ASLAN: Synthesis of Approximate Sequential Circuits

Ashish Ranjan, Arnab Raha, Swagath Venkataramani, Kaushik Roy and Anand Raghunathan
School of Electrical and Computer Engineering, Purdue University
{aranjan,araha,venkata0,kaushik,raghunathan}@purdue.edu

***Abstract*—Many applications produce acceptable results when their underlying computations are executed in an approximate manner. For such applications, approximate circuits enable hardware implementations that exhibit improved efficiency for a given quality. Previous efforts have largely focused on the design of approximate combinational logic blocks such as adders and multipliers. In practice, however, designers are concerned with the quality of outputs generated by a sequential circuit after several cycles of computation, rather than an embedded combinational block.**

We propose ASLAN (*A*utomatic methodology for *S*equential *L*ogic *A*pproximatio*N*), the first effort towards the synthesis of approximate sequential circuits. Given a sequential circuit and an output quality constraint, ASLAN creates an approximate version of the circuit that consumes lower energy, while meeting the specified quality bound. The key challenges in approximating sequential circuits are (i) to model how errors due to approximations are generated, re-circulate through the combinational logic over multiple cycles of operation, and eventually impact quality of the final output, and (ii) to select the most beneficial approximations, *i.e.*, those that result in higher energy savings for smaller impact on quality. ASLAN addresses the first challenge by constructing a virtual Sequential Quality Constraint Circuit (SQCC) and utilizing formal verification techniques to ensure that the selected approximations meet the quality constraint. To address the second challenge, ASLAN identifies combinational blocks in the sequential circuit that are amenable to approximation, generates local quality-energy trade-off curves for them, and uses a gradient-descent approach to iteratively approximate the entire sequential circuit.

We used ASLAN to automatically synthesize approximate versions of ten sequential benchmarks, resulting in energy reductions of 1.20X-2.44X for tight quality constraints, and 1.32X-4.42X for moderate quality constraints. We present case studies of using the approximate circuits generated by ASLAN in two popular applications — MPEG Encoding and K-Means Clustering — obtaining 1.32X energy savings with 0.5% PSNR degradation, and 1.26X energy savings with 0.8% increase in mean cluster radius, respectively.

***Index Terms*—Low Power Design, Approximate Computing, Approximate Circuits, Logic Synthesis, Sequential circuits**

I. Introduction

Applications from several important domains, including recognition, mining, synthesis, multimedia, graphics *etc.*, demonstrate an ability to tolerate inexactness or *approximations* in their underlying computations and produce results of acceptable output quality. This intrinsic resilience stems from several factors, including redundant and noisy input data, the use of statistical and iterative computation patterns, a lack of golden results, or the inability of users to perceive minor differences in the output [1], [2]. Approximate computing is an emerging approach that leverages the intrinsic resilience of applications to improve the energy consumption and performance of computing systems that execute them.

A common approach to approximate computing involves the design of *approximate circuits*, which employ hardware techniques that trade off accuracy for efficiency. These techniques can be classified into: (i) timing approximation, where timing errors are introduced in the circuit by over-scaling the supply voltage [3]–[5], and (ii) functional approximation, where the logic function realized by the circuit is slightly modified so as to lead to a more efficient implementation [6]–[11].

Most research efforts on approximate circuits have focused on manual designs of simple arithmetic circuits such as adders [8], [12]–[14] and multipliers [15]. However, in order to expand the scope of approximate circuits, the ability to create approximate versions of arbitrary circuits without significant designer effort is essential. This leads to the need for an automatic *approximate synthesis* framework. Recognizing this need, recent work has addressed the problem of approximate *combinational* circuit synthesis by employing techniques such as redundancy propagation [7], gate deletion [9], don't care based simplification [6], [10], and node substitution [11].

Notwithstanding their promise, there is a significant gap between these techniques and what is required in practice. In general, the designer is not directly concerned with quality at the output of a combinational circuit block at the end of every cycle; rather, output quality is naturally specified at a coarser granularity *i.e.*, after several cycles of a sequential computation. For example, consider an implementation of JPEG compression, in which we desire to approximate the Discrete Cosine Transform (DCT) block. It is well known that some inaccuracy in the less significant DCT outputs can be tolerated [16], [17]. The DCT computation is often realized as a sequential circuit that operates over multiple cycles. A designer who wishes to use approximate combinational building blocks in the DCT should answer the question of how the quality constraint at the output of a sequential circuit can be translated and specified at the outputs of its constituent combinational blocks. This is a challenging problem, and requires modeling of how "errors" due to approximations are generated and propagated in each cycle of computation. Due to the cyclic nature of sequential circuits, errors may get re-circulated through the approximate circuit before the outputs are generated. From a different perspective, considering the sequential nature of circuits leads to better opportunities for approximation since different cycles or circuit blocks may not have the same significance towards the output. This spatio-temporal disparity can be exploited to approximate the circuit more aggressively in less significant blocks or cycles of operation.

