AREA WORD-LENGTH TRADE OFF IN DSP ALGORITHM IMPLEMENTATION AND OPTIMIZATION

A. Ahmadi, M. Zwolinski

Electronic System Design Group, School of Electronics and Computer Science, University of Southampton
Southampton, UK, {aa03r, mz}@ecs.soton.ac.uk

Keywords: High level synthesis, word length optimization, DSP implementation, digital noise.

Abstract

In this paper we propose a platform for High Level Synthesis of DSP algorithms while emphasising the differences between DSP systems and other digital systems. Accordingly, we allow variable word lengths within the system in order to optimize the system digital noise versus area. Using a particular target architecture, a suitable cost function, together with a synthesiser and optimizer and intermediate data bases have been implemented. Optimization is based on a Genetic Algorithm.

1 Introduction

Digital Signal Processing (DSP) is an important part of many systems. As with other digital systems, reducing the time to market and performance improvement are vital goals in design and implementation methodologies. Since most of the input data in signal processing algorithms are digital representations of floating point values, one of the problems in implementing signal processing algorithms on digital hardware is choosing an appropriate word length for arithmetic units. Traditionally this problem is solved by making a worst case assumption and choosing a single word length for all arithmetic units. This cannot be considered as an optimum choice because there are different types of arithmetic unit in the system and their accuracy has different impacts on the overall accuracy.

Several pieces of work have been reported in this respect with most of them focused on finding an optimal word-length for the algorithm in the first step and then designing or optimizing the system within that constraint, [2], [1]. In this approach the word-length is not considered in the subsequent optimization process. In other studies in which word-length has been considered [9], only simplified cases have been considered such that signals have been categorized into a few groups to constrain the word-length in all functional blocks.

In view of the fact that finding the optimum choice of word length for all system sub-blocks and arithmetic units as well as optimizing other costs is a very difficult and time consuming task, using High Level Synthesis (HLS) methods is inevitable. HLS methods mostly use a data path/controller model as a target structure. The synthesiser has to compile the high level description of the system to this target, where the data path represents the computational parts of the algorithm and the controller is the hardware implementation of the controlling statements in the high level specification. Although this approach may be efficient in some cases; since it is based on compiler methodologies rather than hardware implementation methods, it cannot be expected that it presents the best solutions in specific applications like DSP.

In hardware design, every specific application domain has its own characteristics and requirements which demand different approaches. In the case of DSPs, however, there are a vast variety of DSP algorithms and applications many of which can be classified as a set of matrix-based operations of which the majority are very suitable for distributed or parallel hardware architectures. Accordingly, different types of architectures and structures have been proposed and implemented in academic and industry research; classic examples can be found in [8], [7]. Our work is based on a single-bus distributed-control structure. Every sub-block of the system has its own controller which interacts with controllers at higher and lower levels of hierarchy.

A synthesiser has been designed to generate a synthesisable RTL description of the input algorithm. The input is C-like code. The synthesiser changes this input specification into a data structure with two different parts: the data path graph and the controller graph.

In order to obtain an efficient implementation of a DSP algorithm while computational requirements are satisfied as well as other design costs and constrains, it is vital to have a comprehensive description of the system and its costs. Accordingly a parametric relationship between cost functions based on word length as the controlling parameter has been provided. The cost function for area is based on empirical investigations of the basic blocks and controllers which had previously been done in MOODS [10]. Output digital noise of the system has been modelled using a Linear Time Invariant (LTI) system model of the algorithm. In this model, each arithmetic unit produces an additive noise to its output signal in which its power density depends on the functionality of the unit and the input and output word lengths. These noise sources are considered as a set of independent random inputs with a unified Probability Distribution Function (PDF). As a consequence of superposition, every noise source has its own impact on the output and accumulation of all these noises in the output computes the total digital noise of the system.

The synthesiser and its related data structures have been implemented in C++ and optimization has been done using the proposed cost function and with a Genetic Algorithm.

