An Automatic Word Length Determination Method

MARC-ANDRE CANTIN AND YVON SAVARIA
Electrical Engineering Department
École Polytechnique de Montréal
C.P. 6079, succursale Centre-Ville, Montréal (Quebec), H3C 3A7
CANADA

Abstract: - Automatic word length determination of hardware data paths may require considering several error models, user specifications and hardware costs. A new automatic method for determining the word length of hardware data paths that consider these requirements is proposed and analyzed. The search-based method uses a C/C++ fixed-point simulation tool to model the impact of finite word lengths on overall accuracy. By computing dissimilarities between fixed-point and floating-point simulation results, a procedure searches for a combination of word lengths that meets accuracy criteria specified by the designer. This method is presented with four novel maximization procedures. These four automatic procedures are compared with other procedures given in the literature and adapted into the method. The comparison helps to select a procedure that finds a combination of word lengths that meet user specifications, in a small number of iterations. The procedures are characterized and compared using a dozen DSP algorithms.

Key-Words: - Floating-point, Fixed-Point, Automatic Determination, Optimization, Word Length, Comparison.

1. Introduction

Digital Signal Processing (DSP) algorithms are often expressed in 32- or 64-bit fixed- or floating-point data formats since these are available in most commercial off-the-shelf processors. Yet, in spite of very significant improvements in processor performance and the power efficiency, the requirements of many real-time applications in terms of pure performance or low power operation command the use of specialized hardware [1]. Since cost, power dissipation, and performance are highly dependent on the number of bits used to represent data, and on the use of floating-point operators, the problem of precise word length determination is important. In some complex designs, half of the design time can be spent determining word lengths [2]. Moreover, many algorithms require a careful selection of word lengths to preserve their stability [3]. Therefore, powerful automatic word-length determination methods are required.

For some specific applications, the resulting word length combination found by these methods, must meet various error specifications [4][5]. Furthermore, depending on the application, the resulting word length combination must minimize the hardware costs, latency, area, or power consumption, as well as increases the pure performance. Especially, for search-based maximization procedures, it is important to grade all word length combination toward the optimal solution to guide the search. However, none of methods found in the literature propose a way to handle several error models, user specifications, and implementation cost models in a common optimization process. Thus a new automatic word length determination method, which is based on a recent metric, is proposed.

This paper is structured as follow. Methods found in the literature and previous works on word length determination are reviewed in Section 2. A new automatic word length determination method that considers these requirements is presented and described in Section 3 as well as four novel search procedures. In Section 4, these four novel procedures are compared with procedures given in the literature. The methodology and results obtained in this comparative study are presented in the same section. The main conclusions are summarized in Section 5.

2. Related Work

In the last 10 years, several techniques have been proposed to translate floating-point formulations into fixed-point formulations, especially for specific DSP applications [3][4][6]. Heuristic techniques and

analytical methods dedicated to specific applications have been employed. For general DSP applications, the translation of floating-point formulations into fixed-point formulations consists of evaluating the dynamic range and the minimum accuracy, by determining the Integer Word Length (*IWL*) and the Fractional Word Length (*FWL*), of each operand O_i, *i=0, 1,…, (I-1)*, where *I* is the number of operands to be translated. The Word Length (*WL*) of each translated fixed-point operand may then be obtained as follows:

$$WL_i = IWL_i + FWL_i + s_i \quad (1)$$

where $s_i=0$ for unsigned and $s_i=1$ for two's complement representations of O_i. In Equation (1), the *IWL* and *FWL* can be either positive or negative. For example, a 4-bit binary data 1010 with $s_i = 0$ and $IWL_i=-2$ should be interpreted as 0.001010, which corresponds to a real value of 0.15625. The same 4-bit binary data with $s_i = 0$ and $FWL_i = -2$ will be interpreted this time as 101000, which corresponds to a real value of 40. To ensure that the *WL* is greater than, or equal to 0, the sum of *IWL* and *FWL* must be greater than, or equal to 0; $IWL+FWL \geq 0$.

