PRECISION-WISE ARCHITECTURAL SYNTHESIS OF DSP CIRCUITS

Gabriel Caffarena[*] *and Carlos Carreras*[+]

[*] Universidad CEU-San Pablo, Urb. Montepríncipe, 28668, Boadilla del Monte, Madrid, Spain
email: gabriel.caffarenafernandez@ceu.es
[+]Universidad Politécnica de Madrid, C. Universitaria, 28040, Madrid, Spain.
e-mail: carreras@die.upm.es

ABSTRACT

This paper addresses the combination of wordlength optimization and architectural synthesis as a single design task, aiming at reducing the area of FPGA implementations. These two well-known design tasks are commonly applied sequentially. On one hand, wordlength optimization's goal is to find the fixed-point format of signals that minimizes cost. On the other hand, architectural synthesis optimizes the architecture of the implementation of an algorithm.

These two tasks are highly interdependent, since the wordlength minimization depends on the architecture and the architectural synthesis final output depends on the initial signal wordlengths. By combining them, a wider exploration of the design space can be performed.

A fine-grain combined wordlength optimization and architectural synthesis based on the use of simulated annealing is presented. The optimizer is tuned for DSP algorithms and is able to simultaneously optimize in terms of implementation area and output noise, thus leading to significant improvements. A complete comparison between the traditional sequential approach and the proposed combined approach is provided. Area improvements of up to 21% are reported.

1. INTRODUCTION

The high complexity involved in the design of DSP hardware systems requires the introduction of powerful Computer-Aided Design (CAD) tools to make their physical implementation possible. The design flow is divided into several tasks that allow the designer to traverse the path from the design specification to the implementation. Here, the combination of two well-known design tasks, Wordlength Optimization (WLO) and Architectural Synthesis (AS), as a single task is presented, aiming at the area optimization of DSP DSP hardware architectures.

WLO performs the translation of an initial infinite-precision description of a DSP algorithm into a finite-precision description (i.e. fixed-point). Fixed-point arithmetic is commonly used to implement DSP circuits, as it has proved to enable low-cost, low-power and high-speed implementations [1, 2]. It is necessary to wisely select the wordlengths of the different in the hardware implementation while keeping the output noise within requirements in order to minimize design costs.

AS' final goal is to find the best hardware architecture that complies with the power, area and speed constraints [3, 2, 4]. It is mainly composed of the scheduling, resource allocation and resource binding phases. AS transforms an initial algorithm specification (i.e. a Data Flow Graph (DFG)) into a Register Transfer Level (RTL) description, while meeting certain design constraints.

Traditionally, these two tasks have been carried out sequentially to reduce computational complexity. First, WLO is performed to obtain the wordlengths of the algorithm signals attending to an error constraint. Then, AS explores the design space to transform an initial description of a quantized algorithm and produces the final hardware architecture optimized in terms of cost (i.e. speed, etc.). The main drawback of this two-step methodology is that it does not take into account the interdependencies between these two tasks. On one hand, WLO cannot make use of a proper area/speed/power model since the architecture has not been yet completely defined. On the other hand, the architectural decisions made during AS are highly affected by the wordlengths of the signals. Hence, the final cost of the system after AS is not guaranteed to be minimal, since the architectural models used in both tasks may differ considerably.

The combined application of both design tasks allows the use of a single architectural model during the whole design process, thus leading to highly optimized implementations.

This paper contains the following contributions:

- A novel simulating annealing (SA) based architecture generator that combines AS and WLO into a single task where:
  - A complete set of resources (i.e. functional units, multiplexers and registers) is considered.
  - Functional units are wordlength-wise.
- Results for a set of DSP benchmarks under different error/latency constraint scenarios.

The paper is structured as follows: In section 2, related work is discussed. Section 3 presents the combined WLO and AS proposal. Performance results are collected in section 4. And finally, section 5 draws the conclusions.

2. RELATED WORK

Here, we present some of the most relevant works regarding the combination of WLO and AS.

One of the pioneer works in this field is [5], where the wordlength selection is carried out by minimizing a lower bound on the area cost of a resource sharing architecture. Once the wordlengths are selected, scheduling, resource allocation and resource binding are performed. The interesting point of this work is that quantization makes use of an estimation of the total area based on output latency and wordlengths. The latency of resources is considered variable, although a very simple model is used. The results are compared to an uniform wordlength approach (UWL), but there is no comparison to the traditional sequential approach.

