Optimized Architectural Synthesis of Fixed-Point Datapaths

Gabriel Caffarena[1], Juan A. López[1], Gerardo Leyva[2]
Carlos Carreras[1] and Octavio Nieto-Taladriz[1]
[1]Dep. de Ing. Electrónica, Universidad Politécnica de Madrid, Spain
[2]Dep. de Sistemas Electrónicos, Universidad Autónoma de Aguascalientes, Mexico
e-mail: gabriel@die.upm.es

Abstract

In this paper we address the time-constrained architectural synthesis of fixed-point DSP algorithms using FPGA devices. Optimized fixed-point implementations are obtained by means of considering: (i) a multiple wordlength approach; (ii) a complete datapath formed of wordlength-wise resources (i.e. functional units, multiplexers and registers); and, (iii) a novel resource usage metric that enables the wise distribution of logic fabric and embedded DSP resources.

The paper shows: (i) the benefits of applying a multiple wordlength approach to the implementation of fixed-point datapaths; and (ii) the benefits of a wise use of embedded FPGA resources. The proposed metric enables area improvements up to 54% and the use of a complete fixed-point datapath leads to improvements up to 35%.

1. Introduction

In this paper we deal with the architectural synthesis (AS) of common Digital Signal Processing (DSP) cores implemented on modern FPGAs. Two important factors leading to optimization are the use of Multiple Word-Length (MWL) fixed-point descriptions of the algorithms and the use of both LUT-based and embedded FPGA resources. The former reduces implementation cost notably, and the latter minimizes area and inproves performance in FPGA implementations.

The MWL implementation of fixed-point DSP algorithms [8, 5, 4, 2] has proved to provide significant cost savings when compared to the traditional uniform word-length (UWL) design approach. The introduction of MWL issues this in AS, although it increases optimization complexity, can lead to significant cost reductions [5, 4, 3].

FPGA devices have been extensively used in the implementation of DSP algorithms, especially due to the recent introduction of embedded blocks (i.e. memory blocks, DSP blocks, etc.). Traditional approaches to estimate FPGA resource usage do not apply to heterogeneous-architecture FPGAs since they only account for lookup table (LUT) based resources [10]. This situation calls for new resource usage metrics that can be integrated as part of automatic optimization techniques (i.e. architectural synthesis) to fully exploit the possibilities that embedded resources offer [1, 9].

The main contributions of this paper are: (i) the presentation of a novel resource usage metric that allows minimum resource usage for heterogeneous-FPGA implementations; (ii) the presentation of an architectural synthesis procedure tuned to fixed-point implementations, that handles a complete datapath (FUs, multiplexers and registers); and, (iii) a novel strategy for fixed-point data multiplexing.

The paper is divided as follows: In section 2, the architectural synthesis of DSP cores using multiple wordlength systems and modern FPGAs is introduced. Section 3 deals with the implementation results from synthesizing several well-known DSP benchmarks for different latency constraints and output noise constraints. Finally, in section 4, conclusions are drawn.

2. Generation of Fixed-Point Datapaths

2.1. Formal description

This work focuses on the time constrained resource minimization problem [6]. The notation used is based on [6], and it is similar to that in [5, 2, 3].

Given a sequencing graph $G_S(V, S)$, a maximum latency λ 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 AS to find the time step when each operation is executed (*scheduling*), the types and number

978-0-7695-3474-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ReConFig.2008.48

of resources forming R (*resource allocation*), and the binding between operations and variables to functional units 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 determine 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), where 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 [2].

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 functional units $R_{FU} = R_{A_{LUT}} \cup R_{M_{LUT}} \cup R_{M_{EMB}}$ is composed of LUT-based adders, LUT-based generic multipliers, and embedded multipliers. This set of FUs covers a representative set of modern FPGA devices. An FU $r \in R_{FU}$ is defined by its type $type(r) = \{Adder_{LUT}, Multiplier_{LUT}, Multiplier_{EMB}\}$ 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 [2, 3].

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 $G_C(V \cup R, C)$ [5], 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.

