VLSI Architecture for FFT Using Radix-4 of Complex Valued Data
Ch.Kamala Kumarai, M.Praveena Sindhu, M.B.Rashmi Teja, P.Ramesh, K.Sudhakar, Nasreen and B.Thriveni

1. INTRODUCTION
The Fourier Transform is an inevitable approach in signal processing, particularly for applications in Orthogonal Frequency Division Multiplexing (OFDM) systems. The Fast Fourier transform (FFT) is an appropriate technique to do manipulation of DFT. The algorithm of FFT was devised by Cooley and Tukey in order to decrease the amount of complexity with respect to time and computations. A large number of FFT algorithms have been developed, but among all radix-2, radix-4 and split radix are most widely used for practical applications due to their simple architecture, with constant butterfly geometry and the possibility of performing them ‘in place’.

The hardware of FFT can be implemented by two types of classifications– memory architecture and pipeline architecture.

The memory architecture comprises a single processing element and various units of memory. The merits of memory architecture include low power and low cost when compared to that of other styles. The specific demerits are greater latency and lower throughput. The above demerits of the memory architecture are totally eliminated by pipeline architecture at the expense of extra hardware in an acceptable way.

2. BUTTERFLY ARCHITECTURE
The most important element in FFT processor is a butterfly structure. It takes two signed fixed-point data from memory register and computes the FFT algorithm. The output results are written back in same memory location as the previous input stored. This method is called in-placement memory storage whereby it can reduce the hardware utilization. The butterfly architecture is shown in Fig. 1. The adder sums the input before being multiplied by the twiddle factor. The multiplier forms the partial product of the complex multiplication and produce two times bigger than input bit. Shift register would shift the bits to avoid overflow issue. Output of this butterfly would be kept in the register for the subsequent stage.

Fig 1. Butterfly architecture.

3. RADIX-4 FFT ALGORITHM
The algorithm for 16-point radix-4 FFT is studied for implementation. The FFT can be implemented with decimation either in time or frequency. Here the decimation in time (DIT) block diagram is studied. For the required specification two stages of 4-point butterflies are required. In each stage first part of the value corresponds to cosine function and the second to sine function. The other popular algorithm is the radix-4 FFT, which is even more efficient than the radix-2 FFT. The radix-4 FFT equation is listed below:
While deciding the wordlength needed for an lab. The 
eq \sum_{n=0}^{N-1} \left[ x(n) + (-j)^{\lfloor n/N \rfloor} + (-1)^{\lfloor n/N \rfloor} + j^{\lfloor N/2 \rfloor} \right] W^k_N

5. RESULTS AND DISCUSSION

The Radix 4 FFT algorithm and its functionality were discussed before. Now we deal with the simulation and synthesis results of 16 point Radix 4 FFT design. Here Modelsim tool is used in order to simulate the design and to check the functionality of the design. Once the functional verification is done, the design will be taken to the Xilinx tool for synthesis process and the netlist generation.

The Appropriate test cases have been identified in order to test this modelled Radix 4 FFT design. Based on the identified values, the simulation results describes the operation of the Radix 4 FFT has been achieved. This proves that the modelled design works properly as per the process.

The identified test cases are simulated through the test bench. These test cases will clearly explain the operation of Radix 4 FFT and can be verified.

The device utilization summary is shown below in which it gives the details of number of devices used from the available devices and also represented in %. Hence as the result of the synthesis process, the device utilization in the used device and package is shown below.

### Table 1. Device utilization summary.

<table>
<thead>
<tr>
<th>Device Utilization</th>
<th>Value</th>
<th>Percentage</th>
<th>Utilization</th>
<th>Unused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of BUFGs</td>
<td>1/32</td>
<td>3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of DSP48Es</td>
<td>3/32</td>
<td>9%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of External IOBs</td>
<td>673/960</td>
<td>70%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of LOCed IOBs</td>
<td>0/673</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Slice Registers</td>
<td>198/122880</td>
<td>1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as Flip Flop</td>
<td>198</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as Latches</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as LatchThrus</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Slice LUTS</td>
<td>2080/122880</td>
<td>1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Slice LUT-Flip Flop pairs</td>
<td>2255/122880</td>
<td>1%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.1. Memory Usage:

Peak Memory Usage: 482 MB

6.2. Device Utilization Summary:

- Number of BUFGs: 1 out of 32 (3%)
- Number of DSP48Es: 36 out of 384 (9%)
- Number of External IOBs: 673 out of 960 (70%)
- Number of LOCed IOBs: 0 out of 673 (0%)
- Number of Slice Registers: 198 out of 122880 (1%)
- Number used as Flip Flop: 198
- Number used as Latches: 0
- Number used as LatchThrus: 0
- Number of Slice LUTS: 2080 out of 122880 (1%)
- Number of Slice LUT-Flip Flop pairs: 2255 out of 122880 (1%)

7. CONCLUSION

Fast Fourier Transform (FFT) techniques have revolutionized the Digital Signal Processing techniques in the past 30 years. It does not only provide a fast response but also provide many logic thought to be un-realizable come easily in the range to be realized. The parallel processing of FFT hence has been proposed to be used in multiprocessor algorithms. When an FFT is implemented in special-purpose hardware, errors and constraints due to finite word lengths are unavoidable. While deciding the wordlength needed for an FFT, these quantization effects must be considered.

The algorithm for 16-point radix-4 FFT is studied for implementation. The HDL coding in DIT scheme is done and tested with some test vectors generated using Matlab. The design is synthesized on FPGA Virtex 5 configuration. A different implementation methodology could be used to reduce the quantization errors.
Also, the quantization factor could be made floating, i.e. dependent on the result rather only the inputs. This could provide different scaling levels from the same logic and have a large scope of application of the design.

8. REFERENCES