Published in IEEE FPGA Custom Computing Machine Conference, 2002.

Optimum Wordlength Allocation

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 an approach to the wordlength allocation and optimization problem for linear digital signal processing systems implemented in Field-Programmable Gate Arrays. The proposed technique guarantees an optimum set of wordlengths for each internal variable, allowing the user to trade-off implementation area for error at system outputs. Optimality is guaranteed through modelling as a mixed integer linear program, constructed through novel techniques for the linearization of error and area constraints. Optimum results in this field are valuable since they can be used to assess the effectiveness of heuristic wordlength optimization techniques. It is demonstrated that one such previously published heuristic reaches within 0.7% *of the optimum area over a range of benchmark problems.*

1 Introduction

This paper addresses the problem of hardware synthesis from an initial, infinite precision, specification of a digital signal processing (DSP) algorithm. DSP algorithm development is usually initially performed without regard to finite precision effects, whereas for Field-Programmable Gate Array (FPGA) implementation, finite precision effects are often of critical importance.

It has been argued elsewhere [1] that often the most efficient FPGA implementation of an algorithm is one which supports a wide variety of finite precision representations, so that the best representation can be used for each internal variable. The accuracy observable at the outputs of a DSP system is a function of the wordlengths used to represent all intermediate variables in the algorithm. However accuracy is less sensitive to some variables than to others, as is implementation area.

The contribution of this paper is to present a technique for *optimum* wordlength allocation, for the case where the DSP algorithm to be synthesized is a linear, time-invariant (LTI) system [2]. Existing methods for wordlength allocation are heuristic by nature and thus it is difficult to measure the quality of the solutions produced by these methods. It is this uncertainty that has motivated the work set forth in this paper: to enable accurate characterization of the effectiveness of wordlength optimization techniques with respect to optimum solutions.

The wordlength optimization techniques of interest are those which allow a user-controlled trade-off between implementation area and signal quality at the DSP system outputs, such as those described by [1, 3, 4, 5].

The Mixed Integer Linear Programming (MILP) technique described in this paper has been applied to several small benchmark circuits, and the results compared to the heuristic presented in [1]. Modelling as a MILP permits the use of industrial-strength MILP solvers such as BonsaiG [6]. Although MILP solution time can render the synthesis of large circuits intractable, optimal results even on small circuits are valuable as benchmarks with which to compare heuristic optimization procedures. For this purpose the optimal specifications have been made available for public download for anyone interested in comparing new or existing wordlength optimization techniques.

Although the construction of the MILP is described in detail in this paper, no complete example MILP is given for space reasons. Several examples can be found at:
`http://infoeng.ee.ic.ac.uk/~gac1/OptimumWL`

This paper has the following structure. Section 2 describes the relevant literature in wordlength optimization, before the computation model and associated high-level area models are described in Sections 4 and 5. A brief review of the proposed noise model for LTI systems is then provided in Section 6, before the construction of the proposed MILP is given in Section 7. Results from application of the model to benchmark circuits are given in 8, before concluding the paper in Section 9.

2 Background

In [7] it has been demonstrated that a simplified version of the problem addressed in this paper is NP-hard. There are, however, several published approaches to wordlength optimization. Those offering an area / signal quality tradeoff are all of an heuristic nature [1, 3, 5] or do not support different fractional precision for different internal variables [4].

Bendetti and Perona [8] have proposed an analytic method for wordlength optimization based on interval arithmetic. The authors propose a 'multi-interval' approach, and demonstrate that the addition, subtraction, multiplication and division of the proposed intervals result in similar intervals, which may be propagated through a loop-free data-flow graph in order to estimate the wordlength required for a computation without losing any precision.

The Bitwise Project [9] propose a similar compiler-based technique based on propagating the ranges of integer variables backwards and forwards through data-flow graphs. Their focus is on removing unwanted most-significant bits (MSBs) – no technique is proposed for removing unwanted least-significant bits (LSBs).

The MATCH Project [4] also use compiler-based propagation through data-flow graphs, except they allow variables with a fractional component. All signals in their model must have equal fractional precision – the authors propose an analytic worst-case error model in order to estimate the required number of fractional bits.

Wadekar and Parker [3] have also proposed a methodology for wordlength optimization. Like [4], their technique also allows controlled worst-case error at system outputs, however they allow each intermediate variable to take a wordlength appropriate to the sensitivity of the output errors to quantization errors on that particular variable. A Genetic Algorithm is used to perform the optimization, and Taylor series are used to evaluate an estimate of the worst-case error at a system output for any given internal wordlengths.

Cmar *et al.* [10] have developed a wordlength optimization system which uses a combination of range propagation and simulation with known input vectors to limit the wordlengths of internal variables. An heuristic algorithm is applied whereby the wordlength is decided based on the value of an empirically derived scaling of the error standard deviation for each signal under simulation. The idea behind such an approach is that it sets an upper-bound on the wordlength of each variable, beyond which the least significant bits will be drowned in quantization or external noise. No additional mechanism is proposed to automate the tradeoff of system area against error.

