Automated Fixed-point Data-type Optimization Tool for Signal Processing and Communication Systems

Changchun Shi, Robert W. Brodersen
University of California at Berkeley
{ccshi, rb}@eecs.berkeley.edu

ABSTRACT
A tool that automates the floating-point to fixed-point conversion (FFC) process for digital signal processing systems is described. The tool automatically optimizes fixed-point data types of arithmetic operators, including overflow modes, integer word lengths, fractional word lengths, and the number systems. The approach is based on statistical modeling, hardware resource estimation and global optimization based on an initial structural system description. The basic technique exploits the fact that the fixed point realization is a weak perturbation of the floating point realization which allows the development of a system model which can be used in the optimization process.

Categories and Subject Descriptors
C.3 [**Special-Purpose and Application-Based Systems**]: Signal Processing Systems; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design

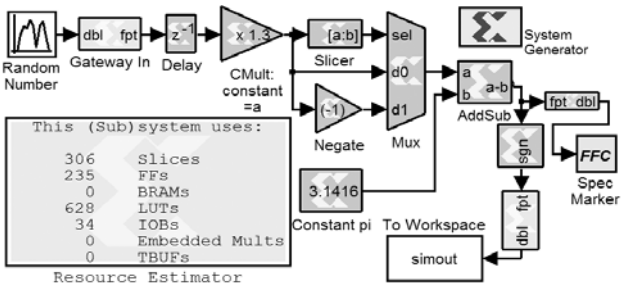
General Terms
Design, Algorithms, Performance

Keywords
Fixed-point arithmetic, optimization, communication systems, digital signal processing, FPGA, ASIC.

1. INTRODUCTION
Recently, the advances of software and hardware co-design environments enable us to capture architectural information at an early design stage [e.g. 1-6] and conduct bit-true and cycle-accurate simulations. The verified design can then be mapped to hardware automatically. A number of these platforms that are based on Simulink[1] have been successful, targeting either FPGA's, such as in System Generator[2] from Xilinx [2,3], or ASIC's [1,4]. Fig. 1 shows such an algorithm description in both algebraic form and structural form that is implemented using System Generator blocks. If 32-bit word-lengths are specified throughout the design, the numerical precision is typically sufficiently high that the structural form is verified to perform the target algebraic functionality without errors from finite wordlength effects, i.e. it is essentially a floating point description. A resource estimation tool based on the Simulink description has been developed that automatically calculates FPGA hardware requirements, such as slices, used in the design. For ASIC designs, a similar tool could be developed which estimates the chip area requirements based on the number of cells used in the block realization. To reduce these hardware-costs, increase the throughput, and reduce power, one essential step is to optimize the fixed-point data-types to use the minimum word lengths possible. This task will be referred to as floating-point (meaning high-precision) to fixed-point conversion (FFC).

$$y(n) = \mathrm{sgn}(|\, a \cdot x(n-1)\,| - \pi)$$

(a)



(b)

Figure 1. (a). The algorithm in algebraic form; (b). A high-precision architectural version with the output of the hardware "Resource Estimator".

For many application domains and algorithms high precision in the computation is wasteful and significant hardware reductions are possible. However, determining the minimum requirements through manual optimization is time-consuming and rarely optimal [5]. To automate this task, a statistical perturbation theory approach was taken which uses an accurate hardware-cost model, and has been applied to FPGA implementations. As seen in Fig. 1, the designer inserts an FFC utility block, "Spec Marker", from the FFC library into the system at critical nodes, which is used to enter user specifications of the system performance and the tool then automatically produces a system with optimized fixed-point data types. Though implemented targeting FPGA's, the same methodology may be applied to other hardware, such as ASIC's, provided a hardware-cost estimator is available.

[1] The name Simulink is registered with Mathworks, Inc.

[2] The name System Generator is registered with Xilinx, Inc.

DAC 2004, June 7–11, 2004, San Diego, California, USA.

