Statistical Quality Modeling of Approximate Hardware

Seogoo Lee[1], Dongwook Lee[1], Kyungtae Han[2], Emily Shriver[2], Lizy K. John[1], and Andreas Gerstlauer[1]

[1]The University of Texas at Austin
{sglee,dongwook.lee,ljohn,gerstl}@utexas.edu

[2]Intel Corporation
{kyungtae.han,emily.shriver}@intel.com

Abstract— Beyond traditional bit truncation, recently proposed arithmetic and logic approximations have enriched the quality versus energy design space for custom hardware kernels in signal processing and other error-tolerant applications. Systematic exploration of such trade-offs requires fast, accurate, and generic quality-energy models that can drive datapath optimizations. Existing quality estimation approaches, however, are either based on slow simulation or limited in supported approximation types and quality metrics. In this paper, we propose a novel semi-analytical quality model that can predict a wide range of statistical metrics for arbitrary hardware approximations with deterministic error behavior. Input and error dependencies are captured using one-time error-free simulation only. Combining our quality estimation with a faithful energy model considering both switching activity and voltage scaling, we provide a complete quality-energy optimization flow. Optimization results for FFT and IDCT benchmarks show that our approach is 28x faster than purely simulation-based exploration, and 2x faster than existing hybrid approaches, all while achieving comparable estimation accuracy.

1. Introduction

Bit truncation and word length optimization have been widely used to exploit quality-energy tradeoffs in the design of signal processing kernels. Recently, approximate computing has emerged to exploit tradeoffs at finer granularity and for a wider range of error-tolerant applications [3]. At the hardware level, quality-configurable approximate arithmetic units have been proposed to design datapaths of custom hardware blocks [11, 14, 15, 19]. However, for systematic and controlled optimization and exploration of design spaces, generic, fast, and accurate quality and energy models are crucial.

Dating back to traditional fixed-point conversion, simulation-based or analytical approaches have been used for quality estimation. Simulation-based approaches are flexible and can compute an exact quality result for arbitrary bit truncation [4] or novel hardware approximations [2, 17]. These approaches, however, are often too time-consuming to be suitable for rapid synthesis. By contrast, analytical approaches are fast, but most of them have limitations in supported approximation types [9, 13] or quality metrics [7, 16]. The work in [13] incorperates a quality constraint in high-level synthesis, but only considers bit truncation. The authors in [9] only analyze relative impact of erroneous operations on output quality in graphs limited to addition and shift operations. In [7, 16], the authors use interval arithmetic, which provides safe minimum and maximum error boundaries, but is known to be conservative, leading to significant over-design when aiming to optimize systems for actual application-specific quality metrics, such as signal-to-noise ratios (SNRs). For fixed-point conversion, the work in [12] introduces affine arithmetic, but it is still a conservative range analysis. Only limited work exists overcoming these limitations. Recent approaches first characterize error behavior of individual hardware operators by simulation, and then analytically propagate either a derived quality metric [1] or input data and error probability mass functions (PMFs) [6]. These approaches work with various approximation types and quality metrics, but still depend on exhaustive simulation to characterize individual unit error behavior. Some work on approximate adders and multipliers has provided their own analytical error models [14, 19], but they simply assume uniformly distributed inputs, which leads to inherent inaccuracies.

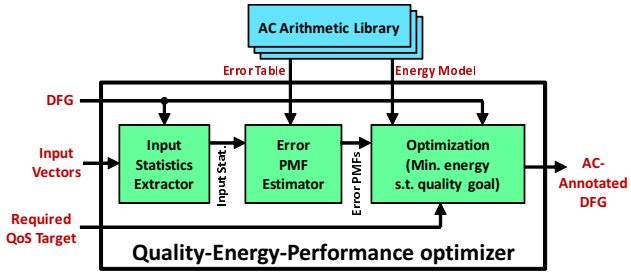


Fig. 1: Quality-energy optimization flow.

In this paper, we present a novel statistical quality analysis technique aimed at resolving drawbacks of previous work. We propose a fast and accurate semi-analytical error model for estimation of a wide range of statistical quality metrics under deterministic hardware approximations in arbitrary dataflow graphs (DFGs). Our approach is general in terms of quality metrics, DFGs, input statistics, and supported hardware approximations. It only requires one-time simulation-based profiling to capture dependency on input statistics. We analytically characterize the error PMFs of various operation units without any further simulation overhead, and this is a main difference from existing work in [1, 6]. The use of general PMFs allows for translation into a variety of quality metrics, such as SNR or min/max error. Quality metrics at primary DFG outputs are then estimated from individual error PMFs and an error propagation model. We combine our quality analysis with an energy model that captures dynamic power savings from switching activity and voltage scaling due to reduced logic complexity and critical path delays. We demonstrate application of our quality-energy analysis to drive optimization of approximations over an entire DFG.