In [1] WLA and HLS are interleaved. First, the system is quantized using a noise constraint a little bit more relaxed

 ISSN 2076-1465

than the required in the specification. Then, a datapath is created by applying scheduling, resource allocation and resource binding. Finally, the architecture is refined by increasing the FUs' wordlengths until the noise constraint is met. The work is valuable, however, the approach is too simplistic: a single optimization iteration, 1-cycle latencies FUs, etc. Again, the authors do not provide any comparison with a traditional two-step approach.

In [6] the combined problem is explored by means of MILP. Due to the very long computation times required by MILP, the problem complexity is reduced through some simplifications: 1-cycle latencies FUs, multiplexers and registers neglected, etc. The purpose of this work is to present initial results on the combined problem that can be used as a starting point to develop heuristics (see section 4). The results prove the validity of the combined approach.

The work in [7] is an extension of [1] and [5]. Here, FUs' costs (area, latency and power consumption) are wordlength dependent. Basically, WLO and AS are introduced within a loop. Operations are grouped after AS, based on mobility and wordlength information. The wordlength of each group is optimized by means of WLO to reduce cost. Every change in the wordlengths of operations produces a new datapath which leads to new groups of operations. The method iterates several times and records the best solution. Once more, the final results are not compared to the sequential approach.

The work presented here is based on a different approach, since we support a fine-grain optimization, where the architectural modifications regarding output precision and architecture components are considered within a simulated annealing (SA) optimization framework. Additionally, some of the drawbacks of previous approaches are overcome: simplified resource latency model [5, 1, 6], no comparison to traditional approach [5, 1, 7], multiplexers and registers neglected [5, 1, 6, 7], etc.

3. PRECISION-WISE ARCHITECTURAL SYNTHESIS

3.1 Formal description

Here we present the resource minimization problem constrained in terms of time (i.e. latency) and output error (i.e. wordlengths). The notation used is based on [6] and [8].

Given a sequencing graph $G_S(V,S)$, a maximum latency λ, a maximum output error ε and a set of resources R (e.g. functional units R_{FU}, registers R_{REG} and steering logic R_{MUX}), it is the goal of the combined WLO and AS to find the time step when each operation is executed (*scheduling*), the types, number and size (i.e. wordlength) of resources forming R (*resource allocation*), and the binding between operations and variables to functional units (FU) and registers (*resource binding*), that comply with the constraints while minimizing cost (i.e. area).

$G_S(V,S)$ is a formal representation of a single iteration of an algorithm, where V is the set of operations and $S \subset V \times V$ is the set signals that determines the data flow. We consider $V = V_M \cup V_G \cup V_A \cup V_D \cup V_I \cup V_O$ composed of typical DSP operations: multiplications, gains, additions, unit delays, and input and output nodes.

Signals are in two's complement fixed-point format, defined by the pair (n,p). Parameter n is the wordlength of the signal – not including the sign bit – and p is the scaling of the signal that represents the displacement of the binary point from the sign bit [6]. In order to obtain the quantization noise generated by a signal (ε_i) it is necessary to also know the *pre-quantization* fixed-point format (n_{pre}, p_{pre}) which expresses the format before any wordlength reduction. For instance, if a signal with a dynamic range of $[0,4)$ and precision of 2^{-8} is quantized reducing its precision to 2^{-5}, this operation can be expressed as a change from format $(n_{pre}, p_{pre}) = (10,2)$ to format $(n,p) = (7,2)$. Eventually, the noise contribution of each signal ε_i can be related to the total output error ε (see subsection 3.2 for more details).

Functional units (R_{FU}) are in charge of executing the set of operations from V. Registers (R_{REG}) store the data produced by FUs and some intermediate values. Finally, steering logic (R_{MUX}) interconnects FUs and registers by means of multiplexers. The set of FUs $R_{FU} = R_A \cup R_M$ is composed of adders and multipliers. An FU $r \in R_{FU}$ is defined by its type $type(r) = \{Adder, Multiplier\}$ and by its *size*, that depends on the input wordlengths. An operation is compatible with an FU if they have compatible types and if the size of the operation is smaller than or equal to the size of the FU [6, 9].

