Substitute-and-Simplify: A Unified Design Paradigm for Approximate and Quality Configurable Circuits

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

***Abstract*—Many applications are inherently resilient to inexactness or approximations in their underlying computations. Approximate circuit design is an emerging paradigm that exploits this inherent resilience to realize hardware implementations that are highly efficient in energy or performance.**

In this work, we propose Substitute-And-SIMplIfy (SASIMI), a new systematic approach to the design and synthesis of approximate circuits. The key insight behind SASIMI is to identify signal pairs in the circuit that assume the same value with high probability, and substitute one for the other. While these substitutions introduce functional approximations, if performed judiciously, they result in some logic to be eliminated from the circuit while also enabling downsizing of gates on critical paths (simplification), resulting in significant power savings. We propose an automatic synthesis framework that performs substitution and simplification iteratively, while ensuring that a user-specified quality constraint is satisfied. We extend the proposed framework to perform automatic synthesis of quality configurable circuits that can dynamically operate at different accuracy levels depending on application requirements. We used SASIMI to automatically synthesize approximate and quality configurable implementations of a wide range of arithmetic units (Adders, Multipliers, MAC), complex data paths (SAD, FFT butterfly, Euclidean distance) and ISCAS85 benchmarks, using various error metrics such as error rate and average error magnitude. The synthesized approximate circuits demonstrate power improvements of 10%-28% for tight error constraints, and 30%-60% for relaxed error constraints. The quality configurable circuits obtain between 14%-40% improvement in energy in the approximate mode, while incurring no energy overheads in the accurate mode.

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

I. Introduction

Many application domains — including multimedia processing, graphics, digital signal processing, and emerging application domains such as search, data analytics, recognition, mining, and synthesis — exhibit an inherent resilience to their underlying computations being performed in an inexact or approximate manner. This resilience is owed to several factors, including redundancies in the input data, statistical nature of computations, lack of a golden result, and limitations in human perception [1], [2]. Approximate computing is a rapidly evolving area of research that exploits the forgiving nature of applications to improve the energy consumption and performance of computing platforms.

A promising and popular approach to approximate computing is to design *approximate circuits* that have lower hardware complexity (switched capacitance, leakage, and critical path) while evaluating the required function within a specified accuracy or quality constraint. In some cases, this tradeoff between efficiency and quality may be performed at design time. However, in many applications, the degree of resilience often varies depending on the application context or the dataset being processed [3]. In such scenarios, it is desirable to construct *quality-configurable circuits*, or circuits that are capable of reconfiguring to adapt their accuracy at run-time. Traditionally, approximations in circuits have been introduced in two different forms: (i) Timing approximation, where the circuit is subject to voltage over-scaling resulting in timing errors, and (ii) Functional approximation, where the circuit realizes a slightly different logic function than specified, resulting in a more efficient implementation.

978-3-9815370-0-0/DATE13/©2013 EDAA

Most research efforts in the area of approximate circuits can be summarized as manual designs of simple arithmetic circuits such as adders [4]–[7] and multipliers [8]. However, the broader adoption of approximate circuits requires a systematic methodology to design approximate implementations for any arbitrary circuit. Moreover, it is critical that such a methodology enable the generation of "correct-by-construction" approximate circuits that are guaranteed to satisfy designer-specified quality constraints, which is often not the case for the aforementioned manual designs.

A few efforts have addressed the problem of automatic synthesis of approximate circuits by pruning gates from a conventional implementation [9], [10], or by identifying certain inputs as external don't cares [11], [12]. Although the problem that we address (in part) is the same, we take a very different approach that significantly extends the scope of optimizations that can be applied to approximate circuit synthesis. In addition, we propose the first automatic synthesis framework for quality-configurable circuits. It is worth noting that the quality-configurable circuits generated using our approach demonstrate significant energy savings in the approximate modes, without incurring any energy overheads when full accuracy is desired.

The key insight behind the proposed approach, called **S**ubstitute-**A**nd-**Sim**pl**i**fy (SASIMI), is to identify near-identical signal pairs, or signal pairs that assume the same value with high probability, and substitute one for the other. Such substitutions may introduce functional approximations — when an input causes the substituted signal to assume an incorrect value, and also causes the incorrect value to propagate to a circuit output. Naturally, it is important to restrict the substitutions chosen such that the impact on output quality is not excessive. On the other hand, well-chosen substitutions can lead to the simplification of the circuit by eliminating some of the logic that generates the substituted signal, while also downsizing logic in its transitive fan-out (since the substitution may introduce timing slack). We propose a procedure to automatically synthesize approximate circuits by iterating substitute-and-simplify in a quality-constrained loop.