Kum and Sung [5] have proposed several wordlength optimization techniques to trade-off system area against system error, some of which have been incorporated in the Cadence Signal Processing Worksystem [11]. These techniques are heuristics based on bit-true simulation of the design under various internal wordlengths. Some similar simulation-based work has been reported by Leong *et al.* [12].

In a previous paper [1] we present an optimization heuristic based on analytic average-case error analysis of LTI systems. Results of between 6% and 45% area reduction were achieved by our heuristic compared to the use of the optimum uniform wordlength design. However until this paper it has been impossible to effectively judge the quality of the solutions achieved due to the lack of an optimum comparative benchmark.

3 Notation

In this paper, the following notation is used.

For a directed graph $G(V, E)$, $\text{pred}(e)$ and $\text{succ}(e)$ indicate the predecessor and successor nodes of an edge $e \in E$. $\text{od}(v)$ denotes the out-degree and $\text{id}(v)$ denotes the in-degree of a node $v \in V$. For a node $v \in V$ with $\text{id}(v) = 1$, $\text{in}(v)$ denotes the signal driving node v. Similarly for a node $v \in V$ with $\text{od}(v) = 1$, $\text{out}(v)$ denotes the signal driven by node v.

Set subtraction is indicated by the operator $\backslash$.

For a function $f : X \to Y$, $f(X' \subseteq X) \subseteq Y$ denotes the subset $\{y \in Y | \exists x \in X' : f(x) = y\}$.

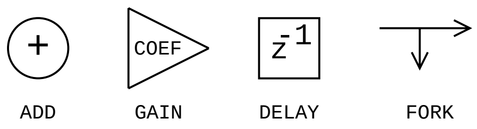
4 Computation Model

A computation graph $G(V, S)$ is the formal representation of an algorithm. V is a set of graph nodes, each representing an atomic computation or input/output port, and $S \subset V \times V$ is a set of directed edges representing the data flow. An element of S is referred to as a *signal*. The set S must satisfy the constraints on indegree and outdegree given in Table 1. We partition the set V into subsets $V = V_G \cup V_I \cup V_O \cup V_A \cup V_F \cup V_D$, representing the set of gain nodes, input nodes, output nodes, adder nodes, fork nodes and delay nodes respectively.

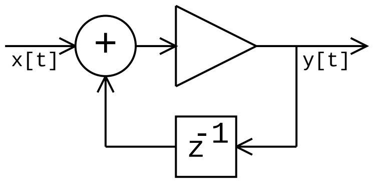
A graphical representation of a simple computation graph is shown in Fig. 1. Adders, constant coefficient multipliers and unit sample delays are represented using different shapes. Edges are represented by arrows

Table 1: Degrees of nodes in a computation graph

type	id(v)	od(v)
INPORT	0	1
OUTPORT	1	0
ADD	2	1
DELAY	1	1
GAIN	1	1
FORK	1	≥ 2

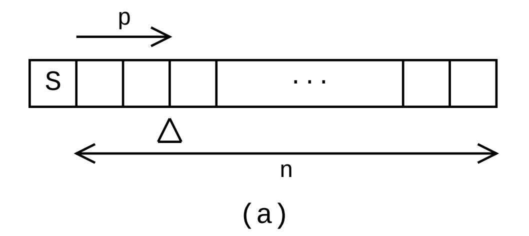


(a) some nodes in a computation graph



(b) an example computation graph

Figure 1: The graphical representation of a computation graph



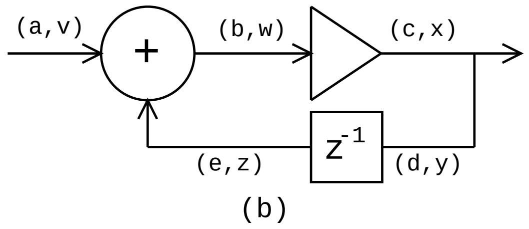


Figure 2: The Multiple-Wordlength Paradigm: (a) Signal Parameters: 's' indicates the sign bit (b) A multiple wordlength architecture,

During the design stage, each wordlength is chosen individually to minimize logic usage while satisfying roundoff or truncation error. The contribution of this paper is to perform this design optimally.

indicating the direction of data flow. Fork nodes are implicit in the branching of arrows. Inport and outport nodes are also implicitly represented, and may be labelled with the input and output names, $x[t]$ and $y[t]$ respectively in this example.

The algorithms described by computation graphs will be implemented using a multiple wordlength architecture, as introduced in [1]. This scheme will be briefly reviewed in order to aid the understanding of the remainder of this paper.

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 [13]. Each signal in a multiple wordlength architecture is allowed to take a distinct wordlength and scaling, appropriate to the internal variable represented by the signal. Fig. 2(a) 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.

5 Area Models

In order to formulate the error-constrained area minimization problem, it is necessary to construct high-level models of the area consumption of each type of node. Only adders, gains, and delays are considered to consume area resources on the FPGA; the remaining nodes are simply wiring or input/output constructs.

