# Transforming Floating-Point Algorithms to Fixed-Point Implementations

**Design Automation for Low-Power Embedded Hardware and Software Design of Digital Signal Processing Algorithms**

Kyungtae Han
Brian L. Evans

August 2009

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Digital signal processing algorithms are typically simulated in floating-point environments for refinement and validation. After validation, the algorithms can be implemented in floating-point hardware or transformed to run on fixed-point hardware. Implementation on floating-point hardware offers the direct use of floating-point programs without conversion. However, when compared to implementation on floating-point hardware, implementation on fixed-point hardware offers lower economic cost, lower power consumption, and faster speed, but with a tradeoff in signal quality. In addition, off-line transformation from floating-point programs to fixed-point programs must be performed.

Fixed-point transformation requires data-type conversion from floating point to fixed point. A fixed-point data type represents a limited range of data compared to that of a floating-point data type. During fixed-point transformation, proper ranges to prevent overflow and underflow at each variable are estimated. Based on range-estimation information, the fixed-point data can be modified to reduce hardware

complexity or power consumption. Modification can be either manual by trial-and-error or automated by computer methods.

Floating-point to fixed-point transformation, including data-type conversion and word length optimization, which is mostly word length reduction, is time-consuming and can sometimes account for up to 50% of the total design effort [1]. Hence, many studies have been performed on fixed-point transformation methods, as shown in Table 1.1.

Many fixed-point environments have been developed to simulate fixed-point signal processing systems. Fixed-point simulation environments support fixed-point arithmetic and range estimation by analytical and/or statistical approaches [1–4]. Analytical estimation extracts data flow in a system and calculates the data ranges. Analytical estimation often produces conservative results. Statistical approaches , which monitor data ranges in variables through simulation, need much longer running times to collect signal statistics by simulation.

Word length optimization methods that determine bit widths of fixed-point variables with tradeoffs in signal quality and hardware complexity have been developed based on word length search methods [2, 10–13, 15]. Sung and Kum use a statistical approach for range estimation with the Signal Processing Worksystem (SPW) [5] for fixed-point simulation [2]. Fixed-point simulation environments for C++ have been developed in [3]. This environment provides range estimation class (*fSig*) and fixed-point data type and arithmetic class (*gFix*). Kum *et al.* [4] developed a program for converting a floating-point C program to an integer C program with scaling optimization (AUTOSCALER) that minimizes the number of shifts in scaling operations. Keding *et al.* [1] proposed a fixed-point conversion tool using annotation and interpolation techniques that are employed in a commercial tool, CoCentric Fixed-point Designer [17]. Nayak *et al.* [9] implemented a forward and backward propagation algorithm that is used for the MATCH project [6] for analytical range estimation.

Table 1.1: Research on floating-point to fixed-point transformation. (A: Analytical, S: Statistical, -: Not available, C: Cost, S: Shift, E: Error, SPW HDS: Signal Processing Worksystem Hardware Design System [5], FRIDGE: Fixed-point Programming Design Environment, MATCH: Matlab Compiler for Heterogeneous Computing Systems [6], MILP: Mixed Integer Linear Programming, CDM: Complexity-and-Distortion Measure)

| Paper | Fixed-point Conversion | | Word length Optimization | | |
|---|---|---|---|---|---|
| | Range Est. | Environment | Error Est. | Search Method | Search Obj. |
| Sung [2] | S | SPW HDS | S | Min+a | C |
| Kim [3] | S | fSig/gFix | - | - | - |
| Kum [4] | S | Autoscaler | - | - | S |
| Keding [1] | A | FRIDGE | - | - | - |
| Cmar [7] | A/S | - | S | - | - |
| Yama. [8] | - | - | A/S | - | - |
| Nayak [9] | A | MATCH | - | - | - |
| Han [10] | - | - | S | Seq./Preplan | E |
| Cantin [11] | - | - | S | Max - 1 | C |
| Const. [12] | - | - | A | MILP/Heur. | - |
| Shi [13] | - | - | A/S | Mosek [14] | C |
| Han [15, 16] | - | - | S | Seq./CDM | E/C |
| Proposed | S | MATLAB | S | Genetic | E/C |

Signal quality in systems according to word length is estimated by analytical or statistical approaches. Analytical approaches, which model and calculate propagated errors through fixed-point data types, could find solutions faster than statistical approaches for simple systems. However, modeling propagated errors in a closed form is sometimes difficult in complex systems. Statistical approaches measure propagated errors by simulation. Statistical approaches can be used for any system, but require long running times to measure errors.

Any consideration of all states of all possible word length combinations is generally impractical except for trivial systems. Word length optimization problems can be solved by optimization algorithms with search methods. Since statistical approaches require a long simulation time, accelerating the running time is one of the research topics in word length optimization.

Sung and Kum [2] proposed the word length search algorithm that first determined the minimum bound of the word length and then tried to determine the cost-optimal solution. Han *et al.* [10] proposed the sequential search algorithm utilizing error information based on the search algorithm from [2]. Cantin *et al.* [11] proposed the *Max-1* algorithm, which starts with the maximum word length. In addition, Cantin *et al.* [11] provide a useful survey of search algorithms for word length determination, and compare word length search algorithms. Constantinides *et al.* [12] and Shi [13] employed mixed integer linear programming (MILP) and Mosek [14], respectively, to solve word length optimization problems. In [15,16], sequential search algorithms utilizing error and cost information were proposed.

This book, which focuses mainly on word length search algorithms in fixed-point transformations, explains the fast search algorithm utilizing gradient information. Genetic and evolutionary algorithms are employed to search a Pareto optimal set in multiple-objective word length optimization.

The book also shows how to reduce power consumption with opti-

mized word length for hardware multipliers. Finally, the book presents a fully automated floating-point to fixed-point transformation environment supporting the search algorithms.

## 1.2 Scope

The works presented in this book are focused on a fixed-point transformation framework. The primary scope of this book are the following:

- Word length optimization algorithms with search methods for multiple objectives as well as a single objective including fast search algorithms for a single objective utilizing gradient information to find data word length and genetic evolutionary search algorithms for multiple objectives to optimize the signal quality vs. implementation complexity tradeoffs.

- Low-power signal processing methods for the embedded hardware and software with word length reduction techniques to reduce power consumption, and mathematical derivation of an expected value of switching activity in word length reduction techniques. The reduction in dynamic power consumption on FPGAs is estimated.

- An automated floating-point to fixed-point transformation environment software including a code generator and word length searchers. This software can automatically transform any floating-point program of digital signal processing to a fixed-point program. The software for this automatic transformation is available at

  http://www.ece.utexas.edu/~bevans/projects/wordlength/

## 1.3 Outline

Chapter 2 gives an overview of word length optimization and provides a mathematical definition of optimum word length. The chapter presents simulation-based search algorithms and genetic and evolutionary algorithms and uses a case study to illustrate how to search word lengths.

The word length search algorithms utilizing gradient information are presented in Chapter 3 with some design examples including an Orthogonal Frequency Division Multiplexing (OFDM) demodulator design and an Infinite Impulse Response (IIR) filter design. Case studies providing performance results of the algorithms are discussed in terms of running time and hardware complexity. The results of multi-objective genetic and evolutionary algorithms are compared.

Shorter word lengths in data can reduce power consumption in digital signal processing systems even though the hardware architecture is fixed. Chapter 4 presents two proposed methods for reducing power consumption by decreasing switching activity and derives mathematically the expected switching values at inputs. Chapter 4 also demonstrates and compares dynamic power reduction in FPGAs employing the methods.

Chapter 5 describes a proposed environment for a completely automating floating-point to fixed-point transformation. The word length optimization structure of the environment is presented. Chapter 5 illustrates the automating transformation with a case study.

Chapter 6 presents the major contributions from the research and directions for future work.

# Chapter 2

# Word Length Optimization

## 2.1 Fixed-Point Data Format

When designers model at a high level, floating-point numbers are useful for modeling arithmetic operations. Floating-point numbers can handle a very large range of values, and they are easily scaled. In hardware, floating-point data types are typically converted or built as fixed-point data types to reduce the amount of hardware needed to implement the functionality. To model the behavior of fixed-point arithmetic hardware, designers need bit-accurate fixed-point data types.

Fixed-point data consists of an integer part and a fractional part. The number of bits assigned to the integer representation is called the integer word length (IWL), and the number of bits assigned to the fraction is the fractional word length (FWL) [18]. Fixed-point word length (WL) corresponds to the following equation:

$$\mathrm{WL} = \mathrm{IWL} + \mathrm{FWL} \tag{2.1}$$

The word length must be greater than 0. Given IWL and FWL, fixed-point data represent a value in the range $R$ with the quantization step

$\Delta$ as follows:

$$\begin{cases} -2^{IWL} \leq R < 2^{IWL} & \text{for signed} \\ 0 \leq R < 2^{IWL} & \text{for unsigned} \end{cases} \tag{2.2}$$

and

$$\Delta = 2^{-FWL}. \tag{2.3}$$

IWL and FWL are determined to prevent unwanted overflow and underflow. IWL can be determined by the following relation:

$$IWL \geq \lceil \log_2 R \rceil. \tag{2.4}$$

Here, $\lceil x \rceil$ is the smallest integer that is greater than or equal to $x$. The range $R$ can be estimated by monitoring the maximum and minimum value or mean and the standard deviation of a signal [3,19]. FWL can be determined by word length optimization or trade-offs in the design parameters during fixed-point conversion.

## 2.2 Related Work

During the floating-point-to-fixed-point conversion process, fixed-point word lengths composed of the IWL and the FWL are determined by different approaches, as shown in Table 2.1. Some published approaches for floating-point-to-fixed-point conversion use an analytical approach for range and error estimation [9, 12, 13, 20, 21], and others use a statistical approach [3, 7, 13, 22]. An analytical approach has a range and error model for integer word length and fractional word length design. Some use a worst-case error model for range estimation [9,20], and some use forward and backward propagation for IWL design [21]. The advantages of analytical techniques are that they do

Table 2.1: Fixed-point conversion approaches for integer word length (IWL) and for fractional word length (FWL) determination.

(a) Analytical Approach

| Range Model for IWL | Error Model for FWL |
|---|---|
| Wadekar 1998 [20] | Constantinides 2003 [12] |
| Stephenson 2000 [21] | Shi 2004 [13] |
| Nayak 2001 [9] | |

(b) Statistical Approach

| Range Statistic for IWL | Error Statistic for FWL |
|---|---|
| Cmar 1999 [7] | Cmar 1999 [7] |
| Kim 1998 [3] | Kum 2001 [22] |
| | Shi 2004 [13] |

not require simulation stimulus and can be faster. However, they tend to produce more conservative word length results.

Statistical approaches have been used for IWL and FWL determination. Some use range monitoring for IWL estimation [3, 7], and some use error monitoring for FWL [7, 13, 22]. The work in [13] also uses an error model that has coefficients obtained through simulation. The advantage of statistical techniques is that they do not require a range or error model. However, they often need long simulation times and tend to be less accurate in determining word lengths.

After obtaining models or statistics of range and error by analytical or statistical approaches, respectively, search algorithms can find an optimal word length. Some published methods search for optimal word length without sensitivity information [2, 23], whereas others do use sensitivity information [11,23,24], as shown in Table 2.2. "Exhaus-

Table 2.2: Optimum word length search methods

| Cost Sensitivity | Error Sensitivity | Non-Sensitivity |
|---|---|---|
| Local [23] | Sequential [24] | Exhaustive [2] |
| Evolutive [25] | Max-1 [11] | Branch and Bound [23] |
| | Preplanned [24] | |
| Complexity-and-Distortion Measure | | |

tive Search" [2] and "Branch-and-Bound" procedure [23] can find an optimum word length without any sensitivity information. However, non-sensitivity methods have an unrealistic search space as the number of word lengths increases.

Some use sensitivity information to search for an optimum word length. The "Local Search" [23] and the "Evolutive Search" in [11] use cost-sensitivity information. The advantage of cost-sensitivity methods is that they can find an optimum word length in terms of cost. The "Sequential Search" and the "Preplanned Search" in [24] and the"Max-1 Search" in [11] use error-sensitivity information. The advantage of employing error-sensitivity methods is that they find the optimal word length in terms of error faster than cost-sensitivity methods. However, neither type of sensitivity method always reaches a globally optimal word length.

Cantin *et al.* provide a useful survey of search algorithms for word length determination. In this work, search algorithms are compared, and the Preplanned Search shows the smallest number of iterations to find a solution. However, the heuristic procedures do not necessarily capture the optimum solution to the word length determination problem, because of nonconvexity in the constraint space [12]. Thus, consideration is given to the distance between a globally optimal word length and a locally optimal word length.

## 2.3 Optimum Word Length

### 2.3.1 Formulation of Optimum Word Length

The word length is an integer value, and a set of $n$ word lengths in a system is defined to be a word length vector, that is, $\mathbf{w} \in \mathbb{I}^n$ such as $\{w_1, w_2, \cdots, w_n\}$. The objective function $f$ is defined by the sum of every word length implementation cost function $c$ as

$$f(\mathbf{w}) = \sum_{k=1}^{n} c_k(w_k) \tag{2.5}$$

where $c_k$ has real value so that $c_k : \mathbb{I} \rightarrow \mathbb{R}$. The quantized performance function $p$ indicates propagated precision or quantized error and is constrained as follows:

$$p(\mathbf{w}) \geq P_{req} \tag{2.6}$$

where $p$ has real value so that $p : \mathbb{I} \rightarrow \mathbb{R}$ and $P_{req}$ is a constant for a required performance. The lower bound word length $\underline{w}$ and upper bound word length $\overline{w}$ are considered as constraints for each word length variable:

$$\underline{w}_k \leq w_k \leq \overline{w}_k \text{ for } \forall k = 1, \cdots, n \tag{2.7}$$

The complete word length optimization problem can then be stated as

$$\begin{cases} \min\limits_{\mathbf{w} \in \mathbf{I}^n} f(\mathbf{w}) \\ \text{subject to } p(\mathbf{w}) \geq P_{req}, \underline{\mathbf{w}} \leq \mathbf{w} \leq \overline{\mathbf{w}} \end{cases} \tag{2.8}$$

The goal of the word length optimization is hence to search for the optimizer $\mathbf{w}^*$ that minimizes the objective function $f(\mathbf{w})$ in (2.8).

### 2.3.2 Finding Optimum Word Length

One of the algorithms for searching the "optimum" word length starts with an initial feasible solution $\mathbf{w^{(0)}}$ and performs an update via /indexfeasible solution

$$\mathbf{w}^{(h+1)} = \mathbf{w}^{(h)} + s\ \xi^{(h)} \tag{2.9}$$

Here, $h$ is an iteration index, $s$ is the integer step size, and $\xi$ is an integer update direction. A sound initial guess, a well-chosen step size, and a well-chosen update direction can reduce the number of iterations to find optimum word lengths.

Optimum word lengths can be found by solving equations when the performance function $p$ is expressed in analytical form. If there is no analytical form to express the performance, then simulation-based search methods can be used to search for optimum word lengths by measuring the performance function. Typical approaches involve assigning word length vector $\mathbf{w}^{(0)}$ to a lower bound, an upper bound, or a vector between the lower and upper bounds. Step size can be fixed or adapted. The update direction is adapted according to the search algorithms in Section 2.4.

During iteration, the stopping criteria are dependent on the search algorithm. The algorithm that starts from the lower bound stops when the performance $P$ reaches the required performance $P_{req}$. The algorithm that starts from the upper bound stops when $P$ falls below $P_{req}$. Other algorithms stop when the performance $P$ or cost $c$ converges within a neighborhood.

## 2.4 Simulation-Based Search Methods

Optimum word lengths can be found by solving equations when the performance function $P$ is expressed in analytical form. If there is

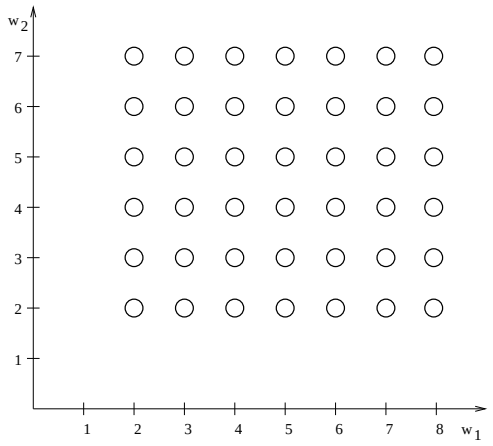



Figure 2.1: The possible word length combinations from searching the entire space in a complete search ($\underline{\mathbf{w}} = \{2, 2\}$;$\mathbf{w} = \{8, 7\}$; trials= 42).

no analytical form to express the performance, then simulation-based search methods can be used to search for optimum word lengths by measuring the performance at the system output.

### 2.4.1 Complete Search

A complete search (CS) tests every possible combination of word lengths between the lower bound and upper bound and measures the performance of each combination by simulation. Then optimum word lengths can be selected from the simulation results.

For example, assuming that the number of independent variables to find optimum word-length is two, and the lower bound and upper bound are $\{2, 2\}$ and $\{8, 7\}$, respectively, the possible word length combinations are shown in Fig. 2.1. The number of trial tests or trials is 42. The optimum word length can be selected from the given simulation results after simulation is completed.

The total number of tests in $N$ word length variables is

$$E_{CS}^{N} = \prod_{k=1}^{N} (\overline{w}_k - \underline{w}_k + 1). \tag{2.10}$$

A complete search is guaranteed to find a global optimum point, but computational time and the number of tests increase exponentially as the number of word length variables increases.

### 2.4.2 Exhaustive Search

Sung and Kum [2] search for the first feasible solution. They search for a word length with the minimum word length as the initial guess and increment the word length by one until the propagated error meets the minimum error. For example, assuming that we are trying to find the optimum word length for two variables, the minimum word lengths are $\{2, 2\}$, and all word length costs are similar, the search path is shown in Fig. 2.2. An optimized point $\{5, 5\}$ is given for a comparison between search methods. The minimum number of trials is 24.

The total number of experiments of the Exhaustive Search can be generalized. The sum of the distance $d$ with $N$ dimensions is defined as

$$d = dw_1 + dw_2 + \cdots + dw_N. \tag{2.11}$$

where $dw_i$ is the distance between the minimum word length and the optimum word length in the $i_{th}$ dimension. The expected number of experiments of the Exhaustive Search is calculated by using the

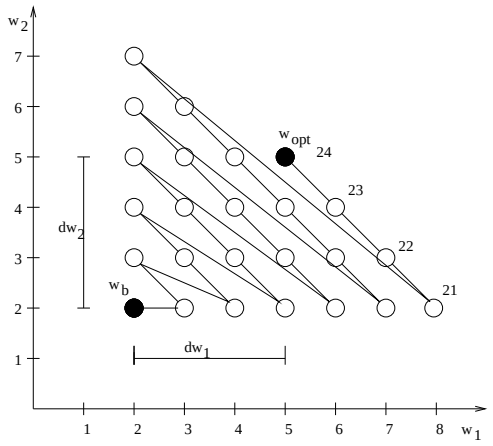



Figure 2.2: The direction of exhaustive search ($\underline{\mathbf{w}} = \{2, 2\}$ ; optimum point $= \{5, 5\}$; distance, $d$, in (2.11) is 6; trials = 24 ).

summation of combination-with-replacement in [26] as

$$\begin{aligned} E_{ES}^{N}(d) &= \sum_{r=0}^{d-1} C^{R}(N, r) \\ &= C^{R}(N+1, d-1) \\ &= \left( \begin{array}{c} N+d-1 \\ d-1 \end{array} \right) \\ &= \frac{(N+d-1)!}{\{(N+d-1)-(d-1)\}!(d-1)!} \\ &= \frac{(d+N-1)\cdots(d+2)(d+1)d}{N!}. \end{aligned} \tag{2.12}$$

The trials may be bounded as

$$E_{ES}^{N}(d) \leq E_{ES}^{N,d} < E_{ES}^{N}(d+1). \tag{2.13}$$

The number of experiments is always less than that of the Complete Search if at least two feasible solutions exist. However, the Exhaustive Search method does not always guarantee that it will find the global optimum.