We present ASLAN, the first automatic methodology for the synthesis of approximate sequential circuits. Given a sequential circuit and a quality constraint at its output, the proposed framework automatically synthesizes an approximate version of the sequential circuit that consumes lower energy while guaranteeing that the specified quality constraint is met.

Two key challenges need to be addressed in approximating sequential circuits. The first challenge is to identify the impact of approximating parts of the circuit on the *global* output quality, which is observed after multiple cycles of operation. In order to address this challenge, we construct a *Sequential*

This work was supported in part by the National Science Foundation under grant no. 1018621.

978-3-9815370-2-4/DATE14/©2014 EDAA

Quality Constraint Circuit (SQCC) composed of the original and approximate versions of the sequential circuit, along with a *Quality Evaluation Circuit* (QEC) that codifies the user-specified constraints on global output quality. We formulate the problem of ensuring the quality constraints as a sequential model checking problem [18], [19] by identifying safety and liveness properties in the SQCC that guarantee the validity of the approximate circuit. We then employ formal verification techniques to ensure that the selected approximations satisfy these properties, thereby enforcing quality during synthesis.

A second challenge is how to choose approximations so as to maximize energy savings for a given output quality constraint. We leverage the considerable body of previous work on approximate combinational circuits for this purpose. We identify circuit blocks (*e.g.*, arithmetic components) within the sequential circuit that are amenable to approximations and iteratively simplify them using existing combinational approximation techniques, while enforcing quality constraints on the entire sequential circuit as described above.

In summary, the key contributions of this work are as follows:

- We propose the first automatic framework to synthesize approximate sequential circuits which meet specified quality constraints.
- The proposed framework is adaptable to a variety of quality metrics and builds upon previously proposed combinational approximate design techniques.
- We apply the proposed framework and demonstrate significant benefits in energy and area across a wide range of benchmarks. We also present case studies of using the approximate circuits generated by our framework in two well known applications — MPEG Encoder and K-Means Clustering — illustrating the utility of the proposed techniques.

The rest of the paper is organized as follows. Section II presents an overview of related previous work. Section III describes the problem formulation and the design approach adopted in ASLAN. Section IV details the ASLAN methodology and the associated heuristics. Section V explains the experimental methodology used to evaluate ASLAN, and the results are presented in section VI. Section VII summarizes and concludes the paper.

II. Related Work

Previous efforts in approximate computing have considered techniques at different levels of design abstraction including algorithms, architecture and circuits. In this section, we limit our discussion to prior work at the circuit level.

The first wave of research efforts in approximate circuit design focused on specific manual designs for simple arithmetic components such as adders [8], [12]–[14] and multipliers [15]. However, to employ approximate computing in a broader context, a generic framework which not only allows approximation of more complex circuits, but also guarantees the user-specified quality constraints [20], is desired.

Recognizing this need, recent efforts have proposed automatic techniques for approximating any arbitrary circuit. The first effort targeted two-level circuits, in which a simplified sum-of-products implementation was obtained by complementing selected minterms [6]. In the case of multi-level circuits, [7] proposed to inject stuck-at-faults at certain nodes in the circuit and then simplify it by propagating the redundancy. In [9], path activation probabilities were employed to delete gates in the circuit that were least active. In contrast, SALSA [10] used the quality constraints to derive external don't cares to the circuit thereby enabling logic approximation through traditional Boolean optimization. Finally, SASIMI [11] took advantage of similarities between the functions computed by circuit nodes to perform node substitution followed by simplification.

All of the above techniques are applicable only to combinational circuits. In contrast, we propose the first solution to address the challenge of approximating sequential circuits, in which quality constraints can be specified at a coarser granularity, *i.e.*, across multiple cycles of computation. Our methodology internally utilizes combinational approximation techniques and is therefore complementary to the above existing techniques.

Some other works which address usage of approximate building blocks in a larger circuit, handle the problem at a different level of abstraction [21], [22]. In these works, the error analysis is done at the level of data flow graphs and are limited to acyclic structures, selective number of arithmetic operations (addition, multiplication) and linear functions. In contrast, by using general purpose verification techniques, we are able to address the problem in a broader and more general context.