2 Motivation: High Level Synthesis and DSP Algorithms

HLS has been considered as a key factor in reducing the distance between initial specification and target design. This approach tries to hide the intermediate activities of system synthesis as much as possible to simplify the design process [3]. Because of the variety of possible applications, this field is still challenging in terms of the efficiency of the final system. As a result, domain-specific HLS tools have more chance to achieve a better efficiency.
There are some distinguishing differences between digital signal processors and other kind of digital systems that must be considered in HLS tools for DSP applications. First, many signal processing applications are not intrinsically digital in nature which means that the input data to the system will be a set of numbers which are an approximation of the real information, not the exact values. In this sort of application, the existence of the computational error is inevitable but the value of the computational error is a matter of concern. The second difference is that signal processing algorithms are massively computational. Most can be implemented by matrix-based operations which are simple arithmetic operations in multi-folded space. On the other hand, these algorithms are expected to run in a high speed (even real time) an environment which requires a massive memory access and management as well as high speed arithmetic operation blocks. These major differences suggest that general purpose HLS tools might not be able to give the best result in such cases. With this in mind, a HLS tool has been proposed to provide a more suitable mean for DSP synthesis.

3. Target Architecture

A HLS tool can be imagined as a compiler which translates a high level specification of the system to a low-level-synthesisable specification. From this point of view, all the techniques which have been invented in the field of compiler design can be used to achieve the best performance of the resulting design. Most synthesis tools split the target structure into two major parts: controller and data-path. The controller is a state machine which keeps the sequence of operations and controls the data-path blocks and the data-path is the part which does the computation. This structure is very suitable for implementing small systems but in the case of complicated systems with a diversity of sub-blocks it will be difficult to optimize the final hardware.
To achieve a better performance we propose a methodology which is based on a soft-architecture as the target design structure. This soft-architecture is a virtual model of the final system which could be imagined as a general structure for the system. Since this architecture must be flexible, to cope with a variety of possible signal processing systems, it has four basic parts: algorithm executers; interfaces; memories; and controllers. This model gives details of the sub-systems, interconnections structure, communication protocols and general aspects of system operation. Details of each sub-system or their functional blocks will be produced by the synthesiser according to system specification. Fig. 1 shows the target architecture and its hierarchical nature.

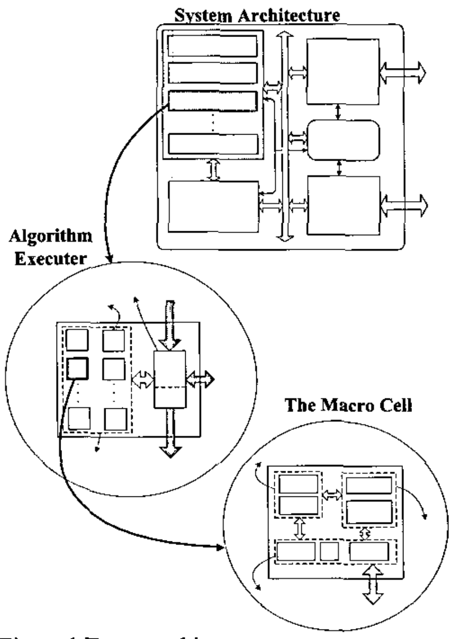

Figure 1 Target architecture

The HLS tool is based on this architecture and the resulting RTL VHDL files are low level specifications of the design matching with this architecture.

4 Cost Functions

There are measures or costs, by which the performance of a system can be evaluated. The most important examples of these are: area, speed and power consumption – which are common to all digital systems. As mentioned, there are differences between general purpose systems and specific application systems which necessitate different emphasises on costs. In the case of signal processing applications, apart from the other costs which are important in every digital system, accuracy has a vital role. Consequently, choosing a proper set of word-lengths for arithmetic units has a great impact on the system accuracy and area. This work investigates area-accuracy trade off in the proposed methodology for HLS of DSPs. Characteristics of area and accuracy parameters have are discussed in the next section.

4.1 Area

The area of a system can be divided into three parts: data paths; controllers; and interconnections. Changing the word length dose not change the controllers' area so it can be considered as a constant value in the cost function. On the other hand, changing the word length affects the area of the data path dramatically. Thus area can be represented by:

$$F_A(W) = A_{Controllers} + A_{Datapath}(W) + A_{Wires}(W) \qquad (1)$$

Where $F_A(W)$ is the total system area, $A_{Controllers}$ is the controller area (constant value), $A_{Datapath}(W)$ is the datapath area and $A_{Wires}(W)$ is interconnection area.