We extend SASIMI to the synthesis of quality configurable circuits, by augmenting the approximate circuit to detect when errors are incurred at runtime, and utilizing an additional clock cycle to re-compute the logic in the transitive fanout of the substituted signal, thereby "correcting" the error. The error correction may be performed universally or selectively, or completely skipped based on the desired accuracy level.

While the accurate mode of operation is reminiscent of variable-latency circuits [13]–[16], we note that our design and synthesis approach differ significantly as the quality constraints imposed in the other approximate modes are considered. In contrast, traditional variable-latency design methodologies are oblivious to the degradation in output quality, since errors are always corrected.

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

- We propose a novel design technique, Substitute-and-Simplify, that judiciously employs signal substitutions to reduce the circuit's power consumption while satisfying the user-specified quality constraints.
- We propose the first approach to synthesize quality configurable circuits by selectively utilizing an additional clock cycle to correct errors that may violate the desired quality.
- We prototype and evaluate an automatic synthesis framework that embodies these concepts, demonstrating significant benefits in power and area across a wide range of arithmetic units, data paths and ISCAS85 benchmarks.

The rest of the paper is organized as follows. Section 2 presents an overview of related previous work. Section 3 details the SASIMI methodology and outlines the trade-offs involved. Section 4 explains the experimental methodology used to evaluate SASIMI, and the results are presented in section 5. Section 6 summarizes and concludes the paper.

II. Related Work

The topic of approximate computing has drawn great interest in the research community, with design techniques spanning software, architecture, and circuits. We limit our discussion to previous efforts on approximate circuit design.

Several approximate design techniques for simple arithmetic blocks have been proposed, including adaptive voltage over-scaling [17], functional approximation [4], [5], [7], [8] and output significance aware error compensation [4], [18]. A recent effort proposed a design for a quality configurable adder based on partial summations [6]. While these efforts demonstrate promising results, a systematic methodology that enables synthesis of arbitrary circuits using any specified quality metric is necessary to bring these approaches into the mainstream.

Approximate designs that employ voltage over-scaling to reduce energy consumption are limited when a large number of paths are critical or near-critical, resulting in drastic increase in errors beyond a critical operating point. To ensure a more graceful degradation in quality, techniques based on cell sizing [19] and common case promotion [20] have been proposed in the context of recovery based designs.

The first synthesis technique for functional approximation derived reduced sum-of-products implementations of two level circuits by designating selected minterms as don't cares [12]. For multi-level circuits, [9] proposed a scheme where stuck-at-faults are injected at selected nodes and the circuit is approximated by propagating this redundancy. Path activation probabilities were used in [10] to prune portions of the circuit that are activated the least. In contrast, [11] forms quality constraint circuits to identify external don't cares to the circuit and uses conventional Boolean optimizations for logic approximation.

SASIMI differs significantly from previous techniques in the following important respects.

- It introduces a new approach for circuit approximation that takes advantage of the correlations (or similarities) that exist among nodes in any circuit implementation, and uses node substitution (and subsequent circuit simplification) as a general approach to approximate circuit design. The circuit optimizations that are discovered through this approach cannot be realized by previous techniques such as removal of gates from the circuit.
- It proposes the first solution to the problem of synthesizing quality configurable circuits. Previous approaches cannot be directly applied or easily extended towards this purpose.

III. Substitute-and-Simplify: Design Approach and Methodology

The goal of approximate logic synthesis is to synthesize an approximate version of the given original circuit while meeting specifications for quality or error[1] at the output. In case of quality configurable circuits, the aim is synthesize a configurable version of the original circuit that, based on additional inputs to indicate the desired quality mode, scales the energy it expends to compute the output while satisfying the corresponding quality constraint. As mentioned earlier, Substitute-and-Simplify (SASIMI) provides a unified paradigm for the design of both types of circuits. This section outlines the basic design approach and the automatic design methodology that is proposed.

A. SASIMI: Design Approach

Figure 1 illustrates the basic approach adopted in SASIMI. The key idea is to identify pairs of signals in the circuit that are similar to each other in their logic functionality and substitute one signal in place of the other, thereby functionally approximating the circuit. The signal that is being replaced is called the target signal (TS) and the signal substituting TS is termed the substitute signal (SS). The candidates for SS can be logic zero, logic one and other signals (and their complements) in the circuit. If chosen judiciously, substitutions have the ability to bring about circuit simplifications (detailed below) that yield significant savings in area and power. For approximate circuit design, the substitute-and-simplify steps are performed successively until the approximate circuit reaches the target error constraint.

