Chapter 1

SYNTHESIS OF DSP ALGORITHMS FROM INFINITE PRECISION SPECIFICATIONS

This is an Article Subtitle

Christos-Savvas Bouganis
Dept. of Electrical and Electronic Engineering
Imperial College London
christos-savvas.bouganis@imperial.ac.uk

George A. Constantinides
Dept. of Electrical and Electronic Engineering
Imperial College London
george.constantinides@ieee.org

Abstract Digital signal processing (DSP) technology is the core of many modern application areas. Computer vision, data compression, speech recognition and synthesis, digital audio and cameras, are a few of the many fields where DSP technology is essential.

Although Moore's law continues to hold in the semiconductor industry, the computational demands of modern DSP algorithms outstrip the available computational power of modern microprocessors. This necessitates the use of custom hardware implementations for DSP algorithms. Design of these implementations is a time consuming and complex process. This chapter focuses on techniques that aim to partially automate this task.

The main thesis of this chapter is that domain-specific knowledge for DSP allows the specification of behaviour at infinite precision, adding an additional 'axis' of arithmetic accuracy to the typical design space of power consumption, area, and speed. We focus on two techniques, one general and one specific, for optimizing DSP designs.

Keywords: DSP, synthesis, infinite precision, 2D filters.

1. Introduction

The aim of this chapter is to provide some insight into the process of synthesising digital signal processing circuits from high-level specifications. As a result, the material in this chapter relies on some fundamental concepts both from signal processing and from hardware design. Before delving into the details of design automation for DSP systems, we provide the reader with a brief summary of the necessary prerequisites. Much further detail can be found in the books by Mitra [Mitra, 2006] and Wakerly [Wakerly, 2006], respectively.

Digital Signal Processing refers to the processing of signals using digital electronics, for example to extract, suppress, or highlight certain signal properties. A signal can be thought of as a 'wire' or variable, through which information is passed or streamed. A signal can have one or many dimensions; a signal that represents audio information is a one-dimensional signal, whereas a signal that represents video information is a two dimensional signal.

A discrete-time signal x is usually represented by using the notation $x[n]$. The value $x[n]$ of the signal x refers to the value of the corresponding continues-time signal at sampling time nT, where T denotes the sampling period.

The z transform is one of the main tools that is used for the analysis and processing of digital signals. For a signal $x[n]$, its z transform is given by (1.1).

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{1.1}$$

The chapter will mainly focus on Linear Time Invariant (LTI) systems, thus it is worthwhile to see how the z transform is useful for such systems. The output signal $y[n]$ of an LTI system with impulse response $h[n]$ and input signal $x[n]$ is given by the convolution of the input signal and the impulse response (1.2).

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{1.2}$$

Using the z transform, (1.2) can be written as (1.3), where $Y(z)$, $H(z)$, and $X(z)$ are the z transforms of the $y[n]$, $h[n]$ and $x[n]$ signals, respectively.

$$Y(z) = H(z)X(z) \tag{1.3}$$

Table 1.1. Degrees of a node in a computational graph

TYPE	INDEGREE	OUTDEGREE
INPORT	0	1
OUTPORT	1	0
ADD	2	1
DELAY	1	1
GAIN	1	1
FORK	1	≥ 2

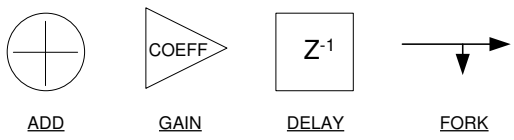


Figure 1.1. Type of nodes

Thus convolution in the time domain is equivalent to multiplication in the z domain, a basic result used throughout this chapter.

Fixed point representation and computational graphs

In this chapter, the representation of DSP algorithm is the *computational graph*, a specialization of a data flow graph graph of Lee *et al.* [Lee and Messerschmitt, 1987]. In a computational graph each element in the set V corresponds to an atomic computation or input/output port, and $S \subseteq V \times V$ is the set of directed edges representing the data flow. An element of S is referred as a *signal*.

In the case of an LTI system, the computations in a computational graph can only be one of several types: input port, output port, gain (constant coefficient multiplier), addition, unit-sample delay and a fork (branching of data). These computations should satisfy the constrains of indegree and outdegree given in Table 1.1. A visualization of the different node types is shown in Figure 1.1. An example of a computational graph is shown in Figure 1.2.

In digital hardware, the traditional number representations are floating-point and fixed-point. The former representation is typically used in the general purpose processing units, whereas the latter is commonly adopted by the DSP processors due to the low area requirements and high throughput that can be achieved. The representation, introduced in [Constantinides et al., 2002], that is used in this is the multiple word-

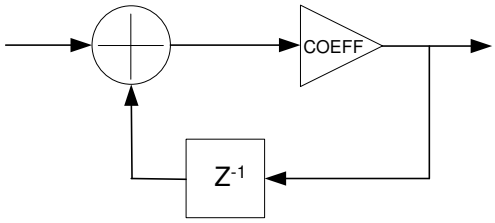


Figure 1.2. An example of a computation graph

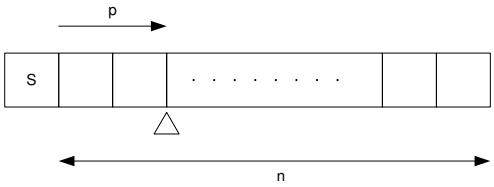


Figure 1.3. Word-length representation [Constantinides et al., 2002]

