DUC: Data Unlimited Compile for FPGAs

Eliminating Memory Constraints for FPGA Accelerators

Penn Engineering, University of Pennsylvania | Electrical and Systems Engineering

Team 25

Matthew Hofmann CMPE (matth2k@seas)
Zhiyao Tang CMPE (zhiyaot@seas)
Aaron Shurberg CMPE (aaronshu@seas)
Richard Chen CMPE (laurelin@seas)
Advisor: Prof. André DeHon (andre@acm.org)

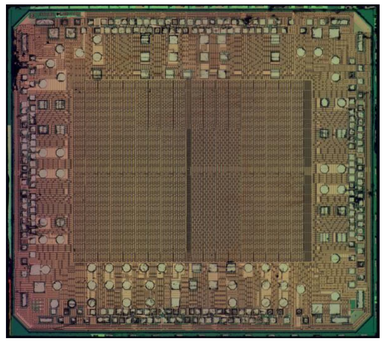
Abstract

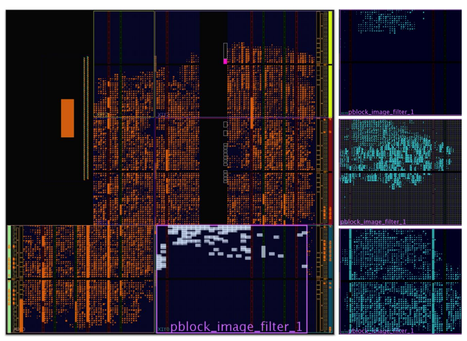
DUC: Data Unlimited Compile for FPGAs is a prototyping design flow that allows for fast-mapping of operators to the FPGA fabric without any data size constraints. HLS C code is diverted from logic synthesis and instead compiled onto RISC-V softcores that fetch memory from DRAM. A typical ratio of DRAM to embedded RAM capacity is 2000:1. Hence, utilizing off-fabric RAM eliminates the data size constraints of prototyping HLS designs for FPGA platforms. By using DUC and PRflow by Xiao[1], complex designs now fit on smaller chips and compile times are reduced from hours or days to minutes.

[1] http://ic.ese.upenn.edu/abstracts/prflow_fpt2019.html

Introduction

A field-programmable gate array (FPGA) is a silicon chip that can implement any digital circuit. Like GPUs, FPGAs are used to accelerate tasks with high amounts of parallelism. Unlike other chips, FPGAs have interconnect which allows for the reconfiguration of new digital circuits on the fly. Compiling for an FPGA versus a CPU is different in two fundamental ways: (1) source code for an FPGA represents hardware instead of instructions and (2) the former requires routing wires across the chip while the latter requires compiling program into a sequence of instructions. Both differences result in much longer compile times for FPGAs. A project compiled to a processor in seconds could take hours to compile to an FPGA. Design space exploration, debugging, and iterative design become impractically time consuming.