### 2.4.3 Sequential Search

The basic notion of the Sequential Search is that each trial eliminates a portion of the region being searched [24]. This procedure is also called a "Min+1 Search" in [11] or "Local Search" in [23]. The Sequential Search method decides where the most promising areas are located, and continues in the most favorable region after each set of experiments [27]. The Sequential Search algorithm can be summarized by the following four steps:

1. For the independent variables, select a set of values that satisfies the desired system performance during the one-variable simulations.

2. Evaluate the system performance.

3. Choose feasible locations at which system performance is evaluated.

4. If the system performance of one point is better than at others, then move to the better point, and repeat the search, until the point has been located within the desired accuracy.

A base point is the minimum word length as an initial word length $\mathbf{w^{(0)}}$ in (2.9). In Step 3, the direction of search, $\xi$ in (2.9) is chosen in

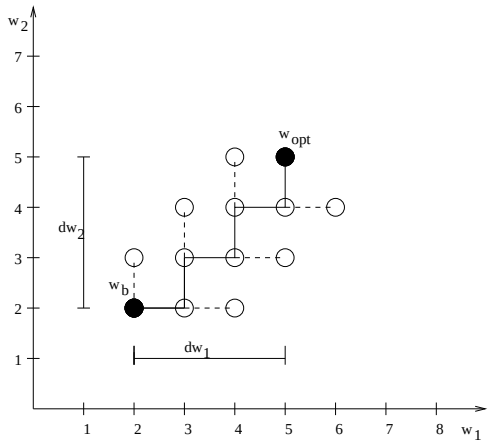



Figure 2.3: The direction of the sequential search ($\underline{\mathbf{w}} = \{2, 2\}$; optimum point $= \{5, 5\}$; distance $d$ in (2.11) is 6; trials = 12)

accordance with the maximum derivative of their performance

$$\xi_j = \begin{cases} \{1, 0, 0, \cdots, 0\} & \text{if } m_j = \nabla \frac{p}{w_1} \\ \{0, 1, 0, \cdots, 0\} & \text{if } m_j = \nabla \frac{p}{w_2} \\ \cdots & \\ \{0, 0, 1, \cdots, 0\} & \text{if } m_j = \nabla \frac{p}{w_N} \end{cases} \tag{2.14}$$

and

$$m_j = \max(\nabla \frac{p}{w_1}, \nabla \frac{p}{w_2}, \cdots, \nabla \frac{p}{w_N}) \tag{2.15}$$

where $\nabla$ is the gradient operator.

In Fig.2.3, starting from the word length base point $\{2, 2\}$, there are two directions of the sequential search in Step 3. If the performance of $\{3, 2\}$ is better than that of $\{2, 3\}$, then a new word length vector moves into $\{3, 2\}$. Simulations are repeated until the desired performance is obtained.

The generalized equation for the trials in the sequential search with $N$ dimensions is

$$E_{SS}^{N} = N \cdot (dw_1 + dw_2 + \cdots + dw_N). \tag{2.16}$$

In this example, the number of trials in (2.16) is 12, as illustrated in Fig.2.3. The number of trials is reduced by using sensitivity information; however, an optimum word length can be a local optima.

A local search [23] uses sensitivity information with the above procedure, but it uses cost sensitivity instead of performance sensitivity.

### 2.4.4 Preplanned Search

A preplanned search [24] is one in which all the experiments are completely scheduled in advance. The directions are obtained from the sensitivity of performance of an independent variable. The optimum point is found by employing the steepest descent among local neighboring points.

The preplanned search algorithm in $N$ dimensions is summarized by the following steps:

1. For the independent variables, select a set of values for the independent variables that satisfies the desired performance during the one-variable simulations.

2. Make a performance sensitivity list from the one-variable simulations.

3. Make a test schedule with the sensitivity list to follow the higher sensitivity points from the base point.

4. Evaluate the performance at those points.

5. Move to the points, until the point has been located within the desired accuracy.

In step 3, the direction of preplanned search is chosen in accordance with the maximum derivative of an independent performance

$$\xi_j = \begin{cases} \{1,0,0,\cdots,0\} & \text{if } m_j = \nabla \frac{p_1}{w_1} \\ \{0,1,0,\cdots,0\} & \text{if } m_j = \nabla \frac{p_2}{w_2} \\ \cdots & \\ \{0,0,1,\cdots,0\} & \text{if } m_j = \nabla \frac{p_N}{w_N} \end{cases} \tag{2.17}$$

where

$$m_j = \max(\nabla \frac{p_1}{w_1}, \nabla \frac{p_2}{w_2}, \cdots, \nabla \frac{p_N}{w_N}) \tag{2.18}$$

In Fig. 2.4, starting from the base point {2,2}, the preplanned search makes a list of the directions of the steepest ascent by comparing the gradients of the independent performances in one dimension from the one-variable simulations. If the gradient, which is calculated from the one-variable simulations at a $w_1$ of 2 bits, is larger than that at a $w_2$ for 2 bits, then the next feasible location is {3,2}. Then, if the gradient at a $w_1$ of 3 is smaller than that at a $w_2$ of 2, the next feasible location is {3,3}. The simulation path would be {2,2}, {3,2}, {3,3}, etc. After scheduling the feasible points, the performance of these points are evaluated until the value of the performance meets the desired accuracy.

The generalized equation for the trials in the preplanned search with $N$ dimensions as

$$E_{PS}^N = dw_1 + dw_2 + \cdots + dw_N. \tag{2.19}$$

In this example, the number of trials in (2.19) is 6, as illustrated in Fig. 2.4. The number of trials is the least among the search methods reported so far. However, finding the global optimum word length is not guaranteed.

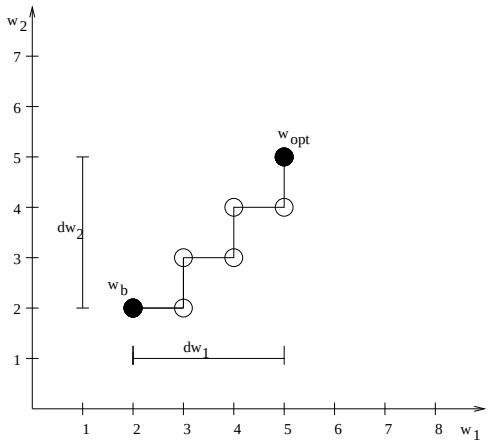



Figure 2.4: The direction of the preplanned search ($\underline{\mathbf{w}} = \{2, 2\}$; optimum point $= \{5, 5\}$; distance $d$ in (2.11) is 6; trials $= 6$)

### 2.4.5 Case Study

Typical demodulators are implemented with an analog block in front of an analog-to-digital converter (ADC) block, as shown in Fig. 2.5(a). As the speed of the ADC increases in communication systems [28], the analog parts are replaced with digital parts. As shown in Fig. 2.5(b), The analog demodulator is replaced with a digital demodulator.

The demodulator converts modulated signals into baseband signals. In the digital demodulator block of Fig. 2.6, the sampled data values output by the ADC are multiplied by a carrier signal to shift the spectrum down to the baseband. The out-of-band signal is removed by the lowpass filter (LPF). The variables in the digital demodulator are given below [10, 29]:

- $B_i$: input word length
- $B_c$: carrier word length

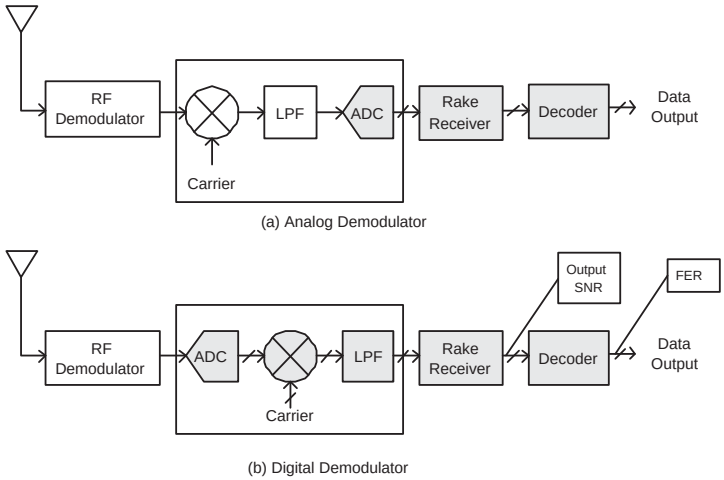



Figure 2.5: Analog and digital demodulators in CDMA receiver and performance measurement position.

- $B_m$: multiplier output word length
- $B_f$: filter output word length
- $B_{fc}$: filter coefficient word length.

Because direct measurement of frame error rate (FER), which is a general measurement to evaluate Code Division Multiple Access

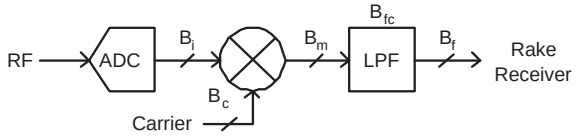



Figure 2.6: A digital demodulator block.

(CDMA) CDMA systems requires at least $10^5$ frames during the simulation [30], the output SNR is used for performance measurement instead of FER. The required output SNR in this system is over 0.8 dB, whereas FER is under 0.03 [10].

For the initial point, the minimum word length is selected by the independent one-variable simulations in which one variable changes while other variables maintain high precision. To satisfy the output SNR of 0.8 dB, the minimum word length of $\{B_i, B_c, B_m, B_f, B_{fc}\}$ is $\{4, 3, 4, 5, 7\}$, which is acquired from the one-variable simulations shown as Fig. 2.7. For a simplified example, the one cost-per-bit is assumed. In the exhaustive search, the next points are searched: $\{5, 3, 4, 5, 7\}$, $\{4, 4, 4, 5, 7\}$, $\{4, 3, 5, 5, 7\}$, $\{4, 3, 4, 6, 7\}$, $\{4, 3, 4, 5, 8\}$, $\{5, 4, 4, 5, 7\}$, etc. The search is continued until the communications performance meets the specified desired requirement. In the sequential search, the next point is one of the following: $\{5, 3, 4, 5, 7\}$, $\{4, 4, 4, 5, 7\}$, $\{4, 3, 5, 5, 7\}$, $\{4, 3, 4, 6, 7\}$, and $\{4, 3, 4, 5, 8\}$. The next point would have the largest communication performance among them. From Table 2.3, $\{4, 3, 4, 6, 7\}$ is the next location because it has the largest communication performance. The simulation moves the current point to the new point and continues to search until the performance exceeds the specified desired requirement, which is an output SNR of 0.8dB in this case. The final point is $\{5, 3, 6, 6, 7\}$, as shown in Table 2.3. From (2.11), the distance between the base and the optimum point is 4. From (2.16), the number of trials for the sequential search to find an optimum word length is 20.

In the preplanned search, the search path is estimated from the sensitivity of each one-variable simulation shown in Fig. 2.7. Starting from the minimum word length or base point, $\{4, 3, 4, 5, 7\}$, the first expected point is $\{4, 3, 4, 6, 7\}$ because, from Fig. 2.7, $B_f$ has the greatest derivative among each word length at the base point.

The sequence of the preplanned search points is $\{4, 3, 4, 5, 7\}$, $\{4, 3, 4, 6, 7\}$, $\{4, 3, 4, 6, 8\}$, $\{4, 3, 5, 6, 8\}$, $\{4, 4, 5, 6, 8\}$, etc. Simulations

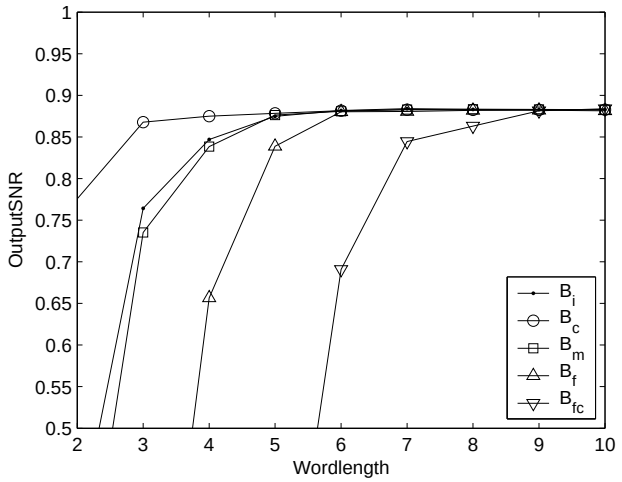



Figure 2.7: Result of the independent one-variable simulations on a CDMA demodulator.

move the current point to the next point until the performance exceeds the specified desired requirement. From (2.11), the optimum point is $\{5, 4, 5, 6, 8\}$ and distance is 5. From (2.19), the number of trials of the preplanned search to find an optimum word length is 5.

### 2.4.6 Comparison

The four search methods are compared according to the trials from (2.10), (2.12), (2.16), and (2.19), as shown in Table 2.4. For each method, the number of trials is calculated in addition to the one-variable simulation, which all of the search methods use. The complete search needs 283920 trials to find the optimum word length from (2.10) with $\overline{w}_k = \{16, 16, 16, 16, 16\}$ and $\underline{w}_k = \{4, 3, 4, 5, 7\}$, assuming that the maximum word length is 16 bits. If the computer simulation to calculate the frame error rate per trial in the CDMA system takes

Table 2.3: Sequence of the sequential search for CDMA demodulator (Traffic Channel Rate Set 1 in Additive White Gaussian Noise, Input SNR = −17.3 dB, Eb/Nt= 3.8, Rate= 9600 bps, Desired performance: Output SNR> 0.8dB, FER< 0.03)

| *Step* | $\{B_i, B_c, B_m, B_f, B_{fc}\}$ | *Output SNR* | *FER* | *Result* |
|---|---|---|---|---|
| 1, 2 | {4, 3, 4, 5, 7} | 0.711 | 0.038 | Fail |
| 3 | {5, 3, 4, 5, 7} | 0.735 | - | - |
| 3 | {4, 4, 4, 5, 7} | 0.694 | - | - |
| 3 | {4, 3, 5, 5, 7} | 0.712 | - | - |
| 3 | {4, 3, 4, 6, 7} | 0.759 | - | Max |
| 3 | {4, 3, 4, 5, 8} | 0.704 | - | - |
| 4 | {4, 3, 4, 6, 7} | 0.759 | 0.035 | Fail |
| 3 | {5, 3, 4, 6, 7} | 0.763 | - | - |
| 3 | {4, 4, 4, 6, 7} | 0.722 | - | - |
| 3 | {4, 3, 5, 6, 7} | 0.773 | - | Max |
| 3 | {4, 3, 4, 7, 7} | 0.751 | - | - |
| 3 | {4, 3, 4, 6, 8} | 0.749 | - | - |
| 4 | {4, 3, 5, 6, 7} | 0.773 | 0.034 | Fail |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 3 | {6, 3, 5, 6, 7} | 0.798 | - | - |
| 3 | {5, 4, 5, 6, 7} | 0.802 | - | - |
| 3 | {5, 3, 6, 6, 7} | 0.805 | - | Max |
| 3 | {5, 3, 5, 7, 7} | 0.803 | - | - |
| 3 | {5, 3, 5, 6, 8} | 0.798 | - | - |
| 4 | {5, 3, 6, 6, 7} | 0.805 | 0.029 | Pass |

about 10 minutes, the complete search to find an optimum word length would require 5 years, which is an unrealistic design time.

By using (2.12), the exhaustive search needs 56 trials, by using (2.12), which is fewer than the complete search. The exhaustive search is, however, inefficient in finding the optimum word length when the word length variables for optimization are numerous and the distance between the base and optimum point is longer.

The sequential search and preplanned search requires 20 and 5 trials, respectively, which are fewer than for the other search methods. Among the search methods, the preplanned search has the smallest number of experiments, but its distance according to (2.11) is greater than that for the sequential search. By implication, therefore, the word length of the sequential search method is closer to a global optimum with respect to hardware cost.

Techniques based on the gradient projection method encounter a loss of direction problem when they employ the sequential search and preplanned search. This problem can be solved by adapting the step size.

The sequential search and the preplanned search reduce the trials by rates of 64% and 91%, respectively, when compared to the exhaustive search for word length optimization in the CDMA demodulator design. However, the preplanned search seldom converges to the same optimum point, and the distance is longer than that of the other search methods.

## 2.5 Genetic Algorithms

In 1975, Holland introduced an optimization procedure that mimics the process observed in natural evolution [31] and is known as the genetic algorithm, or GA [32–34]. This technique of optimization is similar to its associated algorithms such as simulated annealing [35],

Table 2.4: Comparison of complete, exhaustive, sequential and preplanned search ( $N = 5$, $\overline{w}_k = \{16, 16, 16, 16, 16\}$, $\underline{w}_k = \{4, 3, 4, 5, 7\}$, and the term $d$ is defined in (2.11)).

| *Search* | *Distance* ($d$) | *Number of Experiments from* (2.10), (2.12), (2.16), (2.19) | *Trials* |
|---|---|---|---|
| Complete | - | $\prod_{k=1}^{N}(\overline{w}_k - \underline{w}_k + 1)$ | 283920 |
| Exhaustive | 4 | $(d+4)(d+3)\cdots(d)/5!$ | 56 |
| Sequential | 4 | $5 \cdot d$ | 20 |
| Preplanned | 5 | $d$ | 5 |

evolutionary strategies [36], and evolutionary programming [37, 38], which are classified as guided random techniques. Because of its simple implementation procedure, the GA can be used as an optimization tool for designing AI-hybrid systems for real-world applications [39–44].

Genetic and evolutionary algorithms [34] provide optimization techniques that mimic the three major components of natural evolution and selective breeding: selection, exchange of genetic material during reproduction (or mating), and random mutations as shown in Fig. 2.8 [45].

By their definition, genetic algorithms lend themselves to discussion in terms of a biological paradigm. For example, an instance of a system is referred to as an individual. An individual contains a genotypic description, which is the list of attributes (or decision variables) to be optimized. A group of individuals make up a population. Selection is mimicked by comparing the performances of individuals in a population and determining which individuals get to mate. An individual with more desirable features for fitness is given a higher probability of mating. Mating is performed by combining some attributes from one parent and the remaining attributes from the other

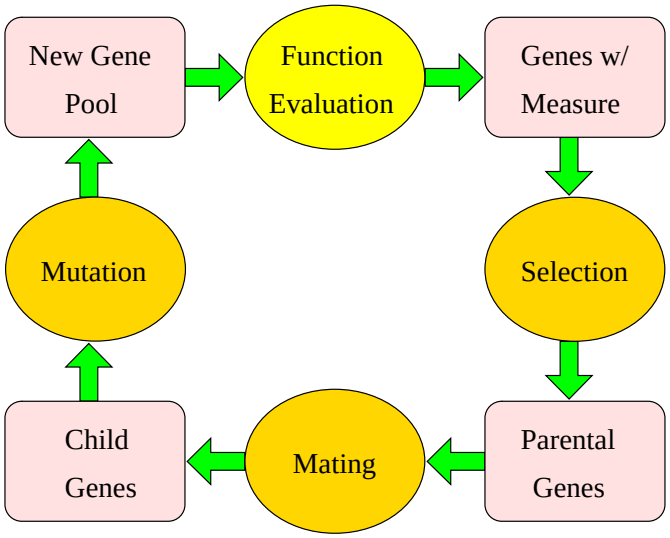



Figure 2.8: Genetic and evolutionary algorithms [45]

parent. The attributes of the resulting individual can then be mutated using any method desired, including normal and uniform distributions. Mutation provides a natural resistance to the optimization process converging on a local minimum or maximum and allows the introduction of new genetic material into the gene pool.