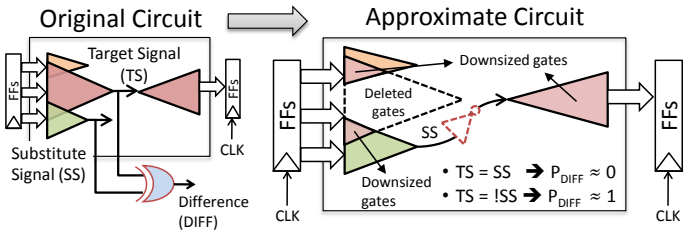

Fig. 1: Approximate circuit design using SASIMI

The trade-offs and desired criteria in choosing the TS-SS substitution pair are discussed below.

As seen from Figure 1, when the target signal gets substituted, the gates that are exclusive to the cone of logic that generates TS can be deleted. The logic in the transitive fan-out of TS, whose timing requirements are constrained by TS, is potentially downsized since the substitution with SS introduces timing slack. Further, the transitive fan-ins of TS that fan-out to other logic cones in the circuit can be sized independent of TS. Thus,

[1]Note that the terms "quality constraints" and "error constraints" are used interchangeably since quality is expressed in terms of acceptable error at the outputs of the circuit

in choosing TS, both the direct effect of logic elimination and the indirect impact of logic downsizing should be considered. Also, signals that fan-out to outputs that cannot tolerate errors, or signals that cause unacceptable degradation in output quality, are undesirable choices for TS.

In choosing the substitute signal, the potential error caused by the substitution should be considered. The error introduced can be inferred using the signal probability of the difference signal (P_{DIFF}), which is the *XOR* of TS and SS. Since each signal and its complement are SS candidates, the ones with least probability product - $P_{DIFF} * (1 - P_{DIFF})$ - of being different from TS is desired. However, it is worth noting that an error at TS need not be sensitized at the primary outputs and hence the actual circuit error has to be separately estimated. Further, to facilitate logic downsizing, substitute signals that introduce as larger slack at TS are desirable. We also impose a constraint that substitutions should not cause combinational cycles in the circuit.

B. *Quality Configurable circuit design using SASIMI*

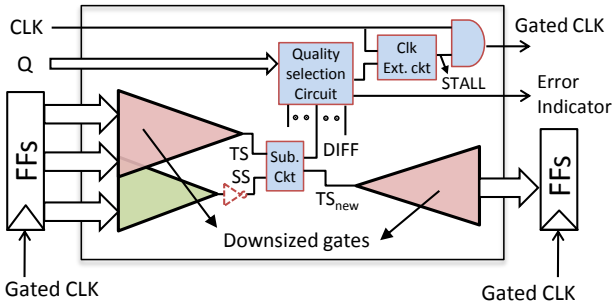

Fig. 2: Quality configurable circuit design using SASIMI

The above approach can be directly applied to the synthesis of quality configurable circuits with multiple quality modes. Without loss of generality, the approach is explained considering two quality modes - the accurate mode and the approximate mode. The key idea is to make the quality configurable circuit latency elastic and "recover" from errors caused due to approximations when the circuit is in the accurate mode. As illustrated in Figure 2, substitutions are performed in the circuit but the logic generating TS is retained and the difference between TS and SS is monitored. In the accurate mode, the circuit operates in a single cycle if TS and SS take the same value. Otherwise, an additional cycle is provided in which the correct result is re-computed from the point of substitution. In the approximate mode, since the error caused by the substitution is tolerable, any difference between TS and SS is ignored and the circuit always operates in a single cycle. Thus, based on the quality mode, the circuit selectively recovers from errors caused by substitutions that are intolerable. As shown in Figure 2, additional control inputs Q to the circuit indicate the desired quality mode.

Realizing quality configurable operation requires additional circuits for selective substitution, quality selection and clock extension, which are shown in Figure 3. The substitution circuit detects the difference between TS and SS, and allows the clock extension circuit to choose which of these signals is fed to downstream logic. At the start of every operation, the SS_i signals are chosen by default and the difference ($DIFF_i$) from all substitutions are accumulated (DS_{acc}). The quality selection circuit uses the accumulated difference signal (DS_{acc}) and the quality control bits (Q) to determine the need for a second clock cycle. If required, the clock extension circuit then gates the clock for a cycle and also sets the EFF flip-flop so that all substitution circuits select the corresponding TS_i signals in the second cycle, thereby correcting the error.

