OpenFPGA: a Complete Open Source Framework for FPGA Prototyping

Baudouin Chauviere*, Aurélien Alacchi, Edouard Giacomin, Xifan Tang, Pierre-Emmanuel Gaillardon
University of Utah, Salt Lake City, Utah, USA
*Email: baudouin.chauviere@utah.edu

***Abstract*—In this paper, we present OpenFPGA, an open-source FPGA IP generator allowing fast prototyping of customizable FPGA fabrics described using a high-level XML architecture description language. OpenFPGA consists of three tools: FPGA-Verilog, FPGA-SPICE and FPGA-Bistream. FPGA-Verilog creates the Verilog netlist of the full FPGA fabric allowing a semi-custom design flow to automatize the layout generation. FPGA-SPICE generates SPICE netlists enabling accurate power and delay analysis using electrical simulations. FPGA-Bitstream provides programming support for the prototyped FPGAs and also enables functional verification during pre-silicon development.**

I. Introduction

The increasing popularity of FPGA-based accelerators, particularly in datacenters demonstrates their neccessity in commercial multicore heterogeneous System-on-Chips. Meanwhile, a large open-source community has been established around RISC-V [1] to democratize industry standard SoC hardware design among academic researcher and even companies, such as Google or Qualcomm. To match the performance of commercial SoC platforms, we strongly believe that the open-source hardware community needs a framework for customizable FPGA IP generation. However, FPGAs are one of the few last digital integrated circuits that require full-custom layouts and floorplaning. Such intensive manual efforts may take academia researchers years to achieve silicon results of their innovative FPGA architectures. Due to this entry barrier, academic FPGA architecture exploration tools, e.g., VPR, rely on analytical models to predict *Performance, Power and Area* (P.P.A.) [2]. Consequently, even though VPR is capable of exploring many innovative FPGA architectures, the complexity of prototyping FPGA chips prohibits researchers from achieving physical proof-of-concept demonstrations. In this paper, we propose OpenFPGA, the first (to the best of our knowledge) open-source FPGA IP generator that enables rapid prototypying of customizable FPGA architectures.

II. OpenFPGA Framework

The OpenFPGA framework is reported in Fig. 1 and consists of three critical parts:

1) *FPGA-Verilog* generates synthesizable Verilog netlists of complete FPGA fabrics, which include the configuration peripheral circuits and the core logic such as *Configurable Logic Blocks* (CLBs), *Connection Blocks* (CBs), *Switch Blocks* (SBs) and I/O blocks. By using a semi-custom design flow (standard cell mapping), the exported Verilog netlists can be translated into physical layouts and be used for silicon prototyping.
2) *FPGA-Bitstream* provides Verilog-to-Bitstream support for the generated FPGA fabrics. First, a Verilog benchmark is synthesized and optimized by Yosys [3]. Then, VPR packs, places and routes the circuit onto the FPGA fabric. Afterwards, FPGA-Bitstream decodes the mapping results on each multiplexer and *Look-Up Table* (LUT) by following their sequence in the scan-chain configuration circuits. Based on the bitstreams, Verilog testbenches are also generated to provide functional verification for FPGA prototypes during the pre-silicon stage.
3) *FPGA-SPICE* outputs SPICE netlists and associated testbenches modeling a configured FPGA fabric. Extracting the parasitics after the generation of the layout allows back-annotation onto the design flow. The use of parasitic-extracted SPICE netlists gives results closer to the silicon power sign-off.