III. ASLAN: Problem Formulation and Design Approach

The primary objective of ASLAN is to enable automatic synthesis of approximate versions of a sequential circuit, which satisfy a designer-specified quality constraint at its primary output. The inputs to ASLAN are: (i) the original sequential circuit, described either using RTL or as a synthesized netlist, and (ii) a *Quality Evaluation Circuit*, in which the quality constraints to be met are codified as a logic function. This section provides a description of the problem formulation employed in ASLAN and outlines the design approach adapted for introducing approximations in sequential circuits.

A. Sequential quality constraint circuit

We formulate the problem of sequential logic approximation by constructing a Sequential Quality Constraint Circuit (SQCC) as shown in Figure 1. The primary function of the SQCC is to characterize the impact of approximations on the primary outputs of the sequential circuit (produced after several cycles), and as a consequence, enable ASLAN to enforce the specified quality constraints during synthesis. The SQCC is a virtual circuit composed of three components: (i) the original sequential circuit, (ii) the approximate sequential circuit and (iii) the Quality Evaluation Circuit (QEC). As illustrated in Figure 1, the inputs to the SQCC are the primary inputs of the original and approximate versions of the circuit. The primary outputs and state registers of both circuits are fed into the QEC, which then evaluates the quality constraints and generates two bits *viz.* Quality (Q) and Quality Valid (V). In the rest of this section, we describe the QEC, derive the properties that must be satisfied for a given approximate circuit to meet the quality constraint, and discuss how this framework can be used to automatically perform quality constrained approximation.

1) Quality evaluation circuit: The quality evaluation circuit (Figure 1) monitors the outputs and state registers from the original and approximate circuits and indicates whether the quality constraints are satisfied through the Q and V output bits. The QEC performs two key functions:

- It checks the state registers of the original and approximate circuits and captures their outputs after the circuits

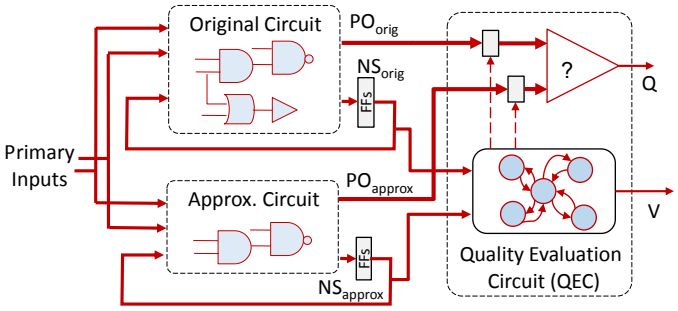


Fig. 1: Sequential quality constraint circuit

have reached completion. The valid output V is set indicating that the approximate circuit output is now ready to be evaluated for quality.

- It then compares the outputs of the original and approximate circuits and the quality output Q is set based on whether the quality constraint is satisfied.

The QEC is similar to a test bench used to verify the functionality of a sequential circuit with respect to golden results; in this case, the golden results are provided by the original circuit, and exact equivalence is relaxed as specified by the quality constraint.

2) Properties to guarantee output quality: In ASLAN, the problem of ensuring that the approximate circuit satisfies the quality constraints is formulated as a sequential model checking problem on the SQCC. Specifically, in order for the approximate circuit to be acceptable, the following Linear Temporal Logic (LTL) [18], [19] properties have to be true:

1) $\square(V \rightarrow Q)$, *i.e.,* in all possible states of the SQCC, if V is *true*, then Q should be *true*.
2) $\Diamond(V)$, *i.e., V* eventually becomes *true* along all possible paths through the state space of the SQCC.

The first property is a safety property, which ensures that whenever both the original and approximate circuits have produced their outputs (V output is '1'), the approximate output should satisfy the quality bounds (Q output is '1'). The second property is a liveness property, which states that original and approximate circuits should eventually produce their respective outputs. The approximate circuit is illegal if the SQCC violates any of the above properties. The violation in quality may be a result of two scenarios: (i) the approximate circuit reaches completion for all input sequences but some approximate outputs do not satisfy the quality constraint. In this case, the SQCC violates the safety property and produces an output of $(V = 1, Q = 0)$, or (ii) alternatively, the approximate circuit may fail to reach completion for certain inputs. In this case, the liveness property is violated and the V output would never reach '1' for those input sequences.

