Fine Grain Precision Scaling for Datapath Approximations in Digital Signal Processing Systems

Seogoo Lee, Andreas Gerstlauer

Fine Grain Precision Scaling for Datapath Approximations in Digital Signal Processing Systems

Seogoo Lee and Andreas Gerstlauer

The University of Texas at Austin
Department of Electrical and Computer Engineering,
Austin, TX, USA
{sglee,gerstl}@utexas.edu

Abstract. Finding optimal word lengths in digital signal processing systems has been one of the primary mechanisms for reducing complexity. Recently, this topic has been explored in a broader approximate computing context, where architectures allowing for fine-grain control of hardware or software accuracy have been proposed. One of the obstacles for adoption of fine-grain scaling techniques is that they require determining the precision of all intermediate values at all possible operation points, making simulation-based optimization infeasible. In this chapter, we study efficient analytical heuristics to find optimal sets of word lengths for all variables and operations in a dataflow graph constrained by mean squared error type of metrics. We apply our method to several industrial-strength examples. Our results show a more than 5,000x improvement in optimization time compared to an efficient simulation-based word length optimization method with less than 10% estimation error across a range of target quality metrics.

Keywords: Power reduction, Approximate computing, Word length optimization, Digital Signal Processing

1 Introduction

Scaling internal system precision continues to be one of the most important mechanisms to reduce implementation complexity and hence improve performance and power consumption in digital signal processing (DSP) systems. Traditionally, fixed-point word lengths of custom hardware or software implementations are set to match worst-case operating conditions determined at design time. In more aggressive recent power saving methods, a system's precision is dynamically and adaptively configured in reaction to changing operating points while maintaining a certain signal quality level [7, 3, 9, 10]. Such dynamic precision scaling mostly targets application-specific hardware implementations [7, 3, 10], although software implementations also exist [9]. Precision scaling can be realized by bit-level clock gating of the least significant bits as shown in Fig. 1. The maximum number of bits for any operation of an application should be enough to support worst-case conditions, but some of the lower bits can remain inactive for different operating points. At a constant clock rate and hence performance, a power reduction comes from the reduced activity of the gated sequential and associated combinational cells.

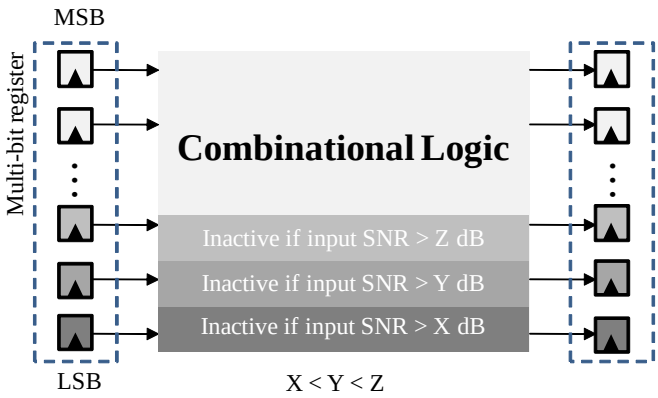

Fig. 1: A realization example of precision scaling

More recently, similar scaling of precision has been considered in the broader context of approximate computing (AC) [22]. Approximate computing generally exploits tradeoffs between computational complexity and energy or performance at various levels ranging from algorithms to transistors. Recent work in this area has been focused on precision-scalable software realizations on general-purpose programmable approximate processors [19, 20]. Similar to traditional fixed-point hardware design, word length of computations in the underlying hardware is thereby flexibly controlled through source-level program annotations and instruction set extensions.

In all cases, there is a significant burden on designers or programmers to carefully control precision of individual variables and operations in order to maximize energy savings while meeting overall application-level quality goals. Given the large number of variables and operating points to optimize for, traditional simulation-based approaches are too time-consuming. This is a main reason why fine-grain precision scaling is not widely employed, where designers and programmers instead resort to conservative worst-case analysis, self-adjusting runtime schemes [7, 3], which come with additional overhead, or coarse-grain approaches that assume the same precision or a limited number of precision levels across the whole design, which leaves additional power saving opportunities through fine-tuning of each variable's word length unexploited.

By contrast, in this chapter, we investigate novel analytical techniques that resolve some of the drawbacks of previous work. Our method calculates the optimal set of word lengths at design time using statistical analysis [23]. Compared to the methods that require long simulations, our approach can dramatically reduce the design time. Moreover, fine tuning of word lengths with low overhead improves power consumption compared to coarse-grain or run-time precision determination. At the core of our precision optimization technique is an accurate and fast error estimation capability, which, when coupled with a similar power model, provides the basis for developing corresponding optimization formulations and algorithms.

The chapter is organized as follows: after a brief summary of the related work in Section 2 and the problem definition in Section 3, our error (Section 4) and power

(Section 5) models at the basis of our optimization formulation are described. Results as applied to several design examples are shown in Section 6. Finally, Section 7 concludes and discusses future work.

2 Related Work

Fixed-point conversion and word length optimization has a long history of research dating back to [1] and [4]. In a fixed-point representation, there are basically two considerations in determining a word length, the number of integer bits and the number of fractional bits. Integer bits are related to the dynamic range of a signal, where an insufficient integer bit width causes saturation errors. By contrast, fractional bits are related to the precision and introduce quantization errors.

For determining the optimal number of both integer and fractional bits, analytical or simulation-based methods have been introduced. Simulation-based methods are widely used to estimate fixed-point performance. For example, Sung et al. [11] add a signal-to-quantization noise ratio (SQNR) block to quantify the finite word-length effects when the word-length in the implementation of the system changes. In [12], various word length search methods are summarized and compared. The efficiency of simulation-based methods is analyzed to determine the number of simulations needed to reach optimum word lengths. The complexity of a full search is $O(N^s)$, where N is the number possible word lengths for each decision variable, and s is the number of variables. It is shown that the complexity can be dramatically reduced down to $O(s)$ by using efficient search methods that rely on sensitivity information but may run into local minima.

Among the various analytical techniques, the authors in [21] show that optimally determining the number of fractional bits in linear time invariant (LTI) systems is a NP-hard problem. In [15], the authors adopt affine arithmetic (AA) to model the min/max error propagation of quantization noise. However, static min/max approaches are not appropriate for fine-grain precision scaling. They are known to be overly conservative. Furthermore, in applications such as communication systems, additive white Gaussian noise (AWGN) sources from the outer channel environment are hard to characterize in a min/max form. The research done by Shi and Brodersen [16] analyzes quantization noise with perturbation theory instead. They measure the sensitivities of input word lengths to output noise by simulations and use this information in their constraint function. Constantinides et al. find optimal word lengths by evaluating the variance of quantization noise through the system transfer function and subsequently formulating the optimization as a mixed integer linear programming (MILP) problem [13]. Finally, in [17] a variance propagation method is applied to quantization noise analysis in a fast Fourier transform (FFT) block, whereas Menard et al. [14] propose a similar method for generalized dataflow graphs. Because their method can be used both for linear and nonlinear systems and is suitable for general DSP applications, we adopt it for statistical word length analysis and optimization in this chapter.

On the application side, the authors of [7] introduce the concept of dynamic precision scaling for power savings in wireless communication systems by forcing lower significant bits to zero if the current signal quality is better than a predetermined minimum requirement. There are two drawbacks in their work: the authors use the same word

length across the whole design and their method requires dedicated training symbols to find the best word lengths at run time in a self-adjusting scheme, which introduces additional overhead that negates some of the power savings. In [3], the authors use both precision and voltage scaling to maximize power reduction. They first optimize word lengths according to the channel environment and then use these word lengths to find the optimal voltage that still satisfies a required error rate. Their method is robust to process variation, but it also incurs run time overhead. In [9] and [10], word lengths are optimized at design time to avoid the run time overhead. In [9], the authors target software-defined implementations of wireless systems, and use simulations to support fine-grain optimization of all variables but only consider powers of 2 as word lengths. In [10], optimal word lengths are also determined by slow simulation, where precision is instead allowed to decrease when it can be absorbed in increasing base noise under varying bit error requirements.

Finally, in the approximate computing domain, the authors in [19] propose Java-based language extensions to support programmer annotations for specifying variables and computations that can be approximated on top of an underlying architecture that supports such approximations. The work in [20] proposes such instruction-set and micro-architecture extensions that enable operation-level precision scaling for energy savings in a computation-oriented vector processor. In all of these cases, however, source- or binary-level annotations have to be manually determined by the programmer using other optimization techniques.

3 Problem Formulation

Our approach applies to world length optimization of fractional bits in DSP datapaths. As such, we assume that the number of integer bits has already been determined by a range analysis. We only consider systems with fixed-point number representations. Since floating-point computations are more accurate but come with a large complexity, fixed-point computations are still preferred in many energy-constrainted applications.

