High Performance Communications Using FPGAs

Dr Chris Dick
Xilinx
IP Development Group
chrisd@xilinx.com
Presentation Outline

• Software radio architecture and justification already covered
  – Significant DSP challenges highlighted
• Why FPGAs for DSP?
• Performance through parallelism
• Some examples
• System design using FPGAs
• What about power?
• EDA and system level tools
• New FPGA architecture - *Virtex* - for DSP
Software Radios - Approaches

• Speakeasy I - Multiprocessor using
  – quad TMS320C40 MCMs
  – Application specific processors (ASP)

• Speakeasy II
  – ‘C44s
  – CHAMPS (FPGA)

• MIT SpectrumWare - workstation hardware
  – processing done in user space

• Berkeley - Infopad

• Comparatively recently: FPGAs
Software Programmable DSPs

• Datapaths optimized for performing common DSP tasks
  – single-cycle multiply-accumulate (MAC)
  – bit-reversed addressing for Cooley-Tukey FFT

• Advantage
  – software programmable ⇒ no silicon NRE

• Disadvantage
  – performance sacrificed
Software Programmable DSPs

- DSP Instruction set defined by device architects
- Application performance depends on
  1. match between computations required in application and type/number/interconnectivity of microprocessor functional units
  2. how well your algorithm maps onto the instruction set
  3. compiler technology
- Mismatch between computational requirements and processor architecture ⇒ WASTED RESOURCES
XC4000 FPGA Architecture
XC4000X Interconnect Hierarchy

Quad  Long  Global Clock  Long  Double  Single  Global Clock  Carry Chain  Direct Connect

New in XC4000EX
XC4000 E

8/6/98 Dr Chris Dick
FPGA DSP - *It’s about time*

- FPGAs are like miniature *silicon foundries* with extremely short turn-around times
- FPGAs open-up the solution space for digital signal processing engineers by decoupling the design process from the preconceptions of the VLSI DSP chip designer
  - Word length - different word lengths are appropriate for different problems or different parts of the same problem
  - Functional unit type and number (parallelism)
  - Functional unit connectivity
  - Memory ⇒ depth, width and bandwidth
FPGA DSP

• What digital radio functions can FPGAs be used for?
  – channelizing
  – adaptive equalizer
  – modulation/demodulation
  – trellis encoder/ Viterbi decoder
  – interleaving/deinterleaving
  – Reed-Solomon encoding and decoding
  – encryption/decryption
Why FPGA DSP?

- Using FPGAs brings an entirely new dimension to DSP hardware
- FPGA DSP hardware is like *liquid hardware*
  - functionality defined by downloading configuration bitstream that defines datapath
- Reconfigurable computing technology ⇒ *Reconfigurable Receiver*
- **ONE PIECE OF HARDWARE CAN BE USED FOR MANY APPLICATIONS**
FPGAs Put the Silicon Back in Your Hands

- Designer can make space-time tradeoffs based on system requirements

![Graph showing performance versus space with Bit-Serial, Bit-Slice, and Word Parallel categories]
What do you do when...

the fastest DSP Processor Is Not Fast Enough?

- Design a custom gate array?
- Add more DSP processors?
  - FPGA-based DSP system
  - CPU and FPGA
FPGA DSP System

- FPGA DSP engine consisting of FPGA device(s) only
  OR
- FPGA reconfigurable co-processor
Some Examples

- FIR Filters
- FFT
- Digital receiver
- Adaptive filter
FIR Filters

- Conventional approach using DSP processors

Sequential Processing

Single MAC

Multiply

Add

- Programmable
- Off-the-shelf, standard part
- Hardware multiplier

- One MAC (Multiply Accumulate)
- Time-Shared
- Performance ceiling
Performance Through Parallel Processing

DSP Processor

RAM

Peripherals

CPU & MAC(s)

Peripherals

RAM

ROM

ROM

Time-share 1 or 2 or 4 MACs

Xilinx FPGA

As many MACs in parallel as you need

8/6/98 Dr Chris Dick
FIR Filters Using *Distributed Arithmetic (DA)*

- DA Filter mechanization

![Diagram of FIR Filter Using Distributed Arithmetic]

- **x(n)**
- **SR**
- **Lookup Table**
- **SCALING ACCUMULATOR**
- **Add/Sub**
- **REG**
- **y(n)**
- **clock**