Section 2 gives a more detailed formulation of the FFC task, followed by a brief review of existing techniques. Section 3 describes our strategy with emphasis on the techniques used in improving efficiency of the automated optimization. Sections 4 and 5 compare the performance of this approach on several typical designs with previously described optimization techniques.

2. PROBLEM FORMULATION AND REVIEW OF EXISTING TECHNIQUES

During signal processing system development, designers typically develop test vectors to evaluate whether the algorithms and architecture are meeting system specifications under the assumption of infinite precision computation. These specifications are often based on data statistics, such as output bit-error rate (BER) or signal-to-noise ratio (SNR), but can be more complex such as convergence time of the algorithm or frequency response requirements. Therefore, the task of FFC is to minimize the hardware using these system test vectors to achieve system performance which is within some small "implementation loss" relative to the ideal realization. Therefore the goal is to solve the optimization problem formulated in Eq. (1):

$$\begin{aligned} &\text{minimize hardware - cost} \\ &\quad f_{HW}(W_{Int,1}, W_{Fr,1}, W_{Int,2}, W_{Fr,2}, \ldots; o_1, o_2, \ldots; n_1, n_2, \ldots) \\ &\text{subject to specifications} \\ &\quad S_j(W_{Int,1}, W_{Fr,1}, W_{Int,2}, W_{Fr,2}, \ldots; o_1, o_2, \ldots; n_1, n_2, \ldots) < 0, \forall j, \end{aligned} \tag{1}$$

where

f_{HW}	hardware-cost function
S_j	the j^{th} system behavioral constraint function
$W_{Int,i}$, $W_{Fr,i}$	integer and fractional word length (always integer)
o_i	overflow mode: wrap-around or saturation
n_i	number system: 2's complement or unsigned magnitude

Round-off is assumed for quantization with the possibility of extending to truncation as described in Section 6. The optimization problem as presented here is always feasible [7].

A few recent strategies that fully or partly automate FFC have been previously proposed and compared [5-6,9]. Among them, it appears from the comparisons that the most promising are those based on simulation with the assumption that increasing word-length improves system performance and increases hardware-cost, but otherwise are unguided. A critical drawback of this method is the simulation time which can be unreasonably large for complex systems and thus sub-optimal solutions are often obtained. Furthermore, several of these approaches use a system description that doesn't contain architectural information, which compromises their results, since the architecture free high-level hardware-cost estimations are often very crude.

We will also use simulations to evaluate various sensitivities, but will use these results to develop a model of the system which is then used in the optimization [6]. This allows us to dramatically reduce the amount of simulation required.

A somewhat similar approach was taken that uses a graphical translator that directly uses the structure of the system to obtain a transfer function for the quantization noise [14]. However, the approach is limited in its applicability, e.g. it does not apply to nonlinear feedback systems and also neglects the correlation of the inputs of an operator. This latter limitation may cause unacceptable errors, since the multiple inputs of an operator may originate partly from a single quantization source earlier in the system.

By analyzing and categorizing signal types and operator types according to their quantization effects, a statistical perturbation approach has been developed that relates quantization effects to the fractional word-lengths under very limited constraints [10-12]. These analyses provide the foundation of the FFC method described here which can provide a global optimization with a relatively small amount of simulation time. Critical tool implementation issues that will be described here involve the development of an accurate hardware-cost function, operator grouping, the optimization techniques and their robustness and a comparison with existing FFC tools.