Applications are inherently error-tolerant DSP systems, which can have input signal noise and allow for a certain amount of additional error noise at their primary outputs. These type of systems exists, for example, in wireless communications, image/video processing and machine learning. We assume that applications are represented as dataflow graphs (DFGs) of addition and multiplication operations. Many data-dominated, regular DSP applications such as filters, transforms, or general matrix computations fall into this category. We further limit our target applications to systems that can be characterized by mean squared error (MSE) quality metrics, such as a given or desired signal-to-noise ratio (SNR) or peak SNR (PSNR) at primary inputs and outputs.

Our word length optimization procedure is shown in Fig. 2. Our optimization problem is to minimize power consumption subject to output quality constraints under given input and quantization noises. We start by building a floating-point model of our system, which is simulated to obtain a set of target output SNRs and a set of possible input SNRs, which form the operating points of an application. These are the inputs to our optimization problem. From the floating-point model, we also extract the DFG of the system. With the DFG, we build power cost and quality constraint functions, which

are functions of an operating point and word lengths. Then, we solve our optimization problem in order to determine an optimal set of word lengths that minimizes power consumption while satisfying the output quality requirement of a chosen operating point. This process is repeated for all possible operating points.

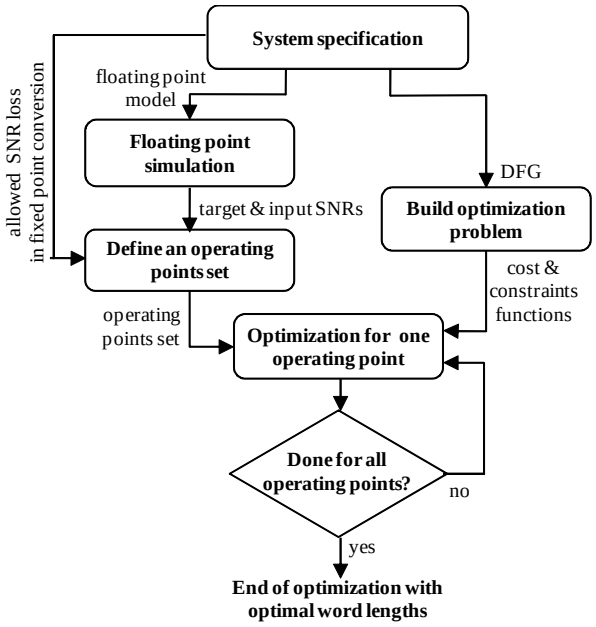

Fig. 2: Optimization procedure

Fig. 3 shows the conceptual change in the output quality and power consumption that results from precision scaling. An operating point is defined by a possible input quality and a required minimum output quality. In the example of Fig. 3, we assume that a single output quality goal must be met across multiple possible input SNRs. Scaling reduces the precision to minimize power consumption such that a constant targeted output quality is maintained for any input condition that would otherwise lead to a better-than-required quality. Note that the same concept can be applied to systems with just a single better-than-worst-case input SNR or with multiple output quality goals for different input conditions.

In general, the system can have one or more possible operating points or operating scenarios. An operating point is defined by a given input quality and a required output quality. Possible operating points of a system are combinations of a set of N possible input conditions $\mathbf{\Phi} = \{\mathrm{Q}_{\mathrm{in},1}, \mathrm{Q}_{\mathrm{in},2}, ..., \mathrm{Q}_{\mathrm{in,N}}\}$ and a set of M desired output quality goals $\mathbf{\Psi} = \{\mathrm{Q}_{\mathrm{out},1}, \mathrm{Q}_{\mathrm{out},2}, ..., \mathrm{Q}_{\mathrm{out,M}}\}$. We only scale precision when the output quality of the system under the current input condition is larger than a current output quality goal, i.e. when there is room for energy reduction by injecting additional quantization

errors. For all other operating points, the application will be configured to work at full precision, i.e. in a best effort manner.

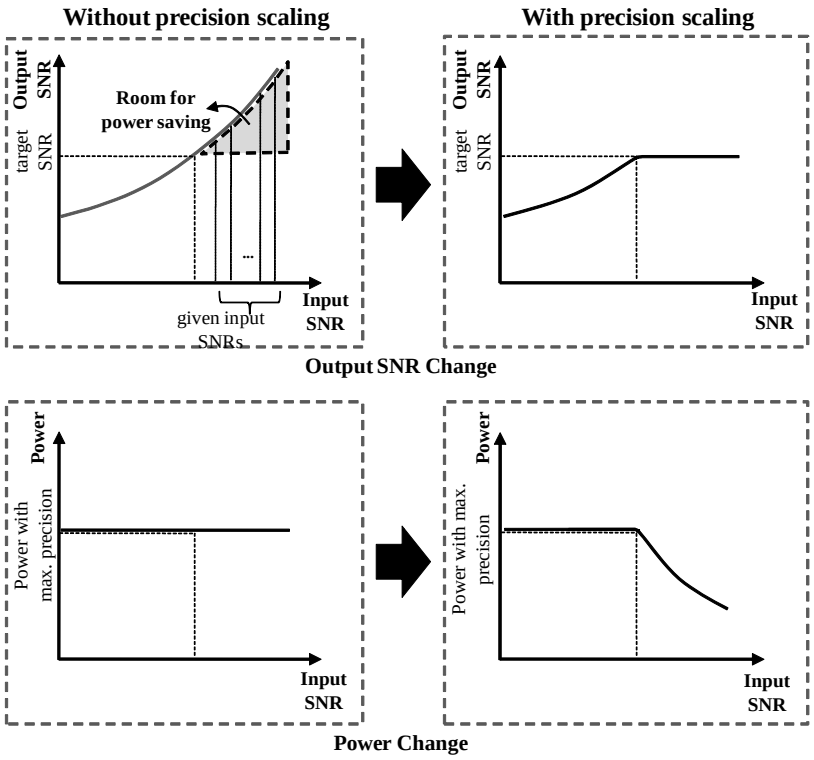

Fig. 3: Quality and power change by precision scaling

The decision variables of the optimization problem are the word lengths of the scalable fixed-point variables in the DFG of the application. Assume that the set $\mathbf{F}$ of K decision variables is:

$$\mathbf{F} = \{F_1, F_2, \ ..., \ F_{\mathrm{K}}\}. \tag{1}$$

In our statistical analysis, we first find the maximum word length of each variable at the worst-case operating point, i.e. where the application should work at its fullest precision. The hardware needs to be designed to support these worst-case word lengths. For a software implementation on a general-purpose processor, the full precision is the maximum precision of the adders and multipliers in the processor's datapath. These word lengths define the upper bound of each decision variable as follows:

$$0 < F_i \leq \mathrm{F}_{\mathrm{MAX},i}, \tag{2}$$

where $\mathrm{F}_{\mathrm{MAX},i}$ is the maximum word length for the i-th decision variable F_i.

We solve different optimization problems for different operating points $(\mathrm{Q}_{\mathrm{in},n}, \mathrm{Q}_{\mathrm{out},m})$. Each optimization problem becomes:

$$\min_{\mathbf{F}} \; P(\mathbf{F}) \tag{3}$$

subject to

$$N(\mathrm{Q}_{\mathrm{in},n}, \mathbf{F}) \leq \mathrm{Q}_{\mathrm{out},m}, \tag{4}$$

$$0 \leq F_i \leq \mathrm{F}_{\mathrm{MAX},i}, \; \forall i, \tag{5}$$

where $P(\mathbf{F})$ and $N(\mathrm{Q}_{\mathrm{in},n}, \mathbf{F})$ represent power and noise models to estimate implementation cost and output quality as a function of word lengths and input conditions. Corresponding fast yet accurate analytical noise and power models are at the core of our statistical optimization and will be described in the following sections.

We introduce our models on two examples, an FFT in an orthogonal frequency division multiplexing (OFDM) wireless communication system and an inverse discrete cosine transform (IDCT) in JPEG decoder. The FFT example can operate under multiple different input conditions depending on the external channel conditions. Similarly, the IDCT example can have multiple operating points as defined by the compression rate in the encoder. The final results of applying our optimization to both examples will be given in Section 6.

4 Noise Model

The output quality is determined by the input noise and the injected noise from word length scaling, the latter being in the form of quantization noise. Our approach is heuristic because 1) we use a pseudo quantization noise (PQN) model [17] instead of exact distribution functions of noise, and 2) we consider the quantization of system coefficients, such as twiddle factors or filter coefficients as additive noise injection, which ignores associated changes of the transfer function. In contrast to other approaches [17], This allows us to consider coefficient quantization noise. The impact on the transfer function and corresponding inaccuracies in our method are, however, specific to a given application.