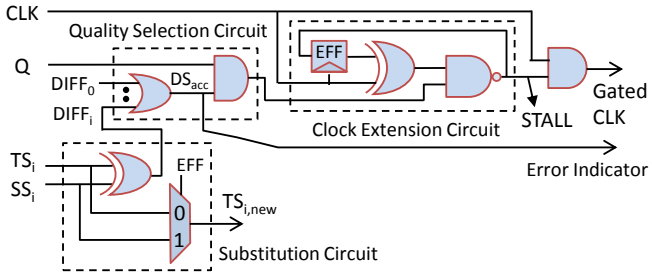

Fig. 3: Selective substitution, quality selection and clock extension circuits

For correct operation of quality configurable circuits, the following timing constraints have to be satisfied. First, to ensure completion of single cycle operations, Equation 1a constrains the delay of paths from inputs to outputs through SS_i. The terms $T_{cq}(I_{FF})$ and $T_{sp}(O_{FF})$ refer to C-Q delay and setup times of input and output FFs, respectively. Equation 1b and 1c ensure that the difference in substitutions are detected and the signals for EFF update and clock extension settle within the clock period. Finally, Equation 1d requires all paths originating from EFF to evaluate within the clock period, thereby ensuring two cycle operation.

$$T_{cq}\left(I_{FF}\right) + T_{max}\left(I \rightarrow SS_i \rightarrow O\right) + T_{sp}\left(O_{FF}\right) \leq T_{clk} \quad (1a)$$

$$T_{cq}\left(I_{FF}\right) + T_{max}\left(I \rightarrow TS_i \rightarrow STALL\right) + T_{sp}\left(E_{FF}\right) \leq T_{clk} \quad (1b)$$

$$T_{clk\_tr} + T_{cq}\left(I_{FF}\right) + T_{max}\left(I \rightarrow TS_i \rightarrow STALL\right) \leq T_{clk} \quad (1c)$$

$$T_{cq}\left(E_{FF}\right) + T_{max}\left(E_{FF} \rightarrow O\right) + T_{sp}\left(O_{FF}\right) \leq T_{clk} \quad (1d)$$

Note that clock skew is ignored in the above constraints for ease of explanation, but can be considered and addressed using conventional techniques.

The trade-offs involved in quality configurable synthesis are similar to approximate synthesis with one notable difference - since no logic is deleted, the potential for logic downsizing is critical. As shown in Figure 2, both the transitive fan-out and the independent logic of TS can be downsized as the timing constraint is relaxed at that point. However, additional area and power is consumed by the substitution, quality selection and clock extension circuits. Also, the occasional need for additional clock cycles in the accurate mode impacts performance. Our experiments demonstrate that, despite these overheads, by choosing proper substitution candidates, the energy consumed is lowered even in the accurate mode. The energy savings are larger in the approximate mode, with no impact on performance. If the switch between quality modes is coarse grained, then the clock extension circuit can be power gated in the approximate mode, leading to additional savings in power/energy. Note that the actual error in the approximate mode is evaluated at the circuit outputs as before. The error indicator output, shown in Figure 2, is conservative since the difference in TS and SS may not be sensitized to the circuit's output.

In cases where multiple approximate modes exist, the substitutions are grouped such that an additional cycle is provided only to selected substitutions that cause intolerable errors. Thus

from exclusive single cycle operation, the quality configurable circuit progressively recovers from more and more errors (or substitutions) as the quality constraints are tightened.

C. Methodology

This section details the automation of the proposed approach for approximate and quality configurable circuit synthesis.

Algorithm 1 Pseudo-code for SASIMI-Approximate

Input: Original Circuit: Ckt_{orig}, Target Error: ERR
Output: Approximate Circuit: Ckt_{approx}
1: **Begin**
2: Initialize: $NewCkt_{approx}$ = Ckt_{orig}
3: Approximate Circuit Error: ACE = 0
4: **while** $ACE \leq ERR$ **do**
5: Ckt_{approx} = $NewCkt_{approx}$
6: <TS,SS> = $get\_substitution\_candidate(Ckt_{approx})$
7: $NewCkt_{approx}$ = $substitute$ TS=SS in Ckt_{approx}
8: $Simplify$ ($NewCkt_{approx}$)
9: ACE = $compute\_error$ (Ckt_{orig}, $NewCkt_{approx}$)
10: **end while**
11: **return** Ckt_{approx}
12: **End**

Algorithm 1 describes the procedure for approximate circuit synthesis. The inputs to the algorithm are the original circuit (Ckt_{orig}) and the target error acceptable in its approximate implementation (ERR). The algorithm is iterative (lines 4-10) and performs successive substitutions until the imposed target error constraints are violated. In each iteration, the following steps are carried out: (i) the best candidate signal pair for substitution is identified (line 6), (ii) the substitution is performed (line 7) and the circuit is simplified (line 8), and (iii) the error in the approximate circuit is estimated (line 9). The algorithm proceeds to the next iteration if the approximate circuit error is less than the specified target error constraint (line 4). If not, the last legal approximate circuit is produced as the output (line 11).

