

A comparison between Lattice, Cascade and Direct-form FIR filter structures by using a FPGA bit-serial Distributed Arithmetic implementation.

Marcos M.Peiró*, Javier Valls*, T.Sansaloni*, A.P.Pascual* and E.I.Boemo**

*Dept. Ingeniería Electrónica Univ. Politécnica de Valencia,
Camino de Vera s/n 46071 Valencia, Spain. e-mail mpeiro@eln.upv.es.

**Escuela Técnica Superior de Ing. Informática. Univ. Autónoma de Madrid, Ctra. Colmenar Viejo Km.15,
28049 Madrid, Spain, e-mail: eduardo.boemo@ii.uam.es

Abstract

In this paper, several bit-serial, high-order implementations of cascade, lattice and direct-form FIR filters using Distributed Arithmetic (DA) are studied. Although lattice and cascade structures present many interesting properties related to quantification error and stability, they DA versions has not been thoroughly compared. Three types of filters with their particular bit-serial DA model error have been constructed using an ALTERA 10K50 FPGA and they area-time figure is analysed. Mains results show that a 60th order bit-serial cascade and direct-form implementation with nearly 4MHz and a 40th order lattice structure with 7.5MHz can be implemented. Moreover, in contrast to the first structures, the lattice filter presents the lower quantification error.

1. Introduction

Distributed Arithmetic (DA) is a well-known method [1,2] to save resources in multiply-and-accumulate structures (MAC) utilised to implement DSP functions. This arithmetic trades memory for combinatory elements, resulting ideal to implement custom DSP (CDSP) in LUT-based FPGAs [3]. 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 utilisation [4].

Cascade and lattice structures present several interesting properties such as low quantification error and high-stability in their coefficients. Moreover, lattice cells can be easily expanded without a full-redesign [5]. In this way, the goal of this paper is to implement FPGA based direct-form, cascade and lattice high-order FIR filters by using bit-serial DA. The resultant topologies are compared in both area and speed. Pipeline techniques and scalable parameters are included in the designs by using a hardware description

language. Finally, DA error models of the three previous structures are described. To the best of our knowledge, in the FPGA-related technical literature mainly DA direct form filters have been studied [6,7]. Furthermore, most of these works do not implement high-order filters.

In the next section, DA fundamentals and the proposed architectures for each kind of filter are reviewed. In Section 3 the results of the FPGA implementation of the structures are presented. Finally, an error model is discussed in Section IV.

2. DA structures

The FIR filter operation (eq.1) can be expressed by (eq.2) using the 2's complement representation of the $x[n]$ input samples of N bits.

$$y = \sum_{t=1}^T A_t x_t \quad (\text{Eq.1})$$

$$\begin{aligned} y &= \sum_{t=1}^T A_t (-x_{t0} + \sum_{n=1}^{N-1} x_{t,n} 2^{-n}) = \\ &= -\sum_{t=1}^T A_t x_{t0} + \sum_{n=1}^{N-1} [\sum_{t=1}^T A_t x_{t,n} 2^{-n}] \end{aligned} \quad (\text{Eq.2})$$

The terms in brackets in Eq.2 can be pre-calculated, saved into a memory and addressed by $x_{t,n}$ (Table I). 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 I

Address					Address Content
$x_{t,N-1}$...	$x_{t,2}$	$x_{t,1}$	$x_{t,0}$	
0	...	0	0	0	0
0	...	0	0	1	A_1
0	...	0	1	0	A_2
0	...	0	1	1	$A_2 + A_1$
1	...	1	1	1	$A_T \dots + A_2 + A_1$

DA direct-form implementation

In a DA bit-serial implementation of a FIR filter, each product term is addressed once per bit (the MSB bit is the sign bit). After the last product-term has been obtained, it is added with its appropriate shift with the rest of the product term previously added.