As an approximation of the datapath area, the area of units like adders, registers, buffers and switches can be assumed to have a proportional relationship to word length while the multiplier area can be modelled by a second order relationship with its word length; Table 1 shows such a simple approximation. Using this rough estimation gives the area by counting the number of each unit in the datapath.

Unit	Simple model
Multiplier	$K_M \cdot W^2$
Adder	$K_A \cdot W$
Register	$K_R \cdot W$
Buffer	$K_B \cdot W$
Bus switch	$K_S \cdot W$

Table 1 Area model for basic units

In Table 1, W is the word length of the unit, and K parameters are constant values.

4.2 Digital Noise

In practice, digital signal processing systems only can offer a finite number of binary digits to represent the signals to be processed. Fitting real values in these limited containers causes effects which can be categorized in several different ways. From a mathematical point of view, using a limited number of bits to represent a real number always means adding or removing unwanted and indeterminate information at the input, which is usually considered as an error or noise. To model this problem in our design tool and to inspect its impact; there are two problems which must be considered: first is a noise model for computational errors and second is a model of noise propagation through the hardware during operation.

Digital noise has been inspected and modelled in literature like [6]. In general, depending on the interaction of these computational errors, three major categories are recognizable: Quantization, Overflow and Round-off noise. Although all of these errors arise from word length limitation, their behaviour and effects on system could be different. Fig 2 shows a two pole IIR filter with its digital noise sources.

Quantization error corresponds to the representation of the input data and system coefficients by finite length digital numbers. These errors change the system transfer function, so some modifications are required to protect the system against this error [6]. Since the effects of these error sources on the system behaviour can be anticipated by finite word length simulations of the system level specification we do not consider this kind of noise in our work.

Round-off noise, on the other hand, is generated by rounding or truncation operations that follow the various arithmetic operations. In ordinary digital systems this error mostly happens in the case of multiplication where the result of product of two W-bit fixed-point fractions is a (2W-1) bit number that must eventually be sized for the next arithmetic unit word length by rounding or truncation as depicted in fig 4. In this study, in which a non-unified word length has been used, this sort of noise is the major form of the error. Consequently, our noise cost function is based on it.

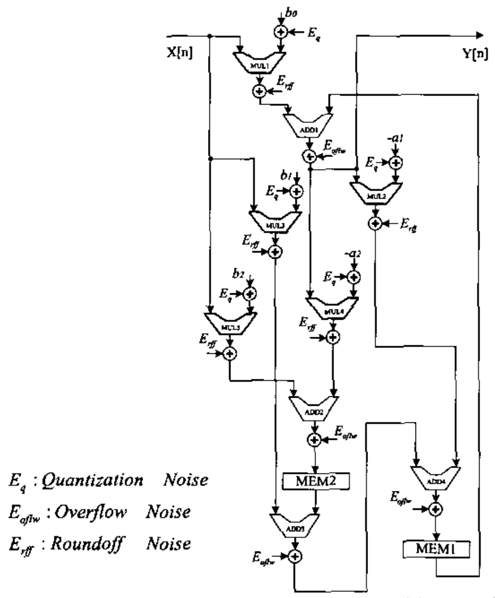
X[n]
Y[n]
MEM2
MEM1
E_q : Quantization Noise
E_{oflw} : Overflow Noise
E_{rff} : Roundoff Noise

Figure 2 A Two Pole IIR Filter structure with its Noise Sources

Overflow error comes from this fact that every arithmetic unit with limited word length has an upper bound for its results and if a result exceeds this bound, it will be changed fatally. However a good design must be protected against this kind of error, in stream computing systems with feedback (Real Time IIR system for instance) there is a possibility of overflow because of accumulation of other noise sources which are non-deterministic. This error can cause instability in the system behaviour [4]. Fig 3 shows a simple structure for this fatal error.