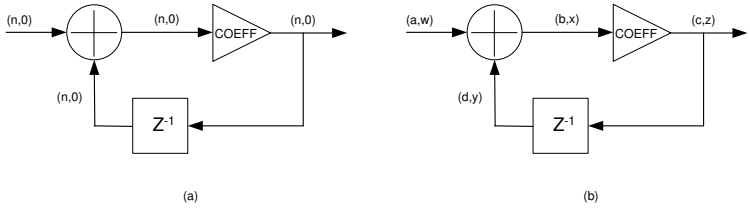


Figure 1.4. An example of a system using (a) fixed-point representation, and (b) multiple word-length representation.

length representation, an extension of fixed-point for parallel hardware design.

According to this scheme, each signal $j \in S$ in a computation graph $G(V, S)$ has two parameters n_j and p_j. The first parameter, n_j, specifies the number of bits in the representation of the signal (excluding the sign bit), while the second parameter, p_j, represents the displacement of the binary point from the sign bit. Figure 1.3 shows an example of a signal j.

Figure 1.4 illustrates the same system using (a) a fixed-point representation and (b) a multiple word-length representation. In the first case all the signals use the same number of bits ($\forall i, j \in S.n_i = n_j$) and scale ($\forall i, j \in S.p_i = p_j$). In the multiple word-length example, each signal can use a different representation.

Peak Value Estimation

In order to make efficient use of the available resources, the scaling of each signal should be selected appropriately. The chosen representation should not be over-wasteful, by allowing the representation of values that are impossible to ever occur, but at the same time should not allow overflow errors to regularly occur.

If the absolute maximum value P of a signal is known, a scaling of $p = \lfloor \log_2 P \rfloor + 1$ should be used since a power of two multiplication is cost free in bit-parallel hardware.

There are three main approaches that can be applied in order to estimate the maximum value of the signals in a system. These are analytic peak estimation, range propagation, and simulation based approaches. We shall briefly elaborate on each of these schemes, below.

Analytic Peak Estimation. In the case of an LTI system, the peak value of each signal in the system can be estimated analytically. This is achieved by calculating the transfer function from each primary input to each signal. In the case where the system in non-recursive (*i.e* the computational graph does not contain cycles), the calculation of the transfer function is a simple task, leading to polynomials in z^{-1}.

In the case of a recursive system, the calculation of the transfer function is more complex. The set of nodes whose outputs correspond to a system *state* should be identified. In this context, this set of nodes is the one which if removed from the computation graph it will break all the cycles. After breaking up the feedback loops, the transfer function matrix $S(z)$ from each input signal to the output of each of these state nodes is expressed as a function of the transfer function matrix $A(z)$ between state nodes and state nodes, and the transfer function matrix $B(z)$ between the primary inputs and state nodes as in (1.4).

$$S(z) = A(z)S(z) + B(z) \tag{1.4}$$

$$H(z) = C(z)S(z) + D(z) \tag{1.5}$$

The transfer function matrix $H(z)$ is calculated as in (1.5), where $C(z)$ is the transfer function matrix between the state-nodes outputs and the outputs of all nodes, where $D(z)$ is the transfer function matrix between the primary inputs and the outputs of all nodes. For a more detailed description for the calculation of $H(z)$, the reader is referred to [Constantinides et al., 2004].

Given the transfer function matrix $H(z)$, with associated impulse response $h[t] = Z^{-1}\{H(z)\}$, the worst-case peak value P_j of any signal

j can be found by maximizing the convolution sum (1.6) [Mitra, 2006], where $x_i[t]$ is the value of input i at time index t. Solving this maximization problem leads to (1.7), where M_i is the maximum absolute value of the signal $x_i[t]$.

$$P_j = \pm \sum_{i \in V_I} \max_{x_i[t']} \left(\sum_{t=0}^{N_{ij}-1} x_i[t'-t] h_{ij}[t] \right) \tag{1.6}$$

$$P_j = \sum M_i \sum_{t=0}^{\infty} |h_{ij}[t]| \tag{1.7}$$

Data Range Propagation. In the case where the system under consideration is not linear, or is not time-invariant, one mechanism to estimate the maximum value of the signals is by considering the propagation of data ranges. It should be noted that this approach is only applicable in the case of non-recursive systems.

Under this mechanism, the data ranges are propagated through the operations of the system. This approach has been formally stated in terms of interval analysis [Benedetti and Prasanna, 2000]. It should be noted that this mechanism can lead to pessimistic results in the case where the system includes data branches which later reconverge.

Simulation driven analysis. Another approach is to use simulation to estimate the peak value of each signal in a system. In this case, the system is simulated using as representative input data as possible and the maximum values for each signal are recorded. After the end of the simulation, the recorded peak values are multiplied by a user-supplied 'safety-factor' $k > 1$, in order to accommodate for values of the signal that did not occur during the simulation, but may occur in practice leading to overflow. More complex forms of the safety-factor have also been considered by researchers in the field [Kim et al., 1998].