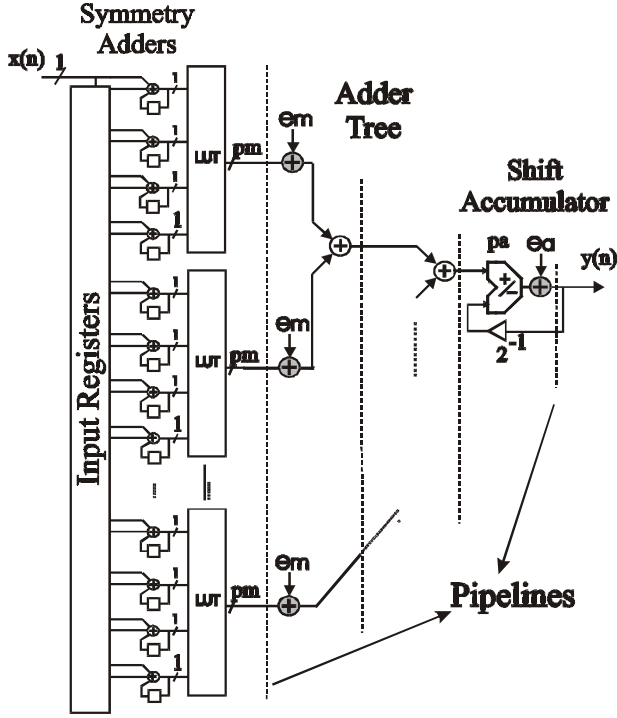


Fig.1 Bit-serial DA direct-form FIR filter

The structure that represents the direct-form FIR filter is showed in fig.1. Considering that the FPGA utilised in this paper has 4-inputs LUTs, the product-terms larger than four need to be divided into r parts so $4 \leq T/r$, where T is the number of taps of the filter. In short, the adders in the tree structure add the r LUT outputs. Eventually, a shift accumulator is required to add and shift each product term. Fig.1 represents a bit-serial implementation of a filter with samples of 8 bits. The output of the filter is obtained each eight clock cycles. When the sign-bit arrives, a subtraction instead of an addition in the shift-accumulator is done. Finally, a symmetrical filter is implemented by using carry save adders before the LUT. Detailed information of the operation can be obtained in [1,4].

Additionally, the range of processing speed can be extended by pipelining the structure. The operation frequency (f_s) can be expressed by eq.4, where L is the latency and n the number of the bits of each input sample.

$$f_s = f_{clk} / (n + L) \quad (\text{Eq.4})$$

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

A linear phase FIR filter can be factorised in several 4th order sections to obtain a reduction in area. The structure of these sections can be DA adapted with the symmetry equation (eq.5) that represents the k^{th} section of a T order filter and its expansion in DA product terms (eq.6). Last equation represents the basic cell of a cascade structure that can be designed by using a bit-serial approach (fig.2).

$$y^k(n) = c_0^k [x^k(n) + x^k(n-4)] + c_1^k [x^k(n-1) + x^k(n-3)] + c_2^k [x^k(n-2)] \quad (\text{Eq.5})$$

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

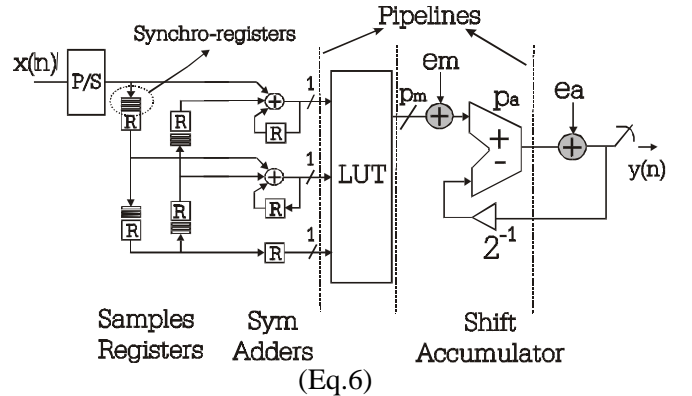


Fig.2. Bit-serial DA cascade 4th order cell.

As a result of the latency generated by the pipeline, the cascade cell has extra registers (one per stage of pipeline) to synchronise the operation of the filter. The eq.7 represents the real-time frequency operation of the cascade structure.

$$f_s = f_{clk} / (n + L) = f_{clk} / 11 \quad (\text{Eq.7})$$

DA lattice filter implementation

The recursive equations that describe the lattice cell structures (eq.8) are used to obtain cascade implementations of M cells [5]. Both the f term and g term represent the forward and the backward prediction in a linear prediction filter structure.