Scheduling is expressed by means of function $\varphi : O \rightarrow Z^+$, which assigns a start time to each operation. Resource binding, is divided into *FU binding* and *register binding*. *FU binding* makes use of the compatibility graph $G_C(V \cup R, C)$ [10], which indicates the compatible resources for each $v \in V$ by means of the set of edges $C \subset V \times R$. The binding between operations and resources is expressed by means of function $\beta : V \rightarrow R \times Z^+$, where $\beta(v) = \{r,i\}$ indicates that operation v is bound to the i-th instance of resource r. The compatibility rules impose that $(v,r) \subset C$. In a similar fashion, register binding links variables $d \in D$ to registers $r \in R_{REG}$ by means of function $\gamma : D \rightarrow R_{REG} \times Z^+$. The set of variables D is extracted from V considering that there is a variable assigned to the output of each operation from the subset $V_M \cup V_G \cup V_A$ and to each delay v_D connected to another delay. Registers have an associated size n_r that determines the maximum allowed wordlength of the variables bound to them.

The steering logic consists of the multiplexers required in front of FUs and registers to send data to and from these two types of resources. R_{MUX} is determined by φ, β and γ, since φ determines when data is generated, β when data is used by FUs, and γ where data is stored.

3.2 Noise estimation

Noise estimation is based on the assumption that the quantization of a signal s_i from n_{pre} bits to n bits can be modeled by the addition of a uniformly distributed white noise with the following statistical parameters [11]:

$$\sigma_i^2 = \frac{2^{2p_i}}{12}\left(2^{-2n_i} - 2^{-2n_i^{pre}}\right) \quad (1)$$

$$\mu_i = -2^{p_i-1}\left(2^{-n_i} - 2^{-n_i^{pre}}\right). \quad (2)$$

Using this model it is possible to quickly estimate the quantization error at the output of the algorithm for any given combination of $n_{pre,i}$ and n_i [4]. In particular we use the fast and accurate estimation methods for both LTI and non-linear systems from [12] that relates the wordlengths of the signals to the output error (see eqns.3-5). For instance, the wordlengths enable the computation of the noise statistic parameters $\vec{\sigma^2} = \left\langle \sigma_0^2, \ldots, \sigma_{|S|-1}^2 \right\rangle$ and $\vec{\mu}) = \langle \mu_0, \ldots, \mu_{|S|-1} \rangle$.

Vectors $\vec{v}$ and $\vec{m}$, and matrix M can be precomputed before WLO following the method in [12]. The use of fast estimator speed up process of WLO significantly (several orders of magnitude).

$$P_o = \vec{\sigma^2} \cdot \vec{v}^T + \vec{\mu} \cdot M \cdot \vec{\mu}^T \quad (3)$$

$$\mu_o = \vec{\mu} \cdot \vec{m}^T \quad (4)$$

$$\sigma_o^2 = P_o - \mu_o^2 \quad (5)$$

3.3 Resource modeling

The area of the circuits is estimated using the resource models in [9]. Resources are divided into three types: functional units (R_{FU}), registers (R_{REG}) and steering logic (R_{MUX}). The area and latency of FUs and registers (i.e. $A(r)$ and $l(r)$) are expressed as functions of the input and output wordlength information. They are obtained by applying curve fitting to hundreds of synthesis results. The use of accurate delay cost functions has proved to provided significant performance improvements (from 12% to 63%) compared to other existent naive approaches (see [9]). Registers are assumed to have a zero latency. The fact that multiplexers and wiring latencies are neglected could be easily overcome by multiplying the latency of FUs by an empirical factor, as hwon in [13].

The area of multiplexers in UWL systems is only affected by the data wordlengths, which set the multiplexers' sizes, and by the number of different data sources (e.g. registers or FUs), which determines the multiplexers' width. However, if the wordlengths are optimized, that is, if a multiple wordlength (MWL) approach is followed, the area of multiplexers depends on the wordlengths of their different inputs and also on the coding that relates multiplexers' control signals and the inputs themselves. A lower bound on the area of a multiplexer is given by [9].

$$\underline{A_{MUX}} = \frac{1}{4} \sum_{i=0}^{N-1} (n_i + 1) \text{ slices}, \quad (6)$$

where N is the maximum wordlength of the mux's input signals and n_i is the wordlength of signal i.

4. SIMULATED ANNEALING BASED APPROACH

An optimal analysis based on MILP of the combined synthesis was carried out in [6]. Even though the problem was simplified to reduce the complexity of the optimal analysis, some fundamental conclusions could be extracted. These conclusions are the basis for the development of a heuristic able to address the combination of WLO and AS efficiently:

- The combination of WLO and AS as a single task produces area reduction.
- The area improvements happen sporadically when some particular combinations of noise and latency constraints occur.
- The following heuristic rules are derived:
  - A shared resource performing operations on sizes smaller than the resource size can be used to reduce noise by increasing all operations' sizes to the maximum permitted. An increase in wordlength (size) produces a decrease in noise, which allows a possible size reduction (that still may show up as noise increase in further optimization steps.

