AFRL-RY-WP-TR-2023-0015

EXTRACTING INFORMATION FROM RICH VIDEO STREAMS: AN AGILE SOFTWARE/HARDWARE APPROACH

Mark Horowitz and Stephen Richardson
Stanford University

APRIL 2023
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with The Under Secretary of Defense memorandum dated 24 May 2010 and AFRL/DSO policy clarification email dated 13 January 2020. This report is available to the general public, including foreign nationals.

Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2023-0015 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signature//

MARC P. HOFFMAN
Program Manager
Sensors Subsystems Branch Division
Aerospace Components & Subsystems Division

//Signature//

TIMOTHY R. JOHNSON, Chief
Sensors Subsystems Branch Division
Aerospace Components & Subsystems

//Signature//

GENE M. WILKINS, Lt Col, USAF
Deputy Chief, Aerospace Components &
Subsystems Technology Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2023	Final	20 July 2018	31 December 2022

4. TITLE AND SUBTITLE

EXTRACTING INFORMATION FROM RICH VIDEO STREAMS: AN AGILE SOFTWARE/HARDWARE APPROACH

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8650-18-2-7861	N/A	62716E
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
N/A	N/A	Y1U3

6. AUTHOR(S)

Mark Horowitz and Stephen Richardson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Stanford University 424 Santa Teresa St, Stanford, CT 94305	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'SREPORT NUMBER(S)
Air Force Research Laboratory, Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command, United States Air Forces	Defense Advanced Research Projects Agency (DARPA/MTO) 675 North Randolph Street Arlington, VA 22203	AFRL/RYDR	AFRL-RY-WP-TR-2023-0015

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with The Under Secretary of Defense memorandum dated 24 May 2010 and AFRL/DSO policy clarification email dated 13 January 2020. This report is available to the general public, including foreign nationals. This material is based on research sponsored by the Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7861. 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 the Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), or the U.S. Government. Report contains color.

14. ABSTRACT

Over the course of 4.5 years, this project has worked to enhance the state of the art in image processing and, by extension, machine learning algorithms and hardware. The projects' four subgroups each focused on a specific set of tasks. 1) The applications group developed a suite of video-processing applications aimed at running with high efficiency on a coarse-grained reconfigurable architecture (CGRA)-based SoC. 2) The tools group produced compilers and generators capable of automatically moving these Halide-coded applications down to efficient hardware, such as the aforementioned CGRA. 3) The hardware group designed and taped out multiple generations of a CGRA-based SoC capable of efficiently running these applications. And 4) a test and validation group made sure that collateral was both correct and robust. This report summarizes progress made in each of these areas.

15. SUBJECT TERMS

image processing, agile hardware, coarse-grained reconfigurable arrays (CGRA), domain-specific languages (DSL), halide, compilers, generators, satisfiability modulo theories (SMT), bit-vector theory, formal proving.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	24

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Marc Hoffman	N/A

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)

Prescribed by ANSI Std. Z39.18

Table of Contents

Section **Page**

List of Tables

Table **Page**

1 INTRODUCTION

The original contract set out three sets of tasks to be accomplished: **application** tasks focused on developing a suite of video-processing applications that would run with high efficiency on a coarse-grained reconfigurable architecture (CGRA)-based SoC; **tool** tasks for generating compilers and generators capable of moving Halide-coded applications down to efficient hardware; and **hardware** tasks in the form of a CGRA-based SoC capable of efficiently running the applications. In addition, we made progress in related **test and validation** tasks to support the project. The rest of this report summarizes progress we made in each of those four areas.

2 PROGRESS AGAINST PLANNED OBJECTIVES: APPLICATIONS

2.1 Summary of Halide applications ported to CGRA

The table below summarizes some of the Halide applications that we built and/or adopted to test the CGRA system. Of these, the working regression test currently includes *gaussian, pointwise, unsharp, 2x2 camera pipeline, harris, cascade,* and *resnet* (individual ResNet layers only, not end-to-end flow (yet)).

Table 1: Halide Apps Ported to CGRA/SoC.