Fig. 1 shows an overview of our analysis and optimization flow. To estimate error PMFs of all operations in a given DFG, we first find efficient representations of both input statistics and unit-level error behavior, which is a key to significantly reducing analysis complexity. Error behavior of approximate hardware units can generally be described as a truth table of erroneous outputs. However, the complexity is exponential in the

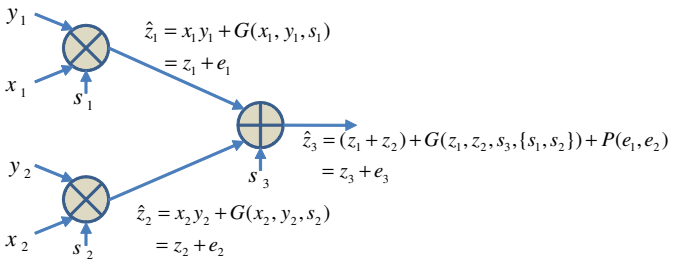


Fig. 2: Example DFG error model.

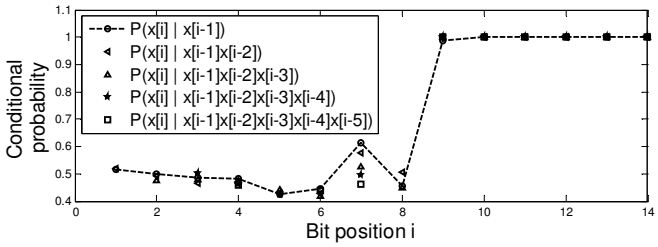


Fig. 3: Validation of conditional independence.

Fig. 4: Error table of a 2-bit LOA example.

number of approximated bits. To reduce this complexity, we propose two techniques to compactly represent 1) the error entries of an approximate arithmetic operation, and 2) the input statistics to the operation. Combined, these will allow error PMF estimation to be performed for all types and parameters of approximate operations.

In this paper, we limit the scope to arbitrary but deterministic logic-level hardware approximations in arithmetic operations of DFGs without feedback loops and conditional branches. Note that branching type of behavior can be incorporated by propagating errors across branches and assuming worst-case or probabilistic behavior at join nodes [10]. We further assume that inputs to an operation are independent.

The rest of the paper is organized as follows: In Section 2, we present details of our approach. In Section 3, we describe the optimization problem and the energy model. In Section 4, we then show our experimental results, while we conclude the paper with a summary and outlook on future work in Section 5.

2. Error analysis

We use an additive model to represent the error at the output of each arithmetic operation in a DFG. Such additive error models have been used in fixed-point quantization noise analysis [18]. The work in [1, 7] also adopts this model. Fig. 2 shows our error model for a simple multiplication and addition (MAC) example. Inputs, error, and output of each operation are random variables. In general, $\hat{z}_i$ is the erroneous output from the i-th operation, where $\hat{z}_i$ is the sum of error-free output z_i and an additive error e_i.

The error e_i is generally the sum of errors G generated within the operation and errors P propagated through the inputs from predecessor operations. G is a function of the error-free inputs (x_i and y_i) and the selected quality-scaling s_i. $P()$ will depend on the specific propagation model employed. Details about $G()$ and $P()$ will be given in Sections 2.3 and 2.4, respectively.

2.1. Representation of input statistics

One of the key contributions of this paper compared to existing work is that we consider input statistics, instead of, for example, simply assuming uniform distributions. In reality, error distributions depend on input patterns, and overall error statistics are highly input-dependent [1]. At the same time, a complete representation of input statistics for all possible patterns is exponential in complexity, requiring 2^{2n} entries for a 2-input n-bit operation.

In this work, we introduce a compact representation of input statistics in the form of 1) bit-wise probabilities, and 2) pair-wise equivalence probabilities of neighboring bits for each operand in the DFG. Bit-wise probabilities are represented as the probability $P_i^1 = P(x[i] = 1)$ of an individual bit i in operand x being one. We further use the pair-wise equivalence probabilities $P_i^= = P(x[i] = x[i-1])$ of two neighboring bits being equivalent. We compute those probabilities as well as mean and variance by input profiling.

A complete representation of input statistics would require pair-wise equivalence probabilities between any two bits of input operands to be considered. Reducing distributions into pair-wise equality/inequality relations between neighboring bits plus individual bit probabilities provides a good approximation under the assumption of conditional independence of non-neighboring bits. Fig. 3 shows the conditional probabilities of various bit positions in inverse discrete cosine transform (IDCT) image data given other operand bits. Except for bit position 7, bit probabilities are largely independent of non-neighboring bits, confirming our conditional independence assumption. Similar results are obtained for Gaussian and uniform distributions. As results will show, the difference in some bit positions (such as bit 7 in this case) is not a significant factor in our PMF estimation.

