Efficient Split Radix FFTs in FPGAs

Tom Dillon
Dillon Engineering, Inc.

This presentation outlines methods for split radix FFTs implemented in FPGAs. Analysis of various algorithms with regards to performance, cost and power consumption are presented.

FPGAs are rapidly finding their way into high performance DSP applications, specifically real time signal processing applications. Large FPGAs offer a significant cost, size, and power advantage over other alternatives for many front end real time processing operations. FPGAs offer the advantage of short and flexible design cycles, high performance and relatively low NRE.

The FFT is at the heart of many real time signal processing applications, including radar, communication, and image processing. Logic for high speed FFTs can account for up to 90% of the cost and power of a given application, making efficient resource usage critical.

Architectures for radix-2 FFTs are well known and have been in use in FPGAs for some time with excellent results. Many applications require bin spacing that can't be achieved with radix-2 FFTs since one is limited to a power of 2 length. In order to attain more useful bin spacing, many times a split radix FFT architecture is used. For example, using a radix-2 with radix-3 architecture can produce 384 (128 x 3), 768 (256 x 3), 1536 (512 x 3) and so on. This architecture is not limited to radix-2 and radix-3, as with proper data routing structures any split radix combination can be implemented.

Two common methods for implementing split radix FFT architectures are Kolba-Parks and Cooley-Tukey. The advantages of both architectures pertaining to FPGA implementations will be detailed.

The basic structure of a 1536 point FFT architecture is given in the following diagram.

![Diagram of High Performance Parallel Radix-2 Times 3 Pt. Architecture 1536 Implementation]

The size, cost, and power consumption of the FPGA are directly proportional to the throughput
**Efficient Split Radix FFTs in FPGAs**

See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.
of this FFT engine since logic is added as required to keep up with processing requirements. Some real examples of FPGA device usage, power consumption and cost will be presented. The architecture as shown accepts 3 inputs and produces 3 outputs per clock cycle, but can easily be extended for higher or lower performance applications. Clock rates in excess of 200MHz can be used with today's FPGA technology.

Another simple extension to this architecture is for ultra long FFTs. A similar architecture has been used to produce 512K and even 1M point FFTs in FPGAs with the simple addition of external memory to store intermediate results. Longer lengths only require more external memory.

The basic structure is shown below and will be presented with device usage, power and cost examples.

Figure 2: Ultra Long FFT Architecture
1M Continuous Data

A strong case is made for using FPGA technology for FFT processing in real time DSP applications.
Implementing Efficient Split-Radix FFTs in FPGAs

- Radix-2 and Radix-4 FFTs are common
- Many applications benefit from other lengths
- OFDM Transceiver, Digital Video Broadcasts, and software defined radios often require FFTs that aren’t a radix-2 or radix-4 length
- Split-radix simplifies the logic for these applications
- Common Split-radix algorithms are Cooley-Tukey, Kolbe-Parks, and Good-Thomas
Split-Radix FFTs

- Split-radix refers to combinations of two (or more) FFT engines
- Split-radix FFTs have a similar structure to 2D FFTs
- Split-radix FFTs provide bin spacing that produce better results for many applications
- Two split-radix approaches employed by DE:
  - Serial (traditional) – lower performance, higher memory requirements by using serial versions of both FFTs
  - Parallel – higher performance by placing larger radix FFTs in parallel and using a parallel version of the smaller radix FFT
- Parallel version mainly used when combining a larger FFT with a 3 or 5 point FFT, since it is feasible to use 3 or 5 large FFTs in a single device
*Continuous data FFTs require enough memory to store two full copies of the data for each re-order stage*
Serial 768-Point Split-Radix FFT

0, 1, 2, ...767

Data Re-order

P-Point FFT (256-Point)

Q-Point FFT (3-Point)

Data Re-order

0, 1, 2, ...767
Serial 768-Point Split-Radix FFT (cont.)