2.2. Handling resource heterogeneity

The recent appearance of specialized blocks in FPGAs calls for new design methods to efficiently exploit their advantages. In [1], it is proposed to use a normalized resource usage vector. Given an FPGA with M different types of resources R_i ($i = 0 \cdots M-1$), each type with a maximum number of $|R_i|$ resources, the resource requirements of a particular design implementation d can be expressed as the following normalized area vector:

$$\hat{\boldsymbol{A}} \equiv \left\langle \frac{\#r_0}{|R_0|}, \frac{\#r_1}{|R_1|}, \cdots, \frac{\#r_{M-1}}{|R_{M-1}|} \right\rangle \tag{1}$$

where $\#r_i$ is the number of resources of type R_i used. Two useful norms are the ∞-norm and the 1-norm:

$$\left\| \hat{\boldsymbol{A}} \right\|_\infty = \max \left\{ \frac{\#r_0}{|R_0|}, \frac{\#r_1}{|R_1|}, \cdots, \frac{\#r_{M-1}}{|R_{M-1}|} \right\} , \tag{2}$$

$$\left\| \hat{\boldsymbol{A}} \right\|_1 = \sum_{i=0}^{M-1} \frac{\#r_i}{|R_i|} . \tag{3}$$

The inverse of ∞-norm represents the number of times that the same implementation of design d can be replicated within the FPGA device (see [1]), and the 1-norm gives information about the overall resource usage of the implementation. Each norm is interesting on their own, but they also have some pitfalls. On the one hand, if two implementations have the same ∞-norm this implies that they can be replicated the same number of times, but there is no way to know which implementation requires less resources. On the other hand, the 1-norm can tell if a design implementation requires less resources than other, but that does not guarantee that the implementation with less resources can be replicated more times than the other. We propose a linear combination of ∞-norm and 1-norm, called $+$-norm (*plus-norm*), that has all the benefits of the norms but none of the drawbacks:

$$\left\| \hat{\boldsymbol{A}} \right\|_+ = K \cdot \left\| \hat{\boldsymbol{A}} \right\|_\infty + \left\| \hat{\boldsymbol{A}} \right\|_1 . \tag{4}$$

The value of constant K can be obtained from the parameters M and R_i. If

$$K > (M-1) \cdot (\max(|R_i|)) - M, \tag{5}$$

then, it is guaranteed that for any two implementations d_i and d_j: (a) if $\|\boldsymbol{A_i}\|_+ < \|\boldsymbol{A_j}\|_+$ then d_i can be replicated more times than d_j; and, (b) if $\|\boldsymbol{A_i}\|_+ \leq \|\boldsymbol{A_j}\|_+$

then d_i can be replicated more times than d_j, or the same number of times consuming less resources. Therefore, minimizing +-norm implies that the design can be replicated within the FPGA the maximum possible number of times while using the minimum possible number of resources.

The metric +-norm has a low computational cost and it is suitable for integer linear programming approaches [2] and heuristic approaches [3].

2.3 Resource modeling

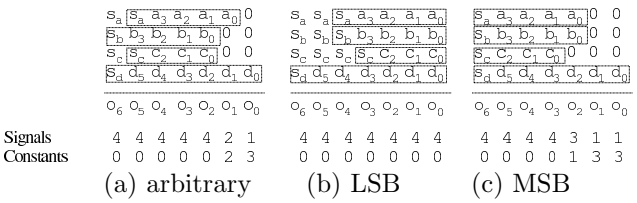
Resources are divided into three types: functional units (R_{FU}), registers (R_{REG}) and steering logic (R_{MUX}). The area and latency of FUs and registers (i.e. $\boldsymbol{A}(r)$ and $l(r)$) are expressed as functions of the input and output wordlength information (p and n). They are obtained by applying curve fitting to hundreds of synthesis results. The use of accurate delay cost functions was proved to provided significant performance improvements (from 12% to 63%) compared to other existent naive approaches (see [3]). Registers are assumed to have a zero latency. Note that $\boldsymbol{A}$ is a vector with as many components as types of FPGA resources. The fact that multiplexers and wiring latencies are neglected could be easily overcome by multiplying the latency of FUs by an empirical factor [7].