Model formulation has proceeded by defining a parameterized high-level area model from knowledge of the internal architecture of a component. The model parameters have then been tuned to the Xilinx Virtex series of FPGAs through the synthesis of many sample library elements using `coregen` and least-squares fitting to the theoretical model. Although the values of the model parameters presented are specific to Xilinx Virtex, the models themselves are general and can easily be re-tuned to alternative FPGA families and manufacturers or for ASIC implementation.

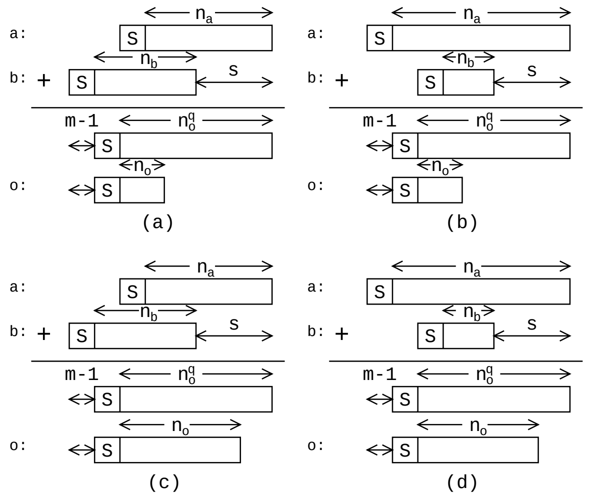


Figure 3: Multiple wordlength adder types

5.1 Adders

Usually adders are implemented in FPGAs as ripple-carry designs, since fast carry chains are provided in modern FPGA architectures [14]. Multiple wordlength implementations of adders can be conceptually quite complex due to the alignment required for signals of different wordlength or with different scaling (binary point location). Fig. 3 illustrates the adder types found in practice in multiple wordlength implementations [15]. The inputs of these adders have been arranged so that binary-point alignment requires left shifting of input b.

Even when the inputs to the adders have been so arranged, there are still four distinct cases as illustrated in Figs. 3(a)–(d). In Figs. 3(a) and (c), input b's most significant bit (MSB) extends beyond that of input a, whereas the opposite is true in Figs. 3(b) and (d). The remaining distinction concerns the output wordlength. In Figs. 3(a) and (b) the output is drawn entirely from the overlapping portions of signals a and b. By contrast the outputs in Figs. 3(c) and (d) draw a portion of their value from signal a alone – this portion is implementation cost-free.

The core integer adder used to implement such multiple wordlength adders will consist of a total of up to $\max(n_a - s, n_b) + 2$ single-bit adders if all MSBs of the result are required. However not all these adders will have equal cost, because those not driving a portion of the output signal, illustrated in Figs. 3(a) and (b), require carry logic but no sum logic. In Figs. 3(c) and (d) there are no such cases. Also it is important to note that not all possible MSBs of the summation may be required by the output. A total of m bits may not be required due to signal scaling information [1].

The output is drawn entirely from the overlap between signals a and b if $n_o + m \leq \max(n_a - s, b) + 1$. Thus the overall area of an adder can be modelled as (1).

$$A = \begin{cases} k_1(n_o+1) + k_2[\max(n_a - s, n_b) - m - n_o + 1], \\ \qquad \text{if } n_o + m \leq \max(n_a - s, b) + 1 \\ k_1[\max(n_a - s, n_b) - m + 2], \\ \qquad \text{otherwise} \end{cases} \tag{1}$$

For a Xilinx Virtex implementation our experiments suggest $k_1 \approx 1.0$ LUTs and $k_2 \approx 0.5$ LUTs.

5.2 Gains

Area estimation for constant coefficient multipliers is significantly more problematic. A constant coefficient multiplier can be implemented in FPGAs as a series of additions and subtractions, through a recoding scheme such as the classic Booth technique [16]. This implementation style causes the area consumption to be highly dependent on coefficient value. Although an ideal area model would account for a recoding-based implementation, this currently remains unimplemented. Instead we propose to use a 'coefficient blind' area model, which has been demonstrated in practice to provide good results [1, 15, 17]. The placed-and-routed area results attainable with the present implementation also provides an upper bound for those attainable by a more sophisticated model.

For the remainder of this section, we consider a gain node with input signal i, output signal o and a coefficient of wordlength CW.

The number of additions required to implement a constant coefficient multiplier is assumed to rise proportionally with the coefficient wordlength. Each of these will be a $(n_o + 1)$-bit addition. However, a total of $n_i + n_c - n_o$ additions along the edge of the multiplier array may not require their sum circuitry, as with the adder case. Note that this area model is equally valid with full-adder based array multipliers and standard Wallace or Dadda-tree [18] multiplier implementations.

$$A = k_3 \text{CW}(n_o + 1) + k_4(n_i + \text{CW} - n_o) \tag{2}$$

For a Xilinx Virtex implementation, our experiments over a wide range of coefficient values and wordlengths suggest values of $k_3 \approx 0.60$ LUTs and $k_4 \approx -0.85$ LUTs.

