INTEGER WORD-LENGTH OPTIMIZATION FOR FIXED-POINT SYSTEMS

R. Nehmeh [1,2]*,D. Menard*[2]*, A. Banciu*[1]*, T. Michel*[1]*, R. Rocher*[3]

[1] STMicroelectronics, 38926 Crolles, France
[2] UEB, INSA, IETR, UMR 6164, 35708 Rennes, France
[3] UEB, University of Rennes, IRISA/INRIA, 22300 Lannion, France

ABSTRACT

Time-to-market and implementation cost are high-priority considerations in the automation of digital hardware design. Nowadays, digital signal processing applications are implemented into fixed-point architectures due to its advantage of manipulating data with lower word-length (*WL*). Thus, floating-point to fixed point conversion is mandatory. However, this conversion is translated into optimizing the integer word length (*IWL*) and fractional word length (*FWL*). Optimizing the IWL can significantly reduce the cost when the application is tolerant to a low probability of overflows. In this paper, we propose a new IWL optimization algorithm that exploits selective simulation technique to reduce both the implementation cost and optimization time. The efficiency of the algorithm is illustrated through experiments, where 17 to 22 % of cost reduction with respect to interval arithmetic and acceleration factor up to 617 with respect to classical max-1 algorithm are reported.

***Index Terms*—** Fixed-point arithmetic, overflow, fixed-point simulation acceleration, fixed-point optimization.

1. INTRODUCTION

DSP applications implemented in most embedded systems are severely constrained by cost, area, power and execution time. Using fixed-point arithmetic allows satisfying such constraints thanks to its ability in manipulating data with lower WL compared to floating-point arithmetic. Thus, DSP algorithms are implemented into fixed-point architectures and floating-point to fixed-point conversion is mandatory. The appearance of High Level Synthesis tools [1, 2] that generate the hardware directly from an abstract C-like code makes this conversion one of the most time consuming part in the hardware design. Thus, time-to-market reduction requires efficient tools to automate the fixed-point architecture synthesis.

The conversion process is an optimization problem [3] divided into two parts corresponding to the determination of the IWL and the FWL. Most of the work is focused on optimizing the FWL while satisfying the accuracy constraint. Nevertheless, optimizing the IWL can significantly decrease the implementation cost when a slight degradation of the application performance is acceptable [4]. Indeed, many applications are tolerant to overflows if the probability of overflow occurrence is low enough.

The IWL optimization is based on determining the data dynamic range. Static analysis methods based on interval arithmetic [5] or affine arithmetic [6] guarantee no overflow. However, they are pessimistic and lead to implementation over-cost. In [7], a combination of statistical approaches, as in [8], and analytical approaches is proposed to determine the dynamic range and the saturation mode. Their aim is to overcome pessimistic results associated with analytical approaches. However, this approach does not provide any information on the overflow occurrence. However, implementation cost can be improved by minimizing the IWL if low probability of overflow occurrence is acceptable. In this context, different techniques have been proposed to determine the probability density function of any type of data. They allow determining the dynamic range for a given overflow occurrence probability. These techniques are based on Extreme Values Theory [9, 10, 11] or stochastic approaches like Karhunen-Loeve expansion [12, 13] and Polynomial Chaos Expansion [14]. However, establishing the link between the application quality criteria and overflow occurrence probability is not trivial in general case. Intuitively, simulation based methods [15, 16, 17] are used to evaluate overflow effects on application quality for fixed-point systems. Although simulations can be performed on any kind of system, they are time consuming and require large number of samples to obtain accurate analysis. This results in a serious limitation on the applicability of simulation based methods.

In this paper, the determination of the IWL for each data is modelled as an optimization problem and a new IWL optimization algorithm is proposed. This algorithm efficiently exploits selective simulation based technique used to evaluate the effects of overflow on application quality criteria. On one hand, this technique accelerates the simulation of overflow effect analysis used in the optimization process. On the other hand, the proposed algorithm reduces the implementation cost by optimizing the IWL. The efficiency of the proposed algorithm is illustrated through experiments. The rest of the paper is organized as follows. The IWL determination process is modelled as an optimization process in Section 2. In Section 3 the technique used to evaluate the overflow effects is summarized and the proposed optimization algorithm is detailed. Experiments and results are presented in Section 4. Finally, Section 5 draws conclusions.

