A (2017) High-level synthesis of approximate hardware under joint precision and voltage scaling. In: *Design, automation and test in Europe*
8. Li C, Luo W, Sapatnekar SS, Hu J (2015) Joint precision optimization and high level synthesis for approximate computing. In: *Design automation conference*, pp 104:1–104:6
9. Liu W, Qian L, Wang C, Jiang H, Han J, Lombardi F (2017) Design of approximate radix-4 booth multipliers for error-tolerant computing. *IEEE Trans Comput* 66(8):1435–1441
10. Martinello O, Ribas RP, Marque F, Reis A (2010) KI-cuts: a new approach for logic synthesis targeting multiple output blocks. In: *Design automation test in Europe*, pp 777–782
11. Miao J, Gerstlauer A, Orshansky M (2013) Approximate logic synthesis under general error magnitude and frequency constraints. In: *Proceedings of the international conference on computer-aided design*, pp 779–786
12. Miao J, Gerstlauer A, Orshansky M (2014) Multi-level approximate logic synthesis under general error constraints. In: *International conference on computer-aided design*, pp 504–510

13. Miettinen P, Vreeken J (2011) Model order selection for boolean matrix factorization. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 51–59
14. Miettinen P, Vreeken J (2014) MDL4BMF: minimum description length for boolean matrix factorization. *ACM Trans Knowl Discov Data* 8(4):18:1–18:31
15. Nepal K, Li Y, Bahar RI, Reda S (2014) ABACUS: a technique for automated behavioral synthesis of approximate computing circuits. In: Design, automation and test in Europe, pp 1–6
16. Ranjan A, Raha A, Venkataramani S, Roy K, Raghunathan A (2014) ASLAN: synthesis of approximate sequential circuits. In: Design, automation & test in Europe conference, pp 1–6
17. Hashemi S, Tann H, Reda S, (2018) BLASYS: approximate logic synthesis using boolean matrix factorization. In: Design automation conference, pp 1–6
18. Venkataramani S, Sabne A, Kozhikkottu V, Roy K, Raghunathan A (2012) Salsa: systematic logic synthesis of approximate circuits. In: DAC design automation conference 2012, pp 796–801
19. Venkataramani S, Roy K, Raghunathan A (2013) Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits. In: Design, automation and test in Europe, pp 1367–1372
20. Wolf C Yosys open synthesis suit. <http://www.clifford.at/yosys/>
21. Xu W, Liu X, Gong Y (2003) Document clustering based on non-negative matrix factorization. In: ACM SIGIR conference on research and development in information retrieval, pp 267–273