SR = Shift register
REG = register

8/6/98 Dr Chris Dick
Distributed RAM & Distributed Arithmetic: Perfect Match

DA Algorithms:
- 4-Input Look-Up-Tables (LUT)
- For higher performance
  Use more LUTs
  = more parallelism
- Efficiency similar to custom solution
  Achievable with LUT logic
  More ASIC gate equivalents
  More cost effective
Xilinx DSP
High Performance Alternative - Parallel Processing

+ Programmable
+ Off-the-shelf, standard part
+ Many Multiplies in one clock cycle!
+ Extend the performance of DSP Processors

Multiple MACs, Parallel Processing
An Example: 120 Million Samples per Second
512-Tap Decimating FIR

- 3.8 Billion MACs
  - >10 DSP uPs
- 5,120 Flip-Flops
  - Just for data buffer
- XC4085XL
- 150,000 Gates
## Serial Distributed Arithmetic FIR Filters

<table>
<thead>
<tr>
<th># CLBs</th>
<th>5 bit</th>
<th>8 bit</th>
<th>10 bit</th>
<th>12 bit</th>
<th>14 bit</th>
<th>16 bit</th>
<th>18 bit</th>
<th>20 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td>33</td>
<td>36</td>
<td>39</td>
<td>42</td>
<td>45</td>
<td>52</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Non</td>
<td>46</td>
<td>54</td>
<td>59</td>
<td>64</td>
<td>69</td>
<td>77</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>16 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td>53</td>
<td>61</td>
<td>69</td>
<td>71</td>
<td>76</td>
<td>81</td>
<td>96</td>
<td>102</td>
</tr>
<tr>
<td>Non</td>
<td>60</td>
<td>95</td>
<td>104</td>
<td>112</td>
<td>123</td>
<td>138</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>24 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td>80</td>
<td>89</td>
<td>101</td>
<td>108</td>
<td>116</td>
<td>127</td>
<td>146</td>
<td>154</td>
</tr>
<tr>
<td>Non</td>
<td>101</td>
<td>114</td>
<td>127</td>
<td>140</td>
<td>153</td>
<td>174</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>32 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td>93</td>
<td>107</td>
<td>118</td>
<td>126</td>
<td>137</td>
<td>148</td>
<td>175</td>
<td>182</td>
</tr>
<tr>
<td>Non</td>
<td>116</td>
<td>138</td>
<td>154</td>
<td>165</td>
<td>179</td>
<td>191</td>
<td>226</td>
<td>239</td>
</tr>
<tr>
<td>40 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td>158</td>
<td>173</td>
<td>187</td>
<td>202</td>
<td>217</td>
<td>246</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>Non</td>
<td>197</td>
<td>215</td>
<td>233</td>
<td>250</td>
<td>268</td>
<td>305</td>
<td>323</td>
<td></td>
</tr>
<tr>
<td>48 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80 tap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Sample Rate

<table>
<thead>
<tr>
<th>Sample Rate</th>
<th>5 bit</th>
<th>8 bit</th>
<th>10 bit</th>
<th>12 bit</th>
<th>14 bit</th>
<th>16 bit</th>
<th>18 bit</th>
<th>20 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symm</td>
<td>13.3</td>
<td>8.9</td>
<td>7.3</td>
<td>6.2</td>
<td>5.3</td>
<td>4.7</td>
<td>4.2</td>
<td>3.8</td>
</tr>
<tr>
<td>Non</td>
<td>16.0</td>
<td>10.0</td>
<td>8.0</td>
<td>6.7</td>
<td>5.7</td>
<td>5.0</td>
<td>4.4</td>
<td>4.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XC4000E-1</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
<th>MHz</th>
</tr>
</thead>
</table>
Greater than 10x DSP uP Performance

16-bit FIR Filter Benchmark

GIGA-MACs

High-Performance DSP uP

S30  S40  4036  4062  4085  40125

Greater than 10x DSP uP Performance
Price

<table>
<thead>
<tr>
<th>Price per Million MACs per Second</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.25</td>
<td>DSP Processor</td>
<td>$0.10</td>
</tr>
<tr>
<td>$0.20</td>
<td>Xilinx XC4000XL</td>
<td>$0.05</td>
</tr>
<tr>
<td>$0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0.05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8/6/98 Dr Chris Dick
High Performance at a Fraction of the Cost

* Prices based on 50,000 PCS
... And with Faster Time-to-Market
Generic Digital Receiver
Alternative Approach - $\Sigma\Delta$ Signal Conditioner