5.3 Delays

The area of a unit sample delay with input i, implemented as a register, is simply expressed as (3). For Xilinx Virtex implementation, our experiments suggest $k_5 \approx 1.0$ LUTs.

$$A = k_5(n_i + 1) \tag{3}$$

6 Noise Model

As shown in [1], since the systems of interest for this work have the LTI property, an analytic model based on [19] can be used to estimate the error at each system output. The variance of the error injected by each truncation of a signal from n_1 bits to n_2 bits is given by (4). If the transfer function from this point to the system output of interest is given in the z-domain as $H(z)$, then the resulting error variance at the output is $\sigma^2 L_2^2\{H(z)\}$, where $L_2^2\{\cdot\}$ denotes the well-known L_2-norm, included as (5) for completeness. ($\mathcal{Z}^{-1}\{\cdot\}$ represents the inverse z-transform).

$$\sigma^2 = 2^{2p}(2^{-2n_2} - 2^{-2n_1}) \tag{4}$$

$$\begin{aligned} L_2\{H(z)\} &= \left(\frac{1}{2\pi}\int_{-\pi}^{\pi} |H(e^{j\theta})|^2 d\theta\right)^{\frac{1}{2}} \\ &= \left(\sum_{n=0}^{\infty} |\mathcal{Z}^{-1}\{H(z)\}[n]|^2\right)^{\frac{1}{2}} \end{aligned} \tag{5}$$

A contribution of this paper is to demonstrate how to linearize these error models and hence incorporate them within a MILP model for the entire optimization problem.

7 MILP Model

The MILP formulation presented relies on some knowledge of integer linear programming. An excellent tutorial is given by Garfinkel and Nemhauser [20] on this topic.

The proposed MILP model contains several variables, which may be classified as: integer signal wordlengths, and signal wordlengths before quantization, binary auxiliary signal wordlengths, and auxiliary signal wordlengths before quantization, binary decision variables, real adder costs, and real fork node errors.

Note that only adders, gains, and delays cost area resources (forks are considered free). However adders have an inherently complex area model and thus while gains and delays are included directly in the objective function, the cost of each adder $V \in V_A$ is represented by a distinct variable A_v.

We are now in a position to formulate an area-based objective function for the MILP model (6), where $\textsc{cw}(v)$ represents the coefficient wordlength of gain node v.

$$\begin{aligned} \text{min: } & \sum_{v \in V_A} A_v + \sum_{v \in V_G} \Big\{(k_3\textsc{cw}(v) + k_4)n_{\text{in}(v)} - \\ & k_4 n_{\text{out}(v)} + (k_3 + k_4)\textsc{cw}(v)\Big\} + \sum_{v \in V_D} k_5 n_{\text{in}(v)} \end{aligned} \tag{6}$$

Constraints on quantization error propagation are much harder to cast in linear form due to the exponentiation, shown in Section 6. In order to overcome this nonlinearity, we propose to use an additional binary variables, $\bar{n}$, one for each possible wordlength that a signal could take. This is expressed in (7), and (8) ensures that each signal can only have a single wordlength value. Here $\setminus$ is used to denote set subtraction. Note that in order to apply this technique, it is necessary to know upper-bound wordlengths $\hat{n}_s$ for each $s \in S$. Techniques to derive these will be discussed in Section 7.1. Note that signals which drive fork nodes are not considered in this way, as fork node error models are considered separately (see Section 7.3).

$$\forall s \in S \setminus \text{pred}(V_F), n_s - \sum_{b=1}^{\hat{n}_s} b \cdot \bar{n}_{s,b} = 0 \tag{7}$$

$$\forall s \in S \setminus \text{pred}(V_F), \sum_{b=1}^{\hat{n}_s} \bar{n}_{s,b} = 1 \tag{8}$$

Using these binary variables it is possible to recast expressions of the form 2^{-2n_j}, which appear in error constraints (see Section 6), into linear form as $\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{j,b}$. Similarly it is necessary to linearize the exponentials in wordlengths before quantization (9)–(10).

$$\forall s \in S \setminus \text{pred}(V_F) \setminus \text{succ}(V_F), n_s^q - \sum_{b=1}^{\hat{n}_s^q} b\bar{n}_{s,b}^q = 0 \tag{9}$$

$$\forall s \in S \setminus \text{pred}(V_F) \setminus \text{succ}(V_F), \sum_{b=1}^{\hat{n}_s^q} \bar{n}_{s,b}^q = 1 \tag{10}$$

For each system output, we propose to use an error constraint of the form given in (11). Note that in this paper we only consider single-output systems, for simplicity of explanation, however the technique is general and our software can optimize multiple-input, multiple-output (MIMO) systems. $\mathcal{E}$ represents a user-defined bound on the error power at the system output, and hence on the signal quality.

$$\sum_{v \in V_F} E_v + \sum_{s \in S \backslash \mathrm{pred}(V_F) \backslash \mathrm{succ}(V_F) \backslash \mathrm{succ}(V_I)} 2^{2p_s} L_2^2\{H_s(z)\}\Big(\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{s,b} - \sum_{b=1}^{\hat{n}_s^q} 2^{-2b}\bar{n}_{s,b}^q\Big) + \sum_{s \in \mathrm{succ}(V_I)} 2^{2p_s} L_2^2\{H_s(z)\}\Big(\sum_{b=1}^{\hat{n}_s} 2^{-2b}\bar{n}_{s,b} - 2^{-2n_s^q}\Big) \leq 12\mathcal{E} \quad (11)$$

Note that those signals driven by system inputs are considered separately, since there is no need for Boolean variables representing the pre-quantization wordlength of a variable, as this parameter is defined by the system environment. Place-holders E_v are used for the contribution from fork nodes; these will be defined by separate constraints in Section 7.3.