The required Integer Word Length can be estimated using 3 methods. The first method computes the IWL_i as follows:

$$IWL_i = \left\lceil \log_2 \left(|O_i|^{\max} \right) \right\rceil \quad (2)$$

where $\lceil \cdot \rceil$ is the "ceiling" function, and where $|O_i|^{\max}$ is the maximum absolute value of the operand O_i extracted during simulations of the DSP algorithm [7][8]. The second method computes each IWL_i like the first method, except that the maximum and minimum values of the operand O_i are replaced by the values of O_i obtained by propagation of the maximum and minimum input values through a Data Flow Graph (DFG) of the DSP algorithm [9]. The third method, introduced by Sung *et al.* [10], extracts the mean m_i and the standard deviation s_i of the operand O_i during a simulation of the DSP algorithm, and then computes the IWL_i as follows:

$$IWL_i = \left\lceil \log_2 \left(\max \left(\mu_i + k \times \sigma_i, |O_i|^{\max} \right) \right) \right\rceil \quad (3)$$

where *k* is set equal to 4 in [10]. The first method does not guarantee that overflow will not occur in the data path. On the other hand, the second method is very fast, but it is also known to be a conservative approach that tends to overestimates IWL_i [11], even though it can also underestimate it if partial results cancel out. Finally, the third method minimizes the probability of overflow in the data path when *k* is set to a high value.

After the Integer Word Length, it is, the minimum accuracy, or minimum Fractional Word Length (*FWL*) of each operand O_i that is determined. Like the *IWL*, the *FWL* can be estimated by DFG analysis. FRIDGE [2] and CoCentric [12] propagate the *FWLs* through the DFG, while ensuring that no information is lost. This technique is called interpolation, and may be illustrated by a simple example: for a = b + c, no information is lost if $FWL(a) = \max(FWL(b), FWL(c))$.

Alternatively, Yasuura *et al.* [8] and Wadekar [9] perform numerical analysis on the DFG of the DSP algorithm.

Determining the *FWL* using a DFG is fast, but the method overestimates the *FWL* like it does for the *IWL*. Furthermore, the DFG analysis requires a fixed-point specification of the input signals, and becomes complex to manage when the algorithm contains data dependencies.

Cmar *et al.* [11] recently proposed a method that combines analytical rules and simulations. The method determines FWL_i by measurement of the mean m_i and standard deviation s_i of the dissimilarities between fixed-point and floating-point simulations.
The C language description of the DSP algorithm requires major modifications for the method proposed by Cmar *et al.*, and the bit resolution analysis is performed only on one operand at a time, instead of on combinations of operands.

Finally, the *FWL* can be determined by simulation only. Sung & Kum [13], Han & al. [14], Choi and Burleson[15] and Fiore and Lee [16] proposed manual procedures and guidelines to optimize the *FWL*. These procedures and guidelines that use a search-based methodology are now examined.

A procedure called ***Sequential Search*** [14] proceeds in three phases. In the first phase, for each operand O_i, *i=0, 1,…, (I-1)*, a minimum bit resolution is found such that the user specification is met, when all other operands O_j, $j=0, 1,..., (I-1)$, $j \neq i$, are in floating-point format. In the second phase of this procedure, the resolution of each operand is set equal to the value found in the first phase. This combination of word lengths is called the *Minimum Word Length (MWL)* combination. In the third phase, an iterative competition takes place between the operands to gain one bit. A bit is temporarily assigned to each operand, and the configuration that produces the least dissimilarities with a floating point simulation is the one that wins the right to keep the bit. Subsequent

competitions take place until the user specifications are met.

The ***Heuristic*** procedure [13] iteratively increases all word lengths by one bit from the MWL combination until the user specifications of the system are met. At that point, the operand that generates the largest cost savings wins the right to loose a bit as long as the error specifications continue to be met.

The ***Exhaustive*** procedure [13], as its name suggest, carries out an exhaustive search, starting from the MWL combination. The exhaustive search temporarily assigns B additional bits over the operands. Every possible assignment combination is tried. If the system specifications are not met, then B is increased, that is B=1, 2, 3, … bits, until one assignment combination meets the system specifications.

A complex procedure for word length determination is proposed in [16]. It uses an approach similar to the classical simulated annealing algorithm [17]. The ***Simulated Annealing*** procedure only considers solutions that meet the error specifications, but may temporarily explore some solutions that increase the total system hardware cost. The procedure starts with a word length combination such that the overall solution meets the system specifications. Then some WL_is are increased in order to allow reducing other WL_js, $j \neq i$, in an attempt to minimize the total system hardware cost. The user determines how many times this step is repeated.

The ***Preplanned*** procedure [14] proceeds in four execution phases. The first phase computes the system performance for each operand O_i, $i=0, 1,\ldots, (I-1)$, when WL_i is set equal to every possible bit resolution, $WL_i = 0, 1, \ldots, (WL^{max}-1)$, where $(WL^{max}-1)$ is the maximum accuracy supported by the fixed point simulation tool. During this first phase, all other operands O_j, $j \neq i$, are in floating-point format, and a sensitivity parameter, x_i, is computed for each operand:

$$x_i = f(WL_i + 1) - f(WL_i) \qquad (4)$$

where $f(.)$ is an objective function. The second phase constructs a global priority list in decreasing order of sensitivity. The third phase computes and sets all the O_i to the MWL bit resolution. The last phase is an iterative process. The width of the operand O_i that has the largest sensitivity is increased by one bit. As long as the error specifications of the system are not met, this iterative process is repeated.

