FPGA-Based FIR Filters using Distributed Arithmetic

M. Martinez-Peiro et al - February 04, 2002

ABOUT THE AUTHORS

M. Martinez-Peiro, R. Colom, F. Ballester, and R. Gadea are teachers with a focus on digital applications at the Technical University of Valencia, Spain. They are currently working on video compression, Custom DSP, HDL simulators, digital filtering, and image lifting.

This paper discusses several bit-serial, high-order implementations of cascade, lattice and direct-form FIR filters based on Distributed Arithmetic (DA). Three types of filters are described using an Altera 10K50 FPGA with analysis done on the filters' area and speed parameters. Results show that we can implement a 60th-order bit-serial cascade and direct-form implementation running at nearly 4 MHz, and a 40th-order lattice structure running at 7.5MHz. The lattice filter also has a lower quantification error.

Introduction
Distributed Arithmetic (DA) is a well-known method to save resources in multiply-and-accumulate structures (MACs) implementing DSP functions. DA trades memory for combinatory elements, resulting in ideal-to-implement custom DSPs (CDSPs) in LUT-based FPGAs. In addition to a DA implementation, the designer also can select from a bit-serial to a full-parallel implementation to trade bandwidth for resource utilization.

Cascade and lattice structures present several interesting properties such as low quantification error and high-stability in the filter coefficients. Moreover, you can expand lattice cells without a full redesign. The goal of this article is to implement FPGA-based direct-form, cascade, and lattice high-order FIR filters using bit-serial DA. We start by comparing the resultant topologies in both area and speed. The designs use an HDL to include pipeline techniques and scalable parameters. We
also describe DA error models of the three structures. The next section reviews DA fundamentals and proposed architectures for each kind of filter. This article also presents the results of the FPGA implementation of the structures and discusses an error model for the filter structures.

Distributed Arithmetic Structures

Equation 2 expresses FIR filter operation (Equation 1) using the 2’s complement representation of the x[n] input samples of N bits.

\[
y = \sum_{t=1}^{T} A_t x_t
\]

(1)

\[
y = \sum_{t=1}^{T} A_t (-x_0 + \sum_{n=1}^{N-1} x_{t,n} 2^{-n}) =
-\sum_{t=1}^{T} A_t x_0 + \sum_{n=1}^{N-1} \left[ \sum_{t=1}^{T} A_t x_{t,n} 2^{-n} \right]
\]

(2)

You can pre-calculate the terms in brackets in Equation 2, save the results in memory, and address these terms by \(x_{t,n}\) in Table 1. Considering that each \(x_{t,n}\) can only take two values (0 or 1), each product term reaches one of the \(2^{(N-1)}\) possible values.

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_{t,N-1}) ... (x_{t,2}) (x_{t,1}) (x_{t,0})</td>
<td>(0)</td>
</tr>
<tr>
<td>0 ... 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 ... 0 0 1</td>
<td>(A_i)</td>
</tr>
<tr>
<td>0 ... 0 1 0</td>
<td>(A_2)</td>
</tr>
<tr>
<td>0 ... 0 1 1</td>
<td>(A_2 + A_i)</td>
</tr>
<tr>
<td>1 ... 1 1 1</td>
<td>(A_T + A_2 + A_i)</td>
</tr>
</tbody>
</table>

Table 1: Distributed Arithmetic pre-calculated terms

DA Direct Form Implementation

A DA bit-serial implementation of a FIR filter addresses each product term once per bit (the MSB bit is the sign bit). After obtaining the last product-term, this term is added, with its appropriate shift, to the rest of the product term previously added.
Figure 1 shows the structure representing the direct-form FIR filter. Considering that the FPGA this paper discusses has four-inputs LUTs, the product-terms larger than four need to be divided into $r$ parts such that $4T/r$, where $T$ is the number of taps of the filter. In other words, the adders in the tree structure add the $r$ LUT outputs. Eventually, you need a shift accumulator to add and shift each product term. Figure 1 represents a bit-serial implementation of a filter with samples of 8 bits. The output of the filter occurs each eight clock cycles. When the sign-bit arrives, a subtraction instead of an addition in the shift-accumulator is done. By using carry save adders before the LUT, you can implement a symmetrical filter\footnote{detailed information of this operation is found in Croisier and Proakis\cite{1,4}}.