7.1 Wordlength Bounds

Upper bounds on the wordlength of each signal, before and after quantization, are required by the MILP model in order to have a bounded number of binary variables corresponding to the possible wordlengths of a signal.

Our bounding procedure proceeds in three stages: perform an heuristic wordlength optimization on the computation graph [1]; use the resulting area as an upper-bound on the area of each gain block within the system, and hence on the input wordlength of each gain block; 'condition' the graph, following the procedure of [1]. The intuition is that typically the bulk of the area consumed in a DSP implementation comes from multipliers. Thus reasonable upper-bounds are achievable by ensuring that the cost of each single multiplier cannot be greater than the heuristically achieved cost for the entire implementation.

Of course this only bounds the wordlength of signals which drive gain blocks. In addition, the wordlength of signals driven by primary outputs is bounded by the externally-defined precision of these outputs. Together this information can be propagated through the computation graph, resulting in upper bounds for all signals under the condition that any closed loop must contain a gain block.

In the remainder of this paper, we denote by $\hat{n}_j$ the so-derived upper bound on the wordlength of signal $j \in S$ and by $\hat{n}_j^q$ the upper bound on the wordlength of the same signal before LSB truncation.

7.2 Adders

It is necessary to express the area model of Section 5.1 as a set of constraints in the MILP. Also a set of constraints describing how the wordlength at an adder output varies with the input wordlengths is required.

7.2.1 Area model

In the objective function, the area for each adder $v \in V_A$ was modelled by a single variable A_v. It will be demonstrated in this section how this area can be expressed in linear form.

Let us define β_v for an adder $v \in V_A$ with input signals a and b (12), where the inputs 'a' and 'b' are chosen to match with Fig. 3 so that it is b which needs to be left-shifted for alignment purposes. s_v is also illustrated in Fig. 3, and models the number of bits by which input b must be shifted.

$$\beta_v = \max(n_a - s_v, n_b) \quad (12)$$

We may then express the area of an adder as (13). Signal o is the output signal for the adder and m_v models the number of MSBs of the addition which are known through scaling to contain no information, as described in [1] and illustrated in Fig. 3. This value is independent of the wordlengths, and for an adder can be expressed as $m_v = \max(p_a, p_b) + 1 - p_o$.

$$A_v = \begin{cases} k_1(n_o + 1) + k_2\left[\beta - m_v - n_o + 1\right], \\ \qquad n_o + m_v \leq \beta + 1 \\ k_1\left[\beta - m_v + 2\right], \\ \qquad \text{otherwise} \end{cases} \quad (13)$$

The non-linearities due to the max operator in (12) and the decision in (13) must be linearized for the MILP model. This is achieved through the introduction of four binary decision variables δ_{v1}, δ_{v2}, δ_{v3} and δ_{v4} for each adder $v \in V_A$.

For the remainder of this section, we consider a general adder with inputs i and j and output o, to distinguish from the more specific case considered above, where input b was used to denote the left-shifted input to an adder. In order to model (12), if $p_j \leq p_i$ then (14)–(17) are included in the MILP. Otherwise (18)–(21) are included in the MILP.

$$n_i - n_j + p_j - p_i < \delta_{v1}(\hat{n}_i + p_j - p_i) \quad (14)$$
$$\beta_v - n_j + p_j - p_i \geq (1 - \delta_{v1})(-\hat{n}_j - p_i + p_j) \quad (15)$$
$$n_i - n_j + p_j - p_i \geq \delta_{v2}(-\hat{n}_j + p_j - p_i) \quad (16)$$
$$\beta_v - n_i \geq (1 - \delta_{v2})(-\hat{n}_i) \quad (17)$$

$$n_j - n_i + p_i - p_j < \delta_{v1}(\hat{n}_j - p_j + p_i) \quad (18)$$
$$\beta_v - n_i + p_i - p_j \geq (1 - \delta_{v1})(1 - \hat{n}_i - p_j + p_i) \quad (19)$$
$$n_j - n_i + p_i - p_j \geq \delta_{v2}(-\hat{n}_i + p_i - p_j) \quad (20)$$
$$\beta_v - n_j \geq (1 - \delta_{v2})(-\hat{n}_j) \quad (21)$$

Note that β_v and α_v are only bounded from below by the constraints given. Inequalities are used in order to allow disjunctions and thus implications, for example selecting $\delta_{v1} = 0$ in (14) gives $n_i - n_j + p_j - p_i < 0$, whereas selecting $\delta_{v1} = 1$ gives $\beta_v - n_j + p_j - p_i \geq 0$. Allowing δ_{v1} as an optimization variable results in $n_i \geq n_j - p_j + p_i \Rightarrow \beta_v \geq n_j + p_j - p_i$. Equality of A_v is guaranteed through its positive coefficient in the objective function.