Nevertheless, this representation comes with several limitations. If input distributions are sparse, i.e. an operand only has a small set of possible patterns, correlations will extend beyond just neighboring bits. Such cases can instead be modeled as distinct operation types for each possible input value. Also, we assume that the two inputs for an operation are independent. If bits from different operands are highly correlated, it is also better to use a customized operation. An extreme example is a square operation representing a product of two identical inputs.

2.2. Modeling of unit-level error behavior

Since we only consider deterministic logic errors, we can generally represent errors $G(x_k, y_k, s_k)$ generated by an operation k realized in a specific hardware implementation s_k as a function $G^{s_k} : \{0,1\}^{n_x} \times \{0,1\}^{n_y} \mapsto \mathbb{N}$ that assigns an error value G in $\mathbb{N}$ to each x_k, y_k input combination. We assume that error tables G^{s_k} are provided in an approximate hardware unit library. A key observation is that many such arithmetic units are de-

TABLE I: Error table for truncated multiplication.

Implicant	Scale of y (e_x)	Scale of x (e_y)	Error (D)
$x'[1]x'[0]y[0]$	0	-1	0
$x'[1]x[0]y[0]$	-1	-1	1
$x[1]x'[0]y[0]$	-1	-2	2
$x[1]x[0]y[0]$	-1	-3	3
$x'[1]x[0]y'[0]$	0	-1	0
$x[1]x'[0]y'[0]$	0	-2	0
$x[1]x[0]y'[0]$	0	-3	0

signed by approximating only the lower significant bits (LSBs) [15]. This allows G to be reduced to only those input bits that result in non-zero errors. Nevertheless, the number of entries in each error table G^{s_k} is still exponential in the number of approximated bits n_x and n_y. We propose an efficient representation of G^{s_k}. Fig. 4 shows an example G^{LOA} for a 2-bit lower-bit OR adder (LOA) [14]. There are seven error entries. Finding a minimal representation of this error table is a covering problem similar to two-level logic synthesis. In this example, error terms of magnitude 3 can be covered by a prime implicant $x[1]x[0]y[1]y[0]$, error terms of magnitude 2 by implicants $x[1]y[1]y'[0]$ and $x[1]x'[0]y[1]y[0]$, and magnitude 1 errors by implicants $x[0]y'[1]y[0]$ and $x'[1]x[0]y[1]y[0]$.

For approximate multipliers, the error is generally not a function of a reduced set of input bits even if only LSBs are approximated. For example, in a multiplier that truncates inputs x and y with errors e_x and e_y, respectively, the output becomes $(x+e_x)(y+e_y) = xy + xe_y + ye_x + e_xe_y$. Here, the last three terms are error terms. The total error depends on e_x and e_y, which are determined not only by how LSBs are approximated, but also by inputs x and y. Thus, even though we only approximate LSBs, to compute accurate errors, we need to know complete distributions of inputs. Furthermore, this requires the multiplier's complete truth table to be known, which is practically infeasible. To avoid the complexity issue, we instead extend error tables stored in the hardware library with additional entries e_x and e_y that indicate the weight factors by which inputs y and x contribute to the error. Note that e_x and e_y again only depend on the number of approximated LSBs of x and y, i.e. can be stored in reduced form. Table I shows the representation for a multiplier with two LSBs of x and one LSB of y truncated, where the error for the x[1]x0[0]y[0] entry is $-x-2y+2$, for example. From this, we compute xe_y and ye_x terms assuming that x and y are uniformly distributed in a range determined by their means and variances, which we obtain from input profiling. Note that we only make this assumption when computing each entry's error value. An entry's probability is still calculated from bit-level input statistics. Our method is different from other approaches in [14, 19] that simply assume inputs being uniformly distributed across their full static value range independent of actual input statistics.