We assume that quantization noise sources as well as input noise sources are independent. It is well known that we can get the variance σ^2 after addition and multiplication of two independent random variables as follows:

$$\text{Addition: } \sigma^2 = \sigma_1^2 + \sigma_2^2 \tag{6}$$

$$\text{Multiplication: } \sigma^2 = \mu_1^2\sigma_2^2 + \mu_2^2\sigma_1^2 + \sigma_1^2\sigma_2^2, \tag{7}$$

where μ_i is the expectation and σ_i^2 is the variance of input random variable i. The output variances after subtraction or division are also available in a similar way.

Quantization noise is modeled as additive noise. If we add quantizers to two independent inputs of an adder, s1 and s2 with signal variances σ_{s1}^2 and σ_{s2}^2, respectively, we add noise sources σ_{n1}^2 to σ_{s1}^2 and σ_{n2}^2 to σ_{s2}^2. At the output of the addition, the noise and signal variances therefore become $\sigma_{\mathrm{n}}^2 = \sigma_{\mathrm{n1}}^2 + \sigma_{\mathrm{n2}}^2$ and $\sigma_{\mathrm{s}}^2 = \sigma_{\mathrm{s1}}^2 + \sigma_{\mathrm{s2}}^2$, respectively. Hence, the total variance is $\sigma^2 = \sigma_{\mathrm{s}}^2 + \sigma_{\mathrm{n}}^2 = \sigma_{\mathrm{s1}}^2 + \sigma_{\mathrm{s2}}^2 + \sigma_{\mathrm{n1}}^2 + \sigma_{\mathrm{n2}}^2$.

4.1 Noise Analysis for one Decision Variable

We use a pipelined 256-point FFT with four series-connected radix-4 stages to derive and demonstrate our noise model. The overall FFT structure and the DFG for one stage of the FFT are shown in Fig. 4. The first two additions are for the butterfly and the following multiplier and adder represent the complex twiddle multiplication. Only calculation of the real phase is shown. The same computation is performed for the imaginary phase. There can be two quantization points in each stage of the FFT that affect the number of fractional bits: quantization of the input and of the twiddle factor. For simplification, we use a single word length for those two quantization points.

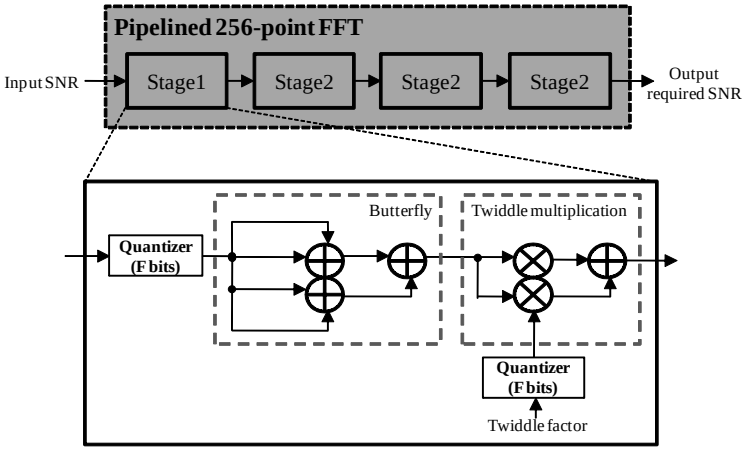

Fig. 4: The pipelined 256-point FFT structure and DFG for one radix-4 stage

The input to the FFT can be modeled as a sum of the error-free input with variance $\sigma^2_{\mathrm{s_{in}}}$ and additive input noise with variance $\sigma^2_{\mathrm{n_{in}}}$. Furthermore, the variance of quantization noise with F fractional bits and uniform distribution under a round-to-nearest rounding is $\sigma^2_{\mathrm{n_{quan}}} = \frac{1}{3}2^{-2F-2}$. The variance at the output of the butterfly is then also a sum $\sigma^2_{\mathrm{butterfly}} = \sigma^2_{\mathrm{s_{butterfly}}} + \sigma^2_{\mathrm{n_{butterfly}}}$ of the error-free output variance $\sigma^2_{\mathrm{s_{butterfly}}} = 4\sigma^2_{\mathrm{s_{in}}}$ and the noise variance $\sigma^2_{\mathrm{n_{butterfly}}} = 4(\sigma^2_{\mathrm{n_{in}}} + \sigma^2_{\mathrm{n_{quan}}}) = 4\sigma^2_{\mathrm{n_{in}}} + \frac{1}{3}2^{-2F}$, including input quantization noise.

The butterfly output becomes the input to twiddle multiplication. The other input, the twiddle factor is a sum of the ideal, sinusoidal twiddle factor with variance $\sigma^2_{\mathrm{s_{twiddle}}} = 1/8$ (for a 4-stage, 256-point FFT) and additive quantization noise $\sigma^2_{\mathrm{n_{quan}}}$. We assume that all signals and variables have zero mean ($\mu = 0$) and that $\sigma^2_{\mathrm{s_{in}}}$ is gain-controlled to 1. Therefore, the output variance after one FFT stage is

$$\sigma^2_{\mathrm{out}} = 2(\sigma^2_{\mathrm{s_{butterfly}}} + \sigma^2_{\mathrm{n_{butterfly}}})(\sigma^2_{\mathrm{s_{twiddle}}} + \sigma^2_{\mathrm{n_{quan}}}) \tag{8}$$

$$\approx 1 + \sigma^2_{\mathrm{n_{in}}} + (\frac{2}{3} + \frac{2}{3}\sigma^2_{\mathrm{n_{in}}} + \frac{1}{12})2^{-2F} \tag{9}$$

$$= 1 + \sigma^2_{\mathrm{n_{out}}} \tag{10}$$

With this, we can use the allowed performance loss in the floating- to fixed-point conversion process for our optimization. For example, if the allowed performance loss in SNR is 0.2dB for fixed-point conversion, given an input SNR of 11.6dB and the signal power of 1 ($\sigma^2_{\mathrm{n_{in}}} = 0.069$), the minimum word length to get a 11.4dB output SNR ($\sigma^2_{\mathrm{n_{out}}} = 0.072$) becomes $F = 5$. We can find the same value through simulation. By contrast, min/max propagation using affine arithmetic [15] with a $\pm 3\sigma$ min/max of the input AWGN would result in $F = 9$. This reflects that static analysis with min/max propagation is conservative.

4.2 Extension to Multiple Stages

We extend the above analysis to an FFT with multiple stages and hence decision variables. The output noise from the first stage becomes the input noise to the second stage and propagates through the whole FFT. At the end of the FFT, a noise constraint function can be represented as a function of the variance of the input noise ($\sigma^2_{\mathrm{n_{in}}}$) and a set of word lengths F_i for each stage i. For our four-stage FFT example, the output noise variance from the first stage is a function of $\sigma^2_{\mathrm{n_{in}}}$ and F_1 as shown in the previous subsection:

$$\sigma^2_{\mathrm{n_{out},1}} = f(\sigma^2_{\mathrm{n_{in}}}, F_1), \tag{11}$$

where $f()$ is defined as

$$f(\sigma, F) = \sigma + (\frac{2}{3} + \frac{2}{3}\sigma + \frac{1}{12})2^{-2F}. \tag{12}$$

Similarly, the output noise variance from the i-th stage ($i > 1$) can be formulated as a function of $\sigma^2_{\mathrm{n_{out},i-1}}$ and F_i:

$$\sigma^2_{\mathrm{n_{out},i}} = f(\sigma^2_{\mathrm{n_{out},i-1}}, F_i). \tag{13}$$

In this formulation, we ignore that the input to the i-th stage is already quantized, i.e. that the re-quantization noise introduced in the i-th stage is in reality lower if $\sigma^2_{\mathrm{n_{out},i-1}}$ does not represent an ideal input signal with perfect precision. This leads to a small estimation error. We will demonstrate how successive intermediate quantization steps can be incorporated into our formulation in the next subsection.

Combining the above output noise formulations, the output noise variance from the last FFT stage is a function of $\sigma^2_{\mathrm{n_{in}}}$ and F_1 through F_4, and the constraint function becomes:

$$N(\sigma^2_{\mathrm{n_{in}}}, F_1), F_2), F_3), F_4) = f(f(f(f(\sigma^2_{\mathrm{n_{in}}}, F_1), F_2), F_3), F_4) \leq \sigma^2_{\mathrm{n_{out}}}. \tag{14}$$

In other words, under different input SNRs $1/\sigma^2_{\mathrm{n_{in}}}$, the targeted output SNR $1/\sigma^2_{\mathrm{n_{out}}}$ should remain constant.