This approach is more suitable for non-linear or time-varying systems where the propagation range methodology provides overly pessimistic results (such as recursive systems). The dependence of the final result on the input data is the main drawback of this approach.

Summarizing, there are three methodologies that can be applied for determining the peak value of the signals in the system. In the case where the system is LTI, the analytic method provides a tight bound on the peak value estimation of the signals. In the case where the system is nonlinear or time-variant, and non-recursive the propagation range method can be applied to provide an upper bound on the peak value

of the signals. In the general case, simulation methods can always be applied, which provide a lower bound on the estimation of the peak value of the signals.

2. Word-length Optimization

The previous section has focused on the scale determination of each signal in the system. This section concentrates on the estimation of the remaining parameter: the word-length of the signals.

In order to optimize the word-length of each signal in the system, a model that determines the error at the outputs of the system for a given set of word-length and scaling parameters is required. We call this problem error estimation. Given an error estimation model, the problem of word-length optimization reduces to a problem of utilizing the available resources, the area of the design in our case, satisfying a set of constraints for the outputs of the system.

Error Estimation Model

The quality of a fixed-point algorithm implementation is usually measured using the Signal-to-noise ratio (SNR). The *fixed-point error* is calculated by subtracting the output sequence of the system under a fixed-point implementation from the output sequence of the same system under an infinite precision implementation. The ratio of the output power resulting from an infinite precision implementation to the fixed-point error power defines the signal-to-noise ratio. In this chapter we assume that the *signal* powers at the outputs are fixed, since they are determined only by the input signal statistics and the computation graph. Thus, it is sufficient to concentrate on noise power estimation.

Word-length propagation

In order to predict the quantization effects in the system, we need to propagate the word-length and scaling parameters from the inputs of each atomic operation to its outputs. Table 1.2 summarizes the word-length and scaling propagation rules for the different atomic operations. The superscript q denotes the signal before the quantization take place, *i.e.* without loss of information.

In a multiple word-length implementation, it is important to ensure that sub-optimal implementations are avoided. For example, consider the case of a GAIN node where the input signal j_1 is (n_{j_1}, p_{j_1}), and the coefficient has format (n, p). If the output signal j_2 has $n_{j_2} > n_{j_1} + n$, then this choice is suboptimal since at most $n_{j_1} + n$ bits are required for representing the results in full precision. Ensuring that these cases do

Table 1.2. Word-length and scaling propagation

TYPE	Propagation rules
GAIN	For input (n_a, p_a) and coefficient (n_b, p_b): $p_j = p_a + p_b$ $n_j^q = n_a + n_b$
ADD	For inputs (n_a, p_a) and (n_b, p_b): $p_j = \max(p_a, p_b) + 1$ $n_j^q = \max(n_a, n_b + p_a - p_b) - \min(0, p_a - p_b) + 1$ (for $n_a > p_a - p_b$ or $n_b > p_b - p_a$)
DELAY or FORK	For input (n_a, p_a): $p_j = p_a$ $n_j^q = n_a$

not arise, is referred to as 'conditioning' of the annotated computation graph [Constantinides et al., 2004]. During the optimization process, ill-conditioned computation graphs may rise, which should be transformed to well-conditioned ones [Constantinides et al., 2004].

Linear Time Invariant Systems

We will first address the error estimation model for linear time-invariant systems. In this case, we can derive analytic models of how the noise due to truncation of a signal is propagated through the computation graph to its outputs.

Noise model. A common approach in DSP is that a truncation or roundoff operation will be performed after a multiplication or a multiplication-accumulation operation. This corresponds to the case of a processor where the result of an n-bit signal multiplied by an n-bit signal, which is a $2n$-bit signal, should be truncated to n-bits in order to fit in an n-bit register. Thus, for a two's complement representation, the error that is introduced to the system assuming $p = 0$ ranges between 0 and $2^{-2n} - 2^n \approx -2^n$. As long as the $2n$-bit result has sufficient dynamic range, it has been observed that the values in that range are equally likely to happen [Oppenheim and Schafer, 1972] [Liu, 1971]. This leads to the formulation of a uniform distribution model of the noise with variance $\sigma^2 = \frac{1}{12}2^{-2n}$, when $p = 0$ [Liu, 1971]. Moreover, it has been observed that the spectrum of the noise tends to be white, due to the fact the the truncation occurs in the low significant bits of the signals, and that roundoff errors that occur at different parts of the system are uncorrelated.

However, in our case the above noise model cannot be applied. Consider the truncation of a signal (n_1, p) to a signal (n_2, p). In the case where $n_1 \approx n_2$, the model will suffer in accuracy due to the discretization of the error probability density function. Also, the lower bound of the error can not be simplified as before since $2^{-n_2} - 2^{-n_1} \approx -2^{-n_1}$ no longer holds. Moreover, in the case of a branching node where the output signals can be truncated to different lengths, the preceding model does not consider the different error signals.