2.3. Operation-level error PMF estimation

Error tables are examined together with the input representation to determine the error PMFs for each DFG operation. Fig. 5 shows an example for a 2-bit LOA. In this example, input LSBs have high equivalence probabilities between neighboring bits, $x[1]$ and $x[0]$, and $y[1]$ and $y[0]$. In addition, bit 0 of x is mostly

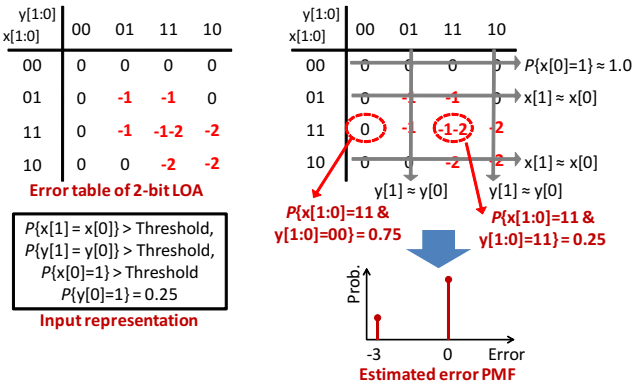


Fig. 5: Error PMF estimation.

1. We regard a bit as constant if a bit-wise probability is higher than a certain threshold. Similarly, if a pair-wise equivalence or inequivalence is higher than the threshold, the two bits are assumed to be always the same or always different. Both thresholds are 0.99 in our experiments. With these input statistics, we can prune out 6 error entries in the error table, and only 1 error entry (error value = -3 with probability 1/4) actually affects the output error. This result is quite different from assuming uniformly-distributed input operands.

We generally compute the probabilities of surviving error entries after pruning using previously gathered input statistics. Known input probabilities P_i^1, P_{i-1}^1, and $P_i^=$ can be decomposed into pair-wise joint probabilities of neighboring bits as follows:

$$
\begin{aligned}
P_i^1 &= P(\{x[i],x[i-1]\}=10)+P(\{x[i],x[i-1]\}=11)\\
P_{i-1}^1 &= P(\{x[i],x[i-1]\}=01)+P(\{x[i],x[i-1]\}=11)\\
P_i^= &= P(\{x[i],x[i-1]\}=11)+P(\{x[i],x[i-1]\}=00)\\
1-P_i^= &= P(\{x[i],x[i-1]\}=01)+P(\{x[i],x[i-1]\}=10).
\end{aligned}
$$

Solving the system of equations, we calculate the pair-wise joint distribution, $P(x[i],x[i-1])$ as:

$$
\begin{aligned}
&P(x[i]=a,x[i-1]=b) =\\
&1/2\times((2a-1)P_i^1+(2b-1)P_{i-1}^1+(2c-1)P_i^=-2ab+1),
\end{aligned}
\tag{1}
$$

where $a,b\in\{0,1\}$, and $c=a\oplus b$ is the exclusive-or of a and b. Given the proposed input and error representations, we can estimate the error PMF at individual arithmetic operation outputs. The algorithm for computing operation-level PMFs generally uses input statistics to compute the probability of each implicant in the operation's error table. For an n_x and n_y bit approximation, the probability of j-th error implicant, $P(m_j)$, is the joint probability of all involved bits. We calculate this probability from bit-wise probabilities and pair-wise joint probabilities in (1) based on an assumption of conditional independence of non-neighboring bits:

$$
\begin{aligned}
P(m_i) &= P(y[n_y-1],\cdots,y[0],x[n_x-1],\cdots,x[0]) =\\
&P(y[1],y[0])\frac{P(y[2],y[1])}{P(y[1])}\cdots\frac{P(y[n_y-1],y[n_y-2])}{P(y[n_y-2]}\\
&\times P(x[1],x[0])\frac{P(x[2],x[1])}{P(x[1])}\cdots\frac{P(x[n_x-1],x[n_x-2])}{P(x[n_x-2]}.
\end{aligned}
\tag{2}
$$