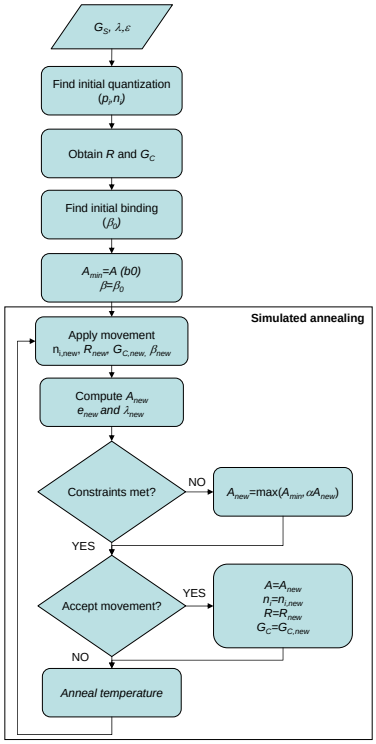


Figure 1: Optimization procedure.

  - A shared resource performing operations on sizes smaller than the resource size can be used to reduce area by setting all operations' sizes to the maximum permitted size minus one unit, thus reducing the size of the shared resource.
  - A combination of the two previous rules.
  - Simultaneous increase and/or decrease of the size of several resources.

All these conclusions and rules have been applied in the development of the optimization procedure based on SA presented here (see Fig.1).

Fig. 1 displays the precision-wise architectural synthesis process. Its inputs are the sequencing graph G_S, the error constraint ε and the total latency constraint λ. Graphs R and G_c are extracted from G_S. First, scaling is carried out and the values of p_i are fixed for each signal. Then a UWL WLO is applied and the initial sets of n and n_{pre} are obtained. With this information, the mapping β_0 that would produce the fastest parallel implementation is selected as the initial resource binding. Then, SA is used to drive the optimization procedure, where *movements* are applied to the current binding β and the sets of n_i and $n_{pre,i}$. The following movements are supported:

- AS movements
  - M^A_{HLS}: Map an operation $o \in O$ to a non-mapped resource.
  - M^B_{HLS}: Map an operation o to another already mapped resource.
  - M^C_{HLS}: Swap the mapping of two compatible opera-

tions mapped to different resources.
  - M_A^{WLA}: Select a signal $s_i \in S \backslash in(V_F) \cup out(V_D)$ and increase n_i one unit .
- WLO movements
  - M_{WLA}^{B}: Select a signal $s_i \in S \backslash in(V_F) \cup out(V_D)$ and decrease n_i one unit .
  - M_{WLA}^{C}: Select a resource $r \in R_{FU}$ and increase one unit the *size* of all operations bound to it that do not increase the original size of the resource.
  - M_{WLA}^{D}: Select two resources $r_1, r_2 \in R_{FU}$. Apply movement M_{WLA}^{C} to r_1 and r_2. Reduce one unit the size of operations bound to r_2 that meet the original size of the resource.

As in any SA approach, each time a movement is performed, the resulting area A as well as the output error and the total latency of the current solution are computed. If the noise constraint is met and the area is smaller than the current minimum area the movement is accepted. If the area is greater than the minimum so far, the movement is accepted with a certain probability that decays with time. Movements that do not comply with the noise or latency constraints can be accepted with a very low probability, thus enabling a wider design space exploration [9]. This is carried out by means of a penalty factor applied to the current area A.

The computation of A is performed after specific wordlength-wise algorithm for scheduling, register binding and multiplexer selection [9]. Scheduling is based on a iterative list-based algorithm that obtain the minimum area solution considering the lower and upper bounds on the number of instances of each resource. Register binding relies on a wordlength-wise modification of the left-corner algorithm. Multiplexer selection is also a novel wordlength-wise algorithm that tries to minimize sign extension in order to reduce multiplexers and registers area. Check [9] for a more detailed explanation. It worth mentioning that every time the quantization change, R and G_C, and therefore A, must be recomputed.

This method provides a robust way to perform the tasks of scheduling, resource allocation, resource binding and wordlength optimization simultaneously. One of its main advantages is that the resource set can be as complex as desired, including variable-latency resources, multiplexers and registers [9].