The solution to the above problems comes by considering a discrete probability distribution for the injected signals [Constantinides, 2003]. In the case of a truncation of a signal (n_1, p) to a signal (n_2, p), the error that is inserted to the system is bounded by (1.8).

$$-2^p(2^{-n_2} - 2^{-n_1}) \leq e[t] \leq 0 \tag{1.8}$$

Assuming, as before, that $e[t]$ takes values from the above range with equal probability, the expectation of $e[t]$, $E\{e[t]\}$ and its variance σ_e^2 are given by (1.9) and (1.10) respectively.

$$\begin{aligned} E\{e[t]\} &= -\frac{1}{2^{n_1-n_2}} \sum_{i=0}^{2^{n_1-n_2}-1} i \cdot 2^{p-n_1} \\ &= -2^{p-1}(2^{-n_2} - 2^{-n_1}) \end{aligned} \tag{1.9}$$

$$\begin{aligned} \sigma_e^2 &= \frac{1}{2^{n_1-n_2}} \sum_{i=0}^{2^{n_1-n_2}-1} (i \cdot 2^{p-n_1})^2 - E\{e[t]\} \\ &= \frac{1}{12} 2^{2p}(2^{2n_2} - 2^{2n_1}) \end{aligned} \tag{1.10}$$

Noise propagation. By considering a computation graph, the truncation of a signal j from (n_j, p_j) to (n_j^q, p_j) in the graph injects a noise in the system according to (1.10). The application of this model is straight forward apart from the case of a fork. Figure 1.5 shows two different approaches for modelling the truncation of the signals. In the first approach, noise is injected at each output of the fork, leading to correlated injected noise. In the second approach, there is cascaded noise injection, leading to a less correlated noise injection, which is in line with the assumption about the noise propagation model.

Given an annotated graph, a set $F = \{(\sigma_p^2, R_p)\}$ of injected input variances, σ_p^2, and their transfer functions to the primary outputs, $R_p(z)$, can be constructed. From this set, and under the assumption that the

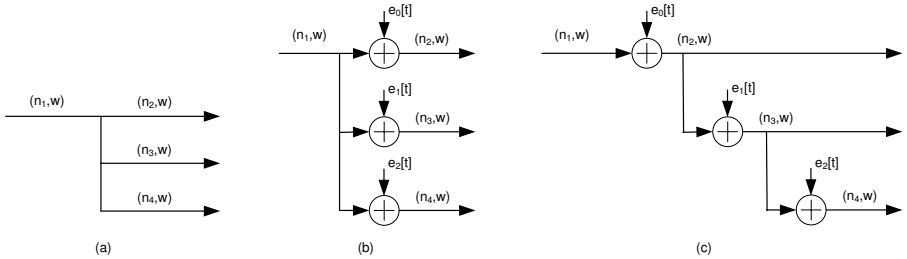


Figure 1.5. Approaches for modelling post-FORK truncation.

noise sources have white spectrum and are uncorrelated, L_2 scaling [Mitra, 2006] can be used to estimate the power of the injected noise at each output k of the system according to (1.11). The L_2 scaling of a transfer function is given in (1.12), where $Z^{-1}\{\cdot\}$ denotes the inverse z-transform.

$$E_k = \sum_{(\sigma_p^2, R_p) \in F} \sigma_p^2 L_2^2\{R_{pk}\} \tag{1.11}$$

$$L_2\{H(z)\} = \left(\sum_{n=0}^{\infty} |Z^{-1}\{H(z)\}[n]|^2 \right)^{\frac{1}{2}} \tag{1.12}$$

Extension to non-linear systems

The same methodology can be applied in an approximate way for non-linear systems, by linearizing the system around some operating point. In the DSP domain, the most common occurrence of non-linearities is from the introduction of general multipliers, which can be found, for example, in adaptive filter systems. A way to approach the problem is by approximating these non-linearities by using the first terms of a Taylor expansion, an idea that is derived from small-signal analysis usually found in analog electronics [Sedra and Smith, 1991].

Let us consider an n-input function $Y[t] = f(X_1[t], X_2[t], \ldots, X_n[t])$, where t is the time index. If we consider a small perturbation x_i in variable X_i, then the perturbation $y[t]$ on variable $Y[t]$ can be approximated as $y[t] \approx x_1[t]\frac{\partial f}{\partial X_1} + x_2[t]\frac{\partial f}{\partial X_2} + \ldots + x_n[t]\frac{\partial f}{\partial X_n}$.

This approximation is linear in each x_i, but the coefficients may vary with time since $\frac{\partial f}{\partial X_i}$ is a function of $X_1, X_2, \ldots, X_n$. Using the above approximation we have managed to transform a non-linear time-invariant system into a linear time-varying system. This linearity allows us to pre-

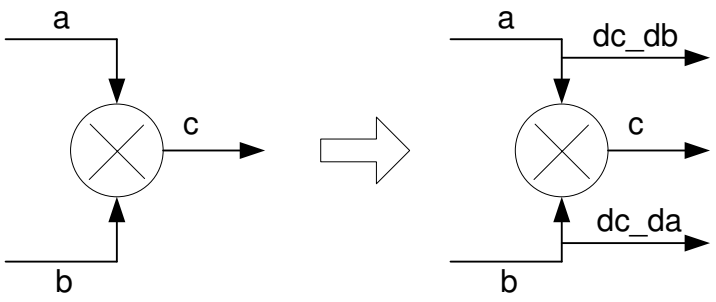


Figure 1.6. Transformation of a multiplier node to insert derivative monitors [Constantinides, 2003].

dict the error at the output of the system due to any scaling of a small perturbation of a signal s analytically, given the simulation-obtained error by a single such perturbation at s.