2. WORD-LENGTH OPTIMIZATION PROBLEM

Optimizing the WL in fixed-point systems is essential to control the overflow occurrence and minimize implementation cost. Classical metrics for calculating implementation cost are area, clock period, latency and power consumption. The optimization process aims at minimizing the implementation cost C while the application quality λ is greater than a minimal value λ_{min}

$$\min\left(C(\mathbf{w_d})\right) \quad \text{subject to} \quad \lambda\left(\mathbf{w_d}\right) > \lambda_{min} \tag{1}$$

where $\mathbf{w_d}$ is a N-length vector containing the word-length of each data.

Fixed-point conversion aims at choosing a fractional part word-length w_f that serves a sufficiently large computation accuracy for the application and an integer part word-length w that limits overflow occurrence. To determine the data word-length $w_d = w + w_f$,

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

a trade-off between high computation accuracy and overflow occurrence has to be investigated.

In classical fixed-point conversion techniques, the integer part word-length is computed so that no overflow occurs. Thus, determining the data word-length is split into two steps. First, the dynamic ranges of the different data are evaluated to determine the number of bits of the integer part. Second, the number of bits of the fractional part is optimized such that the quantization noise, due to the finite word-length, is sufficiently low to maintain the application quality.

In advanced fixed-point conversion techniques, the aim is to optimize both the IWL and the FWL. To obtain reasonable complexity for the fixed-point conversion, the process of IWL and FWL determination are handled separately. For determining the IWL, the problem can be expressed as follows

$$\min\left(C(\mathbf{w}+\tilde{\mathbf{w}}_f)\right) \quad \text{subject to} \quad \Delta\lambda\left(\mathbf{w}\right) < \Delta\lambda_{\max} \qquad (2)$$

where $\Delta\lambda$ is the application quality degradation due to overflow and $\Delta\lambda_{\max}$ is the maximum acceptable quality degradation.

The calculation of the cost requires the knowledge of the global word-length $\mathbf{w_d}$. Thus, the fixed-point conversion process is decomposed into three steps. First, the data dynamic range is evaluated with interval arithmetic or affine arithmetic guaranteeing no overflow occurrence. This step gives a maximum value w^{IA} for each IWL. Second, $\tilde{\mathbf{w}}_f$ is obtained by optimizing the FWL while using w^{IA} as IWL. Third, the IWL is optimized with the approach presented in Section 3 and $\tilde{\mathbf{w}}_f$ is used as FWL.

3. PROPOSED ALGORITHM

The IWL optimization is carried-out with a greedy based algorithm that minimizes the implementation cost until the performance degradation constraint is satisfied. Figure 1 presents our IWL optmization approach combining the proposed optimization algorithm and selective simulation technique for overflow effect evaluation. The proposed algorithm consists of three phases: initial phase, construction phase and refinement phase. The first phase leads to an initial solution for which the performance degradation is still null. In the construction phase, a steepest descent algorithm (max-1 bit) is used for finding a sub-optimal solution w^{mx1}. Then a local search using Tabu search algorithm is applied to refine the solution in the third phase. At each iteration, the best direction is selected and the corresponding variable is modified to converge into an optimized solution.

In this procedure, the application quality degradation due to overflow $\lambda\left(\mathbf{w}\right)$ is evaluated many times. Thus, a time efficient technique is required to accelerate this evaluation. In Subsection 3.1, our technique to evaluate the degradation due to overflow is presented.

3.1. Overflow effect evaluation

The overflow effect evaluation can be thought of as a technique for simulation acceleration by using selective simulations. The application C source of the system under test (SUT) is instrumented to collect information and simulate overflow effects by using operator overloading concept associated with C++ object-oriented language. The analysis of overflow effects on application quality criteria consists of three main tasks described below.

Index classification: In this step the indices of potential overflow for each variable v are identified. The index n for potential overflow is stored in the structure $\mathcal{T}$ if the value $v(n)$ is closed to the extreme values obtained with interval arithmetic.