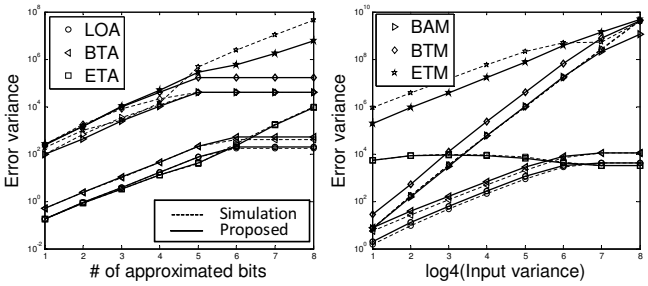
Algorithm 1 Error PMF Estimation

procedure BwChk(Bit,$Prob$,th)
  if $(Bit = 1)\&(Prob < 1-th)$ **then return** true
  if $(Bit = 0)\&(Prob > th)$ **then return** true
  return false
procedure PwChk(Bit_{u},Bit_{l},$Prob$,th)
  if $(Bit_{\mathrm{u}} = Bit_{\mathrm{l}})\&(Prob < 1-th)$ **then return** true
  if $(Bit_{\mathrm{u}} != Bit_{\mathrm{l}})\&(Prob > th)$ **then return** true
  return false
procedure PMF(OpType,$\mathbf{M}$,$(^{min_{\mathrm{x}}}_{max_{\mathrm{x}}})$,$(^{min_{\mathrm{y}}}_{max_{\mathrm{y}}})$,$\mathbf{P}_{\mathrm{x}}$,$\mathbf{P}_{\mathrm{y}}$)
  Initialize $PMF[\,]$
  $EP_{\mathrm{total}} = 0$
  for all implicants $m_j \in \mathbf{M}$ **do**
    $r =$ false
    for all bit position $i \in m_j$ **do**
      if BwChk$(x[i], P^{1}_{\mathrm{x[i]}}, 0.99)$ **then** $r =$ true
      if BwChk$(y[i], P^{1}_{\mathrm{y[i]}}, 0.99)$ **then** $r =$ true
      if PwChk$(x[i], x[i-1], P^{=}_{\mathrm{x[i]}}, 0.99)$ **then** $r =$ true
      if PwChk$(y[i], y[i-1], P^{=}_{\mathrm{y[i]}}, 0.99)$ **then** $r =$ true
    if $r =$ true **then continue**
    if OpType != Multiplication **then**
      $PMF[D(m_j)] += P(m_j)$
    else
      $min = \min_{\substack{x\in\{min_{\mathrm{x}}, max_{\mathrm{x}}\}\\ y\in\{min_{\mathrm{y}}, max_{\mathrm{y}}\}}} (xe_{\mathrm{y}}(m_j) + ye_{\mathrm{x}}(m_j) + D(m_j))$
      $max = \max_{\substack{x\in\{min_{\mathrm{x}}, max_{\mathrm{x}}\}\\ y\in\{min_{\mathrm{y}}, max_{\mathrm{y}}\}}} (xe_{\mathrm{y}}(m_j) + ye_{\mathrm{x}}(m_j) + D(m_j))$
      for all $min < k < max$ **do**
        $PMF[k] += P(m_j)/(max - min - 1)$
    $EP_{\mathrm{total}} += P(m_j)$
  $PMF[0] = 1 - EP_{\mathrm{total}}$

Algorithm 1 summarizes the computation of error PMFs per operation. For each implicant m_j in the operation's error implicant set $\mathbf{M}$, we prune unlikely patterns using input statistics $\mathbf{P}_{\mathrm{x}}$ and $\mathbf{P}_{\mathrm{y}}$ from initial profiling. For each surviving m_j, we then calculate its error magnitude $D(m_j)$ and probability $P(m_j)$. In case of multipliers, we assume that errors are uniformly distributed over the range of the implicant's error distribution, and we compute the minimum (min) and maximum (max) of m_j's error range from parameters e_{x}, e_{y}, D and the (min_{x} ... max_{x}) and (min_{y} ... max_{y}) ranges of uniform input operand distributions extracted during initial profiling. Using this information, we construct a histogram of error magnitudes and their probabilities. Finally, we calculate the zero error probability such that the total probability is 1.

For 8-bit approximation of LOA, our results show that the Hellinger distance between estimated and simulated error PMFs is less than 1%. Furthermore, using our compact error table representation, we can reduce the number of implicants and corresponding error PMF estimation time by 10x as compared to an exhaustive analysis.

In general, for a certain type of approximation, error distributions depend on two factors: 1) input statistics, and 2) approximation parameters, such as the number of lower bits for an LOA. Fig. 6 shows comparisons of estimated and simulated error variances under different number of approximated bits and different input statistics for an LOA, a bit truncation adder (BTA), an error-tolerant adder (ETA) [19], a broken array multiplier (BAM) [14], a bit truncation multiplier (BTM), and an error-tolerant multiplier (ETM) [11]. When sweeping bits, we



(a) vs. # of approximated bits (b) vs. input variance

Fig. 6: Unit-level error variance estimation.

fix input variance to 1,000. Similarly, when sweeping variance, we fix the number of approximate bits to 8. Overall, results verify that our PMF estimation can track true error behavior across a general range of parameters and input statistics. The estimation error is larger (up to 6dB) for multipliers. This inaccuracy is mainly due to our reconstruction of input probabilities using a uniform distribution. Note, however, that we still consider actual input statistics $P(m_j)$ in calculations, and only use uniform distribution to determine how $P(m_j)$ is spread in a range between min and max. Existing analytical approaches [14] do not consider any input statistics and instead assume a uniform distribution across all bits. This generates error estimates with fixed variance, independent of the actual input distribution.