### 2.5.1 Multi-Objective Evolutionary Optimization

Most problems in nature have several objectives (normally conflicting with each other) that need to be achieved at the same time. These problems, called "multi-objective" optimization problems, were originally studied in the context of economics [46]. Because of the conflicting nature of their objectives, multi-objective optimization problems do not normally have a single solution, and, in fact, they even require the definition of a new notion of "optimum." The most com-

monly adopted notion of optimality in multi-objective optimization is that originally proposed by Edgeworth [47] and later generalized by Pareto [48]. Such a notion is called *Edgeworth-Pareto optimality* or, more commonly, *Pareto optimality*.

**Definition 2.5.1** (Definition of Pareto Optimality)**.** A vector of decision variables $\mathbf{x}^* \in \mathcal{F}$ is *Pareto optimal* if there does not exist another $\mathbf{x} \in \mathcal{F}$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one j. Here, $\mathcal{F}$ denotes the feasible region of the problem (i.e., where the constraints are satisfied).

In words, this definition says that $\mathbf{x}^*$ is Pareto optimal if there exists no feasible vector of decision variables $\mathbf{x} \in \mathcal{F}$ that would decrease some criterion without causing a simultaneous increase in at least one other criterion. Unfortunately, this concept almost always gives not a single solution, but rather a set of solutions, called the *Pareto optimal set*. The vectors $\mathbf{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *non-dominated*. The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the *Pareto front*, as shown in Fig. 2.9.

### 2.5.2 Pareto Rank

A fitness value of a chromosome is required in GA operations. However, it is difficult to combine the objectives both in linear and/or nonlinear fashion to reflect the fitness of the chromosome. Therefore, a Pareto multi-objective ranking approach [49] is adopted. Consider an individual $x_i$ at generation $t$ that is dominated by $p_i^{(t)}$ individuals in the current population. Its current position in the individuals' rank can be given by

$$\text{rank}(x_i, t) = 1 + p_i^{(t)} \tag{2.20}$$

All non-dominated individuals are assigned Rank 1. Fitness is assigned to each chromosome according to its rank in the population.

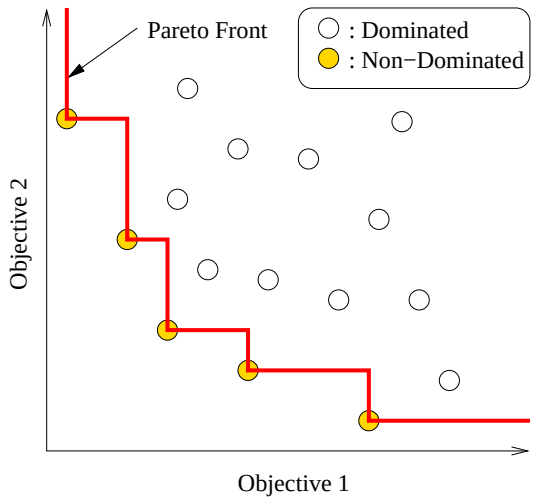



Figure 2.9: Pareto front in two objectives

### 2.5.3 Word Length Optimization with Multi-Objective Evolutionary Algorithms

When implementing a digital filter in hardware, filter coefficients have to be represented with finite word length. Several methods have been proposed to effectively design finite impulse response (FIR) filters with linear programming [50–52]. Xu and Daley [53] show that GA is superior to integer programming techniques in filter design.

Being powerful optimization tools, the genetic and evolutionary algorithms have explored a large number of applications [54] in signal processing, such as adaptive IIR filtering [55], nonlinear model selection, time-delay estimation, active noise control, and speech processing.

Genetic algorithms have been used in filter design [53, 56–60]. The initial use of a genetic algorithm for filter design was reported in [56]. Genetic algorithms have been used to design multiplierless filters [61–

63] or filters represented with cannoical signed-digits (CSD) [64–67].

Genetic algorithms have been applied to word length design in digital signal processing. Word lengths in digital signal processing are analogous to genes, and each set of word lengths is analogous to a chromosome. The GA is used to determine word length in filter coefficients [60] and to optimize the word length of input data and coefficients in a FFT processor [68] with a single objective.

Some papers have employed multiple objectives for word length optimization. Leban and Tasic [59] used mean square error, delay, and area as objectives. Signal-to-noise ratio and power are used as objectives by Sulaiman and Arslan [69]. These works employed a weighted sum as a fitness function. As in the case of the weighted sums methods, the relative importance of objectives should be specified using weights (quantitatively). Furthermore, a simple weighted-sum technique only finds a single solution of the many possible optimal solutions in the objective space. Thus, the single solution does not provide the ability to understand the various trade-offs that are possible in objective space [45].

In this book, Pareto ranking approaches [45] are used for multiple objective evolutionary algorithms to optimize word length, and the results are shown in Section 3.5.

## 2.6 Summary

This chapter briefly explains fixed-point word length optimization and several search algorithms. Table 2.5 summarizes the advantages and disadvantages of the algorithms mentioned in this chapter. As shown in Table 2.5 the complete method and the genetic method have more advantages compared to the other methods. However, the complete search method is impractical. The sequential and the preplanned methods show less iteration than the other methods. However, these

Table 2.5: Advantages/disadvantages of word length search algorithms in this chapter

| Advantages | Disadvantages |
|---|---|
| 1. Global optima | 1. Local optima |
| 2. Pareto ranking | 2. Weights in objectives |
| 3. Handle multi-objectives | 3. Single objective |
| 4. Amenable to parallelism | 4. Limited parallelism |
| 5. Low algorithm complexity | 5. High algorithm complexity |
| 6. Fewer iterations | 6. More iterations |
| | 7. Impractical |

| Methods | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Advantages | | | | | | Disadvantages | | | | | | |
| Complete | Y | | | Y | Y | | | | Y | | | Y | Y |
| Exhaustive | | | | | Y | | Y | | Y | Y | | Y | |
| Sequential | | | | | Y | Y | Y | | Y | Y | | | |
| Preplanned | | | | | Y | Y | Y | | Y | Y | | | |
| Genetic/weighted | | | Y | Y | | | | Y | | | Y | Y | |

methods cannot handle multiple objectives. The next chapter proposes a modified sequential search algorithm that can handle multiple objectives.

# Chapter 3

# Word Length Optimization for Hardware Implementation

## 3.1 Introduction

As described in Section 2.4, there are many search algorithms used in word length optimization. Gradient-based search algorithms utilize gradient information of word length to find better solutions. The gradient information can be obtained from sensitivity measurements of complexity or distortion according to the word length set. This chapter presents the complexity-and-distortion measurement method that utilizes all sensitivity information simultaneously. Case studies in word length design in OFDM demodulators and IIR filters demonstrate the proposed algorithms. Simulation results from multi-objective genetic algorithms are also shown and compared.

## 3.2 Sensitivity Measurements

The sensitivity information or gradient information used to update directions in (2.9) can help reduce the search space dramatically. The sensitivity information can be obtained by measuring hardware complexity and distortion or propagated quantized precision loss. The complexity measure is used for the hardware cost function in [23]. The distortion measure in [24] utilizes the sensitivity information of a propagated quantization error. The complexity-and-distortion measure in [15] combines the two measures to update the search direction.

### 3.2.1 Complexity Measure (CM)

The complexity measure method considers the hardware complexity function as the cost function in (2.5) and uses the sensitivity information of the complexity as the direction to search for the optimum word lengths. The local search in [23] uses the complexity measure.

The complexity measure method updates word lengths from the direction of the lowest sensitive complexity until a system meets a required performance, such as $P_{req}$ in (2.6). The complexity measure method searches the word lengths that minimize hardware complexity; however, it demands a large number of iterations since it does not use any distortion sensitivity information that can speed up the search for the optimum word lengths. For example, in a system composed of adders and multipliers, the complexity sensitivity of a multiplier is larger than that of an adder. The complexity measure method increases the word length in the adder with the priority during an increase procedure even if the word length in the multiplier affects the propagated quantized performance more. It would waste computer simulation time if the complexity sensitivity of an adder is much smaller than that of a multiplier.

### 3.2.2 Distortion Measure (DM)

The distortion measure method considers the distortion function as the objective function in (2.5) and uses the sensitivity information of the distortion for the direction to search for the optimum word lengths. The sequential search method uses the distortion measure. This method assumes that every cost or complexity function will be the same or equal to 1 and selects word lengths with the update direction according to the distortion sensitivity information.

The complexity objective function is replaced with the distortion objective function $d(\mathbf{w})$ as

$$f_d(\mathbf{w}) = d(\mathbf{w}) \tag{3.1}$$

and the complexity minimization problem is changed into a distortion minimizing problem by

$$\begin{cases} \min\limits_{\mathbf{w}\in\mathbf{I}^n} f_d(\mathbf{w}) \\ \text{subject to } d(\mathbf{w}) \le D_{req}, c(\mathbf{w}) \le C_{req}, \underline{w} \le \mathbf{w} \le \overline{w} \end{cases} \tag{3.2}$$

where $D_{req}$ is the required distortion, and $C_{req}$ is a complexity constant.

The sensitivity information is also calculated by the gradient of the distortion function. For the steepest descent direction, the update direction is

$$\xi_{\mathbf{DM}} = -\nabla f_d(\mathbf{w}) \tag{3.3}$$

For the distortion, Fiore and Lee [70] computed an error variance, and Han *et al.* [24] measured an output SNR.

The distortion measure method reduces the number of iterations for searching the optimum word lengths, because the search direction depends on the gradient information of the distortion. This method

rapidly finds the optimum word length satisfying the required performance within fewer iterations than the complexity measure method. However, the method does not guarantee the optimum word lengths in terms of the complexity.

### 3.2.3 Complexity-and-Distortion Measure (CDM)

The complexity-and-distortion measure combines the complexity measure with the distortion measure by using a weighting factor. In the objective function, both complexity and distortion are considered simultaneously. The complexity and the distortion function can be normalized by complexity and distortion values at base word length, respectively. The two objectives can be added with complexity and distortion weighting factors, $\alpha_c$, and $\alpha_d$, respectively.

Thus, the new objective function is

$$f_{cd}(\mathbf{w}) = \alpha_c \cdot c_n(\mathbf{w}) + \alpha_d \cdot d_n(\mathbf{w}) \tag{3.4}$$

where $c_n(\mathbf{w})$ and $d_n(\mathbf{w})$ are the normalized complexity function and distortion function, respectively. The relation between the weighting factors is

$$\alpha_c + \alpha_d = 1 \tag{3.5}$$

where

$$0 \leq \alpha_c \leq 1, 0 \leq \alpha_d \leq 1. \tag{3.6}$$

Using (3.4), the objective function gives a new optimization problem:

$$\begin{cases} \min\limits_{\mathbf{w} \in \mathbf{I}^n} f_{cd}(\mathbf{w}) \\ \text{subject to } d(\mathbf{w}) \leq D_{req}, c(\mathbf{w}) \leq C_{req}, \underline{w} \leq \mathbf{w} \leq \overline{w} \end{cases} \tag{3.7}$$

where $D_{req}$ and $C_{req}$ are the required distortion and a complexity constant, respectively. This optimization problem is to find word lengths that minimize complexity and distortion simultaneously according to the weighting factors.

Setting the complexity and distortion weighting factor, $\alpha_c$ and $\alpha_d$, from 0 to 1, the complexity-and-distortion method searches for an optimum word length with tradeoffs between the complexity measure method and distortion measure method. The complexity-and-distortion measure becomes the complexity measure or the distortion measure when $\alpha_d = 0$ or $\alpha_c = 0$, respectively.

The complexity-and-distortion measure method can reduce the number of iterations for searching the optimum word lengths, because the distortion sensitivity information is utilized. This method can more rapidly find the optimum word length that satisfies the required performance because it requires fewer iterations than the complexity measure method. However, the word lengths are not guaranteed to be optimal in terms of the complexity.

## 3.3 Case Study

### 3.3.1 Orthogonal Frequency Division Multiplexing Demodulator Design

Digital communication systems have digital blocks such as demodulators that need word length optimization. The searching algorithms in Section 2.4 were applied to the word length optimization of a CDMA demodulator design in Section 2.4.5. From the CDMA case study, the sequential search appears to be one of the promising methods for finding an optimum word length. In this section, the complexity measure, distortion measure, and complexity-and-distortion measure in Section 3.2 are applied in the sequential search framework to determine word

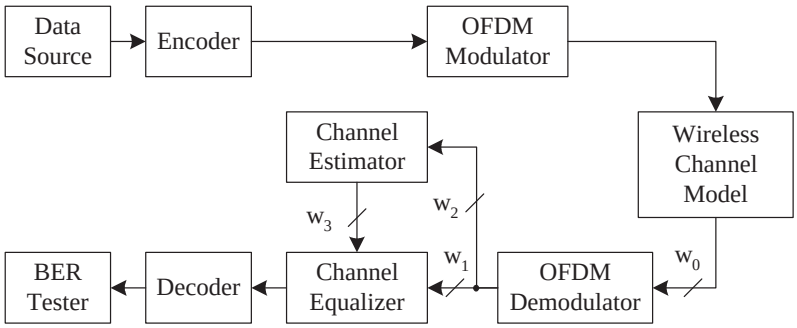



Figure 3.1: Word length model for a fixed-broadband wireless access demodulator.

lengths for a fixed-broadband wireless demodulator.

Fixed-broadband wireless access technology is intended for high-speed voice, video, and data services, which are presently dominated by cable and digital subscriber line technologies [71]. One of the designs for orthogonal frequency division multiplexing (OFDM) demodulators for fixed-broadband wireless access is shown in Fig. 3.1. For the wireless channel, Stanford University Interim models are used [72] [73].

The main blocks in the demodulator for finite word length determination are the fast Fourier transform (FFT), equalizer, and estimator. The word lengths, that have the most significant impact on complexity and distortion in the system, are chosen. In the OFDM demodulator, word length variables of $w_0$, $w_1$, $w_2$, and $w_3$ are selected for the FFT input, equalizer right input, channel estimator input, and equalizer upper input, respectively, as shown in Fig. 3.1.

The internal word lengths of the given blocks are assumed to be predetermined. In simulation, only the inputs to each block are constrained to be in fixed-point type, whereas the blocks themselves are simulated in floating-point type.

For the hardware complexity, the number of multiplications is measured assuming that processing units are not reused. The number of multiplications in a $K$-point FFT block is

$$Cost_{FFT} = \frac{K}{2} \log_2 K. \tag{3.8}$$

where $K$ is the number of taps. The cost of the 256-point FFT in the fixed broadband wireless access is estimated to be 1024. The simplified complexity vector $\mathbf{c}$ of the word length per bit is assumed to be approximately $\{1024, 1, 128, 2\}$ from [2] [74].

Another assumption is that the complexity increases linearly as word length increases to simplify demonstration. For the distortion measurement, the bit error rate (BER) is measured. The minimum word length searched by changing one word length variable while other variables have high precision (i.e., 16 bits) is used for the initial word length [2] [24]. The simulation for the minimum word length is shown in Fig. 3.2.

Assuming the minimum performance of BER is $5 \times 10^{-3}$, from the Fig. 3.2, the minimum word length is $\{5, 4, 4, 4\}$. Starting from the minimum word length, word lengths are increased according to the sensitivity information of different measures in Section 3.2. The number of iterations was measured until they find their own optimum word length satisfying the required performance, such as BER $\leq 2 \times 10^{-3}$, without a channel decoder. For the optimum word length, the hybrid procedure [11] which combines a word length increase followed by a word length decrease was used. The simulation results are presented in Section 3.4.

### 3.3.2 Infinite Impulse Response Filter

The OFDM demodulation case requires a large number of long simulations, which becomes especially time-consuming when each simulation

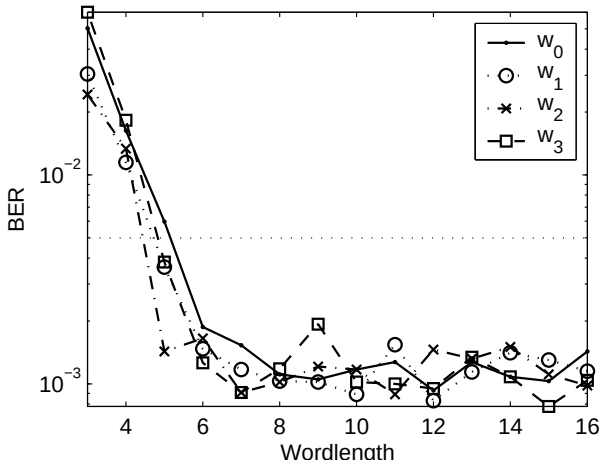



Figure 3.2: Word length effect for the demodulator in Fig. 3.1, with Stanford University Interim wireless channel model Number 3, SNR of 20 dB, FFT length of 256, and least squares comb type channel estimator without error control coding.

Table 3.1: Simulation results of several search methods starting from the minimum word length for the demodulator arcs in Fig. 3.1. $N = 4$, $\underline{w}_k = \{5, 4, 4, 4\}$, $\overline{w}_k = \{16, 16, 16, 16\}$. CDM is complexity-and-distortion measure. $\alpha_c$ is weighting factor.

| *Search Method* | $\alpha_c$ | *Number of Trials* | *Word lengths for Variables* | *Complexity Estimate* |
|---|---|---|---|---|
| Sequential [24] | 0 | 16 | {10,9,4,10} | 10781 |
| CDM | 0.5 | 15 | {7,10,4,6} | 7702 |
| Local [23] | 1 | 69 | {7,7,4,6} | 7699 |

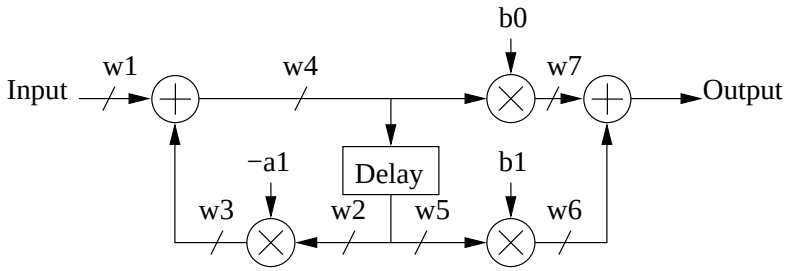



Figure 3.3: First-order direct form-II IIR filter.

takes hours of BER estimation. For a more general case, the infinite impulse response (IIR) filter that has 7 word lengths is simulated as shown in Fig. 3.3. The pole/zero plot for the IIR filter used in this case study is shown in Fig. 3.4. There are various methods for deriving the error function and cost function. For simplifying the simulation, the mean square error (MSE) is measured for the error function, and a linear cost function of word length is assumed. The required performance of the IIR filter is assumed to be an MSE of 0.1. The results are presented in Section 3.4.

