ABSTRACT: In this paper, metrics regarding different architectures for distributed arithmetic based FIR filters in FPGA are presented. Main filter parameters are described as well as diverse design techniques applied: pipelining, bit-serial, digit-serial and bit-parallel. Each filter description was written in VHDL at RTL level. For achieving this goal no relative location (rloc) technique was used what redounds on more generic and expensive designs than those available through Core Generator tool. Implementation has been carried out over FPGAs belonging to Xilinx Virtex II family.

I. INTRODUCTION
Today's FPGAs high density has allowed a new field of application: single-chip systems design with embedded DSP algorithms [1,2]. Advantages of this solution are multiple: extra component usage is avoided, off-chip connections are reduced and DSP-core can be simplified and optimized with regard to the application, taking account of different aspects such as required data rate, precision, etc. Frequently bit-parallel circuits are used, but these implementations are expensive in terms of area yielding a greater speed than required. Digit-serial architectures are important choices for efficiently implementing a wide range of circuits with real time signal processing. This solution allows the designer to get results expressed in terms of area-speed, ranging between both implementations: bit-serial and bit-parallel. [3]

Distributed arithmetic-based FIR filters are widely used for implementations that require high performance. Distributed arithmetic is basically a bit-serial operation that performs two vector product (one of which is constant). This technique avoids multiplications through the utilization of lookup tables (LUTs) taking highest advantage of technologies with plenty of memory elements such as RAM based FPGAs, where designers can use LUTs and adders to compute the product. [4]

In this paper a comparison between different single-rate FIR filter architectures based on distributed arithmetic in FPGA is shown. With this aim, RTL descriptions in VHDL and automatic synthesis tools were used. This study involves bit-parallel, bit-serial, digit-serial and pipelining architectures. For descriptions, relative location (rloc) techniques [5] were not used and this allows generic designs to be implemented in any kind of FPGA although at a higher cost because the place and route is performed by the automatic synthesis tool at ease.

II. DISTRIBUTED ARITHMETIC
A FIR filter with T taps constant coefficients is characterized by:

\[ y[n] = \sum_{k=0}^{T-1} h[n].x[n-k] \]  

An efficient way for implementing this filter in FPGA consists in using distributed arithmetic [6]. The main idea underlying this technique is the computation of sum of products such as
\[ y = h^t \cdot x^t = \sum_{i=0}^{T-1} h_i x_i \]  
\( y = \sum_{i=0}^{T-1} h_i (-x_{i0} + \sum_{k=1}^{N-1} x_{i1k} \cdot 2^k) \)  
\[ y = -\sum_{i=0}^{T-1} h_i x_{i0} + \sum_{i=0}^{T-1} (\sum_{k=1}^{N-1} h_i x_{i1k}) \cdot 2^k \]  
\[ y = -A_0(x_{00}, x_{10}, \ldots, x_{T-10}) + \sum_{k=1}^{N-1} A_k(x_{0k}, x_{1k}, \ldots, x_{T-1k}) \cdot 2^k \]  

Where \( h_i \) stands for known values conforming filter coefficients, \( T \) the number of taps, \( N \) input value precision and \( x \) the input data.

\[ A_k(x_{0k}, x_{1k}, \ldots, x_{T-1k}) = \sum_{i=0}^{T-1} h_i x_{ik} \]  
\[ y = - (h_0 x_{00} + h_1 x_{10} + \ldots + h_{T-1} x_{T-10}) + (h_{0} x_{01} + h_1 x_{10} + \ldots + h_{T-1} x_{T-11}) \cdot 2^1 + (h_{0} x_{02} + h_1 x_{11} + h_2 x_{21} + \ldots + h_{T-1} x_{T-12}) \cdot 2^2 + \cdots + (h_{0} x_{0N-1} + h_1 x_{1N-1} + \ldots + h_{T} x_{T-1N-1}) \cdot 2^{(N-1)} \]  

\( A_k \) only can have a finite number of values \((2^T)\), what means that every single term in (7) is stored either in ROM or LUT. In fig. 1 is shown the way of implementing the sum of products where ROM size is \(2^T \times W_{\text{ROM}}\), with \( W_{\text{ROM}}\) the precision of output data.

\[ X_{00}, \ldots, X_{01}, X_{00}, \ldots, X_{10}, X_{10}, \ldots, X_{11}, X_{11}, \ldots, X_{T-10}, \ldots, X_{T-11}, X_{T-11}, \ldots, X_{T-10} \rightarrow \text{ROM} \rightarrow y(n) \]  

**II. MODELS USED IN THIS WORK**

Operations to be performed by architectures based on distributed arithmetic consist in sequentially searching tables, adding, subtracting and shifting the input signal. All of these can be efficiently implemented through FPGAs.