3. AUTOMATING FLOATING POINT TO FIXED-POINT CONVERSION (FFC)

3.1 Optimization strategy

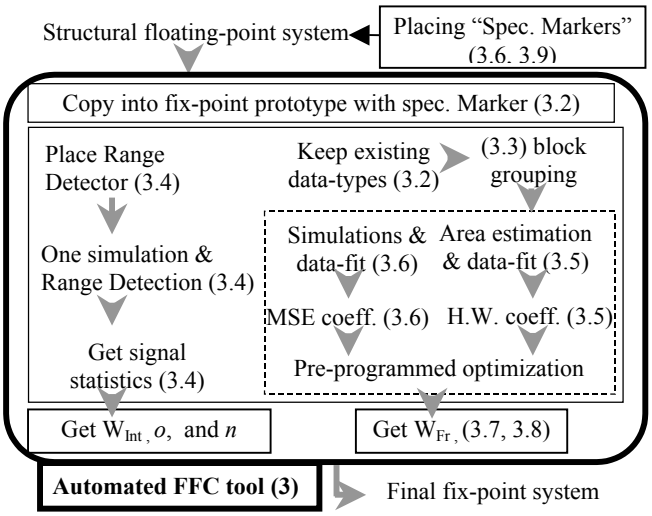


Figure 2. The FFC tool automated flow.

In Fig. 2, the flow of the various components of the FFC system is shown along with the sub-section number which describes each function. The user enters a structural description of the architecture along with "Spec. Markers" that are used with the designers test vectors to verify adequate system performance after optimization. The remaining functions within the bold line are the flow of the automated tool. The following sections describe each step in detail.

3.2 Keep useful fixed-point information

The first task of our FFC is to copy the floating-point design into a temporary prototype system. This ensures the subsequent modifications by the FFC tool leave the original system untouched.

The initial high-precision system may contain many signals whose fixed-point data types are already specified. We call these

"fixed" signals [11-12], and are considered outside the optimization process. For example, Fig. 1 shows that the Select port of the MUX is a 1-bit unsigned integer and no change is needed (or desired) on such a signal. An analysis of the library blocks is therefore initially undertaken to define the obvious fixed signals, but even some arithmetic signals, which have been pre-defined (such as those determined by availability of existing hardware) are also considered "fixed" and left untouched. These signals can be simply tagged by the system designer, or configured through an additional rule, which for example may limit the signal data-types to be greater than a minimum value and when that minimum level is reached in the optimization are set to fixed status. The remaining signals are all considered "arithmetic" and subject to be changed freely by the FFC tool.

3.3 Block grouping

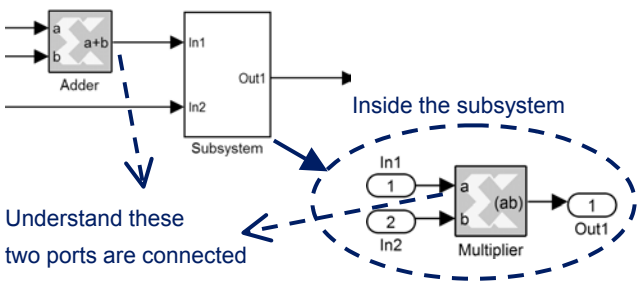


Fig. 3. The importance of resolving block connectivity.

The search space is then reduced further by grouping blocks together and assigning them the same fractional word-lengths [6,10]. For example, the two signal inputs d0 and d1 of the Mux in Fig. 1 and the Mux output are set to have the same fixed-point data type, since there would be little hardware savings if these numbers were left independent. This means the block in front of the two signal inputs and the Mux itself may be grouped together to have the same fractional word-length. Grouping is simply a way to reduce the search variables by adding constraints to the optimization problem (1).

In a block-diagram based environment, we differentiate blocks as being either *functional* or *supportive*. Fig. 3 shows that both an Adder and a Multiplier are functional since they are essential to accomplish the signal processing computation. The Subsystem block and the associated In and Out blocks are supportive to the hierarchical description in that they don't directly perform computation. Before block grouping, the connectivity among functional blocks needs to be resolved, which basically flattens the design hierarchy. This is important since grouping involves frequent cross-references of adjacent functional blocks.

3.4 Integer overflow

For dealing with overflow a conventional approach is followed that acquires information about the dynamic range at each internal node. We briefly repeat the strategy and describe how to implement it in our environment. Monte-Carlo simulations of the floating-point system with the input test vectors provide the statistics for each signal node which is used to determine W_{Int}, n and o in Eq. (1). This effectively separates all integer fixed-point data types in the optimization problem into individual optimization problems to be solved independently.