Application	Functionality
Gaussian	3x3 convolution using normalized values.
Cascade	Two back-to-back convolutions.
Harris	Corner detector.
Fast corner	Corner detector using FAST algorithm (sixteen comparisons).
Unsharp	Mask to sharpen the image.
Demosaic	Applies demosaic to input to create rgb image.
Demosaic + harris	Demosaic followed by harris corner detector.
Demosaic + flow	Demosaic followed by optical flow, pixel frame movement.
Stereo	Creates depth map from images from two viewpoints.
Camera pipeline	Camera pipeline with hot pixel suppression, deinterleave, demosaic, color correct, gamma correction.
Canny	Edge detection.
Bilateral filter	Perform edge-preserving blur.
Optical flow	Pixel movement between two frames.
Seedark	Algin, warp, and blend underexposed images for low light.
Resnet layer gen	Create a resnet layer with generator parameters.

2.2 Aetherling

The applications group produced Aetherling, a library for creating statically scheduled, data-parallel *pipelines* in hardware. Aetherling has been presented at several external events, including PLDI, and supports backends for both Chisel and Magma. For more information, see our list of issues and milestones in the github repo [AetherlingRepo].

Aetherling was only one of many accomplishments in the applications area. Working with industry partners such as Adobe, we modeled realistic workloads on reconfigurable accelerators. We explored beyond traditional image processing and AI apps, where applications of interest included sparse tensor and relational algebra, and various cryptographic functions. Eventually, the work concentrated on four general areas -- **sparsity, video processing, security,** and **extended reality.**

2.3 Sparsity

For sparse space-time database research, we identified benchmarks for investigating sparse space-time databases and worked on general support for sparse applications. For in-depth

exploration of real-world sparse data, we looked at basketball and moved on to video-game queries, which are more interesting in several dimensions. As an example, we did a deep dive on support for sparsity in the form of manipulating space-time databases for a challenging video-game application, namely the interactive multiplayer first-person shooter game Counter Strike: Global Offensive. This research led to advanced work in tools such as video game cheat detection algorithms to explore efficient sparse database access.

We developed sparse dataflow hardware and submitted a paper to ASPLOS related to work building a backend for the TACO Tensor Algebra Compiler, targeting Spatial/Capstan DSL for describing sparse dataflow accelerators [Liu2022].

2.4 Video Processing

Support for applications based on image processing technology included a simple audio pipeline and RNN layers. In addition, we were able to create schedules for large multi-rate applications; neural networks, where we looked at the boundaries of our support for ResNet. We created a more efficient camera pipeline and looked at histograms for bilateral filters and such. We added new video processing benchmarks to our existing stable, including synthetic exposure fusion, jitnet, and an ER pipeline.

2.5 Security/cryptography

We started a concerted effort to explore more efficient cryptographic algorithms for security, looking closely at efficient GCD / XGCD algorithms and hardware. This culminated in the construction of an actual XGCD hardware acceleration unit as a separately testable IP block on the Onyx chip.

2.6 Extended Reality

Finally, in the AR/VR space, we used HLS tools to build a complete SLAM pipeline and started analyzing the performance of specific kernels.

3 PROGRESS AGAINST PLANNED OBJECTIVES: TOOLS / HALIDE AND CLOCKWORK

A major objective for this project included the ability to compile image-processing benchmarks from their Halide source code all the way down to the bare hardware, mapping efficiently to multiple targets including our CGRA accelerator.

Along with existing various CPU/FPGA targets, the compiler group's Halide-to-Hardware system eventually generated code not only for our CGRA accelerator, but also for the machine hosting the accelerator. To accomplish this, we looked at developing and working with a new abstraction for hardware-software interfacing between, e.g., a hardware accelerator and a host processor. This Reconfigurable Device Access Interface, or RDAI, lets us use all three levels of available memory when mapping applications---host buffer, global buffer, and memory tiles.

We implemented new Halide IR transformations that yield efficient hardware for processing Gaussian pyramids. We also created a set of commands to run memory mapping from Halide, which lets us easily check the status of all applications from a single repository.

Our memory compiler also properly uses Lake collateral (see Magma-tools section below), so that it can map to different memory tiles as our design decisions change. This lets us look at more applications and debug their issues. One such issue was solved when we improved the banking algorithm for CGRA mapping, by relaxing previously enforced banking assumptions. We can also map applications with memory hierarchies, like register files (ponds) and large SoC SRAM buffer (global buffer).