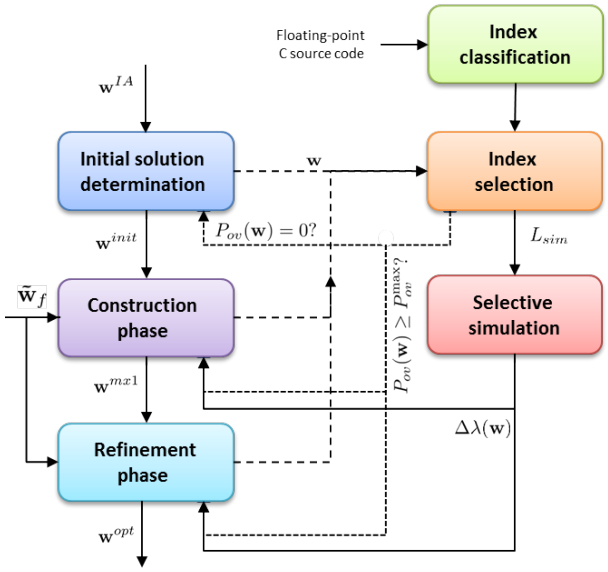

Fig. 1. IWL optimization integrating approach for overflow effect analysis.

Index selection: This step aims at constructing L_{sim}, the list of indices to be simulated for a given configuration of IWL. The analysis of L_{sim} list allows determining the overflow probability P_{ov} corresponding to the variable v.

Selective simulation: In this step, the SUT is simulated only for the different indices of L_{sim}. No overflow occurs for the indices not included in L_{sim} and the output of the reference simulation is used. Intuitively, selective simulation will accelerate the overflow effect analysis, especially in the case of limited overflow occurrence.

3.2. Initial solution determination

The aim of the initial phase is finding a starting solution for the construction phase. This phase starts with the solution obtained with interval arithmetic $\mathbf{w}^{IA}$. In [18], it has been shown that there exist one or several IWLs for each variable k of the vector $\mathbf{w}$ that are lower than $\mathbf{w}_k^{IA}$ and with null P_{ov} . These IWL configurations can be easily detected using the *Index Selection (IS)* step. The number of overflow occurrence is null if the list L_{sim} contains no element.

The initial phase is presented in Algorithm 1, where the IWL $\mathbf{w}_k$ for each variable k is decreased while P_{ov} is null *i.e.* quality degradation is null. At each iteration, the P_{ov} of the IWL configuration is determined using IS step. The solution obtained in this phase is $\mathbf{w}^{init}$.

Algorithm 1 Initial phase

```
w ← w^IA
for all 1 ≤ k ≤ N do
    while P_ov(w) = 0 do
        w_k = w_k − 1
    end while
    w_k^init = w_k + 1
end for
```

3.3. Construction and refinement phases

The construction phase is based on a steepest descent greedy algorithm (max-1 bit), *i.e.* the IWL of each variable is reduced while satisfying the performance criterion. Starting from $\mathbf{w}^{init}$, this phase allows obtaining a sub-optimal solution. Then, a refinement around this solution is applied to improve the quality of the solution in the refinement phase. This phase is carried-out with a Tabu-search algorithm, a heuristic procedure for finding good solutions of combinatorial optimization problems [19]. The combination of greedy algorithm and Tabu search may achieve better local search and avoid unnecessary movements [20]. The proposed algorithm for construction and refinement phases is presented in Algorithm 2.

3.3.1. Criterion for direction selection

At each iteration of the algorithm, the IWL of specific variable is modified to move toward the final solution. To select the best direction, a criterion has to be defined. We consider a criterion that computes the gradient of the application quality as follows

$$f_{\nabla}^{\lambda}\left(\mathbf{w}_k^{\pm}, \mathbf{w}_k\right) = \frac{\lambda\left(\mathbf{w}_k^{\pm}\right) - \lambda\left(\mathbf{w}_k\right)}{\|\mathbf{w}_k^{\pm} - \mathbf{w}_k\|} \tag{3}$$

where $\mathbf{w}_k = [w_0, ..., w_k, ...w_{N-1}]$ and $\mathbf{w}_k^{\pm} = [w_1, ..., w_k + d, ...w_{N-1}]$. The term d represents the direction and is equal to 1 for middle ascent algorithm (*min+1*) and -1 for steepest descent algorithm (*max-1*). Thus, $w_k + d$ corresponds to the previous or next value of w_k.

