Symbolic Noise Analysis Approach to Computational Hardware Optimization

Arash Ahmadi, Mark Zwolinski
Electronic Systems and Devices Group
School of Electronic and Computer Science, University of Southampton, UK
Email: {aa5,mz}@ecs.soton.ac.uk

ABSTRACT

This paper addresses the problem of computational error modeling and analysis. Choosing different word-lengths for each functional unit in hardware implementations of numerical algorithms always results in an optimization problem of trading computational error with implementation costs. In this study, a symbolic noise analysis method is introduced for high-level synthesis, which is based on symbolic modeling of the error bounds where the error symbols are considered to be specified with a probability distribution function over a known range. The ability to combine word-length optimization with high-level synthesis parameters and costs to minimize the overall design cost is demonstrated using case studies.

Categories and Subject Descriptors

B.2 [**Arithmetic and Logic Structures**]: Performance Analysis and Design Aids

General Terms

Algorithms, Design

Keywords

Computational error, word-length optimization, high level synthesis, computer arithmetic

1. INTRODUCTION

Very Large Scale Integrated (VLSI) circuit design and implementation has profoundly changed the size and speed of computing structures by making available an immense amount of computational resources on a single chip. The rapid increase in the size of VLSI systems, and the need to reduce the circuit development time have resulted in a need for CAD tools that can help choose the most suitable design parameters at the early stages of the design process. A variety of methods and approaches are presented for VLSI design, which make it practical to customize designs for specific applications to improve the performance of the system. This customization can be applied to different features of the hardware, ranging from the architecture level to the lowest level of specification. One of the important issues in this regard is the arithmetic characteristics of the functional units in which designers need to determine the most suitable features for arithmetic operations in the algorithm including: word-length (WL) (integer and fractional parts of the numbers), truncation mode (either roundoff or truncation) and overflow mode (either saturation or wrap-around) in respect of design requirements and restrictions.

From a High level Synthesis (HLS) point of view, choosing different features of arithmetic operations can be formulated as an optimization problem provided appropriate models are supplied. The objective of such an optimization method is the highest accuracy with the minimum design costs including area, power consumption and latency. Regarding computation accuracy, a variety of models and approaches are presented in this field of research in which different aspects of the problem are considered.

The objective of this work is to introduce a new method of computational error modeling called Symbolic Noise Analysis (SNA), which is applicable in arithmetic features selection of hardware implementation of the algorithms. A WL optimization method is provided in which the SNA method is used to analyze the computational error at every point of the hardware, without restrictive assumptions about the statistical model of the signals. This model is applied to a Multi-Objective Optimization (MOO) method to find the minimal WL at each point in the hardware implementation.

The paper is organized as follows: section 2 provides a review of related work in this field and a brief review of computational error modeling methods is presented in section 3. The proposed computational error model is presented in section 4 and synthesis results are reported in section 5.

2. RELATED WORK

In [1], Cmar et al. employed interval propagation analysis for range width determination and a simulation-based method for precision bit-width optimization. Their simulation was utilized by a concurrent program which performed the same calculation as a reference and as a custom fixed-point format, and compared the error between the two values. For precision evaluation, the first and second moments of the error at each signal point was examined. No additional mechanism was proposed to automate the tradeoff of

DAC 2008, June 8–13, 2008, Anaheim, California, USA.

system area against error.

Kum and Sung [2] introduced several heuristic WL optimization methods to trade-off system area against Signal-to-Quantization-Noise Ratio (SQNR). The techniques are heuristics based on bit-true simulation of the design under various internal WLs. This method measures the performance of a fixed-point algorithm using simulation results. A reference system is designed without overflows or signal quantization effects and it iteratively modifies the WL of a signal to find a set of optimum WLs satisfying the fixed-point performance. Signal grouping is used to reduce the number of simulations to calculate the uniform WL and minimum WL configuration.

Constantinides et al. focused on developing algorithms for WL optimization [3]. These methods employ static analytical digital noise analysis for DSP applications applied to Linear Time Invariant (LTI) systems implemented as custom processing units. Optimization techniques are proposed which allow the user to trade off implementation area for arithmetic error at the system outputs. Constantinides later extended the previous efforts to nonlinear components in a datapath by employing a small signal approach and investigating the effect of precision optimization on power reduction as a by-product of the WL optimization [4]. [5] also introduces a similar method based on perturbation theory for nonlinear systems. Again, power consumption is not an objective in the optimization heuristic.