For the case of a general multiplier, $f(X_1, X_2) = X_1 X_2$, $\frac{\partial f}{\partial X_1} = X_2$ and $\frac{\partial f}{\partial X_2} = X_1$.

Within a synthesis tool, such Taylor coefficients can be recorded during a simulation run through the modification of the computational graph to include so-called *monitors* [Constantinides, 2003]. These data can then be used later for the error calculation step. Figure 1.6 shows a multiplier node and its transformation prior to simulation where the appropriate signals for monitoring the derivatives have been inserted.

Linearization. The linearization of the general multiplier is performed by transforming the general multiplier component in the computational graph into its Taylor approximation component as it is shown in Figure 1.7. Note that the model still has a general multiplier node, however one input is external to the model ensuring linearity. These new added external signals to the system read data from the derivative monitor files created by the above large-scale simulation.

Noise injection. In the case of a linear time-invariant system, the L_2 scaling was used to analytically estimate the variance of the noise at the outputs of the system. In this section, an extension of this approach is proposed for the case of non-linear systems.

The advantage of transforming the non-linear components of the system to linear components in the small-signal model is that if the variance of an output signal is V when it is excited by an error with variance σ^2 that it is injected to a signal of the system, then the same output signal

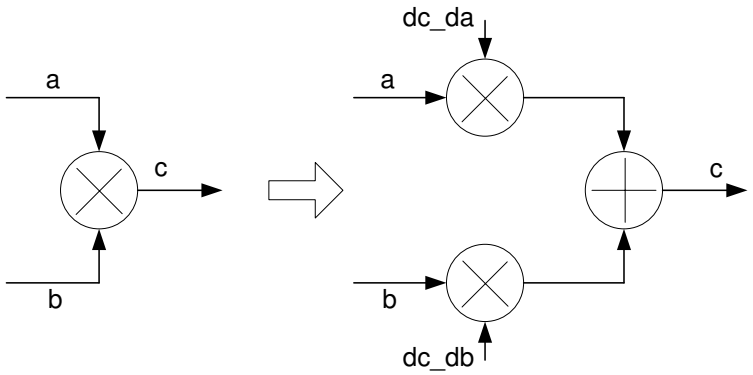


Figure 1.7. Transformation of a multiplier node to produce a linear model [Constantinides, 2003].

will have a variance aV when the injected error has variance $a\sigma^2$. This implies that if we can calculate the variance of an output signal for a given known variance of the injected error only once through simulation, then we can analytically calculate the variance of the output signal for any variance of the injected error.

In order to achieve that, an additional adder node is added in the system, which injects the error to the signal under investigation, and a simulation is performed for a known error variance. In the simulation, the error that is injected to the system due to truncation of two's complement signal is independent and identically distributed over the range $[-2\sqrt{3}, 0]$. The selection of unit variance for the injected noise allows us to make the measured output response an unscaled 'sensitivity' measure. To finalize the small-signal model, zeros are propagated through the original inputs of the system during the simulation leading to faster simulation results [Aho et al., 1986].

Word-length Optimization Algorithm

Given a computation graph $G(V, S)$ of a system, section 1.0 has already described how a scaling vector $\mathbf{p}$ can be derived. The total required area of the system can be expressed by a single metric $A_G(\mathbf{n}, \mathbf{p})$, which combines the area models of the components of the system. Finally, the error variances of the output of the system can be combined in a single vector $E_G(\mathbf{n}, \mathbf{p})$.

The Word-length optimization problem can be formulated as follows: Given a computation graph $G(V, S)$, select $\mathbf{n}$ such that $A_G(\mathbf{n}, \mathbf{p})$ is minimized subject to $\mathbf{n} \in N^{|S|}$ and $E_G(\mathbf{n}, \mathbf{p}) < \mathcal{E}$ where $\mathcal{E}$ is a vector that

defines the maximum acceptable error variance for each output of the system.

It can be demonstrated that the error variance at the outputs of the system may not be a monotonically decreasing function in each internal word-length. Moreover, it can be shown that error non-convexity may occur, causing the constraint space to be non-convex in $\mathbf{n}$. As it is demonstrated in [Constantinides et al., 2004], as long as the system remains well-conditioned, increasing the word-length of the output of the node types GAIN, ADD or DELAY can not lead to an increase of the observed error at the outputs of a system. However, a computation graph containing a 2-way FORK can exhibit such behavior that is not monotonic in the word-length vector. Moreover, in the case of systems that incorporate a 3-way FORK, non-convexity may arise. This non-convexity makes the word-length optimization problem a harder problem to find solutions [Fletcher, 1981].

A heuristic approach. It has been shown in [Constantinides and Woeginger, 2002] that the word-length optimization problem is NP-hard. Thus, a heuristic approach has been developed to find the word-length vector that minimizes the area of a system given under the set of constraints on the error variance at the outputs of the system.

The method starts from determining the scaling vector $\mathbf{p}$ of the system. After that, the algorithm estimates the minimum uniform word-length that can be used for all the signals in the system such that the error constraints are not violated. Each word-length is scaled up by a factor $k > 1$ which defined the upper bound that the signal can reach in the final design. A conditioning step is performed to transform an ill-conditioned graph that may has arise to a well-conditioned one.