4.3 Extension to Multiple Inputs with Intermediate Quantization

Fig. 5 shows the data flow graph of an IDCT, which has 64 inputs with different statistical properties for one IDCT calculation. In the IDCT case, input data values are

already quantized integer values in the frequency domain with zero fractional bits. The inputs experience two multiplications with cosine values and are then summed up to generate a final output value. This process is repeated with different coefficients to generate 64 different outputs. There are four internal quantization points, where the last output quantization step simply removes all fractional bits. Accordingly, there are three decision variables, F_1, F_2 and F_3, where a fourth word length is hardcoded to $F_4 = 0$.

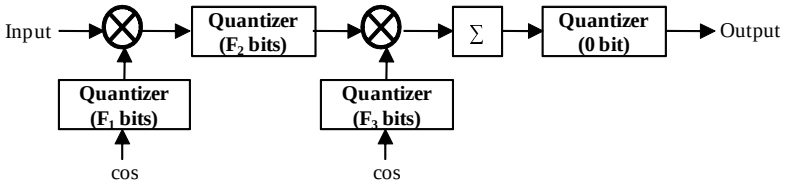

Fig. 5: DFG for 64-point IDCT

Different from the FFT example, we consider the effect of re-quantizing an already quantized signal down to a smaller number of bits [13]. The variance of the additional noise introduce by such a re-quantization can be modeled as the difference in quantization noise at the input and output of an internal quantizer:

$$\sigma^2 = \frac{1}{3}(2^{-2F_{\mathrm{OUT}}-2} - 2^{-2F_{\mathrm{IN}}-2}), \tag{15}$$

where F_{IN} is the number input bits and F_{OUT} is the number of fractional bits at the quantizer output. The formulation accurately considers the number of bits in an already quantized input signal, and it shows that there is no additional noise if $F_{\mathrm{OUT}} = F_{\mathrm{IN}}$, i.e. that F_{OUT} must be the same or smaller than F_{IN} to achieve an additional quantization effect. Note that quantization in the integer domain can be modeled through negative values of F.

We present two different formulations for IDCT noise estimation. Similar to the FFT example, we first formulate our noise model by considering all 64 input values and all cosine coefficients as realizations of one random variable each, i.e. inputs and coefficients are represented by two lumped variances. This allows us to apply the same formulation approach as in the FFT. For such a lumped model, the quantization noise variances in the first and the second coefficient quantizers are

$$\sigma^2_{\mathrm{n_{c1}}} = \frac{1}{3}2^{-2F_1-2}, \tag{16}$$

$$\sigma^2_{\mathrm{n_{c2}}} = \frac{1}{3}2^{-2F_3-2}. \tag{17}$$

Assuming an input signal with error-free signal variance $\sigma^2_{\mathrm{s_{in}}}$, noise variance $\sigma^2_{\mathrm{n_{in}}}$, and integer format with zero fractional bits ($F_0 = 0$), the noise variance after the first multiplication and the following quantizer becomes:

$$\sigma^2_{\mathrm{n_{m1}}} \approx \sigma^2_{\mathrm{s_{in}}}\sigma^2_{\mathrm{n_{c1}}} + \sigma^2_{\mathrm{n_{in}}}\sigma^2_{\mathrm{s_c}} + \frac{1}{3}(2^{-2F_2-2} - 2^{-2F_1-2}), \tag{18}$$

where $\sigma^2_{\mathrm{s_c}}$ is the error-free cosine variance. After the second multiplication and summation, the noise variance becomes:

$$\sigma^2_{\mathrm{n_{m2}}} = 64(\sigma^2_{\mathrm{s_{in}}}\sigma^2_{\mathrm{s_c}}\sigma^2_{\mathrm{n_{c2}}} + \sigma^2_{\mathrm{s_c}}\sigma^2_{\mathrm{n_{m1}}} + \sigma^2_{\mathrm{n_{c2}}}\sigma^2_{\mathrm{n_{m1}}}). \tag{19}$$

Finally, the output quantizer rounds off all the fractional bits to make the final output values be integers. We target a PSNR metric with a fixed and known peak signal variance as quality goal. Hence, the constraint can be formulated as:

$$N(\sigma^2_{\mathrm{n_{in}}}, F_1, F_2, F_3) = \sigma^2_{\mathrm{n_{m2}}} + \frac{1}{3}(2^{-2} - 2^{-2(F_2+F_3)-2}) \leq \sigma^2_{\mathrm{n_{out}}}, \tag{20}$$

where the output noise variance is constrained to be the same or smaller than $\sigma^2_{\mathrm{n_{out}}}$.

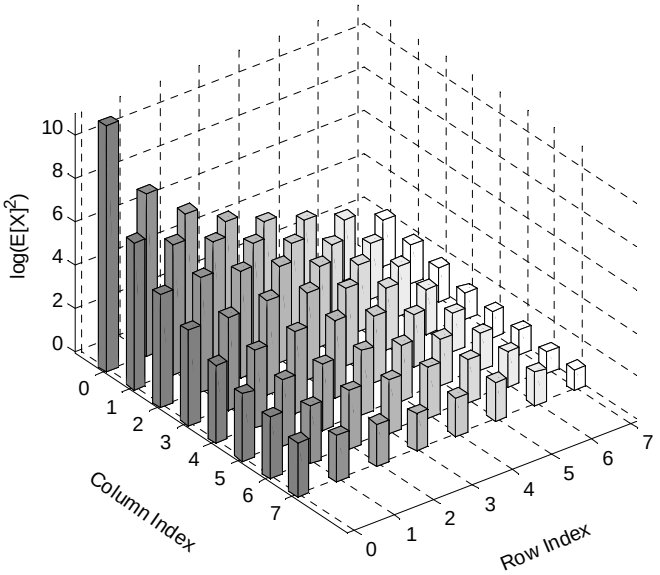

Fig. 6: Power distribution for 8×8 IDCT input with compression rate = 10

As a second formulation, we develop a separated model in which each of the inputs and coefficients is considered as a different random variable. Such an approach is possible for applications that operate with a fixed number of inputs in a block-wise manner, as is the case in our IDCT example. This requires more information about the functionality and the inputs of a DFG, but can achieve a more accurate noise estimation. We use different variances for each of the inputs based on expected or actual observations. Fig. 6 shows the log-scaled input variance distribution for typical 8×8 IDCT frequency-domain inputs. As is well-known, we can observe that most of the input information is located at low frequencies. Especially the DC input element has a significantly higher

variance than other input elements. Furthermore, the cosine values it is multiplied with are a constant, $\frac{1}{\sqrt{2}}\cos(0)$, across all outputs being computed. This fact allows us to model the quantization noise for the DC input coefficient as a constant instead of a uniformly distributed variance, which increases estimation accuracy. All other coefficients vary per output and are thus modeled in lumped form as shown in (16) and (17).

We build a separated optimization problem treating each input element as one random variable with signal and noise variance $\sigma_{\mathrm{s}_{\mathrm{in},i,j}}$ and $\sigma_{\mathrm{n}_{\mathrm{in},i,j}}$, where i and j are the column and row index of the input element. Also, each of the associated cosine coefficients is treated as a random variable with pre-computed error-free signal and quantization noise variances of $\sigma_{\mathrm{s}_{\mathrm{c1},i,j}}$ and $\sigma_{\mathrm{n}_{\mathrm{c1},i,j}}$ for the first cosine coefficients, and $\sigma_{\mathrm{s}_{\mathrm{c2},i,j}}$ and $\sigma_{\mathrm{n}_{\mathrm{c2},i,j}}$ for the second ones. After the first multiplication and the following quantizer, the noise variance of the (i,j) element becomes:

$$\sigma^2_{\mathrm{n}_{\mathrm{m1},i,j}} \approx \sigma^2_{\mathrm{s}_{\mathrm{in},i,j}}\sigma^2_{\mathrm{n}_{\mathrm{c1},i,j}} + \sigma^2_{\mathrm{n}_{\mathrm{in},i,j}}\sigma^2_{\mathrm{s}_{\mathrm{c1},i,j}} + \frac{1}{3}(2^{-2F_2-2} - 2^{-2F_1-2}). \tag{21}$$

With this, the noise variance after the second multiplication and summation becomes:

$$\sigma^2_{\mathrm{n}_{\mathrm{m2}}} = \sum_i \sum_j (\sigma^2_{\mathrm{s}_{\mathrm{in},i,j}}\sigma^2_{\mathrm{s}_{\mathrm{c1},i,j}}\sigma^2_{\mathrm{n}_{\mathrm{c2},i,j}} + \sigma^2_{\mathrm{s}_{\mathrm{c2},i,j}}\sigma^2_{\mathrm{n}_{\mathrm{m1},i,j}} + \sigma^2_{\mathrm{n}_{\mathrm{c2},i,j}}\sigma^2_{\mathrm{n}_{\mathrm{m1},i,j}}). \tag{22}$$

The final output noise variance and constraint is the same as in (20).