Nayak et al. in [6] presented a compiler that takes high-level signal processing algorithms described in MATLAB and generates optimized hardware. Their precision analysis algorithm determines the minimum number of bits required by a model of errors through the Data Flow Graph (DFG), where a set of error transfer functions determines the error contribution of each node. In [7] Roy and Banerjee presented an approach to automate the conversion of floating-point MATLAB programs into fixed-point MATLAB programs for mapping to FPGAs by profiling the expected inputs in order to estimate errors. The algorithm attempts to minimize the hardware resources while constraining the quantization error within a specified limit.

In [8] Le-Gal et al. proposed a methodology that employs an annotated formal model with bit-width information and dynamic range values in order to extract bit-wise information to optimize the area and power consumption of hardware architectures provided by high-level synthesis tools. High-level synthesis is used to formally transform the application into an architecture observing a set of constraints. Then an architecture optimization stage is completed in order to adapt both possible operator and register bit-widths of the design.

Han and Evans [9] also reported a sensitivity and complexity approach to WL optimization. In this work they discussed a pre-planned exhaustive search which utilizes the sensitivity information of hardware complexity and the system output error with respect to the signal WL.

Several papers also describe applications of symbolic analysis in computational error analysis [10]. A basic implementation of this method is known as Affine Arithmetic (AA). In this method, unlike Interval Arithmetic (IA), see [10], error source dependency is taken into account in a parametric representation of the error at different points in data flow graph. In [11] Lee et al. implemented an AA-based method which categories the problem into two parts, range analysis and precision evaluation. The former gives the integer part of the data whereas the latter provides the fractional part of the variables at every point on the DFG. An Adaptive Simulated Annealing (ASA) heuristic is applied to find the near-to optimal points. Similar to this work, a study was reported by Pu and Ha [12] which applied AA with a different heuristic. In the later work, inspired by [13], by applying the central limit theorem, the first and second moments of the output noise are approximated from a symbolic representation of the output noise. Doi et al. also in [14] presented a nonlinear programming method for floating-point to fixed-point conversion in High Level Synthesis (HLS). An Affine Arithmetic error model is employed for error propagation in the data flow graph which is integrated into a nonlinear problem specification. In this optimizer, the bit-width of the functional units is considered as the implementation cost.

3. COMPUTING WITH UNCERTAINTY

It is not trivial to analyze the finite precision errors in actual designs. Computational error is data dependent in that different input data sets can produce different patterns of error in the system. Furthermore, since finite precision effects are nonlinear, it is observable that computational error in algorithms can be dependent on the sequence of the local operations. It means that during the high level synthesis process, allocation and scheduling might affect the predicted error in the output if this dependency has not been considered. There is a variety of approaches to deal with computation accuracy in high level synthesis, which can be classified in many different ways. Here, we categorize these methods regarding their approaches to the computational error modeling.

The first category is based on Noise Analysis (NA), which is widely practised in digital signal processing design and optimization [2, 3, 4, 5, 7]. In this method error sources are considered as independent Wide-Sense Stationary (WSS) noise sources with uniform Probability Density Function (PDF) [15]. This assumption has a great impact on efficiency of the method ; however, there are arguments about different issues in this regard. For instance, in many practical computations, intermediate results are strongly dependent on each other which can violate the independency assumption of the error sources. In addition, there are strong concerns about the noise propagation model. Inspired by system theory studies, the primary works in this field were based on a LTI assumption of the computational systems. Since many of the computational algorithms result in nonlinear system specifications in practice, applying these methods, that are designed for LTI systems, are subject to conditions [3, 5]. Furthermore, there are applications that are required to be implemented in the form of time dependent or adaptive systems. In these cases, the LTI model is not valid and cannot be applied to the error analysis method.