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 multiplexer width. An estimation of the area of an N-input multiplexer of wordlength M for Virtex-II devices is given by

$$A_{MUX} = M \cdot N/4 \quad \text{slices.} \tag{6}$$

In MWL systems, data must be aligned before being processed by FUs or stored in registers. In [11] the problem of data alignment and multiplexing is tackled by means of alignment blocks introduced before multiplexers. In this work, multiplexers are used for both data multiplexing and data alignment, since the combination of these two tasks leads to a reduction in the number of control signals, and therefore, control logic. In addition, the chances for logic optimization are greater than if two separate blocks (an alignment block and a multiplexer) are used.

Fig. 1 presents three different types of alignments for a 4-input multiplexer with inputs signals s_a, s_b, s_c and s_d and output o: arbitrary alignment (Fig. 1(a)), LSB alignment (Fig. 1(b)), and MSB alignment (Fig. 1(c)). Note that, on one hand, sign extension (Fig. 1(a) and



(a) arbitrary (b) LSB (c) MSB

Figure 1. Signal alignment

Fig. 1(b)) does not offer any opportunity for logic optimization, since the signs bits must be multiplexed. On the other hand, zero padding (Fig. 1(a) and Fig. 1(c)) does offer it, due to the reduction in the number of signals and the introduction of constant bits (zeros) that can be hard-wired into the multiplexer logic. In fact, it is MSB alignment (Fig. 1(c)) the option that allows greater logic reduction. Therefore, it is proposed to apply this alignment whenever possible.

A lower bound on the multiplexers' area if the MSB alignment is adopted can be computed as

$$\underline{A_{MUX}} = \frac{1}{4}\sum_{i=0}^{N-1}(n_i + 1) \text{ slices,} \tag{7}$$

where N is the maximum wordlength present and n_i is the wordlength of signal i.

2.4. Datapath automatic synthesis

The optimization procedure is based on the use of Simulated Annealing (SA). The inputs to the optimizer are the sequencing graph $G_S(V, S)$ and the total latency constraint λ, from which it is possible to extract the set of resources R and the compatibility graph G_C. The optimization is based on changing the current FU binding (β) and computing the area of the datapath. The *movements* are the following:

- M_A: Bind an operation $v \in V$ to a non-used resource $r \in R_{FU}$.
- M_B: Bind an operation $v \in V$ to another already used resource $r \in R_{FU}$.
- M_C: Swap the binding of two compatible operations v_1 and v_2 mapped to different resources r_1 and r_2.

Every time a movement is produced, the resulting area, expressed as in (4), is obtained by, first, a list-based resource-constraint scheduling tuned to MWL systems [3], and after that, register binding based on the left-edge algorithm [6] and multiplexers generation based on a MSB-alignment.

MWL issues are handled automatically due to the wordlength-wise cost modeling of resources, which have wordlength-dependent area and latency costs . FPGA resource heterogenity is handled by means of the use of +-norm, which enables the automatic selection of LUT-based resources and embedded resources, without adding any additional step to the SA approach. This method provides a robust way to obtain optimized implementations of DSP algorithms using FPGAs.

3. Results

The following benchmarks are used for the analysis: (i) an ITU RGB to YCrCb converter (*RGB*); (ii) a 3rd-order lattice filter (LAT_3); (iii) a 4th-order IIR filter (IIR_4); and (iv) an 8-th order linear-phase FIR filter (FIR_8). 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 quantization noise that would present the output of the benchmark if quantized to 8 bits ($n = 7$).