## 3.4 Results of Sensitivity Measurements

The word length optimization problem is a discrete optimization problem with a nonconvex constraint space [75]. This nonconvexity makes search for a global optimum solution more difficult [76]. Table 3.1 and Table 3.2 show that there are several local optimum word lengths that satisfy error specification and minimize hardware complexity in the case studies. In this section, word length optimization methods used in the case studies are compared in terms of number of iterations and hardware complexity, and future work is discussed.

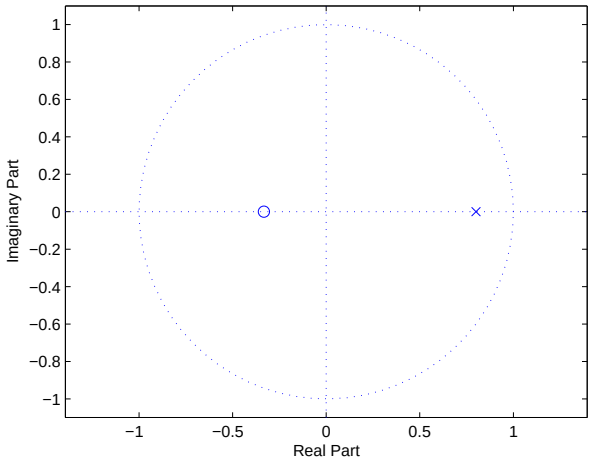



Figure 3.4: Pole/zero plot for the IIR filter

Table 3.2: Simulation results in IIR filter of several search methods. $N = 7$, $\underline{w}_k = \{1,1,1,1,1,1,1\}$, $\overline{w}_k = \{16,16,16,16,16,16,16\}$. CDM is the complexity-and-distortion measure. $\alpha_c$ is weighting factor. (Max-1 search starts from $\overline{w}_k$. Sequential search starts from $\underline{w}_k$)

| *Search Method* | $\alpha_c$ | *Number of Trials* | *Word lengths for Variables* | *Complexity Estimate* |
|---|---|---|---|---|
| Max-1 [11] | 0 | 94 | {4,5,4,5,2,2,4} | 378 |
| Sequential [24] | 0 | 56 | {4,5,4,5,2,2,4} | 378 |
| CDM | 0.25 | 44 | {4,5,4,4,2,2,5} | 366 |
| CDM | 0.5 | 33 | {6,5,5,4,1,2,4} | 363 |
| CDM | 0.75 | 71 | {6,4,4,4,2,16,13} | 561 |
| Local [23] | 1 | 126 | {9,5,16,4,1,16,16} | 723 |

### 3.4.1 Number of Iterations

The number of iterations to search an optimum word length in the OFDM demodulator design is shown in Fig. 3.5. The initial word length does not satisfy the desired performance. After a number of trials by updating the word length as in (2.9), the error at the system output decreases. The sequential search and the CDM search reach a feasible area after 15 trials. However, the local search takes 38 trials. After arriving at the feasible area, an optimum word length is searched again. In this case, the word lengths, which are searched by the sequential search or the CDM search, already arrive at an optimum word length. However, the local search needs more iteration to find an optimum word length. The total number of trials to find an optimum word length in each method for the OFDM case is shown in Table 3.1. The sequential search and the CDM method can find an optimum solution in one-fourth of the time that the local search method takes.

In the IIR filter design, the number of iterations to search an optimum word length is shown in Fig. 3.6. This figure demonstrates the number of trials in infeasible area and feasible area. After the search methods reach a feasible region where the MSE of the IIR filter is under 0.1, the search methods continue searching for an optimum word length. The sequential search and the local search need a total of 56 and 126 iterations, respectively, including iterations in feasible and infeasible areas, as shown in Table 3.2. The max-1 search starting from the feasible area needs 96 iterations. The CDM methods with weighting factors of 0.25, 0.5, and 0.75 are used for comparison. When $\alpha_c$ is less than 0.5, the CDM methods have the properties of the sequential search. When $\alpha_c$ is greater than 0.5, the CDM methods search in the same manner as the local search does. In Fig. 3.6, the CDM methods with weighting factors of 0.25 and 0.75 show shapes similar to those of the sequential search and the local search, respectively. In the IIR filter case, the CDM method with an $\alpha_c$ of 0.5 can find an optimum

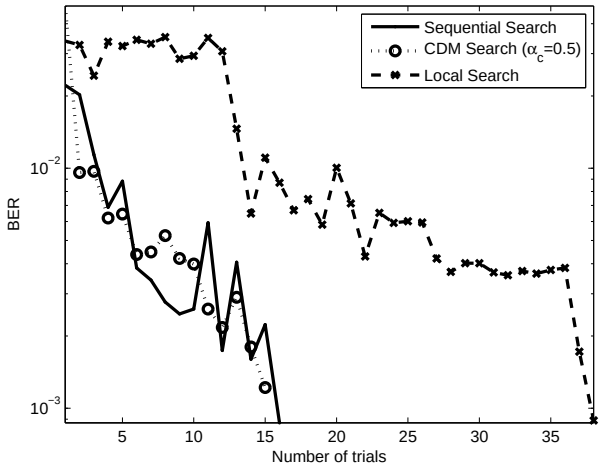



Figure 3.5: Number of iterations for optimum word length with various search algorithms in OFDM demodulator word length design.

solution in one-fourth the time that the local search method takes.

In general, if error sensitivity information for searching an optimum word length is used, the number of iterations can be reduced. The sequential search and the CDM method with an $\alpha_c$ less than 1 use the error sensitivity information. Thus, two methods converge quickly into an optimum word length that satisfies the required error performance.

### 3.4.2 Hardware Complexity

Table 3.1 and Table 3.2 show the hardware complexity according to the searched optimum word lengths in various methods. The results show that the sequential search method, which only uses error sensitivity information for the update direction, finds an optimum word length that has higher complexity than those found by the CDM method and the local search in the OFDM demodulation case study. However, an

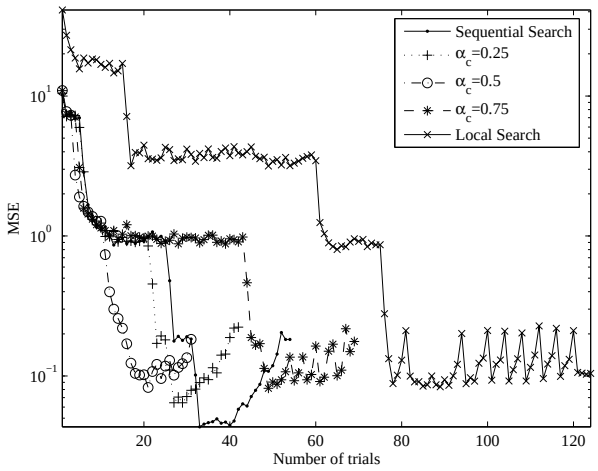



Figure 3.6: Number of iterations for optimum word length in IIR filter with various search algorithms.

optimum word length searched by the local search method, which uses hardware complexity information, has higher complexity in the IIR filter case study. If the design space is convex and has only one optimum solution, then various search methods can find the optimum solution. However, word length optimization problem has many local optimum solutions because of the nonconvex space. As the number of word length variables increases and as the system becomes more complicated, the probability of becoming stalled in a local optimum solution increases. In the IIR filter case with seven elements in the word length vector, the word lengths searched by the local search method are far from globally optimal.

In the IIR filter case study, the CDM search with a weighting factor of 0.5 finds an optimum word length that has the lowest hardware complexity. The CDM search with a weighting factor of 0.75 tends to

Table 3.3: Simulation results in noise cancellation with Wiener filter [77] of several search methods. $N = 5$, $\underline{w}_k = \{1, 1, 1, 1, 1\}$, $\overline{w}_k = \{16, 16, 16, 16, 16\}$. CDM is complexity-and-distortion measure. $\alpha_c$ is weighting factor.

| *Search Method* | $\alpha_c$ | *Number of Trials* | *Word lengths for Variables* | *Complexity Estimate* |
|---|---|---|---|---|
| Sequential [24] | 0 | 21 | {4,5,5,3,2} | 1331 |
| CDM | 0.25 | 23 | {4,4,5,4,2} | 1200 |
| CDM | 0.5 | 24 | {5,4,4,5,4} | 1074 |
| CDM | 0.75 | 167 | {4,4,4,5,4} | 1073 |
| Local [23] | 1 | 170 | {5,4,4,15,3} | 1082 |

be the local search. The hardware complexity from the CDM method with a weighting factor of 0.75 is between that of the CDM with a weighting factor of 0.5 and the local search. Similarly, the complexity from the CDM method with a weighting factor of 0.25 is between that of the sequential search and the CDM with a weighting factor of 0.5.

For more examples, additional optimum word length search results from a noise cancellation with the Wiener filter [77] are shown in Table 3.3.

### 3.4.3 Discussion

The CDM method, which uses error and complexity sensitivity for optimum word length search, has the advantages of the sequential search and the local search. This method reduces the number of iterations because of the error sensitivity that helps to reach a feasible boundary quickly. At the same time, this method finds a near-optimum word length that has lower hardware complexity because of the sensitivity

of hardware complexity. The proposed method is robust for the search of an optimum word length in a non-convex space because this method is not easily captured by local optimum solutions.

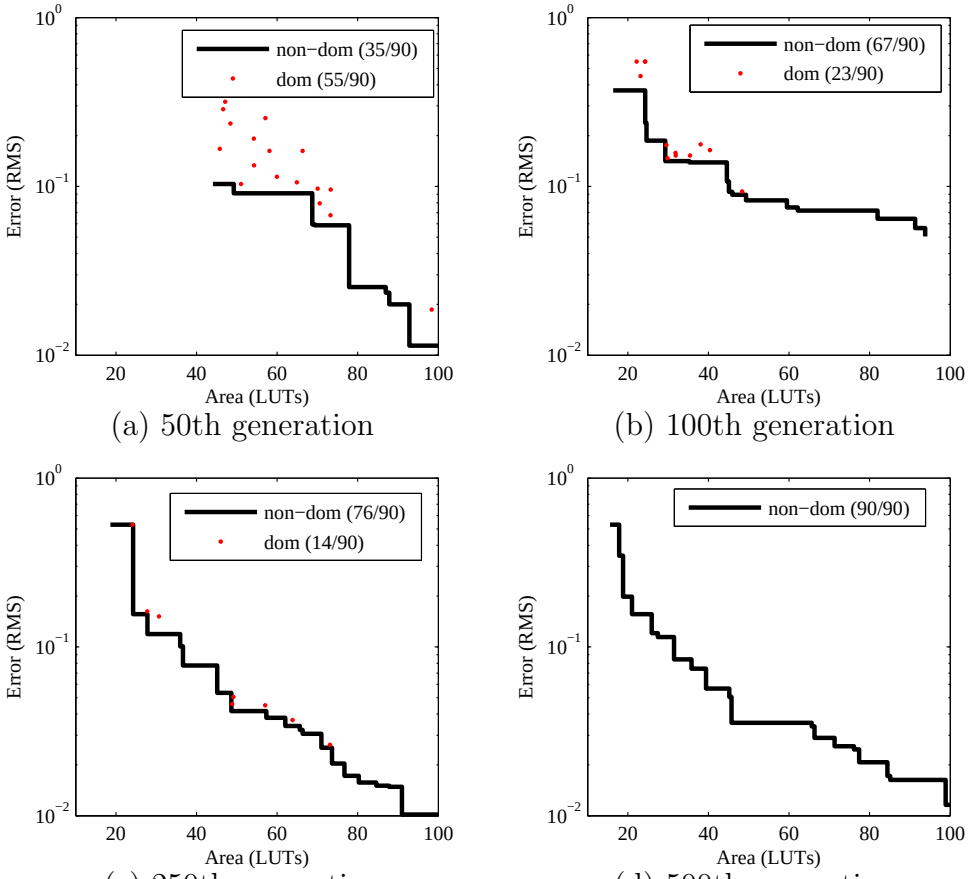
The complexity-and-distortion measure method has the flexibility to search for an optimum word length by setting a weighting factor. The designer can select the weighting factor $\alpha_c$ as in (3.5). The $\alpha_c$ of 0.5 means that the CDM method uses equally the sensitivity information of the error and of the complexity. The $\alpha_c$ of 0.5 in CDM is reasonable for optimum word length search algorithms.

For an extension of this work, various methods can be combined for word length optimization. Word length grouping [2] can be used to reduce a word length vector. An error model or error monitoring instead of error measuring can be used to reduce the simulation time. An actual cost model [13] can be used to obtain accurate results. For the search method, different methods, such as a binary search, can be combined. The pre-planned search, which is the fastest error sensitivity search method according to comparisons in [11], can employ CDM methods to reach a near-optimum word length more quickly.

## 3.5 Results of Genetic and Evolutionary Algorithms

Simulation results utilizing multiple objective genetic and evolutionary algorithms on word length design in an IIR filter are shown in Fig. 3.7 and Fig. 3.8. The total hardware area, one of the objectives, is evaluated by the area model of the arithmetic unit in [12]. The error between the floating-point output and fixed-point output are measured by simulations.

The results in the IIR filter with seven word length variables are shown in Fig. 3.7. Since the genetic algorithm mimics the evolution-




(a) 50th generation (b) 100th generation

(c) 250th generation (d) 500th generation

Figure 3.7: Results of multiple objective genetic and evolutionary algorithms in the IIR filter case study with seven word length variables (Population for one generation is 90)

ary process of plants and animals, each generation shows different results. Fig. 3.7 shows nondominated and dominated (inferior) solutions at each generation. The plot of the objective functions whose nondominated vectors are in the Pareto optimal set is called the Pareto front. After many generations, the Pareto front tends to move toward the left and downward. The number of dominated solutions decreases as the number of generations increases. The 500th generation has only non-dominated solutions.

Designers have a choice of word length solutions according to the Pareto front. The Pareto front gives the tradeoff curve in hardware area and finite precision output error. A smaller area requires a larger error, and a larger area needs a smaller error. Thus, the obtained Pareto front gives designers flexibility in a system design.

Note that the Pareto front in a descendant is not always better than that in ancestor. The solution for an error of $10^{-2}$ at the 250th generation required 90 lookup tables (LUTs). However, the solution for the same error at the 500th generation needs at least 100 LUTs. Thus, the 250th generation has a better solution for the error of $10^{-2}$ than the 500th generation. The offspring could be worse than their ancestors because the genetic and evolutionary algorithm utilizes a random process throughout selection, mating, and mutation.

The genetic and evolutionary algorithm is computationally intensive. It requires many simulations for errors in each population as well as the genetic operations of selection, mating, and mutation for each generation. Considering only the number of simulations for errors, the 500th generation requires 45000 ($= 500$ generations $* 90$ populations) simulations.

A reduced number of variables can reduce the number of simulations. Fig. 3.8 shows results in the IIR filter study with three word length variables. The word length at the output of the multipliers are selected for three variables. The results show the same trends as with seven word length variables. However, all solutions at the 250th

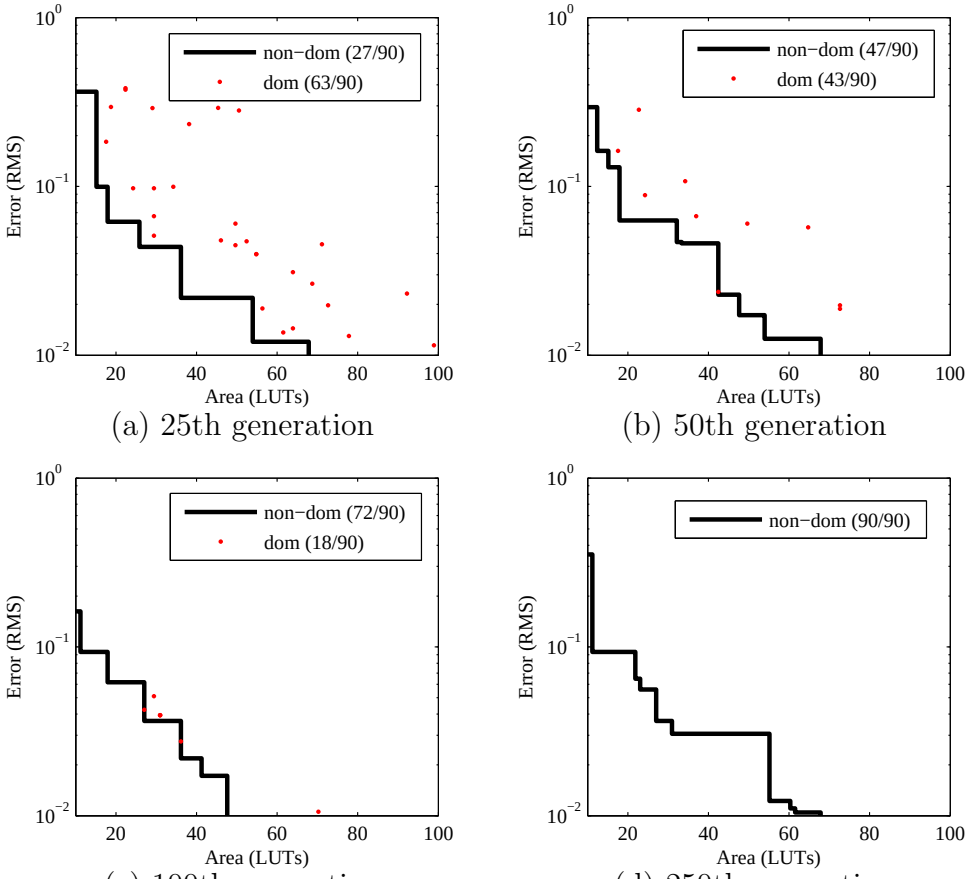
generation are non-dominated. Thus, 22500 (= 250 generations $*$ 90 populations) are sufficient for three variables in this case study.

The Pareto fronts in Fig 3.7 and Fig 3.8 are calculated based on the current generation. Offspring sometime show worse Pareto fronts than their predecessors. For example, the Pareto front at the 250th generation in Fig. 3.8 looks partly worse than that at the 100th generation. Storing and comparing non-dominated solutions in all generations can generate a better Pareto front.

## 3.6 Comparison

The gradient-based search algorithms and genetic algorithms are compared. Results from gradient-based search algorithms with the FPGA area models are superposed on the results from a genetic algorithm in Fig. 3.9 and Fig. 3.10. Three desired root mean square (RMS) values of 0.08, 0.1, and 0.12 are given, since the gradient-based search algorithms generate one solution each. DM, CDM, and CM are used as gradient-based search algorithms. Automated transformation tools from floating-point to fixed-point, which are explained in Section 5, are used for the gradient-based search algorithm and the genetic algorithm as a search engine. One word length variable at each output is generally assigned for the automated transformation in the IIR filter. For example, in Fig. 3.3, the delay output has one word length variable instead of two word length variables, and the one word length variable is added to the output of the IIR filter. Seven word length variables are used in simulations of the fixed-point IIR filter.

