Enabling FPGA domain-specific compilers through open source

Chris Lavin, Alireza Kaviani
Xilinx Research Labs., San Jose, CA, USA
chris.lavin | alireza.kaviani@xilinx.com

I. Introduction

Silicon transistors and wires are not getting much better due to the slowdown of Moore's Law, and the power per chip area is increasing (reflecting the end of Dennard scaling). Computer architects are now widely subscribed to domain-specific architectures as being the only path left for major improvements in performance-cost-energy. FPGAs (Field Programmable Gate Arrays) are highly adaptable devices—making them the candidate of choice for a wide range of emerging domains from compute to networking.

Advances in process technology have enabled FPGAs to grow in capacity and implement large heterogenous systems in a device. As a result, future compilers need to go beyond their traditional role of mapping a generic design input to a specific hardware platform. Emerging domain-specific compilers must subscribe to a broader view in which compilers guide both the selection of modules in an application and the customization of hardware components to implement their corresponding tasks. Today, generic vendor backend compilers are the only available mechanism to realize a broad range of applications in many domains. The necessity of breadth coverage by commercial tools often leads to implementations that do not take full advantage of the underlying hardware. Domain-specific compilers, on the other hand, can deliver near-spec performance by taking advantage of both application attributes and architecture details. This suggests the need for a framework capable of interfacing between vendor backend tools and front-end domain architects or compilers.

To address this need, we present RapidWright, an open source platform providing a gateway to Xilinx's back-end implementation tools that raises the implementation abstraction, while maintaining the full potential of advanced FPGA silicon. It works synergistically with Vivado through design checkpoints (DCPs) to produce highly-tuned, custom implementations for emerging domains (see Fig. 1). DCPs are proprietary binary files that include all netlist, place & route, and physical constraints information necessary to generate a bitstream. RapidWright builds on the premise of two key observations: (1) Vendor tools such as Vivado often produce high performance results for small modules of a design. (2) Emerging applications such as compute-bound machine learning designs grow in size by module replication and reuse. Vivado can produce highly optimized implementations for key modules of a design to deliver the highest performance. RapidWright can then replicate, relocate and assemble these tuned modules to compose a complete application while maintaining high performance.

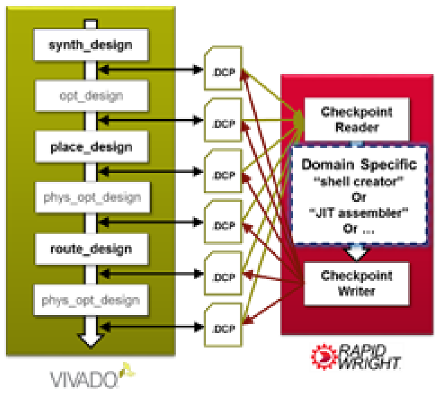


Fig. 1. RapidWright is a gateway to Vivado.

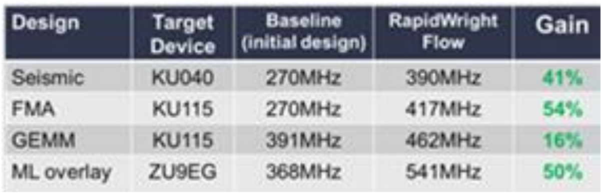
II. RapidWrigh Value Proposition

RapidWright's native gateway to Vivado sets the groundwork for an ecosystem aimed at further advancing FPGA tools. It empowers academic and industry researchers by combining the commercial credibility of FPGA tools with the agility of an open source framework, leading to innovative solutions that might not be feasible otherwise. These innovations need to improve the end figures of merit in either performance, productivity, or predictability for the target domain. A key promise of RapidWright is enabling a modular pre-implemented design methodology. This approach implements large designs by: (1) identifying common denominator modules of the design (restructuring), (2) strategically matching pre-implemented modules with programmable fabric structures, and (3) leveraging an automated RapidWright flow that stitches modules into a final implementation.

Table 1 summarizes our results for four large representative designs, showing more than 50% performance improvement [1]. The RapidWright platform achieves this by employing three key capabilities. First, it preserves high quality placement and routing of pre-implemented blocks. Second, it enables reuse and replication of blocks to leverage Vivado's efforts of achieving high quality results. And third, RapidWright stitches the blocks together with minimal or no loss of quality. Leveraging these capabilities, we demonstrate how to augment Vivado and close the performance gap between device specification and achievable results through vendor tools.

An overarching theme for improving either performance or productivity of FPGA designs has been building overlays or static shells. Overlays such as arrays of soft processors aim

TABLE I. POTENTIAL PERFORMANCE RESULTS



Design	Target Device	Baseline (initial design)	RapidWright Flow	Gain
Seismic	KU040	270MHz	390MHz	41%
FMA	KU115	270MHz	417MHz	54%
GEMM	KU115	391MHz	462MHz	16%
ML overlay	ZU9EG	368MHz	541MHz	50%

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

to raise the design abstraction, improving productivity. It is highly desirable to achieve near spec performance for these shells and overlays in order to improve the end figures of merit in the overall application. RapidWright is an excellent vehicle to build domain-specific shells, especially when customers require many variations.