The ***Branch and Bound*** procedure [15] proceeds in 3 execution phases. The first phase finds a minimum uniform word length combination, called the upper bound solution. The second phase computes the MWL combination. The third phase tries all word length combinations exhaustively in the search space between the upper bound and the MWL combination and keeps the best solution found. Limiting the search space to this range tends to drastically prune the search space as compared to a brute force exhaustive search.

3. An Automatic Optimization Method

Modeling dissimilarities between floating-point and fixed-point simulation results of a DSP algorithm may require computing several error models. For example, the CAS standards Committee of the IEEE Circuits and Systems Society proposes in [5] five error models for the Inverse Discrete Cosine Transform (IDCT) algorithm: the pixel peak error, the pixel mean square error, the pixel mean error, the overall mean square error and the overall mean error. However, for all the simulation-based procedures cited in Section 2, there is no method in the literature that explain how to handle several error models, user specifications and hardware cost models, in a single unified word length optimization process. Thus, an automatic word length determination method is proposed, and the metric on which it is based is presented in this section. Four novel procedures adapted to the method are also introduced.

3.1 Description of the Automatic Method

The proposed method is built upon a fixed-point simulation tool. Two fixed-point simulation tools were successfully used: SystemC [18] and a fixed-point utility developed by W. Sung *et al*. [19]. These fixed-point simulation tools convert a floating-point DSP program written in C or C++ into a fixed-point equivalent description. Fixed-point arithmetic operations, instead of floating-point arithmetic, are conducted automatically due to the operator overloading capability of the C++ language. Except for declaring which operands belong to a fixed-point class type, renaming the main function and adding a fixed-point header file, no other part of the original C program needs to be changed to use the proposed method.

The proposed method is composed of six steps, as shown in Figure 1.

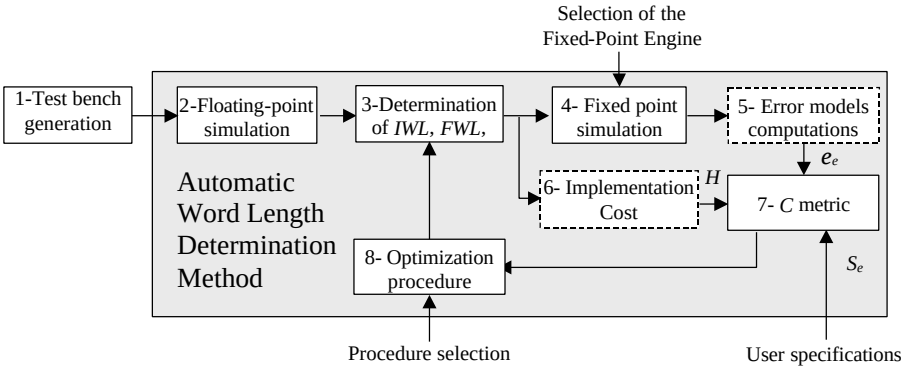

Fig. 1. Proposed method for the automatic determination of word lengths

1- Test bench generation

The user generates test benches representative of the application. They must stimulate all operands, $\{O_i\}$, in the DSP algorithm, and all data paths of interest. The method considers I user-specified operands, and outputs a word length for each one. Each test bench is a file containing inputs to stimulate the DSP algorithm.

2- Floating-point simulation

A floating-point simulation of the DSP algorithm is performed for each test bench. Several parameters for each operand O_i are extracted and computed from the simulation: the maximum absolute value $|O_i|^{max}$, the sign flag, s_i (see Equation (1)), the mean μ_i, and the variance σ_i. These parameters and output results produced by the DSP algorithm in floating-point resolution are recorded for future reference.

3- Determination of the fixed-point formats.

For each operand O_i, the integer word length IWL_i is set equal to Equation (3), and s_i to 0 if the operand O_i is always positive, and to 1 otherwise. The Fractional Word Length FWL_i is initialized according to the selected maximization procedure to be run in Step 8 below.

4- Fixed-point simulation

A fixed-point simulation is performed for each test bench. Output results produced by the DSP algorithm in fixed-point resolution are recorded.

5- Error computation

Floating-point and fixed-point simulation results stored in Steps 2 and 4 respectively are used to compute the dissimilarity between the floating- and fixed-point models according to various error e_e, for $e=0, 1, \ldots E-1$ where E is the number of error computations. Error computations e_e, such as the maximum absolute error and mean square error between each correspondent floating- and fixed-point output, are provided to quantify the dissimilarities between the floating- and fixed-point models. However the user is also allowed to provide his error computations e_e by using predetermined C functions to get the floating- and fixed-point output results and return the resulting e_e values.