Also, the compiler was expanded to handle cases where multiple accelerators exist in a single application, e.g., to support multiple layers in a deep neural network. In addition, the compiler group has made significant efforts to support testing the physical chip and board with bug workarounds and such.

We made progress on pipelining applications for higher-frequency performance, with significant progress in implementing and analyzing ResNet on the CGRA, and we investigated ways that the memory hierarchy and the compiler could be modified to suit new application domains such as sparse-matrix and crypto.

We tuned the tool chain for ResNet and looked at a potential method for sharing kernels to minimize the number of required processing elements (PE tiles). Plus, we worked on support for new applications that use e.g. histograms and bilateral grids. Meanwhile, pipelining efforts yielded 3x clock frequency improvements, with the promise of even more improvement to come.

3.1 Clockwork

The Halide-to-Hardware team fully integrated new support for polyhedral analysis in the form of a new framework we called Clockwork. Clockwork gives us a new way to optimize memories before mapping them to hardware. Adding Clockwork to the application flow lets us seamlessly run from Halide application to memory mapping, hardware testing, and evaluation all using a single script.

We completed a codegen code generator connecting our front-end language, Halide, to this new framework. The Halide codegen generates separate files for the application's computation and memories. A couple of optimization passes were added to Halide to simplify the analysis needed by Clockwork. These changes let Clockwork function efficiently in our existing stencil applications (convolution, Harris corner detector, camera pipeline), as well as DNN applications (ResNet layers).

4 PROGRESS AGAINST PLANNED OBJECTIVES: TOOLS / MAGMA

At the beginning of the contract period, we had an existing simple Perl-based in-house Verilog generation platform called *Genesis2*. As the project progressed, we transitioned wholesale to a much more sophisticated Python-based generator Magma. These "generators" allow us to write high-level extremely parameterizable code that then gets transformed into the SystemVerilog that describes our chip.

As the chip evolved, we added support for multiple new ALU operations, including transcendental functions and such, to map target applications more efficiently. To that end, we developed a domain-specific language and compiler, Peak, to describe and build PEs. At the same time, we enhanced the Halide compiler to use these functions when building target applications from source code.

The Peak PE-generator team eventually completed work for, and submitted a paper on, automatic rewrite rule generation for mapping IR code to actual hardware, including both RISC-V and our own CGRA [Noetzli2022]. Also, we used Peak for PE design space exploration.

Along with the Peak PE language and generator, we eventually added an interconnect generator Canal and a memory generator Lake, all operating within the aegis of the larger Magma generator platform.

Lake focuses on how exactly we create our CGRA memories, and then how mapping will occur. The general architecture includes an SRAM, address generator, ingress reorder buffer, and egress reorder buffer. This design space uses a DSL to describe the space that the memory can be used, and then rewrite rules help create the mappings to the generated CGRA memory tile. Eventually, Lake was refined to support smaller memory structures such as register files, which, in the Lake context, we call ponds.

We improved the tool chain by providing CoreIR MLIR compatibility, repo safety, and linking improvements, as well as symbol tables for translating CoreIR to Verilog and Magma. Similarly, we added SSA symbol table logic to Magma for better debugging. Plus, we added support for ready-valid channels in our FPGA compilation backend. Eventually, improvements to the new MLIR backend for Magma yielded 2-10x runtime improvement on our benchmark suite. And we now have support for arrays in Magma.

Finally, we started an effort that we call Smart Components, as a way of using session types in Magma to describe and then check the correctness of protocol-based hardware interactions. A prominent example includes memory-controller features such as power-on sequencing and Built-In-Self-Test. Smart Components describes such interactions as high-level abstract types, then lowers the type description to actual hardware.

The Smart Components methodology can help avoid logical bugs late in the design process, providing designing a system for correct composition of SoC components combining static verification and unit testing techniques. We were excited to be able to engage Facebook and observe how this research could be practically useful in an industrial chip design flow.