III. Open source community role

The tasks in building a domain-specific flow are categorized in two main groups: (1) domain design tasks and (2) backend implementation tool tasks. RapidWright offers a foundation to automate the second group of tasks while providing guidelines and a framework for the first group. Domain-specific tasks involve restructuring the designs to emphasize modularity with latency-flexible inter-connectivity. For example, domain 3 applications in Fig. 2 can directly leverage RapidWright because restructuring is done manually. The results in Table 1 include both benefits of restructuring and RapidWright implementation tasks. The best opportunity for domain design tasks lies with domain application architects and the path to automation would require a domain-specific front-end compiler. Such compiler may be an LLVM data flow graph parser that can automatically identify domain operators with high replication. This is depicted in Fig. 2 by application examples in domains 1 and 2.

RapidWright is enabling a new level of optimization and customization for the domain architect to further exploit FPGA silicon capabilities. We invite the open source community to act on this confluence of capabilities (RapidWright and domain specific architectures) to create a new category of compilation flows that match domain needs while being maximally efficient with FPGA implementations. Researchers with interest and expertise in certain domains are the ideal initial collaborators, as summarized in the figure. Once success is achieved in building front-end compilers that achieve near-spec performance on FPGA platforms, a new layer of domain specific languages can be created that further abstracts domain experts from the low level details of architecture. This both boosts productivity and expands the impact of the tool flows by improving their accessibility to a much larger audience. With community growth and after front-end compilers evolve, the domain architects no longer need to be fully aware of the backend implementation and architecture details. This will eventually lead to a broader usage of FPGA devices by domain scientists with less hardware expertise or interest. For additional examples and open source code (with Apache 2.0 license), please visit www.rapidwright.io.

IV. Final remarks

RapidWright allows the FPGA community to earnestly explore the boundaries of performance and productivity, while enjoying the credibility of commercial FPGA tools. With RapidWright enabling fine grained control of interfacing with vendor tools, a new branch of FPGA implementation strategies can emerge---boosting performance beyond previous expectations. Enabling users to capture domain-specific implementations will lead to significant improvements in circuit performance, user productivity and timing closure predictability. Creating reusable, near-optimal, modular implementations with the help of the FPGA community will lead to shells and overlays that will potentially unlock a class of accelerator designs rarely realized.

The proposed tool flow is more amenable to domains with high reuse and latency tolerant kernels. As FPGAs play a larger role in emerging and datacenter-acceleration applications, more designs exhibit these favorable attributes. The open source nature of RapidWright empowers application architects and enables users of FPGAs to explore innovative ways to liberate the full potential of advanced silicon technology. We also envision RapidWright augmented by powerful algorithmic engines---such as SAT and ILP solvers---to realize efficient, localized placement and routing solutions. We anticipate great interest to maximize existing FPGA silicon performance and we believe RapidWright to be the enabler for a significant part of that exploration.

References

[1] C. Lavin and A. Kaviani, “RapidWright: Enabling Custom Crafted Implementation for FPGAs,” IEEE FCCM 2018.

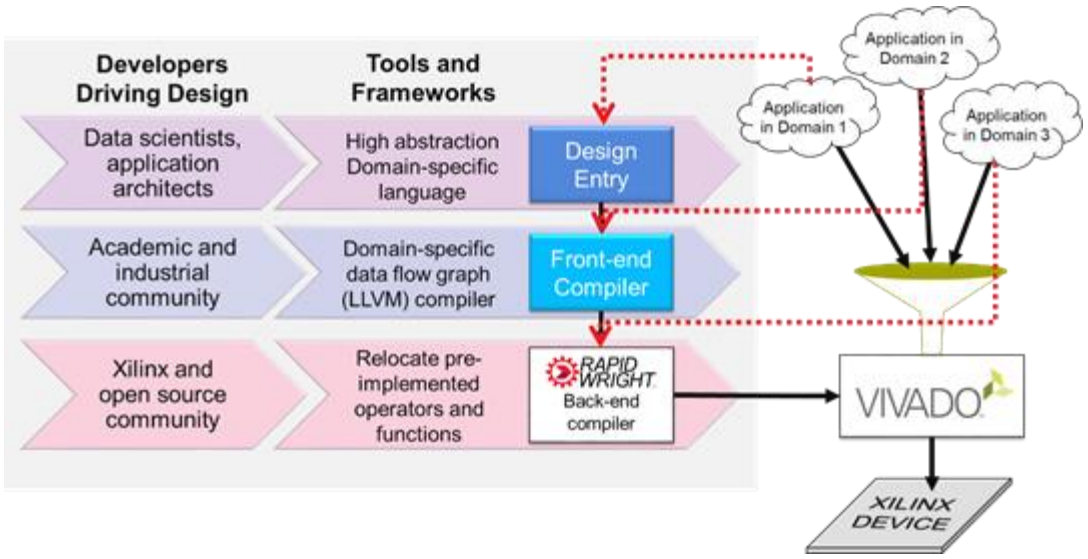

Developers Driving Design
Tools and Frameworks
Data scientists, application architects
High abstraction Domain-specific language
Design Entry
Academic and industrial community
Domain-specific data flow graph (LLVM) compiler
Front-end Compiler
Xilinx and open source community
Relocate pre-implemented operators and functions
RAPID WRIGHT Back-end compiler
Application in Domain 2
Application in Domain 1
Application in Domain 3
VIVADO
XILINX DEVICE

Fig. 2. RapidWright and the community involvement.