3) Formal verification of quality: In ASLAN, we utilize formal verification methods to guarantee that the safety and liveness properties described above are preserved during synthesis. In particular, we employ time frame expansion, a technique widely used in sequential circuit analysis and verification, on the SQCC to verify the validity of the approximate circuit. Figure 2 depicts the procedure used to unroll the SQCC over time. The SQCC is unrolled iteratively by one time frame in each iteration until the logical OR of the V signals from each time frame ($V_{comp} = V_0|V_1|...|V_N$) evaluates to a tautology. The logical OR is performed because the circuit, based on its inputs, may take variable number of cycles to complete and hence the V signal will reach logic high in any one of the unrolled instance. This ensures that the SQCC satisfies the liveness property. Next, a MUX is connected to the Q output signals from each time frame, with its select signals fed from the corresponding V output. The composite quality output (Q_{comp}) from the MUX is checked for tautology. This verifies the safety property and the approximate circuit is deemed to satisfy the quality constraints.

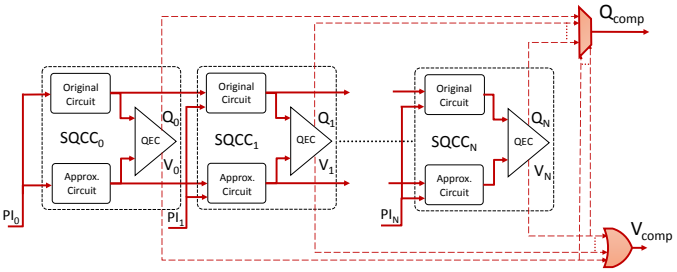


Fig. 2: Formal verification of quality using SQCC

B. Quality constrained approximations

A second key challenge is the manner in which approximations are chosen in order to maximize savings in energy for a given quality bound. In ASLAN, we first identify combinational blocks such as arithmetic components (adders, multipliers *etc.*) within the sequential circuit that are amenable to approximation. After identifying suitable candidates for approximation, we utilize existing combinational approximate design techniques to generate local quality *v.s.* energy $(Q-E)$ graphs for each of the candidates. The problem then boils down to choosing the right $(Q - E)$ operating point for each candidate such that the overall energy consumption of the circuit is minimized while the quality constraints are preserved.

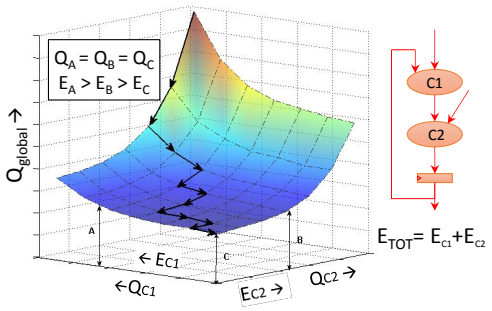


Fig. 3: Selection of components for approximation

Consider the sequential circuit shown in Figure 3 that contains two components ($C1$,$C2$) that are candidates for approximation. Figure 3 also shows the surface plot of global output quality (Q_{global}) obtained with respect to the output quality (and corresponding energy) of each candidate (Q_{C1}/E_{C1}, Q_{C2}/E_{C2}). In particular, the three points A, B and C with different local quality configurations result in the same global quality. However, the total energy consumed ($E_{TOT} = E_{C1} + E_{C2}$) is vastly different for each of these configurations. Thus in order to identify an energy-efficient configuration for a given global output quality, we adopt a gradient descent approach as shown in Figure 3. In each iteration, we employ heuristics to rank the candidates based on the error they introduce when approximated and the corresponding energy savings they engender. We select the best candidate for approximation and utilize the SQCC formulation to verify that the global quality constraint is satisfied. This process is

repeated until none of the candidates can be approximated any further without violating the quality constraint.

IV. ASLAN: Design Methodology

This section describes the methodology adopted in ASLAN to realize the approach discussed in Section III. The various heuristics employed to select different combinational blocks and the amount by which they should be approximated are also detailed.

A. Overall methodology

Algorithm 1 describes the procedure employed in ASLAN for synthesis of approximate sequential circuits. The inputs to the algorithm are the original sequential circuit (Ckt_{orig}) and the quality evaluation circuit (Ckt_{QEC}), which as described in Section III.A.1, specifies the global quality constraint to be met at the output of the sequential circuit. First, the arithmetic blocks that are amenable to approximation are identified within the sequential circuit (line 3) and are designated as candidates for approximation ($ApproxCand_{List}$). Next, for each approximation candidate, existing combinational approximate design techniques are utilized to obtain a series of approximate versions having different local quality levels. Using these approximate versions, a local quality *vs.* energy trade-off graph (QE_{List}) is constructed for each of the approximation candidates (line 4).