Additionally, you can extend the range of processing speed by pipelining the structure. Equation 3 expresses the operation frequency ($f_s$), where $L$ is the latency and $n$ the number of the bits of each input sample.

$$f_s = \frac{f_{ck}}{n + L} \quad (3)$$

Despite the increment of registers in the DA pipeline version, the final area resources increase slightly, due to the FPGA structure.
DA Cascade-Filter Implementation
You can factor a linear-phase FIR filter into several 4th order sections to obtain an area reduction. The structure of these sections can be DA adapted with the symmetry equation (Equation 4) that represents the kth section of a T-order filter. Equation 5 shows the expansion in DA product terms of Equation 4. Equation 5 represents the basic cell of a cascade structure you can design using a bit-serial approach (Figure 2).

\[ y^k(n) = c_0^k[x^k(n)+x^k(n-4)] + 
+ c_1^k[x^k(n-1)+x^k(n-3)]+c_2^k[x^k(n-2)] \]

\[ y^k = [c_0^k(x_{o,0} + x_{4,0}) + c_1^k(x_{1,0} + x_{3,0}) + c_2^k(x_{4,0})] + 
[+ c_5^k(x_{5,1} + x_{4,1}) + c_1^k(x_{1,1} + x_{3,1}) + c_2^k(x_{4,1})] \cdot 2^{-1} + ... 
.+ [c_0^k(x_{o,N-1} + x_{4,N-1}) + c_1^k(x_{1,N-1} + x_{3,N-1}) + 
+ c_2^k(x_{4,N-1})] \cdot 2^{-N-1} \]

Figure 2: Bit-serial DA cascade 4th-order cell

As a result of the latency generated by the pipeline, the cascade cell has extra registers (one per stage of pipeline) to synchronize the operation of the filter. Equation 6 represents the real-time frequency operation of the cascade structure.

\[ f_s = f_{clk}/(n+L) = f_{clk}/11 \]

DA Lattice-Filter Implementation
You use the recursive equations that describe the lattice cell structures (Equation 7) to obtain cascade implementations of M cells. Both the f and g terms represent the forward and the backward predictions respectively in a linear-predictive filter structure.
Using the DA equations of Equation 8, we can reproduce the f and g terms with two LUTs, where $g'$ represents the g(n-1) term.

\[
\begin{align*}
  f_m(n) &= f_{m-1}(n) + K_m g_{m-1}(n-1) \\
  g_m(n) &= K_m f_{m-1}(n) + g_{m-1}(n-1)
\end{align*}
\]  

(7)

\[
\begin{align*}
  f_m(n) &= -f_{m-1,0} + \sum_{n=1}^{N-1} f_{m-1,n} 2^{-n} + K_m [-g'_{m-1,0} + \sum_{n=1}^{N-1} g'_{m-1,n} 2^{-n}] \\
  g_m(n) &= K_m [-f_{m-1,0} + \sum_{n=1}^{N-1} f_{m-1,n} 2^{-n}] + [-g'_{m-1,0} + \sum_{n=1}^{N-1} g'_{m-1,n} 2^{-n}]
\end{align*}
\]  

(8)

Figure 3: Bit-serial DA lattice cell
**Figure 4:** Bit-serial improved lattice cell

**Figure 3** represents the formal DA implementation of Equation 8. However, in this article we also propose an improved structured in **Figure 4** that reduces the memory requirements of the DA lattice cell. With the structure of **Figure 4**, you only need to save the coefficient (km). You can get both the km+1 and 1 values from the carry input of the scaling accumulator.

FPGA Implementation

We implemented the previously described filters in an Altera EPF10K50RC240-3 FPGA using default placement-and-routing compilation options. We used the lpm_add_sub function from Altera. This parameterized component maps the pipelined carry-save adder. **Figures 5 and 6** show both the real-time frequency operation and the area of the three types of filters. They indicate that the lattice filter uses more area than the cascade and direct-form structures. However, the latter two options represent symmetrical versions that use approximately half the area of non-symmetrical implementations of the same structures.