An FFC library block, "Range Detector" is used to automate the proceeding process by simply introducing it into the graphical system description. This block performs running-average estimates of the first four moments of its input during a simulation. Internally, only the current averages and sample size—a total of five numbers—are saved for each Range Detector during the simulation. After the signal statistics are estimated, the FFC tool automatically removes all these Range Detectors during the remaining optimization.

3.5 Analytical hardware-cost function

Only one hardware cost function is minimized in Eq. (1). This could be area, power consumption, power delay product, or other more complex functions [13]. For example, given the architecture choice and technology parameters, the area of a block can be uniquely characterized as a function of its fractional word-lengths. For a system that consists of arithmetic units such as adders, MUXs, multipliers, constant multipliers, and so on, the hardware-cost is simply modeled as a quadratic function of W_{Fr}, a vector formed by all different fractional word-length group. That is,

$$f_{HW}(W) \approx \frac{1}{2} W_{Fr}{}^{T} \mathrm{H}\, W_{Fr} + h^{T} W_{Fr} + h_0. \qquad (2)$$

The hardware estimator using FPGA's as a target operates at the pre-netlisting stage and can perform an adequately accurate estimate (<10%) taking on the order of seconds [3]. This is used to function-fit the coefficients of f in (2). f_{HW} is an affine function of H, h, and h_0, and can be written as $f_{HW}(W_{Fr}) = W_{long}^{T} \mathrm{H}_{long}$, where the long column vector H_{long} captures all the independent entries of symmetric matrix H, vector h, and scalar h_0,

$$\mathrm{H}_{long} = (\mathrm{H}_{11},...,\mathrm{H}_{1m},\mathrm{H}_{22},...,\mathrm{H}_{2m},....,\mathrm{H}_{mm},h_1,...h_m,h_0)^{T}.$$

H_{long} has dimension $[(m\ (m+1))/2 + m + 1] \times 1$ where m is the number of fractional word-length groups. W_{long} is the corresponding column vector that formed by the quadratic and linear combinations of the scalar entries of vector W_{Fr}, as well as a constant for h_0. For example, a few entries of W_{long}, corresponding to H_{11}, H_{1m}, where $m>1$, h_m and h_0 are depicted as following

$$W_{long} = (W_{Fr,1} \times W_{Fr,1},...,2 \cdot W_{Fr,1} \times W_{Fr,m},....,W_{Fr,m},1)^{T}.$$

One way to determine H_{long} is to solve the least-squared problem which minimizes the 2nd order norm of relative errors—$\underset{\mathrm{H}_{long}}{\text{minimize}} \left\| 1 - \hat{W}^{T} \mathrm{H}_{long} \right\|_2$, where matrix $\hat{W}$ is a stack of W_{long}, denoting $(\hat{W}_{long,1}, \hat{W}_{long,2},...)$, each of which is a different realization of vector W_{Fr}, normalized by its corresponding hardware-cost estimated by the resource estimator tool. The condition that the hardware-cost is monotonically increasing and non-negative translates to the condition that each entry of H_{long} is non-negative, that is, $\mathrm{H}_{long} \geq 0$. So, the estimation of H_{long} becomes a Quadratic programming problem [7]. In Matlab, lsqnonneg($\hat{W}^{T}$, 1) solves it efficiently.

3.6 Analytical specification functions

To solve the optimization problem (1), we need to repeatedly verify the constraint functions. This is normally performed using digital simulations. The total time required for these verifications is the number of optimization iterations multiplied by the average

simulation time for each one. Both of these factors are used to reduce the total simulation time.

3.6.1 Directly use the difference as a specification

The most common specifications used in communication systems are signal-to-noise ratio (SNR) and bit-error rate (BER) and for typical systems the noise generated by quantization effects and the BER can be quite small values. For accurate estimates a large number of samples are therefore required, which is further increased during optimization since the performance of two realizations must be compared to determine the direction for parameter optimization.

