REDUCED AREA FULLY PARALLEL AND FULLY SERIAL FIR FILTER ON FIELD-PROGRAMMABLE GATE ARRAY

NILESH B. BOSMIYA, R. C. PATEL
PG Student L.D. College of Engineering, PROF. L.D. College of Engineering

Abstract—In this paper fully serial and fully parallel FIR filters are designed with different quantization on FPGA for low area requirement. Modified fully serial and fully parallel band-pass FIR filters with same specification as in reference paper designed and simulated on ISE. The suggested implementations are synthesized with Xilinx ISE 14.2 version. Results show that compare with reference paper low area and better speed are achieved.

Keywords—FIR Filter, FPGA, Quantization, Area, Speed, Verilog HDL

I. INTRODUCTION

Digital filters are important part of digital signal processing. Before development of FPGA digital filter were implemented on digital signal processor. Digital signal processors are still widely used but they are not capable for high speed application available in present. After the advancement of microelectronic techniques, communication signal processing has come to third generation and forth generation period, so there is a challenge for adaptive processing techniques that the processing speed needs to be high so FPGA based signal processing techniques is mostly used in latest mobile communication, military communication, consumer electronics and aerospace tracking etc so that It is necessary to find the answer of how to increase operation speed of signal processing algorithms and reduce hardware resources by adopting FPGA to implement every kinds of tasks of digital signal processing. So we look forward for design of digital filter with low area and high speed.

Benefits of reducing area:

(a) Less power required
(b) Area benefits for other application on same chip
(c) We can use versions of FPGA which have less capability.

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by \( y = f * x \) where \( f \) is the filter’s impulse response, \( x \) is the input signal, and \( y \) is the convolved output. The linear convolution process is formally defined by[1]:

\[
y[n] = x[n] * f[n] = \sum_{k} x[k]f[n-k]
\]

LTI digital filters are generally classified as being finite impulse response (i.e., FIR), or infinite impulse response (i.e., IIR). Calculating the constant coefficients of such a digital filter involves considerable amount of computation and this is generally performed using software tools[1].

With available digital filter design software the production of FIR coefficients is a straightforward process. The Filter Design and Analysis (FDA) tool packaged along with MATLAB is such a tool. The double length floating point notation for filter coefficients, used by the FDA tool poses immense challenges in terms of cost and resources, while implementing on an FPGA[1].

The challenge remains is to map the FIR design into a suitable architecture. To overcome this, the filter coefficients have to be quantized to a fixed point notation. The result of coefficient quantization is that the actual implemented transfer function is different from the ideal transfer function. The simplest and most widely used approach to the problem is to round off the optimal infinite precision coefficients to a b-bit representation[1].

II. PARALLEL AND SERIAL ARCHITECTURES

The basic equation for a single-channel FIR filter is shown in equation [1]

\[
y(n) = \sum_{k} x(n-k)f(k)
\]

The terms in the equation can be described as input samples, output samples, and coefficients. Imagine \( x(n) \) as a continuous stream of input samples and \( y(n) \) as a resulting stream (i.e., a filtered stream) of output samples[1].

The \( n \) and \( k \) in the equation correspond to a particular instant in time, so to compute the output sample \( y(n) \) at time \( n \), a group of input samples at \( N \) different points in time, or \( x(n), x(n-1), x(n-2), \ldots \).
x(n-N+ 1) is required. The group of N input samples are multiplied by N coefficients and summed together to form the final result y(n). Fig. 1 shows the logical structure of an FIR Filter[1].

A fully parallel architecture uses a dedicated multiplier and adder for each filter tap; all taps execute in parallel, thereby creating fully parallel implementation. This architecture is optimal for speed. However, it requires more multipliers and adders than a serial architecture, and therefore consumes more chip area. Fig. 2 shows the fully serial architecture of 64 tap FIR Filter. The single-multiplier MAC FIR is one of the simplest DSP filter structures. A fully serial architecture conserves area by reusing multiplier and adder resources sequentially. Therefore reduces hardware by a factor of N, but also reduces filter throughput by the same factor. Fig. 3 shows the fully serial architecture of 64-tap FIR Filter[1].

### III. FPGA SIMULATION AND RESULT COMPARISON

An FIR Band Pass equiripple filter is designed as per the specifications given in reference paper[1].

<table>
<thead>
<tr>
<th>Filter Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling Frequency</td>
<td>48 MHz</td>
</tr>
<tr>
<td>Stop band Frequency 1</td>
<td>7.8 MHz</td>
</tr>
<tr>
<td>Pass band Frequency 1</td>
<td>9.6 MHz</td>
</tr>
<tr>
<td>Pass band Frequency 2</td>
<td>12.4 MHz</td>
</tr>
<tr>
<td>Stop band Frequency 2</td>
<td>14.4 MHz</td>
</tr>
<tr>
<td>Stop band Attenuation 1</td>
<td>84 dB</td>
</tr>
<tr>
<td>Pass band Attenuation</td>
<td>2 dB</td>
</tr>
<tr>
<td>Stop band Attenuation 2</td>
<td>84 dB</td>
</tr>
</tbody>
</table>

A special class of FIR filter that is particularly effective in meeting such specifications is called the equiripple FIR filter. An equiripple design protocol minimizes the maximal deviations (ripple error) from the ideal transfer function. The filter designed for the mentioned specifications using equiripple design method is of order 64[1].

Fig. 4 is for the response of the reference filter without quantization. With the help of FDA Tool, quantization of the reference filter to Q 16.14 and Q8.7 fixed point numeric representation format is done. Fig. 5 is for quantized Q16.14 filter response and Fig. 6 is for quantized Q8.7 filter response.
Improved fully parallel and fully serial filters were designed and their behavioural simulation was done using Xilinx ISE 14.2. Resource utilization for reference paper filters are shown in table 2. Resource utilization for optimized filters is shown in table 3.

Table 2
Reference paper resource utilization

<table>
<thead>
<tr>
<th></th>
<th>Fully parallel</th>
<th>Fully serial</th>
<th>Fully parallel</th>
<th>Fully serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantization</td>
<td>Q16.14</td>
<td>Q16.14</td>
<td>Q8.7</td>
<td>Q8.7</td>
</tr>
<tr>
<td>Slices</td>
<td>3297</td>
<td>1097</td>
<td>1243</td>
<td>760</td>
</tr>
<tr>
<td>Slice Flip Flops</td>
<td>1071</td>
<td>1146</td>
<td>894</td>
<td>952</td>
</tr>
<tr>
<td>LUTs</td>
<td>5263</td>
<td>1162</td>
<td>1474</td>
<td>533</td>
</tr>
<tr>
<td>Max.Freq (Mhz)</td>
<td>7.2</td>
<td>51.42</td>
<td>10</td>
<td>59.15</td>
</tr>
</tbody>
</table>

Now the Starting portion of simulation results of optimized filters is shown in Fig. 7 and Fig. 8.
CONCLUSION AND FUTURE SCOPE OF WORK

We can clearly see that from table 3 area is reduced and speed increased compared to reference paper. Further optimization can be done with respected to speed or area.

REFERENCES