The second category is based on the understanding of the required range of variable values at different points of a given computational procedure to compute the minimum bit-width requirements in a hardware implementation of an algorithm [16, 1, 6, 8, 10, 11, 12]. Assuming that the variation ranges of the input variables are known, these approaches try to predict the variation range of the output data. Therefore, instead of a single value, every number is represented by a range of values between the upper and lower bounds. In other words, regardless of the real place

of the number in the range, these methods are concerned with data range dilation and contraction by data propagation through the operations in the computation tree of the algorithm. Since this method is focused on value bounds, there is no information about how the actual value might be placed in the range. Several methods are introduced in this category such as IA [16], AA [11] and the Taylor Model [10]. These sub-categories are altered in the way of their range representation and approximation.

In sum up, the noise analysis method relies on statistical specification of the error. The error range analysis methods, on the other hand, provide an expression of the error range at the output to evaluate the computation accuracy. By combining these two basic approaches, we propose a new method which is called Symbolic Noise Analysis (SNA).

4. SYMBOLIC NOISE ANALYSIS

Dealing with errors as symbols which have some statistical information with them is the core idea of our method in which uncertainty is represented in the form of an algebraic combination of the noise symbols, as in Equation (1).

$$\hat{x} = F_x\left([x_1, x_2, \cdots, x_N], [\epsilon_{x_1}, \epsilon_{x_2}, \cdots, \epsilon_{x_N}]\right), \tag{1}$$

where $\epsilon_{x_i} \in [-1, +1]$, $F_x(\cdot)$ is a fractional function of polynomials and $(x_1, x_2, \cdots, x_N)$ are the coefficients of the polynomials. $[\epsilon_{x_1}, \epsilon_{x_2}, \cdots, \epsilon_{x_N}]$ is array of *Noise Symbols*, which carry the uncertainty of the represented value, $\hat{x}$, and each symbol ϵ_i is assumed to have a known PDF, P_{ϵ_i}. Unlike the AA method or the Taylor model, the PDFs of the noise symbols are taken into account, in which a PDF can be found for the output uncertainty to show the probability of the output taking each value inside the bounded interval.

As shown in Equation (1), the probability density of the noise symbols is the initial information in the SNA method of error analysis. In general, these error symbols originate from different sources, which means that they might have different PDFs. Because there is no limitation on PDF types in this method, the error model of each sub-block in the design can be replaced by a practically extracted or stimulus based model. This option can be interpreted as a mixed method of dynamic and static analysis methods in which every subsystem can model its noise symbols by simulation. Since SNA is based on a probabilistic analysis of the noise symbols, proper methods of representation and algebraic calculations on PDFs are required.

From a probability viewpoint the interval methods (IA or AA for instance) have implicitly made the assumption that the modeled values are situated within the bounds of the specified intervals with a probability of 1. Accordingly, when an operation, $\circ$, is applied to values $x \in \hat{x}$ and $y \in \hat{y}$, where $\hat{x}$ and $\hat{y}$ are intervals, to get a result $z = x \circ y$, it can be said that $z \in \hat{z} = \hat{x} \circ \hat{y}$ where the probability $P(z \in \hat{z})$ conforms to:

$$P(z \in \hat{z}) = P(x \in \hat{x}) \cdot P(y \in \hat{y}) = 1 \times 1 = 1, \tag{2}$$

assuming $\hat{x}$ and $\hat{y}$ are independent. It is also assumed that all the numbers in the interval have the same probability. Interval representation of the uncertain values, therefore, implies that the uncertain value is a random number with a uniform distribution over the specified interval. This probabilistic view of the interval operations is the core idea of the proposed method in [17], called the *Histogram Method* for doing operations on probability density functions.

Noise symbols in our method (ϵ_i) are bounded random values in a fixed predefined range of $[-1, 1]$. To approximate the density functions, the input range of the function is divided into a number of non-overlapping intervals (bins) with a uniform probability values the same as the value of the function over the range. To use the same standard of discretization for all the PDFs, we divide all the noise symbols into 2^{l+1} subintervals. A histogram (H) can be defined formally as a partition of the error symbol ϵ in terms of intervals I_i and local probabilities p_i such that: $H = \left\{(I_i, p_i) \middle| I_i = \left[-1 + i \cdot 2^{-l}, -1 + (i+1) \cdot 2^{-l}\right]\right\}$, where $i = 0, \cdots, 2^{l+1} - 1$ and l represents the granularity of the histogram H. Corresponding to each interval I_i in the H, a probability p_i is defined which represents the PDF value in the interval I_i. Assuming interval operands are represented according to H, the new operators return the result of the operation also represented in terms of H. When this result includes more than one interval, the operator distributes the probability, p_i, among the output intervals depending on the behavior of the specific arithmetic operation. In order to have consistent input and output data types, this arithmetic can be formulated as histogram arithmetic, as inputs can be viewed as histograms having a single interval. Using our definition, the computation model is modified as:

1. Consider the input space is the set of intervals describing the histogram of the input i in terms of $H_i = \{(I_{ij}, p_{ij})\}$, where $I_{ij} = [a_{ij}, b_{ij}]$.
2. For each vector $[\cdots, ([a_{ij}, b_{ij}], p_{ij}), \cdots]$ of the input space:
   (a) Compute the probability $p_k = \prod_{i=1}^{I} p_{ij}$;
   (b) For each operation with input histograms H_i and for each combination of intervals from the Cartesian product of the intervals of histograms H_i do:
      i. Obtain a histogram result using histogram-based arithmetic;
      ii. Proceed with the next operation if there are any more (step c) otherwise return;
   (c) Set the probability of the resulting histogram interval I_k by its calculated p_k;
3. Collect the result histogram to produce the output histogram (H_{out}).

According to the algorithm, all algebraic operations on histograms can be expanded to the interval operations. Consider a set histograms $H_i = \{(I_{i,j}, p_{i,j})\}$ and function F which is applied to them to produce another histogram as in Equation (3).

$$\begin{aligned} H_{out} &= F(H_1, H_2, \cdots, H_m) \\ &= F\left(\{(I_{1,j}, p_{1,j})\}, \{(I_{2,j}, p_{2,j})\} \cdots, \{(I_{m,j}, p_{m,j})\}\right), \end{aligned} \tag{3}$$

where $H_{out} = \{(I_{out,k}, p_{out,k})\}$ is the result histogram. Applying the algorithm means that $F(\cdot)$ must be applied to all the Cartesian combinations of the input histograms as in Equation (4).

$$\{(I_{out,k}, p_{out,k})\} = \{F\left((I_{i,j}, p_{i,j})\right)\}. \tag{4}$$

All the operations between intervals of the histograms are exactly the same as in the IA method, see [10]. These operations on the input intervals produce a set of overlapping intermediate intervals with the corresponding probability values which need to be mapped into the output histogram

bins. The overall algorithm for implementation of the symbolic noise analysis method consists of three basic major steps:

1. Build up a Noise Symbol representation for signals in the computation tree and their relationship with the arithmetic characteristics of the nodes,
2. Find the symbol propagation through the tree and its relationship in the output node(s),
3. Find the histogram representation of the output PDF and the corresponding bounds and noise powers.

The first step of the algorithm refers to the fact that the errors in each computational node are a function of the arithmetic characteristics of the node such as WL, arithmetic system and so on. The noise symbols assigned to each node are dependent on these characteristics. Noise symbols are created in a data structure which contains their source and PDF in the form of histograms with suitable granularity. The second step consists of polynomial operations to build up the output error relationship with the noise symbols sources from different points in the computation tree of the datapath. The third step is based on the algorithm discussed in previous section, by which the output PDF and the corresponding bounds can be calculated.

The proposed method, comparing with the other methods, provides more comprehensive information about the output error, however it has more computation overheads and a more complicated data structure. However, the complexity of the method gives freedom to the designer to adapt the required granularity with available computation resources in an optimization procedure. The following examples provide some results to compare the different methods.

As our first example, consider the following quadratic equation with the input and coefficients error bounds where $x \in [-1, +1]$, $a \in [9, 10]$, $b \in [-4, -6]$ and $c \in [6, 7]$.

$$y = ax^2 + bx + c \tag{5}$$

IA, AA and SNA analysis results are presented in Table (1), where $\epsilon \in [-1, +1]$ and x_l and x_h are calculated for SNA method with different granularity in Table(2). Applying the SNA method also results in the histograms of Figure (1), where different histograms are represented which are calculated with different granularity. Table (2) also presents the mean, variance, lower bound and upper bound for the error calculated by SNA with different granularity.

Table 1: Error range for quadratic equation.

