IEEE Symposium on FPGAs for Custom Computing Machines, 2001.

The Multiple Wordlength Paradigm

George A. Constantinides, Peter Y.K. Cheung, Wayne Luk
Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT.
Department of Computing, Imperial College, London SW7 2BZ.

Abstract

This paper presents a paradigm for the design of multiple wordlength parallel processing systems for DSP applications, based on varying the wordlength and scaling of each signal in a DSP block diagram. A technique for estimating the observable effects of truncation and roundoff error is illustrated, and used to form the basis of an optimization algorithm to automate the design of such multiple wordlength systems. Results from implementation on a reconfigurable computing platform show that significant logic usage savings and increased clock rates can be obtained by customizing the datapath precision to the algorithm according to the techniques described in this paper. On selected DSP benchmarks, we obtain up to 45% area reduction and up to 39% speed increase over standard design techniques.

1 Introduction

Multiple wordlength or multiple precision algorithms have recently become a flourishing area of research. The move away from a single arithmetic computational unit towards forms of parallel processing has shown that the required precision of a calculation is an important dimension in designing fast, low-power, and area-efficient processors [1]. In FPGAs this is a particularly important trend, since the designer has the freedom to construct datapaths with precision customized for each algorithm loaded into the FPGA.

This paper explores and examines the multiple wordlength approach for digital signal processing (DSP) applications. We propose a paradigm based on varying signal scaling and precision between parallel processors implemented in an FPGA. It is shown that the fixed-point quantization effects of this approach can be estimated, how they are estimated, and how the design process can be automated from an initial infinite-precision behavioural specification.

The synthesis technique presented is a *lossy synthesis* approach. This term is used to denote that, while the behaviour of the synthesized system is not identical to that of the algorithm specification (due to fixed-point quantization effects), nevertheless the error can be specified and controlled by the user.

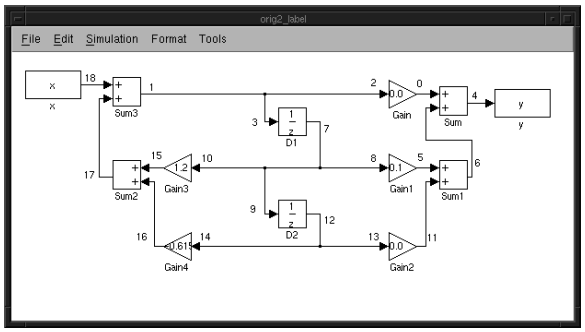

Figure 1: A Simple Block Diagram

The ideas presented have been implemented within the Synoptix high-level synthesis system [2] which, given a Simulink [3] block diagram and user-specified bounds on roundoff noise for each of the outputs, will provide an area-optimized DSP algorithm implementation in a hardware description language. An example block diagram used for specification is shown in Fig. 1.

Synoptix works by exploiting the differing properties of different paths between signals in the block diagram and outputs. Any given signal can be the input to many different blocks, and therefore its bit-width will impact significantly on the area consumed by each of those blocks. In addition, between the signal and each output there exists a system transfer function [4] that will scale any roundoff error injected at that signal. These are two independent effects, and their independence allows Synoptix to optimize the structure by assigning longer wordlengths to those signals whose area-impact is low and whose noise-impact is high, and by assigning shorter wordlengths to those signals whose area-impact is high and whose noise-impact is

low. The aim of the optimization tool is to minimize the implementation area of the filter, while meeting the user-specified constraints. Our approach is entirely analytical in nature, and therefore neither depends on an appropriate user-specified input data set, nor requires lengthy simulation runs to determine the extent of finite-wordlength effects.

The main contributions of this paper are therefore: a description and analysis of the multiple wordlength approach to designing DSP systems, a definition and analysis of 'well-conditioned' multiple wordlength systems and their use in predicting fixed-point error, and a demonstration of the advantages of the multiple wordlength approach on several benchmarks. While simplified descriptions of our approach have been reported elsewhere [2, 5], here we include an in-depth treatment of the optimization procedure, the noise model, and the results from our implementation.

This paper has the following structure. Section 2 reviews some of the other work in the area of wordlength specification and determination. Section 3 describes the proposed multiple-wordlength implementation style, together with transformations and optimizations on multiple wordlength structures. Section 4 presents some results from the optimizations, before Section 5 concludes the paper.

2 Background

The effects of using finite register length in fixed-point systems have been studied for some time. Oppenheim and Weinstein [4] and Liu [6] provide standard models for quantization errors and error propagation through linear time-invariant (LTI) systems. This early work tended to assume a fixed-wordlength computational machine, which leads to the assumption of a single uniform signal width. FPGAs give us the freedom to design special purpose hardware to implement DSP functions, with the datapath wordlength shaped to fit the application. In [7] alternative noise models are presented, more appropriate to structures with non-uniform signal widths.