The procedure used in the identification of candidate signal pairs for substitution is detailed in Algorithm 2. For each signal (S) in the given circuit, the following metrics are computed in lines 6-10. $S.TFI_{ind}$ (line 6) calculates the size of the independent logic that can be removed if S is substituted. This gives the measure of the logic deletion potential of S. $S.TFO_{crit}$ (line 8) is the size of the transitive fan-out of S, whose arrival times are constrained by it. This can be evaluated by setting the arrival time at S to be zero and recomputing the arrival times of its transitive fan-outs and identifying the gates whose arrival times have been relaxed. This, combined with the actual arrival time of S (maximum slack that can be introduced) gives the maximum potential for logic downsizing of S. A weighted sum of the normalized deletion and downsizing potentials is used to compute the signal score ($S.Score$) in line 10. The $S.Sens$ field (line 9) indicates if the signal fans-out to an output that cannot tolerate errors, in which case the $S.Score$ is made zero. Note that all the above fields can be computed efficiently in a single topological and reverse topological traversal of the circuit. The signals within a top fraction of the maximum score are designated as target signal TS candidates ($TS_{candidates}$).

Using the set $TS_{candidates}$, the best substitution pair is identified by lines 15-25 in Algorithm 2. For each signal in $TS_{candidates}$ and each legitimate substitute signal (SS) candidate, the substitution score ($SUB.Score$) is computed in line 19. $SUB.Score$ is similar to the signal score, except that the estimates of logic deleted and downsizing possible are refined. Further, the potential error introduced by the substitution (P_{diff}) is also taken into account. Using $SUB.Score$ as the metric, the best substitution pair is identified. It is worth noting that the computed $SUB.Score$ can be reused in the next iteration, if both TS and SS are unaffected during the simplify step, greatly reducing the runtime of Algorithm 2. The parameter α used to weight the logic deletion and downsizing potentials is chosen empirically based on the objective of the synthesis. In the case of approximate circuits, the deletion potential is given higher weight (larger α), since it is more significant that downsizing. In quality configurable synthesis, since the independent logic is retained, the downsizing potential carries more importance. For our experiments, α values of 0.75 and 0.25 were used for approximate and quality configurable synthesis, respectively.

Algorithm 2 Pseudo-code to obtain Substitution Candidate

Input: Circuit: Ckt
Output: Target-Substitute signal pair: TS_{best}, SS_{best}
1: **Begin**
2: Read Ckt and sort signals in topological order
3: A = Area of the circuit
4: D = Delay of circuit
5: **for** each S: Signals $\in Ckt$ **do**
6: $S.TFI_{ind}$ = Independent logic size in Tr.fan-in of S
7: $S.AT$ = Arrival time of S
8: $S.TFO_{crit}$ = Size of Tr.fan-out made critical by S
9: $S.Sens$ = 0 if S fans out to a sensitive output; else 1
10: $S.Score = S.Sens * \left[\alpha(\frac{S.TFI_{ind}}{A}) + (1-\alpha)(\frac{S.AT}{D})(\frac{S.TFO_{crit}}{A})\right]$
11: **end for**
12: $TS\_max\_score = \max(S.Score) \quad \forall S \in Ckt$
13: $TS_{candidates} = \{S \ni (S.Score \geq \beta * TS\_max\_score)\}$
14: SUB_max_score = 0
15: **for** each $TS \in TS_{set}$ **do**
16: **for** each $SS \in \{0, 1, S \in Ckt \ni S.AT < TS.AT\}$ **do**
17: P_{diff}: Probability of $TS \neq SS$
18: $SS = (P_{diff} < 0.5)\ ?\ K : \overline{K}$
19: $SUB.Score = \left[\alpha\left(\frac{TS.TFI_{ind} - (TS.TFI_{ind} \cap SS.TFI_{ind})}{A}\right) + (1-\alpha)\left(\frac{TS.AT - SS.AT}{D}\right)\left(\frac{S.TFO_{crit}}{A}\right)\right] \Big/ P_{diff} * (1 - P_{diff})$
20: **if** ($SUB.Score > SUB\_max\_score$) **then**
21: $SUB\_max\_score = SUB.Score$
22: <TS_{best}, SS_{best}> = <TS, SS>
23: **end if**
24: **end for**
25: **end for**
26: **return** TS_{best}, SS_{best}
27: **End**

Note that in choosing the best substitution candidate for a given circuit, the potential benefits in terms of logic deletion and downsizing are weighed against the error introduced by the substitution. Hence, although the substitutions are chosen in a greedy manner within each iteration of Algorithm 1, the use of benefits-to-error ratio (instead of choosing just based on the benefits) in the selection process implies that the algorithm does not entirely exhaust the error constraints in a given iteration but rather retains some flexibility for future iterations.