Method	Output Range
IA	$y = [0, 23]$
AA	$y = 6.5 + 16.5\epsilon_y$
SNA	$y = 6.5 + [x_l, x_h]$

This simple example shows how the probability distribution of the noise symbols over the predefined range can affect the final prediction of the error and also how linear combination of the noise symbols might result in overestimated error bounds. It can be seen from Table (2) and Figure (1) that the higher granularity produces higher precision results but with more calculation overheads. This flexibility is especially useful in the optimization process, where the low granularity calculations can be used for preliminary analysis to limit the feasible space of the optimization search.

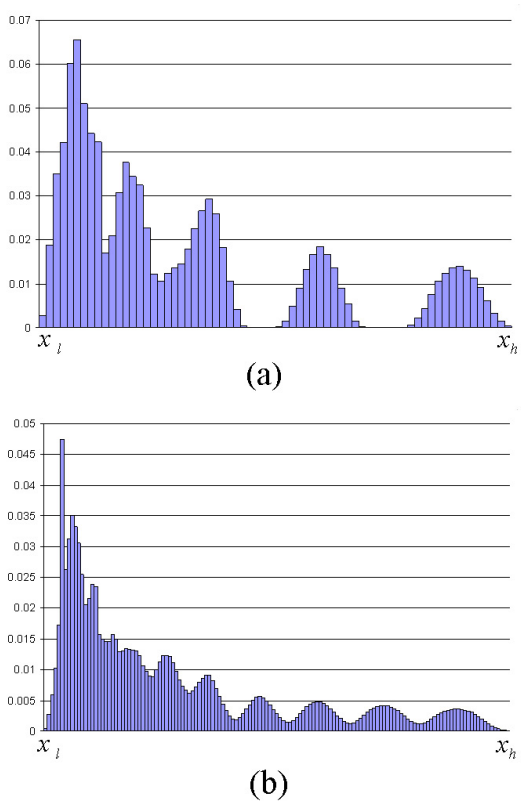


Figure 1: Output error histogram for quadratic equation with different granularities a)g=8 c)g=16.

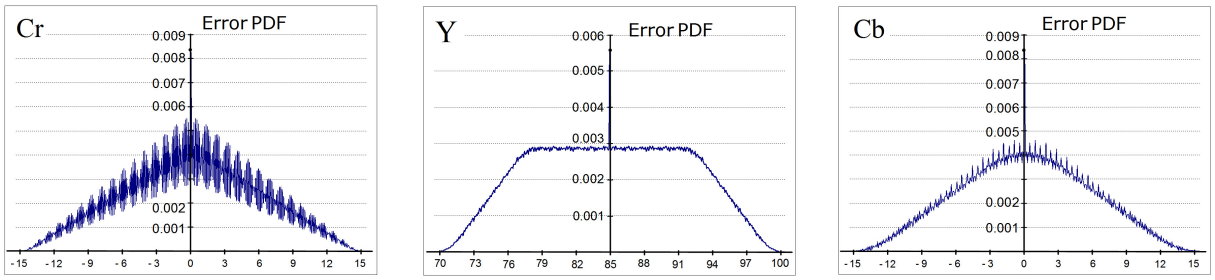
Table 2: Estimated parameters with histogram method for quadratic equation (g=granularity).

g	Mean	Variance	x_l	x_h
2	6.1481	36.274348	0.0	16.0
4	4.6688	28.173507	-1.0	16.5
8	3.9082	22.467426	-1.25	16.5
16	3.5347	19.489651	-1.375	16.5
32	3.3500	18.014349	-1.4375	16.5
64	3.2581	17.286116	-1.46875	16.5
Actual Values	3.17	16.57	-1.5	16.5

The second example is a RGB to YCrCB converter as depicted in Figure (2)[18]. For the sake of comparison, with results of [18], let us assume that the range of the three input signals is $[70, 100]$. Applying the SNA method gives the output error ranges and distributions as shown in Figure (3). Compared with the results of [18] our results are not only more accurate, but also provide more information regarding error distributions over the ranges.

5. RESULTS

Four case studies were implemented with ST 0.12μm technology and applying this method in combination with the high level synthesis method and tools presented in [19] and



(a) (b) (c)

Figure 3: Error PDF for RGB output signals, produced by SNA with g=16.