The target devices belong to the Xilinx Virtex-II family. The area results are normalized with respect to the XC2V40 device (256 slices, 4 embedded 18x18 multipliers) and expressed according to Eqn. 2. For instance, $\left\|\hat{\boldsymbol{A}}\right\|_\infty \leq 1$ implies that the device XC2V40 is the smallest-cost device able to hold the design, whereas $1 < \left\|\hat{\boldsymbol{A}}\right\|_\infty \leq 2$ implies that XC2V80 is the smallest-cost device, and so on. The datapath allow resource sharing – for both adders and multipliers – and there are generic LUT-based multipliers ($R_{M_{LUT}}$) as well as and 18x18 and 36x18 generic multipliers (forming $R_{M_{EMB}}$).

Before AS, each algorithm is translated to a fixed-point specification by means of two wordlength optimization procedures, that follow an UWL approach and an MWL approach, respectively. Once the fixed-point signals formats are available, AS is applied to each possible combination of latency, quantization scenario and FPGA architecture (homogeneous or heterogeneous) for a total of 120 implementations per benchmark).

3.1. Homogeneous UWL vs. MWL

Fig. 2(a) contains the area vs. latency curves of the homogeneous implementations of IIR_4 (*MWL-HOM* and *UWL-HOM*, grey symbols). The latency ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{UWL-HOM} + 10$. It can be seen how both the UWL and MWL areas decrease as the latency increases, which is expected since the chances for FU reuse increases. The area improvements obtained by means of using an MWL approach are up to 65%. Also, the minimum latency that each implementation achieves differs considerably.

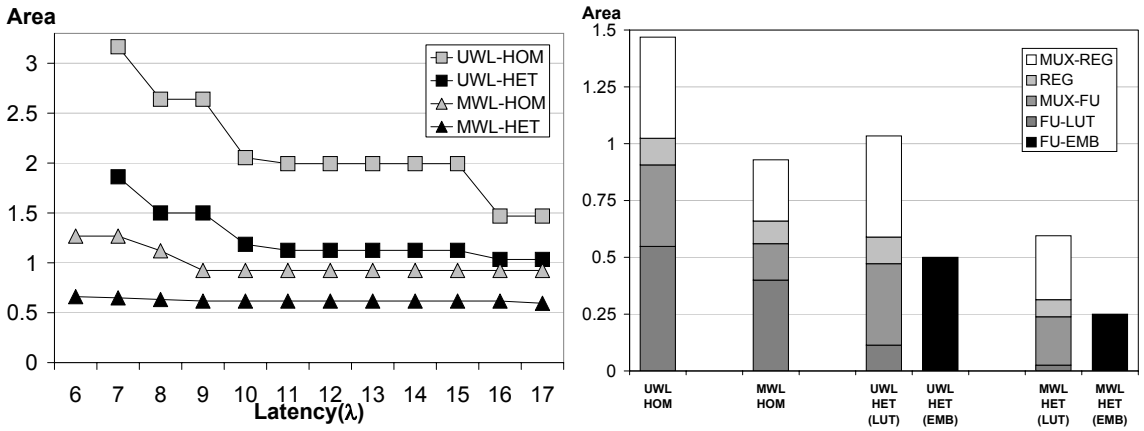
Fig. 2(b) displays the detailed resource distribution for the IIR_4 UWL and MWL implementations corresponding to $\sigma^2 = 10^{-5}$ and $\lambda = 17$ (see *UWL-HOM* and *MWL-HOM* bars). The separate resource usages of FUs (*FU-LUT* and *FU-EMB*), multiplexers (*MUX-FU* and *MUX-REG*) and registers (REG) are displayed. The overall area saving is 36%, and it is due to the fact that the wordlengths of the majority of signals, which impact on FUs, multiplexers and registers, have been highly reduced.It is important to highlight that the area due to multiplexers and registers, although smaller than the FUs' area, makes up a significant part of the total area (37% for UWL and 43% for MWL). Hence the importance of including these costs within the optimization loop.