5. RESULTS

The following benchmarks are used for the analysis:

- ITU RGB to YCrCb converter (*RGB*).
- 3rd-order lattice filter (LAT_3).
- 4th-order IIR filter (IIR_4).
- 8-th order linear-phase FIR filter (FIR_8).
- 4-point DCT transform (DCT_4 [14])
- 8-point DCT transform (DCT_8 [14])

All algorithms are assigned 8-bit inputs and 12-bit constant coefficients. The algorithm implementations have been tested under different latency and output noise constraint scenarios assuming a system clock of 125 MHz. In particular, the noise constraints were $\sigma^2 = \{10^{-k}, 10^{-(k+1)}, 10^{-(k+2)}\}$, where k is the minimum number that makes 10^{-k} as close as possible to the variance of the output quantization if the output is quantized to 8 bits ($n = 7$). The target devices belong to the Xilinx Virtex-II family and Xilinx XST v.9.2 tool [15] was used to extract the resource model.

For all noise-latency scenarios the benchmarks were implemented using both a sequential approach and the proposed combined approach. In the sequential approach, first a SA-based WLO was applied ([12]), followed by a SA-based AS [9].

Fig. 2-a plots the area improvements of DCT_4 obtained by the combined approach for three different noise constraints and different algorithm latencies. The latencies range from λ_{min}, which is the minimum achievable algorithm latency, to $\lambda_{min} + 10$, and it is expressed in the graph using the latency offset $\Delta\lambda$ from the λ_{min}. The results show that area improvements are always obtained. However, there are only a few error/latency scenarios where the improvements are significant ($\sigma^2 = 10^{-3}$, $\Delta\lambda = \{1,3\}$; $\sigma^2 = 10^{-4}$, $\Delta\lambda = [5,10]$; $\sigma^2 = 10^{-5}$, $\Delta\lambda = [4,10]$) since they range from 6.6% to 9.6%. This situation recalls that of the optimal case [6], where improvements did not happen all the time. However, now there is an improvement for the majority of cases. This is due to the fact that the design space is now larger. Also, the latencies of resources are now dependent on the wordlengths, which adds new optimization possibilities.

Fig. 2-b displays the DTC_4 detailed resource distribution for $\sigma^2 = 10^{-4}$ and $\Delta\lambda = 1$ ($\lambda = 5$). The area required for functional units (FU), FU's multiplexers ($MUX - FU$), registers (REG) and registers' multiplexers ($MUX - REG$) for the sequential and combined implementations are shown. The accumulation of the area is the total area of the implementation. In this case, the improvements of 9% obtaind by the combined approach are mainly due to a reduction in FUs' multiplexers and FUs.

The graph on Fig. 3 depicts the overall results regarding the combined approach. For each quantization scenario the latency ranges from λ_{min} to $\lambda_{min} + 10$, and the mean and maximum values of the area improvements obtained by the combined implementations in comparison to the sequential implementations are computed. The improvements suggest that there is a tendency to obtain better results as the complexity (i.e. number of operations) of the algorithms increases. Summarizing, are improvements are obtained in most cases, with an overall mean improvement of 3.93% and a maximum area improvement of 21.32% (DCT_8). These should be considered as very good results, since they describe improvements over highly optimized results obtained from quasi-optimal sequential WLO and AS.

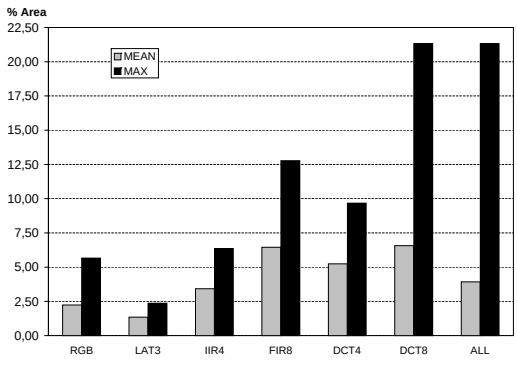
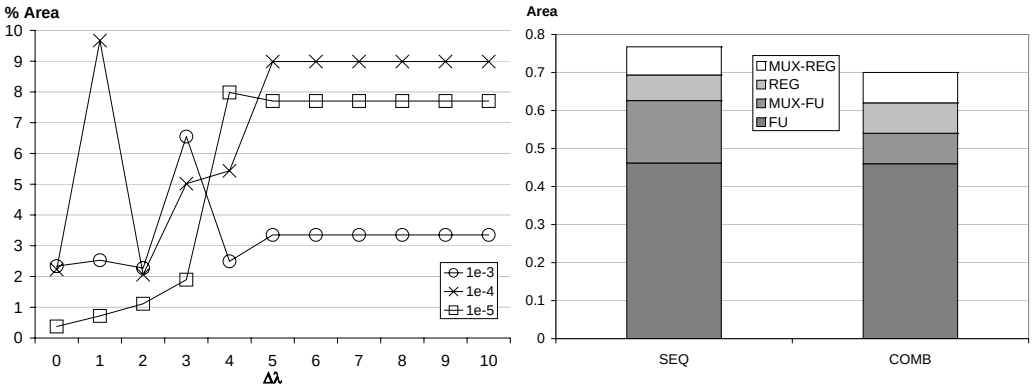


Figure 3: Overall area improvement for all benchmarks.