As tables sizes can become huge, memory can be partitioned in \( T/k \) k-bit partitions. In this way it is possible to change from a memory with \(2^T\) size to \( T/k \) memories with \(2^k\) size, incorporating adders because of the partitioning.

As stated above, bit-serial, digit-serial, bit-serial with pipelining, digit-serial with pipelining and bit-parallel architectures are considered in this paper.
Distributed arithmetic based filters have a bit-serial pattern (fig. 2). For N-bit precision inputs N clock cycles are need to yield an output, so that if CR is the clock rate, filter sampling rate (SR) is CR/N.

![Fig. 2 – Bit-serial FIR filter](image)

To indirectly increase the sampling rate, pipeline can be used. With this technique (pipelining) clock rate (CR) can be increased decreasing logical paths through the inclusion of registers. In fig. 3 is shown that the portion of segmented bit-serial architecture is that related with the cascaded adders that operate on the pre-stored values coming from ROMs or LUTs.

![Fig. 3 – Pipelined bit-serial FIR filter](image)

To directly improve filter sampling rate, more bits can be processed serially (fig. 4) using digit-serial and digit size D. In this way, the sampling rate is accelerated D times (SR=D,SR [bit-serial]). Furthermore, sampling rate can be increased by means of pipelining producing an increased clock rate. Maximum rate is obtained when D equals N, i.e. parallel situation.
Let \( CR \) be the clock rate, \( SR \) the filter's sampling rate, \( N \) the input size, \( D \) the digit size, therefore the following assertions may be assumed as valid (theoretically):

\[
SR[\text{bit-serial}] = \frac{CR[\text{bit-serial}]}{N} \\
SR[\text{digit-serial}] = D \cdot SR[\text{bit-serial}] \\
SR[\text{parallel}] = SR[\text{digit-serial}] \\
CR[\text{bit-serial con pipelining}] > CR[\text{bit-serial}] \quad \Rightarrow \quad SR[\text{bit-serial con pipelining}] > SR[\text{bit-serial}] \\
CR[\text{digit-serial con pipelining}] > CR[\text{digit-serial}] \quad \Rightarrow \quad SR[\text{digit-serial con pipelining}] > SR[\text{digit-serial}]
\]

When FPGAs are used, it is not possible to certify these assertions because the implementation strongly depends on the successive stages of the automatic synthesis process [7]. As a matter of fact, in this experience neither relative location (rloc) technique nor physical or logical constraints were used. Therefore the place and route tool proceeded at ease when performing physical design for the different alternatives.

**III. EXPERIMENTAL RESULTS**

Distributed-arithmetic FIR filters architectures with 8, 12, 16, 24, 32 and 64 taps were implemented using a Virtex 2 FPGA from Xilinx (XC2v2000-6bg575). Synthesis was provided with XST (Xilinx Synthesis Technology) Tools [8] running on Xilinx ISE (Xilinx System Environment) version 5.1 [9].

Different filters were tested with 8-bit and 12-bit inputs and coefficients. Precision and bit-size of output data were automatically computed taking account the number of taps, input data precision and coefficient values. Thus, calculations were performed in advance aiming the filter can compute the sums without overflow as well as an extension for successive shifts applied was considered.

The following 8-bit inputs and coefficients filters were implemented:
• FIR bit-serial
• FIR bit-serial with pipelining
• FIR digit-serial (digit size 2)
• FIR digit-serial (digit size 2) with pipelining
• FIR digit-serial (digit size 4)
• FIR digit-serial (digit size 4) with pipelining
• FIR bit-parallel

<table>
<thead>
<tr>
<th>Taps</th>
<th>B-Serial</th>
<th>B-Serial Pipe</th>
<th>D-Serial (2)</th>
<th>D-Serial (2) Pipe</th>
<th>D-Serial (4)</th>
<th>D-Serial (4) Pipe</th>
<th>B-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>101,94</td>
<td>149,48</td>
<td>85,32</td>
<td>133,69</td>
<td>62,03</td>
<td>73,26</td>
<td>43,63</td>
</tr>
<tr>
<td>12</td>
<td>92,25</td>
<td>144,30</td>
<td>73,21</td>
<td>123,91</td>
<td>54,79</td>
<td>81,57</td>
<td>36,79</td>
</tr>
<tr>
<td>16</td>
<td>80,71</td>
<td>138,12</td>
<td>68,68</td>
<td>112,99</td>
<td>51,02</td>
<td>70,97</td>
<td>32,16</td>
</tr>
<tr>
<td>24</td>
<td>73,69</td>
<td>142,04</td>
<td>63,01</td>
<td>117,92</td>
<td>40,75</td>
<td>77,34</td>
<td>30,07</td>
</tr>
<tr>
<td>32</td>
<td>69,98</td>
<td>131,93</td>
<td>55,01</td>
<td>106,72</td>
<td>39,81</td>
<td>71,17</td>
<td>29,58</td>
</tr>
<tr>
<td>64</td>
<td>53,05</td>
<td>92,51</td>
<td>46,30</td>
<td>90,01</td>
<td>36,35</td>
<td>63,86</td>
<td>29,88</td>
</tr>
</tbody>
</table>