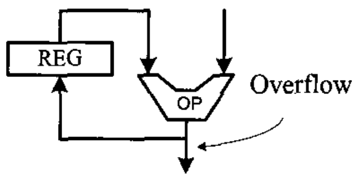

Figure 3 Limit Cycle production

As a common assumption these noise sources have been formulated as independent white noise in DSP design [5]. In this assumption, knowing the variance and mean value will be enough to provide an acceptable approximation for noise in the system output(s).

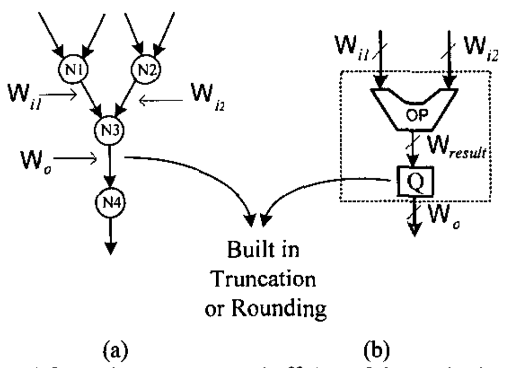

Figure 4 Operation output round off a) DFG in synthesiser data base b) hardware representation

To provide a noise propagation model, it must be recalled that many DSP algorithms can be considered as LTI systems. This assumption allows us to use superposition of independent noise sources to compute the noise effect on the system output, [2], [5]. The effects of noise sources on the output can be approximated using Equation (2).

$$E_{Output} = \sum_{k=1}^{M} \sigma_k^2 \cdot (L_2\{H_k(z)\})^2 , \qquad (2)$$

where $H_k(z)$ is the Z-transform of transfer function $(h[n])$ from the kth noise source to the output and $L_2\{\ \}$ is the L-Norm [5], given by Equation (3).

$$L_m\{H(z)\} = \left[\sum_{n=0}^{\infty} \left| Z^{-1}\{H(z)\}[n] \right|^m \right]^{\frac{1}{m}} . \qquad (3)$$

In addition, σ_k can be found in a multi word-length paradigm as in [2] from Equation (4).

$$\sigma_k^2 = \frac{1}{12} 2^{2p} \left(2^{-2n_2} - 2^{-2n_1} \right), \qquad (4)$$

where n_1 is the present arithmetic unit word length and n_2 is the next arithmetic unit word length and p is the position of the decimal point.
This function is a good approximation of the output noise but not useful as a cost function. The output noise factor is a linear function of σ_k, which has an exponential relationship with word length, as in Equation (5).

$$E_k = \sum_{k=1}^{M} a_k \cdot e^{-1.386 \cdot W_k} , \qquad (5)$$

where a_k is a constant which can be shown to be related to the system structure and parameters. This relationship means that the sensitivity of the cost function to the word length, as the controlling factor, has been reduced exponentially. This loss of precision causes difficulties in optimization of the design by reducing the speed of convergence of the optimization algorithm.

5 Implementation

The proposed system design methodology starts from a hierarchical specification of the target system. The design methodology is based on three parts: the functional blocks data base; soft-architecture; and the synthesiser and optimizer; as depicted in Figure 5.

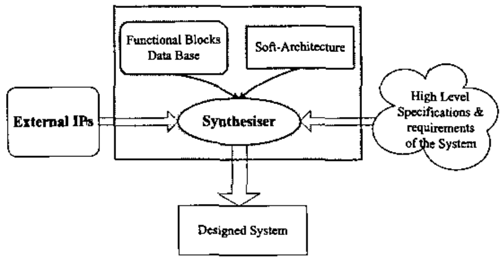
Functional Blocks Data Base
Soft-Architecture
External IPs
Synthesiser
High Level Specifications & requirements of the System
Designed System

Figure 5 General description of proposed HLS methodology.

The functional block database is a library of functions and sub-systems which may be used in a signal processing system. There are four kinds of such sub-systems in our method: algorithm executers, interfaces, memories and controllers, which each contain further functional and sub-blocks. In addition to sub-system implementation information, this database provides the required information for the design optimizer cost functions including: area, accuracy, delay and power consumption.