6- Implementation cost

The implementation cost, denoted by H, is some measure of the hardware cost, power dissipation, or processing time for the word length combination $\{WL_i\}$, and is computed in the range [1, ∞. By default the implementation cost H is defined as follow:

$$H = \sum_{i=0}^{I-1} g_i \cdot WL_i \tag{5}$$

where WL_i is the total length in bits of the i^{th} operand. Since Equation (5) is a simplification of reality and for no loss of generality, the user is allowed to provide his own computation of the implementation cost H based on the word length WL_i of each operand O_i.

7- Computation of the C metric

Based on the error computations e_e produced in Step 5, the Implementation cost H produced in Step 6, and the user specifications S_e, a pas/fail threshold for each error computation e_e, the following C metric recently proposed in [20], is computed.

$$C = \left[\sum_{e=0}^{E-1}\left(\frac{S_e - e_e}{S_e}\right)d_e\right] \cdot H + \left[\left(I \cdot WL^{\max}\right) - \left(\frac{1}{E}\sum_{e=0}^{E-1}\left(1 - \frac{S_e - e_e}{S_e}\right)\right)\right] \cdot \frac{1}{H}\prod_{e=0}^{E-1}(1 - d_e) \tag{6}$$

where d_e=0 if $S_e \geq e_e$, and d_e=1 if $S_e < e_e$. Note that Equation (6) is composed of two terms that are mutually exclusive as a function of whether the

specifications are met (left side) or not (right side). The *C* metric grades all word length combination towards the optimal solution with the term $\sum_{e=0}^{E-1}\left(\left(S_e - e_e\right)/S_e\right)$ that could be found in both side of the metric. User specifications are normalized to provide and equal weight in the metric for each of them. Thus, for two word length combinations having the same implementation cost, the one that minimizes dissimilarities between the floating-point and fixed-point simulations yields a higher value of the metric. For two word length combinations producing the same output accuracy, the one that minimizes the implementation costs yields a higher value of the metric. This behavior comes from the implementation cost term *H* that could be found in both side of the metric. The maximum value of the metric is obtained at the optimal solution.

8- Maximization procedure

A maximization procedure tries to find the combination of word lengths $\{WL_i\}$ that maximizes the metric *C* as defined above. In the following sub-section, 4 new procedures are proposed.

3.2 Maximization Procedures

In this work, four maximization procedures are examined and compared.

a) A procedure called ***Min + B bit*** is an extension to the procedure called *sequential search* [13]. It proceeds in three phases. The first two phases remain the same as these of the *sequential search* procedure. In the third phase of the *Min + B bit* procedure, an iterative competition takes place between the operands to gain *B* bits. For some DSP algorithms, it was found that increasing the bit resolution of a single operand at a time by 1 bit could fail to converge to an acceptable solution. This happens, for instance, when the overall accuracy only increases if two or more operands are simultaneously widened. Therefore, when adding 1 bit fails to provide significant accuracy improvements, 2, 3,... , *B* bits are temporarily distributed on the *I* operands until the error decreases. At all steps, the procedure explores all *L* possible ways of assigning *B* bits to *I* operands, where *L* corresponds to the Pascal triangle (see Equation (7)). The value of *B* is kept as small as possible, and is increased only as needed to reduce the error. Once the error is reduced, *B* is reset to 1, and the iterative competition is repeated until the user specifications are met.

$$L = \frac{(I+B-1)!}{(I-1)!(B)!} \qquad (7)$$

b) A procedure called by ***Max – 1 bit*** starts with the maximum bit resolution, $WL^{\max} - 1$, allowed by the fixed-point simulation tool for all operands. Then the operands compete to loose their bits as follows. One bit is temporarily removed from each operand O_i, while all other operands O_j, $j \neq i$, remain unchanged. The operand O_i for which the *C* metric is maximized wins the competition and looses its bit. The process is repeated for another bit, and so on, as long as the error specifications are met. The pseudo-code of the *Max-1 bit* procedure is given below:

With the *Max-1* bit procedure, the final solution is found when removing 1 bit of any single operand at a time fails to meet the error specification. However for some DSP algorithm, if two or more operands are simultaneously shortened, a better solution may be found. This could happen when the quantization error of more than 2 operands compensate. Therefore when 1 bit fails to find a better solution, 2, 3, ..., *B* bits can be removed simultaneously over the *I* operands. It is not clear what limit should be used for *B,* and this issue was left for future research, thus the *Max-B bits* procedure was not explored further.