Table 1 – Clock rate (CR) for filters with 8-bit inputs and coefficients [Mhz]

Table 1 shows that bit-serial filters with pipelining have the best CR for 8-bit inputs and coefficients but this is not what happened with the sampling rate (SR) because of the different digit sizes involved. In table 2 can be seen that, best SR filters (for all tap numbers, except for 8) were those implemented as digit-serial with pipelining and digit size equal 4.

<table>
<thead>
<tr>
<th>Taps</th>
<th>B-Serial</th>
<th>B-Serial Pipe</th>
<th>D-Serial (2)</th>
<th>D-Serial (2) Pipe</th>
<th>D-Serial (4)</th>
<th>D-Serial (4) Pipe</th>
<th>B-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>12,74</td>
<td>18,68</td>
<td>21,33</td>
<td>33,42</td>
<td>31,01</td>
<td>36,63</td>
<td>43,63</td>
</tr>
<tr>
<td>12</td>
<td>11,53</td>
<td>18,04</td>
<td>18,30</td>
<td>30,98</td>
<td>27,39</td>
<td>40,78</td>
<td>36,79</td>
</tr>
<tr>
<td>16</td>
<td>10,09</td>
<td>17,26</td>
<td>17,17</td>
<td>28,25</td>
<td>25,51</td>
<td>35,48</td>
<td>32,16</td>
</tr>
<tr>
<td>24</td>
<td>9,21</td>
<td>17,75</td>
<td>15,75</td>
<td>29,48</td>
<td>20,37</td>
<td>38,67</td>
<td>30,07</td>
</tr>
<tr>
<td>32</td>
<td>8,74</td>
<td>16,49</td>
<td>13,75</td>
<td>26,68</td>
<td>19,90</td>
<td>35,58</td>
<td>29,58</td>
</tr>
<tr>
<td>64</td>
<td>6,63</td>
<td>11,56</td>
<td>11,57</td>
<td>22,50</td>
<td>18,17</td>
<td>31,93</td>
<td>29,88</td>
</tr>
</tbody>
</table>

Table 2 – Sampling rate (SR) for filters with 8-bit inputs and coefficients [Mhz]

Tables 3 and 4 show slices and equivalent gates occupation for the 8-bit input filters. As expected, parallel filters are those more expensive in terms of slices and equivalent gate occupation.

<table>
<thead>
<tr>
<th>Taps</th>
<th>B-Serial</th>
<th>B-Serial Pipe</th>
<th>D-Serial (2)</th>
<th>D-Serial (2) Pipe</th>
<th>D-Serial (4)</th>
<th>D-Serial (4) Pipe</th>
<th>B-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>63 (1)</td>
<td>72 (1)</td>
<td>80 (1)</td>
<td>85 (1)</td>
<td>117 (1)</td>
<td>120 (1)</td>
<td>178 (1)</td>
</tr>
<tr>
<td>12</td>
<td>92 (1)</td>
<td>106 (1)</td>
<td>120 (1)</td>
<td>135 (1)</td>
<td>184 (1)</td>
<td>201 (1)</td>
<td>298 (2)</td>
</tr>
<tr>
<td>16</td>
<td>122 (1)</td>
<td>135 (1)</td>
<td>161 (1)</td>
<td>174 (1)</td>
<td>244 (2)</td>
<td>261 (2)</td>
<td>395 (3)</td>
</tr>
<tr>
<td>24</td>
<td>188 (1)</td>
<td>209 (1)</td>
<td>259 (2)</td>
<td>287 (2)</td>
<td>408 (3)</td>
<td>447 (4)</td>
<td>680 (6)</td>
</tr>
<tr>
<td>32</td>
<td>239 (2)</td>
<td>252 (2)</td>
<td>324 (3)</td>
<td>335 (3)</td>
<td>494 (4)</td>
<td>506 (4)</td>
<td>825 (7)</td>
</tr>
<tr>
<td>64</td>
<td>477 (4)</td>
<td>508 (4)</td>
<td>660 (6)</td>
<td>699 (6)</td>
<td>1030 (9)</td>
<td>1089 (10)</td>
<td>1718 (15)</td>
</tr>
</tbody>
</table>

Table 3 – slices occupation for FIR with 8-bit inputs and coefficients
(Percentage over a total of 10752 slices)
The following 12-bit inputs and coefficients filters were designed and implemented:

- FIR *bit-serial*
- FIR *bit-serial* with *pipelining*
- FIR *digit-serial* (digit size 2)
- FIR *digit-serial* (digit size 2) with *pipelining*
- FIR *digit-serial* (digit size 4)
- FIR *digit-serial* (digit size 4) with *pipelining*
- FIR *digit-serial* (digit size 6)
- FIR *digit-serial* (digit size 6) with *pipelining*
- FIR *bit-parallel*

Table 5 shows that *bit-serial* filters with *pipelining* have the best CR for 12-bit inputs and coefficients (as happened with 8-bit precision). Regarding the sampling rate (SR), in table 6 can be seen that, best SR filters were those implemented as *digit-serial* with *pipelining* and digit size equal 6.