The solutions from the gradient-based search algorithms are similar to the Pareto front in the genetic algorithm at the 50th generation. The RMS error of 0.1 needs approximately 50 LUTs in both methods. However, at the 500th generation the genetic algorithm finds better solutions than the gradient-based search algorithms, as shown




(a) 25th generation (b) 50th generation

(c) 100th generation (d) 250th generation

Figure 3.8: Result of multiple objective genetic and evolutionary algorithms in the IIR filter case study with three word length variables (Population for one generation is 90)

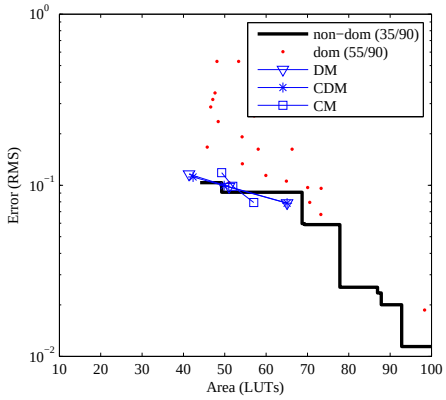



Figure 3.9: Overlap genetic evolutionary algorithm results in 50th generation with gradient-based search results for the IIR filter case study with seven word length variables (Required RMS for gradient-based search $\leq \{0.12, 0.1, 0.08\}$

in Fig. 3.10. This difference demonstrates that the gradient-based search methods are trapped by local optima. However, the genetic algorithm can avoid local optima.

With the respect to computational complexity, gradient-based search algorithms need smaller numbers of computation compared to the genetic algorithm. The gradient-based search algorithms require 145 simulations for CDM and 417 for CM, whereas the genetic algorithm needs 4,500 simulations to obtain a similar result and 45,000 simulations for the 500th generation. Furthermore, the genetic algorithm requires computations to execute genetic operations at every generation.

A random search algorithm, which randomly selects samples or candidates, can find solutions. For comparision with the random search algorithm, word length samples of 45,000 are randomly selected.

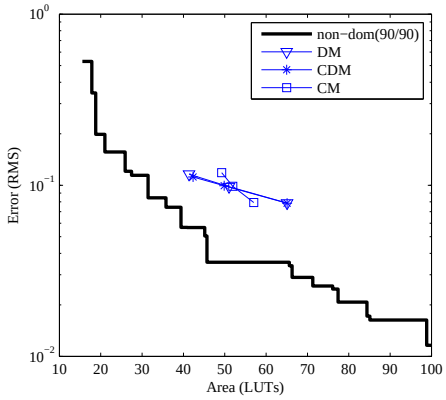



Figure 3.10: Overlap genetic evolutionary algorithm results in 500th generation with gradient-based search results for the IIR filter case study with seven word length variables (Required RMS for gradient-based search $\leq \{0.12, 0.1, 0.08\}$

The Fig. 3.11 shows the result of random search. The Pareto front of the genetic algorithm as shown in Fig. 3.7 (d) is better than that of the random search algorithm as shown in Fig. 3.11. This simulation result shows that the genetic algorithm outperforms the random search algorithm with the same number of samples.

Parallel processing can decrease the running time. The genetic algorithm is amenable to parallel and distributed simulations. The genetic algorithm can be parallelized up to the number of populations since individuals can be evaluated independently. However, gradient-based search algorithms are limited in their use of parallelism because gradient-based search algorithms evaluate the next neighbors and move a current point to one of the neighbors. Thus, the gradient-based search algorithm can be parallelized only up to the number of neighbors or word length variables.

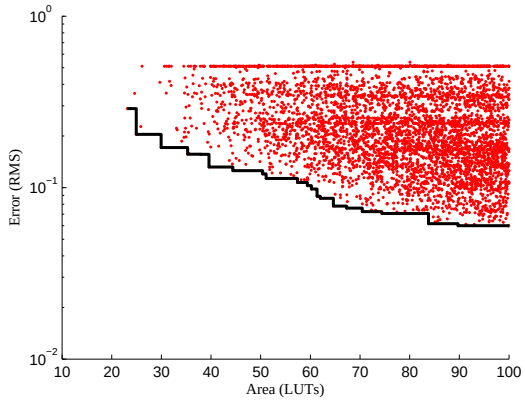



Figure 3.11: Result of random search algorithm in the IIR filter study with seven word length variables (45,000 samples)

## 3.7 Summary

Word length optimizing methods with sensitivity measures are generalized by equations and compared. The proposed complexity-and-distortion measure equation can express the local search or sequential search by changing the weighting factors. The weighting factor can reduce the number of iterations and the hardware complexity compared to those of the local search and the sequential search, respectively. In our case studies, the complexity-and-distortion method is simulated and compared. The proposed method can find the optimum solution in one-fourth of the time that the local search takes. In addition, the optimum word length searched by the proposed method has 30% less hardware implementation costs than the sequential search in wireless demodulators.

Multi-objective genetic and evolutionary algorithms provide a Pareto front with which designers can decide on an optimum word length set

with tradeoffs in signal distortion and hardware complexity. When the genetic algorithms are compared with the search algorithms, the word length from the gradient-based search algorithms are local optima. However, genetic algorithms have long running time.

In conclusion, word length search algorithms utilizing genetic and evolutionary algorithms can optimize the tradeoff between signal quality and implementation complexity. Alternatively, word length search algorithms utilizing gradient information can provide faster ways to find data word lengths, but they get caught in local optima.

# Chapter 4

# Word Length Reduction for Lowering Power Consumption

## 4.1 Introduction

Computing systems demand the minimization of power dissipation, because of limited battery power in portable computing and the difficulty of cooling during high-speed signal processing. Many methods have been developed to reduce power consumption. Lowering the supply voltage and minimizing the hardware are used for low-power hardware [78]. Changing the instruction order and reducing the number of operations are used for low-power software [79]. A major focus of low-power design is to reduce switching activity to the minimal level required to perform the computation, since, to a first order, the power consumption of CMOS circuits is proportional to the number of gate transitions [80].

Multipliers are usually a major source of power consumption in

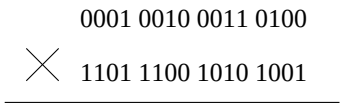
typical DSP applications. Multiprecision multipliers have been developed for low-power consumption [81,82]. In multiprecision multipliers, multiplications are performed by 8-bit, 16-bit, or 24-bit circuits, according to the input operand size. Power reductions of up to 66% are achieved in [81] and 58% in [82]. However, arbitrary operand sizes such as 10 bits are not accommodated efficiently in these approaches. A word length reduction technique has been proposed in [83] to select any word size. The word length reduction technique shows a 72% reduction of average gate transitions. An extension of the word length reduction technique is presented in this chapter.

Overviews of word length reduction techniques and power reduction methods are presented in Sections 4.2 and 4.3, respectively. Expectation values of bit switching in inputs are derived in Section 4.4. A radix-4 modified Booth multiplier and a Wallace multiplier, which are used in simulations, are explained Section 4.5. Power consumption in these multipliers is estimated for FPGA implementations in Section 4.6. Also, the power consumption of multipliers where the operands are of different sizes is estimated and compared.

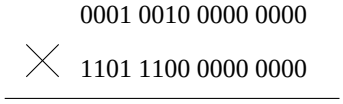
## 4.2 Word Length Reduction

Multiprecision multipliers have a few choices of operand precision due to hardware limitations [81, 82]. The multiprecision multiplier does not accommodate arbitrary precision, because of its fixed hardware structure. For example, with 10-bit operands, a multiprecision multiplier, which supports 8-bit and 16-bit multiplication, has to use 16-bit multiplication with 6 unnecessary bits. Data word length reduction techniques can reduce the unnecessary switching activity.

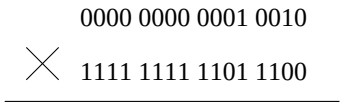
There are two kinds of data word length reduction. One is reduction via right-shifting, and the other is reduction via left-shifting, that is, with truncation. The right-shifting method moves data from




(a) Original multiplication




(b) Reduction by truncation




(c) Reduction by signed right shift

Figure 4.1: Example of 8-bit data word length reduction

the most significant (MS) side to least significant (LS) side with sign extension. The sign extension bits are all 1s when the operand is negative and all 0s when the operand is positive. The truncation method removes data from the LS side. An example of an 8-bit reduction from 16-bit multiplication is shown in Figure 4.1. The original 16-bit multiplication is shown in Figure 4.1(a). The reduction by an 8-bit right-shift moves 8 bits data in the MS side to the LS side with sign extension as shown in Figure 4.1(b). The signed right shifted value becomes 1111 1111 1101 1100, because the original value, 1101 1100 1010 1001, is negative. The reduction by 8-bit truncation removes the 8-bit data in the LS side by masking the input data with 1111 1111 0000 0000, with the result shown in Figure 4.1(c).

## 4.3 Power Consumption

### 4.3.1 Power Analysis

There are three major sources of power dissipation in digital CMOS circuits that are summarized in the following equation [80]:

$$P_{avg} = P_{switching} + P_{short-circut} + P_{leakage}. \tag{4.1}$$

The first term represents the switching component of power, the second term derives from the direct-path short-circuit current conducting directly from the supply to the ground, and the leakage power is primarily determined by the fabrication technology.

The switching component of average power is

$$P_{switching} = \alpha C_L V_{dd}^2 f_{clk} \tag{4.2}$$

where $\alpha$ is the switching activity parameter, $C_L$ is the load capacitance, $V_{dd}$ is the operating voltage, and $f_{clk}$ is the operating frequency. The switching power can be reduced through operation reduction, choice of number representation, exploitation of signal correlations, logic design, and physical design. The switching activity can also be reduced by optimizing the ordering of operations and by minimizing the number of operations.

The term $\alpha C_L$ can also be viewed as the effective switching capacitance of the transistor nodes from charging and discharging. Therefore, minimizing switching activities can effectively reduce power dissipation without impacting the operational performance of the circuit [84].

Directly measuring the power consumption is difficult. The average number of transitions is usually used as an estimate of the requirement.

### 4.3.2 Software Power Minimization

Tiwari *et al.* [85] attempted to systematically model the software power cost, because of the increasing demand for a software power analysis tool. They formulated an instruction-level power model for the microprocessor after measuring the power of the instruction sets. This approach made it possible to compare programs in terms of their energy consumption.

Lee *et al.* [79] developed power analysis and minimization techniques for embedded DSP software. They found that in typical DSP applications, the multiplier in the multiply-and-accumulate (MAC) unit is usually a major source of power consumption. A micro architectural power model for the multiplier was developed and analyzed for further power minimization. They observed a wide power variation of MAC instructions mainly according to the two values being multiplied in the MAC unit.

They also used the operand-swapping technique for a Booth multiplier [86], which does not treat the two inputs symmetrically. Their experiment showed that swapping the operations in Register A and B can reduce the power for MAC instructions. They also used instruction packing, instruction scheduling, and memory bank assignment to reduce energy consumption.

### 4.3.3 Minimizing Word Length for Low Power

Chandrakasan *et al.* [87] showed that the word length affects all key parameters of a design, including speed, area, and power. Choi and Burleson [23] presented a general search-based methodology for word length optimization and used a switching power model for power dissipation. Considering a voltage-dropping factor and the area of the computing elements according to the word lengths, they analyzed the switching power consumption, assuming that the power dissipation

was proportional to the area of computing element.

Erdogan and Arslan [88] showed low-power multiplication schemes for finite impulse response (FIR) filters on DSP processors. They used a data bus and a coefficient bus separately for the filtering operation. They measured the switching activity of 8-, 16-, and 32-bit array multipliers for filter orders of 32, 64, and 128. They achieved up to a 63% reduction in switching activity by ordering of coefficient and using a pre-calculated value memory.

Chen *et al.* [84] presented low-power two's complement multipliers by minimizing the switching activities of partial products using the Radix-4 modified Booth algorithm [89]. They used the fact that the switching activities of the unused functional blocks are minimized when the input bits of unused functional blocks remain unaltered. They increased the probability that the partial products become zero by swapping input data.

Word length can also be changed by reconfiguring the multiplier. Kim and Papaefthymiou [90] proposed a reconfigurable pipelined multiplier architecture by adapting its structure to computational requirements over time. It can efficiently cope with variable data-rate multimedia applications such as video processing. The multiplier structures can dynamically reconfigure to lower their power consumption based on zero-valued inputs and input-rate variations.

### 4.3.4 Power Reduction via Word Length Reduction

Minimizing switching activity can effectively reduce power dissipation without impacting circuit performance [84]. Word length reduction methods in Section 4.2 can minimize switching activity at the expense of data precision, as in [83]. The minimized switching activity reduces power consumption as shown in (4.2).

The word length reduction methods can be applied to low-power instruction-based processors or FPGA/reconfigurable hardware. The truncation method is implemented by adding mask modules, which consist of N-bit AND gates, in front of the multiplier inputs. The signed right-shift method uses shift registers and sign extension units. Therefore, the truncation method needs less extra hardware than the signed right-shift method for its implementation.

## 4.4 Expectation of Switching

Power consumption in CMOS digital circuits is proportional to switching activity in logic gates. Logic gates in multipliers are switched after input multiplicand data are changed from the previous data. The total number of gates that switch is used to calculate switching power consumption. It is difficult to predict the overall number of gates that switch in a multiplier because of the glitch effect, which unexpectedly increases the switching activity.

Multiplicand inputs propagate the switching activity into inner logic gates in a combinational multiplier. The expected value of input switching is a meaningful factor to predict the number of gates that switch in a multiplier. In this section, the expected value of the number of gates is estimated that switch in $L$-bit inputs and $M$-bit reduction by truncation or signed right-shift methods.

### 4.4.1 $L$-Bit Input

Let $X$ be a random variable of the number of total bits switched in word length $L$, as in Fig. 4.2. Each bit in the data has equal probability of bit switching such as zero to one or one to zero, when new input data are given in previous data locations. The probability of the switching of each bit is $\frac{1}{2}$. The switching probability in $X$ has

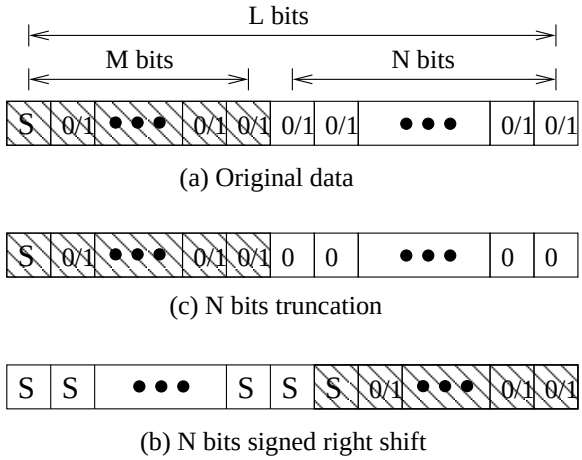



Figure 4.2: Bit operation in effective bits, $M$. $S$ is a signed bit

binomial distribution:

$$P_X(x) = \binom{L}{x}(\frac{1}{2})^x(\frac{1}{2})^{L-x} \tag{4.3}$$

The expected value of $X$ is

$$E(X) = \sum_{x=0}^{L} x \cdot P_X(x) \tag{4.4}$$

The expected value of a binomial distribution with probability $p$ and the number of trials $l$ is $l \cdot p$. The expected value of switching in $L$ bits can be simplified to

$$E(X) = L \cdot p \tag{4.5}$$

$$= \frac{L}{2}. \tag{4.6}$$

Thus, expected value of switching in $L$ bits is half of $L$ bits.

### 4.4.2 $N$-Bit Truncated Data in $L$-Bit Input

The effective bit width can be reduced by truncation. When truncated data are consecutively used as input data, only the remaining bits have a probability of switching, as shown in Fig. 4.2(b). $N$-bit truncated data in $L$-bit width input have $L - N$ effective width to be switched, whereas $N$ bits have always zero values. The expectation of $N$-bit truncated data in $L$-bit inputs is

$$E_{tr}(X) = \frac{L-N}{2} \tag{4.7}$$

$$= \frac{M}{2} \tag{4.8}$$

where $M$ is the number of bits that are not truncated. These equations show that the expectation value of switching in truncated data is half of the remaining data width.

### 4.4.3 Signed Right Shift

The effective bit width can be reduced by right shifting. The signed right shift moves data to right side with the sign bit filled into the vacated bit positions. $N$-bit signed right-shifted data in $L$-bit input add $N$ additional sign bits, as shown in Fig. 4.2 (c). The expected value of switching in $N$-bit signed right-shifted data can be obtained using a conditional expectation [91] with a random variable, Y, of a sign bit switching as follows:

$$E_{rs}(X) = E(E(X|Y)) \tag{4.9}$$

$$= \sum_{s=0}^{1} P(Y=s)E(X|Y=s) \tag{4.10}$$

$$= \frac{1}{2}E(X|Y=0) + \frac{1}{2}E(X|Y=1) \tag{4.11}$$

Where: $s$ is the sign bit. The first term in the right side in (4.11) gives the expected value when the sign bit is not changed. Thus, only $M-1$ bits change. From Eqs. (4.3), (4.4), and (4.6), the first term of conditional expectation value (4.11) becomes

$$E(X|Y=0) = \sum_{x=0}^{M-1} x \cdot \binom{M-1}{x} (\frac{1}{2})^x (\frac{1}{2})^{M-1-x} \tag{4.12}$$

$$= \frac{M-1}{2} \tag{4.13}$$

where $M = L - N$.

The second term in the right side in (4.11) is the conditional expectation when the sign bit is switched. The $N$-bit signed right-shifted data have $N+1$ sign bits as shown in Fig. 4.2 (c). The conditional expectation of the switched sign bit, $E(X|Y=1)$, is

$$\sum_{x=0}^{M-1} (x+N+1) \binom{M-1}{x} (\frac{1}{2})^x (\frac{1}{2})^{M-1-x} \tag{4.14}$$

The $x$ in the summation in (4.14) can be separated as

$$\frac{M-1}{2} + \sum_{x=0}^{M-1} (N+1) \binom{M-1}{x} (\frac{1}{2})^x (\frac{1}{2})^{M-1-x} \tag{4.15}$$

$$= \frac{M-1}{2} + \sum_{x=0}^{M-1} (N+1) \binom{M-1}{x} (\frac{1}{2})^{M-1} \tag{4.16}$$

In general, the sum of all the combinations of $K$ distinct items gives

$$\sum_{x=0}^{K} \binom{K}{x} = 2^K \tag{4.17}$$

Table 4.1: Expectation of switching in $L$ bit input

| Inputs | Expectation of switching |
|---|---|
| Full length used | $L/2$ |
| $N$ bit truncation | $M/2$ |
| $N$ bit signed right shift | $L/2$ |

Using (4.17) and (4.16) yields the conditional expectation value as follows:

$$E(X|Y=1) = \frac{M-1}{2} + (N+1)(\frac{1}{2})^{M-1}2^{M-1} \quad (4.18)$$

$$= \frac{M}{2} + N + \frac{1}{2} \quad (4.19)$$

From (4.13) and (4.19), the expectation of switching data in (4.11) can be simplified to

$$E_{rs}(X) = \frac{1}{2}(\frac{M-1}{2}) + \frac{1}{2}(\frac{N}{2} + N + \frac{1}{2}) \quad (4.20)$$

$$= \frac{M+N}{2} \quad (4.21)$$

$$= \frac{L}{2} \quad (4.22)$$

The expected value of the number of bits switched in $N$-bit signed right-shifted data in $L$-bit input is half of $L$ regardless of the signed right shift. Therefore, the expected value of switching in signed right-shifted input is the same as for an unshifted input. The expected values are summarized in Table 4.1 and shown in Fig. 4.3.