5 Power Model

The cost in our optimization problem is determined the the dynamic power consumption of the system. The power consumption is the same for all input SNRs, and it only differs with the word lengths of operations in the DFG. The energy cost for an addition is thereby different from the cost for a multiplication, and both costs are affected by the active word lengths of the arithmetic computations. The total energy cost $P(\mathbf{F})$ is the sum of the energy costs P_i for each operation in the dataflow graph. P_i is a function of the arithmetic operation type performed to compute the i-th controllable variable with its active word length F_i:

$$P(\mathbf{F}) = \sum_i P_i(F_i). \tag{23}$$

We assume that all the bits in our design have the same transition rate of 0.5. With this assumption, dynamic power consumption is linearly proportional to the area of the circuit that is toggling. Hence, our power cost function can be represented as the number of unit hardware blocks. For combinational logic such as adders and multipliers, the cost for each stage is the same and can be represented as the number of 1-bit full adder equivalents. For sequential logic, the power consumption of a 1-bit D flip-flop (DFF) is compared to that of a 1-bit full adder in the TSMC 0.18μm target technology library used in our validations.

5.1 Datapath Power Analysis

For the example of one stage of the FFT, I_{a} is the number of integer bits at the input to the FFT and I_{b} is the number of integer bits for the twiddle factor. Then, the cost for one butterfly is

$$c^{'} = 2 \times (I_{\mathrm{a}} + F) + (I_{\mathrm{a}} + 1 + F) \tag{24}$$

and the cost for one twiddle multiplication becomes

$$c^{''} = 2 \times (I_{\mathrm{a}} + 2 + F) \times (I_{\mathrm{b}} + F) + (I_{\mathrm{a}} + I_{\mathrm{b}} + 2 + 2F). \tag{25}$$

$I_{\mathrm{a}} = 3$ including the sign bit is enough not to affect decoding performance. Also, I_{b} is 1 since the range of twiddle factors is within ± 0.5.

In each FFT stage, two intermediate values are stored: $(I_{\mathrm{a}} + F)$ bits of data after input quantization and $(I_{\mathrm{a}} + F + 2)$ bits of data after the butterfly. Hence, the number of DFFs used in one FFT stage becomes $(2F + 8)$. According to our synthesis results, the ratio in power consumption between a 1-bit DFF and a 1-bit full adder is 8.4, and this is used as a weight of the normalized sequential logic cost:

$$c^{'''} = 8.4(2F + 8). \tag{26}$$

With this, the total cost of one FFT stage becomes:

$$C(F) = c^{'} + c^{''} + c^{'''} = 2F^2 + 33.4F + 32. \tag{27}$$

Note that for large FFTs, intermediate data is usually stored in SRAMs. However, since scaling is only performed for DFFs and combinational logic, the power consumption of SRAMs is not included in our analysis.

The power model for a complete 4-stage FFT is a straightforward extension of the 1-stage model. Likewise, we obtain the power model for our IDCT example in the exactly the same way as for the FFT example.

5.2 Overhead Analysis

If precision scaling is applied at run time to dynamically change word lengths in response to varying input conditions, an input SNR measurement block and a mapping between measured input SNRs and word lengths for all decision variables has to be added to the system. Also, if not already part of a programmable micro-architecture, combinational gates have to be added in front of the DFFs to control clock gating.

Most wireless communication systems already include SNR measurement capabilities for various uses such as channel state information feedback. In such cases, it is assumed that our approach uses the existing SNR measurement block and we do not include it in overhead analysis. Similarly, for image/video processing applications it is assumed that compression rates used in the encoder and hence SNR at the inputs of the decoder is communicated through existing out-of-band mechanisms.

By contrast, with binary on/off decisions stored in mapping tables, their size becomes $N_{\mathrm{s}} \times N_{\mathrm{d}}$, where N_{s} is the number of SNR steps and N_{d} is the number of DFFs

to control. For example, $N_{\mathrm{s}} = 11$ if there are 11 input SNR steps from 6dB to 16dB, and $N_{\mathrm{d}} = 24$ for an FFT with four stages, where each stage has 6 DFFs to be controlled. The overhead in power consumption of the mapping table and additional clock gating logic is included in our power analysis shown in the results.

6 Results

In the following, we validate our optimization model and present optimization results demonstrating achievable gains for the FFT and IDCT examples. Since our optimization problem is neither linear nor convex, we apply adaptive simulated annealing (ASA) [18] for solving the optimization as in [15]. ASA is known to be able to adapt to changing sensitivities and has faster convergence compared to traditional simulated annealing approaches.

We perform power estimation of the generated gate-level netlists using Synopsys Design Compiler and Power Compiler with a TSMC 0.18μm library at a 40MHz clock. We do not specify the actual activity factors for power estimation and use the default options in the tools. We include both dynamic and leakage power consumption in all reported results. Our optimization is only targeted at dynamic power, and leakage is less than $1\,\mu$W for the more complex FFT example in 0.18μm. For more advanced technology nodes with a larger fraction of leakage power, design techniques such as power gating can be combined with dynamic word length scaling.

6.1 FFT Optimization Results

We apply our approach to a 256-point FFT example in a quadrature phase-shift keying (QPSK) OFDM receiver with a cyclic prefix length of 64 assuming perfect synchronization. As shown in Fig. 7, the OFDM receiver consists of a synchronization block, a 256 point FFT, an equalizer, and a symbol de-mapper. An AWGN channel model is assumed to exist between transmitter and receiver. The FFT is used as an example to be designed in dynamically scaled fixed point form. Without loss in generality, among many implementation schemes, we assume that a pipelined radix-4 FFT is used. As presented earlier, the 256-point FFT has four radix-4 stages and each stage contains a radix-4 butterfly and a twiddle multiplication. Since we change the SNR of the system by adding quantization noise, the targeted SNR of the FFT is defined as a desired SNR at its output, which is affected both by a given input SNR and internal quantization noise sources. At design time, statistical analysis determines multiple sets of word lengths for all internal FFT variables and at all input SNRs defined through floating-point simulations. At run time, a SNR block measures the FFT's input SNR and a word length controller selects the best set of word lengths that is suitable for the current input SNR to maintain a pre-defined output SNR. We assume that perfect SNR measurement is possible. We only use fixed-point numbers with a round-to-nearest rounding method.

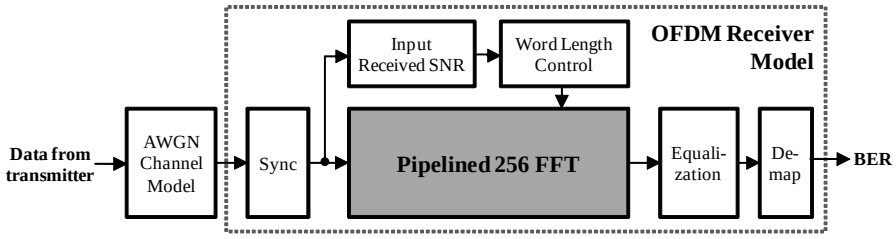

Fig. 7: An OFDM receiver

The final performance metric for a wireless communication system is usually the coded frame error rate (FER). In this chapter, however, we use uncoded bit error rate (BER) instead. Every FER has a corresponding BER, which is not affected by frame length and coding scheme. Our goal is to find FFT word lengths that satisfy a desired BER for any given input SNR. BER is closely related to SNR. However, BER is decision error and the relationship between SNR and BER is not linear, but a function of the noise's probability density function (PDF). It is hard to find the exact PDF of noise for a general DSP system that has quantization noises. In this chapter, we therefore assume that propagated noise at the output of an FFT stage is Gaussian distributed. From the central limit theorem, it follows that the noise at the output of a radix-4 butterfly can be approximated as Gaussian. Our simulation results also show that the output noise from twiddle multiplications, since additive, can be approximated to be Gaussian. Furthermore, the input signal is assumed to be Gaussian. This is true considering the time-domain signal of an OFDM system. Hence, although we use an SNR metric in our analysis, under the above assumptions we can estimate BER from SNR.

The method presented in this chapter, which we call dynamic scaling by variance propagation (DS-VP), is compared against four conventional methods: 1) non-scaling by full simulation (NS-FS), which only finds one set of word lengths for the worst-case operating point, 2) coarse dynamic scaling by full simulation search (CS-FS), which finds multiple sets of word lengths by full simulation, but using a single word length for all variables in a set, 2) dynamic scaling by full simulation search (DS-FS), which finds optimal sets of word lengths using full simulation, and 4) dynamic scaling by efficient simulation search (DS-ES), which finds multiple sets of word lengths using the efficient simulation approach from [12].

Table 1 shows the sets of word lengths found by the different methods across different target BERs and corresponding input SNRs. The sets of word lengths in Table 1 are the word lengths for Stage 1 to Stage 4 of the FFT, i.e. $\{F_1, F_2, F_3, F_4\}$. The table also includes estimated power consumption and optimization runtime for each approach. All experiments were performed on an Intel Core i7 workstation running at 2.7ĠHz. The sets of word lengths from DS-FS are optimal and used as word length and power reference.