To improve the decision of the best direction selection, the cost due to the IWL modification is taken into account. This will give a good trade-off between the implementation cost and the application quality. The new criterion selects the direction which minimizes the cost increase and maximizes the application quality increase as follows

$$f_{\nabla}^{\lambda/C}\left(\mathbf{w}_k^{\pm}, \mathbf{w}_k\right) = \frac{\lambda\left(\mathbf{w}_k^{\pm}\right) - \lambda\left(\mathbf{w}_k\right)}{\mathcal{C}\left(\mathbf{w}_k^{\pm}\right) - \mathcal{C}\left(\mathbf{w}_k\right)} \tag{4}$$

3.3.2. Algorithm description

At each iteration, the procedure moves to one of the neighbourhood of the current solution $\mathbf{w}$ according to the value of d. The Tabu list T is the set of variables no longer used. This list is updated at each iteration to avoid useless or infinite loops. The next position of $\mathbf{w}$ for each variable k not belonging to T is calculated in line 6 of Algorithm 2. Let $\mathcal{W}_k$, be the set of valid values for the variable k. If the next position $\mathbf{w}_k^{\pm}$ does not belong to $\mathcal{W}_k$ or leads to an overflow probability greater than the maximum value $P_{ov}^{\max}$, the variable k is added to the Tabu list T as shown in line 8. The advantage brought by this condition is the ability to calculate the number of overflow occurrence in the IS step as shown in Figure 1. Thus, simulations are avoided when the overflow occurrence is too high, which results in a gain of time. If the movement is valid, the criteria for direction search associated with this variable is evaluated.

The line 13 verifies that the set T is not complete. Then in lines 14 to 29, the variable leading to the best direction is selected. Depending on the selected direction, its IWL is increased or decreased by one bit. When $\Delta\lambda(\mathbf{w})$ exceeds the constraint $\Delta\lambda_{\max}$, the direction d is reversed. The iterative process stops when the set T is complete.

Algorithm 2 Tabu search in word-length optimization

```
1:  T ← ∅                          {Empty list of tabu variables}
2:  w^opt ← ∅
3:  d ← −1                         {set the direction for steepest descent}
4:  while |T| < N do
5:    for all 1 ≤ k ∉ T ≤ N do {calculate criterion}
6:      w_k^± = w_k + d
7:      if P_ov(w^±_k) ≥ P_ov^max ∨ w_k^± ∉ W_k then
8:        T ← T ∪ {k}
9:      else
10:       ∇_k ← f_∇ (w_k^±, w_k)
11:     end if
12:   end for
13:   if |T| < N then
14:     if d > 0 then
15:       j ← argmax∇_k            {Steepest descent alg. }
16:       w_j ← w_j + 1
17:       if Δλ(w) ≤ Δλ_max then
18:         d ← −1
19:         T ← T ∪ {j}
20:       end if
21:     else
22:       j ← argmin∇_k            {Middle ascent alg. }
23:       w_j ← w_j − 1
24:       if Δλ(w) > Δλ_max then
25:         d ← 1
26:       end if
27:     end if
28:   end if
29: end while
30: w^opt ← w
31: return w^opt
```

4. EXPERIMENTS AND RESULTS

4.1. Experiment results

The proposed algorithm can be applied on any application described with a C code. For the experiments, an OFDM (Orthogonal Frequency-Division Multiplexing) receiver is considered. The proposed approach is applied on the FFT part, which is a high computational part and the most challenging module in the receiver [21]. The radix-2 64-point FFT is implemented with 6 stages carrying-out the butterfly operations. In these experiments, the effect of overflow is considered for the output of the 6 stages.

The energy consumption metric is used for calculating the implementation cost C. A library of characterized operators for FPGA target [22] is used to compute the cost according to the word-length $\mathbf{w_d}$. The performance degradation is evaluated through the Bit Error Rate (BER) degradation $\triangle_{BER}$ defined as follows

$$\triangle_{BER} = \frac{BER_{ovf} - BER_{ref}}{BER_{ref}} \tag{5}$$

where BER_{ovf} and BER_{ref} are the BER obtained respectively with and without overflows. The BER_{ref} is the BER obtained in the case of IWL calculated with interval arithmetic.

4.2. Cost-quality trade-off

The IWL optimization is carried-out for different BER degradation constraints corresponding to the term $\Delta\lambda_{\max}$ in equation 2. The optimized cost C_{opt}, obtained by our IWL optimization algorithm, is

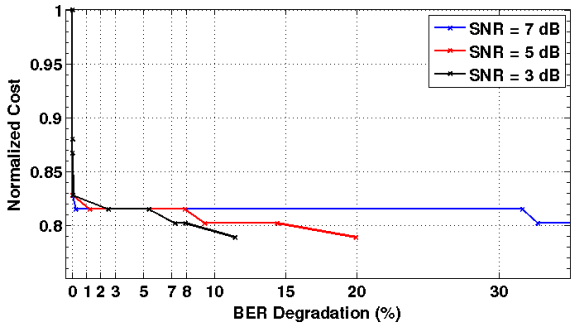

Fig. 2. Pareto-curves of the normalized cost C_{opt} and maximum BER degradation $\Delta BER_{\max}(\%)$ for different SNR

normalized to the cost obtained with interval arithmetic. Figure 2 represents the curves of the normalized cost as function of the maximum BER degradation. The curves evolve by levels leading to a step curve due to the integer values of the IWL.