5 PROGRESS AGAINST PLANNED OBJECTIVES: HARDWARE

Early on, we aimed for the goal of having a one-button flow that could build a domain-specific coarse-grain-reconfigurable-array (CGRA) and, at the same time, compile and run real applications on that CGRA. We originally targeted the domain to include a narrower group of video and image processing applications, later expanding that to include more general machine-learning algorithms. In the end, we taped out four chips, **Jade, Garnet[1], Amber,** and **Onyx.**

Table 2: We Produced Four Chips with Successively more Advanced Complexity and Capabilities

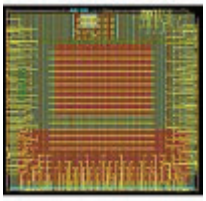
Jade's PE tile supported only basic integer operations and simple line buffer access patterns, and so could efficiently run only the most basic of image processing applications. The final chip, Onyx, had a hardware support for integer, floating point and transcendental functions, plus single-cycle tertiary adds and multiply-accumulate operations, plus a 24 KB register file.

	Jade	**Garnet**	**Amber**	**Onyx**
Process, Size	TSMC16 5mm x 5mm	TSMC16 5mm x 5mm	TSMC16 5mm x 5mm	GF12 5mm x 5mm
Tapeout Date	Summer 2018	June 2020 (virtual)	December 2020	December 2022
RTL Generator	Genesis2 (Perl)	Magma (Python)	Magma (Python)	Magma (Python)
Physical Design	Custom/manual	mflowgen	mflowgen	mflowgen
SoC?	No	M3-based SoC	M3-based SoC	M3-based SoC
CGRA	16x16	32x16	32x16	32x16
PE Tile	Integer only	FP + complex arith	Local register file	TADD, MAC
Mem Tile SRAM	2K	4K	4K	4K
Global Buffer	No	Yes	Yes	Yes
Mem Access	Line buffer only	General affine access pattern	General affine access pattern	Sparse memory access patterns
Applications	Image Processing	IP + ML (dense)	IP + ML (dense)	IP + ML (sparse)

[1] We produced a tape for Garnet, but it was not sent to the manufacturer and no actual silicon got built.

5.1 Jade

The first chip to tape out, early in the project, consisted of just a standalone CGRA constructed as a largely custom effort, thus motivating the later push toward one-button configurable tapeout scripts. We called this first chip *Jade,* and it taped out in December of 2018.



Continuing from this first, more primitive CGRA, we added support for multiple new operations, e.g. transcendental functions, to map target applications more efficiently. To that end, we developed a domain-specific language and compiler, Peak*,* to describe and build PEs. At the same time, we enhanced the Halide compiler to use these functions when building target applications from source code. The new CGRA was targeted to be part of a complete SoC called *Garnet.*

5.2 Garnet

We used the custom physical design scripts from Jade as a basic starting point to create a functioning physical design flow for Garnet. In addition to incorporating components that did not exist in Jade (global buffer and ARM processor) we made several changes to the existing flow aimed at improving both the PPA of the resulting chip and robustness of the flow itself to small design changes. To that end, we updated the flow to make a fully abutted array of CGRA tiles.

For the memory tile, we went beyond the initial goal of adding support for double buffers. We instead implemented support for a unified buffer abstraction, enabling us to efficiently support many new memory-access patterns which were not possible with our previous memory tile.

We added a global buffer to provide a second level of memory hierarchy above the memory tiles and enable fast parallel reconfiguration of the CGRA tiles.

In pursuit of better energy efficiency, we added power domains such that, for a given application, CGRA tiles could be turned off when not in use, regardless of their position relative to active tiles in the array. Our SoC did not have a DRAM controller, so we created a TLX interface to support high-bandwidth communication with a DRAM controller on an external FPGA.

Garnet was an ambitious ground-up redesign of the physical design flow. That, plus the added burden of putting the CGRA into the context of a larger SoC with on-board CPU controller and global memory, resulted in a rare schedule miss in terms of producing an actual chip.

Although Garnet missed the initial manufacturing deadline, we achieved DRC- and LVS-clean virtual tapeout in the spring of 2019. The design was shown to be functionally correct in simulation, with much more extensive testing than had been done with Jade. Applications ran end-to-end on the (simulated) Garnet SoC, exercising all components including the ARM Cortex M-3, DMA and ACI interfaces, and our global buffer, a large on-chip memory store. This global, or unified, buffer, itself marked a significant step forward as compared to the prior chip, allowing swift execution of apps with significantly lower I/O overhead.