In order to model (13), (22)–(25) are included in the MILP. These terms model the choice in (13) as a pair of implications, in an identical manner to that described above.

$$n_o - \beta_v + m_v - 1 \geq \delta_{v3}(m_v - \hat{\beta}_v) \quad (22)$$
$$A_v + (k_2 - k_1)n_o - k_2\beta_v + k_2(m_v - 1) - k_1 \geq$$
$$(1 - \delta_{v3})\left[(k_2 - k_1)\hat{n}_o - k_2\hat{\beta}_v + k_2(m_v - 1) - k_1\right] \quad (23)$$
$$n_o - \beta_v + m_v - 1 < \delta_{v4}(\hat{n}_o + m_v - 2) \quad (24)$$
$$A_v + k_1(m_v - \beta_v - 2) \geq (1 - \delta_{v4})k_1(m_v - \hat{\beta}_v - 2) \quad (25)$$

7.2.2 Output Wordlength

The pre-quantization output wordlength of an adder with inputs i and j and output o is given by $n_o^q = \max(n_i - p_i, n_j - p_j) + p_o$. We may express this as (26)–(27), since before-quantization wordlengths only appear with negative coefficient in the error so the error constraints can be relied upon to reduce n_o^q to achieve equality.

$$n_o^q \geq n_i - p_i + p_o \quad (26)$$
$$n_o^q \geq n_j - p_j + p_o \quad (27)$$

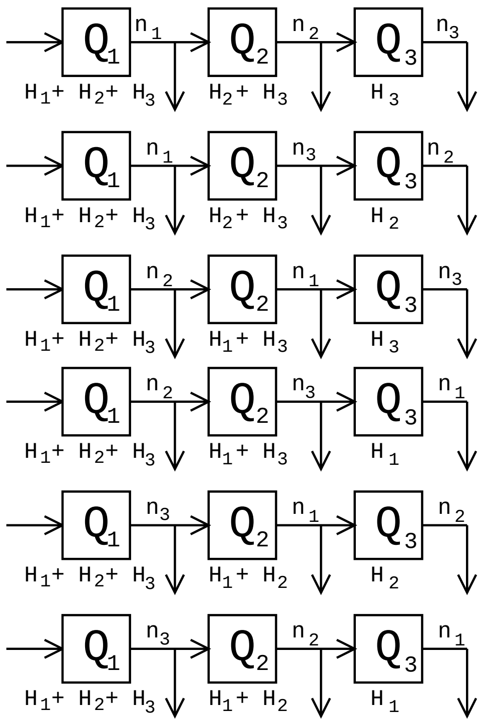


Figure 4: Possible output permutations in a 3-way fork

7.3 Forks

As demonstrated in [15], fork nodes can lead to unusual error behaviour due to the different possible orderings of wordlength at their outputs, which are required in order to guarantee freedom from statistical correlation and hence an accurate error model. Fig. 4 illustrates the six different possible configurations of a 3-way fork with outputs n_1, n_2 and n_3. For example, the top left figure corresponds to $n_1 \geq n_2 \geq n_3$ and the bottom right to $n_3 \geq n_2 \geq n_1$. Each of the 'Q' blocks is a truncation of least-significant bits in a signal. The z-domain transfer function from the truncation error injected, to the system output, is shown underneath the relevant 'Q' block.

In order for the MILP to fully model this behaviour it is necessary to consider each of the possible orderings. Let σ_v be a w-tuple, representing an order $(a, b, \ldots, f)$ on a w-way fork node $v \in V_F$ with input signal i. Thus, for example, $\sigma_v(2)$ is the second largest signal width. We may express the error resulting from truncation of those signals driven by node v as (28), with one constraint per possible σ, a total of $w!$. Here $\wedge$ represents Boolean conjunction.

$$\bigwedge_{r=1}^{w-1} (n_{\sigma_v(r)} \geq n_{\sigma_v(r+1)}) \Rightarrow$$
$$E_v = 2^{2p_i} \left(\sum_{r=1}^{w-1} L_2^2 \left\{ \sum_{h=1}^{w-r} H_{\sigma_v(h)} \right\} (2^{-2n_{\sigma_v(r+1)}} -2^{-2n_{\sigma_v(r)}}) + L_2^2 \left\{ \sum_{h=1}^{w} H_{\sigma_v(h)} (2^{-2n_i^q} - 2^{-2n_w}) \right\} \right) \tag{28}$$

Applying DeMorgan's theorem and linearizing the resulting disjunction gives (29)–(33). Each exponential is then further linearized through the procedure described in Section 7. The ϵ and η variables in (29)–(33) are additional binary decision variables and the right-hand side of each inequality consists of a trivial bound on the left-hand side, multiplied by a decision variable. At least one inequality is non-trivial, a property ensured by (33).