There has been significant recent research into multiple precision implementations. This work can be broadly classified into three categories: correctness preserving transformations [1, 8], techniques for trading off truncation / roundoff error with area [5, 9, 10, 11, 12, 13], and high-level synthesis for multiple wordlength algorithms [14, 15, 16].

Correctness preserving transformations aim to reduce the wordlengths in computations to the minimum possible while preserving the behaviour specified in the original fixed-point algorithm description. Stephenson et al. [1] have proposed a technique based on compiler-passes involving data-flow analysis, where the precision information is carried by the data-flow. They propose propagating known information on data ranges both forward and backward through a data flow graph. The authors then compare a silicon compilation [17] of the original specification to one of the wordlength-minimized specification, showing a significant area reduction. This work has the advantage of being applicable to algorithms containing loop-carried dependencies, although the results can be pessimistic when compared to our approach due to the general nature of the loops modeled. Benedetti et al. [8] propose a similar approach based on interval analysis, limited to systems containing no feedback. These transformation based approaches can easily be incorporated into a high-level synthesis [18] data-flow.

A more general framework for solving the wordlength determination problem revolves around the idea we describe as *lossy synthesis*. This is the approach that correctness need not be preserved in the strict sense, but the resulting behaviour must not differ from the original by more than a user-defined amount. The applicability of lossy synthesis is clearly dependent on an appropriate definition of error in behaviour, and a metric for its quantification. In the present paper, we restrict the class of algorithms to linear time-invariant (LTI) discrete time systems [19], an area that allows us to define error in terms of signal to quantization noise ratio (SNR).

Most of the work on addressing the problem of trading quantization error with area has considered the issue from a software profiling perspective [9, 11, 12]. These approaches typically use operator overloading in object-oriented programming to maintain both an 'accurate' (usually double precision floating-point) and a fixed-point representation of each signal simultaneously. After simulation with a user-specified input data set, the range of each signal can be estimated and so the scaling can be decided. By comparison between the accurate and fixed-point versions of the outputs, empirical signal distortion statistics can be calculated for a given set of signal parameters.

A somewhat different approach is taken by [10, 13], where simulation results are used in combination with 'format propagation' and/or user interaction. These techniques use simple analytical methods, such as determining the maximum value of an addition from the independent maxima of its two inputs, in order to propagate simulation data through the algo-

rithm specification without losing information. These systems require less simulation overhead than those in [9, 11], as simulation is typically used only for a subset of signals in the specification. However format propagation can be a highly pessimistic approach: for example the two inputs to the adder may never reach their respective maxima during the same iteration, due to correlation between the two inputs. In addition feedback loops cause significant problems for these approaches, typically requiring user intervention. In contrast, our approach uses such format propagation only for determining the 'natural' wordlength of the result of each atomic computation. We then allow for a truncation or rounding of each signal before it forms the input to further operations. We can therefore elegantly deal with both correlation and feedback.

While the simulation-based approaches allow the use of nonlinear and time-varying components, the quality of the resulting SNR estimates are highly dependent on the input data sets used for simulation, or the user help given. In addition, long run-times are necessary for the simulations that form the basis of the optimization routines [11]. In contrast, we restrict the systems of interest to linear time-invariant discrete time systems which allow us to use an analytic framework for wordlength optimization incorporating both recursive (i.e. feedback) and non-recursive structures.

3 Multiple Wordlength

The multiple wordlength implementation style can best be introduced by comparison to more traditional fixed-point and floating-point implementations. DSP processors typically use fixed-point representations, as this leads to area and power efficient implementations, often as well as higher throughput than floating-point. In FPGAs, it is well known that a fixed-point implementation is generally more efficient than a floating point implementation for most DSP algorithms with low dynamic range [20]. A traditional fixed-point implementation is illustrated in Fig. 2(a). Each signal j in the block diagram representing a recursive DSP data-flow is annotated with a tuple (n_j, p_j) showing the wordlength n_j and scaling p_j of the signal. Fig. 2(d) shows the meaning of these two parameters: n_j is the number of bits in the representation of the signal (excluding the sign bit), and p_j is the displacement of the binary point from the LSB side of the sign bit towards word LSB. In a traditional fixed-point implementation, all signals have the same wordlength and scaling, although shift operations are often incorporated in order to provide an element of scaling control [11]. Fig. 2(b) shows a standard floating-point implementation where the scaling of each signal is a function of time.