Having completed virtual tapeout of Garnet, we moved on to the next iteration of the chip design, which we called *Amber.*

5.3 Amber

An early goal of this project was to have a one-button physical design flow that could take the place of custom-built scripts. To that end, we adopted and embarked upon a serious effort to support and enhance the *mflowgen* tool developed by Chris Torng, who later joined the project as a post-doc from Cornell University. Mflowgen is a lightweight modular flow specification and build-system generator for ASIC and FPGA design-space exploration. As part of this effort, the chip became more hierarchical, with all the major blocks laid out and assembled separately before integrating to build the final die.

The switch to mflowgen required a big change in methodology.

Initially, we had looked at using UC Berkeley's HAMMER system. We were pleased not only with how HAMMER enabled technology-independent physical flows, but also the way it tried to provide tool-independence. HAMMER provides its own YAML interface, with corresponding plugins that turn the YAML into physical flows for either Synopsys or Cadence. But we felt that “hijacking” the TCL interface to the back-end tools would make things difficult when we inevitably needed to do something that was not entirely standard, or had unexpected hiccups.

Mflowgen (modular flow generator), on the other hand, does not try to provide independence over tools, i.e. PD scripts remain in TCL and not some intermediate representation like YAML. It does provide a standard interface to technology files, letting us write technology-independent modular flows, and it enables easy design space exploration through parameter sweeps. It lets us define the full physical flow as a graph of nodes, each with code to execute plus a set of inputs and a set of outputs. Overall, we felt that mflowgen was a more lightweight and easy-to-use

solution than HAMMER, and it still addressed our main physical design issue: generating reusable, technology-independent physical flows. In addition, Chris Torng, the primary developer and maintainer of mflowgen, joined the group, making it far easier for us to use and develop the tool, especially in support of May tapeout.

The ensuing fully hierarchical physical design flow relied heavily on the mflowgen build system to encapsulate synthesis, place-and-route, LVS and DRC checks, culminating in a GDS tape.

In May 2020, we finished a tapeout-ready DRC-and LVS-clean GDS for the Amber SoC in TSMC 16nm, using the full suite of AHA hardware design tools. The SoC consisted of a reconfigurable array of PE and memory tiles produced using Peak and Lake, respectively; a large 4 MB memory called the global buffer; and an ARM Cortex M3 processor subsystem. The tile array and global buffer taped out at 750 MHz, while the processor subsystem ran at 500 MHz.

Unfortunately, a miscommunication with the manufacturer resulted in us not having a spot on their shuttle, and so the tape did not get sent.

The manufacturer set a new target date for October 2020. This gave us a chance to further tweak the design, plus we volunteered to be guinea pigs for TSMC's new, experimental cloud-based Virtual Development Environment (VDE) tool chain. The entire design was to be ported to this new flow, which provided a golden opportunity to stress-test our decision to use mflowgen.

Meanwhile, progress on our memory-generator tools Lake and Pond resulted in more efficient memory tiles and new register-file functionality in the PE tiles. Plus, we added new features such as pipelined configuration buses, and a better clock distribution strategy.

But then the target tapeout date shifted again, with a new shuttle deadline of December 1, 2020. Why? Because, as the original October deadline neared, TSMC began "to complete internal procedures" without first checking our reservation status. When they came to the end of that process, they found that October was full. They then offered us the December slot. We said yes.

So in December, we finally taped out the Amber chip. Along the way, we had managed to overcome several challenges, notably including the complete last-minute port of our entire design to VDE. This port was helped immensely by reliance on generators such as Magma and mflowgen, and an overall agile flow including continuous integration. Our success provided solid proof of concept for using such an approach.

Having successfully taped-out Amber, the chip design group refocused on tasks related to getting ready to have it powered up and tested. We were told to expect packaged die sometime in early May of 2021 (spoiler: actually got them in mid-April) So we busied ourselves building boards and test rigs in preparation.

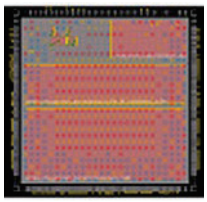
When the packaged Amber SoC chips finally arrived, the team set about working to test and approve them. After implementing workarounds for a couple of bugs including memory address problems, we were able successfully to run several applications on the board.

By Q4 2021, we had Amber SoC chips on working boards in the lab, with pipelined apps using all PEs at about 600MHz. But we wanted to improve the boards and quantify them with more and more applications and tests. To that end, we designed a new / better board and had it manufactured ASAP. Among other things, the new board had standalone power and clock. Using the board to demonstrate our technology, we won Best Demo award at the IEEE Symposium on VLSI Technology and Circuits in Hawaii.