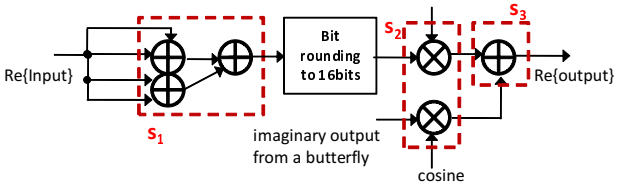
2.4. Error propagation

Several methods can be employed to propagate errors G across a DFG in order to estimate error terms P. In general, raw error PMFs can be propagated under assumptions of statistical independence. Alternatively, to reduce computational complexity, we can first extract various metrics, such as mean, variance, or minimum/maximum error from operation-level PMFs. These metrics can be used to apply existing efficient propagation methods, such as interval arithmetic [7] or variance propagation for SNR estimation [1].

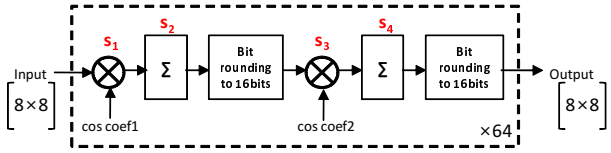
Another aspect to consider in error propagation is that errors at the output of an operation will affect input statistics of its following operation, which can in turn affect successor operations' error statistics. For example, bit truncation always changes the approximated bits to zeros. If this output is given to another bit truncation with a smaller number of approximated bits, the successor operation will not generate an error at all. By contrast, some approximate units such as the LOA do not significantly affect output bit probabilities. We model this aspect by distinguishing between approximations that mask output bits to zero or leave statistics unchanged.

3. Application-level optimization

We combine our quality model with an energy model, where total energy consumption is the sum of all operations' energy. Energy gains of approximations primarily stem from (a) reduced switching activity and (b) reduced critical path delays, which can be exploited for energy savings through voltage scaling. We synthesize approximate adders and multipliers using Synopsys Design Compiler to obtain the dynamic power numbers and critical path delays under different numbers of approx-



(a) A stage of FFT



(b) IDCT

Fig. 7: Example DFGs.

TABLE II: Experimental setup.

DFG	# of s_i	Input data	Qual. metric	Baseline hardware
FFT	10	$\mathcal{N}(\prime,\infty\prime^{\supset})$	SNR	16-bit ADD, 16-bit MUL
IDCT	4	Lena image	PSNR	32-bit ADD, 16-bit MUL

imated bits. Critical path delays are used to determine the minimum voltage the unit can operate at without causing timing violation [5]. In general, the relationship between voltage and delay will vary non-uniformly for different cells along a critical path. To convert delay reductions to voltage savings, we run Synopsys HSPICE simulations sweeping voltages for all the standard cells in the given library. We then estimate average voltage-delay relationship across all cells by function fitting.

Our energy model consists of two tables: dynamic energy E^s at normal operating voltage versus approximated configuration s, and voltage versus approximated bits, provided as a library V^s. We determine a uniform voltage for a complete DFG as the maximum among the voltages for all approximated operations in the graph. With this, the energy cost function becomes

$$E(\mathbf{s}) = \sum_{i=1}^{\mathrm{N}} E^{s_i} \times \frac{(\max(\mathbf{V^s}))^2}{V_{\mathrm{ref}}^2}. \quad (3)$$

Here, variables s_i correspond to the approximation type and scaling level of the i-th operation, $\mathbf{s} = \{s_1, s_2, ..., s_{\mathrm{N}}\}$, where N is the number of operations in the DFG. V_{ref} is the normal operating voltage, and $\mathbf{V^s} = \{V^{s_1}, V^{s_2}, ..., V^{s_{\mathrm{N}}}\}$.

We apply our quality and energy models to optimize energy consumption while satisfying a quality requirement at the primary output of a DFG. The inputs to our optimizer are estimated operation-level error metrics, energy models of all possible hardware units, an application DFG, and a required output quality goal ($\mathrm{Q_{min}}$). Formally, the optimization problem is

$$\min_{\mathbf{s}} E(\mathbf{s}), \text{ subject to } \mathrm{Q}(\mathbf{k},\mathbf{s}) > \mathrm{Q_{min}}. \quad (4)$$

Here, Q() is the application-specific quality metric computed as described in Section 2. The optimization problem is a non-linear, non-convex integer problem, and meta-heuristics can generally be used to solve the optimization.