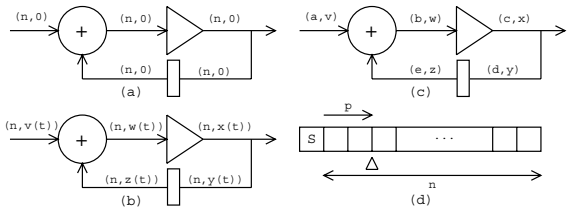

Figure 2: The Multiple-Wordlength Paradigm: (a) fixed-point DSP, (b) Floating Point, (c) Multiple Wordlength, (d) Signal Parameters: 's' indicates the sign bit

A single uniform system wordlength is common to both the traditional implementation styles. This is a result of implementation on single, or multiple, pre-designed fixed point arithmetic units. FPGAs can help us overcome this restriction for two reasons. Firstly, by allowing the parallelization of the algorithm so that different operations can be performed in physically distinct computational units. Secondly, by allowing the customization (and re-customization) of these computational units, shaping the precision of the datapath to the requirements of the algorithm. Together these freedoms point to an alternative implementation style shown in Fig. 2(c). This multiple wordlength implementation style inherits the speed, area, and power advantages of traditional fixed-point implementations since the computation is fixed-point with respect to each individual computational unit. However by potentially allowing each signal in the original specification to be encoded by binary words with different scaling and wordlength, the degrees of freedom in design are significantly increased.

During the design stage, each wordlength can be chosen individually to minimize logic usage while satisfying roundoff or truncation error. Similarly, each binary point position can be chosen individually to minimize logic usage while satisfying overflow or saturation error. Clearly if the new degrees of freedom in design are to be used appropriately it is necessary to have a good model of the lossy synthesis process: a quantification of the distance between the original specification and its implementation is required in error terms. The remainder of Section 3 describes how to estimate this, and how to optimize the wordlengths accordingly.

Table 1: Propagation of wordlengths $(n_j^{q\prime}, p_j') = (n_a, p_a)$ op (n_b, p_b)

Operation	Effect
Multiplication	$p_j' = p_a + p_b$ $n_j^{q\prime} = n_a + n_b$
Addition	$p_j' = \max(p_a, p_b) + 1$ $n_j^{q\prime} = \max(n_a, n_b + p_a - p_b) -$ $\min(0, p_a - p_b) + 1$

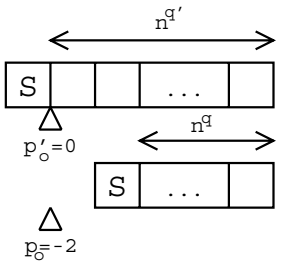

Figure 3: Wordlength and Scaling adjustment

3.1 Format Propagation

If each wordlength and scaling is specified independently, it means that there can be implied LSB-side width reduction (truncation or rounding) and/or MSB-side width reduction (inverse sign-extension) at each signal. Assuming that MSB-side width reduction involves no error (which can be ensured through using transfer-function based l_1 scaling [5]), we calculate the expected error injected at each signal and how this propagates to the output. For simplicity of implementation, we assume truncation is the chosen method of reducing wordlength at the LSB-side. In what follows, we assume that the wordlength and scaling of each signal has been pre-specified (for example by the optimization algorithm described in Section 3.3).

We first propagate the wordlength values and scalings from the inputs of each operator to its output, as shown in Table 1, where addition and multiplication are described. Together with branching and delays, these operations are sufficient to implement any linear time-invariant system. These wordlength values are then adjusted according to the known scaling of the output signal. If the desired binary point location at signal j is p_j, whereas the propagated value derived is p_j' $(> p_j)$, this corresponds to an MSB-side width-reduction. We then make the adjustment $n_j^q \leftarrow n_j^{q\prime} - (p_j' - p_j)$ where $n_j^{q\prime}$ is the propagated wordlength value, as illustrated in Fig. 3. The 'q' superscript indicates that this is a wordlength before quantization (truncation or rounding). This analysis allows us to efficiently make use of correlation information derived from a transfer function based scaling, for example the two inputs to an adder may never reach their peaks simultaneously. This is an example of a higher level 'don't-care condition' than that normally considered by synthesis tools.