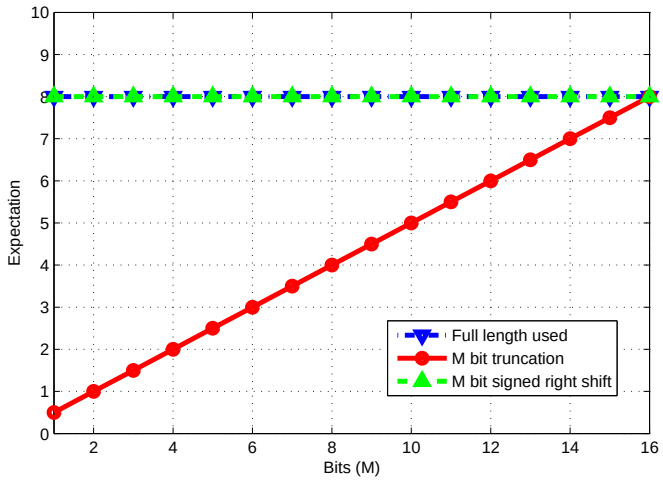



Figure 4.3: Expectation of number of switching bits in inputs

## 4.5 Multiplier

The hardware multiplier on most programmable DSPs uses either a Wallace multiplier or a Radix-4 modified Booth multiplier [89]. For example, the TI TMS320C64 uses the Wallace algorithm, and the TI TMS320C62 uses the Radix-4 modified Booth algorithm.

### 4.5.1 Wallace Multiplier

In a tree-based multiplier, partial products are added using full adders or half adders. In 1964, Wallace showed a tree structure, which is an efficient method to add partial products [92]. A Dadda dot diagram of a 4-bit Wallace multiplier is shown in Figure 4.4. Rows are grouped into sets of three during each reduction stage. Within each three-row set, (3,2) counters reduce columns with three bits to two bits

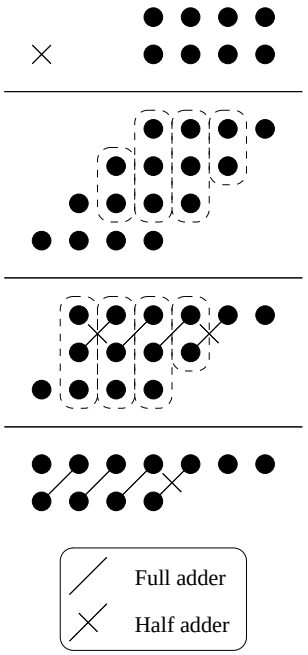



Figure 4.4: Dadda dot diagram for a 4-bit Wallace multiplier

and (2,2) counters reduce columns with only two bits. Rows that are not part of a three-row set are transferred to the next stage without modification [93].

### 4.5.2 Radix-4 Modified Booth Multiplier

Booth recoding is a commonly used technique to recode one of the operands in binary multiplication. Fig. 4.5 shows a radix-4 modified Booth multiplier of $a \times x$. A two's complement multiplier, $x$, is recoded as a radix-4 number, $z$, that dictates the multiples -2a, -a, 0, a, and 2a to be added to the cumulative partial product. The radix-4 Booth's

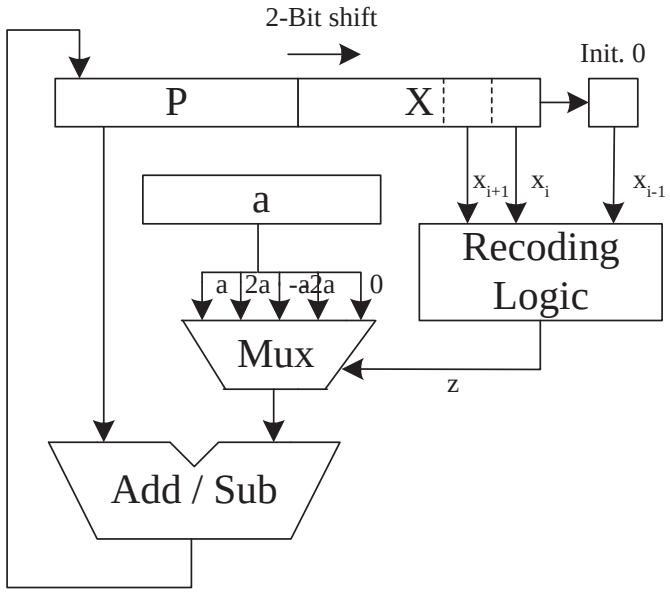



Figure 4.5: A Radix-4 multiplier based on Booth's recoding. The $a$ and $x$ are multiplicands. $P$ is product of multiplication. Three bits in $X$ are recoded to $z$.

recoding is shown in Table 4.2.

## 4.6 Simulation Results

A 16-bit Wallace multiplier and a 16-bit Radix-4 modified Booth multiplier are used for power estimation with data word length reduction. The multipliers are synthesized for Xilinx, XC3S200-5FT256 FPGA [94]. The XPower tool estimates the power consumption of

Table 4.2: Radix-4 Booth's recoding. The $a$ and $x$ are multiplicands. Three bits of $x$ are recoded into $z$.

| $x_{i+1}$ | $x_i$ | $x_{i-1}$ | $z$ | *action* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | a |
| 0 | 1 | 0 | 1 | a |
| 0 | 1 | 1 | 2 | 2a |
| 1 | 0 | 0 | -2 | -2a |
| 1 | 0 | 1 | -1 | -a |
| 1 | 1 | 0 | -1 | -a |
| 1 | 1 | 1 | 0 | 0 |

this FPGA with different operand sizes. The dynamic power is estimated across VCCINT, which is a power-supply pin of the dedicated internal core with a 1.2 V supply. The operational frequency of the multipliers is set to 1 MHz.

Power estimates for a 16-bit Wallace multiplier are shown in Figure 4.6. An average power of 0.45 mW is consumed with 16-bit data operands in the Wallace multiplier. As the operand size is reduced, the truncation method decreases the power consumption. The average power reduction in 8-bit word length reduction by the truncation method is 56%. The right-shift method shows little or no power reduction due to the sign extension. The extended sign bits are added to the input whenever a right shift occurs. These bits affect the switching activity. Therefore, the signed right-shift method is not recommended for low-power Wallace multipliers.

Power estimates for a 16-bit radix-4 modified Booth multiplier are shown in Figure 4.7. A power of 0.52 mW is consumed with 16-bit operands in the Booth multiplier. As the data word length is reduced

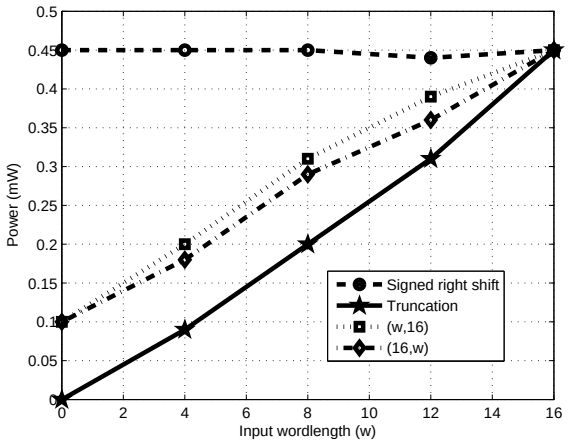



Figure 4.6: Dynamic power consumption in 16-bit × 16-bit Wallace multiplier (1MHz)

by either the truncation method or the signed right-shift method, the average power consumption decreases. The average power consumption for multipliers with 8-bit operands implemented by the signed right-shift and the truncation methods are decreased by 25% and 31%, respectively.

The power consumption for the Wallace multiplier as shown in Figure 4.6 shows a trend that matches the expectations from Figure 4.3. The amount of switching is not changed in signed right-shift input, but it is changed in truncated input as the effective input word length changes. However, in the Booth multiplier, the power consumption of the signed right-shift input as shown in Figure 4.7 is changed as the input effective word length changes.

The average power consumption is also estimated when operands have unequal sizes. One of the operands is reduced with the truncation method, while the other operand is fixed at 16 bits. The first and the

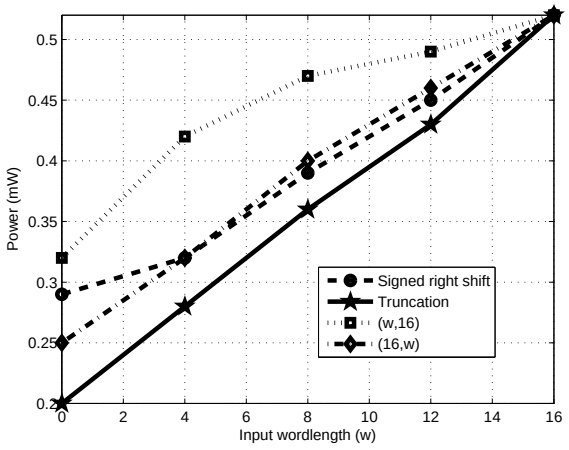



Figure 4.7: Dynamic power consumption in 16-bit × 16-bit Radix-4 modified Booth multiplier (1MHz)

second element in the parentheses in Figure 4.6 and Figure 4.7 represent two multiplicands in multiplication. When operands are swapped, such as (A by X) to (X by A), the power consumption shows different results. For the Wallace multiplier, there is a small difference, but the Booth multiplier has a large difference because of its asymmetric structure. The first and the second operand in the Booth multiplier represent a recoded input, $X$, and a non-recoded input, $A$, respectively, as shown in Fig. 4.5. The result shows that when the non-recoded level of input precision is reduced, the average power decreases by 13% more than when the recoded input is reduced for an 8-bit word length reduction. The reason is that the non-recoded input, which is routed to multiplexers and to adder/subtracter logic, affects more power consumption than the recoded input. Therefore, in the Booth multiplier, data word length reduction in the non-recoded operand achieves more power reduction than that in the recoded operand.

## 4.7 Summary

Two kinds of input data word length reduction methods in multipliers have been examined and analyzed for low power consumption. A truncation method with 8 bits reduces power consumption by 56% in a 16-bit Wallace multiplier and 31% in a 16-bit radix-4 modified Booth multiplier. A signed right shift method exhibits no power reduction in the Wallace multiplier and 25% reduction in the Booth multiplier. When the operands have different sizes, the multipliers also show power reduction. In particular, the non-recoded operand in the Booth multiplier is 13% more sensitive in power consumption than the recoded multiplicand. This difference can be exploited in a low-power digital filter design with low-precision coefficients.

# Chapter 5

# Automating Transformation to Fixed Point in Software

## 5.1 Introduction

Realization of digital signal processing algorithms with the fixed-point data type provides many benefits, such as savings in power and area. Typically, however, the algorithms are developed with the floating-point data type and later transformed to the fixed-point data type with tradeoffs in signal precision and complexity. Since the transformation process is time consuming and error prone, many methods have been proposed and developed to automate the fixed-point transformation [3, 4, 11, 13, 15, 24].

Fixed-point transformation consists of fixed-point conversion and word length optimization. Fixed-point conversion is a process of converting floating-point programs to fixed-point programs. During the conversion, floating-point data type is changed into the fixed-point data type, and floating-point arithmetic operations are modified to fixed-point arithmetic operations.

Fixed-point digital signal processors (DSP), in which word lengths have already been given, require fixed-point conversion followed by scaling to prevent the overflow and underflow of signals. Word length optimization is required only when the given word length is too short to satisfy the desired performance or to reduce power consumption, as explained in Chapter 4.

For the customized IC or FPGA, word lengths of digital systems can be chosen to any number of widths with tradeoffs in objectives. Shorter width usually achieves savings in area and power, while signal distortion is higher. Word length can be optimized by optimization algorithms [2, 11–13, 15, 24] and those algorithms can also automate the optimization process.

Most of the algorithms minimize hardware area by satisfying error specifications. Sometimes, designers make tradeoffs between error specifications and hardware area instead of fixing one objective. This book proposes multi-objective word length optimization, which optimizes more than one objective at the same time. Furthermore, an environment for automating fixed-point transformation is proposed and demonstrated with a case study.

## 5.2 Related Work

### 5.2.1 Fixed-Point Simulation Environment

Many methods have been developed to model fixed-point systems. TI developed a fixed-point data type in the C++ class to develop fixed-point DSP algorithms [95]. In [3, 96], a fixed-point simulation environment is implemented for C and C++ at Seoul National University. By modifying variable declarations of the floating-point code and overloading operators in the *gFix* class, the floating-point data type is converted to the fixed-point data type. A range estimation

utility, which estimates statistical data range, is also developed.

In [1], annotation and interpolation techniques to convert the floating-point data type to fixed-point data type with an analytical range estimation are employed. A commercial tool, the CoCentric Fixed-point Designer, proposed by Synopsys, is based mainly on the technology developed in the FRIDGE project [17].

There are many commercial tools for fixed-point simulation environments. The Hardware Design System (HDS) developed by CoWare [5] provides library, which enables the definition of the design at the hardware implementation level. The Fixed-Point Toolbox in MATLAB and the Fixed-Point Blockset in Simulink [97,98] were developed by MathWorks. AccelChip [99,100] released an automatic fixed-point simulation environment.

In addition, code conversion tools for DSP have been developed [101,102]. In [101], for instance, an integer code generator based on the FRIDGE environment is developed.

### 5.2.2 Word Length Optimization

Sung and Kum [2] developed a simulation-based word length optimization algorithm. Optimum word lengths are searched from a minimum word length state by increasing word lengths with priority on a hardware block having the lowest hardware cost.

In [12], an area model and a noise model are proposed. For an objective in optimization, area-based objective functions and error models are used. A mixed integer linear programming (MILP) model and heuristic search methods are employed to solve the word length optimization problems.

Shi and Brodersen [13] use simulation-based methods to evaluate various sensitivities. Simulation results are used to develop a model of the system that is used in the optimization. The Mosek optimizer, which handles single objective optimization, is used as a search engine.

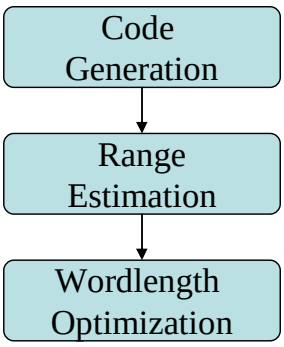



Figure 5.1: Three phases in automating transformation from floating point to fixed point

This book presents an environment for automated floating-point-to-fixed-point transformation that includes a fixed-point conversion and word length optimization.

## 5.3 Automating Transformation from Floating Point to Fixed Point

As shown in Fig. 5.1, the transformation from floating point to fixed point has three phases: code generation, range estimation, and word length optimization. A code generator converts floating-point programs to fixed-point programs that handle fixed-point data types and arithmetic. The code generator also creates other auxiliary programs for an automatic transformation environment. A range estimator finds range information in the fixed-point system to prevent overflow and underflow. Word length optimization finds the optimum word length according to objectives such as signal distortion and hardware complexity.

```
Function c = adder(a, b)
c = 0;
c = a + b;
```

(a) Floating point program for adder

```
Function [c] = adder_fx(a, b, numtype, mathtype)
c = 0;
a = fi (a, numtype.a, mathtype.a);
b = fi (b, numtype.b, mathtype.b);
c = fi (c, numtype.c, mathtype.c);
c(:) = a + b;
```

(b) Converted fixed-point program for adder (only the core code is shown)

Figure 5.2: Conversion to fixed point by a code generator

### 5.3.1 Code Generation

The first process in the automating transformation to fixed point is code generation. A given floating-point program shown in Fig 5.2 (a), is converted to a fixed-point program, which can handle variable word lengths, by a code generator after analyzing the given floating-point program. The variable word length can be realized by a parameterized input, as shown in Fig. 5.2 (b).

The *fi* is one of the functions in the Fixed-Point Toolbox in MATLAB to define a fixed-point data type [97]. Each fixed-point variable is defined via an input parameter and number type instead of constants. This input parameter is controlled by word length optimization programs.

The code generator also creates several programs for a fixed-point

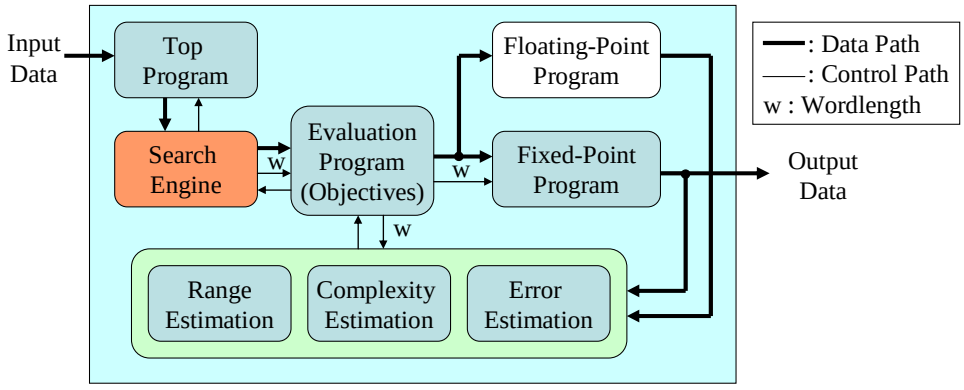



Figure 5.3: Automated transformation environment

transformation environment, as shown in Fig. 5.3. The top program generated by a code generator plays a role as headquarters in the transformation to fixed-point. It establishes an environment for a range estimation and optimum word length search according to configurations that can be modified by designers. The top program mainly executes range estimations and word length optimizations.

The optimum word length depends on the input signal properties. The input signal is passed by the top code, search engine, and objective code. The top code calls a search engine, which explores the word length space to find the optimum word length. The genetic algorithm can be used as a search engine for multi-objective optimization.

The objective code collects objective values according to word length states and input signal. One of objectives can be a signal error, which is the difference between the floating-point output and fixed-point output. Complexity, power consumption, or timing information can be used as a objective.

### 5.3.2 Range Estimation

Range information is used to determine integer word length in order to prevent overflow and underflow. A signal range can be estimated by two methods. One is a simulation-based method and the other is an analytical method. A simulation-based method monitors the signal range of variables and finds a maximum value and a minimum value. An analytical method calculates signal ranges by using a range-propagation property through operations. Simulation is not necessary in the analytical method. However, the calculated result from an analytical method is conservative, and word length could grow infinitely in feedback systems. A simulation-based method is useful for complicated systems, including loops; however, it needs time for the simulation. Both methods can be used selectively. The simulation-based method can be used in feedback parts, and the analytical method can be used in other parts.

### 5.3.3 Optimum Word Length Search

Optimum word length can be found with word length optimization algorithms that have search algorithms, such as a complete search, or sequential search, as described in Chapter 2. One of the more powerful search engines is the genetic and evolutionary search engine. The genetic search engine handles multi-objectives and finds a Pareto front, although the computation complexity of this algorithm is very high.

The search engine generates word length candidates, which the evaluation function then evaluates. The information of the objective values could be used to generate the next candidates. For the evaluation, the error value or difference value between the floating-point programs and fixed-point programs can be obtained by an analytical or statistical approach. The analytical approach models the error and

estimates the error at each system output. In the statistical approach, simulation is used to estimate the error.

Cost value can be obtained by modeling the fixed-point systems. Modeling the exact implementation scheme used would be specific to the vendor. Area models in [12, 103] are used for complexity estimation.

## 5.4 Case Study

A multiplier and accumulator (MAC) is commonly used in digital signal processing such as filtering. Floating-point programs can be converted to fixed-point programs by a code generator. One of the options in conversion is that all variables and arithmetic units are converted to fixed-point representations. The other option is that the designated variables, which are indicated by a symbol, are converted. For this case study, the second option is developed. Fig. 5.4 shows an example of the MAC program in floating point MATLAB. The *fx* in the comments is a known symbol directing a code generator to change floating-point variables into fixed-point variables.