Our method shows a significant gain in design time compared to simulation-based methods, which makes dynamic scaling feasible even for large systems with many variables and operating points. For one operating point in our FFT example, the number of simulations using a full search is 6^4 (4 decision variables and with a range from 1 to 6

Table 1: Optimized word lengths for various target SNRs (BERs)

	Channel	NS-FS		CS-FS		DS-FS		DS-ES		DS-VP	
	SNR	$\{F_i\}$	Power	$\{F_i\}$	Power [mW]	$\{F_i\}$	Power [mW]	$\{F_i\}$	Power [mW]	$\{F_i\}$	Power [mW]
7.25dB (1%)	<8dB	{4,4,4,3}	2.52 mW	{4}	2.53 (0.4%)	{4,4,4,3}	2.57 (2.0%)	{4,4,4,3}	2.57 (2.0%)	{4,4,4,3}	2.57 (2.0%)
	8dB			{3}	2.27 (-9.9%)	{3,3,3,2}	2.20 (-12.7%)	{3,3,3,2}	2.20 (-12.7%)	{3,3,3,2}	2.20 (-12.7%)
	9dB			{3}	2.27 (-9.9%)	{3,3,3,1}	2.13 (-15.5%)	{3,3,3,1}	2.13 (-15.5%)	{3,2,3,2}	2.16 (-14.3%)
	10dB			{3}	2.27 (-9.9%)	{3,3,2,1}	2.05 (-18.7%)	{3,2,3,1}	2.06 (-18.3%)	{3,2,2,1}	1.97 (-21.8%)
	11dB			{2}	1.94 (-23.0%)	{3,2,2,1}	1.97 (-21.8%)	{3,2,2,1}	1.97 (-21.8%)	{2,2,3,1}	1.99 (-21.0%)
	12dB			{2}	1.94 (-23.0%)	{2,2,2,1}	1.90 (-24.6%)	{2,2,2,1}	1.90 (-24.6%)	{2,2,2,1}	1.90 (-24.6%)
9.7dB (0.1%)	<10dB	{4,5,4,4}	2.63 mW	{5}	2.94 (11.8%)	{4,5,4,4}	2.66 (1.1%)	{4,5,4,4}	2.66 (1.1%)	{4,5,4,4}	2.66 (1.1%)
	10dB			{4}	2.53 (-3.8%)	{5,5,4,3}	2.60 (-1.1%)	{5,5,4,3}	2.60 (-1.1%)	{4,4,4,4}	2.57 (-2.3%)
	11dB			{3}	2.27 (-13.7%)	{3,3,3,2}	2.20 (-16.3%)	{3,3,3,2}	2.20 (-16.3%)	{3,3,3,2}	2.20 (-16.3%)
	12dB			{3}	2.27 (-13.7%)	{3,3,3,2}	2.20 (-16.3%)	{3,3,3,2}	2.20 (-16.3%)	{3,3,3,1}	2.13 (-19.0%)
	13dB			{3}	2.27 (-13.7%)	{3,3,2,2}	2.16 (-17.9%)	{3,3,2,2}	2.16 (-17.9%)	{3,2,3,2}	2.16 (-17.9%)
	14dB			{3}	2.27 (-13.7%)	{3,3,2,2}	2.16 (-17.9%)	{3,3,2,2}	2.16 (-17.9%)	{3,2,2,2}	2.06 (-21.7%)
11.4dB (0.01%)	<12dB	{5,4,5,4}	2.75 mW	{5}	2.94 (6.9%)	{5,4,5,4}	2.83 (2.9%)	{5,4,5,4}	2.83 (2.9%)	{5,4,5,4}	2.83 (2.9%)
	12dB			{4}	2.53 (-8.0%)	{4,5,3,3}	2.49 (-9.5%)	{4,5,3,3}	2.49 (-9.5%)	{4,4,4,3}	2.57 (-6.5%)
	13dB			{3}	2.27 (-17.5%)	{4,3,3,2}	2.29 (-16.7%)	{4,3,3,2}	2.29 (-16.7%)	{4,3,3,2}	2.29 (-16.7%)
	14dB			{3}	2.27 (-17.5%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,3,2}	2.20 (-20.0%)
	15dB			{3}	2.27 (-17.5%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,2,3}	2.22 (-19.3%)
	16dB			{3}	2.27 (-17.5%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,3,2}	2.20 (-20.0%)	{3,3,2,2}	2.16 (-21.5%)
Optim. time		3.6 hours		6 min.		21.6 hours		1-2 min.		1.2-1.8 msec.	

bits each). For each simulation trial, we run 10,000 OFDM symbols corresponding to 5 million bits in order to achieve enough simulation accuracy. Each such simulation takes about 10 seconds. To find the optimal word lengths using an exhaustive search requires 3.6 hours. With the preplanned simulation method from [12], the number of trials can be significantly reduced. For example, if the search starts from $\{2,2,2,2\}$, and the optimal word length set is $\{4,3,3,2\}$, optimal word lengths can be obtained with only 4 simulations. However, for dynamic scaling, a search is required for each operating point and total optimization time increases linearly with the number of operating points. Thus, even efficient simulation-based methods may still not be suitable for design-time optimization in the presence of dynamic scaling.

By contrast, our analysis method requires only about 2ms to find a set of word lengths for one operating point, which is 5,000 times faster than the time for one simulation trial. Considering that word length optimizations can take up to 50% of design time with conventional simulation-based approaches [12], this represents a significant improvement in productivity.

To validate the optimality and accuracy of our approach, achievable power figures using various methods are compared to those of the reference DS-FS approach. Fig. 8 shows that our cost function used for optimization correlates well with the final gate-level power numbers. Nevertheless, the DS-VP method results in up to a 5% difference in power consumption, which is a downside of achieving large gains in design time. The DS-ES method also exhibits a small 0.5% difference in some isolated cases where it is not able to guarantee the optimal solution. We also compared fine-grain DS-based methods against dynamic scaling with coarse optimizations, i.e. using a single word length for all variables (CS-FS). Power numbers using fine-grain scaling are always the same or smaller with a reduction of up to 13.6% even considering additional overhead for control at finer granularity.

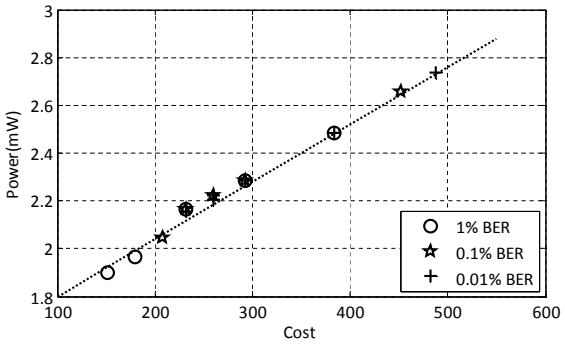

Fig. 8: Accuracy of cost function

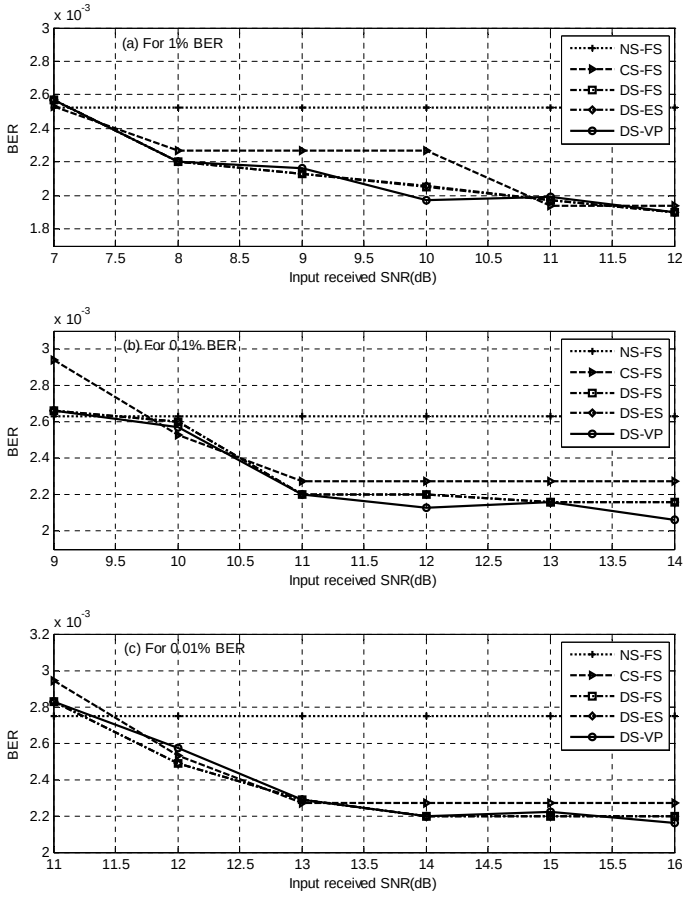

Fig. 9: Power comparison

In terms of overhead, compared to a method with no scaling (NS-FS), the extra power consumption for dynamic scaling is less than 3% according to our synthesis results. This overhead is small compared to the average 17% power reduction that can be achieved by dynamic scaling across varying input SNR levels. At SNR levels that are lower than the required SNR, the power numbers are larger than those for NS-FS due to the overhead of finely tuned dynamic scaling. The system, however, is not usually in such a poor environment. Hence, on average, large power savings can be expected.