4. Experiments and results

We use an adaptive simulated annealing (ASA) [8] meta-heuristic to solve the optimization problem. Although ASA

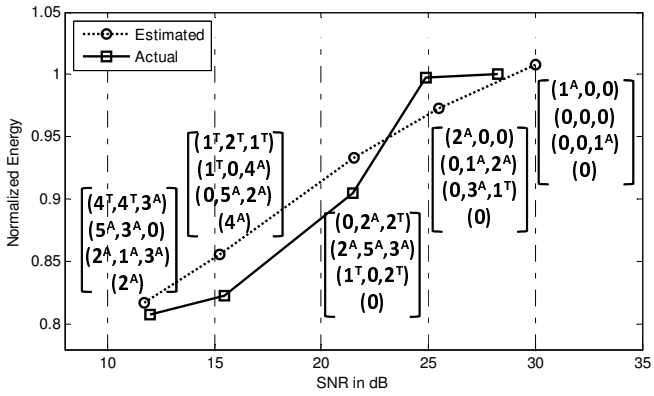


Fig. 8: FFT optimization result.

does not guarantee an optimal solution, an annealing-based approach is effective in solving this type of non-linear integer programming problem. We have applied our quality-energy analysis framework to two application examples. Even though our approach also works for other quality metrics, such as min/max error, we present results targeting SNRs for a 256-point, 4-stage pipelined FFT with radix-2^2 structure (Fig. 7a) and a 2-dimensional IDCT (Fig. 7b). Table II shows a summary of our experimental setup. Note that we allow up to 8 approximated bits for both FFT adders and multipliers, and 16 and 8 bits for IDCT adders and multipliers, respectively, as accuracy scaling scheme. We verify optimization results by synthesizing optimized designs down to a gate-level netlist using Synopsys Design Compiler with a 32/28nm generic standard cell library and vectorless power estimation.

Fig. 8 shows our FFT optimization results targeting different SNR levels from 10 dB to 30 dB in 5 dB increments, which is a typical operating range for wireless communication systems. For each SNR target, the annotated set of numbers shows the types and values of the decision variables obtained from the optimization for each FFT stage. A subscript of A denotes use of a LOA, ETA, BAM or ETM unit with selected number of approximated bits. Otherwise, T indicates use of a bit truncated implementation. Optimization results are plotted using both estimates from our analysis framework as well as actual PSNR and energy values obtained from simulation and synthesis, respectively. Power savings are normalized with respect to the accurate, non-approximated baseline FFT design with 32 dB SNR.

Up to 18% energy savings can be achieved at the lowest quality level. Optimized designs use a mix of approximate and traditional bit-truncated implementations, where lower quality goals lead to increased use of more aggressive bit truncation. Savings generally grow linearly with an increasing number of approximated bits. However, due to gate-level optimizations in Design Compiler, there are fluctuations in actual energy results, which lead to gaps with estimation. Nevertheless, we can observe that our quality-energy analysis tracks actual results to within a 1.75 dB SNR and 4% energy difference. With the exception of small estimation errors at high SNRs, results meet or exceed the originally targeted quality constraints.

Fig. 9 shows IDCT optimization results for peak signal-to-noise ratio (PSNR) targets from 30 dB to 45 dB with a 5 dB step size. The accurate, non-approximated reference design

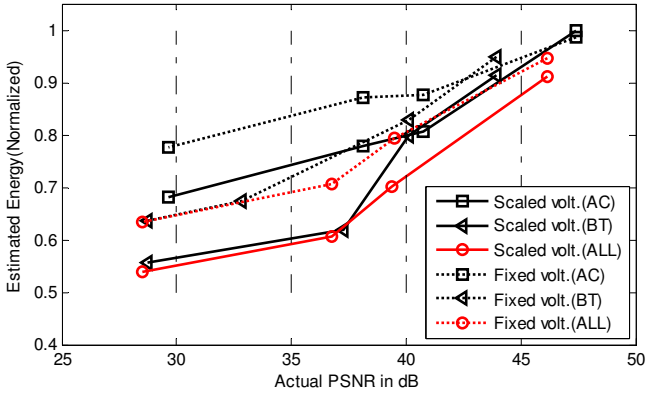
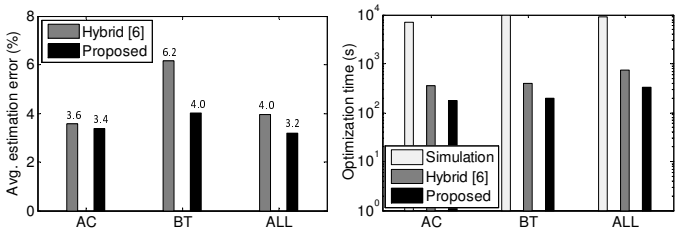


Fig. 9: IDCT optimization result.