- Single engine of each radix (256-point FFT followed by 3-point FFT)
- Lower device utilization, with performance suitable for most applications
- High memory requirements for data re-ordering
- Speeds up to continuous data, slower data rates require less logic
- Same structure (with external memory) used for ultra-long FFTs
Parallel 768-Point Split-Radix FFT

- P-Point 256 FFT
  - 0, 3, 6, … 765

- 256 FFT
  - 1, 4, 7, … 766

- 256 FFT
  - 2, 5, 8, … 767

- Q-Point 3 FFT (Parallel)
  - Col 0
  - Col 1
  - Col 2

- 3 Point Re-order
  - A0
  - A2
  - A1
  - 0, 1, 2, … 255
  - 256, 257, … 511
  - 512, 513, … 767
Parallel 768-Point Split-Radix FFT Data Flow

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

... ... ...

765 766 767

3 x 256-pt on columns with rotate

<table>
<thead>
<tr>
<th>0</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>257</td>
<td>513</td>
</tr>
<tr>
<td>2</td>
<td>258</td>
<td>514</td>
</tr>
</tbody>
</table>

... ... ...

255 511 767

256 x 3-pt FFT on rows with rotate

<table>
<thead>
<tr>
<th>0</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>257</td>
<td>513</td>
</tr>
<tr>
<td>2</td>
<td>258</td>
<td>514</td>
</tr>
</tbody>
</table>

... ... ...

255 511 767

Out

Dillon ENGINEERING

Copyright © 2003, Dillon Engineering Inc. All Rights Reserved.
Parallel 768-Point Split-Radix FFT (cont.)

- Combines 256-point FFT with 3-point FFT
  - 3 x 256-point FFT executions
  - 256 x 3-point FFT executions
- Eliminates the need for intermediate memory
- Higher resource (logic) usage as more computations are performed in parallel
- Very high performance – perform a new 768-point FFT every 256 clock cycles (1.7uS @ 150 MHz)
## Virtex II Performance

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of Butterflies</th>
<th>Latency (uS)</th>
<th>FFT Rate (uS)</th>
<th>Sizes</th>
<th>Block RAM</th>
<th>Multipliers</th>
<th>Power (mW)</th>
<th>Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>1</td>
<td>20.67</td>
<td>20.42</td>
<td>3,562</td>
<td>12</td>
<td>4</td>
<td>756</td>
<td>290</td>
</tr>
<tr>
<td>Serial</td>
<td>4</td>
<td>15.67</td>
<td>5.12</td>
<td>5,224</td>
<td>18</td>
<td>16</td>
<td>969</td>
<td>500</td>
</tr>
<tr>
<td>Parallel</td>
<td>3</td>
<td>6.96</td>
<td>6.83</td>
<td>4,180</td>
<td>24</td>
<td>12</td>
<td>974</td>
<td>350</td>
</tr>
<tr>
<td>Parallel</td>
<td>6</td>
<td>3.54</td>
<td>3.41</td>
<td>6,260</td>
<td>45</td>
<td>24</td>
<td>1,331</td>
<td>700</td>
</tr>
<tr>
<td>Parallel</td>
<td>12</td>
<td>1.84</td>
<td>1.70</td>
<td>11,123</td>
<td>75</td>
<td>48</td>
<td>1,840</td>
<td>1,400</td>
</tr>
</tbody>
</table>

- Latency: Time from last point in to first out
- FFT rate: Rate to input FFT data sets
- Power: Estimate via Xilinx XPower
- Cost: Based on single piece XC2V3000-6 from Partminer.com
Other Dillon Engineering Resources

- ParaCore Architect (parameterized core builder)
- DSP Algorithms
  - Ultra-long FFTs (2k x 2k = 4M points)
  - 2D FFTs for image processing
  - Fixed or floating-point FFTs
  - Floating point math library
- System level DSP
  - OFDM Transceivers
  - Radar Processing on single FPGA
  - Image Compression/Processing
- FPGA-based DSP development platforms
- Hardware/Software SOC
  - High speed Ethernet Appliances
  - Linux Based SOC in FPGA
  - MicroBlaze application