# AN AUTOMATED FLOATING-POINT TO FIXED-POINT CONVERSION METHODOLOGY

*Changchun Shi and Robert W. Brodersen*

Berkeley Wireless Research Center, Department of EECS, University of California, Berkeley, USA

**ABSTRACT**

This work proposes a floating-point to fixed-point conversion (FFC) methodology for digital VLSI signal processing systems. The past techniques used to facilitate FFC are first reviewed, followed by a description based on a statistical approach and global optimization which allows a high degree of automation.

## 1. INTRODUCTION

The algorithms used by communication systems are typically specified as floating-point (infinite precision) operations. On the other hand, digital VLSI implementations of these algorithms rely on fixed-point (finite precision) approximations to reduce cost of hardware while increasing throughput rates. One essential step of top-down design flow is to determine the fixed-point data type of each signal node, namely the word-length, truncation mode and overflow mode [1-7]. However conventional approaches are typically both time-consuming and error-prone since ad-hoc assignments of fixed-point data type are performed manually and iteratively [3].

In section 2 the FFC problem is first formulated into an optimization framework. The optimization variables are defined by the fixed-point data-types to be determined. In this unified point of view the past techniques are compared. A primary goal is to make the optimization automatic and fast which required an understanding of the relationships between optimization functions and variable. A critical step is the identification of the right metric that judges the quality of an FFC that is sufficiently general.

## 2. PROBLEM FORMULATION OF FFC

The first part of this section formulates FFC as an optimization problem. The drawbacks of some past techniques are discussed in this unified framework in the second sub-section.

### 2.1. Optimization Formulation

The optimization variables are the fixed-point data-types to be determined, including integer word-lengths $W_{Int,i}$, fractional word-lengths $W_{Fr,i,}$ overflow modes $o_i$, and quantization modes $q_i$. In our formulation word-lengths should be integers; overflow mode can be 0 (wrap-around) or 1 (saturation); quantization mode can be 0 (2's complement truncation towards lower value) or 1 (round-off). The objective to be minimized is the hardware cost as a function of these variables, with the constraint functions as specifications that depend on the behavioral quality of the fixed-point system, such as bit-error rate or signal-to-noise ratio. In general, longer word-lengths improve (or keep) the system performance with an increase in hardware cost. Putting all of the functions together, FFC can be transformed into the following optimization problem

minimize hardware-cost

$f(W_{Int,1}, W_{Fr,1}; W_{Int,2}, W_{Fr,2}; \ldots; o\text{-modes}, q\text{-modes})$

subject to specifications

$S_j(W_{Int,1}, W_{Fr,1}; W_{Int,2}, W_{Fr,2}; \ldots; o\text{-modes}, q\text{-modes}) < 0, \forall j$

stopping criteria

$f < (1 + a) f_{opt}$ where $a > 0$.

The stopping criteria means the optimization iteration stops when the hardware-cost $f$ is within a small difference $a$ away from the optimal hardware-cost $f_{opt}$. In section 3 the choices of hardware-cost function and specification functions will be discussed in detail. In general the constraint functions should be chosen such that they are feasible, i.e. there exists a point in the optimization space that satisfies all the constraints.

### 2.2. Review of the past techniques

Recently a few strategies have been proposed to automate FFC for communication systems described in C/C++ [2-4]. While the integer-part word-length and overflow modes of the fixed-point operands are commonly determined by avoiding signal overflow, the determination of the

fractional word-lengths relies on different methods. In both [2] and [3] there is no gross hardware-cost function given; neither is the problem treated as an optimization. However the implicit goal is to minimize all the word-lengths at the same time. The task of minimizing multiple objective functions is unrealistic unless the constraints are special so that they all can be minimized simultaneously. This is done in [3] by having one constraint function for each word-length: the input word lengths are pre-assigned; for the rest of the word lengths the integer part should be sufficiently large to cover the signal ranges it governs (same for both [2] and [4]), and the fractional part should be sufficiently large so the local quantization noise power is much smaller than the one caused by quantizations of the inputs. These strongly decoupled constraint functions are always feasible and can minimize all word lengths at the same time. However, the gross quantization effects from these locally justified quantization sources altogether can still be much greater than the one induced by input quantizations. Therefore it is still necessary to have a final constraint on system performance (e.g. SNR or bit-error rate) as a function of all the word lengths. This becomes the reason for unbounded number of iterations.