Algorithm 1 ASLAN Algorithm

Input: Original Sequential Circuit: Ckt_{orig},
Quality Evaluation Circuit: Ckt_{QEC}
Output: Approximate Sequential Circuit: Ckt_{approx}
1: **Begin**
2: Initialize: Ckt_{approx} = Ckt_{orig}
3: $ApproxCand_{List}$ = *get_approx_candidates*(Ckt_{orig})
4: QE_{List} = *form_local_qe_curves*($ApproxCand_{List}$)
5: `NextIteration:`
6: $ApproxCand_{Sorted}$ = *sort_approx_candidates* (Ckt_{approx},$ApproxCand_{List}$, QE_{List})
7: **for** each C *in* $ApproxCand_{Sorted}$ **do**
8: C_{approx} = *get_next_approx_version*(C,QE_{List})
9: Ckt^{new}_{approx} = replace C with C_{approx} in Ckt_{approx}
10: Q = verify_quality(Ckt_{orig}, Ckt_{QEC}, Ckt_{approx})
11: **if** (Q == ***true***) **then**
12: Ckt_{approx} = Ckt^{new}_{approx}
13: **goto** `NextIteration`
14: **end if**
15: **end for**
16: **return** Ckt_{approx}
17: **End**

The ASLAN algorithm is iterative (lines 5-15) and is performed until none of the approximation candidates, identified in line 3 can be further approximated. In each iteration, the following steps are performed. In order to identify the best candidate to approximate, the candidates are sorted (line 6) based on various heuristics described in Algorithm 2. These heuristics utilize the pre-generated Q-E graphs of the candidates to estimate the energy savings when approximated and the corresponding error introduced in the implementation. The candidate at the head of the sorted list ($ApproxCand_{Sorted}$) is selected (line 8), and is replaced by its approximate version in the sequential circuit (line 9). Once the new approximate circuit is formed, the SQCC formulation explained in Section III-A3 is used to verify if the approximate circuit meets the global quality constraint specified in the QEC. If the quality constraint is met (line 11), the algorithm proceeds to the next iteration (line 13) where the steps described above (lines 5-15) are repeated. On the other hand, if the quality constraint is violated, the current approximation is ignored and lines 7-15 are repeated with the next candidate from the sorted candidate list ($ApproxCand_{Sorted}$). If all the approximation candidates in the $ApproxCand_{Sorted}$ list are found to violate the quality constraint, the procedure terminates and the current approximate version of the circuit is returned as the output.

B. Heuristics for selecting approximation candidates

One of the key steps in Algorithm 1 is the manner in which approximations are chosen in order to maximize the energy savings for a given quality constraint at the output. Algorithm 2 illustrates the various heuristics employed in ASLAN to select the candidates for approximation. In this procedure, a figure of merit ($C.FOM$) is computed for each of the candidates based on the following parameters (line 9): (i) the proportion of energy consumed by the candidate in the overall circuit (line 5), (ii) the additional savings in energy obtained by further approximating the candidate (line 8), and (iii) the local error introduced in the circuit when the candidate is approximated (line 6). These parameters are directly obtained from the quality *v.s.* energy trade-off curves that were pre-generated for each of the candidates. The candidates are ranked by their figure of merit and a sorted list of candidates ($ApproxCand_{Sorted}$) is produced as the output.

Algorithm 2 Selecting candidates for approximation

Input: Ckt: Circuit,
$ApproxCand_{List}$: List of Candidates for Approximation,
QE_{List}: Q-E trade-off curve for each component
Output: $ApproxCand_{Sorted}$: Sorted list of candidates
1: **Begin**
2: $Ckt.E$: Energy Consumed by the Circuit
3: **for** each C in $ApproxCand_{List}$ **do**
4: $C.Q_{curr}$ = *get_current_quality*(C)
5: $C.E_{curr}$: Energy of C with quality $C.Q_{curr}$
6: $C.QStep$: Step size of the Q-E curve for C
7: $C.Q_{new}$ = $C.Q_{curr}$ + $C.QStep$
8: $C.E_{new}$: Energy of C with quality $C.Q_{new}$
9: $C.FOM = \frac{(C.E_{curr}-C.E_{new})*C.E_{new}}{Ckt.E} \Big/ C.QStep$
10: **end for**
11: $ApproxCand_{Sorted}$ = sort candidates by $C.FOM$
12: **return** $ApproxCand_{Sorted}$
13: **End**

Thus ASLAN enables automatic synthesis of approximate sequential circuits that are guaranteed to meet the desired quality specifications.

V. Experimental Methodology

In order to evaluate the proposed methodology, we utilized ASLAN to synthesize approximate versions of a wide range of sequential benchmarks listed in Table I. The CAD flow used to implement ASLAN and the quality metrics used in the experiments are described below.