To reduce this sensitivity, we directly measure the system degradation caused by the fixed-point implementation in comparison to the infinite (high) precision solution for the same input signal. This difference specification is then used in the optimization and greatly reduces the required number of simulation iterations.

SNR specifications are directly related to the mean squared error (MSE) of the difference between floating-point system and fixed-point system, denoted as MSE(flpt-fxpt). Even for BER type specifications, defined at the output of strong decision-making blocks (a slicer), are also directly related to this MSE error [12] (some exceptions to this will be mentioned in Section 3.9). A Specification Marker block from the FFC library is manually inserted into the floating-point system at the appropriate places and it calculates the MSE during the optimization [12].

It is fundamental to the approach that the fixed-point implementation is a close approximation of the floating-point system which allows the use of perturbation theory which linearizes the MSE dependencies. The MSE(flpt-fxpt) as an explicit function of fixed-point data types in the system [10-11] is given by,

$$\mathrm{MSE(flpt-fxpt)} = \overline{\mu}^T \mathrm{B}\overline{\mu} + \sum_{i\in\{\text{Data Path}\}} \mathrm{C}_i 2^{-2\mathrm{W}_{\mathrm{Fr},i}}, \qquad (3)$$

where unknown coefficient matrix B is a positive semi-definite, unknown coefficient $\mathrm{C}_i \geq 0$, and the i^{th} entry of vector $\overline{\mu}$ is $\mathrm{fxpt}(c_i, 2^{-\mathrm{W}_{\mathrm{Fr},i}}) - c_i$ for constant c_i—an example of such constant is parameter *a* in Fig. 1. Function fxpt(*c*, *d*) uses the value among the set {integer × *d*} that is the closest to *c*. This approach is quite general and applies in non-linear systems with even non-stationary inputs. With Eq. (3), it is possible to find the analytical function between this MSE and the fixed-point data-types. The number of simulations is significantly less than unguided characterization in which the form in Eq. (3) is not assumed.

3.6.2 Use an ergodic average

A statistical expectation of an output at time *n* can be estimated based on a large-ensemble average which requires many repeated simulations. Fortunately, a random process at the output is often locally stationary; so that we can use an ergodic average - based on different output samples in one simulation—to estimate the ensemble. This can save orders of magnitude of simulation time and has been found to be adequate.

3.7 Optimization step

Since pre-grouping of operators is done to reduce the number of simulations in Section 3.3, many constant inputs are often grouped together. These multiple quantization sources in each of the constant-group behave similar to a random noise with 0-mean and variance proportional to $2^{-2\times \mathrm{WFr}}$. With this simplification and the results in Section 3.5 and 3.6, (1) becomes

$$\begin{aligned} &\underset{\mathrm{W_{Fr,i}}\geq 0}{\text{minimize}}: \quad f(W_{Fr}) = \frac{1}{2} W_{Fr}^T \,\mathrm{H}\, W_{Fr} + h^T \cdot W_{Fr} + h_0 \\ &\text{subject to}: \sum_{i\in\{\text{Data Path and constant - groups}\}} \mathrm{C}_{i,k} 2^{-2\mathrm{W_{Fr,i}}} - A_k < 0, \qquad (4) \\ &\qquad \text{with } \mathrm{C}_{i,k} \geq 0, \text{and } A_k > 0; i = 1,2,...,m; k = 1,2,....,\mathrm{K} \end{aligned}$$

Because all the groupings in Section 3.3 are done automatically, no ambiguity arises in the definition of index *i* in (4) as it is just an ordering of all the word-length groups. The dimension of *k*, denoted as K, specifies the number of critical nodes to place Spec Markers, which is usually less than 100. In Fig. 1, K is just 1.