Tables 7 and 8 show *slices* and equivalent gates occupation for the 12-bit input filters. As with 8-bit, *parallel* filters are those more expensive in terms of *slices* and equivalent gate occupation.
Let $\delta$ be the relationship between the sampling rate (throughput) and the area or number of equivalent gates. This quantity was computed for every single filter\((i,j)\), where $i$ stands for the coefficient number or taps (row of tables) and $j$ stands for the technique applied (column of tables). Hence:

$$\delta[\text{filter}(i,j)] = \left[\frac{\text{SR}(i,j)}{\text{EG}(i,j)}\right] \times r(i) \tag{8}$$

where EG is the equivalent gate number for the design and $r(i)$ a factor used for normalizing $\text{SR}(i,j)$ as well as $\text{EG}(i,j)$:

$$r(i) = \frac{\text{max}[\text{EG}(i,j)]}{\text{max}[\text{SR}(i,j)]} \tag{9}$$

maximums are computed for all $j$ or filters designed for a certain tap ($i$ row).

Tables 10 and 11 show speed (rate)-area ratio for filters with 8-bit and 12-bit inputs respectively. Values were computed through (8)

<table>
<thead>
<tr>
<th>Taps</th>
<th>B-Serial (Pipe)</th>
<th>B-Serial (2)</th>
<th>D-Serial (2) Pipe</th>
<th>D-Serial (2)</th>
<th>D-Serial (4) Pipe</th>
<th>D-Serial (4)</th>
<th>D-Serial (6) Pipe</th>
<th>B-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>716</td>
<td>732</td>
<td>741</td>
<td>756</td>
<td>770</td>
<td>785</td>
<td>785</td>
<td>805</td>
</tr>
<tr>
<td>12</td>
<td>825</td>
<td>856</td>
<td>868</td>
<td>880</td>
<td>891</td>
<td>900</td>
<td>900</td>
<td>900</td>
</tr>
<tr>
<td>16</td>
<td>815</td>
<td>826</td>
<td>832</td>
<td>842</td>
<td>850</td>
<td>860</td>
<td>860</td>
<td>860</td>
</tr>
<tr>
<td>24</td>
<td>789</td>
<td>811</td>
<td>811</td>
<td>818</td>
<td>824</td>
<td>834</td>
<td>834</td>
<td>834</td>
</tr>
<tr>
<td>32</td>
<td>776</td>
<td>792</td>
<td>796</td>
<td>800</td>
<td>806</td>
<td>816</td>
<td>816</td>
<td>816</td>
</tr>
<tr>
<td>64</td>
<td>697</td>
<td>709</td>
<td>716</td>
<td>720</td>
<td>724</td>
<td>728</td>
<td>728</td>
<td>728</td>
</tr>
<tr>
<td>Avg.</td>
<td>0.769</td>
<td>0.999</td>
<td>0.987</td>
<td>1.23</td>
<td>0.975</td>
<td>0.982</td>
<td>0.892</td>
<td></td>
</tr>
</tbody>
</table>

Table 10 – Speed-area ratio ($\delta$) for filters with 8-bit inputs and coefficients
### IV. CONCLUSIONS

A distributed arithmetic based single-rate FIR in FPGAs study was presented. 8-bit and 12-bit inputs and coefficients filters were analyzed as well as a wide coefficient diversity.

As no relative location (rloc) technique was used, designs are generic, i.e. can be implemented over any FPGA.

Although the synthesis tool had the maximum freedom degree for implementing at ease, results obtained have a close correspondence with that expected through theoretically considerations. Bit-parallel filters are more expensive in terms of area but are faster (in terms of sample rate) than digit-serial filters, which in turn are more expensive and faster than bit-serial filters. Through the application of pipelining techniques speed and area were increased.

Finally, was concluded that digit-serial with pipelining and digit size 2 implementation has the best speed-area ratio (δ) among all filters considered in this work.

### V. REFERENCES


5. Xilinx, inc. "Constraints Guide - ISE5.1" section 2-4, 2-15 "Relative Location (RLOC) and Relationally Placed Macros (RPMs)", 2002.