In [2] the unjustified pre-assignments of date-types on certain signal nodes provide a number of constraint equations. The deterministic propagation methodology yields inequalities among the fractional word lengths, e.g. the fractional word length at the output of a multiplier should be no less than the sum of those of the two inputs, while the output fraction word length of a delay component should be no less than the input one. Besides the overly pessimistic consideration of quantization effects, feedback loops such as the one in an accumulator can yield contradictive inequalities. This is solved in [2] by possible user interaction using engineering decisions, which results in undetermined design time.

Work in [4] implies similar problem formulation to ours, and again has the same treatment on integer word lengths. The constraints are chosen to be the system specification functions directly. However the lack of investigations on the closed form specification function limits their optimization algorithm to be heuristic and time-consuming search. In addition, the Monte Carlo simulations among iterations can be inconsistent which adds further complications.

## 3. PRACTICAL, RELIABLE AND COST-EFFICIENT FFC

The same constraint functions as in [2-4] are adopted in the current work for integer word lengths. The assumption that overflow noise hurt the quality of the design greatly is reasonable in general. However be aware that in some special situations occasional overflows in saturation mode are acceptable and even expected, which won't be covered in this work. Only one behavioral simulation is needed to estimate the ranges of signal nodes. With all $W_{Int}$ and *o*-modes can be determined separately, these variables are dropped out from the optimization problem in Section 2.1. In the following subsections the hardware-cost function and constraint functions will be studied.

### 3.1. Analytical form of hardware-cost function

Only one hardware cost function is to be minimized. This could be area, power consumption, etc. High-level estimations of hardware resources such as area, energy and delay have been studied extensively, e.g. [1,7]. For system level optimization, it often suffices to adopt the parameterized library based approach [1]. The area of each block of the library can be modeled as a function of parameters related to fixed-point data types as well as other important technology factors such as feature size and voltage. Provided the architecture choice with all other parameters fixed, the area cost of a library block is uniquely characterized as a function of the fixed-point data-type parameters. The total area of the system can then be estimated as a sum of all the required blocks plus a certain routing overhead. This usually yields a hardware-cost that is a quadratic function of $W_{Fr,i}$ and $q_i$.

### 3.2. Choices and analytical forms of constraint functions

Unlike in [3] where a number of additional constraint functions are created to solve the multiple-objective-function situation, only the system specifications (such as bit-error rate and SNR) that are eventually used to judge the quality of the design are first considered as the constraints. Furthermore instead of employing the system specifications directly as the optimization constraints as used in [4], a more robust specification scheme is proposed based on the statement that the fixed-point system is expected to deviate only little from the floating-point origin. One natural alternative specification replacing (NOT in addition to) the system specifications is their relative changes between floating and fixed point systems. An innovative perturbation theory is developed, and shows that the change to the first order is a linear combination of all the first and second-order statistics of the quantization noise sources. With the widely used theoretical models of quantization noises [5,6], this alternative specification function can further be written into closed form

$$|\text{sys.spec.(fxpt)} - \text{sys.spec.(flpt)}| = \overline{\mathrm{M}}^{\mathrm{T}}\,\overline{m} + \sum_{i\in\{\text{Data Path}\}} e_i\, 2^{-2W_{\mathrm{Fr},i}} + \ldots,$$

where $e_i$'s are constants and $\overline{\mathrm{M}}$ is a constant column vector, the $i^{\text{th}}$ entry of vector $\overline{m}$ is

$$m_i = \begin{cases} -\frac{1}{2} q_i 2^{-\mathrm{W_{Fr},}\,i} & \text{for datapath} \\ \mathrm{fxpt}(c_i, 2^{-\mathrm{W_{Fr},}\,i}) - c_i & \text{for constant } c_i \end{cases}, \quad q_i = \begin{cases} 0, \text{if round-off} \\ 1, \text{truncation} \end{cases}$$

as defined in Section 2.1, function fxpt($c$, $d$) means the value among the set {integer × $d$} that is the closest to $c$, and $c_i$'s are the constants (e.g. filter coefficients) that appear in the floating-point design.

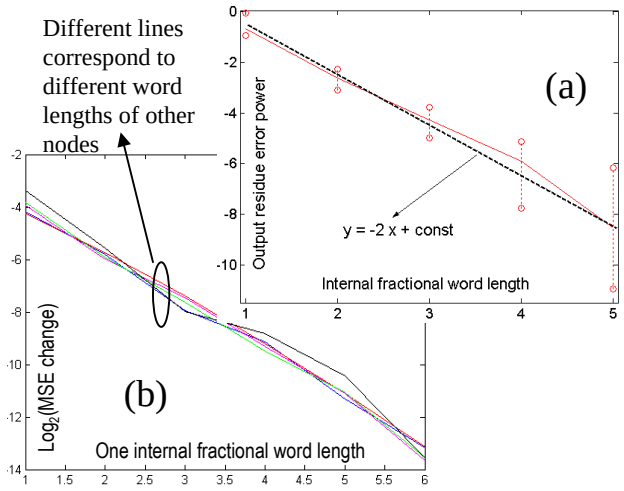


Figure 1. Simulation results of a 12-tap LMS filter. Both the alternative specification (in a) and MSE specification (in b) follow the perturbation theory. Note y-axis in b is the change of MSE, instead of MSE, in order to show the linear characteristics.