The graph on the left of Fig. 3 depicts the overall results regarding homogeneous implementations. For each quantization scenario the latency ranges from $\lambda_{min}^{UWL-HOM}$ to $\lambda_{min}^{UWL-HOM} + 10$, and the mean and maximum values of the area improvements obtained by the MWL implementations in comparison to the UWL implementations are computed. The area improvements obtained are remarkable: *RGB* obtains a mean improvement of 66%; LAT_3 of 41%; IIR_4 of 47%; FIR_8 of 30%. Note that the mean improvements obtained for all benchmarks are relatively close to the maximum value. The overall average improvement obtained is 46% and the maximum achieved is 77%. Regarding latency, the minimum latency achievable by UWL implementations is reduced in average 22%. The results clearly show that an MWL AS approach achieves significant area reductions.

3.2. UWL vs. MWL: Heterogeneous case

The curves in Fig. 2(a) – *UWL-HET* and *MWL-HET*, black symbols – yield that, again, there is a significant gain in using an MWL approach, since the improvements are up to 65%. Fig. 2(b) (*UWL-HET* vs *MWL-HET*) shows that the improvements are mainly due to an overall wordlength reduction. Also, it can be seen that the LUT-based resources are almost entirely devoted to data storing and multiplexing.

The central graph in Fig. 3 holds the results regarding heterogeneous implementations. For each quanti-



(a) $\sigma^2 = 10^{-5}$ (b) $\sigma^2 = 10^{-5}$, $\lambda = 17$

Figure 2. Implementation results for IIR_4: (a) area vs. latency curves; (b) resource distribution.

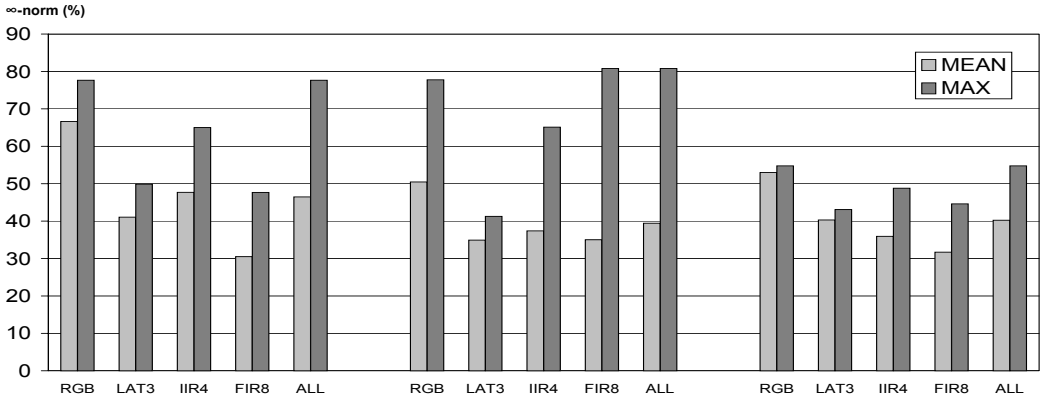


Figure 3. Overall resource usage improvement for all benchmarks.

zation scenario, the latency ranges from $\lambda_{min}^{UWL-HET}$ to $\lambda_{min}^{UWL-HET} + 10$, and the mean and maximum values of the area improvements obtained by the MWL implementations in comparison to the UWL implementations are presented. The area improvements obtained are, again, remarkable: RGB obtains a mean improvement of 50%; LAT_3 of 34%; IIR_4 of 37%; FIR_8 of 35%. The average improvement obtained is 39% and the maximum achieved is 81%. The latency analysis throws that the minimum UWL latency is reduced an average 19%. Again, an MWL AS approach achieves significant area reductions for heterogeneous implementations.

3.3. Homogeneous vs. Heterogeneous

Fig. 2(a) – *MWL-HOM* and *MWL-HET*, triangles – and Fig. 2(b) –*MWL-HOM* and *MWL-HET*– contain the MWL vs. UWL heterogeneous implementation results for IIR_4. The area vs. latency curves show that the introduction of embedded multipliers dramatically reduces the overall resource usage with improvements of up to 65%. Fig. 2(b) indicates that the improvements are mainly due to a migration of LUT-based multipliers to embedded multipliers.