$$n_{\sigma(1)} - n_{\sigma(2)} < \epsilon_{v\sigma(1),\sigma(2)} \hat{n}_{\sigma(1)} \tag{29}$$
$$n_{\sigma(2)} - n_{\sigma(3)} < \epsilon_{v\sigma(2),\sigma(3)} \hat{n}_{\sigma(2)} \tag{30}$$
$$\cdots$$
$$n_{\sigma(w-1)} - n_{\sigma(w)} < \epsilon_{v\sigma(w-1),w} \hat{n}_{\sigma(w-1)} \tag{31}$$
$$E_v - 2^{2p_i} \left(\sum_{r=1}^{w-1} L_2^2 \left\{ \sum_{h=1}^{w-r} H_{\sigma(h)} \right\} (2^{-2n_{\sigma_v(r+1)}} -2^{-2n_{\sigma_v(r)}}) + L_2^2 \left\{ \sum_{h=1}^{w} H_{\sigma(h)} \right\} (2^{-2n_{\sigma_v(w)}} - 2^{-2n_o^q}) \right) \geq -\eta_{v\sigma} 2^{2(p_i-1)} \sum_{r=0}^{w-1} L_2^2 \left\{ \sum_{h=1}^{w-r} H_{\sigma(h)} \right\} \tag{32}$$
$$\sum_{r=1}^{w-1} \epsilon_{v\sigma(r),\sigma(r+1)} + \eta_{v\sigma} \leq w-1 \tag{33}$$

It is not necessary to explicitly consider quantization of the input signal to a fork node, since the above constraints use the pre-quantization wordlength of the fork input n_i^q. It is necessary, however, to guarantee that the input signal provides enough wordlength for the largest of its outputs (34).

$$\begin{aligned} n_i &\geq n_a \\ n_i &\geq n_b \\ &\cdots \\ n_i &\geq n_f \end{aligned} \tag{34}$$

7.4 Gains

In contrast to adders and fork nodes, gain nodes are straight-forward. The area of a gain node has already been modelled in the objective function (Section 7). The only remaining constraint required is to model the pre-quantization output wordlength of a gain $v \in V_G$ with input signal a, output signal o and coefficient of wordlength $\textsc{cw}(v)$ and scaling $\textsc{sc}(v)$ (35). This constraint is already in linear form.

$$n_o^q = n_a + \textsc{cw}(v) - p_a - \textsc{sc}(v) + p_o \tag{35}$$

7.5 Delays

Delay nodes also have a simple relationship between their input wordlength and their output wordlength before quantization, shown in (36) for the case of a delay node with input i and output o.

$$n_o^q = n_i \tag{36}$$

7.6 MILP Summary

A MILP model for the wordlength optimization problem has been proposed. It remains to quantify the number of variables (37) and constraints (38) present in the model. Note that the number of constraints given does not include integrality constraints, the unit upper bounds on Boolean variables, or the trivial fork constraints in (34) which do not form part of the optimization problem.

$$\begin{aligned} \text{vars} = & \sum_{s \in S \setminus \text{pred}(V_F)} (\hat{n}_s + 1) + \\ & \sum_{s \in S \setminus \text{pred}(V_F) \setminus \text{succ}(V_F)} (\hat{n}_s^q + 1) + \\ & |V_F| + \\ & 6|V_A| + \\ & \sum_{v \in V_F} \text{od}(v)(\text{od}(v)-1)\{1 + (\text{od}(v)-2)!\} \end{aligned} \tag{37}$$

$$\begin{aligned} \text{cons} = & 2|S \setminus \text{pred}(V_F)| + \\ & 2|S \setminus \text{pred}(V_F) \setminus \text{succ}(V_F)| + \\ & 1 + \\ & 10|V_A| + \\ & \sum_{v \in V_F} \text{od}(v)(\text{od}(v)-1)\{1 + 2(\text{od}(v)-2)!\} + \\ & |V_G| + |V_D| \end{aligned} \tag{38}$$

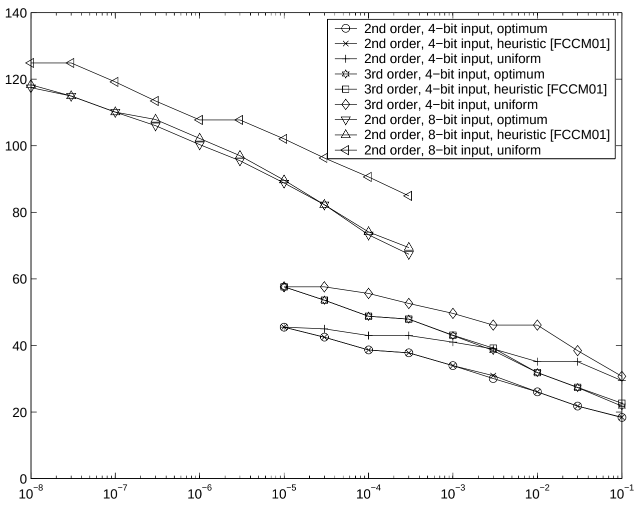


Figure 5: Area / Error tradeoffs compared for a 2nd and 3rd order FIR filter

It can be seen that so long as the number of large-fanout fork nodes are limited, the number of constraints in the MILP model grows approximately linearly in the number of nodes and signals. Under the same conditions the number of variables can grow up to quadratically with the number of signals because the upper bounds on each signal wordlength will vary approximately linearly with the number of large area-consuming nodes. Both parameters are dominated by any large-fanout fork nodes, since the number of η variables and their associated constraints grow combinatorially in the fanout.

