Me, Myself and High Performance Network Functions for Programmable Dataplanes

Salvator Galea
Introduction

Salvator Galea: 1\textsuperscript{st} year student for CPGS (PhD)

Andrew W. Moore: Supervisor, Reader

Gianni Antichi: 2\textsuperscript{nd} Advisor, Senior Research Associate

Department of Computer Science and Technology
NetOS group, SRG
Introduction

Salvator Galea: 1st year student for CPGS (PhD)
Andrew W. Moore: Supervisor, Reader
Gianni Antichi: 2nd Advisor, Senior Research Associate
Introduction

Salvator Galea: 1st year student for CPGS (PhD)
Andrew W. Moore: Supervisor, Reader
Gianni Antichi: 2nd Advisor, Senior Research Associate
Roberto Bifulco: Industrial Advisor, Senior Researcher
Agenda

- **Net**FPGA project
- Emu
- OSNT
- Future work
NetFPGA

A line-rate, flexible, open-networking platform for teaching and research
NetFPGA

A line-rate, flexible, open-networking platform for teaching and research
So what?

- Network Interface Card
- Hardware Accelerated Linux Router
- IPv4 Reference Router
- Traffic Generator
- Openflow Switch
- More Projects
- Add Your Project
NetFPGA

So who, how, why?

- Researchers, Teachers, Students
- To build modular designs
- To prototype new network systems and measure network performance
NetFPGA SUME Community (since Feb 2015)
Over 600 users, using over 300 cards at
200 universities in 47 countries
Emu: Rapid Prototyping of Networking Services
Published in USENIX ATC'17

Using FPGAs for acceleration

Using FPGA is great because of...

- Programmability
- Performance
- Predictability
- Power efficiency

But why FPGAs have never became mainstream?
Programming FPGAs

– It is because of the cost doing FPGA engineering.

– The programming and reprogramming is done in complex, low-level hardware description languages like Verilog and VHDL.

– Lack of FPGA developers compared to number of software developers.

Led to the development of High Level Synthesis tools
  • Use High-level programming languages
  • For Scientific Applications
Kiwi

- HLS open-source compiler
- Transforms C# and F# to HDL
- Any .NET bytecode to Verilog

Accelerating network services

- Write the network application in C#
- Compile to Verilog using Kiwi Compiler
- Run on the FPGA
- End of story :)

15
Accelerating network services

- Write the network application in C#
- Compile to Verilog using Kiwi Compiler
- Run on the FPGA
- End of story :)

Kiwi is just a compiler, not a linker, neither provides networking libraries
EMU
The Network Library for Kiwi
Emu: Accelerating Network Services

**Goal**: Rapid prototyping of network services

Emu provides a framework with:
- A library of functions
- Compiling to multiple targets
- A runtime environment
- Automatic implementation on FPGA
- Advanced debug capabilities
Emu complements a high-level synthesis compiler with a library to support network-related high level programming.

Emu maps programming and networking abstractions to bus protocols, memory interfaces, and basic frame-handling functionality used on the NetFPGA-SUME.

Emu framework provides a reference design path through which the user can compile the C# code and run it directly on the NetFPGA-SUME without further development intervention.
Emu: Hardware Integration

- Multiple FPGA targets using templates
- Support integration with hardware IP cores
- Support of multicore Emu cores
- Library of networking functions

`NetFPGA` Datapath
Emu: Efficiency Comparison – Layer-2 Switch

<table>
<thead>
<tr>
<th></th>
<th>Emu</th>
<th>Verilog</th>
<th>P4FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic resources</td>
<td>3509</td>
<td>2836</td>
<td>24161</td>
</tr>
<tr>
<td>Memory resources</td>
<td>118</td>
<td>87</td>
<td>236</td>
</tr>
<tr>
<td>Latency [cycles]</td>
<td>8</td>
<td>6</td>
<td>85</td>
</tr>
<tr>
<td>Throughput [Mpps]</td>
<td>59.52</td>
<td>59.52</td>
<td>59.52</td>
</tr>
</tbody>
</table>
**Emu: Use Cases**

- Networking devices
  - Layer-2 Switch

- Network Services
  - NAT
  - DNS Server
  - ICMP echo reply

- Performance sensitive applications
  - Memcached Server
### Emu: Use cases evaluation – Throughput

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Emu [mq/s]</th>
<th>Host [mq/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP Echo</td>
<td>3.226</td>
<td>1.068</td>
</tr>
<tr>
<td>TCP Ping</td>
<td>2.105</td>
<td>1.012</td>
</tr>
<tr>
<td>DNS</td>
<td>1.176</td>
<td>0.226</td>
</tr>
<tr>
<td>NAT</td>
<td>2.439</td>
<td>1.037</td>
</tr>
<tr>
<td>Memcached</td>
<td>1.932</td>
<td>0.876</td>
</tr>
</tbody>
</table>
Emu: Use cases evaluation – Latency

![Graph showing latency comparison between Emu and Host for different use cases: ICMP Echo, TCP Ping, DNS, NAT, and Memcached.]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP Echo</td>
<td>1.09</td>
<td>1.11</td>
<td>12.28</td>
<td>22.63</td>
</tr>
<tr>
<td>TCP Ping</td>
<td>1.27</td>
<td>1.29</td>
<td>21.79</td>
<td>65.00</td>
</tr>
<tr>
<td>DNS</td>
<td>1.82</td>
<td>1.86</td>
<td>126.46</td>
<td>138.33</td>
</tr>
<tr>
<td>NAT</td>
<td>1.32</td>
<td>1.34</td>
<td>2444.76</td>
<td>6185.27</td>
</tr>
<tr>
<td>Memcached</td>
<td>1.21</td>
<td>1.26</td>
<td>24.29</td>
<td>28.65</td>
</tr>
</tbody>
</table>
Every program comes with bugs
Emu: Extended Debug Capabilities

› Use *Directed Packets* to inspect the state of a device in the field

› Support *extension points* in the code
  – Observe the program from that point
  – Influence program state

› Example supported commands:
  – Print, trace, count, (un)break,(un)watch, backtrace

› Implemented using *an embedded controller* and *a program director*
### Emu: Debug core overhead

<table>
<thead>
<tr>
<th>Artefact</th>
<th>Utilisation (%)</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logic</td>
<td>Latency</td>
</tr>
<tr>
<td>DNS</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>+R</td>
<td>103.40</td>
<td>100.00</td>
</tr>
<tr>
<td>+W</td>
<td>115.05</td>
<td>99.45</td>
</tr>
<tr>
<td>+I</td>
<td>109.79</td>
<td>99.45</td>
</tr>
<tr>
<td>Memcached</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>+R</td>
<td>99.17</td>
<td>100.00</td>
</tr>
<tr>
<td>+W</td>
<td>99.80</td>
<td>100.49</td>
</tr>
<tr>
<td>+I</td>
<td>100.63</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Emu: Conclusion

So what you get with the Emu framework?

- Rapid prototyping of networking services
- Code in .NET, compile to multiple targets
- Accelerates the development and debug process
- High throughput and low latency
- Open source
OSNT: Open Source Network Tester

Open source hardware and software platform for network monitoring and testing

https://osnt.org

Low cost, flexible to update, scale-out, no CPU usage, nanosecond resolution measurements
OSNT

- 4x10Gbps traffic generator.
- Capture card with high resolution timestamp (6.4nsec).
- GPS-ready synchronized measurement kit.
Future work

Hybrid solution for network applications
– Parts of the application run on HW
– Parts of the application run in host

Customized per network application generator and monitor tool
– more accurate latency predictor
– interactive traffic generator
Questions?