The obtained curves are Pareto curves decomposed into three parts. The first part of each curve can be assimilated as a line characterized by a very high slope. The slope of the curve depends on the probability density function (*PDF*) of the application input data. In this example, the input data follow a platykurtic distribution [23] and the short tails of the PDF result in the very high slope. This part shows that the implementation cost can be significantly reduced compared to the starting solution w^{IA}, with a very low BER degradation. The pessimistic solutions obtained with interval arithmetic explain this phenomena. The second part of each curve corresponds to the "bend". This zone is the interest of the designer, since it represents a good trade-off between cost and quality. For the three SNR, the implementation cost is reduced between 16% and 18% with a low BER degradation. In the third part, the implementation cost can be reduced but at a high price of BER degradation.

4.3. Optimization time

Comparisons between three approaches have been carried-out to verify the time efficiency of the proposed algorithm. The approach *Opt+Ssim* corresponds to the proposed optimization algorithm described in Section 3, which uses selective simulation technique to evaluate overflow effects summarized in Subsection 3.1. The second approach, *Mx1+Csim*, is a classical steepest descent algorithm (max-1) with conventional (non-selective) simulations to evaluate the effect of overflows. It represents the reference technique where all the samples are simulated. The third approach, *Opt+Csim*, combines our IWL optimization algorithm (Section 3) and classical (non-selective) simulations. For the three approaches, the starting solution is w^{IA} obtained by interval arithmetic.

Figure 3 shows the evolution of the optimization time of the three approaches for different SNR per bit with respect to the BER degradation constraint $\Delta BER_{\max}$. Results show that *Mx1+Csim*, the reference approach, is the most time consuming. Using *Opt+Csim* reduces the optimization time up to 2.8 times. However, *Opt+Ssim* accelerates significantly the optimization time, where an acceleration factor between 72 and 617 is reported with respect to *Mx1+Csim*. Moreover, *Opt+Ssim* reduces the optimization time between 43 and 176 times with respect to *Opt+Csim*.

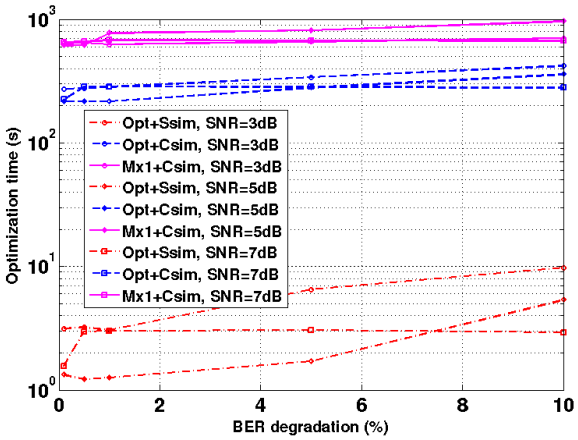

Fig. 3. Optimization time evolutions (s) according to the maximum BER degradation $\Delta BER_{\max}(\%)$ of the three approaches and for different SNR

These results emphasize the efficiency of the proposed optimization approach, and especially when it is combined with selective simulations. For different SNR per bit, the number of iterations in the optimization algorithms increases when the BER degradation constraint is relaxed *i.e.* high $\Delta BER_{\max}$. This results in an increase of the optimization times of the three approaches. Moreover, the increase of $\Delta BER_{\max}$ results in higher overflow occurrence. This increases the number of indices to be simulated (size of L_{sim}) in the case of *Opt+Ssim* and thus increases the simulation time of each iteration, and relatively higher optimization times are observed. For the different possibilities of SNR per bit and $\Delta BER_{\max}$, our approach provides a solution leading to the same or a better cost in comparison with the reference technique *Mx1+Csim* while accelerating the optimization time.