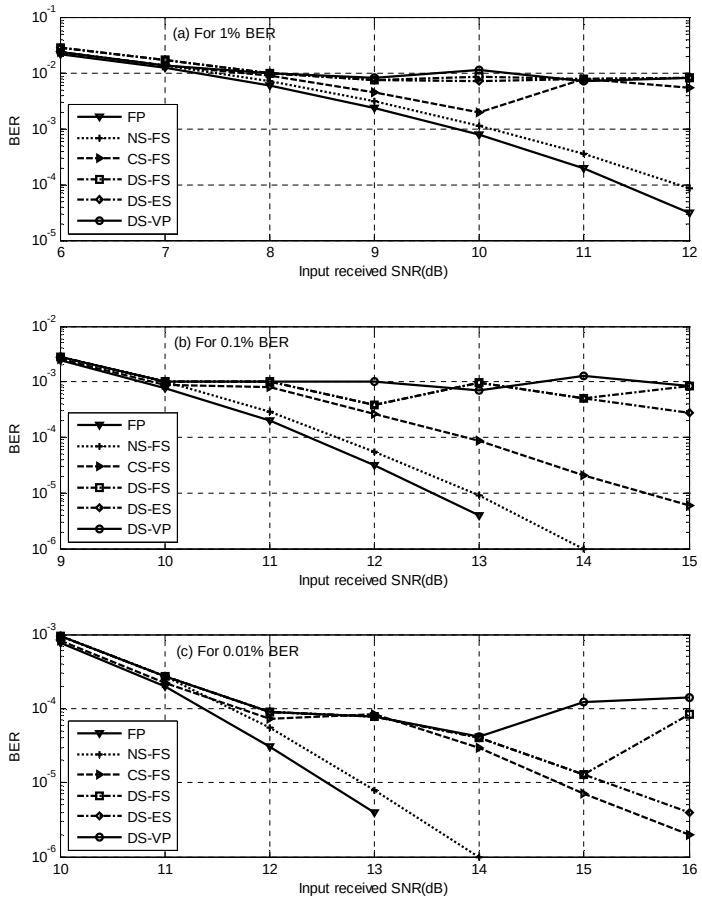

Fig. 10: QPSK BER comparison

Finally, Fig. 9 and Fig. 10 plot the results of performance simulations. Using DS-type methods, the system is able to maintain the targeted output BER over the full input SNR range leading to a large power reduction at higher SNR values. In Fig. 10, the BER of floating point model (FP) is also plotted as a reference. The measured BER

for a targeted BER of 0.01% ranges from 0.004% to 0.014% using our DS-VP method. Note that while in some cases the power consumption can be lower than in other DS-based methods, this comes at the cost of violating the BER constraint for those operating points. This mismatch is caused by the heuristic nature of our optimization approach.

6.2 IDCT Optimization Results

We further apply our approach to an IDCT block within an overall JPEG image processing chain. As shown in Fig. 11, a JPEG encoder performs color conversion, a discrete cosine transform (DCT), quantization, zigzag ordering, and finally Huffman encoding. The decoder implements a reverse processing chain, using the IDCT as its main image reconstruction block. The algorithmic quantization step in the encoder is the key for achieving lossy compression in the JPEG algorithm. It uses the fact that most of the information in a natural image exists in the low frequency region to non-uniformly quantize and scale the frequency-domain DCT components. Coupled with subsequent run-length encoding, this achieves a size reduction of the encoded bit stream at the expense of a reduced image quality after decoding. This tradeoff is controllable by the quantization and compression factor selected in the encoder.

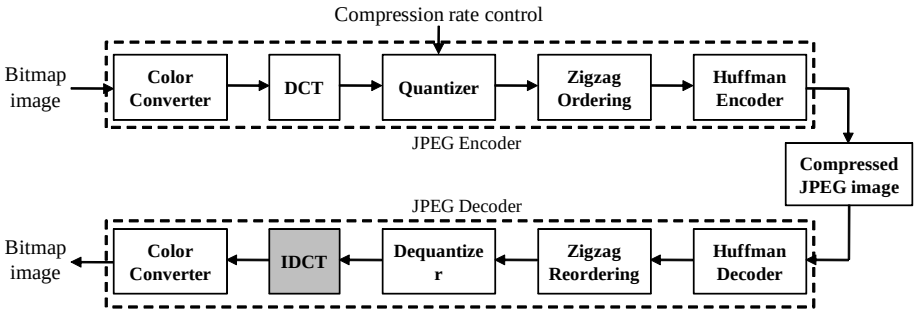

Fig. 11: A JPEG encoder/decoder

We apply our word length optimization to the IDCT block in the JPEG decoder, where we optimize IDCT word lengths for different operating points as determined by the algorithmic quantization factor selected in the encoder. Changing the encoder's compression rate will influence the frequency-domain noise at the input of the IDCT and hence the PSNR of the decoded image at the IDCT output. This allows us to apply different precision scaling levels depending on the compression level of the image data at the decoder input.

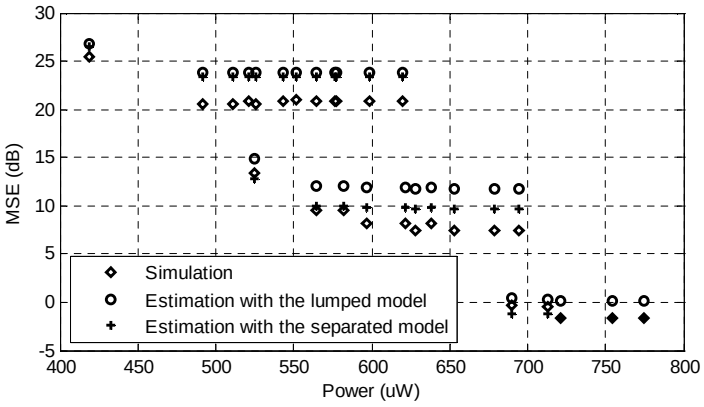

Fig. 12: IDCT design space of power consumption versus quality loss

We use the standard Lena image file as sample for all our experiments. Fig. 12 shows the design space of power consumption and image quality loss for the IDCT. In Fig. 12, we evaluate quality by only considering image degradation due to internal IDCT quantization noise, i.e. assuming that image data at the IDCT input represents an error-free reference signal. In our estimation models, we therefore set input noise to be zero. For simulations, the output bitmap image of the fixed-point IDCT is compared against the output of a reference floating-point IDCT using real image data. Note that quality results in Fig. 12 are different from typical PSNR measurements, since errors reported here do not include losses incurred by the encoder's compression. In all cases, power results were obtained from RTL synthesis. For each of the possible word length combinations (F_1, F_2, and F_3 sweeped from 2 to 10 with $F_2 \leq F_1$) we plot: 1) image quality obtained from simulation, 2) quality given by our lumped estimation model, and 3) quality estimated by our separated noise model. The estimation error of the lumped model is less than 12% compared to a simulation of the same design, with an average estimation error of 8.3%. For the separated model, the maximum and average estimation error is 10.1% and 5%, respectively. These results show that the estimation accuracy can be increased with more information about the inputs to the DFG.

Fig. 13 shows the final output image quality of a precision-scaled IDCT in reference to a floating-point IDCT for various operating points as defined by the encoder compression rate. Different from the FFT example that targeted a constant output SNR under varying input conditions, we optimize the IDCT to achieve a constant quality loss. We use a separated model and from simulations, we first obtain the variances of individual IDCT inputs as a function of the encoder compression rate as shown in Fig. 6. Again, our optimization problem is only concerned with quality losses incurred in the IDCT. Furthermore, it can not be assumed that compression noise is independent from frequency-domain image data. As such, we formulate a simplified model that considers encoder compression and IDCT scaling independently, i.e. we treat combined image data with compression noise as the IDCT input signal with no separate noise sources (set to zero). Then, for any compression rate, we set the targeted quality loss of the op-

timization problem such that the final output image PSNR will become 1 dB lower than the corresponding ideal PSNR of a simulated floating-point IDCT. For example, with a JPEG compression rate of 5, the ideal image PSNR at the IDCT output is 42.6 dB, and we optimize the IDCT to achieve an overall 41.6 dB output PSNR instead. We use our optimization framework to find the sets of optimal word lengths, and subsequently perform simulations to determine the actual PSNR losses. As shown in Fig. 13, actual PSNR losses as compared to an ideal implementation can reach 1.8 dB, which is an artifact of errors in our estimation model.