c) An ***Evolutive*** procedure starts with all operands having floating-point precision. The word length of a first operand, say O_i, is set equal to $WL^{\max}$-1 and gradually decreased until the system meets the error specifications. The value of WL_i is then increased by one additional bit, and a second operand is analyzed in the same way. This is repeated until all WL_j are fixed. The user is allowed to determine the order in which the operands are processed. Because the *Evolutive* procedure tends to minimize the first operands processed, it is recommended to process first the operand having the highest hardware cost and end with the operand having the lowest hardware cost. Thus, it is up to the user to define the order in which the operands are analyzed by determining their declaration order in the C language DSP program.

d) A ***Hybrid*** procedure combines the *Min + B bit* followed by a minimization of the word lengths as performed in the *Max – 1 bit* procedure. For some DSP algorithms, it was found that a higher value of the metric *C* can be obtained by using both procedures together than if either one is used independently.

4. Comparison

The set of procedures implemented for comparison includes the *Heuristic*, *Exhaustive*, *Simulated Annealing*, *Preplanned*, *Branch and Bound*, *Min + B bit*, *Max – 1 bit*, *Evolutive* and *Hybrid* procedures. The *Sequential Search* [14] procedure was not considered in the comparison because it sometimes fails to converge to a solution, and corresponds to the *Min + B bit* procedure with B=1.

In the *Simulated Annealing* procedure, the O_is must be initialized to values that already meet the system specifications. The *Min + B* procedure was used to find this initial starting point. In the *Simulated Annealing* procedure, the best solution found after $10 \times I$ annealing phases where that I is the number of operands, is kept as the final solution. In the *Preplanned* procedure, only the required system performances, and then the sensitivity performances are computed instead of *all* system performances. The upper bound solution of the *Branch and Bound* procedure is computed using a dichotomic search algorithm to reduce the number of iterations. For the procedures that require the MWL combination, this combination is found by using a dichotomic search for each operand. For the heuristic procedure, the *Max-1* bit procedure was selected to reduce the word length of operands whose have the highest cost per bit. All the procedures were adjusted in order to use the C metric, making them fully automatic procedures able to handle simultaneously several error models, user specifications and hardware costs.

We applied the nine-optimization procedures to the determination of word lengths for 12 DSP algorithms [21]. The algorithms include the four elementary operations (+, −, ×, ÷), the fifth order elliptic FIR filter [22], another FIR filter, an IIR filter, an adaptive filter, the CORDIC algorithm [23], the IDCT algorithm [24], a frequency estimation algorithm [25], and a neural network algorithm [26] and are denoted by DSP_1 through DSP_{12}, respectively. For the filters, the word lengths of both coefficients and data-paths were analyzed. The hardware architecture and operands for the fifth order elliptic filter were taken from [12], the IDCT from [27], the frequency estimation algorithm from [28] and the neural network algorithm from [29].

Relevant error models were selected for each DSP algorithm. For the filters, the fast Fourier transform was selected to compute the accuracy of the output frequency responses. The errors models for the IDCT were taken from the IEEE standards specifications [5]. The Rand measurement [30] was used as a quality metric for clustering produced by the neural network. For the other DSP algorithms, the maximum and mean square errors were used.

For the IDCT, the characteristics of the inputs presented were specified in [5]. For the remaining 11 DSP algorithms, the test bench consisted of applying 10000 pseudo-random inputs.

Word lengths were found for the 12 DSP algorithms by the 9 maximization procedures with using K=100 different user specifications. For each procedure, two results are given in Table 1. The first one, $\overline{\Delta WL_i}$, is a sum of word length differences averaged over K.

$$\overline{\Delta WL} = \frac{1}{K} \cdot \sum_{k=1}^{100} \Delta WL_k \tag{8}$$

where

$$\Delta WL_k = \frac{1}{I} \cdot \sum_{i=0}^{I-1} WL_{ki} - WL_{ki}^{\text{opt}} \tag{9}$$

and where WL_i^{opt} is the word length of the operand O_i resulted by the procedure that obtained the best word length combination. $\overline{\Delta WL_i}$ is normalized by the number of operands I. The second result reported in Table 1, $\overline{\Delta N}$, is the difference between the number of iterations, N, required to obtain a solution and N^{opt}, the number of iterations required by the procedure that obtained a solution with the lowest number of iterations (generally, the procedure producing N^{opt} does not correspond to the procedure that producing WL_i^{opt}). Note that $\overline{\Delta N}$ is normalized by the number of operands I_n that is,