Moreover this perturbation theory works on general criterions, as long as they can be represented as large ensemble averages of functions of the signal outputs. For example, convergence time of a LMS system also follows the perturbation theory above.

The linear coefficients $\overline{\mathrm{M}}$ and $e_i$'s can be data-fitted by running polynomial numbers of Monte-Carlo simulations. However an unacceptably large sample size may be needed for each simulation to detect the small perturbation of a large value, who suffers its own estimation error. This important issue is resolved by choosing the mean-squared error (MSE) as the specification function. The MSE is the output difference between the floating-point and the fixed-point system. Following the perturbation theory, it can be written as

$$\mathrm{MSE} = \overline{m}^T \mathrm{B}\overline{m} + \sum_{i \in \{\text{Data Path}\}} \mathrm{C}_i 2^{-2\mathrm{W_{Fr},}\,i},$$

where B is a positive semi-definite matrix, denoted as $\mathrm{B} \succeq 0$, and $\mathrm{C}_i \geq 0$. Simulations are generated for a least-mean-square adaptive filter (LMS), as shown in Figure 1 (a) and (b). It verifies that both the alternative specification and MSE follow the provided formula. Since MSE directly measures the deviation between floating-point and fixed-point systems, the sample size to estimate MSE is indeed orders of magnitude (4 orders in this LMS example in Figure 1) less than that to estimate the perturbation of a general specification, which in this case is the output residue error power. Essentially similar results on the LMS are obtained in [5] by complicated analysis, which also provides the analytical expressions for the constant B and C matrices. However it is clear pure analysis is limited to special cases only. In stead our perturbation method applies in general, with the expressions of B and C left to Monte-Carlo simulations.

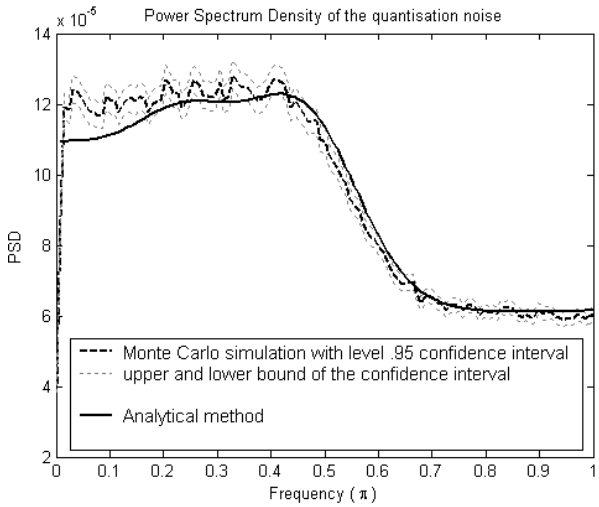


Figure 2. Comparison between analytical form and Monte-Carlo simulation for biquad system.

A separate study on general multiple-input-multiple-out linear-time-invariant (MIMO LTI) systems based on transfer function method has been conducted in [6] and yield the following result:

$$\mathbf{R}_{\overline{O}\overline{O}} = \Delta\mathrm{H}_{YX} \cdot \mathbf{R}_{XX} \cdot \Delta\mathrm{H}_{YX}^{\mathrm{H}} + \mathrm{H}_{YQ} \cdot \mathbf{R}_{QQ} \cdot \mathrm{H}_{YQ}^{\mathrm{H}},$$

where $\mathbf{R}_{\overline{O}\overline{O}}$ is the MSE matrix at the multi-dimensional outputs at a given frequency; $\mathbf{R}_{XX}$ and $\mathbf{R}_{QQ}$ are the power spectrum matrices of inputs and quantization sources, respectively; and H and $\Delta$H are the transfer function matrix and its perturbation due to quantizations of coefficients (see [6] for details). A closer inspection shows that the formula above gives exactly the MSE formatted early in this sub-section; in addition it provides the analytical expression of B and C matrices. Based on the LTI theory, a simple Biquad IIR system is examined. Theoretical result of the power spectrum density at the output matches very well to the simulation result as shown in Figure 2. This verifies the LTI theory above and therefore the perturbation theory.