In each iteration of the algorithm, each signal is visited in turn and its word-length is reduced until the maximum reduction in its word-length is found that does not violate the error constraints. The signal with the largest reduction in the area is chosen. Each signal's word-length is explored using binary search.

For completeness, in the case where the DSP system under investigation is an LTI system, optimum solutions can be found using a Mixed Integer Linear Programming formulation (MILP). However, it should be noted that the solution time of MILP formulations render the synthesis of large systems intractable. The interested reader is referred to [Constantinides et al., 2004].

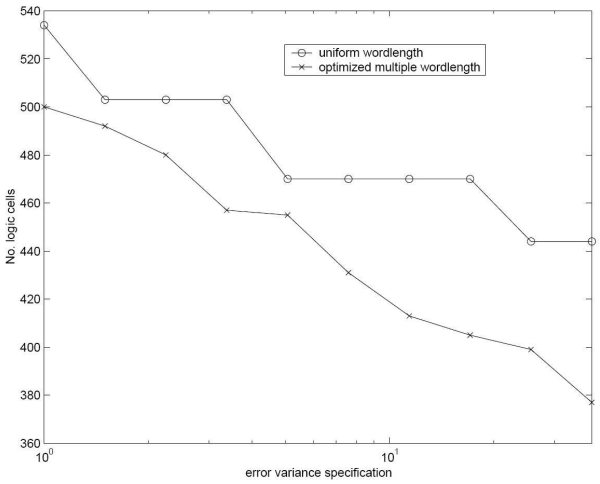


Figure 1.8. Area resources versus specified error variance for an IIR biquadratic filter [Constantinides et al., 2004] (published with kind permission of Springer Science and Business Media).

Some results

Figure 1.8 shows the place-and-routed resource usage versus the specified error variance at the output of an IIR biquadratic filter. The target device is an Altera Flex10k. This plot is a representative plot of the plots obtained by many designs. The plot shows the results obtained when uniform word-length is used in the system and when the multiple word-length scheme is applied. It can be seen that the multiple word-length approach results in designs that use between 2% and 15% less area for the same error specification at the output.

3. Synthesis and Optimization of 2D FIR filter designs

The previous section discusses the optimization of general DSP designs, focusing on peak value estimation and word-length optimization of the signals. This section focuses on the problem of resource optimization in Field Programmable Gate Array (FPGA) devices for a specific class of DSP designs. The class under consideration is the class of designs performing two-dimensional convolution *i.e* 2D FIR filters.

The two-dimensional convolution is a widely used operator in image processing field. Moreover, in applications that require real-time performance, in many cases engineers select as a target hardware platform an FPGA device due to its fine grain parallelism and reconfigurability

properties. Contrary to the firstly introduced FPGA devices consisting of reconfigurable logic only, modern FPGA devices contain a variety of hardware components like embedded multipliers and memories.

This section focuses on the optimization of a pipelined 2D convolution filter implementation in a heterogeneous device, given a set of constraints regarding the number of embedded multipliers and reconfigurable logic (4-LUTs). As before, we are interested in a "lossy synthesis" framework, where an approximation of the original 2D filter is targeted which minimizes the error at the output of the system and at the same time meets the user's constraints on resource usage. Contrary to the previous section, we are not interested in the quantization/truncation of the signals, but to alter the impulse response of the system optimizing the resource utilization of the design. The exploration of the design space is performed at a higher level than the word-length optimization methods or methods that use common subexpressions [Pasko et al., 1999; Dempster and Macleod, 1995] to reduce the area, since they do not consider altering the computational structure of the filter. Thus, the proposed technique is complementary to these previous approaches.

Objective

We are interested to find a mapping of the 2D convolution kernel into hardware that given a bound on the available resources, it achieves a minimum error at the output of the system. As before, the metric that is employed to measure the accuracy of the result is the variance of the noise at the output of the system.

From [Mitra, 2006] the variance of a signal at the output of a linear time invariant (LTI) system, and in our specific case of a 2D convolution, when the input signal is a white random process is given by (1.13), where σ_y^2 is the variance of the signal at the output of the system, σ_x^2 is the variance of the signal at the input, and $h[n]$ is the impulse response of the system.

$$\sigma_y^2 = \sigma_x^2 \sum_{n=-\infty}^{\infty} |h[n]|^2 \tag{1.13}$$

Under the proposed framework, the impulse response of the new system $\hat{h}[n]$ can be expressed as the sum of the impulse response of the original system $h[n]$ and an error impulse response $e[n]$ as in (1.14).

$$\hat{h}[n] = h[n] + e[n] \tag{1.14}$$

The new system can be decomposed into two parts as shown in Figure 1.9. The first part has the original impulse response $h[n]$, where the second part has the error impulse response $e[n]$. Thus, the variance of

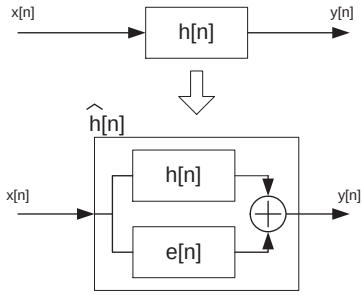


Figure 1.9. The top graph shows the original system, where the second graph shows the approximated system and its decomposition to the original impulse response and to the error impulse response.

the noise at the output of the system due to the approximation of the original impulse response is given by (1.15), where SSE denotes the sum of square errors in the filter's impulse response approximation.

$$\sigma^2_{noise} = \sigma^2_x \sum_{n=-\infty}^{\infty} |e[n]|^2 = \sigma^2_x \cdot SSE \tag{1.15}$$

It can be concluded that the uncertainty at the output of the system is proportional to the sum of square error of the impulse response approximation, which is used as a measure to access the system's accuracy.