$$\overline{\Delta N} = \frac{1}{K} \cdot \sum_{k=1}^{100} \Delta N_k \tag{10}$$

where

$$\Delta N_k = \frac{1}{I_n} \left(N_k - N_k^{\text{opt}} \right) \tag{11}$$

If a procedure produces $\overline{\Delta WL} = 0$ and $\overline{\Delta N} = 0$, then it compares favorably to all other procedures. For DSP_1 and DSP_2, all the procedures found the optimal word lengths. No procedure was able to find the optimal word length for all DSPs. For some DSP algorithms (DSP_4, DSP_7, DSP_{10}, DSP_{11} and DSP_{12}) no procedure produced $\overline{\Delta WL} = 0$. This corresponds to the situation where different procedures find the optimal solution for different user specifications. By

analogy, this situation occurs for DSP_1, DSP_3, DSP_4, DSP_5, DSP_7, DSP_8, DSP_{10} and DSP_{11} for $\overline{\Delta N}$. Note that a difference that may appear small in $\overline{\Delta WL_i}$, for example 1.65 for the *Min + b* procedure applied to DSP_{11}, may correspond to a maximum difference as large as 38 bits in total operand widths, when comparing a solution to the optimal solution for some system specification. Since the number of iterations required finding a solution dominates the processing time, a small difference on $\overline{\Delta N}$ is not very significant when a small number of operands are processed.

For instance, in some applications, up to 1000 operands are processed [2]. A difference of $\overline{\Delta N}$ =10 for example would imply $10 \times 1000 \times 0.75\text{s} = 2.08$ additional hours of processing time if 750ms were required to performed one fixed-point simulation of relevant test cases. $\overline{\Delta N}$ significantly larger than 10 have been observed. This may translate in very long additional processing time.

The *Hybrid* procedure always reaches a solution equivalent or better than the *Min + B bit* procedure, resulting in lower hardware cost. By analyzing the details of the simulations, we found two explanations: 1) An optimal solution can be reached with less hardware cost than the MWL combination. This counterintuitive result was observed several times when the quantization errors contributed by two operands or more compensate each other. 2) Finding a solution with the *Min + B bit* procedure does not ensure that all operands have their minimum word length. Therefore, for both situations, the *Hybrid* procedure takes advantage of using the *Max – 1 bit* procedure. However since the *Hybrid* procedure adds steps to the *Min + B* bit procedure, it obviously requires additional iterations.

Procedures such as the *Max – 1 bit* and the *Evolutive* start from a solution that already meets the system error specifications, and then try to find a better solution. They can be trapped in a local optimum instead of finding the global optimum. The *Heuristic* procedure produced optimal solutions with a relatively small number of iterations. The *Simulated Annealing* procedure always produced the same solutions as the *Min + B* procedure, and therefore it does not appear to bring any advantage, at least for the DSP algorithms considered here.

Most of the time, the *Preplanned* procedure required the smallest number of iterations to find a solution. However, the solutions it produces are not always the best in terms of hardware cost. Moreover, the *Preplanned* procedure, as the *Min + B bit* and the *Branch and Bound* procedures, do not consider solutions that require less hardware than the MWL combination that were found feasible in some cases. The *Branch and Bound*, *Exhaustive* and *Max - 1 bit* procedures a large number of iterations to find a solution. They may become prohibitive when a problem with a large number of operands is analyzed.

From these observations, it is found that the *Heuristic* and the new *Hybrid* procedures are the procedures very good solutions, albeit not always the optimal one. The *Hybrid* procedure often produces solutions with less hardware costs, at the expense of additional iterations. This hybrid procedure, proposed by the authors, appears to be a good alternative for rapidly finding a combination of word lengths that meets user specifications.