We now define a *well-conditioned* wordlength-annotated block diagram to be one where the wordlength n_j for each signal j obeys $n_j \leqslant n_j^q$, where n_j^q is calculated as above. During wordlength optimization, as described in Section 3.3 below, ill-conditioned block diagrams could result. An ill-conditioned block diagram can always be transformed into an equivalent well-conditioned form in the following iterative manner. For each signal j such that $n_j > n_j^q$, we set $n_j^{(2)} \leftarrow n_j^q$. After this correction has been made for all violating signals, the propagation process described above is repeated to obtain new $n_j^{q(2)}$ values, which are then in turn checked for conditioning. This process is repeated, obtaining $n_j^{q(i)}$ values from $n_j^{(i-1)}$ through propagation, and assigning $n_j^{(i)}$ to the value of $n_j^{q(i-1)}$. Since this process only results in reductions in successive n_j values, it is clear from Table 1 that the process will terminate in a well-formed structure, where for some k, $n_j^{(k+1)} = n_j^{(k)}$ for all signals j. Recursive structures may require more than one iteration for termination.

3.2 Noise Model

The noise modeling process assumes a well-conditioned wordlength-annotated block diagram. We linearize the truncation process by replacing each truncation by an addition with an error signal, as illustrated in Fig. 4 [6]. Fig. 4(a) shows the specified (infinite precision) block diagram, illustrating data flow between three operations, $o1$, $o2$ and $o3$. Fig. 4(b) shows the implemented circuit, with wordlengths and quantization shown explicitly. In Fig. 4(c) the quantization blocks are replaced by additions for modeling purposes.

We assume quantization is possible at each signal j in the block diagram (Fig. 4). Where a signal is truncated, a noise source with variance σ_j^2 is constructed. These sources are assumed to be uncorrelated with each other and with themselves at non-zero time shifts [6]. However the noise model does not

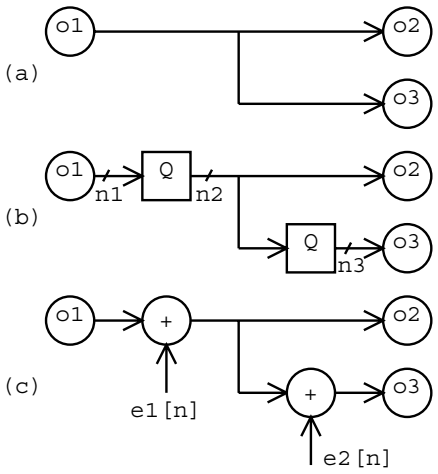

Figure 4: Truncation/Roundoff Model

follow the standard continuous rectangular distribution suggested by [6] since if the quantization involves the truncation of only a few bits, a discrete probability density function would provide a better approximation [7]. Thus if a signal with binary point p is truncated from n_1 to n_2 bits in a two's complement representation, the error variance injected at that signal is given by Eqn. 1 [7]. This value is derived from the assumption that each of the combinations of the low-end truncated bits is equally likely, an assumption that holds true in practice if the signals have sufficient dynamic range over that bit-width [19].

$$\sigma_j^2 = \frac{1}{12} 2^{2p} (2^{-2n_2} - 2^{-2n_1}) \tag{1}$$

Although we are now in a position to calculate the expected error variance injected at any signal within the structure, the user is unlikely to be worried about such information. The critical information from the user's perspective is the error variance at the *outputs* of the system. We therefore propagate these injected errors to each output in order to estimate their effect on overall system noise-performance. This is the stage of processing where the linearity and time-invariance of the system becomes essential. It is known that for a multiple-input, multiple-output LTI system with uncorrelated inputs I each of variance σ_i^2, the output $k \in K$ has variance given by Eqn. 2. Here λ_{ik} is a scaling factor which can be determined easily through an L_2 transfer function-based analysis [5]. This can clearly be applied to the error-prediction problem, where there are as many error inputs as signals in the

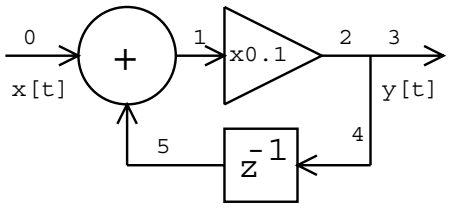

Figure 5: Noise estimation - example

original system specification.

$$\sigma_k^2 = \sum_{i \in I} \lambda_{ik} \sigma_i^2 \tag{2}$$

As an example of error noise prediction, consider the block diagram shown in Fig. 5 (z^{-1} represents a unit-sample delay). Let us set the input wordlength to 16 bits, with a peak input value of 1.0 and the coefficient wordlength to 8 bits. Let us further set each of the binary point positions and wordlengths for each of the signals labelled 0 to 5 to the values shown in Table 2 (p values have been chosen to match the theoretical peak signal value shown, which have been calculated through an l_1 scaling knowing the transfer function from the input to each signal). Using wordlength propagation, n^q values can be derived as described, from which variance values are calculated for each individual noise input Eqn. 1. The λ values are derived through an L_2 scaling by knowing the transfer function from each additional noise input to the output. The transfer functions used to calculate l_1 and L_2 scalings are shown in Table 2 using z-transform notation [19]. Using Eqn. 2 we get a total predicted error variance of $1.0274e-07$.