A. CAD flow

The CAD flow employed in the implementation of ASLAN is shown in Figure 4. In our experimental setup, the circuits were synthesized using Synopsys Design Compiler [23] and mapped to the NangateOpenCell library based on the FreePDK

TABLE I: Circuits used in experiments

Name	Function	Gate Count	I/O	FFs
FIR	16-tap FIR filter	2373	10/16	18
IIR	8-tap IIR filter	1919	18/18	20
L1	Sum of Absolute Difference	639	34/12	13
L2	Euclidean Distance	1113	30/17	18
MAC	Vector Dot Product	1067	30/16	17
DCT4	4-input Discrete Cosine Transform Block	3001	34/72	74
DCT8	8-input Discrete Cosine Transform Block	5823	58/144	146
SOB	Sobel Operation (Used in edge detection algorithm)	708	66/13	22
BUT	Butterfly Operation (Used in FFT Computation)	474	18/26	20
NUR	8-input Neuron	1839	130/24	26

45nm technology node. The algorithms and heuristics described in Section IV were implemented as a custom tool. Time frame expansion of the SQCC was carried out using Odin II [24] and ABC [25], and the unrolled circuit was converted to Conjuctive Normal Form (CNF) using Verilog2CNF [26]. The minisat [27] SAT solver was then used to determine if the unrolled SQCC circuit evaluated to a tautology.

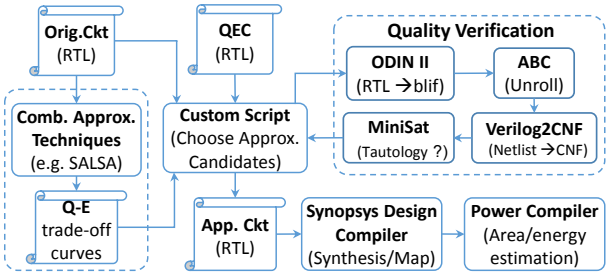


Fig. 4: CAD flow for ASLAN

Two approximate design techniques *viz.* precision scaling and SALSA [10] were used to generate the quality *v.s.* energy graphs for each approximation candidate and Synposys Design Compiler was employed to simplify the sequential circuit after the approximations were introduced. Finally, Synopsys Power Compiler [23] was used to estimate the overall power consumption of the synthesized approximate circuits. The approximate sequential circuits were also used in two applications — MPEG encoder and K-Means clustering — and the impact on application level output quality was evaluated.

B. Quality metrics

The benchmarks were evaluated for two different quality metrics, maximum error magnitude and relative error. The maximum error magnitude, given in Equation 1, is defined as the absolute difference in magnitude between the outputs of the original and approximate circuits.

$$MaximumErrorMagnitude = |O_{orig} - O_{approx}| \quad (1)$$

Relative error, given in Equation 2, is defined as the absolute value of the difference between 1 and the ratio of the approximate output to the original output.

$$RelativeError = \left|1 - \frac{O_{approx}}{O_{orig}}\right| \quad (2)$$

VI. Experimental Results

This section presents the results of various experiments that compare the approximate circuits generated using ASLAN for a wide range of sequential benchmarks. We later present case studies that analyze the impact of the approximate circuits generated using ASLAN at the application level.

A. Comparison of area and energy for approximate circuits

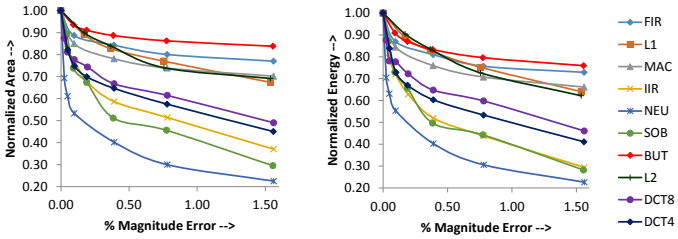


Fig. 5: Area and energy benefits for max. error metric

Figure 5 shows the relative area and energy (the ratio of approximate circuit value to original circuit value) obtained at iso-delay using different maximum error magnitude constraints for the benchmark circuits. The maximum error magnitude on the X axis is expressed as a percentage of the maximum output value of the circuit. We used precision scaling as the underlying combinational approximation technique for the constituent components. Because of the exponential increase in the significance of bits from LSB to MSB, the approximation capability using precision scaling decreases with increasing error magnitude. This is reflected in the diminishing returns in area and energy as the actual magnitude of error increases in Figure 5. Across all benchmarks, the area benefits range from 1.18X-2.49X for tight quality constraints ($< 0.5\%$ degradation in output quality) and from 1.25X-4.44X for moderately relaxed quality constraints ($< 2\%$ degradation in output quality). The corresponding energy savings range between 1.20X-2.44X and 1.32X-4.42X for tight and relaxed quality bounds respectively.