- IF sampling collects data at a high rate with expectation that the band selection and bandwidth reduction will in the DSP block
- If output bandwidth is a small fraction of collected bandwidth, can argue that desired signal has been oversampled
- Use tunable $\Sigma\Delta$ modulator to requantize input to a small number of bits while preserving SNR in selected band
- Subsequent processing can proceed with reduced complexity hardware
Digital Receiver - $\Sigma \Delta$ Front-End

- This architecture highlights potential offered by FPGAs to realize a solution not available using software programmable DSPs.
Channelizer with $\Sigma\Delta$ Front-End
**ΣΔ Modulator Architecture**

- Quantizing input samples to a lower precision before polyphase filter avoids multiplies ⇒ area efficient design

\[
\hat{X}(z) = X(z) + Q(z)(1 - P(z))
\]

In spectral region of interest \(|P(z)|=1\) and \(P(z)\) has leading phase

\[Q(z)(1 - P(z)) = 0\] and \(\hat{X}(z) \approx X(z)\) in bandwidth of interest
Predictor Filter Frequency Response

• Center frequency $= 0.375 \, f_s$

- 0 dB gain in region of interest
- Leading phase - predictor
Performance

- Input spectrum - 3 tones
- Required to recover tone at $0.375 f_s$
- Re-Quantized Spectrum
  - dynamic range preserved in region of interest and sacrificed in spectrum *don’t care region*
Performance

Filter Response

Filter Output

8/6/98 Dr Chris Dick
Receiver with ΣΔ Pre-processor

- No need for full multipliers in polyphase filter
- 2 CLBs per tap using bit-serial approach
  - (independent of coefficient precision)
- 12 CLBs per tap using 12b coefficients parallel approach
- Example highlights the architectural tradeoffs possible using FPGA DSP that are not accessible using software DSP
High-Performance FPGA Adaptive Filter

• Decision-directed equalizer
Adaptive Filter

- Consider inner-product and coefficient update

\[ c_{k+1} = c_k + \mu \varepsilon_k x_k \]
Adaptive Filter

- Build adaptive FIR with filter slices
- These can be employed in a parallel arrangement to provide higher performance
- Adder-tree used to combine slice outputs

- 16-b precision slice 246 CLBs
- 12-b precision slice 151 CLBs
Adaptive FIR Performance

- Space-time tradeoffs highlighted
- 16-bit precision, 70 MHz clock

<table>
<thead>
<tr>
<th>$N$</th>
<th># SLICES</th>
<th>$f_s$ (MHz)</th>
<th>CLBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>2.2</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.4</td>
<td>501</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>8.8</td>
<td>1011</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>17.6</td>
<td>2024</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>35.2</td>
<td>4071</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>1.1</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.2</td>
<td>501</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.4</td>
<td>1011</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8.8</td>
<td>2024</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17.6</td>
<td>4071</td>
</tr>
</tbody>
</table>
Adaptive FIR Performance

- Space-time tradeoffs highlighted
- 12-bit precision, 80 MHz clock

<table>
<thead>
<tr>
<th>$N$</th>
<th># SLICES</th>
<th>$f_s$ (MHz)</th>
<th>CLBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>2.5</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5.0</td>
<td>501</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10</td>
<td>1011</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>20</td>
<td>2024</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>40.0</td>
<td>4071</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>1.25</td>
<td>246</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.5</td>
<td>501</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5.0</td>
<td>1011</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>10.0</td>
<td>2024</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>20.0</td>
<td>4071</td>
</tr>
</tbody>
</table>
FPGAs, FFTs and OFDM

- OFDM modulation and interpolation implemented by zero-extended FFT
FPGA FFT Performance

- 16-bit complex input/output
- 16-bit phase factors
- Table highlights FPGA capacity for Space-Time tradeoff

<table>
<thead>
<tr>
<th>$N$</th>
<th>CLB COUNT / EXECUTION TIME $\mu$s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLBs</td>
</tr>
<tr>
<td>16</td>
<td>1512</td>
</tr>
<tr>
<td>32</td>
<td>800</td>
</tr>
<tr>
<td>64</td>
<td>1295</td>
</tr>
<tr>
<td>256</td>
<td>1855</td>
</tr>
<tr>
<td>1024</td>
<td>1495</td>
</tr>
</tbody>
</table>

8/6/98 Dr Chris Dick
System Design Using FPGAs

- Schematic capture, HDLs
- High-level modules - COREs - available to enhance system design, development, productivity
  
  Decrease time-to-market

- Some DSP CORE examples
  - FFTs
  - FIR filters
  - Reed-Solomon
  - DDS
  - many others
DSP LogiCOREs Exploit FPGA Architecture