The ability to predict the output error variance for a given set of wordlengths allows us to formulate an optimization, constrained by output error variance. The following explains this optimization in detail.

3.3 Wordlength Optimization

Given a block diagram annotated with binary point positions for each signal, the wordlength optimizer defines a wordlength n_j for each signal j, referred to collectively in this section as the vector $\mathbf{n}$, with one element for each signal. Let E_k be the user-specified maximum allowable error variance on output $k \in K$, at which the estimated quantization error variance for wordlengths $\mathbf{n}$ is $\sigma_k^2(\mathbf{n})$ (calculated as shown in Section 3.2). Then given a logic usage estimator function $A_G(\mathbf{n})$, the problem may be defined as in Eqn. 3,

Table 2: Noise estimation example - signal parameters ('Transfer function' abbreviated to 'T.F.')

Signal	Peak	p	n	n^q	σ^2	λ
0	1.000	1	8	16	5.0862e-06	0.0101
1	1.111	1	8	12	5.0664e-06	0.0101
2	0.111	-3	15	15	0	1.0101
3	0.111	-3	15	15	0	1.0000
4	0.111	-3	8	15	1.9867e-08	0.0101
5	0.111	-3	8	8	0	0.0101

Signal	T.F. from Input	T.F. to Output
0	1	$\frac{0.1}{1-0.1z^{-1}}$
1	$\frac{1}{1-0.1z^{-1}}$	$\frac{0.1}{1-0.1z^{-1}}$
2	$\frac{0.1}{1-0.1z^{-1}}$	$\frac{1}{1-0.1z^{-1}}$
3	$\frac{0.1}{1-0.1z^{-1}}$	1
4	$\frac{0.1}{1-0.1z^{-1}}$	$\frac{0.1z^{-1}}{1-0.1z^{-1}}$
5	$\frac{0.1z^{-1}}{1-0.1z^{-1}}$	$\frac{0.1}{1-0.1z^{-1}}$

where Z^+ is the set of positive integers.

$$\begin{array}{l} \text{Minimize } A_G(\mathbf{n}) \text{ subject to} \\ \forall j \in J: \quad n_j \in Z^+ \\ \forall k \in K: \quad \sigma_k^2(\mathbf{n}) \leqslant E_k \end{array} \tag{3}$$

In our synthesis scheme we use a dedicated resource binding [18], so $A_G(\mathbf{n})$ can be calculated as the sum of individual area estimates for each library element. We have developed both a library of parameterizable arithmetic units and area models for these units, for implementation in Altera Flex10k FPGAs [21] in order to use the synthesized system on the Sonic reconfigurable computing platform [22].

This optimization is a Mixed-Integer Nonlinear Programming problem (MINP), and is complicated by the fact that neither the objective function nor the constraints are necessarily monotonic in any n_j [23]. However it has been shown that for more traditional uniform wordlength structures ($\forall j \in J : n_j = u$), $\sigma_k(\mathbf{n})$ does decrease monotonically with system wordlength u [23]. This allows us to construct an optimal uniform wordlength structure through the use of a binary search to find the smallest u such that all constraints are satisfied. This uniform structure then forms the starting point for our heuristic optimization procedure.

The heuristic procedure is illustrated in Fig. 6. After finding the optimal uniform wordlength u, this value is scaled up by a factor ($k = 2$ is sufficient for our benchmark set). This is because the main optimization loop only reduces signal wordlengths, so by an initial scaling $k > 1$ wordlengths are allowed to take on values greater than u. At this point that the structure may become ill-conditioned, requiring reduction to a well-conditioned structure, as described in Section 3.1. This then forms a starting point from which one signal wordlength is reduced by one bit each iteration. The signal wordlength to reduce is decided in each iteration by reducing all signal wordlengths until they violate an output noise constraint. At this point there is likely to have been some pay-off in reduced area, and the signal whose width reduction provided the largest pay-off is chosen.

Algorithm WLopt
Input: Block diagram, scaling factor k
Output: Optimized wordlength vector $\mathbf{n}$

```
u ← minimum uniform wordlength satisfying
      error constraints (binary search)
n_j ← ku for all j
reduce to a well-conditioned structure (Section 3.1)
do
    currentcost ← A_G(n)
    foreach signal j do
        w ← n
        bestmin ← currentcost
        minval ← minimum value of A_G(w)
            for w_j ∈ [0, n_j] such that
            error constraints are satisfied
        if( minval < bestmin ) do
            bestj ← j
            bestmin ← minval
        end
    end
    if( bestmin < currentcost )
        n_bestj ← n_bestj − 1
while( bestmin < currentcost )
```

Figure 6: The Wordlength Optimization Heuristic

Although this is a greedy algorithm, both the constraints and the objective function play a role in determining the direction of movement towards the solution. As a result, this algorithm is less dependent on local information than a pure steepest-descent search. Results from the synthesis of real circuits using this algorithm are presented in Section 4.