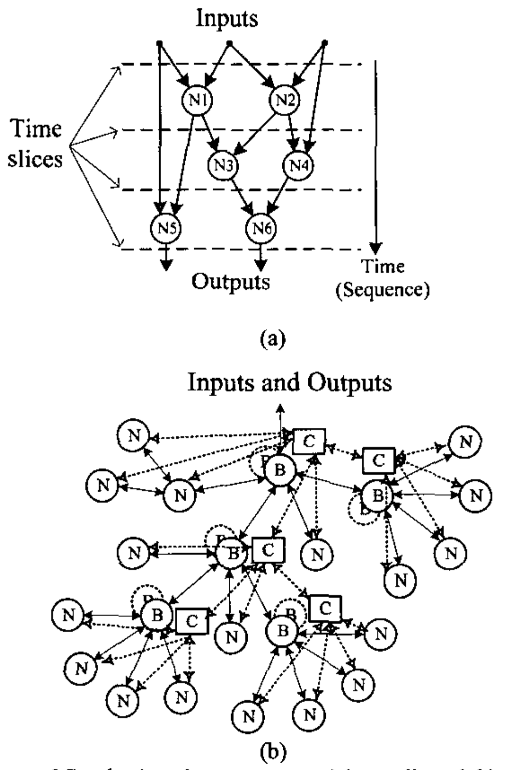

Figure 6 Synthesiser data structures a) input digraph b) implementation architecture digraph.

The synthesiser is a synthesis-optimization tool. Its input is a high level specification of the algorithm in a C-like format. Basically there is a pre-defined hierarchical architecture for the target system to which the target system must be mapped. The starting specification of the system and the final implementation are both represented by a directed graph (digraph) data structure. Figure 6 shows a general form of the input and output data structure of the synthesiser.

In Figure 6(a), a simple scheduling diagram of the input algorithm has been depicted in the form of a digraph where numbered nodes are arithmetic operations in the algorithm. Fig 6(b) gives an example of an implemented algorithm in the form of a digraph. N-type nodes in this digraph are hardware implementations of functional units which each could be a nested digraph and represent a sub system. B nodes represent bus switches; they control the data communication between levels of architecture or functional units. C nodes are controllers which control the bus switches and N nodes. Apart from controlling datapaths, controllers have a hierarchal relationship with each other which makes them a single controller in distributed form. From this point of view, this data structure is divided into two parts: a network of controllers and a network of data paths.

The synthesiser uses a set of library files to produce Intermediate Code (ICD) files. The library files contain basic blocks of the system and their cost relationships (noise, area, power, and delay) are functions of word length. These cost parameters can be used in a cost evaluation program after scheduling, allocation and binding to optimize the design.

As it can be seen from Figure 6(b), units in the data paths are connected to each other in a form of single bus structure. This means that the bus length always is equal to the maximum value of the word lengths. In addition, regardless of the number of inputs and outputs only one word length (w) has to be assigned to each unit. Therefore, we define a vector of word lengths for units in the data paths as in Equation (6)

$$W = [w_1 \quad w_2 \quad w_3 \quad \ldots \quad w_M]. \qquad (6)$$

The cost functions depend on this word length vector. Here, the overall cost function is a linear combination of the noise and area cost functions, Equation (7).

$$F(W) = A_A \cdot F_A(W) + A_N \cdot F_N(W), \qquad (7)$$

where A_A and A_N are constant values.

Optimization has been done using a Genetic Algorithm (GA). It is a multi-objective optimization of area and noise. From experience, using a simple random walk might take a very long time to reach the optimum point in this type of problem. For example in the case of word length optimization of a system with 50 units (M=50), the number of feasible solutions for word length between 4 and 32 ($4 \leq w \leq 32$) is $\approx 10^{73}$. Optimization would need a very long time to converge to an acceptable value. To improve this problem we designed a GA in such a way that in each generation three new types of individual are added: first are individuals which result from cross over between selected (by weighted roulette) last generation members; second is a set (with random number) of new randomly-produced members and third are new members which have been produced by slightly changing selected single members. Our experience shows that these changes improve the speed significantly. Table 2 gives the parameters which have been set in this optimizer. In this table K_1, K_2, K_3, K_4 are constant values and M is the number of units with variable word length; $P(x)$ is a random value. These all are dependent on the number of word lengths which have to be chose.