Matrix of 16 by 1 RAM primitives
- Look-up-table logic
- FIFOs, shift-registers, …
- Multiple small memories

10,000 RAM primitives on a chip
Regular, monolithic, scalable structure
Efficient: 1 - 3 Million MACs per CLB
CORE Generator Design Flow
DSP CORE Generator Outputs

- Schematic symbol
- VHDL or Verilog HDL instantiation code
- Simulation model
- Design netlist with constraints

Predictable Performance regardless number of cores

8/6/98 Dr Chris Dick

© XILINX®
Predictable Size & Performance

- Built for System Performance - Not Benchmarks.
- Generated with RPM (Relationally Placed Macro).

RPM Macro Level Advantages
- Predictable size.
- Close proximity of communicating elements
- Alignment of Critical paths
- Accessible I/O signals
- Improves Density

RPM System Level Advantages
- Rapid progress for automatic and manual design methods (1 macro, NOT 100’s of elements!)
- Consistent performance anywhere on the die.
- Packing density very high
- Adequate set-up times

Filling a device with Xilinx Cores does not reduce performance

8/6/98 Dr Chris Dick
Performance Independent of core location

- Xilinx LogiCOREs deliver the same performance for any placement
- Non-segmented routing FPGAs can’t do this
Performance Independent of Device Size

- Same performance for a 4005 or 4085
- Non-segmented routing FPGAs can’t do this
Performance Independent of Device Utilization

- Xilinx has performance independent of the number of cores added
- Non-segmented routing FPGAs can’t do this
Segmented Vs Non-Segmented Routing

Segmented = More Predictable and Repeatable
Segmented Routing = Low Power

**Xilinx FPGA Architecture**

- Lower capacitance on short lines means lower power

**Other Architecture**

- Higher capacitance on each net means higher power

*Xilinx* routing is less than 2/3 non-segmented FPGA power - assuming same process
Segmented Interconnect Yields Lower Power

Segmented = Lower Power, Faster Operation
Power Dissipation Advantage
Often the Limiting Factor In DSP

- Xilinx Advantage over competitive FPGAs
  - Segmented routing is essential in DSP applications
  - 3:1 advantage over non-segmented architecture
- Xilinx advantage over DSP processors:
  - Half the power of popular DSPs
    - Independent study by Stanford
New Devices for FPGA DSP - *Virtex*

- **CLB**
- **PLL**
- **Segmented routing**
- **66 MHz PCI**
- **SelectI/O Pins**
- **Block SelectRAM Memory**
- **SSTL3**
- **Vector Based Interconnect delay=f(vector)**
- **Distributed SelectRAM Memory**

8/6/98 Dr Chris Dick
Virtex Configurable Logic Block

- Polarity of all control signals selectable
- Fast arithmetic and multiplier circuitry
- Optimized for synthesis

8/6/98 Dr Chris Dick
Segmented Routing Interconnect

- Fast local routing within CLBs
- General purpose routing between CLBs
- Fast Interconnect
  - 8ns across 250,000 system gates
- Predictable for early design analysis
- Optimized for five layer metal process
Simplified CLB

- 4 Logic Cells per CLB
- Carry logic (2 independent chains)
- 4 FFs/Latches, 2 BUFTs per CLB
- 16 bits of SelectRAM per LUT (single / dual-port)
Block RAM

- Configure as: 4096 bits with variable aspect ratio
- 8-32 blocks per device
- True dual-port, fully synchronous operation
  - Cycle time <10 ns
- Flexible block RAM configuration
  - 5 blocks: 2K x 10 video line buffer
  - 1 block: 512 x 8 ATM buffer (9 frames)
  - 4 blocks: 2K x 8 FIFO
  - 9 blocks: 4K x 9 FIFO with parity
Conclusion

• FPGAs
  – maintain flexibility with high-performance
  – track standards evolution

• Performance through parallelism
  ⇒ high sample rates
  ⇒ high BW
  ⇒ concurrent I/O

• FPGA based design opens new-dimensions in the design space
  – use bit-width appropriate for your design and for different modules in same design
Conclusion (cont’d)

- CORE generator facilitates DSP datapath implementation
- System level tools
  - Elanix System View
  - others under development
Conclusion (cont’d)

• New architectures - Virtex
• Virtex 1000  
  – 0.25 micron  
  – 75 million transistors  
  – 100 MHz  
• DSP support  
• Partially reconfigurable for hardware sharing