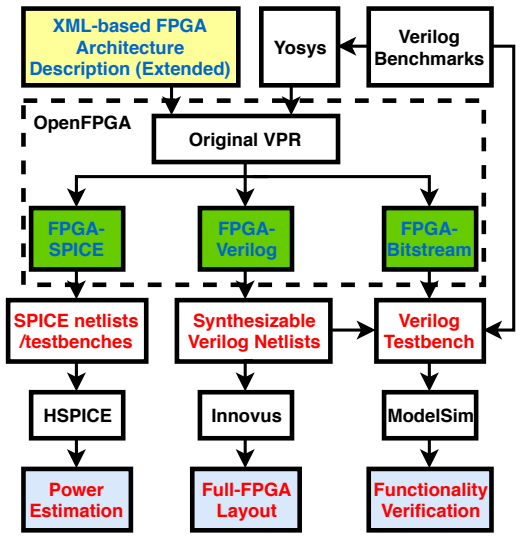

Fig. 1: Detailed OpenFPGA EDA flow extended from VPR

To support flexible FPGA architectures and circuit designs, we extended the XML-based VTR architecture description language by enriching it with parameters capturing transistor- and gate-level circuit modeling. These modifications are exhaustively presented in [4] and partially illustrated by the following description of a SB multiplexer::

```
<circuit_model type="mux" name="sb_mux"/>
  design_technology type="cmos" structure="one-level"/>
  input_buffer exist="on"
circuit_model_name="inv1"/>
  pass_gate_logic circuit_model_name="tgate"/>
  port type="input" prefix="in" size="4"/>
  port type="output" prefix="out" size="1"/>
  port type="sram" prefix="sram" size="4"/>
</circuit_model>
```

Since FPGA accelerators are now fully integrated in SoC developments, having an open-source FPGA IP customized for particular computer architecture requirements allows for

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7855. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

This work was presented at the first Workshop on Open-Source Design Automation (OSDA), 29 March 2019, in Florence, Italy. https://osda.gitlab.io

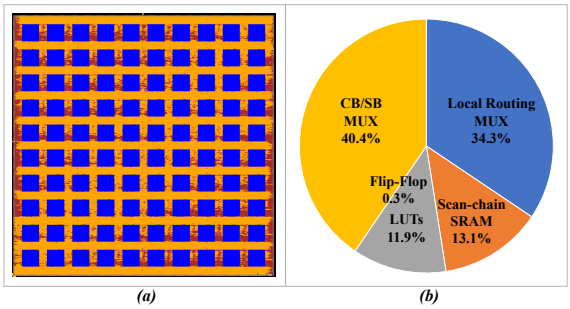

Fig. 2: SRAM-based FPGAs configured by SCFF: (a) full-chip layout, and (b) area breakdown

greater optimization and better hardware software co-design. Layout generation and functional verification are made through commercial tools [1].

OpenFPGA can benefit academic researchers in many aspects: (1) It empowers researchers to prototype innovative FPGA architectures with limited human resources. In addition, it can also be used to validate more effectively novel FPGA *Computer-Aided Design* (CAD) algorithms. (2) It provides a generic evaluation method for any technology node. For instance, by applying industry-standard sign-off tools on FPGA layouts, realistic evaluations on area, delay and power can be derived. (3) It helps to calibrate existing analytical models or develop more accurate ones that are well aligned to different technology nodes. (4) It can spark more research topics on FPGA architecture, such as hotspot management, performance degradation from IR drop, *etc.*, which are hard to be captured by current FPGA architecture exploration tools. The OpenFPGA framework is available for download at [5] under MIT license and permits commercial usage.

III. OpenFPGA Showcases

In this section, we present two case studies using the OpenFPGA flow: (1) a layout generation and area breakdown analysis; and (2) a functional verification for benchmark circuits mapped on the fabric. All the circuit components and FPGA architectures are optimized by considering a commercial 40nm technology. We consider a FPGA architecture similar to an Altera Stratix IV device [6], with a scan-chain-based configuration circuit. The transistor-level circuit designs of SRAMs, FFs and multiplexers are derived from [7]. The channel width, W, is set 300, being similar to commercial FPGAs. The experiments are run on a 64-bit RedHat Linux server with 28 Intel Xeon Processors and 256Gb memory. To fit the capability of our Linux server without losing representativity, we limit our study to a 10×10 FPGA. Examples of architectures and benchmarks are available at [5].