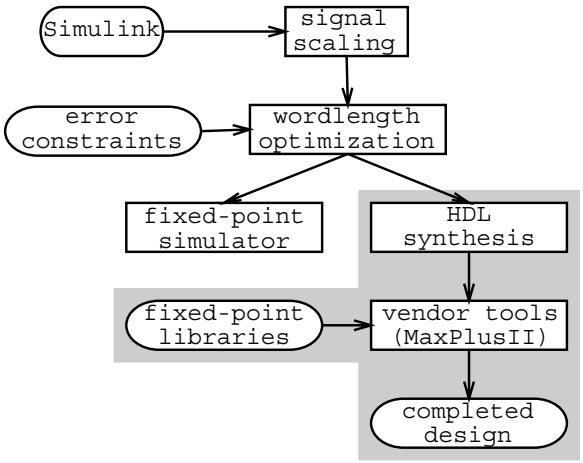

Figure 7: Synoptix Design Flow

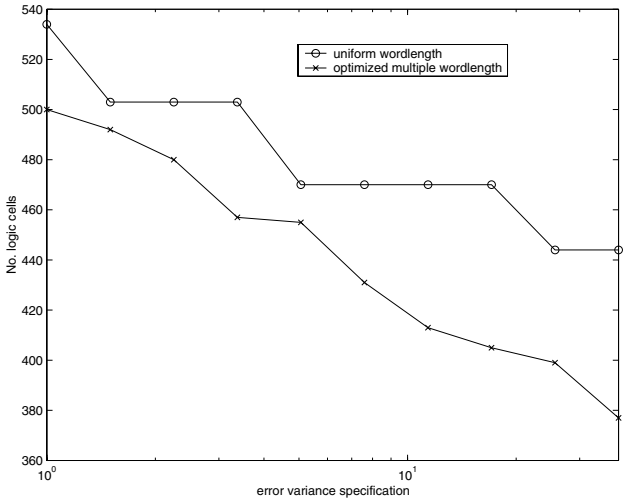

Figure 8: Circuit Area against Specified Error Power for an IIR Biquadratic Filter

4 Results

Synoptix, a complete synthesis system incorporating the algorithms in Section 3, has been developed for implementation of multiple wordlength systems in FPGAs. The input to the Synoptix is a Simulink block diagram, and the output is a structural description in VHDL or AHDL. FPGA vendor tools are then used to perform the low-level logic synthesis, placement and routing of the designs. The design-flow for implementation on the Sonic platform [22] is illustrated in Fig. 7, with the Sonic-specific parts shaded.

The system has been tested on several benchmark circuits, including finite impulse response (FIR) and infinite impulse response (IIR) filters, a discrete cosine transform (DCT), a polyphase filter bank (PFB), and an RGB to YCrCb convertor.

Shown in Fig. 8 is a graph of resource usage (measured in Altera Flex10k logic cells [21]) against specified error-power, which is representative in terms of the general shape of the plots achieved. The design is a simple second order (biquadratic) IIR digital filter. Both the multiple wordlength design and the optimized uniform wordlength structure are shown. The plot of area for a uniform wordlength decreases in steep steps. This is because there is a sudden change when the next-lowest wordlength becomes feasible with respect to the error constraints. This is not the case for the optimized multiple wordlength structures, since there are many more variables and hence the range of achievable error powers is much broader. Also the heuristic line lies consistently below the uniform line (by 2 to 15%), showing a consistent area saving for this design.

Table 3 illustrates some further results from larger benchmark circuits. Both the number of logic cells (LCs) and maximum clock frequency are given. Each of these results corresponds to a single point on the area-error power tradeoff curve for the circuit, placed and routed in an Altera Flex10k70RC240-3 device (as used in the Sonic [22] platform), except where otherwise stated. The FIR filter is a 126-tap linear phase low-pass Direct Form II [24] structure, suggested by [25] as a representative DSP design. The DCT is an 8-point, 1-dimensional decimation in time structure from [26] which has also been suggested as a benchmark by [25]. Two versions of this benchmark were synthesized, one (DCT^1) with equal error tolerance on all outputs, and the other (DCT^2) with required signal to noise ratio (SNR) reducing by 3dB per DCT coefficient, so that low frequency coefficients are less noisy than high-frequency ones. The IIR filter is of 4th order, as used by [27] and is of interest since it has a recursive (feedback) structure. The polyphase filter bank (PFB) is the design given in [28] for evaluation of the Streams-C compiler. The RGB to YCrCb is of the form suggested by the ITU [29], and allows some rounding error in the Cr and Cb outputs whereas the Y output is error-free. This is of particular interest since the multiple wordlength approach can clearly be used to customize the datapath in order to achieve these differential specifications. All these circuits were synthesized twice, once using an optimal