To recap, Amber ended as a system-on-chip (SoC) with a coarse-grained reconfigurable array (CGRA) for acceleration of dense linear algebra applications such as machine learning (ML), image processing, and computer vision. So far, in the lab, we have seen a peak energy efficiency of 538.0 INT16 GOPS/W and 483.3 BFloat16 GFLOPS/W. We maximize CGRA utilization and minimize reconfigurability overhead through (1) dynamic partial reconfiguration of the CGRA that enables higher resource utilization by allowing multiple applications to run at once, (2) efficient streaming memory controllers supporting affine access patterns, and (3) low-overhead transcendental and complex arithmetic operations. Compared to a CPU, a GPU, and an FPGA, Amber has achieved up to 3902x, 152x, and 88x better energy-delay product (EDP). More details can be found in our paper at the 2022 IEEE Symposium on VLSI Technology and Circuits [Carsello2022a].

5.4 Onyx

In Q1 2022, we had a large ResNet application working on-chip using the new boards. With little pause, work began on a new follow-on chip, *Onyx*, to be manufactured in Global Foundries' GF12 technology, a further test of the portability of our mflowgen-based physical design.



Onyx was targeted to have several improvements over Amber, including significant improvements in one of its more power-hungry components, the global buffer; an enhanced instruction set for faster and more efficient performance on target applications; a higher operating frequency by providing enhanced hardware support for pipelining applications; and significant hardware support for linear algebra operations on sparse data sets.

The Onyx chip taped out successfully early in Q1 2023, with the final completed physical design being submitted on October 24. And work has started on the next design *Opal.*

Although this contract has ended, the project will continue independently, as we move along to the next design, *Opal.* The new design will have more/better support for sparse applications, to efficiently support AI/ML and a wide variety of other/similar domains. In addition, we are investigating what other features may or may not be profitably included on the chip.

6 PROGRESS AGAINST PLANNED OBJECTIVES: TESTING AND VALIDATION

6.1 Testing and Validation: Fault

Leonard Truong's Fault, a Python package for testing hardware, is part of the Magma ecosystem. It includes support for interactive debugging along with our in-house Kratos code generator.
*Fault: https://github.com/leonardt/fault
*Kratos: https://github.com/kuree/kratos

Fault includes a Python interface to specify SVA (SystemVerilog assertions)-style properties referring to Magma circuit values. This raises the level of abstraction in verification of Magma circuits by allowing design verification engineers to write properties at the Python/Magma generator level, rather than on the generated verilog. Raising the level of abstraction lets engineers employ the same techniques used in the design process, including metaprogramming with introspection, and referring to generator parameters in their properties. The current implementation generates SVA syntax that is compatible with commercial simulators.

Fault includes support for SystemVerilog bind patterns that can inject verification code into the RTL without having to change the RTL. This provides confidence that the synthesis and verification design are the same (rather than having two different versions for each task). We also developed examples for writing synthesizable test benches, a practice that has proven useful in the Chisel community. Many lower-level unit tests can be written using synthesizable constructs (Magma), so it doesn't require the complexity added by many verification environments. This is particularly useful for lowering the overhead of writing small unit tests quickly, which in turn promotes the construction of more tests.

In addition, the Fault language was extended with preliminary support for the *pono* model checker and the assertion language was improved to support *cover* properties and *immediate* assertions.

6.2 Testing and Validation: Bit-Vector Theory

Significant progress in Clark Barrett's group concerned support for bit-vector theory, also see https://github.com/barrett-lab/AHA-goals/milestone/9 , https://github.com/barrett-lab/AHA-goals/milestones

We completed a basic design for a new bit-vector rewriting infrastructure for CVC4, an open-source automatic theorem prover for satisfiability modulo theories (SMT) problems (https://github.com/barrett-lab/AHA-goals/milestone/6?closed=1).

We developed and released CVC5, an exciting major version bump of the well-known CVC4 solver and started the work to incorporate the SyGus synthesizer for formal configuration finding.