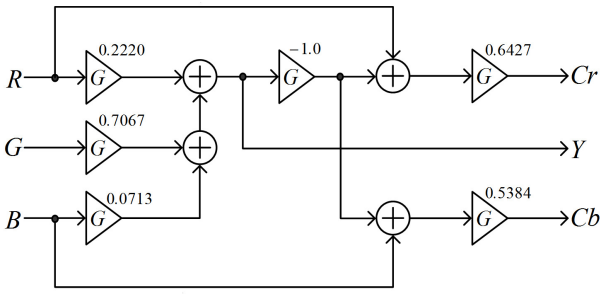


Figure 2: Standard ITU RGB to YCrCb converter.

[20]. Design I is an order-18 difference equation, Design II is a Filter (FIR-25), Design III is an 8-point FFT and Design IV is a DCT 4x4.

Since, in practical implementations, there are pre-defined constraints which must be satisfied and therefore, other costs must be optimized with respect to them, an exhaustive set of synthesis optimizations were performed to show the design cost dependency on WL as a synthesis parameter along with the other classical synthesis parameters.

Tables (3) to (6) show the optimization results for designs I to IV respectively. In these tables, the first column of data gives the basic costs (design area, power consumption, delay and digital noise variance in the output node) for different assumptions of uniform WL (W=8, 16, 24 and 32) in all the design points. This set of information is used as the basis for comparison with other optimization results where noise is constrained to its value in this column. The second column presents results of the WL optimization using our method assuming noise in constrained and third column shows the improvement percentage which have been made after the optimization.

In these tables, area costs are in μm^2, average power consumption cost is presented in μWatt and latency cost (delay) of the design represents the number of clock cycles. As can be observed from the tables, these results show a considerable saving on different costs when our method is applied.

Table 3: Optimization results for Design I.

Word Length	Cost	Fixed WL	Optimized WL	Improv. %
WL=8	Area	4152	3633	12.5
	Power	5672.73	4478.08	21.10
	Delay	168	150	10.74
	Noise	1.03E-2	constrained	
WL=16	Area	8304	7785	6.25
	Power	20779.2	18350.9	11.69
	Delay	311	293	5.79
	Noise	4.04E-5	constrained	
WL=24	Area	12456	11937	4.17
	Power	45753.7	42091.6	8.00
	Delay	454	436	3.96
	Noise	1.58E-7	constrained	
WL=32	Area	16608	16089	3.125
	Power	80602.9	75706.9	6.07
	Delay	598	580	3.01
	Noise	6.16E-10	constrained	

Table 4: Optimization results for Design II.

Word Length	Cost	Fixed WL	Optimized WL	Improv. %
WL=8	Area	22184	20646	6.93
	Power	7143.11	6074.11	14.97
	Delay	58	53	8.62
	Noise	1.28E-2	constrained	
WL=16	Area	44368	43349	2.30
	Power	26929.2	24996	7.18
	Delay	82	79	3.66
	Noise	5.09E-5	constrained	
WL=24	Area	66552	64495	3.01
	Power	59584.1	55273.2	7.24
	Delay	106	101	4.72
	Noise	1.99E-7	constrained	
WL=32	Area	88736	87323	1.59
	Power	105061	100628	4.22
	Delay	130	126	3.08
	Noise	7.62E-10	constrained	

6. CONCLUSION

This study presents a new method of computational error analysis for minimizing the hardware implementation of algorithms by optimizing the word-length of the data in each functional unit. This method, called SNA, provides more comprehensive information about computational error compared with other methods. The proposed method is used in combination with models of power consumption, circuit area and delay. Results from example designs demonstrate a considerable saving in costs when these optimizations are applied.

Table 5: Optimization results for Design III.

Word Length	Cost	Fixed WL	Optimized WL	Improv. %
WL=8	Area	14456	12649	12.5
	Power	9631.03	7572.32	21.38
	Delay	100	99	1.00
	Noise	2.95E-2	constrained	
WL=16	Area	44368	41595	6.25
	Power	35813.4	31676.3	11.55
	Delay	110	107	2.73
	Noise	1.15E-4	constrained	
WL=24	Area	89736	88645	1.22
	Power	78909	76467.4	3.09
	Delay	121	114	5.79
	Noise	4.50E-7	constrained	
WL=32	Area	119648	116178	2.90
	Power	138029	128521	6.69
	Delay	145	135	6.90
	Noise	1.76E-9	constrained	

Table 6: Optimization results for Design IV.