Table 3: Lossy Synthesis Results

Design	Uniform wl			Multiple wl	
	#LCs	f_{clk} (MHz)	width (bits)	#LCs	f_{clk} (MHz)
FIR†	6125	29.15	16	3356	36.23
DCT1*	1394	12.95	13	1311	13.67
DCT2*	1367	13.03	12	1164	13.53
IIR	701	9.57	12	623	9.32
PFB	321	30.03	15	273	31.34
RGB	438	11.58	18	272	16.15
* implemented on Flex10k70GC503-3					
† implemented on Flex10k200SRC240-1					

Table 4: Lossless synthesis results

Design	Multiple wl	
	#LCs	f_{clk}
FIR†	11110	†
DCT*	1530	12.77
PFB	332	38.31
RGB-YCrCb	547	11.99
* implemented on Flex10k70GC503-3		
† design too large for Flex10k200 device		

uniform wordlength structure and once using the multiple wordlength structure generated by the Synoptix tool. The DCT designs were synthesized on a device with a larger number of I/O pins, due to the I/O-limited nature of the designs, whereas the FIR filter was synthesized on a device with a significantly larger logic capacity.

It should be noted that even for the uniform wordlength structures, Synoptix was used to automatically insert power-of-two scaling [30], a good design practice in DSP design. Also note that for both uniform and multiple wordlength structures, these circuits represent a completely unpipelined implementation of the specification to aid direct comparison of maximum clock rate f_{clk} reported.

Table 3 illustrates that area reductions of between 6% and 45% (mean 22%) have been achieved by using the multiple-wordlength approach described in this paper. These area reductions have been accompanied by a speedup in maximum clock frequency between -3% and 39% (mean 12%), even though the estimated speed is not currently considered as part of our optimization (see Section 3.3). Interestingly, the only benchmark to have been slowed down slightly as a result of the optimization is the IIR filter. This is due to the increase of some signal wordlengths on the critical path around the feedback loops in this filter. Importantly, the largest area reductions and speedups have occurred in both the FIR filter, which is the largest design shown, and the RGB to YCrCb convertor, which has a structure ideally suited to multiple wordlengths since the error free Y is calculated first, from which Cr and Cb are derived [29].

With the exception of the IIR filter, all benchmarks were also synthesized with a specification of zero error. This is a degenerate case of our algorithm corresponding to *lossless* synthesis, and has been performed for comparison with other lossless approaches. (Note that the IIR filter cannot be synthesized in a lossless way due to the feedback.) Table 4 illustrates these results. Note that the FIR result for area is only an estimate reported by the synthesis tools since it was unable to be placed in the largest device supported by MaxPlusII, and for the same reason there is no clock frequency result for this benchmark under lossless synthesis. The most important fact to be gleaned from these results is that the correctness preserving transformation approach to high-level synthesis of DSP structures from floating-point specifications (see Section 2) is insufficient by itself to achieve results matching or improving on traditional DSP design techniques, when some output roundoff error can be tolerated. For as long as output error is not considered as a design variable by high-level synthesis systems, design specification languages must explicitly consider bit-width or sub-optimal designs will result.

5 Conclusion

This paper has introduced the design approach behind the multiple wordlength paradigm. It has been demonstrated that the multiple wordlength approach provides an elegant way of customizing data-flow intensive designs for efficient implementations in reconfigurable hardware. A technique has been presented for the synthesis and optimization of such multiple wordlength systems, and results show that it is desirable to use such optimizations on practical benchmark circuits.

We believe that it is time for high-level synthesis to start considering the design axis of *error*, alongside the traditional axes of area, latency, throughput and power. Error is a design parameter that current de-

sign practice leaves to a higher level of design, before implementation in a hardware description language. By considering error at the synthesis stage, we believe it is possible to achieve significant improvements.

We are currently extending our optimizations to certain nonlinear algorithms, and looking at the impact of resource sharing on multiple wordlength structures. Future work should consider in more detail the effect of multiple wordlength structures on design latency and throughput, and quantify the power-saving implications of this work.

Acknowledgements

The authors would like to acknowledge the support of Hewlett-Packard Laboratories and the U.K. Engineering and Physical Sciences Research Council.