The graph on the right in Fig. 3 holds the results regarding heterogeneous vs. homogeneous implementations. For each quantization scenario, the latency ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HOM} + 10$, and the mean and maximum values of the area improve-

ments obtained by the heterogeneous implementations in contrast to the homogeneous implementations are presented. The area improvements obtained are remarkable: RGB obtains a mean improvement of 52%; LAT_3 of 40%; IIR_4 of 35%; FIR_8 of 31%. Note that the mean improvements obtained for all benchmarks are quite close to the maximum values. The average improvement obtained is 40% and the maximum is 55%. The results clearly show that a wise use of embedded resources leads to highly optimized datapath implementations. Regarding latencies, the minimum latency achievable by both kinds of implementations is the same for the experiments performed. This is due to the fact that the latency of resources are very similar in the particular conditions used for the tests. The same experiments presented in this section were repeated increasing the constant wordlength to 16 bits, obtaining that heterogeneous implementations reduced 7% the minimum latency of homogeous implementations.

Summarizing, the efficient use of embedded resources highly improves the final implementation results.

3.4. Effect of using a complete datapath

All these experiments were repeated excluding registers and multiplexers from the available resources during the optimization process, and adding them afterward to compute the final datapath area. The homogeneous implementations were degraded up to 5%, while the heterogeneous up to 35%. It can be concluded that the use of a complete datapath description is specially significant in heterogeneous implementations, since they are strongly based on optimizing the balance between LUT-based and embedded resources.

4. Conclusions

In this paper an architectural synthesis approach able to produce optimized fixed-point implementations using modern FPGA devices is presented. The key to success is provided by the use of highly accurate models of the datapath resources, a complete datapath resource set that includes multiplexer and registers, a novel method to handle fixed-point data alignment and multiplexing, and also the introduction of a novel novel resource usage metric that can cope with LUT-based and embedded FPGA resources.

The AS procedure produces area improvements of up to 80% when compared to uniform-wordlength implementations, and minimum latency improvements of up to 22%. The efficient use of embedded resources achieves area improvements of up to 54% when compared to homogeneous implementations. Also, the inefficiency of current FPGA architectures to implement data steering was exposed.

These results are intented to be further improved by means of including the fixed-point refinement process as part of the architectural synthesis [2].

5. Acknowledgment

This work was supported by the Spanish Ministry of Education and Science under Research Project TEC2006-13067-C03-03.

References

[1] C.-S. Bouganis, G. Constantinides, and P. Cheung. A Novel 2D Filter Design Methodology for Heterogeneous Devices. In *Proc. FCCM*, pages 13–22, 2005.

[2] 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*, 53(5):339–343, May 2006.

[3] G. Caffarena, J. A. López, C. Carreras, and O. Nieto-Taladriz. High-Level Synthesis of Multiple Word-Length DSP Algorithms using Heterogeneous-Resource FPGAs. In *Proc. FPL*, pages 675–678, Madrid, Spain, 2006.

[4] J. Cong, Y. Fan, G. Han, Y. Lin, J. Xu, Z. Zhang, and X. Cheng. Bitwidth-Aware Scheduling and Binding in High-Level Synthesis. In *Proc. ASP-DAC*, pages 856–861, 2005.

[5] G. Constantinides, P. Cheung, and W. Luk. Heuristic Datapath Allocation for Multiple Wordlength Systems. In *Proc. DATE*, pages 791–796, 2001.

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

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

[8] K. I. Kum and W. Sung. Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems. *IEEE Trans. Circuits Syst.*, 20(8):921–930, Aug. 2001.

[9] X. Liang, J. Vetter, M. Smith, and A. Bland. Balancing FPGA Resource Utilities. In *Proc. ERSA*, pages 156–162, 2005.

[10] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee. Accurate Area and Delay Estimators for FPGAs. In *Proc. DAC*, pages 862–869, 2002.

[11] K. Schoofs, G. Goossens, and H. De Man. Bit-Alignment in Hardware Allocation for Multiplexed DSP Architectures. In *Proc. DATE*, pages 289–293, 1993.