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + K_m g_{m-1}(n-1) \\ g_m(n) &= K_m f_{m-1}(n) + g_{m-1}(n-1) \end{aligned} \quad (\text{Eq.8})$$

Using the DA equations (eq.9) we can reproduce the f and g terms with two LUTs, where g' represents the g(n-1) term.

$$f_m(n) = -f_{m-1,0} + \sum_{n=1}^{N-1} f_{m-1,n} 2^{-n} + K_m [-g'_{m-1,0} + \sum_{n=1}^{N-1} g'_{m-1,n} 2^{-n}]$$

$$g_m(n) = K_m [-f_{m-1,0} + \sum_{n=1}^{N-1} f_{m-1,n} 2^{-n}] + [-g'_{m-1,0} + \sum_{n=1}^{N-1} g'_{m-1,n} 2^{-n}]$$

(Eq.9)

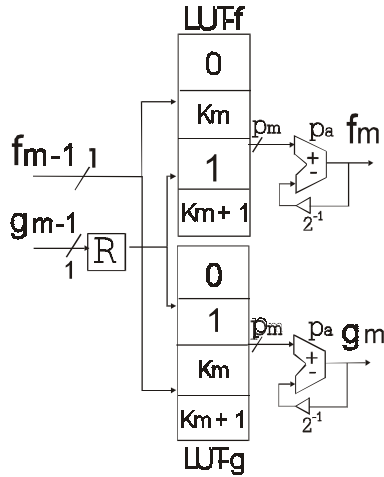


Fig.3. Bit-serial DA lattice cell.

The fig.3 represents the formal DA implementation of these equations. Nevertheless, in this work also is proposed an improved structured that reduces the memory requirements (fig.4). Only is needed to save the coefficient (km) whereas both km+1 and 1 values can be obtained from the carry input of the scaling accumulator.

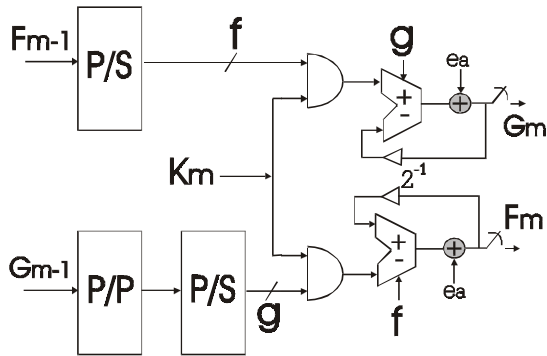


Fig.4. Bit-serial improved lattice cell.

3. FPGA implementation

The filters described above were implemented in an ALTERA EPF10K50RC240-3 FPGA [8]. Default placement and routing compilation options were selected. The lpm_add_sub function from ALTERA was used. This parameterised 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 this two options represent symmetrical version that approximately halves the area of a non-symmetrical implementation of the same structures.

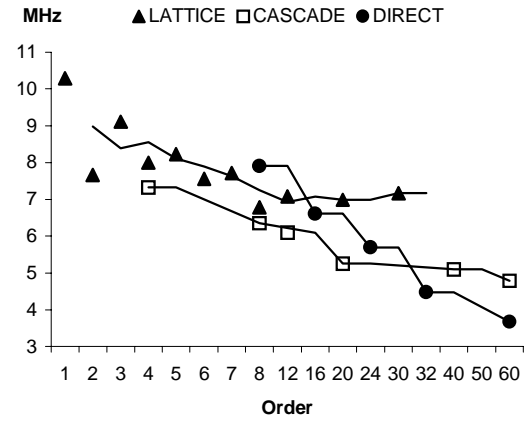


Fig.5. DA bit-serial results in frequency.

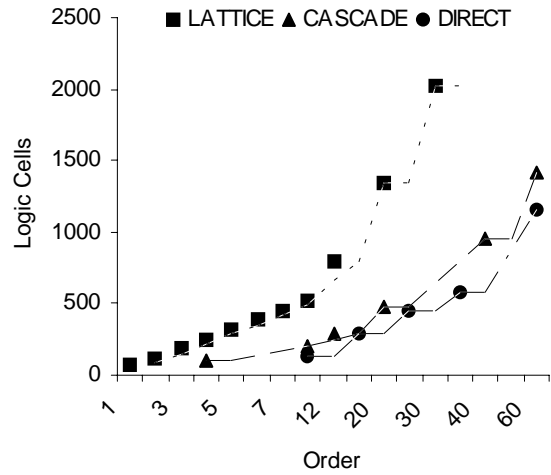


Fig.6. DA bit-serial results in area.

The bit-serial implementation of the lattice structure reaches a real-time operation nearly to 7.5MHz. In the cascade and direct-form structure it continuously decreases to 4MHz.

4. Bit-serial DA Error Model

In this section, a DA error model of each structure taking the assumption that the rounding error (e[n]) and the data input (x[n]) are non correlated [5] is proposed.