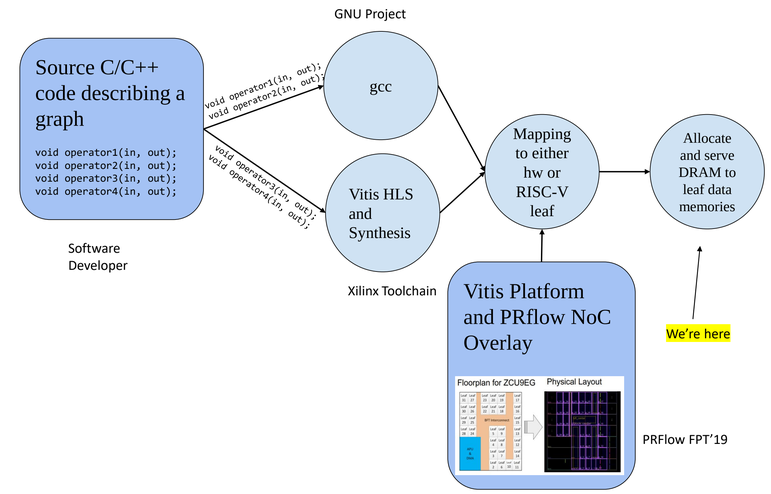




FPGAs can selectively partially reconfigure the hardware present on the chip. Hence, it is advantageous to break down a large design into smaller building blocks and only recompile blocks as they change. Even more time is saved by using softcore processors to emulate the final behavior of synthesized digital logic. This however presents new challenges. Previous works each solve one aspect of the problem: PRflow provides a packet-switch network to connect the building blocks together and rvpld extends a RISC-V softcore processor to stream into said network. Our contribution is increasing the relatively tiny amounts of memory RISC-V softcores have available on the FPGA fabric. This is accomplished by serving off-chip DRAM memory to the RISC-V cores.

Design Flow

Give a design written in C for High Level Synthesis, our design flow allows the developer to choose whether to compile certain C functions through logic synthesis as a hardware leaf or into a RISC-V binary for a precompiled softcore leaf. Because the packet-switched network is precompiled ahead of time, only the minimal amount of code goes through Vitis HLS, synthesis, placement, and routing. Hence, this efficiency in the design flow can save hours of compiling time as compared to the normal Xilinx tool flow. Once all the C operators are mapped into a leaf on the packet-switched network, DUC evaluates the data memory requirements for each softcore leaf. DUC allocates a range of DRAM addresses and provides a hardware interface for the RISC-V to serialize its network transactions over the network on a chip.



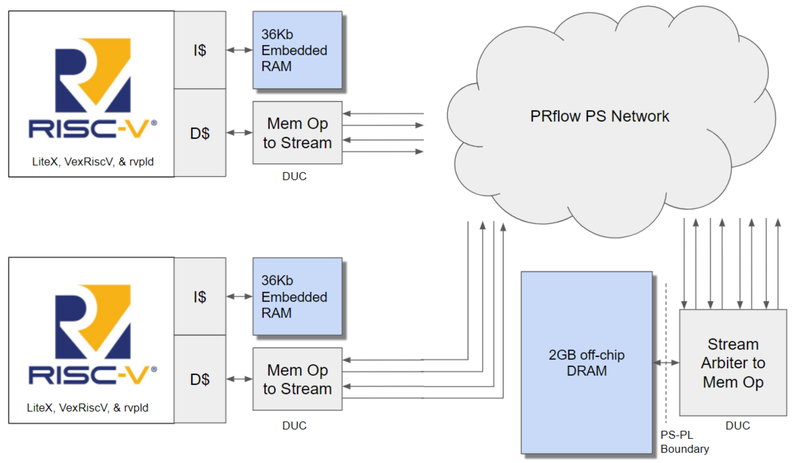
Stakeholders

The stakeholders for DUC are both current FPGA developers and software engineers requiring an acceleration platform for applications like machine learning. The primary benefit of our tool flow is the compile time savings and ability to handle more data-intensive designs. However, other benefits include enhanced design space exploration and increased modularity of the design.

Being open source, DUC allows anyone to use and modify the code without any hidden costs. However, our framework does not support safety-critical applications. There is no warranty of any form. Nonetheless, DUC is a valuable tool for those wanting to accelerate their application with minimal time commitment and dedication of resources.

Architecture

DUC is the digital logic that interfaces between the RISC-V cache and the packet-switched network. When cache lines are filled or evicted, DUC hardware converts this memory operation into a serial sequence of data that can be streamed over the network. On the DRAM side, there is logic to arbitrate the memory requests for multiple RISCV cores.



Results and Future Goals

The compile time speedup using PRflow with Vivado ranges from 2.7x to 10.72x for designs in the Rosetta Benchmark Suite (Xiao, FPT'19). DUC allows these speedups to be attained on much more complex and data-intensive designs and on smaller chips. Our experiments target the Ultra96 which has 71,000 LUTs, 141,000 flip-flops, and 7.6Mb of embedded RAM. For this platform, DUC increases the memory capacity by over 2000x (<1MB to 2GB). For an application like Sobel filtering, this means being able to handle a full-sized image rather than a 48x48 downscaled version. Future work will find ways to minimize DRAM and NoC latency penalties. Moreover, more work remains to scale DUC with a larger number of softcore RISC-V leaves running concurrently.