Similar trends in area and energy were observed with the relative error metric. However, in this case, we used SALSA [10] to synthesize approximate versions for the constituent combinational blocks. Figure 6 illustrates the relative area and energy savings obtained at iso-delay. Area and energy improvements between 1.1X-1.55X and 1.1X-1.58X were observed for this quality metric.

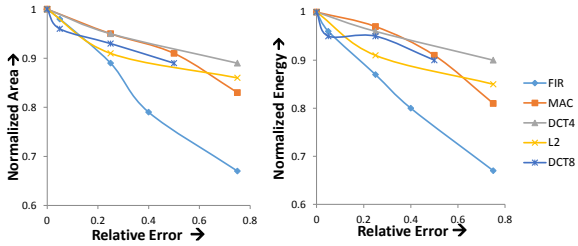


Fig. 6: Area and energy benefits for relative error metric

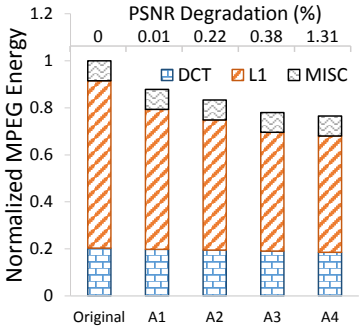
These results demonstrate the generality of ASLAN to different quality metrics, constraints and combinational approximate design techniques.

B. Application-level evaluation of ASLAN circuits

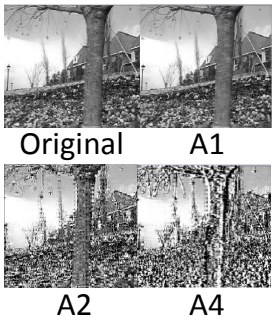
We utilized the circuits synthesized using ASLAN in two commonly used applications *viz.* MPEG video encoding and K-Means clustering of handwritten character images, and analyzed the impact of approximations at the application level.

	A1	A2	A3	A4
DCT	0.01	0.01	0.02	0.05
L1-Norm	0.37	0.76	1.54	3.10

(a) Maximum error (%) constraints in DCT and L1-Norm units



(b) Energy savings obtained using different approximate versions



(c) Snapshot of MPEG encoder output for "garden" benchmark

Fig. 7: MPEG encoder results with maximum error metric

1) MPEG Encoder: We used the MPEG encoder implementation discussed in [8] to compress 20 frames of 6 YUV video benchmarks [28] *viz.* Akiyo, Garden, Bowling, Container, Coastguard and Foreman. We identified the L1-norm (used in motion estimation) and DCT blocks in the MPEG encoder to be computationally dominant, accounting for 71% and 20% of the energy consumption respectively. We used ASLAN to synthesize four different approximate versions (*A1*, *A2*, *A3* and *A4*) for both the blocks with maximum error constraints shown in Figure 7(a). The quality constraints for the approximate implementations of the blocks were chosen empirically based on their relative impact on the output video quality (PSNR: Peak signal-to-noise ratio). Accordingly, we applied tighter quality constraints to the DCT block compared to L1-norm. Across all the video benchmarks, as shown in Figure 7(b), we obtained energy savings between 1.15X-1.34X at the application level for an average PSNR degradation in the range of 0.01%-1.31%. Figure 7(c) also shows some representative frames from the "garden" video using different approximate implementations.

2) K-Means Clustering: The algorithm for K-means clustering was implemented using the architecture described in [29]. The L2-norm block, which consumed 55% of the total energy, was approximated using ASLAN with different maximum error constraints. Figure 8 shows the application energy and the corresponding application output quality for different error constraints at the output of L2-norm. In this case, output quality is defined as the average distance of the points from the centroids of their respective clusters. We obtained savings of 1.26X in application energy for a negligible loss (0.87%) in output quality.

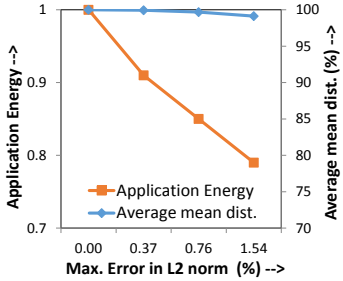


Fig. 8: K-Means results with maximum error metric

In summary, the results demonstrate the effectiveness of ASLAN in synthesizing energy efficient approximate sequential circuits that meet specified quality constraints.