2D filter optimization

The main idea is to decompose the original filter into a set of separable filters, and to one non-separable filter which encodes the trailing error of the decomposition.

A 2D filter is called separable if its impulse response $h[n_1, n_2]$ is a separable sequence, *i.e.*

$$h[n_1, n_2] = h_1[n_1]h_2[n_2].$$

The important property is that a 2D convolution with a separable filter can be decomposed into two one-dimensional convolutions as $y[n_1, n_2] = h_1[n_1] \otimes (h_2[n_2] \otimes x[n_1, n_2])$. The symbol $\otimes$ denotes the convolution operation.

The separable filters can potentially reduce the number of required multiplications from $m \times n$ to $m+n$ for a filter with size $m \times n$ pixels. The non-separable part encodes the trailing error of the approximation and still requires $m \times n$ multiplications. However, the coefficients are intended to need fewer bits for representation and therefore their multiplications

are of low complexity. Moreover, we want a decomposition that that enforces a ranking on the separable levels according to their impact on the accuracy of the original filter's approximation.

The above can be achieved by employing the Singular Value Decomposition (SVD) algorithm, which decomposes the original filter into a linear combination of the fewest possible separable matrices [Bouganis et al., 2005].

By applying the SVD algorithm, the original filter $\mathbf{F}$ can be decomposed into a set of separable filters $\mathbf{A}_j$ and into a non-separable filter $\mathbf{E}$ as follows:

$$\mathbf{F} = \sum_{j=1}^{r} \mathbf{A}_j + \mathbf{E} \tag{1.16}$$

,where r notes the levels of decompositions. The initial decomposition levels capture most of the information of the original filter $\mathbf{F}$.

Optimization algorithm

This section describes the optimization algorithm which has two stages. In the first stage the allocation of reconfigurable logic is performed, where in the second stage the constant coefficient multipliers that require the most resources are identified and mapped to embedded multipliers.

Reconfigurable logic allocation stage. In this stage the algorithm decomposes the original filter using the SVD algorithm and manifests the constant coefficient multiplications using only reconfigurable logic. However, due to the coefficient quantization in a hardware implementation, quantization error is inserted at each level of the decomposition. The algorithm reduces the effect of the quantization error by propagating the error inserted in each decomposition level to the next one during the sequential calculation of the separable levels [Bouganis et al., 2005].

Given that the variance of the noise at the output of the system due the quantization of each coefficient is proportional to the variance of the signal at the input of the coefficient multiplier, which is the same for the coefficients that belong to the same 1D filter, the algorithm keeps the coefficients of the same 1D filter to the same accuracy. It should be noted that only one coefficient for each 1D FIR filter is considered for optimization at each iteration, leading to solutions that are computational efficient.

Embedded multipliers allocation. In the second stage, the algorithm determines the coefficients that will be placed into embed-

Table 1.3. Filters tests

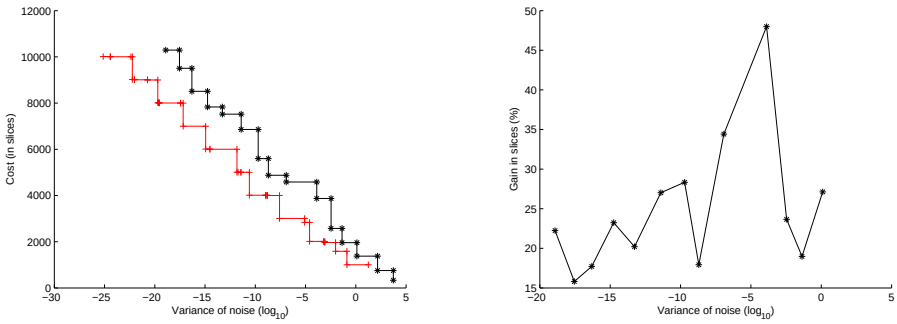
Test Number	Description
1	9×9 Gabor filter $F(x,y) = \alpha \sin\theta e^{-\rho^2(\frac{\alpha}{\sigma})^2}, \rho^2 = x^2 + y^2, \theta = \alpha x,$ $\alpha = 4, \sigma = 6$
2	9×9 Laplacian of Gaussian filter $LoG(x,y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}}$, $\sigma = 1.4$

ded multipliers. The coefficients that have the largest cost in terms of reconfigurable logic in the current design and reduce the filter's approximation error when are allocated to embedded multipliers, are selected. The second condition is necessary due to the limited precision of the embedded multipliers (*e.g.* 18 bits in Xilinx devices), which in some cases may restrict the approximation of the multiplication and consequently to violate the user's specifications.