Algorithm 3 describes the methodology for quality configurable synthesis. A list of error constraints corresponding

to the desired quality modes is provided as input. For each quality mode (lines 3-12), the procedure resembles approximate synthesis (Algorithm 1), except that in addition, the substitutions are grouped based on the quality mode in the quality selection circuit (line 9). The substitution and clock extension circuits are also appropriately added to the circuit. Note that, in the proposed procedure, the set of substitutions causing two cycle operation in a given quality mode is a strict super set of all modes with lower quality constraints. This greatly simplifies the design of the quality selection circuit.

Algorithm 3 Pseudo-code for SASIMI-Quality-Configurable

Input: Original Circuit: Ckt_{orig}, Error List: ERR_{list}
Output: Quality Configurable Circuit: Ckt_{qc}

```
1:  Begin
2:  Initialize: NewCkt_qc = Ckt_orig + Clk_Extension_Ckt
3:    for each ERR ∈ ERR_list do
4:      while QC Circuit Error: QCE < ERR do
5:        Ckt_qc = NewCkt_qc
6:        <TS,SS> = get_substitution_candidate(Ckt_qc)
7:        # Insert substitution circuit
8:        # Add substitution to Quality selection circuit
9:        NewCkt_qc = form_qc_ckt TS=SS in Ckt_qc
10:       Simplify (NewCkt_qc)
11:       QCE = compute_error (Ckt_orig, NewCkt_qc)
12:     end while
13:   end for
14:   return Ckt_qc
15: End
```

Using Algorithms 1, 2 and 3, the SASIMI paradigm can automatically synthesize approximate and quality configurable circuits for any given circuit and quality constraint.

IV. Experimental Methodology

We evaluated SASIMI on a wide range of benchmarks including (i) arithmetic circuits *viz.* adders - Kogge stone (KSA), Ripple carry (RCA), Carry lookahead (CLA), multipliers - Wallace tree (WTM), array (MUL) and Multiply and Accumulate (MAC), (ii) complex datapath modules *viz.* Sum of Absolute difference (SAD), Euclidean Distance unit(EUDIST), FFT Butterfly (BUT), and (iii) circuits from the ISCAS85 benchmark suite. The benchmarks were synthesized using Synopsys Design Compiler Ultra [21] and mapped to the IBM 45nm technology library. The metrics used for identifying substitution candidates were computed using custom tools that analyzed the synthesized netlist. The signal probability calculation engine in Synopsys Power Compiler was used to obtain difference probabilities among signal pairs. For the experiments, all input vectors were considered equi-probable and the synthesized approximate and quality configurable circuits were evaluated for area and power at iso-delay. In the case of quality configurable circuits, the additional hardware overheads in substitution, clock extension and quality selection circuits are included during area and power estimation, and an energy comparison is performed taking into account the additional clock cycles incurred.

The circuits were synthesized for two different target quality metrics *viz.* Error Rate and Average Error Magnitude. The error rate, given in Equation 2, is defined as the percentage of inputs vectors for which the approximate circuit output differs from the original circuit.

$$ErrorRate = \frac{Total\ Inputs \ni O_{orig} \neq O_{approx}}{Total\ number\ of\ Inputs} \quad (2)$$

It is estimated by ORing the difference (O_{orig} xor O_{approx}) at all output bits and then computing its static signal probability.

The average error magnitude, shown in Equation 3, is the absolute difference in magnitude between the original and approximate circuits, averaged over all inputs.

$$AverageError = \frac{\sum_{\forall inputs} |O_{orig} - O_{approx}|}{Total\ number\ of\ Inputs} \quad (3)$$

The average error magnitude is evaluated by summing the difference signal probabilities of all output bits weighted by their numerical significance.

V. Results

This section presents the results of various experiments that compare circuits generated by SASIMI to well-optimized, accurate baselines.

A. Area and power comparison for approximate circuits

We begin by comparing the area and power consumed by the approximate circuits relative to the original circuit at iso-delay, for a range of error rate and average error magnitude constraints. In the case of error rate, as shown in Figure 4 for ISCAS85 benchmarks, we see significant benefits that amount to 15%-25% in area and 10%-28% in power are achieved for tight error rates (less than 0.5%). As the error constraints are relaxed (less than 2%), area and power improvements between 30%-60% are obtained.