VII. Conclusion

Emerging applications demonstrate intrinsic resilience, providing new opportunities to design energy efficient computing systems. One of the promising ways to exploit this resilience is to design approximate circuits that evaluate the computations in an approximate manner, subject to quality constraints specified by the designer. In this work, we proposed ASLAN, an automated framework for synthesizing approximate versions of sequential circuits. Given an RTL description for the circuit to be approximated and a quality constraint at the output, ASLAN automatically generates approximate versions of the circuit that meet the specified constraint. The approximate circuits synthesized using ASLAN demonstrate significant benefits in both area and energy across a wide range of benchmarks. We further demonstrated the utility of ASLAN by using it in the context of two widely used applications.

References

[1] M. A. Breuer. Multi-media applications and imprecise computation. In *Proc. Euromicro Conf. on Digital System Design*, pages 2–7, Sept. 2005.
[2] S. T. Chakradhar and A. Raghunathan. Best-effort computing: Rethinking parallel software and hardware. In *Proc. DAC*, pages 865 –870, June 2010.
[3] R. Hegde and N. R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proc. ISLPED*, pages 30–35, 1999.
[4] K. Palem et. al. Sustaining moore's law in embedded computing through probabilistic and approximate design: Retrospects and prospects. In *Proc. CASES*, pages 1–10, 2009.
[5] P. K. Krause and I. Polian. Adaptive voltage over-scaling for resilient applications. In *Proc. DATE*, pages 1 –6, March 2011.
[6] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Proc. DATE*, pages 957–960, Mar. 2010.
[7] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *Proc. DATE*, Mar. 2011.
[8] V. Gupta et. al. IMPACT: Imprecise adders for low-power approximate computing. In *Proc. ISLPED 2011*, pages 409 –414, Aug. 2011.
[9] A. Lingamneni K. Palem et. al. Energy parsimonious circuit design through probabilistic pruning. In *Proc. DATE*, Mar. 2011.
[10] S. Venkataramani et. al. Salsa: systematic logic synthesis of approximate circuits. In *Proc. DAC*, pages 796–801, 2012.
[11] S. Venkataramani et. al. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *Proc. DATE*, pages 1367–1372, 2013.
[12] D. Shin and S. K. Gupta. A re-design technique for datapath modules in error tolerant applications. In *Proc. ATS*, pages 431 –437, Nov. 2008.
[13] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proc. DAC*, pages 820–825, 2012.
[14] M. Olivieri et. al. Analysis and implementation of a novel leading zero anticipation algorithm for floating-point arithmetic units. *Circuits and Systems II: Express Briefs, IEEE Trans. on*, 54(8):685 –689, Aug. 2007.
[15] P. Kulkarni et. al. Trading accuracy for power with an underdesigned multiplier architecture. In *Proc. VLSI Design*, pages 346 –351, Jan. 2011.
[16] N. Banerjee et al. Process variation tolerant low power dct architecture. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1–6, 2007.
[17] K. Lengwehasatit et al. Scalable variable complexity approximate forward dct. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(11):1236–1248, 2004.
[18] O. Grumberg E. Clarke and D. Peled. Model Checking, MIT Press. 1999.
[19] Z. Manna and A. Pnueli. The Temporal Logic of Reactive and Concurrent Systems, Springer-Verlag. 1992.
[20] R. Venkatesan et. al. MACACO: Modeling and analysis of circuits for approximate computing. In *Proc. ICCAD*, pages 667 –673, Nov. 2011.
[21] J. Huang J. Lach et. al. A methodology for energy-quality tradeoff using imprecise hardware. In *Proc. DAC*, pages 504–509, 2012.
[22] Z. Kedemi et. al. Optimizing energy to minimize errors in dataflow graph using approximate adders. In *CASES*, pages 177–186, 2010.
[23] Design Compiler, Synopsys Inc.
[24] P. Jamieson et. al. Odin II - An Open-source Verilog HDL Synthesis tool for CAD Research. In *Proc. FCCM*, pages 149–156, 2010.
[25] ABC: A System for Sequential Synthesis and Verification, Release 130314. http://www.eecs.berkeley.edu/~alanmi/abc/.
[26] Verilog2CNF,"http://web.eecs.umich.edu/~valeria/teaching/eecs578_F09_old".
[27] Minisat,"http://minisat.se/ ".
[28] http://media.xiph.org/video/derf/ .
[29] S. Venkataramani et. al. Quality-programmable vector processors for approximate computing. In *Proc. MICRO*, pages 1–12, 2013.