The converted fixed-point code is shown in Fig.5.5. First, the function name is changed by appending *_fx*, and two input parameters, *numtype* and *fimathtype*, and one output parameter are inserted in the first line. The variable, *numtype*, has word length information regarding variables. The variable *fimathtype* has a fixed-point arithmetic property. The output parameter is used for range information.

An initialization code is also inserted. The designated variables can be initialized according to the transformation phase. In the range estimation phase, word lengths are assigned to a high precision. In the other phase, given word lengths are assigned to each variable in fixed-point data type. For example, the variable *acc* is assigned to fixed-point data type according to information in the *numtype.acc* variable

```
Function acc = mac(in);

acc = 0;            % fx
coef = 0.1;         % fx

for i= 1:length(in)
    t1 = in(i);     % fx
    t2 = t1 * coef; % fx
    acc = acc + t2;
end
```

Figure 5.4: Example of MAC floating-point program

during the other phase. At the end of the converted code, the *fxlog* function returns the range of each variable.

In the assignment, fixed-point variables have parentheses with the value 1. In MATLAB, this method can be used to transfer the value of the right-side part to the left-side part without changing the data type.

A code generator also creates a cost function, a objective function, and a top function. During parsing, the arithmetic relationship in the fixed-point data type is stored to the file *_cost*. MAC arithmetic information is shown in Fig. 5.6. This file is called to obtain cost information in objective functions. The cost of the multiplier and the adder is predefined at each function.

A multi-objective function generated by the code generator is shown in Fig. 5.7. There are two objectives in this code: signal distortion and cost. This objective function calls a floating-point code and a fixed-point code with word length for signal distortion information. The cost function is called by this function to obtain cost information. The objective values according to the input word length state are returned

```
function [acc, logrep] = mac_fx(in, numtype, fimathtype)

fipref('LoggingMode','On');
if nargin < 2, numtype = gen_numerictype_init; end
if nargin < 3, fimathtype = gen_fimathtype_init; end

acc = [];
coef = [];
t1 = [];
t2 = [];
if ~isstruct(numtype)
    % For range estimation
    acc =  fi(acc, numtype, fimathtype );
    coef =  fi(coef, numtype, fimathtype );
    t1 =  fi(t1, numtype, fimathtype );
    t2 =  fi(t2, numtype, fimathtype );
else
    acc =  fi(acc, numtype.acc, fimathtype);
    coef =  fi(coef, numtype.coef, fimathtype);
    t1 =  fi(t1, numtype.t1, fimathtype);
    t2 =  fi(t2, numtype.t2, fimathtype);
end

acc(1) = 0  ; % fx
coef(1) = 0.1  ; % fx

for i= 1:length(in)
t1(1) = in(i)  ; % fx
t2(1) = t1 * coef; % fx
acc(1) = acc + t2; %
end

logrep = fxlog(acc,coef,t1,t2);
```

Figure 5.5: Automatically converted fixed-point code for MAC

```
function cost = mac_cost(numtype)
% Automatically generated by fxconv.m
% cost_xxx(In1, In2, Out)
cost = 0;
cost = cost + cost_mul(numtype.t1.WordLength, ...
       numtype.coef.WordLength, numtype.t2.WordLength);
cost = cost + cost_add(numtype.acc.WordLength, ...
       numtype.t2.WordLength, numtype.acc.WordLength);
```

Figure 5.6: Generated MAC cost function

to the objective function, which is called a search engine. Any search algorithm can be used as a search engine; however, multi-objective optimization requires an engine that can handle multi-objectives.

A multi-objective genetic algorithm can handle multi-objective problems. A genetic and evolutionary algorithm toolbox (GEATbx) [104] is employed as a search engine in the case study. This search engine calls a given objective function to search the Pareto optimal set.

The top-level file calls the search engine and calculates the Pareto front. The top file for MAC is shown in Fig. 5.8. This file is also created by a code generator. Maximum and minimum word lengths are copied from a general configuration file, which is called *config.m*. The configuration for the search engine is also copied from a genetic algorithm configuration file, which is called *configgea.m*.

There are three phases in the top file. The first phase is range estimation, which finds the proper range information at each fixed-point variable. The second phase is the searching phase, in which optimum word lengths are searched. The third phase is analysis, in which the Pareto front is drawn and stored.

The above mentioned files can be generated by the commands, which can be used by a batch file, as shown in Fig. 5.9.

```
function ObjVal = mac_obj(Chrom, logrep, in)
% This file was automatically generated by gen_obj

% Length of Chrom
[Nind, Nvar] = size(Chrom);

for i = 1:Nind
    vec_wl = Chrom(i,:);

    % Get numeric type according to range information and default
    wl_vec
    numtype = gen_numerictype(logrep, vec_wl);

    % Set fimath type
    fimathtype = gen_fimathtype;

    % excute with given wordlength
    out_fx = feval('mac_fx', in, numtype, fimathtype);
    out_fl = feval('mac', in);

    % Object value
    rms = (mean((double(out_fx)-out_fl).^2)).^0.5;
    cost = feval('mac_cost', numtype);
    ObjVal(i,:) = [rms cost];
end
```

Figure 5.7: Generated MAC objective function

```
function [acc, pareto_fr] = mac_top(in);

MAX_WL = 15;
MIN_WL = 1;

[acc, logrep] = feval('mac_fx', in);

len_vec = length(logrep);
vec_min_wl = ones(1,len_vec) * MIN_WL;
vec_max_wl = ones(1,len_vec) * MAX_WL;

GeaOpt = tbx3int;
GeaOpt = geaoptset(GeaOpt, 'Selection.RankingMultiobj',15);
GeaOpt = geaoptset(GeaOpt, 'Selection.Pressure', 1.3, ...
                    'Selection.RankingMethod', 1);
GeaOpt = geaoptset(GeaOpt, 'Termination.MaxGenerations', 25, ...
                  'Termination.Method', 1);

objfun = 'mac_obj';
 VLUB = [vec_min_wl; vec_max_wl];

[xnew, GeaOpt] = geamain2(objfun, GeaOpt, VLUB, [], logrep, in);

ObjV = feval(objfun, xnew, logrep, in);

RankOpt = [GeaOpt.Selection.Pressure; GeaOpt.Selection.RankingMethod; ...
            GeaOpt.Selection.RankingMultiobj]';
[FitnV, RankMOV] = ranking(ObjV, RankOpt, 1,
           GeaOpt.System.ObjFunGoals);
plotmop(xnew, ObjV, RankMOV, 'Best individuals at end of optimization');
NonDomInd = xnew(RankMOV==0,:);
NonDomIndObj = ObjV(RankMOV==0,:);

NonDom = [NonDomIndObj NonDomInd];
pareto_fr = sortrows(NonDom)
save mac_result pareto_fr xnew ObjV RankMOV;
```

Figure 5.8: Generated MAC top file

```
> filename = 'mac.m';

% Generate Fixed-point file & Cost function
> fxconv(filename);

% Generate Object file
> gen_obj(filename);

% Generate block top file
> gen_top(filename);
```

Figure 5.9: Main batch file for code generation

The generated top file can be executed with any input data as

```
> in = rand(1,10)
> mac_top(in)
```

Then, the Pareto front is searched and drawn, as in Fig. 3.8 and Fig. 3.7 in Section 3.5.

## 5.5 Summary

This chapter presents techniques for the automating transformation from floating point to fixed point in software. This software provides an environment to transform floating-point programs to fixed-point programs for digital signal processing algorithms. Fixed-point conversion and word length optimization are executed in this environment. A genetic algorithm is employed to handle multi-objective optimization. The automating transformation software is available at

http://www.ece.utexas.edu/~bevans/projects/wordlength/

# Chapter 6

# Summary and Future Work

## 6.1 Summary

This book has examined efficient methods for transformation from floating point to fixed point for implementing digital signal processing algorithms in fixed-point hardware, which offers lower cost and lower power consumption.

Several search methods to find the optimum word length in the transformation are compared. Table 6.1, which summarizes the advantages and disadvantages of the search algorithms, shows that the complete method and genetic methods have more advantages than the other methods. However, the complete search method is impractical and the sequential and preplanned methods require less iteration than the other methods. However, those methods are not able to handle multiple objectives. The CDM search can handle multiple objectives with a weighted sum method. Overall, however, the genetic algorithms with the Pareto ranking approach have more advantages than the other methods.

Word length search algorithms utilizing genetic and evolutionary

Table 6.1: Advantages/disadvantages of word length search algorithms

| Advantages | Disadvantages |
|---|---|
| 1. Global optimum | 1. Local optimum |
| 2. Pareto ranking | 2. Weights in objectives |
| 3. Handle multi-objectives | 3. Single objective |
| 4. Amenable to parallelism | 4. Limited parallelism |
| 5. Low algorithm complexity | 5. High algorithm complexity |
| 6. Fewer iterations | 6. More iterations |
| | 7. Impractical |

| Methods | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Advantages | | | | | | Disadvantages | | | | | | |
| Complete | Y | | | Y | Y | | | | Y | | | Y | Y |
| Exhaustive | | | | | Y | | Y | | Y | Y | | Y | |
| Sequential | | | | | Y | Y | Y | | Y | Y | | | |
| Preplanned | | | | | Y | Y | Y | | Y | Y | | | |
| Genetic(Weighted) | | | Y | Y | | | | Y | | | Y | Y | |
| Proposed methods | | | | | | | | | | | | | |
| CDM | | | Y | | Y | Y | Y | Y | | Y | | | |
| Genetic(Pareto) | | Y | Y | Y | | | | | | | Y | Y | |

algorithms can optimize the signal quality vs. implementation complexity tradeoffs. Alternatively, word length search algorithms utilizing gradient information can provide faster ways to find data word lengths, but they become stalled in local optima.

Based on word length design case studies for a wireless communication demodulator, the speed improvement from adding sensitivity information is by a factor of four. In the same case studies, the local optimum word length searched by the proposed method also yields 30% lower implementation costs.

Word length reduction methods of signed right shift and truncation show a reduction in power consumption. The expected values of the number of gates that switch during multiplication of the inputs for the two methods are mathematically derived. The two methods are applied to a 16-bit radix-4 modified Booth multiplier and a 16-bit Wallace multiplier. The truncation method with 8-bit operands reduces the power consumption by 56% in the Wallace multiplier and 31% in the Booth multiplier. The signed right-shift method shows 25% power reduction in the Booth multiplier, but no power reduction in the Wallace multiplier.

A fully automated method for transforming floating point to fixed point in software is developed. This software provides an environment for transforming floating-point programs into fixed-point programs in digital signal processing algorithms. Automated conversion to fixed-point and word length optimization are executed in the proposed environment. This environment can employ genetic algorithms to handle multi-objective optimization. The automating transformation software from floating point to fixed point is available at

http://www.ece.utexas.edu/~bevans/projects/wordlength/

## 6.2 Future Work

There are many ways in which the work presented in this book could be expanded to develop new search methods and low-power consumption techniques. In this section some of the possibilities will be expanded upon.

### 6.2.1 Advanced Word Length Search Algorithms

**Hybrid word length optimization:** The search methods used in this book have their own strong points and weak points. The various search methods could be used together to compensate for their disadvantages. The genetic algorithm requires considerable running time, but it is not trapped in local optima. Gradient-based search algorithms take less running time, but they do become trapped in local optima. One way to combine the two approaches follows. The first step is use a gradient-based method until it converges to a solution. The second step is to use the trajectory of feasible solutions obtained by the gradient-based search (either with or without the initialization phase solutions included) as the initial generation for the genetic algorithm. The third and final step is to run the genetic algorithm with a high mutation rate to create a genetically diverse population. The insight in taking this approach comes from Fig. 3.9, in which the gradient-based search methods find a solution comparable to genetic algorithms but with two orders of magnitude fewer system simulations. Another way to combine the two approaches is to run a few generations of the genetic algorithm, and then run the gradient-based search in parallel on each solution obtained from the genetic algorithm. This is probably the less promising of the two ways to combine the methods.

**Dynamic word length bounds:** During word length searching process, the bounds of word length are fixed to a lower bound and

an upper bound. In arithmetic operations, the output bound can be changed according to the input word length bound. The upper bound of the word length at the multiplier output is the sum of the upper bounds at the multiplier inputs. Thus, the bound at input or output can be dynamically changed according to the word length bound at input or output. This approach could reduce the running time due to the reduced search space.

**Variable reduction:** The running time in the word length search algorithms is proportional to the number of variables. Given that all variables in a floating-point program are to be converted to fixed-point variables, the optimum word length of each variable is searched. Trivial variables can be removed from the list of word length optimization for greater speed. One variable is sufficient in the delay block, which has input and output variables. The word length output of the adder can be removed from the list since the maximum word length at output is at most one more than the input word length.

**Adaptive step size:** The step size of update directions in gradient-based search algorithms is an integer value. In this book, the value 1 is used for the integer step size. If the step size is larger than 1, word length set could reach a neighbor of the optimum. After that, the step size of value 1 can be used for refinement. This approach can reduce running time.

### 6.2.2 Further Analysis on Search Algorithms

**Analysis in GA with different genetic parameters:** Genetic algorithms (GA) mimic the process of plant and animal evolution. There are many options to realizing a genetic algorithm. The gene can be encoded as a real, integer, or binary number. Since word

length is expressed in an integer number, in this book, an integer encoding method is employed. Thus, any number within bounds is selected during generation. If the binary encoding method is used, the genetic operation could be refined and results would be different. A comparison of the different options in GA would provide valuable information. It is worthy of comparing different options in GA.

**Weighted sum approach in GA:** Multi-objective optimizations have more than one objective. Weighted sum approaches assign weights on each objective to derive a single objective. Pareto ranking approaches assign rank on each candidate by calculating the Pareto rank. In this book, the Pareto ranking approach is employed because Rohling [45] shows many disadvantages in the weighted-sum approach. Research is needed to demonstrate the performance degradation of the weighted-sum approach compared to the Pareto ranking approach in word length optimization.

**Comparison with other algorithms:** Simulated annealing (SA) and genetic algorithms (GA) are two stochastic methods currently in wide used for difficult optmization problems. The GA is used in book to optimize word length because of its simple implementation procedure. Some papers show that a GA can outperform a SA for design in some applications [105, 106]. However, it would be useful for future research to consider simulated annealing algorithm in word length optimization area.

### 6.2.3 Low Power Consumption

**Low power consumption at the system level:** Power consumption can be reduced by decreasing word length in the system, although hardware architectures are given. Chapter 4 shows the word length

reduction techniques and estimates the power savings on multiplier units. Word length reduction can also reduce power consumption within other components such as memory. Powell and Chau show a model for estimating power book in a class of DSP VLSI chips according to word length [107]. Thus, it may be useful for future research to explicitly consider power reduction with word length reduction at the system level.

**Low-power in floating-point hardware:** In some applications, the optimum word length in floating point can reduce power consumption by 66% [108]. This book shows that power consumption in multipliers can be reduced by using word length reduction techniques. The multiplier would be a major power-consuming unit; however, units for addition and normalization in floating-point hardware could also consume considerable power. Thus, the analysis or estimation of power reduction in floating-point hardware could result in low-power consumption by word length reduction techniques.

**Distribution of power consumption** Average power consumption is estimated by using a Xilinx Xpower tool in this book. Small multiplications on large Wallce and Booth multipliers could have different distribution of power consumption. Distribution information of power consumption could be helpful for algorithm developments to reduce power consumption.

### 6.2.4 Electronic Design Automation Software

**Enhanced code generator:** A simple parser that interprets one arithmetic operation at each line has been developed to implement the code generator in this book. Multiple operations do not work on current parsers. Thus, a given floating-point program should have at

most one arithmetic operation for each line. Code generator could be enhanced by adding a decomposition of multiple arithmetic operations.

A parser can search a given commented symbol preceded with floating-point variables in floating-point programs and convert the variables to a fixed-point data type. Search engines search the optimum word length of the converted fixed-point variables; however, designers sometime want to set constraints, such as specific word length, and rounding methods. The code generator used in this book could not handle fixed-point constraints in variables. Enhancements are necessary for generator to handle such constraints.

**Range estimation:** Range can be estimated by analytical or statistical approaches. An analytical approach offers faster results by analyzing relationship of operations in the dataflow. However, the estimated range results from an analytical approach are conservative. Furthermore, the estimated range in a feedback loop could grow infinitely. The statistical approach offers robust range estimation at the expense of long simulation times. Thus, the two approaches could be combined. The statistical approach could be used in a feedback loop part, and an analytical approach could be used in other parts.

**Dataflow approach:** In this book, simulation-based approaches are used. Range information has been monitored at each variable, and propagated quantization errors are measured at the output of systems. However, the range information and the propagated error can be obtained by analyzing dataflow graphs of systems. This approach could reduce time and reduce word length variables.

### 6.2.5 Optimum DSP Algorithms

This work takes an algorithm/system as is and quantifies the word length of variables in terms of signal quality vs. implementation complexity. Thus, the algorithm/system is not changed. The algorithm/system can also be optimized in terms of signal quality vs. implementation complexity. Rearranging the algorithm/system in filters results in a better finite word length effect [78, 109]. The SPIRAL project optimizes the digital signal processing algorithm and automates software and hardware development [110]. SPIRAL is a generator of libraries for fast software implementation of signal processing transforms. These libraries are adapted to the computing platform and can be re-optimized as the hardware is upgraded or replaced [111]. The next level of abstraction is to develop a system that simplifies, rearranges, and expands the algorithm/system in search of a better tradeoff between signal quality and implementation complexity.

### 6.2.6 Area Model

The area model of field programmable gate array (FPGA) [12] is used in this work. Adder, gain, and delay units are modeled in terms of word length. Other units such as transcendental function computation can be modeled by using series of arithmetic units. Transcendental signal generations such as sine and cosine waveforms can be modeled by using difference equations or lookup tables.

In conclusion, while this work has made inroads into automating transformation to fixed-point, there is a wide variety of search methods and low power-consumption alternatives open to further study.

# Appendix A

# Acronyms

| | |
|---|---|
| ADC | : analog-to-digital converter |
| BER | : bit error rate |
| CM | : complexity measure |
| CDM | : complexity-and-distortion measure |
| CDMA | : code division multiplex access |
| CMOS | : complementary metal oxide semiconductor |
| CS | : complete search |
| DSP | : digital signal processing |
| DM | : distortion measure |
| ES | : exhaustive search |
| FER | : frame error rate |
| FFT | : fast Fourier transform |
| FPGA | : field programmable gate array |
| FRIDGE | : fixed-point programming design environment |
| FWL | : fraction wordlength |
| GEA | : genetic and evolutionary algorithm |
| HDS | : hardware design system |
| IC | : integrated circuit |
| IIR | : infinite impulse response |

| | |
|---|---|
| IWL | : integer wordlength |
| LPF | : lowpass filter |
| LS | : least significant |
| MAC | : multiply and accumulate |
| MATCH | : Matlab compiler for heterogeneous computing systems |
| MILP | : mixed integer linear programming |
| MOEA | : multi-objective evolutionary algorithm |
| MS | : most significant |
| MSE | : mean square error |
| OFDM | : orthogonal frequency division multiplexing |
| PS | : preplanned search |
| RMS | : root mean square |
| SNR | : signal-to-noise ratio |
| SPW | : signal processing worksystem |
| SRS | : signed right shift |
| SS | : sequential search |
| TI | : Texas Instruments |
| WL | : wordlength |

# Appendix B

# Notation

The notation used in this book is listed in Table B.1.