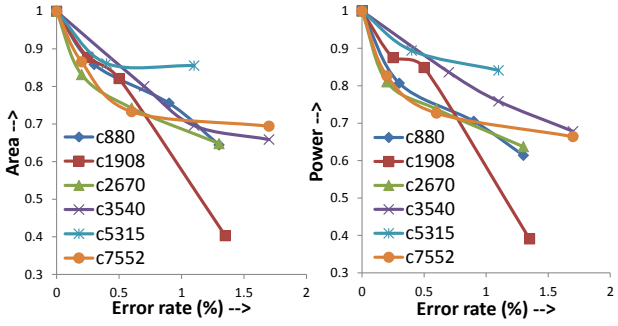

Fig. 4: Area and Power benefits for Error Rate metric

Similar trends are observed for the average error magnitude metric. As depicted in Figure 5, for average errors of less than 0.5% of the maximum value, improvements in the range of 15%-65% in area and 20%-68% in power are obtained. These results demonstrate the applicability and effectiveness of the SASIMI design approach and synthesis methodology.

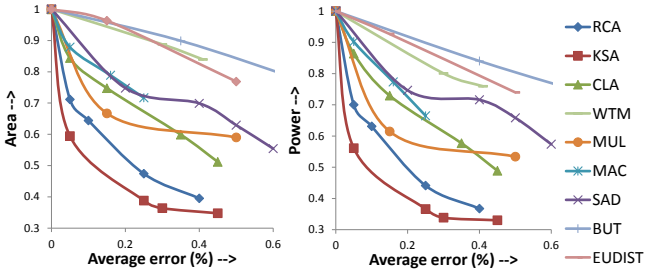

Fig. 5: Area and Power benefits for Average Error metric

To demonstrate the effectiveness of functional approximations across the entire design space, the approximate circuits were synthesized for a range of delay values. Figure 6 shows the area and power *vs.* delay plots for a 32-bit Kogge stone adder with error rate metric. The approximate circuit outperforms the

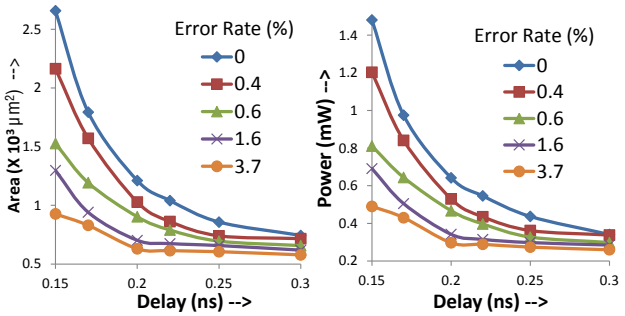

Fig. 6: Area and power of KSA with delay sweep

TABLE I: Quality configurable circuits synthesized for Average error with two quality modes

Circuit	Delay (ns)	Area (%)	Power (%)	P_{2cycle}	$Energy_{acc}$ (%)	$Energy_{app}$ (%)	Avg.Error (%)
RCA	0.65	16.2	21.6	0.24	2.69	36.7	0.03
CLA	0.17	29.5	26.72	0.067	21.8	34.9	0.01
MAC	0.55	13.7	16.8	0.014	15.6	18.3	0.01
EUDIST	0.47	24.1	22.24	0.0075	21.6	24	0.12
MUL	0.35	32.9	36.5	0.05	33.3	40.05	1.2
SAD	0.5	11.6	12.1	0.002	12.0	14.44	0.01

original circuit at all delay values, which testifies to the wide scope of the optimizations performed by SASIMI.

B. Quality configurable circuits

Next, we present the results obtained for quality configurable designs with two quality modes synthesized by SASIMI. Table I tabulates the area, power and energy improvements obtained relative to the original circuit for the average error magnitude metric. Column 3 shows the percentage reduction in area. Despite the area overheads of the additional circuits, the overall area savings range from 13%-32% compared to the original circuit. These benefits come from the logic downsizing that was possible by the substitutions. Column 4 provides the corresponding improvement in power. The probability of two cycle operations in the accurate mode is listed in Column 5. The total energy savings in the accurate and approximate modes are listed in Columns 6 and 7. In the accurate mode, the circuit recovers from all errors, and the energy benefits are due to the savings from logic downsizing outweighing the overheads of the added control circuitry and two cycle operation. In the approximate mode, the additional energy savings stem from two sources - the circuit has fewer or no two cycle operations as it does not recover from all errors, and the clock extension circuit could be power gated in the lowest quality mode. Due to these reasons, net energy savings between 14%-40% are obtained. Finally, column 8 gives the average error magnitude in the approximate mode of the circuit.