(a) $\sigma^2 = 10^{-5}$ (b) $\sigma^2 = 10^{-4}$, $\Delta\lambda = 1(\lambda = 5)$

Figure 2: Implementation results for DCT_4: (a) area improvements vs. latency curves; (b) area resource distribution (normalized with respect to XC2V40 device).

6. CONCLUSIONS

The combination of WLO and AS have been addressed by presenting a simulated annealing based approach is presented that deals with the complex optimization process involved. The proposal is capable of generating an architecture that meets both latency and error constraints, having as input a signal flow graph representation of a infinite-precision DSP algorithm. The architectural optimization accounts for datapaths composed of functional units, registers and multiplexers. Also, the fixed-point format of signals is determined during the optimization process.

The results show that, in most cases, the combined approach improve the area cost, with area improvements up to 21% when compared to the traditional approach.

Future work will involve the extension of the combined approach to include DSP embedded blocks.

7. ACKNOWLEDGMENT

This work was supported by Research Project TEC2009-14219-C03-02 (Spanish Ministry of Science and Innovation) and by the Universidad CEU-San Pablo.

REFERENCES

[1] K. I. Kum and W. Sung, "Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems," *IEEE Trans. Circuits Syst.*, vol. 20, pp. 921–930, Aug. 2001.

[2] G. Caffarena, A. Fernandez, C. Carreras, and O. Nieto-Taladriz, "Fixed-Point Refinement of OFDM-Based Adaptive Equalizers: A Heuristic Approach," in *Proc. EUSIPCO*, pp. 1353–1356, 2004.

[3] G. De Michelli, *Synthesis and Optimization of Digital Circuits*. Series in Electrical and Computer Engineering, New York: McGraw-Hill, 1994.

[4] D. Menard, R. Rocher, P. Scalart, and O. Sentieys, "SQNR Determination in Non-Linear and Non-Recursive Fixed-Point Systems," in *Proc. EUSIPCO*, pp. 1349–1352, 2004.

[5] S. Wadekar and A. Parker, "Accuracy Sensitive Word-Length Selection for Algorithm Optimization," in *Int. Conf. on Comp. Design*, pp. 54–61, 1998.

[6] G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal Combined Word-Length Allocation and Architectural Synthesis of Digital Signal Processing Circuits," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 5, pp. 339–343, 2006.

[7] N. Herve, D. Menard, and O. Sentieys, "About the Importance of Operation Grouping Procedures for Multiple Word-Length Architecture Optimizations," in *Int. Workshop on Applied Reconfigurable Computing*, pp. 107–122, 2007.

[8] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength Optimization for Linear Digital Signal Processing," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 10, pp. 1432–1442, 2003.

[9] G. Caffarena, J. López, G. Leyva, C. Carreras, and O. Nieto-Taladriz, "Architectural Synthesis of Fixed-Point DSP Datapaths Using FPGAs," *Int. J. of Reconfigurable Computing*, vol. 2009, pp. 1–14, 2009.

[10] G. Constantinides, P. Cheung, and W. Luk, "Heuristic Datapath Allocation for Multiple Wordlength Systems," in *Design, Automation and Test in Europe*, pp. 791–796, 2001.

[11] G. Constantinides, P. Cheung, and W. Luk, "Truncation Noise in Fixed-Point SFGs," *IEE Electronics Letters*, vol. 35, no. 23, pp. 2012–2014, 1999.

[12] G. Caffarena, *Combined Word-Length Allocation and High-Level Synthesis of Digital Signal Processing Circuits*. PhD thesis, Universidad Politécnica de Madrid, http://oa.upm.es/1822/, 2008.

[13] R. Enzler, T. Jeger, D. Cottet, and G. Tröster, "High-Level Area and Performance Estimation of Hardware Building Blocks on FPGAs," in *Proc. FPL*, pp. 525–534, 2000.

[14] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[15] Xilinx Inc. http://www.xilinx.com.