The constraints of problem (4) represent a joint set of sublevels of exponential-sum convex functions; thus it is a convex set. Unfortunately, H is usually not positive semi-definite; so, (4) is not a convex optimization problem [7]. For example, a simple multiplier of input W_1 and W_2, and output $\mathrm{W}_1 + \mathrm{W}_2$ has a hardware-cost that is approximately proportional to $\frac{1}{2}(W_1, W_2)\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} W_1 \\ W_2 \end{pmatrix}$, which gives a non positive semi-definite Hessian. However, the procedures proposed in [9] can directly apply, but can be computed orders of magnitude faster since it is now based on analytical expressions instead of Monte-Carlo simulations. On the other hand, an approximation of the objective using a local affine approximation of hardware cost function, being convex itself, can break Eq. (4) into external iterations based on the inner convex optimization. However, we found that since the number of word length groups is usually less than 50, so that the procedures in [9] are usually sufficiently fast.

3.8 Robust optimization

The accuracy of the overall optimization needs to be understood, since simplified modeling of the hardware-cost and specification functions introduces errors. For example, given the relationship between MSE and W_{Fr} in Eq. (3), about *m* experiments are enough to fit the *m*×K coefficients $C_{i,k}$'s, where *m* and K are as defined in Section 3.7. Either due to estimation error or under-modeling of MSE, such as various simplifications in Section VI, an estimation of $\mathrm{C}_{i,k}$ is given by a real valued interval [$\mathrm{C}_{i,k,\mathrm{lower}}$, $\mathrm{C}_{i,k,\mathrm{upper}}$], and the interval size depends on the confidence of estimation of $\mathrm{C}_{i,k}$. The design of W_{Fr} should satisfy constraints in Eq. (4) for any $\mathrm{C}_{i,k}$ that is in interval [$\mathrm{C}_{i,k,\mathrm{lower}}$, $\mathrm{C}_{i,k,\mathrm{upper}}$]. Thus, we just need to substitute $\mathrm{C}_{i,k}$ by $\mathrm{C}_{i,k,\mathrm{upper}}$ in Eq. (2) to get a robust version.

The algorithm remains almost the same as in Eq.(4) except for the replacement of $\mathrm{C}_{i,k}$'s, so the procedure in 3.7 still applies. Because of the exponential relationship, optimal wordlength design increases only about 1/2 $\log_2(\mathrm{C}_{i,k,\mathrm{upper}}/\ \mathrm{C}_{i,k})$, so that word length optimization is quite insensitive to MSE estimation errors.

3.9 User interface

In addition to the Range Detector and Spec Marker blocks that are implemented as Simulink library blocks, the rest of the tool is fully implemented as Matlab functions that realize each proceeding subsection of Section 3 in a pre-programmed flow. The user places Spec Marker(s) into the high-precision system and specifies the performance levels, and then the tool will execute all the functions in Fig. 2 and output the optimal fixed-point design.

Occasionally, MSE becomes a poor way to approximate a specification. For example, the regulation on off-band transmitting power of a radio often translates to the requirement that the digital filter in the transmitter needs to satisfy a specification on the maximum side-lobes of its frequency response. Therefore, there is an option of Spec Marker to choose as a specification function using customized Matlab functions. In this case, convex optimization method as described in Section 3.7, often can not be used.

For systems with non-stationary inputs, the program monitors the MSE at each Spec Marker at different times. There will be a specification level corresponding to each time. So, Eq. (4) retains its format, but with more constraints.

Finally, for comparison purposes the tool also provides the capability of using a pure-simulation-based approach, similar to those in [6]. That is, no analytical hardware-cost function and constraint functions are drawn. However, it still takes advantage of the discussions in 3.6.1 and 3.6.3, as well as 3.2 and grouping method in 3.3.

