# Emu: Rapid FPGA Prototyping of Network Services in C#

Salvator Galea*, Nik Sultana*, Pietro Bressana†, David Greaves*,
Robert Soulé†, Andrew W Moore*, Noa Zilberman*
*University of Cambridge, Email: *firstname.lastname*@cl.cam.ac.uk
†Università della Svizzera italiana, Email: *firstname.lastname*@usi.ch

***Abstract*—General-purpose CPUs and OS abstractions impose overheads that make it challenging to implement network functions and services in software. On the other hand, programmable hardware such as FPGAs suffer from low-level programming models, which make the rapid development of network services cumbersome. We demonstrate Emu, a framework that makes use of an HLS tool (Kiwi) and enables the execution of high-level descriptions of network services, written in C#, on both x86 and Xilinx FPGA. Emu therefore opens up new opportunities for improved performance and power usage, and enables developers to more easily write network services and functions. We demonstrate C# implementations of network functions, such as Memcached and DNS Server, using Emu running on the NetFPGA-SUME platform, and show that they are competitive to natively written hardware counterparts while providing a superior development and debug environment.**

## I. The Emu framework

Emu is a framework for network functions on FPGAs. Emu builds upon the Kiwi compiler [1], which allows computational scientists to program FPGAs with .NET code. The relationship with Emu to .NET/Kiwi is roughly analogous to that of the stdlib to C/GCC: Emu provides the implementation for essential network functionality. The combination of Emu and HLS provides a powerful substrate for developers to rapidly implement and deploy network functions using a high-level language.

Moreover, Emu virtualizes the hardware context of the network pipeline, allowing users to write code that is portable across several different heterogeneous targets. Our current implementation supports CPUs, simulation environments and FPGAs. Using Emu, developers can run their network functions as normal processes, using virtual or real NICs, and using network simulators, simplifying debugging and testing.

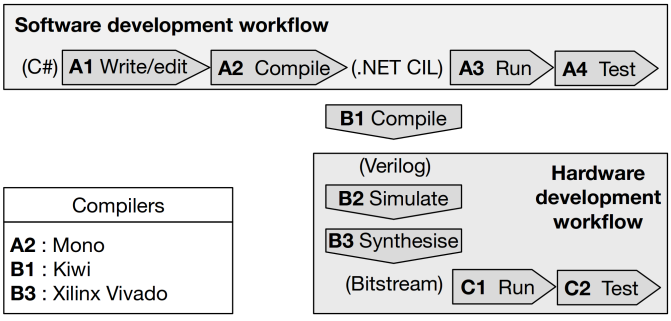


Fig. 1. Emu workflow

```
1  // If the frame does not contain an IPv4 packet then we implicitly
       drop the frame.
2  if (dataplane.tdata.EtherType_Is(EtherTypes.IPv4))
3  {
4    //Set the appropriate output port in the metadata
5    if (dstmac_lut_hit) {
6      NetFPGA.Set_Output_Port(ref dataplane, lut_element_op);
7    } else {
8      NetFPGA.Broadcast(ref dataplane);
9    }
10 }
11 Kiwi.Pause();
12 // Learn new source MAC addresses
13 if (!srcmac_lut_exist)
14 {
15   LUT[free] = srcmac_port;
16   free = (free > (LUT_SIZE - 1)) ? 0 : free++;
17 }
```

Fig. 2. Part of a switch implementation showing use of our API for protocols (Line 2) and NetFPGA (Line 6).

Using NetFPGA [2] as a hardware target, Emu is made available under an open-source license, and the projects developed under it are contributed to the NetFPGA community.

## II. Demonstration

In the demo, we present the development flow using Emu, from the C# level to the hardware. We demonstrate some of Emu's unique capabilities, such as running the same code in heterogeneous environments. We also demonstrate the integrated debug capabilities embedded in Emu, enabling in-field debug of operational networking devices, developed using Emu. We used Emu to implement different networking services, such as L2 switch, a memcached server, and NAT, and present a live demonstration of some of these designs.

## III. Acknowledgments

This work has received funding from the EPSRC grant EP/K034723/1, Leverhulme Trust Early Career Fellowship ECF-2016-289, European Union's Horizon 2020 SSICLOPS (grant agreement No. 644866) and Swiss National Science Foundation (grant agreement No. 166132).

## References

[1] S. Singh and D. J. Greaves, "Kiwi: Synthesis of FPGA Circuits from Parallel Programs," in *Field-Programmable Custom Computing Machines*, pp. 3–12, 2008.

[2] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as Research Commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, 2014.