DESIGN AND SIMULATION TOOLS FOR EMBEDDED NOCS ON FPGAS

Mohamed S. Abdelfattah, Andrew Bitar, Ange Yaghi and Vaughn Betz

Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada
`{mohamed,bitar,vaughn}@eecg.utoronto.ca, ange.yaghi@mail.utoronto.ca`

ABSTRACT

We propose embedding hard NoCs on FPGAs to improve system-level communication as detailed in our previous studies [1–6]. This demo paper outlines the three main design and simulation tools that we have been using to experiment with Embedded NoCs on FPGAs.

1. `NoC Designer`

`NoC Designer` is an online tool that estimates *raw* efficiency and bandwidth metrics for various soft or hard NoCs and their components. These metrics spurred out of research that investigated the difference between hard and soft NoCs [1, 2, 4, 5]. To measure efficiency, we varied NoC parameters such as link width, number of virtual channels, buffer depth and number of ports per router, then we synthesized the router components on both the 65-nm TSMC standard-cell technology and Altera's Stratix-III FPGAs (that also use the 65nm TSMC process). This gave us a database of area and frequency measurements for each NoC component whether it's implemented soft (out of the FPGA fabric) or hard (in ASIC standard cells). Finally, power was measured using gate-level simulations with uniform traffic patterns on both the FPGA and ASIC platforms, completing the efficiency trinity of area, speed and power.

`NoC Designer` serves as a front-end to all the measured data showing the results both in table and graph form. The user enters NoC parameters and can view the efficiency metrics for *any* NoC. Because the design space is very large, it isn't easy to synthesize all possible NoCs; this is why we only synthesized a representative subset and we linearly interpolate between the results to find the area/speed/power of an unsynthesized NoC. By combining efficiency metrics with available bandwidth, we compute composite metrics such as *energy-per-data* or *area-per-bandwidth*. This allows us to compare NoCs to other forms of interconnect thus quantifying each interconnect's efficiency in utilizing its area/power in moving data. Finally, **`NoC Designer`** also visualizes the floorplan a hard NoC on a modern FPGA taking into account the multiplexers and wires to connect the hard NoC to the FPGA fabric.

2. `RTL2Booksim`

`RTL2Booksim` is a simulation framework that allows the flexible simulation of NoCs with RTL (Verilog/VHDL) designs. The conventional way of doing this is to write/verify an NoC using RTL – a daunting task for a feature-full packet-switched NoC such as the one we use. This is especially difficult to do if we want to be able to vary NoC parameters and explore the design space easily. To circumvent this, we use instead a C-based cycle-accurate NoC simulator called Booksim [7]. To be able to use Booksim within an RTL simulation (such as one using Modelsim), we created an RTL interface using SystemVerilog Direct Programming Interface (DPI). Any of the NoC parameters can be easily configured in a configuration file, and this NoC can be part of an RTL simulation by simply including the interface files. Our recent work used **`RTL2Booksim`** to simulate a JPEG compression application as well as an Ethernet switch and find cycle-accurate performance and latency data [6].

3. `LYNX`

`LYNX` is a design tool whose purpose is to interconnect pieces of a design using either an Embedded NoC or soft interconnect, or a mixture of both. The design is entered either in XML language or graphically (which is then translated into an XML description). In the XML description, a `module` tag is an IP block in the design and it can have any number of `ports` and `bundles`. `Bundles` are a collection of `ports` that contain a ready and valid port thus allowing latency-insensitive communication between `bundles`. Finally, `connection` tags describe the logical connectivity between `bundles`.

`LYNX` parses the application connectivity graph and creates flexible object-oriented data representations that it uses to generate interconnect for the application. The goal is to have an open-source framework for prototyping, comparing and combining different interconnect types. Our research focuses on using **`LYNX`** to connect an application using an embedded NoC for coarse-grained connections. For fine-grained connections, we use a light-weight low-latency interconnect (such as GENIE [8]) implemented using soft logic in the FPGA fabric.

4. LINKS

- `www.eecg.utoronto.ca/~mohamed/noc_designer`
- `www.eecg.utoronto.ca/~mohamed/rtl2booksim`
- `www.eecg.utoronto.ca/~mohamed/lynx`

REFERENCES

[1] M. S. Abdelfattah and V. Betz, "Design Tradeoffs for Hard and Soft FPGA-based Networks-on-Chip," *FPT*, 2012, pp. 95–103.

[2] ——, "The Power of Communication: Energy-Efficient NoCs for FPGAs," *FPL*, 2013, pp. 1–8.

[3] ——, "The Case for Embedded Networks-on-Chip on Field-Programmable Gate Arrays," *IEEE Micro*, vol. 34, no. 1, pp. 80–89, 2014.

[4] ——, "Networks-on-Chip for FPGAs: Hard, Soft or Mixed?" *TRETS*, vol. 7, no. 3, pp. 20:1–20:22, 2014.

[5] ——, "Power Analysis of Embedded NoCs on FPGAs and Comparison With Custom Buses," *TVLSI*, 2015.

[6] M. S. Abdelfattah, *et al.*, "Take the Highway: Design for Embedded NoCs on FPGAs," *FPGA*, pp. 98–107, 2015.

[7] N. Jiang *et al.*, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," *ISPASS*, 2013, pp. 86–96.

[8] A. Rodionov, *et al.*, "Fine-Grained Interconnect Synthesis," *FPGA*, pp. 46–55, 2015.