5. CONCLUSION

Automating the fixed point architecture synthesis is essential to reduce the time-to-market of hardware design. In this paper, we propose a new algorithm for IWL optimization that exploits the selective simulation technique used to evaluate the overflow effects on application performance. This algorithm does not only reduce the cost, but also allows overcoming the long execution time of classical simulation based algorithms. Through experiments applied on the FFT part of an OFDM chain, the proposed algorithm results in a significant reduction of cost with acceptable degradation of quality criteria. At the same time, results show huge enhancement in the optimization time, where the acceleration factor reaches 617 times with respect to max-1 bit. This work is a step in accelerating the design process of fixed-point systems.

6. REFERENCES

[1] Thomas Bollaert, "Catapult synthesis: a practical introduction to interactive c synthesis," in *High-Level Synthesis*, pp. 29–52. Springer, 2008.

[2] T. Feist, "Vivado Design Suite ," White papers, Xilinx, june 2012.

[3] C. Shi and R. Brodersen, "An automated floating-point to fixed-point conversion methodology," in *Proc. IEEE Interna-*

tional Conference on Acoustics, Speech, and Signal Processing (ICASSP), Hong Kong, april 2003, pp. 529–532.

[4] A. Bancu, *A Stochastic Approach For The Range Evaluation*, Ph.D. thesis, University of Rennes, Feb. 2012.

[5] R. Kearfott, "Interval Computations: Introduction, Uses, and Resources," *Euromath Bulletin*, vol. 2, no. 1, pp. 95–112, 1996.

[6] L.H. de Figueiredo and J. Stolfi, "Affine arithmetic: Concepts and applications," *Numerical Algorithms*, vol. 37, no. 1, pp. 147–158, 2004.

[7] R. Cmar, L. Rijnders, P. Schaumont, and I. Bolsens, "A Methodology and Design Environment for DSP ASIC Fixed Point Refinement," in *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*, Munich, march 1999, pp. 271–276.

[8] S. Kim and W. Sung, "Fixed-Point Error Analysis and Word Length Optimization of 8x8 IDCT Architectures," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 935–940, december 1998.

[9] Emre Özer, Andy P. Nisbet, and David Gregg, "A stochastic bitwidth estimation technique for compact and low-power custom processors," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, pp. 34:1–34:30, May 2008.

[10] E. Ozer, A.P. Nisbet, and D. Gregg, "Stochastic Bitwidth Approximation Using Extreme Value Theory for Customizable Processors," Tech. Rep., Trinity College, Dublin, october 2003.

[11] A. Chapoutot, L.S. Didier, and F. Villers, "Range estimation of floating-point variables in simulink models," in *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2012, pp. 1–8.

[12] B. Wu, J. Zhu, and F. Najm, "An analytical approach for dynamic range estimation," in *Proc. ACM/IEEE Design Automation Conference (DAC)*, San Diego, june 2004, pp. 472–477.

[13] A. Banciu, E. Casseau, D. Menard, and T. Michel, "Stochastic modeling for floating-point to fixed-point conversion," in *Proc. IEEE International Workshop on Signal Processing Systems, (SIPS)*, Beirut, october 2011.

[14] B. Wu, J. Zhu, and F.N. Najm., "Dynamic range estimation for nonlinear systems," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2004, pp. 660–667.

[15] Mathworks, *System-Level Design Products for DSP and communications*, 2001.

[16] Mentor Graphics, *Algorithmic C Data Types*, Mentor Graphics, version 1.3 edition, march 2008.

[17] Open SystemC Initiative, "SystemC User's Guide (ver 2.0)," Tech. Rep., www.systemc.org, 2001.

[18] A. Banciu, E. Casseau, D. Ménard, and T. Michel, "A Case Study Of The Stochastic Modeling Approach For Range Estimation," in *Proc. Workshop on Design and Architectures for Signal and Image Processing (DASIP)*, Edinburgh, october 2010, pp. 301–308.

[19] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, May 1986.

[20] H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel Algorithms for Word-length Optimization," in *Proc. European Signal Processing Conference (EUSIPCO)*, Barcelona, september 2011.

[21] Koushik Maharatna, Eckhard Grass, and Ulrich Jagdhold, "A 64-point fourier transform chip for high-speed wireless lan application using ofdm," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 3, pp. 484–493, 2004.

[22] N. Herve, D. Menard, and O. Sentieys, "Data Wordlength Optimization for FPGA Synthesis," in *Proc. IEEE SIPS*, 2005.

[23] B. Chissom, "Interpretation of the kurtosis statistic," *The American Statistician*, vol. 24, no. 4, pp. 19–22, 1970.