A. Full FPGA Fabric Layout and Area Breakdown

Fig. 2(a) shows the full layout of a SRAM-based FPGA configured by *Scan-Chain Flip-Flop* (SCFF). Fig. 2(b) depicts the area breakdown of the generated fabric. Routing multiplexers take 40.4-34.3% of the total area being the highest contribution. LUTs and FFs stand only up to 12.2% of the total area, while scan-chain and configuration circuit occupy 13.1% of the total area. The FPGA generated has ∼140k memories forming the scan-chain which is similar to a medium/small

[1]Cadence Innovus 17.11 and Mentor ModelSim 10.7b are used to showcase our framework.

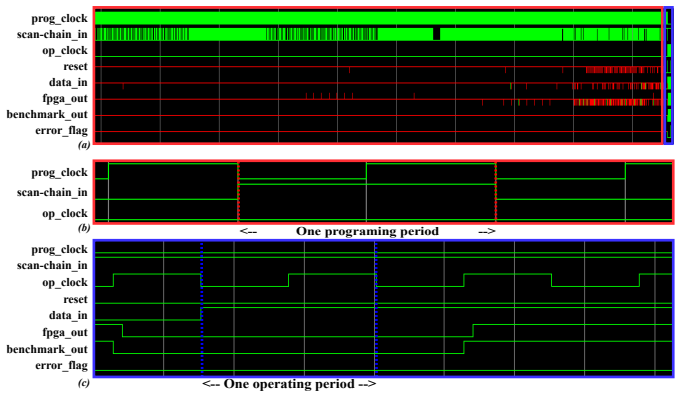

Fig. 3: Waveforms of a shift register, obtained through ModelSim simulation: (a) full waveforms with configuration phase highlighted in the red rectangle and operation phase highlighted in the blue rectangle; (b) an example of a programming clock cycle; (c) an example of a operating clock cycle.

size commercial FPGA, such as iCE40 [8]. Finer tuning of the layout can be achieved through feedback loops on the design allowing updates on the timing and area constraints.

B. Functional verification for Full FPGA Fabric Netlists

To validate both core logic and configuration peripherals, a Verilog testbench generated by OpenFPGA includes two phases:

(1) A configuration phase, where the scan-chain is programmed serially according to the bitstream. Each programming cycle configures the next memory in the chain. During this period, the programming clock is enabled and the data inputs are tied to logic `0`.

(2) An operating phase, where the mapped circuit is tested using test vectors fed to the inputs. During this period, the programming clock is disabled. Fig. 3 shows the waveforms of functional verification of a shift register. The red rectangle highlights the waveform during configuration phase, while the blue rectangle highlights the waveform during the operation phase.

IV. Conclusion

This paper presents OpenFPGA, an open-source IP framework, that is capable of generating synthesizable Verilog netlists of FPGAs, based on which prototypes can be built using semi-custom design approaches. OpenFPGA also implements the full CAD support for the generated prototypes by mapping application circuits on the fabric and exports the associated bitstreams and validation testbenches. We showcased OpenFPGA by (1) presenting a 10×10 FPGA layout area similar to the capacity of a commercial FPGA; and (2) validating a widely-used FIFO component on the mapped full FPGA fabric using HDL simulation.

References

[1] *https://riscv.org/*
[2] J. Rose *et al.*, *The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing*, International Symposium on Field Programmable Gate Array (FPGA), Monterey, California, 2012, pp. 77-86. doi: 10.1109/TVLSI.2018.2883923
[3] C. Wolf, *Yosys: Open Synthesis Suite*, Available online. http://www.clifford.at/yosys/about.html, 2018
[4] *https://openfpga.readthedocs.io/en/master/index.html*
[5] *https://github.com/LNIS-Projects/OpenFPGA*
[6] Altera Corporation, *Stratix IV device handbook version SIV5V1-1.1*, July 2008. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf
[7] V. Betz *et al.*, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1998.
[8] http://www.latticesemi.com/en/Products/FPGAandCPLD/

This work was presented at the first Workshop on Open-Source Design Automation (OSDA), 29 March 2019, in Florence, Italy. https://osda.gitlab.io