4. CONVERSION EXAMPLES

Table I. Summary of the three systems converted

Systems:	**SVD U-Sigma**	**UWB**	**BPSK Transceiver**
Direct system specifications	Eigen-value convergence	Detection error and BER	BER
slices of hand-tuned system	8858	6695	13665*
Number of Spec. Markers	1	10	1
Slices of the FFC'ed system	1704	4610	265
FFC duration	40 minutes	2 hours	5 minutes

*The fixed-point data types in this system are not hand-tuned and are simply set to 60-bits everywhere before automated FFC.

Three typical systems are converted to test the FFC tool. The first one is a BPSK transceiver system [15]. Both the two up-sampled low-pass root-Raised-Cosine filters on transmitter and receiver have 23 taps. This system tests FFC on a BER type of system specification, and on systems with a variety of signal processing functions and different clock rates. As in the other systems, multiple hierarchies are used to describe the system. The second system is an SVD U-Sigma system used in multi-carrier multi-antenna applications [14]. This system, is based on LMS filters but substantially more complicated, tests the FFC tool when there are nonlinear feedback and the convergence time is a concern. The third system is an ultra-wideband system designed also at Berkeley Wireless Research Center [14]. It tests the FFC tool on complicated systems that have been previously hand-tuned, so that it contains a number of fixed signals, and requires multiple specifications. Table I summarizes the results.

Depending on the hand-tuning efforts, the FFC tool can reduce the hardware-cost by 1.4x-50x times. In particular, the UWB contains about 2000 arithmetic and logic units. The system designer had manually optimized it to take 6695 slices in fixed-point implementation. The FFC tool started from this partly optimized system and reduced it to 4610 slices.

The conversion time is from minutes to hours of machine time, a considerable improvement over the weeks of designer's effort if manually performed [5].

5. COMPARISON WITH EXISTING TECHNIQUES

This section compares the FFC tool with those techniques that are based on pure-simulation. The FFC techniques are superior to other existing tools because of its more general applicability and due to the improved optimality obtained [9].

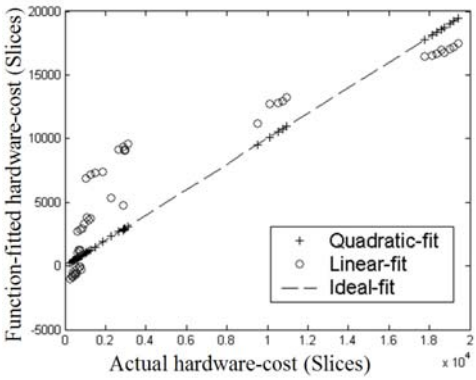


Figure 4. Hardware-cost using various models for the estimate, showing that a quadratic model is adequate (for the BPSK transceiver in Section 4).

First of all, our tool uses accurate resource estimation instead of handwritten linear hardware model that is used in all previous techniques. Fig. 4 shows that a linear model can differs greatly from actual the hardware-cost, while a quadratic model is quite accurate. Furthermore, since existing techniques that do not start from a structural description often do not model the hardware required for blocks such as a MUX, they suffer significant modeling errors.

Second, using simulation based FFC, but without adapting our techniques discussed in Section 3.6, the conversion becomes often intolerably long. Even using the ergodic approach described in Section 3.6.2 does not make it acceptable. For example, for the relatively simple BPSK system in Section 4 that has BER=0.00078 in the floating-point design, and a target BER=0.00085 for the fixed-point, each BER simulation take hours to finish. The iterations needed for optimization would further prolong the conversion time. In fact, without using the methods in Section 3.6.1, to finish the conversion listed in Table I

for the BPSK system would take at least 7 days, which is 10^3-10^4 times slower than our proposed method.

Even by further adopting part of our techniques in Section 3.6.1 to avoid BER type of simulation—using MSE(flpt-fxpt) as a direct measure of fix-point system performance, existing techniques are still at least 5 to 6 times longer. This is because the number of simulations in any unguided optimization is at least 5 or 6 times the number of independent wordlength groups, denoted by *m* [6], while the FFC approach described here only requires about *m* simulations to characterize all the specification functions.

