RFNoC
Network on Chip for Easy FPGA Development

Matt Ettus
<matt@ettus.com>
Ettus Research

2015
# USRP FPGA Capability

<table>
<thead>
<tr>
<th></th>
<th>Gen 1</th>
<th>Gen 2</th>
<th>Gen 3 E300</th>
<th>Gen 3 X310</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPGA</strong></td>
<td>Cyclone I</td>
<td>Spartan 3</td>
<td>Zynq</td>
<td>Kintex 7</td>
</tr>
<tr>
<td><strong>Logic Cells</strong></td>
<td>12K</td>
<td>53K</td>
<td>85K</td>
<td>406K</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>26KB</td>
<td>252KB</td>
<td>560KB</td>
<td>3180KB</td>
</tr>
<tr>
<td><strong>Multipliers</strong></td>
<td>NONE!</td>
<td>126</td>
<td>220</td>
<td>1540</td>
</tr>
<tr>
<td><strong>Clock Rate</strong></td>
<td>64 MHz</td>
<td>100 MHz</td>
<td>200 MHz</td>
<td>250 MHz</td>
</tr>
<tr>
<td><strong>Total RF BW</strong></td>
<td>8 MHz</td>
<td>50 MHz</td>
<td>128 MHz</td>
<td>640 MHz</td>
</tr>
<tr>
<td><strong>Free space</strong></td>
<td>none</td>
<td>~50%</td>
<td>~75%</td>
<td>85+%</td>
</tr>
</tbody>
</table>
The Challenges

- FPGA computational capability scales with Moore’s Law but our ability to use it does not
- Domain experts in algorithms not necessarily skilled in FPGA design, networking issues, verilog, or the internals of a particular radio system
- Poor reusability of computational elements across multiple designs due in part to lack of standard interfaces
- Difficulty migrating CPU code to FPGA
- FPGA reconfigurability is on a very slow time scale
  - Seconds to load a configuration
  - Minutes to hours to compile a new configuration
  - Hours to months to develop a new design
- Difficulty scaling designs beyond one FPGA
- Highly interconnected large designs make it difficult to meet timing requirements and slow down compile cycles
The Opportunities

- Massive new FPGAs allow a lot of freedom
  - Don’t have to design like a “rewritable ASIC”
- Considering and valuing *composability* of blocks and code reuse early in the process can lead to large productivity gains
- Be able to take advantage of high level design languages and tools
  - can provide a better way for algorithm experts to express computation than in Verilog or VHDL
  - MyHDL, System Verilog, LabVIEW FPGA, VivadoHLS, Coregen, Simulink, etc.
The Vision
RF NoC

CPU

PCIe

ETH/IP/UDP

ETH0

ETH1

Radio0

Radio1

Radio Transport Router

Computation Engines
RFNoC in GNU Radio
RF NoC Principles

- Distributed asynchronous implementation of Kahn Process Networks
- Protocol has some similarities to Rapid IO
- Very simple Interface/API – 64-bit AXI FIFO in and out
- All communication is packet-oriented over FIFO interfaces
- Data (baseband samples, symbols, packets, etc.) and Control are carried over the same path
- Data and Control carried in the same packet format
- All endpoints are created equal
  - Any-to-any communications
  - No “host” is necessary
- All communication is flow-controlled (both fine and coarse)
- Each block can be in its own clock domain
Radio Blocks

- Contain everything which must happen in the sample clock domain
  - All precise timekeeping and sequencing
  - All control of radio hardware
  - All sample-rate DSP (DUC, DDC, IQ Balance, most sample rate conversion)

- Produces and consumes packets of baseband samples at a fixed rate
Computation Engines

- Can perform arbitrary processing on both data and control
  - Generic FIR engine
  - FFT machine
  - OFDM sync
  - Turbo decoding
  - Frequency hopping state machine
  - AGC state machine
  - Protocol processing
  - Upper layer stacks
  - CPU (i.e. Microblaze)
  - Large memory buffers
  - etc.
External Interfaces

- Packets entering or leaving FPGA via
  - 1G/10G Ethernet
  - PCIe
  - USB3
  - AXI interfaces to ARM and System RAM in Zynq
  - Adaptation layer to other processing paradigms (i.e. massive multicore, GPU, etc.)

- Filters out non-RFNoC traffic and passes it to the control processor
  - e.g. ARP, Ping
  - Out-of-band control like setting up the network router, misc. hardware controls, etc.
  - Network diagnostics

- Strips off other protocol layers and compresses headers on ingress, reverse operation on egress

- Maps RFNoC address to external protocol address (i.e. MAC and IP addresses and UDP port)
Network Fabric

- Full crossbar to route packets between blocks
- Routing tables are loaded out of band by the control processor, and are set up by the application
- All routing decisions are made entirely based on the first line of every packet
- No packet is allowed into the crossbar until it is complete
  - Prevents slow packets from causing congestion
  - Ensures data flows through at full rate
- Flow control assures that any packet entering the router can also exit because there is buffer space guaranteed on the other side
Simple MIMO Example
Status

- Working implementation of radios, network fabric, external interfaces, and flow control across multiple products with a single code base
- A number of interesting computation engines have been implemented
- Automatic setup and routing based on a GNU Radio flowgraph
- Automatically builds FPGA image with user selected computation engines

To Do

- Implement more interesting computation engines
- Demonstrate multi-FPGA flowgraphs
- Take advantage of partial reconfiguration
- Produce skeleton reference computation engines for other design environments
  - MyHDL, LabVIEW FPGA, Simulink, VivadoHLS, etc.
- Automatically migrate processing from host to computation engines