References

[1] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," in *Proc. SIGPLAN Programming Language Design and Implementation*, Vancouver, British Columbia, June 2000.

[2] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Multiple precision for resource minimization," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*. April 2000, IEEE Computer Society.

[3] MathWorks, "Simulink," http://www.mathworks.com.

[4] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast fourier transform," *Proc. IEEE*, vol. 60, no. 8, pp. 957–976, 1972.

[5] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Roundoff-noise shaping in filter design," in *Proc. IEEE International Symposium on Circuits and Systems*, May – June 2000.

[6] B. Liu, "Effect of finite word length on the accuracy of digital filters – a review," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 6, pp. 670–677, 1971.

[7] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Truncation noise in fixed-point SFGs," *IEE Electronics Letters*, vol. 35, no. 23, pp. 2012–2014, November 1999.

[8] A. Benedetti and P. Perona, "Bit-width optimization for configurable DSP's by multi-interval analysis," in *Proc. 34th Asilomar Conference on Signals, Systems and Computers*, 2000.

[9] J. Choi, H. Jun, and S. Hwang, "Efficient hardware optimization algorithm for fixed-point digital signal processing ASIC design," *IEE Electronics Letters*, vol. 32, no. 11, May 1996.

[10] M. Willems, V. Bürsgens, H. Keding, T. Grötker, and M. Meyer, "System-level fixed-point design based on an interpolative approach," in *Proc. 34th Design Automation Conference*, June 1997, pp. 293–298.

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

[12] M. P. Leong, M. Y. Yeung, C. W. Fu, P. A. Heng, and P. H. W. Leong, "Automatic floating to fixed point translation and its application to post-rendering 3D warping," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, 1999, pp. 240–248.

[13] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens, "A methodology and design environment for DSP ASIC fixed point refinement," in *Proc. Design Automation and Test in Europe*, München, 1999.

[14] M. Ercegovac, D. Kirovski, and M. Potkonjak, "Low-power behavioural synthesis optimization using multiple precision arithmetic," in *Proc. 37th Design Automation Conference*, 1999.

[15] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Optimal datapath allocation for multiple-wordlength systems," *IEE Electronics Letters*, vol. 36, no. 17, pp. 1508–1509, August 2000.

[16] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Heuristic datapath allocation for multiple-wordlength systems," in *Proc. Design Automation and Test In Europe*, March 2001.

[17] J. Babb, M. Rinard, A. Moritz, W. Lee, M. Frank, R. Barua, and S. Amarasinghe, "Parallelizing applications into silicon," in *Proc. IEEE Symposium*

on Field-Programmable Custom Computing Machines, April 1999.

[18] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, 1994.

[19] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.

[20] W. B. Ligon, S. McMillan, G. Monnn, F. Stivers, and K. Underwood, "A re-evaluation of the practicality of floating-point operations on FPGAs," in *Proc. IEEE Symposium on FPGAs for Custom Computing Machines*, 1998.

[21] Altera Corporation, San Jose, *Altera Databook*, 1998.

[22] S. D. Haynes, J. Stone, P. Y. K. Cheung, and W. Luk, "Video image processing with the Sonic architecture," *IEEE Computer*, vol. 33, no. 4, pp. 50–57, April 2000.

[23] G. A. Constantinides, "Synthesis and optimization of arithmetic intensive circuits," MPhil. to PhD. Trans. Rep., Imperial College London, January 2000.

[24] S. K. Mitra, *Digital Signal Processing*, McGraw-Hill, New York, 1998.

[25] C. Lee, D. Kirovski, I. Hong, and M. Potkonjak, "DSP QUANT: Design, validation, and applications of DSP hard real-time benchmark," in *Proc. IEEE International Conference on Acoustics Speech and Signal Processing*, 1997, vol. 1, pp. 671–682.

[26] K. Parhi, *VLSI Digital Signal Processing Systems*, Wiley and sons, New York, 1999.

[27] K.-I. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: An optimizing floating-point to integer C program convertor for fixed-point digital signal processors," *IEEE Trans. Circuits and Systems II*, vol. 47, no. 9, pp. 840–848, September 2000.

[28] J. Frigo, M. Gokhale, and D. Lavenier, "Evaluation of the streams-C C-to-FPGA compiler: An applications perspective," in *Proc. ACM/SIGDA International Symposium on FPGAs*, 2001.

[29] B. L. Evans, "Raster image processing on the TMS320C7X VLIW DSP," http://www.ece.utexas.edu/~bevans/hp-dsp-seminar/07_C6xImage2/sld001.htm.

[30] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, February 1970.