Finally, since the tool first completely characterizes the hardware functions and specifications, optimizations against different specification levels just repeat the optimization procedure, which is usually performed in seconds. Therefore, it becomes straightforward to produce curves such as shown in Fig. 5 in which the hardware cost versus specification can be presented.

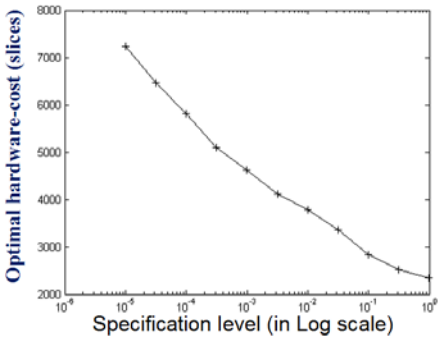


Figure 5. Hardware-cost and specification trade-off for the SVD U-sigma system in Section 4.

6. CONCLUSION

A comprehensive automated approach for floating-point to fixed-point conversion (FFC) has been presented. An implementation in a self contained tool in the Xilinx System Generator and Mathworks Simulink environment has been developed and the application of this tool to several real designs has been presented. Hardware-cost information has been modeled and a perturbation approach to determining the specification sensitivity has been implemented and found to give orders of magnitude in speedup over our simulation based techniques. The tool uses FPGA's as the hardware model merely to demonstrate the feasibility of our methodology, but a similar approach would apply to ASIC designs as well. The combinatorial optimization problem associated with quantization-modes, which may also be chosen from truncation and round-off modes, also needs further studies.

7. ACKNOWLEDGMENTS

C. Shi thanks the System Generator group of Xilinx Inc. for their permission of using the source code involved in the hardware resource estimation tool; he thanks Mike Chen and Dejan Markovic for their UWB and SVD implementation.

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.

8. REFERENCES

[1] W. R. Davis, *et al*, "A design environment for high-throughput low-power dedicated signal processing systems," *IEEE Journal of Solid State Circuits*, vol. 37, No. 3, pp. 420-431, Mar. 2002.

[2] K. Kuusilinna, *et al*, "Real-time System-on-Chip Emulation," Chapter 10, *Winning the SoC Revolution*, Kluwer Academic Publishers, pp. 229-253, 2003.

[3] The MathWorks, Inc. Simulink. [Online]. Available: http://www.mathworks.com.

[4] The Xilinx, Inc. System Generator. [Online]. Available: http://www.xilinx.com.

[5] 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.

[6] S. Kim, K. Kum and W. Sung, "Fixed-point optimization utility for *C* and *C++* based digital signal processing programs," *IEEE Trans. Circ. Sys. II: Analog and Digital Signal Processing*, vol. 45, no.11, pp1455-1464, 1998.

[7] S. Boyd, and L. Vandenberghe. *Convex optimization*. [Online] http://www.stanford.edu/~boyd/cvxbook.html.

[8] D. Menard, and O. Sentieys, "A methodology for evaluating the precision of fixed-point systems," *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, vol. 3, 2002. pp. 3152-3155.

[9] M. A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," *Proc. IEEE Int. Sym. Circs. and Sys,* 2002, vol. 2, pp. 612 -615.

[10] C. Shi, and R. W. Brodersen, "Floating-point to fixed-point conversion with decision-errors due to quantization," *IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, 2004.

[11] C. Shi, and R. W. Brodersen, "A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs," *IEEE Int. Sym. Circs. and Sys,* 2004.

[12] C. Shi, and R. W. Brodersen, "An automated floating-point to fixed-point conversion methodology," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Vol. 2, pp. 529-532, April 2003.

[13] 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.

[14] Project descriptions of Berkeley Wireless Research Center. [Online] Available: http://bwrc.eecs.berkeley.edu

[15] C. Shi, "A tutorial on using floating-point and fixed-point our conversion tool," [Online] Available: http://bwrc.eecs.berkeley.edu/people/Grad_Students/ccshi/research/FFC/documentation.htm