Word Length	Cost	Fixed WL	Optimized WL	Improv. %
WL=8	Area	29912	26889	10.11
	Power	18256.5	16856.9	7.67
	Delay	121	119	1.65
	Noise	3.26E-2	constrained	
WL=16	Area	111344	100915	9.37
	Power	71076.6	69308.4	2.49
	Delay	130	126	3.08
	Noise	1.27E-4	constrained	
WL=24	Area	174744	163027	6.71
	Power	156085	147610	5.43
	Delay	152	145	4.61
	Noise	4.97E-7	constrained	
WL=32	Area	222688	219987	1.21
	Power	273138	265048	2.96
	Delay	178	168	5.62
	Noise	1.94E-9	constrained	

7. REFERENCES

[1] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens, "A methodology and design environment for DSP ASIC fixed point refinement," in *DATE'99*, 1999, p. 56.

[2] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Trans. CAD*, vol. 20, no. 8, pp. 921–930, 2001.

[3] G. A. Constantinides, "Word-length optimization for differentiable nonlinear systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 1, pp. 26–43, 2006.

[4] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis and Optimization of DSP Algorithms (Fundamental Theories of Physics S.)*. Kluwer Academic Publishers, 2004.

[5] C. Shi and R. W. Brodersen, "A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs," in *ISCAS'04*, vol. 3, 2004, pp. III–373–6.

[6] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs," in *DATE'01*, 2001, pp. 722–728.

[7] S. Roy and P. Banerjee, "An algorithm for trading off quantization error with hardware resources for MATLAB-based FPGA design," *IEEE Trans. Comput.*, vol. 54, no. 7, pp. 886–896, 2005.

[8] B. Le-Gal, C. Andriamisaina, and E. Casseau, "Bit-width aware high-level synthesis for digital signal processing systems," in *IEEE International SOC Conference*, 2006, pp. 175–178.

[9] K. Han and B. L. Evans, "Wordlength optimization using sensitivity information," *EURASIP Journal on App. Sig. Proc.*, no. 5, pp. 1–14, 2006.

[10] N. S. Nedialkov, V. Kreinovich, and S. A. Starks, "Interval arithmetic, affine arithmetic, taylor series methods: Why, what next?" *Numerical Algorithms*, vol. 37, no. 1-4, pp. 325–336, 2004.

[11] D.-U. Lee, A. Abdul-Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. CAD*, vol. 25, no. 10, pp. 1990–2000, 2006.

[12] Y. Pu and Y. Ha, "An automated, efficient and static bit-width optimization methodology towards maximum bit-width-to-error tradeoff with affine arithmetic model," in *ASP-DAC'06*, 2006, pp. 886–891.

[13] C. F. Fang, R. A. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *ICCAD'03*, 2003, pp. 275–282.

[14] N. Doi, T. Horiyama, M. Nakanishi, and S. Kimura, "Bit-length optimization method for high-level synthesis based on non-linear programming technique," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E89-A, no. 12, pp. 3427–3434, 2006.

[15] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Pearson US Imports and PHIPEs, 1999.

[16] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: a fixed-point design and simulation environment," in *DATE'98*, 1998, pp. 429–435.

[17] D. Berleant, "Automatically verified reasoning with both intervals and probability density functions," *Interval Computations*, no. 2, pp. 48–70, 1993.

[18] J. A. Lopez, C. Carreras, and O. Nieto-Taladriz, "Improved interval-based characterization of fixed-point LTI systems with feedback loops," *IEEE Trans. CAD*, vol. 26, no. 11, pp. 1923–1933, 2007.

[19] A. Ahmadi and M. Zwolinski, "Multiple-width bus partitioning approach to datapath synthesis," in *ISCAS'07: IEEE International Symposium on Circuits and Systems, 2007.*, May 2007, pp. 2994–2997.

[20] ——, "A symbolic noise analysis approach to word-length optimization in DSP hardware," in *ISIC'07: International Symposium on Integrated Circuits*, September 2007, pp. 497–500.