Table 1. Results of the comparative study

Bench-marks	I_n	Heuristic		Exhaustive		Simulated Annealing		Preplanned		Branch and Bound		Min + b bit		Max – 1 bit		Evolutive		Hybrid	
		$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$	$\overline{\Delta WL}$	$\overline{\Delta N}$
DSP_1	3	0.00	0.2	0.00	3.2	0.00	20.2	0.00	1.9	0.00	2.1	0.00	2.2	0.00	70.6	0.00	1.3	0.00	3.2
DSP_2	3	0.00	0.0	0.00	1.0	0.00	20.0	0.00	0.0	0.00	2.7	0.00	0.0	0.00	72.5	0.00	4.0	0.00	1.0
DSP_3	5	0.00	1.0	0.61	3.2	0.61	52.2	0.61	1.7	0.72	2.0	0.61	2.2	0.72	56.0	0.72	0.5	0.61	3.2
DSP_4	7	0.04	1.6	0.41	2.1	0.19	98.2	0.23	0.2	0.34	52.1	0.19	0.2	0.68	37.2	0.11	3.4	0.04	1.6
DSP_5	5	0.00	0.7	0.48	3.6	0.48	52.0	0.56	1.7	0.54	5.2	0.48	2.0	0.67	63.5	0.54	0.7	0.48	3.0
DSP_6	4	0.39	0.0	0.00	2.3	0.39	40.0	0.34	0.0	0.00	3.6	0.39	0.0	0.00	97.5	0.00	5.0	0.00	2.3
DSP_7	6	0.07	0.2	0.76	7.5	0.42	60.7	0.90	0.2	0.20	2.4	0.63	0.6	0.07	134.0	0.07	8.2	0.04	5.1
DSP_8	4	0.03	2.3	0.31	2.5	0.06	98.8	0.18	0.2	0.18	4.8	0.12	0.2	0.06	84.0	0.05	4.5	0.00	1.6
DSP_9	15	0.40	2.6	0.00	0.0	0.40	450.0	0.38	0.0	0.18	40.25	0.40	0.0	0.00	105.9	0.00	5.5	0.00	2.6
DSP_{10}	3	0.22	2.2	1.01	4.1	0.98	18.4	1.30	0.1	0.83	4.2	1.01	0.4	0.52	75.6	0.25	5.5	0.22	3.7
DSP_{11}	36	0.18	2.5	1.75	15.5	1.53	2593	3.07	1.2	1.99	2964	1.65	1.5	0.45	172.9	0.18	12.4	0.21	12.5
DSP_{12}	3	0.23	1.72	0.11	0.7	0.02	18.2	0.05	0.0	0.03	7.6	0.02	0.2	0.42	67.3	0.04	2.3	0.01	1.3

With these procedures, the method finds rapid and accurate solutions for several user specifications. It reduces the design time, implementation costs, and power dissipation, as well as allowing performance increases. Furthermore, the method can be used by procedures already proposed in the literature and it makes them able to handle several error models, user specifications and implementation costs. The method enables a platform to compare various maximization procedures, it enables a framework for architecture exploration by hardware designers. Finally, as presented in [31], the proposed method can be used for the formal analysis of DSP algorithms.

5. Conclusion

For the purpose of pure performance, low power operation and to reduce design time, an automatic method for the determination of word lengths in fixed-point implementations of DSP algorithms has been proposed, implemented and tested. The method, which uses a search-based simulation methodology, computes a *C* metric for each word length combination according to user error models, user specifications and implementation cost. A set of procedures that maximizes this metric and finds a combination of word lengths in a minimum number of iterations was proposed and implemented. Representative procedures proposed in the literature to optimize word lengths of DSP algorithms have been reviewed, implemented and adapted to the framework of our automatic tool. The hardware costs and number of iterations required by these procedures were compared with those obtained using our 4 novel procedures through testing on a dozen DSP algorithms.

The proposed method allows hardware designer to find rapidly accurate solutions for several user specifications, while reducing design time, implementation costs, and power dissipation, as well as allowing performance increases. Furthermore, the method can be used by procedures already proposed in the literature and it makes them able to handle several error models, user specifications and implementation costs in a common optimization process. Finally the method enables a platform to compare various maximization procedures, and a framework for architecture exploration by hardware designers.

References:

[1] D. Oseli, M. Mraz, and N. Zimic, Design considerations of hardware based fuzzy controllers, *WSEAS Transaction on Electronics*. Iss.2, Vol.3, 2004, pp. 811-816.

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

[3] M.-A. Cantin, Y. Blaquière, Y. Savaria, P. Lavoie, and E. Granger, Analysis of quantization effects in a digital hardware implementation of a fuzzy ART neural network algorithm, *IEEE Int. Symposium on Circuits and Systems*, Vol.3, 2000, pp. 141-144.

[4] J. Yli-Kaakinen, and T. Saramaki, An efficient algorithm for the design of lattice wave digital filters with short coefficient wordlength, *Int. Symposium on Circuits and Systems*, Vol.3, 1999, pp. 443-448.

[5] CAS Standards Committee of the IEEE Circuits and Systems Society, IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform, 1991.

[6] I.-T. Lim, and J. Bahn, Optimal wordlength determination of AC-3 decoding hardware based on fixed-point analysis and simulations of AC-3 algorithm, *IEEE Workshop on Signal Processing*, 1997, pp. 301-310.

[7] T. Aamodt, Floating-Point to Fixed-Point Compilation and Embedded Architectural Support, *Masters Thesis*, University of Toronto, 2001.

[8] H. Yamashita, H. Yasuura, F. N. Eko, and C. Yun, Variable Size Analysis and Validation of Computation Quality, *Proc. of Workshop on High-Level Design Validation and Test (HLDVT'00)*, 2000, pp. 95-100.