Table II reports similar results for circuits synthesized using the error rate metric. Energy benefits in the range of 22%-32% are obtained in the approximate mode for error rates less than 1%. Note that the rate of two cycle operations is a little larger than the actual error rate at the outputs, because not all errors at the substitution points are sensitized at the outputs. In other words, the clock extension mechanism and the error indicator output bit are conservative. In summary, our experiments suggest that SASIMI is a promising approach for the synthesis of approximate and quality configurable circuits.

TABLE II: Quality configurable circuits synthesized for Error Rate with two quality modes

Circuit	Delay (ns)	Area (%)	Power (%)	P_{2cycle}	$Energy_{acc}$ (%)	$Energy_{app}$ (%)	Error rate (%)
KSA	0.2	16.3	14.79	0.009	14.02	22.27	0.7
c880	0.22	13.1	18.03	0.064	12.78	31.9	4.8
c1908	0.25	13.8	22.9	0.0102	22.11	31.31	0.95
c2670	0.22	5.09	15.68	0.0051	15.25	24.51	0.2
c3540	0.36	21.94	19.72	0.008	19.08	23.56	0.65
c7552	0.32	12.79	19.18	0.064	14.01	22.54	4.8

VI. Conclusion

The inherent resilience prevalent in many applications provides designers with new avenues for optimizing design by forsaking exact equivalence between hardware implementations and their specifications. We propose Substitute-and-Simplify, a unified approach to the synthesis of approximate and quality configurable circuits. The key idea is to identify substitutions in the circuit that foster logic deletion and downsizing (simplification) while introducing minimal error. Given a user specified quality requirement, the substitution and simplification steps are iteratively performed while the error constraints are satisfied. We further extended the approach to synthesize quality configurable circuits, where at runtime, processing of selected input vectors is given an additional cycle to correct errors due to approximations. Across a range of benchmark circuits, our approach results in significant area, power and energy benefits.

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

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: Re-thinking parallel software and hardware. In *Proc. DAC*, pages 865 –870, June 2010.
[3] V. K. Chippa et. al. Dynamic effort scaling: managing the quality-efficiency tradeoff. In *Proc. DAC*, pages 603–608, 2011.
[4] 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.
[5] V. Gupta et. al. IMPACT: Imprecise adders for low-power approximate computing. In *Proc. ISLPED 2011*, pages 409 –414, Aug. 2011.
[6] A. B. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *Proc. DAC*, pages 820–825, 2012.
[7] 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.
[8] P. Kulkarni et. al. Trading accuracy for power with an underdesigned multiplier architecture. In *Proc. VLSI Design*, pages 346 –351, Jan. 2011.
[9] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *Proc. DATE*, Mar. 2011.
[10] A. Lingamneni K. Palem et. al. Energy parsimonious circuit design through probabilistic pruning. In *Proc. DATE*, Mar. 2011.
[11] S. Venkataramani et. al. Salsa: systematic logic synthesis of approximate circuits. In *Proc. DAC*, pages 796–801, 2012.
[12] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Proc. DATE*, pages 957–960, Mar. 2010.
[13] L. Benini, E. Macii, and M. Poncino. Telescopic units: increasing the average throughput of pipelined designs by adaptive latency control. In *Proc. DAC*, pages 22–27, 1997.
[14] D. Baneres, J. Cortadella, and M. Kishinevsky. Variable-latency design by function speculation. In *Proc. DATE*, pages 1704 –1709, april 2009.
[15] S. Ghosh, S. Bhunia, and K. Roy. Crista: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation. *IEEE Trans. on CAD*, 26:1947–1956, nov. 2007.
[16] S. L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67 – 73, mar 2004.
[17] P. K. Krause and I. Polian. Adaptive voltage over-scaling for resilient applications. In *Proc. DATE*, pages 1 –6, March 2011.
[18] N. Zhu et. al. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE Trans. on VLSI systems*, 18(8):1225 –1229, aug. 2010.
[19] A. B. Kahng et. al. Slack redistribution for graceful degradation under voltage overscaling. In *Proc. ASP-DAC*, pages 825 –831, Jan. 2010.
[20] L. Wan et. al. CCP: Common case promotion for improved timing error resilience with energy efficiency. In *Proc. ISLPED*, pages 135–140, 2012.
[21] Design Compiler Ultra, Synopsys Inc.