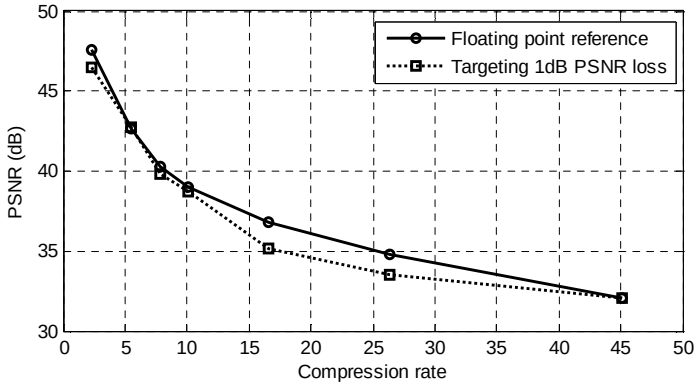

Fig. 13: Output image PSNR for different IDCT implementations

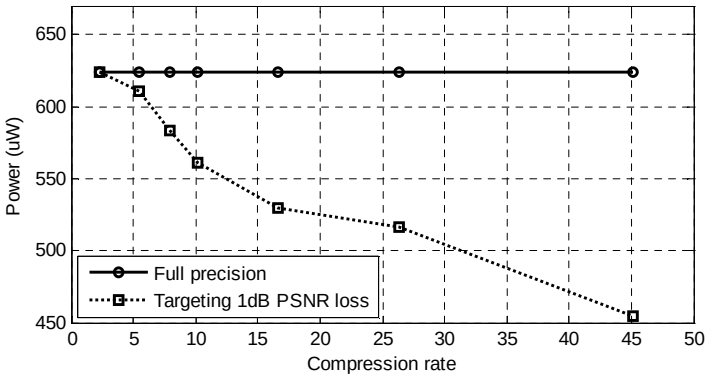

Fig. 14: IDCT power reduction

Finally, Fig. 14 shows the power reduction achieved through word length scaling. As we can observe, a higher power reduction is achieved when the compression rate is

high and accordingly the input PSNR is low. The power reduction is up to 27.1% for a compression rate of 45. This result shows another application for precision scaling: if the input quality decreases, we can reduce power by injecting more quantization noise while keeping the output quality degradation within an allowed range. This is due to the fact that, at higher compression levels, input data is already quantized algorithmically in the encoder, requiring less precision and energy to decode. As shown in Fig. 15, this allows for significant power savings with no visually perceivable differences in decoding performance across a wide range of JPEG compression rates.

(a) Floating-point IDCT, compression rate=2.3 (b) Fixed-point IDCT, compression rate=2.3

(c) Floating-point IDCT, compression rate=45 (d) Fixed-point IDCT, compression rate=45

Fig. 15: Floating- and fixed-point IDCT output at different compression rates

7 Summary and Conclusions

In this chapter, we introduced a statistical analysis method using variance propagation for word length optimization. A fine-grain optimization of precision scaling is possible and results in significant power savings. A fast yet accurate static design- or compile-time approach thereby avoids run-time overhead and the need for time-consuming exhaustive simulations. In the future, we plan to generalize our method to other types of operations and blocks in DSP systems, including optimization for other metrics, such as coded BER, and other error models, such as the ones arising from other approximation techniques. Furthermore, we plan to automate the approach, including generation of optimized hardware description language code and clock-gating logic within our flow.

Acknowledgments

This work has been supported by Intel and the National Science Foundation under Grant No. CCF-1018075. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Widrow, B.: Statistical analysis of amplitude-quantized sampled-data systems. In: T. American Institute of Electrical Engineers Part II: Applications and Industry, pp. 555–568. (1961)
2. Miyazaki, N. and Yoshizawa, S. and Miyanaga, Y.: Low-Power Dynamic MIMO Detection for a MIMO-OFDM Receiver. In: IEEE International Symposium on Intelligent Signal Processing and Communications Systems, pp. 1–6. Bangkok (2011)
3. Nisar, M.M. and Chatterjee, A.: Test Enabled Process Tuning for Adaptive Baseband OFDM Processor. In: 26th IEEE International Symposium on VLSI Test, pp. 9–16. San Diego (2008)
4. Oppenheim, A.V. and Weinstein, C.J.: Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform. In: IEEE Proceedings, pp. 957–976. (1972)
5. Yoshizawa, S. and Miyanaga, Y.: Tunable Word Length Architecture for Low Power Wireless OFDM Demodulator. In: IEEE International Symposium on Circuits and Systems, pp. 2789–2792. Island of Kos (2006)
6. Shi, C. and Brodersen, R.W.: A Perturbation Theory on Statistical Quantization Effects in Fixed-Point DSP with Non-Stationary inputs. In: IEEE International Symposium on Circuits and Systems, pp. III-373–6. Vancouver (2004)
7. Kim, J. and Yoshizawa, S. and Miyanaga, Y.: Dynamic Wordlength Calibration for Energy Reduction FFT Processors in Wireless LAN. In: 54th IEEE Midwest Symposium on Circuits and Systems, pp. 1–4. Seoul (2011)
8. Nisar, M.M. and Chatterjee, A.: Environment and Process Adaptive Low Power Wireless Baseband Signal Processing Using Dual Real-Time Feedback. In: 22nd IEEE International Conference on VLSI Design, pp. 57–62. New Delhi (2009)
9. Novo, D. and Bougard, B. and Lambrechts, A. and Van der Perre, L. and Catthoor, F.: Scenario-Based Fixed-point Data Format Refinement to Enable Energy-scalable Software Defined Radios. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 722–727. Munich (2008)

10. Nguyen, H. N. and Menard, D. and Romuald, R. and Sentieys, O.: Energy Reduction in Wireless System by Dynamic Adaptation of the Fixed-Point Specification. In: Conference on Design and Architectures for Signal and Image Processing, pp. 132–139. Brussels (2008)
11. Kum, K-I. and Sung, W.: Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems. IEEE T. Computer-Aided Design of Integrated Circuits and Systems. vol. 20, n. 8, 921–930 (2001)
12. Han, K. and Evans, B. L.: Optimum Wordlength Search Using Sensitivity Information. EURASIP J. Advances in Signal Processing. vol. 2006, n. 1, 1–14 (2006)
13. Constantinides, G.A. and Cheung, P. Y K and Luk, W.: Wordlength Optimization for Linear Digital Signal Processing. IEEE T. Computer-Aided Design of Integrated Circuits and Systems. vol. 22, n. 10, 1432–1442 (2003)
14. Menard, D. and Sentieys, O.: Automatic Evaluation of the Accuracy of Fixed-Point Algorithms. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 529–535. Paris (2002)
15. Lee, D.-U. and Gaffar, A.A. and Cheung, R. C C and Mencer, O. and Luk, W. and Constantinides, G.A.: Accuracy-Guaranteed Bit-Width Optimization. IEEE T. Computer-Aided Design of Integrated Circuits and Systems. vol. 35, n. 10, 1990–2000 (2006)
16. Shi, C. and Brodersen, R.W.: Automated Fixed-Point Data-Type Optimization Tool for Signal Processing and Communication Systems. In: 41st Design Automation Conference, pp. 478–483. San Diego (2004)
17. Widrow, B. and Kollár, I.: Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications. Cambridge University Press, New York (2008)
18. ASA 25.15, `http://www.ingber.com//#ASA`
19. Sampson, A. and Dietl, W. and Fortuna, E. and Gnanapragasam, D. and Ceze, L. and Grossman, D.: EnerJ: Approximate Data Types for Safe and General Low-power Computation. In: 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 164–174. San Jose (2013)
20. Venkataramani, S. and Chippa, V. K. and Chakradhar, S. T. and Roy, K. and Raghunathan, A.: Quality Programmable Vector Processors for Approximate Computing. In: 46th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 1–12. Davis (2013)
21. Constantinides, G.A. and Woeginger, G.J.: The Complexity of Multiple Wordlength Assignment. J. Applied Mathematics Letters. vol. 15, n. 2, 137–140 (2002)
22. Han, J. and Orshansky, M.: Approximate Computing: an Emerging Paradigm for Energy-Efficient Design. In: 41st IEEE European Test Symposium, pp. 1–6. Avignon (2013)
23. Lee, S. and Gerstlauer, A.: Fine grain word length optimization for dynamic precision scaling in DSP systems. In: IFIP/IEEE International Conference on Very Large Scale Integration, pp. 266–271. Istanbul (2013)