We developed a new model-based API fuzzing tool *Murxla* for rigorous testing of SMT solver APIs. We also developed and integrated a new local search library for solving quantifier free bit vector formulas, which implements our propagation-based local search approach (presented at FMCAD 2020) in our SMT solver *Bitwuzla.*

And finally, we implemented a framework for synthesizing a configuration mapping function and worked on benchmarking its capabilities on complex applications.

6.3 Testing and Validation: Other Efforts

We additionally made progress in three key areas. We support *interactive testing* at the front end, using the Magma (Python) simulator and an interpreter for our CoreIR intermediate code. In addition, a *synchronous tester* was released to better handle synchronous test drivers that might use e.g. non-blocking assignments to drive input vectors. Finally, there is now support for *inline Verilog* within Magma, as a quick and dirty way for users to write verification collateral such as assertions, covergroups, and logging/display statements.

Major verification efforts for the CGRA focused on virtualization support for running multiple applications on the same fabric. A novel approach for quantified formulas was added to our solver infrastructure, improving the capabilities of formal methods-based verification efforts. The team developed a library *pysv* that integrates python functional models into System Verilog simulation to facilitate simpler integration with our Python code base.

7 REFERENCES

[AetherlingRepo] See https://github.com/David-Durst/aetherling .

[Barbosa2022] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds. cvc5: A Versatile and Industrial-Strength SMT Solver. Ying Sheng, Cesare Tinelli, Yoni Zohar. TACAS 2022.

[Barbosa2022a] Haniel Barbosa, Andrew Reynolds, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Arjun Viswanathan, Scott Viteri, Yoni Zohar, Cesare Tinelli, Clark Barrett. Flexible Proof Production in an Industrial-Strength SMT Solver. In Automated Reasoning (IJCAR), pages 15-35, 2022.

[Barbosa2022b] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, Yoni Zohar. cvc5: A Versatile and Industrial-Strength SMT Solver. In Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pages 415-442, 2022.
SCP Best Tool Paper Award.

[Carsello2022] Alex Carsello, James Thomas, Ankita Nayak, Po-Han Chen, Mark Horowitz, Priyanka Raina, Christopher Torng. Enabling Reusable Physical Design Flows with Modular Flow Generators. Design Automation Conference (DAC), July 2022.

[Carsello2022a] Alex Carsello, Kathleen Feng, Taeyoung Kong, Kalhan Koul, Qiaoyi Liu, Jackson Melchert, Gedeon Nyengele, Maxwell Strange, Keyi Zhang, Ankita Nayak, Jeff Setter, James Thomas, Kavya Sreedhar, Po-Han Chen, Nikhil Bhagdikar, Zachary Myers, Brandon D'Agostino, Pranil Joshi, Stephen Richardson, Rick Bahr, Christopher Torng, Mark Horowitz, Priyanka Raina. Amber: A 367 GOPS, 538 GOPS/W 16nm SoC with a Coarse-Grained Reconfigurable Array for Flexible Acceleration of Dense Linear Algebra. IEEE Symposium on VLSI Technology & Circuits (VLSI), June 2022. Best Demo Paper Award.
Paper: https://ieeexplore.ieee.org/document/9830509
VLSI Demo Session: https://mobile.twitter.com/VLSI_2022/status/1536825690196217856

[Carsello2022b] Alex Carsello, James Thomas, Ankita Nayak, Po-Han Chen, Mark Horowitz, Priyanka Raina, and Christopher Torng. mflowgen: A Modular Flow Generator and Ecosystem for Community-Driven Physical Design. ACM/IEEE Design Automation Conference (DAC), July 2022. Paper: https://ctorng.com/pdfs/carsello-mflowgen-dac2022.pdf
DOI: https://doi.org/10.1145/3489517.3530633

[Chen2023] Po-Han Chen, Charles Tsao, Priyanka Raina. An Open-Source 4x8 Coarse-Grained Reconfigurable Array Using SkyWater 130nm Technology and Agile Hardware Design Flow. To appear in IEEE International Symposium on Circuits and Systems (ISCAS), May 2023.

[Daly2022] Ross Daly, Caleb Donovick, Jack Melchert, Raj Setaluri, Nestan Tsiskaridze, Priyanka Raina, Clark Barrett, Pat Hanrahan. Synthesizing Instruction Selection Rewrite Rules from RTL using SMT. Conference on Formal Methods in Computer-Aided Design (FMCAD), 12 October 2022 (p. 139).