**Figure 5:** Operational frequency characteristics of DA bit-serial filter implementations
The bit-serial implementation of the lattice structure achieves a real-time operation of 7.5 MHz. In the cascade and direct-form structure, filter operational frequency continuously decreases to 4 MHz as filter order increases.

Bit-Serial DA Error Model

This section proposes a DA error model of each structure under the assumption that the rounding error \( e[n] \) and the data input \( x[n] \) are non-correlated.

In DA bit-serial cascade structures (Figure 2), the error is modeled by Equation 9, where \( e_m \) and \( e_a \) (shaded red in Figure 2) are the LUT and shift-accumulator rounding errors.

\[
e_y = \sum_{i=0}^{p-1} 2^{-i} \cdot e_m + \sum_{i=0}^{p-2} 2^{-i} \cdot e_a
\]  

(9)

The variance of this error is expressed in Equation 10, where \( p_m \) and \( p_a \) are the number of bits in both memory and the shift-accumulator.

\[
\text{var}[e_{\text{cell}}] = \left( \frac{1 - 2^{-2p}}{1 - 2^{-2}} \right) \cdot \frac{2^{-2p_a}}{12} + \left( \frac{1 - 2^{-2(p-1)}}{1 - 2^{-2}} \right) \cdot \frac{2^{-2p_a}}{12}
\]  

(10)

Equation 11 represents a direct-form DA error model, where \( r \) is the number of data-sample partitions or memory partitions.

\[
e_y = \sum_{i=0}^{p-1} 2^{-i} \cdot \sum_{j=1}^{r} e_m + \sum_{i=0}^{p-2} 2^{-i} \cdot e_a
\]  

(11)

Furthermore, as a result of the four-input LUT structures, the partition of the memories in the FPGA case is limited by \( r < T/4 \) (T is the order of the filter).

Equation 12 shows the variance of the error in the direct-form structure.

\[
\text{var}(e_{df}) = \frac{K}{4} \cdot \left[ \frac{1 - 2^{-2p}}{1 - 2^{-2p/q}} \right] \cdot \text{var}(e_m) + \left[ \frac{1 - 2^{-2p(q-1)/q}}{1 - 2^{-2p/q}} \right] \cdot \text{var}(e_{acc})
\]  

(12)
Figure 4 shows the improved lattice cell with both error sources $e_m$ and $e_a$ shaded in red. Equation 13 shows the error and the variance in this cell:

$$
\epsilon_{\text{latt}} = \sum_{i=0}^{p-2} 2^{-i} \cdot \epsilon_{\text{acc}} \quad \text{var}(\epsilon_{\text{latt}}) = \left[ \frac{1 - 2^{-(p-1)}}{1 - 2^{-2}} \right] \cdot \frac{2^{-2}p_a}{12}
$$

As an example, we used a T-order FIR filter with $p=p_m=p_a=8$ bits to compare the three models. The results in the direct-form, cascade and lattice implementations are $T4.2384e-07+1.6953e-06$, $T3.3907e-06$, and $T2.5868e-11$, respectively. The lattice filter has the lowest error, while the cascade form has the highest error. Finally, the direct-form structure also has a high error compared with the lattice-cell structure.

Conclusions
We presented a distributed-arithmetic, bit-serial approach to implement the three classical structures of a FIR filter: direct-form, cascade, and lattice. Moreover, we developed a comparative DA error model of the three structures. Conclusions of our study are:

- We can implement a bit-serial 40th-order lattice filter in a 10K50 device with a real-time frequency operation of 7.5 MHz. The pipelined cascade and direct-form bit-serial implementations reach 4.5 MHz for a 60th-order structure in their symmetrical implementations.

- We have been presented a new improved lattice cell reduces memory usage by using the input carry in the shift-accumulator.

- We presented a DA error model that shows that the lattice structure represents the lowest rounding error while cascade structure has the highest error.