8 Results

Fig. 5 illustrates area-error tradeoff curves for both a second and a third order linear phase FIR filter [2]. For the second order filter, results for both 4-bit and 8-bit inputs are given. For the third order filter, only results for a 4-bit input have been obtained. Three curves are present in each plot: the optimum uniform wordlength implementation, the heuristically derived multiple wordlength implementation from [1], and the optimum multiple wordlength implementation achieved by solving the MILP presented in this paper.

The results clearly illustrate the high-quality solutions achievable by the heuristic solution, averaging only 0.7% with a maximum of 3.9% worse than the optimum result.

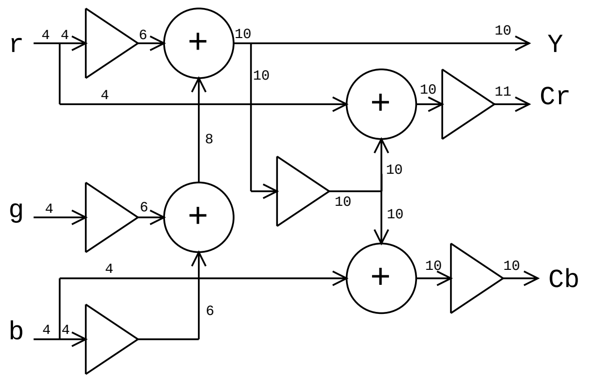


Figure 6: Optimal wordlength allocations for the ITU RGB to YCrYb converter

An optimum wordlength allocation for an RGB to YCrCb convertor described in [1] with 4-bit inputs has also been performed. This result shows an optimal cost of 78.61 LUTs, equal to the result achieved by the heuristic presented in [1].

Fig. 6 illustrates the structure [21] and optimum wordlengths of the RGB to YCrCb converter for 4-bit inputs (of range ± 112), 4-bit coefficients, and with an error-free Y, whereas a bounded error power of up to 10^{-2} has been allowed for Cr and Cb. We believe such optimum results, even for small circuits, to be highly valuable as a benchmark against which many new and existing heuristics [1, 3, 4, 5, 8, 9, 10] may be compared. For this reason we are making several optimum wordlength benchmarks publicly available at `http://infoeng.ee.ic.ac.uk/~gac1/optimumWL`.

The BonsaiG MILP solver [6] was used to solve the MILP models: execution time ranged from 2 seconds to 6 minutes on an AMD Athlon 1.2 GHz with 512 MB RAM. This compares to less than 0.2 second for the heuristic solutions on the same machine. Limits on the scale of the MILP solvable are due to both excessive run-time and numerical instabilities in the MILP solver.

9 Conclusion

This paper presents an approach to construct a mixed integer linear program (MILP) from an error-constrained area optimization problem, in order to perform wordlength allocation. High-level area models of parameterizable library blocks have been proposed and fitted to a Xilinx Virtex implementation. These form the basis of the objective function for the optimization, which is performed subject to user-specified constraints on output signal quality.

Results indicate that our previously proposed heuristic solution [1] produces results reaching the optimum in most cases and, on average, deviating only 0.7% from the optimum area.

Our current and future work is concentrating on wordlength optimizations of nonlinear DSP algorithms, and on including other models such as power consumption into the optimization procedure.

References

[1] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "The multiple wordlength paradigm," in *Proc. IEEE Symposium on Field Programmable Custom Computing Machines*, Rohnert Park, CA, April–May 2001.

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

[3] S. A. Wadekar and A. C. Parker, "Accuracy sensitive word-length selection for algorithm optimization," in *Proc. International Conference on Computer Design*, Austin, Texas, October 1998, pp. 54–61.

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

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

[6] L. Hafer, "Bonsaig," http://www.cs.sfu.ca/~lou/BonsaiG.

[7] G. A. Constantinides and G. J. Woeginger, "The complexity of multiple wordlength assignment," *Applied Mathematics Letters*, vol. 15, no. 2, pp. 137–140, February 2001.

[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] 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.

[10] 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.

[11] "Signal processing worksystem," http://www.cadence.com/eda_solutions/sld_spdv_13_index.html.

[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] 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.

[14] Xilinx, Inc., San Jose, *Field Programmable Gate Arrays*, 1998.

[15] G. A. Constantinides, *High Level Synthesis and Word Length Optimization of Digital Signal Processing Systems*, Ph.D. thesis, University of London, 2001.

[16] A. D. Booth, "A signed binary multiplication technique," *Quarterly J. Mechan. Appl. Math.*, vol. 4, no. 2, pp. 236–240, 1951.

[17] 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.

[18] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, Oxford, U.K., 2000.

[19] 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.

[20] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*, John Wiley and sons, New York, 1972.

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