Table B.1: Notation used in this book

| Notation | Meaning |
|---|---|
| $\alpha_c$ | complexity weighting factor |
| $\alpha_d$ | distortion weighting factor |
| $c$ | cost function |
| $c_n(\mathbf{w})$ | normalized complexity function |
| $d$ | sum of distance; L1 norm |
| $dw$ | distance between minimum and optimum wordlength |
| $d_n(\mathbf{w})$ | normalized distortion function |
| $f$ | objective function |
| $\nabla$ | gradient of function |
| $p$ | performance function |
| $s$ | integer step size |
| $\mathbf{w} = [w_1, ..., w_n]$ | wordlength vector |
| $w$ | Wordlength |
| $\overline{w}$ | upper bound in $w$ |
| $\underline{w}$ | lower bound in $w$ |
| $w^k$ | wordlength in $k$th iteration |
| $\xi$ | integer update direction |
| $C_{req}$ | complexity constant |
| $D_{req}$ | required distortion |
| $I^n$ | $n$-dimensional integer space |
| $P_{req}$ | required performance |

# Bibliography

[1] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: A fixed-point design and simulation environment," in *Proc. IEEE Design Automation and Test in Europe*, France, France, Feb. 1998, pp. 429–435.

[2] W. Sung and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[3] S. Kim, K. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Trans. Circuits Syst.*, vol. 45, no. 11, pp. 1455–1464, Nov. 1998.

[4] K. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: An optimizing floating-point to integer c program converter for fixed-point digital signal processors," *IEEE Trans. Circuits Syst.*, vol. 47, pp. 840–848, Sept. 2000.

[5] *Hardware Design System*, CoWare. [Online]. Available: http://www.coware.com

[6] P. Banerjee, N. Shenoy, A. Choudhary, S. Hauck, C. Bachmann, M. Haldar, P. Joisha, A. Jones, A. Kanhare, A. Nayak, S. Periy-

acheri, M. Walkden, and D. Zaretsky, "A MATLAB compiler for distributed, heterogeneous, reconfigurable computing systems," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2000, pp. 39–48.

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

[8] H.Yamashita, H. Yasnura, F. Eko, and Y. Cao, "Variable size analysis and validation of computation quality," in *Proc. Int. High-Level Design Validation and Test Workshop*, 2000, pp. 95–100.

[9] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGA's," in *Proc. Design Automation and Test in Eur.*, Munich, Germany, Mar. 2001, pp. 722–728.

[10] K. Han, I. Eo, K. Kim, and H. Cho, "Bit constraint parameter decision method for cdma digital demodulator," in *Proc. CDMA Int. Conf. and Exhibition*, vol. 2, Seoul, Korea, Nov. 2000, pp. 583–586.

[11] M. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proc. IEEE Int. Sym. on Circuits and Systems*, Phoenix-Scottsdale, Ariz, USA, May 2002, pp. 612–615.

[12] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans.*

*Computer-Aided Design*, vol. 22, no. 10, pp. 1432–1442, Oct. 2003.

[13] C. Shi and R. W. Brodersen, "Automated fixed-point data-type optimization tool for signal processing and communication systems," in *Proc. Design Automation Conference*, San Diego, CA, June 2004, pp. 478–483.

[14] *MOSEK ApS optimization software*, MOSEK ApS, Copenhagen, Denmark. [Online]. Available: http://www.mosek.com

[15] K. Han and B. L. Evans, "Wordlength optimization with complexity-and-distortion measure and its applications to broadband wireless demodulator design," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 5, Montreal, Quebec, Canada, May 2004, pp. 37–40.

[16] K. Han and B. Evans, "Optimum wordlength search using a complexity-and-distortion measure," *EURASIP Journal on Applied Signal Processing*, vol. 2006, no. 5, pp. 103–116, 2006.

[17] *CoCentric Fixed-Point Designer*, Synopsys. [Online]. Available: http://www.sysnopsis.com

[18] *SystemC 2.0 User's Guide*, 2002. [Online]. Available: http://www.systemc.org

[19] S. Kim and W. Sung, "A floating-point to fixed-point assembly program translator for the TMS 320C25," *IEEE Trans. Circuits Syst.*, vol. 41, no. 11, pp. 730–739, 1994.

[20] S. A. Wadekar and A. Parker, "Accuracy sensitive sensitive word-length selection for algorithm optimization," in *Proc. Int. Conf. Computer Design*, Austin, TX, USA, Oct. 1998, pp. 54–61.

[21] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," in *Proc. SIGPLAN Program. Lang. Design Implementation*, Vancouver, BC, Canada, June 2000, pp. 108–120.

[22] K. Kum and W. Sung, "Combined word-length optimization and high-level sysnthesis of digital signal processing systems," *IEEE Trans. Computer-Aided Design*, vol. 20, no. 8, pp. 921–930, Aug. 2001.

[23] H. Choi and W. P. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proc. IEEE Workshop on VLSI Signal Processing*, vol. VII, Calif, USA, Oct. 1994, pp. 198–207.

[24] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proc. IEEE Int. Sym. on Circuits and Systems*, vol. 4, Sydney, NSW, Australia, May 2001, pp. 290–293.

[25] M. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *Proc. IEEE Int. Sym. on Circuits and Systems*, vol. 5, Sydney, NSW, Australia, May 2001, pp. 53–56.

[26] K. H. Rosen, *Handbook of Discrete and Combinatorial Mathematics*. Boca Raton, Fla, USA: CRC Press, 2000.

[27] G. S. G. Beveridge and R. S. Schechter, *Optimization: Theory and Practice*. New York, NY, USA: McGraw-Hill, 1970.

[28] J. A. Wepman, "Analog-to-digital converters and their applications in radio receivers," *IEEE Comm. Magazine*, vol. 33, no. 5, pp. 39–45, 1995.

[29] S. Nahm, K. Han, and W. Sung, "A CORDIC-based digital quadrature mixer: Comparison with ROM-based architecture," in *Proc. IEEE Int. Sym. on Circuits and Systems*, vol. 4, Monterey, CA, USA, May-June 1998, pp. 385–388.

[30] J. S. Wu, M. L. Liu, H. P. Ma, and T. D. Chiueh, "A 2.6V, 44 MHz all-digital QPSK direct-sequence spread-spectrum transeiver IC [wireless LANs]," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 10, pp. 1499–1510, 1997.

[31] C. Darwin, *The Origin of Species by Means of Natural Selection.* Blatimore, Maryland: Penguin Books, 1859.

[32] D. Goldberg, "Genetic and evolutionary algorithms come of age," *Communications of the ACM*, vol. 37, no. 3, pp. 113–119, Mar. 1994.

[33] J. Holland, *Adaptation in Natural and Artifical Systems: An Introductory Analysis with Application to Biology, Control, and Artifical Intelligence.* Ann Arbor, MI: University of Michigan Press, 1975.

[34] D.E.Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

[35] S. Kirkpatrick, C. G. Jr., and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[36] H. Schwefel, *Numberical Optimization of Computer Models.* Chichester: John Wiley, 1981.

[37] D. Fogel, *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling.* Needham Heights, MA: Ginn Press, 1991.

[38] D. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: John Wiley and Sons, 1966.

[39] K. Tang, K. Man, and C. Y. . Chan, "Fuzzy control of water pressure using genetic algorithm," in *Proc. IFAC Workshop on Safety, Reliability and Applications of Emerging Intelligent Control Technologies*, 1994, pp. 15–20.

[40] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, no. 2, pp. 26–33, 1991.

[41] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 39–53, 1994.

[42] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[43] S. Hung and H. Adeli, "A parallel genetic/neural network learning algorithm for MIMD shared memory machines," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 900–909, 1994.

[44] K. Tang, K. Man, and C.Y.Chan, "Genetic structure for NN topology and weight optimization," in *Proc. IEE/IEEE Int. Conf. GAs in Engineering Systems: Innovations and Application*, Sept. 1995, pp. 12–14.

[45] G. Rohling, "Multiple objective evolutionary algorithms for independent, computationally expensive objective evaluations," Ph.D. dissertation, School of Electrical and Computer Engineering, Georgia Institute of Technology, November 2004.

[46] C. A. C. Coello, "Guest editorial: Special issue on evolutionary multiobjective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 7, pp. 97–98, April 2003.

[47] F. Y. Edgeworth, *Mathematical Physics*, London, U.K., 1881.

[48] V. Pareto, *Cours D'Economie Politique*, Lausanne, Switzerland, 1896.

[49] C.M.Fonseca and P.J.Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. Int. Conf. Genetic Algorithm*, July 1993, pp. 416–423.

[50] D.M.Kodek, "Design of optimal finite wordlength FIR digital filter using linear programming techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 3, pp. 304–308, June 1980.

[51] N. I. Cho and S. U. Lee, "Optimal design of finite precision FIR filters using linear programming with reduced constraints," *IEEE Trans. Signal Process.*, vol. 46, no. 1, pp. 195–199, Jan 1998.

[52] Y. C. Lim, S. R. Parker, and A. G. Constantinides, "Finite word length FIR filter design using integer programming over a discrete coefficient design,space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-30, no. 4, pp. 661–664, Aug 1982.

[53] D. J. Xu and M. L. Daley, "Design of optimal digital filter using a parallel genetic algorithm," *IEEE Trans. Circuits and Systems*, vol. 42, no. 10, pp. 673–675, Oct 1995.

[54] K. Tang, K. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, pp. 22–37, November 1996.

[55] S.C.NG, S. Leung, C.Y.Chung, A.Luk, and W.H.Lau, "The genetic search approach," *IEEE Signal Processing Magazine*, pp. 38–46, Nov. 1996.

[56] D. M. Etter, M. J. Hicks, and K. H. Cho, "Recursive adaptive filter design using an adaptive genetic algorithm," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1982, pp. 635–638.

[57] D. Suckley, "Genetic algorithm in the design of FIR filters," *IEE Proc. Circuits, Devices and Systems*, vol. 138, pp. 234–238, April 1991.

[58] K.-S. Tang, K.-F. Man, S. Kwong, and Z.-F. Liu, "Design and optimization of IIR filter structure using hierarchical genetic algorithms," *IEEE Trans. Industrial Electronics*, vol. 45, no. 3, pp. 481–487, June 1998.

[59] M. Leban and J. F. Tasic, "Word-length optimization of LMS adaptive FIR filters," in *Proc. IEEE Mediterranean Electrotechnical Conference*, May 2000, pp. 774–777.

[60] M. Haseyama and D. Matsuura, "A filter coefficient quantization method with genetic algorithm, including simulated annealing," *IEEE Signal Processing Letters*, vol. 13, pp. 189–192, April 2006.

[61] R. Cemes and D. Ait-Boudaoud, "Genetic approach to design of multiplierless FIR filters," *IEE Electronics Letters*, vol. 29, no. 24, pp. 2090–2091, 1993.

[62] G. Wade, A. Roberts, and G. Williams, "Multiplier-less FIR filter design using a genetic algorithm," *Proc. IEE Vision, Image and Signal Processing*, vol. 141, pp. 175–180, 1994.

[63] Y. Yu and Y. Lim, "Genetic algorithm approach for the optimization of multiplierless sub-filters generated by the frequency-response masking technique," in *Proc. Int. Conf. Electronics, Circuits and Systems*, 2002, pp. 1163–1166.

[64] F. Ashrafzadeh and B. Nowrouzian, "Crossover and mutation in genetic algorithms employing canonical signed-digit number system," in *Proc. IEEE Midwest Symp. Circuits and Systems*, 1997, pp. 702–705.

[65] A. Fuller, B. Nowrouzian, and F. Ashrafzadeh, "Optimization of FIR digital filters over the canonical signed-digit coefficient space using genetic algorithms," in *Proc. IEEE Midwest Symp. Circuits and Systems*, 1998, pp. 456–459.

[66] A. Lee, M. Ahmadi, G. Jullien, W.C.Miller, and R.S.Lashkari, "Digital filter design using genetic algorithm," in *Proc. IEEE Symp. Advances in Digital Filtering and Signal Processing*, 1998, pp. 34–38.

[67] A. Fuller and B. Nowrouzian, "A novel technique for optimization over the canonical signed-digit number space using genetic algorithms," in *Proc. Int. Symp. Circuits and Systems*, 2001, pp. 745–748.

[68] N. Sulaiman and T. Arslan, "A genetic algorithm for the optimisation of a reconfigurable pipelined FFT processor," in *Proc. Evolvalbe Hardware*, June 2004, pp. 104–108.

[69] ——, "A multi-objective genetic algorithm for on-chip real-time optimisation of word length and power consumption in a pipelined FFT processor targeting a MC-CDMA receiver," in *Proc. Evolvalbe Hardware*, June 2005, pp. 154–159.

[70] P. Fiore and L. Lee, "Closed-form and real-time wordlength adaptation," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 4, Phoenix, Ariz, USA, Mar. 1999, pp. 1897–1900.

[71] H. Bolcskei, A. J. Paulraj, K. V. S. Hari, R. U. Nabar, and W. W. Lu, "Fixed broadband wireless access: state of the art, challenges, and future directions," *IEEE Comm. Magazine*, vol. 39, no. 1, pp. 100–108, 2001.

[72] V. Erceg, K. V. S. Hari, M. S. Smith, K. P. Sheikh, C. Tappenden, J. M. Costa, D. S. Baum, and C. Bushue, "Channel models for fixed wireless applications," in *IEEE 802.16. proposal 802.16.3c-01/29*, 2001.

[73] D. S. Baum, "Simulating the SUI channel models," Information Systems Laboratory, Stanford University, Stanford, Calif, USA, Tech. Rep., 2001.

[74] B. Shim and N. R. Shanbhag, "Complexity analysis of multicarrier and single-carrier systems for very high-speed digital subscriber line," *IEEE Trans. Signal Processing*, vol. 51, no. 1, pp. 282–292, 2003.

[75] G. Constantinides, "High level synthesis and word length optimization of digital signal processing systems," Ph.D. dissertation, Department of Electronic and Electrical Engineering, University College London, London, U.K., Sept. 2001.

[76] R. Fletcher, *Practical Methods of Optimization, Vol. 2: Constrained Optimization*. New York, NY, USA: John Wiley and Sons, 1981.

[77] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. New York, NY, USA: John Wiley and Sons, 1996.

[78] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York, NY: John Wiley and Sons, 1999.

[79] M. T. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. on VLSI Systems*, vol. 5, pp. 123–135, Mar. 1997.

[80] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[81] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 273–285, June 2000.

[82] H. Lee, "A power-aware scalable pipelined Booth multiplier," in *Proc. IEEE International Systems-On-Chip Conference*, Sept. 2004, pp. 123–126.

[83] K. Han, B. L. Evans, and E. Swartzlander, "Data wordlength reduction for low-power signal processing software," in *Proc. IEEE Int. Workshop on Signal Processing Systems*, Austin, TX, USA, Oct. 2004, pp. 343–348.

[84] O. T.-C. Chen, S. Wang, and Y.-W. Wu, "Minimization of switching activities of partial products for designing low-power

multipliers," *IEEE Trans. on VLSI Systems*, vol. 11, pp. 418–433, June 2003.

[85] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, San Jose, CA, Nov. 1994, pp. 429–435.

[86] A. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, pp. 236–240, 1951.

[87] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 12–31, Jan. 1995.

[88] A. T. Erdogan and T. Arslan, "Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors," *IEE Electronics Letters*, vol. 32, pp. 1959–1960, Oct. 1996.

[89] B. Parhami, *Computer Arithmetic Algorithm and Hardware Designs*. Oxford University Press, 2000.

[90] S. Kim and M. Papaefthymiou, "Reconfigurable low-energy multiplier for multimedia system design," in *Proc. IEEE Workshop on VLSI*, Apr. 2000, pp. 129–134.

[91] G. Grimmett and D. Stirzaker, *Probability and Random Processes*. Oxford University Press, 2001.

[92] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. on Computers*, vol. 13, pp. 14–17, 1964.

[93] K. C. Bickerstaff, E.E. Swartzlander, Jr., and M. J. Schulte, "Analysis of column compression multipliers," in *Proc. IEEE Symposium on Computer Arithmetic*, June 2001, pp. 33–39.

[94] *Spartan-3 FPGA Family: Complete Data Sheet*, Xilinx, Jan. 2005. [Online]. Available: http://www.xilinx.com/bvdocs/publications/ds099.pdf

[95] W. Cammack and M. Paley, "Fixpt: a c++ method for development of fixed point digital signal processing algorithms," in *Proc. Hawaii Int. Conf. on System Sciences*, Wailea HI, Jan. 1994, pp. 87–95.

[96] K. Kum, J. Kang, and W. Sung, "A floating-point to fixed-point C converter for fixed-point digital signal processors," in *Proc. SUIF Compiler Workshop*, Aug. 1997.

[97] *Fixed-Point Toolbox User's Guide*, The MathWorks, Inc., Natick, MA, USA. [Online]. Available: http://www.mathworks.com

[98] *Simulink Fixed Point*, The MathWorks, Inc., Natick, MA, USA. [Online]. Available: http://www.mathworks.com

[99] *AccelChip DSP Synthesis*, AccelChip, Inc. [Online]. Available: http://www.accelchip.com

[100] P. Banerjee, D. Bagchi, M. Haldar, A. Nayak, V.Kim, and R. Uribe, "Automatic conversion of floating point MATLAB programs into fixed point FPGA based hardware design," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2003, pp. 263–264.

[101] M.Coors, H. Keding, O.Luthje, and H. Meyr, "Integer code generation for the TI TMS320C62X," in *Proc. IEEE Int. Conf. on*

*Acoustics, Speech, and Signal Proc.*, vol. 2, May 2001, pp. 1133–1136.

[102] D. Menard, D. Chillet, F. Charot, and O. Sentieys, "Automatic floating-point to fixed-point conversion for DSP code generation," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems*, Grenoble, France, 2002, pp. 270–276.

[103] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Optimum wordlength allocation," in *Proc. of the 10th Annual IEEE Sym. on Field-Programmable Custom Computing Machines*, Apr. 2002, pp. 219–228.

[104] *Genetic and Evolutionary Algorithm Toolbox for Use with Matlab*. [Online]. Available: http://www.geatbx.com

[105] F. Ares, S. Rengarajan, E. Villaneuva, E. Skochinski, and E. Moreno, "Application of genetic algorithms and simulated annealing technique in optimising the aperture distributions of antenna array patterns," *Electronics Letters*, vol. 32, no. 3, pp. 148–149, 1996.

[106] D. R. Thompson and G. L. Bilbro, "Comparison of a genetic algorithm with a simulated annealing algorithm for the design of an ATM network," *IEEE Communications Letters*, vol. 4, no. 8, pp. 267–269, 2000.

[107] S. R. Powell and P. Chau, "A model for estimating power dissipation in a class of DSP VLSI chips," *IEEE Trans. Circuits and Systems*, vol. 38, no. 6, pp. 646–650, June 1991.

[108] F.Fang, T. Chen, and R. Rutenbar, "Floating-point bit-width optimization for low-power signal processing applications," in

*Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol. 3, May 2002, pp. 3208–3211.

[109] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[110] M. Püschel, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gačić, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, "SPIRAL: Code generation for DSP transforms," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 232–275, 2005.

[111] M. Püschel, B. Singer, J. Xiong, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, and R. W. Johnson, "SPIRAL: A generator for platform-adapted libraries of signal processing algorithms," *Int'l Journal of High Performance Computing Applications*, vol. 18, no. 1, pp. 21–45, 2004.

# Index