(a) Average accuracy (b) Total runtime

Fig. 10: Comparisons for optimization time and accuracy.

has a PSNR of 48 dB. We show optimization results using only approximate LOA, ETA, BAM and ETM hardware resources ('AC'), only bit-truncated adders and multipliers ('BT'), or both ('ALL'). We execute final designs with each selected configuration to measure actual PSNR. All designs remain within 3 dB of the original PSNR goal. PSNR estimation errors are below 3.1 dB (9.1%) and the average error is 1.3 dB (3.5%).

We show estimated energy costs with and without consideration of voltage scaling. Without voltage scaling, energy savings can reach 39% in the given PSNR range. The critical path delay reduction is up to 5.6%, which increases energy savings to 46% when scaled. Even though not the main goal of this paper, results confirm the savings potential of approximate computing.

We compare PSNR estimation accuracy (Fig. 10a) and optimization time (Fig. 10b) using our quality analysis versus a simulation-based exploration and the hybrid error estimation approach from [6], which combines exhaustive operation-level simulation with analytical propagation. To compute accuracy, we average differences between the simulated and estimated PSNRs of the final solutions from optimizing for 4 different target PSNRs. Fig. 10a shows that our approach has an up to 4.0% estimation error, while the hybrid approach differs by up to 6.2%. As such, overall estimation is comparably accurate.

The total runtime for IDCT exploration of 4 target PSNRs (Fig. 10b) is less than 6 minutes with our approach. This is about 28× faster compared to simulation-based exploration, which takes more than 170 minutes. It is also 2× faster than using the hybrid model. The total runtime for the FFT exploration across 5 SNR levels is less than 20 seconds. This includes 3 s for one-time simulation to collect input statistics, 14 s to pre-compute error PMFs for all operations and accuracy levels, and less than 3 s on average per ASA run. By contrast, an exploration and optimization using a simulation-based approach takes around 1 hour. This is more than 180× slower than our approach. All experiments were performed on an Intel Core i7 machine running at 2.67GHz.

5. Summary and conclusions

In this paper, we presented a novel approach for statistical quality-energy optimization of dataflow graphs. The quality estimation is fast and accurate requiring only one-time profiling to capture error and input dependencies. Our proposed method can be used for any type of hardware approximations that cause deterministic errors. To reduce the quality analysis complexity, we propose compact input and error representations. Quality models are combined with energy estimates to drive optimizations across an entire DFG. Results show that quality-energy explorations across a range of input statistics, approximation methods and quality goals can be performed with high accuracy and speed. In future work, we will focus on extending our approach to integrate control flow analysis and other, non-deterministic or random error sources.

Acknowledgments

This work was partially supported by Intel and NSF Grant CCF-1018075.

References

[1] W.-T. Chan et al. Statistical analysis and modeling for error composition in approximate computation circuits. In *ICCD*, 2013.
[2] V. Chippa et al. Analysis and characterization of inherent application resilience for approximate computing. In *DAC*, 2013.
[3] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *ETS*, 2013.
[4] K. Han and B. L. Evans. Optimum wordlength search using sensitivity information. *EURASIP Journal on Advances in Signal Processing*, 2006(1):1–14, 2006.
[5] K. He et al. Controlled timing-error acceptance for low energy IDCT design. In *DATE*, 2011.
[6] J. Huang et al. A methodology for energy-quality tradeoff using imprecise hardware. In *DAC*, 2012.
[7] J. Huang and J. Lach. Exploring the fidelity-efficiency design space using imprecise arithmetic. In *ASP-DAC*, 2011.
[8] L. Ingber. ASA 25.15. `http://www.ingber.com//#ASA`, 2004.
[9] Z. Kedem et al. Optimizing energy to minimize errors in dataflow graphs using approximate adders. In *CASES*, 2010.
[10] K.-I. Kum and W. Sung. Combined word-length optimization and high-level synthesis of digital signal processing systems. *TCAD*, 20(8):921–930, 2001.
[11] K. Y. Kyaw et al. Low-power high-speed multiplier for error-tolerant application. In *EDSSC*, 2010.
[12] D.-U. Lee et al. Accuracy-guaranteed bit-width optimization. *TCAD*, 25(10):1990–2000, 2006.
[13] C. Li et al. Joint precision and high level synthesis for approximate computing. In *DAC*, 2015.
[14] H. Mahdiani et al. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *TCAS I*, 57(4):850–862, 2010.
[15] J. Miao et al. Modeling and synthesis of quality-energy optimal approximate adders. In *ICCAD*. ACM, 2012.
[16] S. Misailovic et al. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. In *OOPSLA*, 2014.
[17] K. Nepal et al. ABACUS: A technique for automated behavioral synthesis of approximate computing circuits. In *DATE*, 2014.
[18] B. Widrow and I. Kollár. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press, 2008.
[19] N. Zhu et al. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *TVLSI*, 18(8):1225–1229, 2010.