In DA bit-serial cascade structures (Fig.2), the error is modelled by eq.11, where e_m and e_a (in grey) are the LUT and shift- accumulator rounding errors.

$$e_y = \sum_{i=0}^{p-1} 2^{-i} \cdot e_m + \sum_{i=0}^{p-2} 2^{-i} \cdot e_a \quad (\text{Eq.11})$$

The variance of this error can be expressed as eq.12, where p_m and p_a are the number of bits in both the memory and the shift-accumulator.

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

A direct-form DA error model [9] is presented in (eq.13); r is the number of data samples partitions or memory partition.

$$e_y = \sum_{i=0}^{p-1} 2^{-i} \cdot \sum_{j=1}^r e_m + \sum_{i=0}^{p-2} 2^{-i} \cdot e_a \quad (\text{Eq.13})$$

Furthermore, as a result of the 4-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). Eq.14 shows the variance of the error in the direct-form structure.

$$\text{var}(e_{df}) = \frac{K}{4} \cdot \left[\frac{1-2^{-2p}}{1-2^{-2p/q}} \right] \cdot \text{var}(e_m) + \left[\frac{1-2^{-2p(q-1)/q}}{1-2^{-2p/q}} \right] \cdot \text{var}(e_{acc}) \quad (\text{Eq.14})$$

We have shown (fig.4) the improved lattice cell with both errors sources e_m and e_a (in grey). The error and the variance in this cell can be expressed by:

$$e_{\text{latt}} = \sum_{i=0}^{p-2} 2^{-i} \cdot e_{acc} \quad \text{var}(e_{\text{latt}}) = \left[\frac{1-2^{-2(p-1)}}{1-2^{-2}} \right] \cdot \frac{2^{-2p_a}}{12} \quad (\text{Eq.15})$$

As example, a T order FIR filter with $p=p_m=p_a=8$ bits can be used to compare the three models. The results in the direct-form, cascade and lattice implementations are $T \cdot 4.2384e-07 + 1.6953e-06$, $T \cdot 3.3907e-06$ and $T \cdot 2.5868e-11$, respectively. As consequence, the lattice filter presents the lowest error, meanwhile cascade form the highest. Finally, the direct-form structure has also a high error compared with the lattice cells.

5. Conclusions

A distributed arithmetic bit-serial approach has been presented to implement the three classical structures of a FIR filter: direct-form, cascade and lattice. Moreover, we have developed a comparative DA

error model of the three structures. The conclusions from the research presented are summarised as follows:

- 1) We can implement a bit-serial 40th order lattice filter in a 10K50 device with a real-time frequency operation of 7.5MHz. The pipelined cascade and direct-form bit-serial implementations reach 4.5MHz and a 60th order in their symmetrical implementations.
- 2) We have been presented a new improved lattice cell than reduce the memory occupation by using the input carry in the shift-accumulator. The cell can be used to decrease or to increase the order of the lattice filter with the same performance.
- 3) We have been offered a DA error model, which show that lattice structure represents the lowest rounding error while cascade has the highest.
- 4) Direct-form is more scalable than the rest of the structures and we can easily select the result precision (simple, double and full precision) whereas cascade and lattice present more difficulty to change the inner precision.

6. References.

- [1] A.Peled and B.Liu. "A New Hardware Realization of Digital Filters". IEEE Trans. On ASSP, vol. ASSP-22, n°6. Pp456-462, Dec. 1974.
- [2] A.Croisier et al. "Digital Filter for PCM encoded signals", U.S. Patent 3 777 130, Dec. 3, 1973
- [3] Altera Conference Paper. "Improving Fixed-Point DSP Processor System Performance with PLDs as a DSP Coprocessor". 1997.
- [4] S.A.White. "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review". IEEE ASSP Magazine July 1989, vol. 6, n°3.
- [5] J.G.Proakis. "Digital Signal Processing". Prentice-Hall, Inc. 1996.
- [6] L. Mintzer. "FIR Filters with FPGA". Journal of VLSI Signal Processing,6, pp119-127. 1993.
- [7] Greg Goslin. "A Guide to Using FPGAs for Application-Specific Digital Signal Processing Performance". Xilinx, Inc.1995.
- [8] ALTERA Data Book.1998
- [9] W.P. Burleson et al. "A VLSI Methodology for Distributed Arithmetic". Journal of VLSI Signal Proc., 2, pp235-252. 1991