[9] S.A. Wadekar, Accuracy Sensitive Word-Length Selection for Algorithm Optimization, *Proc. of the Int. Conference on Computer Design: VLSI in Computers and Processors*, 1998, pp. 54-61.

[10] S. Kim, K. Kum, and W. Sung, Fixed-point optimization utility for C and C++ based digital signal processing programs, *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol.45, No.11, 1998, pp. 1455-1464.

[11] 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, pp. 271-276.

[12] Press Release: Synopsys Accelerates System-Level C-Based DSP Design With CoCentric Fixed-Point Designer Tool. Synopsys Inc., 2000.

[13] W. Sung, and K. Kum, Simulation-based word-length optimization method for fixed-point digital signal processing systems, *IEEE Trans. on Signal Processing*, Vol.43, No.12, 1995, pp. 3087-3090.

[14] K. Han, I. Eo, K. Kim, and H. Cho, Numerical word-length optimization for CDMA demodulator, *IEEE Int. Symposium on Circuits and Systems*, Vol.4, 2001, pp. 290-293.

[15] H. Choi, and W. P. Burleson, Search-based wordlength optimization for VLSI/DSP synthesis, *VLSI Signal Processing VII*, 1994, pp. 198-207.

[16] P.D. Fiore, and Li Lee, Closed-form and real-time wordlength adaptation, *Proceedings of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, Vol.4, 1999, pp. 1897-1900.

[17] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Optimization by simulated annealing, *Science*, No.220, 1983, pp. 671-680.

[18] M. Speitel, and B. Niemann, SystemC design language for development of ASICs and systems, *Elektronik*, Vol.50, No.13, 2001, pp. 78-83.

[19] S. Kim, Ki-Il Kum, and W. Sung, Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs, *IEEE Trans. on Circuits and System*s II, Iss.11, Vol.45, 1998, pp. 1455-1464.

[20] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, A Metric for Automatic Word Length Determination of Hardware Datapaths, Submitted to *IEEE Trans. on Computer-Aided Design*, 2003.

[21] M.-A. Cantin, Y. Savaria and P. Lavoie, A comparison of automatic word length optimization procedures, *IEEE Int. Symposium on Circuits and Systems*, Vol.2, 2002, pp. 612-615.

[22] L. Claesen, F. Catthoor, D. Lanneer, G. Goossens, S. Note, J. Van Meerbergen, and H. De Man, Automatic Synthesis of Signal Processing Benchmark using the CATHEDRAL Silicon Compilers, *Proc. of IEEE Custom Integrated Circuits Conference*, 1988, pp. 14.7.1-14.7.4.

[23] J.E. Volder, The CORDIC Trigonometric Computing Technique, *IRE Trans. on Electronic Computers*, V. EC-8, No.3, 1959, pp. 330-334.

[24] T. Miyazaki, T. Nishitani, M. Edahiro, and I. Ono, DCT/IDCT processor for HDTV developed with DSP silicon compiler, *Journal of VLSI Signal Processing*, Vol.5, 1993, pp. 39-47.

[25] S.N. Crozier, Performance and Complexity of Discrete-Time Frequency Estimation Algorithms, *17th Biennial Symposium on Communications*, Queen's University, Ontario, Canada, 1994.

[26] G.A. Carpenter, and S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics and Image Processing*, Vol.37, 1987, pp. 54-115.

[27] S. Kim, and W. Sung, Fixed-Point Error Analysis and Word Length Optimization of 8 x 8 IDCT Architectures, *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.8, No.8, 1998, pp. 935-940.

[28] L.-P. Lafrance, M.-A. Cantin, Y. Savaria, S.H. Sung, and P. Lavoie, Architecture and performance characterization of hardware and software implementations of the Crozier frequency estimation algorithm, *IEEE Int. Symposium Circuits and Systems*, Vol.4, 2002, pp. 823-826.

[29] M.-A. Cantin, Y. Blaquière, Y. Savaria, E. Granger, and P. Lavoie, Implementation of the fuzzy ART neural network for fast clustering of radar pulses, *IEEE Int. Symposium on Circuits and Systems*, Vol.2, 1998, pp. 458-461.

[30] W.M. Rand, Objective Criteria for the Evaluation of Clustering Methods, *Journal of the American Statistical Association*, Vol.66, No.336, 1971, pp. 846-850.

[31] S. Catudal, M.-A. Cantin, and Y. Savaria, Performance Driven Validation Applied to Video Processing, *WSEAS Transaction on Electronics*. Iss.3, Vol.1, 2004, pp. 568-575.