Some results

The performance of the proposed algorithm is compared to a direct pipelined implementation of a 2D convolution using Canonic Signed Digit recoding [Koren, 2002] for the constant coefficient multipliers. Filters that are common in the computer vision field are used to evaluate the performance of the algorithm (see Table 1.3). The first filter is a Gabor filter which yields images which are locally normalized in intensity and decomposed in terms of spatial frequency and orientation. The second filter is a Laplacian of Gaussian filter which is mainly used for edge detection.

Figure 1.10(a) shows the achieved variance of the error at the output of the filter as a function of the area, for the described and the reference algorithms. In all cases, the described algorithm leads to designs that use less area than the reference algorithm, for the same error variance at the output. Figure 1.10(b) illustrates the relative reduction in area achieved. An average reduction of 24.95% and 12.28% is achieved for the Test case 1 and 2 respectively. Alternative, the proposed methodology produces designs with up to 50dB improvement in the signal to noise ratio requiring the same area in the device with designs that are derived from the reference algorithm. Moreover, Test filter 1 was used for evaluation of the performance of the algorithm when embedded multipliers are available. 30 embedded multipliers of 18×18 bits are made available in the algorithm. The relative percentage reduction achieved



(a) (b)

Figure 1.10. (a) Achieved variance of the noise at the output of the design versus the area usage of the proposed design (+) and the reference design (*) for Test case 1. (b) illustrates the percentage gain in slices of the proposed framework for different values of the variance of the noise. A *slice* is a resource unit used in Xilinx devices.

by the algorithm between designs that use the embedded multipliers and designs that realized without any embedded multiplier is around 10%.

4. Summary

This chapter focused on the optimization of the synthesis of DSP algorithms into hardware. The first part of the chapter described techniques that produce area-efficient designs from general block-based high level specifications. These techniques can be applied to linear time invariant (LTI) systems as well as to non-linear systems. Examples of these systems vary from finite impulse response (FIR) filters and infinite impulse response (IIR) filters to polyphase filter banks and adaptive least mean square (LMS) filters. The chapter focused on peak value estimation, using analytic and simulation based techniques, and on word-length optimization.

The second part of the chapter focused on a specific DSP synthesis problem, which is the efficient mapping into hardware of 2D FIR filter designs, a widely-used class of designs in the image processing community. The chapter described a methodology that explores the space of possible implementation architectures of 2D FIR filters targeting the minimization of the required area and optimizes the usage of the different components in a heterogeneous device.

References

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers: Principles, Techniques and Tools*. Addison-Wesley.

Benedetti, K. and Prasanna, V. K. (2000). Bit-width optimization for configurable dsps by multi-interval analysis. In *34th Asilomar Conference on Signals, Systems and Computers*.

Bouganis, C.-S., Constantinides, G.A., and Cheung, P.Y.K. (2005). A novel 2d filter design methodology for heterogeneous devices. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 13–22.

Constantinides, G. A. and Woeginger, G. J. (2002). The complexity of multilple wordlength assignment. *Applied Mathematics Letters*, 15(2):137–140.

Constantinides, George A. (2003). Perturbation analysis for word-length optimization. In *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*.

Constantinides, George A., Cheung, Peter Y.K., and Luk, Wayne (2002). Optimum wordlength allocation. In *10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 219 – 228.

Constantinides, George A., Cheung, Peter Y.K., and Luk, Wayne (2004). *Synthesis and Optimization of DSP Algorithms*. Kluwer Academic Publishers, 1 edition.

Dempster, A. and Macleod, M. D. (1995). Use of minimum-adder multiplier blocks in FIR digital filters. *IEEE Trans. Circuits Systems II*, 42:569 – 577.

Fletcher, R. (1981). *Practical Methods Of Optimization, Vol. 2: Constraint Optimization*. Wiley and Sons, New York.

Kim, S., Kum, K., and Sung, W. (1998). Fixed-point optimization utility for c and c++ based digital signal processing programs. *IEEE Trans. on Circuits and Systems II*, 45(11):1455–1464.

Koren, Israel (2002). *Computer Arithmetic Algorithms*. New Jersey: Prentice-Hall Inc., 2nd edition.

Lee, E. A. and Messerschmitt, D. G. (1987). Synchronous data flow. *IEEE Proceedings*, 75(9).

Liu, B. (1971). Effect of finite word length on the accuracy of digital filters - a review. *IEEE Transactions on Circuit Theory*, 18(6):670–677.

Mitra, Sanjit K. (2006). *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, third edition edition.

Oppenheim, A. V. and Schafer, R. W. (1972). Effects of finite register length in digital filtering and the fast fourier transform. *IEEE Proceedings*, 60(8):957–976.

Pasko, R., Schaumont, P., Derudder, V., Vernalde, S., and Durackova, D. (1999). A new algorithm for elimination of common subexpressions. *IEEE Transactions on computer-aided design of integrated circuit and systems*, 18(1):58–68.

Sedra, A. S. and Smith, K. C. (1991). *Microelectronic Circuits*. Saunders.

Wakerly, John F. (2006). *Digital Design Principles and Practices*. Pearson Education, Inc., fourth edition edition.