Parameter	**Value**
Number of Individuals in the Population	$K_1 \cdot M$
Number of crossovers	$K_2 \cdot M.P_2(x)$
Number of brand new Individuals	$K_3 \cdot M.P_3(x)$
Number of Increment/decrement Mutations	$K_4 \cdot M.P_4(x)$
Number of Generations (Iterations)	$K_5 \cdot M$

Table 2 Genetic Algorithm Parameters

The result of the synthesis is synthesizable RTL VHDL.

6 Results and Discussion

In our experiments we considered two basic examples, an order-18 FIR filter and a 4x4-DCT, to clarify how word length choice or optimization can affect the basic costs like circuit area and output noise. These algorithms use very basic structures, which have been repeated in many DSP applications with minor modifications.

Mathematical specifications of the algorithms [4] are used as the synthesiser input. RTL-Synthesizable VHDL files are produced by our tool based on target architecture.

Finally, area and noise costs are linearly combined in a multi-objective optimization by GA and Table 3 shows the comparative results for designs with uniform word lengths and the optimized case. In optimization $K_A = 1$ and $K_N = 10^{11}$ have been used as the weight parameters in costs combination of equation 7. The unit area is calculated based on the model of Table 1 and the noise model is based on Equations 3 and 4.

Design	**Costs**	**W=8**	**W=16**	**W=32**	**Optimized W**
FIR Filter	**Area**	13376	26873	53504	35816
	Output Noise	5.7×10^{2}	2.2×10^{4}	9×10^{9}	9.3×10^{6}
DCT	**Area**	28688	57376	114752	92246
	Output Noise	3.2×10^{2}	1.2×10^{4}	1.9×10^{9}	1.1×10^{7}

Table 3 Results for different word length

7 Conclusions

It is observed from the results that there is a meaningful relationship between area-noise cost and word length as a controlling parameter. In addition, since in a hardware implementation of an algorithm, unlike general

programmable CPU or DSP processors, word length can be chosen for each operational unit independently and these word lengths can be different; finding an optimized word length can balance contradictory costs like circuit area and output digital noise in complicated designs.
In this study, we proposed a methodology and target for implementing DSP algorithms and, accordingly, models of circuit area and output noise and their relationship with word length have been created. However this relationship is complicated and in a multi-objective optimization it might take a very long time to find the best choice for word length, but some modifications to the method make it achievable. Two basic algorithms have been implemented in our methodology with different word lengths. By comparison, word lengths which have been optimized method show a lower output noise with smaller circuit area in both designs.

References

[1] M. L. Chang and S. Hauck, "Precis: A Usercentric Word-Length Optimization Tool," *IEEE Design & Test of Computers*, **22**, 349 - 361, (2005).

[2] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis and Optimization of DSP Algorithms (Fundamental Theories of Physics S.)*: Kluwer Academic Publishers, (2004).

[3] G. De Micheli, *Synthesis and Optimization of Digital Circuits*: McGraw-Hill Education, (1994).

[4] P. S. R. Diniz, E. da Silva, S. L. Netto, and E. A. B. da Silva, *Digital Signal Processing: System Analysis and Design*: Cambridge University Press, (2002).

[5] A. V. Oppenheim and C. J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform.," *IEEE Proceedings*, **60**, 957-976, (1972).

[6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*: Pearson US Imports & PHIPEs, (1998).

[7] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*: John Wiley & Sons Inc, (1999).

[8] M. A. Richards, "The Rapid Prototyping of Application Specific Signal Processors (Rassp) Program: Overview and Status," presented at Workshop on Rapid System Prototyping (1994).

[9] W. Sung and K. Kum, "Simulation-Based Word-Length Optimization Method for Fixed-Point Digital Signal Processing Systems," *IEEE Transactions on Signal Processing* **43**, 3087 - 3090, (1995).

[10] A. C. Williams, A. D. Brown, and M. Zwolinski, "Simultaneous optimisation of dynamic power, area and delay in behavioural synthesis", *IEE Proc. C&DT,* **147**, 383-90, (2000).