Now the FFC problem is safely reduced to

minimize

$$\text{Quadratic } f(\mathrm{W}_{\mathrm{Fr},1}, \mathrm{W}_{\mathrm{Fr},2}, \ldots; q_1, q_2, \ldots)$$

subject to

$$\bar{m}^T \mathrm{B}_k \bar{m} + \sum_{i \in \{\text{Data Path}\}} \mathrm{C}_{i,k} 2^{-2\mathrm{W}_{\mathrm{Fr},i}} - A_k < 0,$$

$$\text{with } \mathrm{B}_k \succeq 0, \mathrm{C}_{i,k} \geq 0, \text{and } A_k > 0.$$

Here vector $\bar{m}$ is defined in the same way as before, and $A_k$ is the tolerance of the $k^{\text{th}}$ MSE error. The problem is feasible because as all $\mathrm{W}_{\mathrm{Fr}}$ increase, the left sides of the constraint functions asymptotically converge to $-A_k$'s which are always less than 0. Physically that means the fixed-point system becomes infinite precision.

### 3.3. Design automation

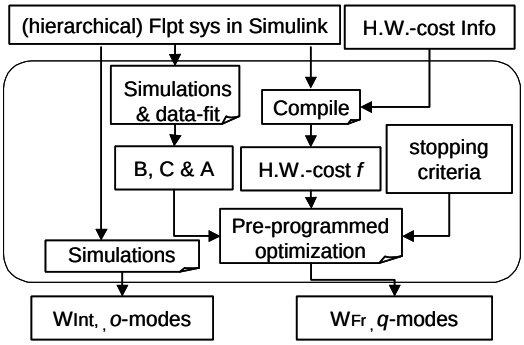


Figure 3. FFC design flow graph.

An essential part of a practical FFC is to automate the process of obtaining the analytical hardware cost function and the analytical specification function. This is achieved following the design flow in Figure 3. First the signal ranges need to be estimated automatically by running one simulation. This simulation also provides us the MSE tolerance vector, A. Secondly, a number of simulations can be conducted following the MSE formula to find out matrix B and C. The analytical hardware-cost function can be achieved by automatically reading the system parameters, provided the hardware-cost formula for each block. Our current design environment is Mathwork Simulink™, in which all the tasks above can be automated. The number of Monte-Carlo simulations need to be done is approximately $[\dim(\mathrm{B})^2 + \dim(\mathrm{C}) + 1]$. Finally the optimization algorithm specifically suitable for this problem should be preprogrammed.

### 3.4. Scalability

A partition of the system MSE specification into block-wise MSE specifications can factorize the problem into smaller optimization problems. Moreover as in [2], many of the word lengths along forward-directional data path can be pre-related to reduce the number of optimization variables. These two strategies ensure the applicability of the proposed methodology on large systems.

## 4. CONCLUSION

This work formulates FFC into an optimization problem. The past techniques are reviewed in this new scenario. An automatic FFC methodology is presented and validated. The work provides new insights on finite-word-length effects in a stochastic environment. Besides digital ASIC communication systems, the approach is readily applicable to other statistical signal processing systems and hardware platforms (e.g. FPGA).

## 5. ACKNOWLEDGEMENTS

This work was sponsored by DARPA and the SIA under the MARCO focus centers program as well as the sponsors of the Berkeley Wireless Research Center.

## 6. REFERENCES

[1] N. Zhang, B. Haller, and R. W. Brodersen, "Systematic architecture exploration for implementing interference suppression techniques in wireless receivers," *Proc. IEEE Workshop on Signal Processing Systems*, LA, October 2000.

[2] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: a fixed-point design and simulation environment," *Proceedings of Design, Automation and Test in Europe*, pp. 429 –435, 1998.

[3] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens, "A methodology and design environment for DSP ASIC fixed point refinement", *Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings*, pp. 271 – 276, 1999.

[4] W. Sung, and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Processing*, vol. 43, no. 12, Dec. 1995.

[5] J. Cioffi, and M. Ho, "A finite precision analysis of the block-gradient adaptive data-driven echo canceller," *IEEE Trans. Communications.* vol. 40, no. 12, May 1992.

[6] Changchun Shi, "Statistical method for floating-point conversion," 2002, Master Thesis, Department of EECS, Univ. of California, Berkeley. (Advisor: Robert W. Brodersen).

[7] M. Nemani, and F. N. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Tran. Computer-Aided Design of Integrated Circuits and Systems*, vol 18, pp. 697-713, June 1999.