[Durst2020] David Durst, Matthew Feldman, Dillon Huff, David Akeley, Ross Daly, Gilbert Louis Bernstein, Marco Patrignani, Kayvon Fatahalian, Pat Hanrahan. Type-Directed Scheduling of Streaming Accelerators. 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020). https://pldi20.sigplan.org/

[Durst2020a] David Durst, Matthew Feldman, Dillon Huff, David Akeley, Ross Daly, Gilbert Louis Bernstein, Marco Patrignani, Kayvon Fatahalian, Pat Hanrahan. Aetherling: Type-Directed Scheduling of Streaming Accelerators. Presentation to Computer Architecture and Programming Abstractions research groups at Cornell, Wednesday July 22, 2020.
[Durst2020b] David Durst, Matthew Feldman, Dillon Huff, David Akeley, Ross Daly, Gilbert Louis Bernstein, Marco Patrignani, Kayvon Fatahalian, Pat Hanrahan. Aetherling: Type-Directed Scheduling of Streaming Accelerators. Presentation to Circuit IR Compilers and Tools (CIRCT) Weekly Meeting, Dec. 9th, 2020.
[Feng2022] Kathleen Feng, Alex Carsello, Taeyoung Kong, Kalhan Koul, Qiaoyi Liu, Jackson Melchert, Gedeon Nyengele, Maxwell Strange, Keyi Zhang, Ankita Nayak, Jeff Setter, James Thomas, Kavya Sreedhar, Po-Han Chen, Nikhil Bhagdikar, Zachary Myers, Brandon D'Agostino, Pranil Joshi, Stephen Richardson, Rick Bahr, Christopher Torng, Mark Horowitz, Priyanka Raina. Amber: Coarse-Grained Reconfigurable Array-Based SoC for Dense Linear Algebra Acceleration. To appear in Hot Chips: A Symposium on High Performance Chips, August 2022.
[Hsu2023] Olivia Hsu, Maxwell Strange, Ritvik Sharma, Jaeyeon Won, Kunle Olukotun, Joel S Emer, Mark Horowitz, and Fredrik Kjolstad. The Sparse Abstract Machine. Architectural Support for Programming Languages and Operating Systems (ASPLOS) conference, March 2023. Also see ArXiv: https://arxiv.org/abs/2208.14610 ,
DOI: https://doi.org/10.48550/arxiv.2208.14610
[Huff2021] Dillon Huff, Steve Dai, and Pat Hanrahan. Clockwork: Resource-Efficient Static Scheduling for Multi-Rate Image Processing Applications on FPGAs. In The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '21). Association for Computing Machinery, New York, NY, USA, 145–146.
DOI: https://doi.org/10.1145/3431920.3439457
[Kong2022] Taeyoung Kong, Kalhan Koul, Priyanka Raina, Mark Horowitz, Christopher Torng. Hardware Abstractions and Hardware Mechanisms to Support Multi-Task Execution on Coarse-Grained Reconfigurable Arrays. To appear in Workshop on Democratizing Domain-Specific Accelerators (WDDSA) at MICRO, October 2022.
[Koul2022] Kalhan Koul, Jackson Melchert, Kavya Sreedhar, Leonard Truong, Gedeon Nyengele, Keyi Zhang, Qiaoyi Liu, Jeff Setter, Po-Han Chen, Yuchen Mei, Maxwell Strange, Ross Daly, Caleb Donovick, Alex Carsello, Taeyoung Kong, Kathleen Feng, Dillon Huff, Ankita Nayak, Rajsekhar Setaluri, James Thomas, Nikhil Bhagdikar, David Durst, Zachary Myers, Nestan Tsiskaridze, Stephen Richardson, Rick Bahr, Kayvon Fatahalian, Pat Hanrahan, Clark Barrett, Mark Horowitz, Christopher Torng, Fredrik Kjolstad, and Priyanka Raina. AHA: An Agile Approach to the Design of Coarse-Grained Reconfigurable Accelerators and Compilers. ACM Transactions on Embedded Computing Systems (April 2022).
https://doi.org/10.1145/3534933
[Kremer2021] G. Kremer, A. Niemetz, M. Preiner. ddSMT 2.0: Better Delta Debugging for the SMT-LIBv2 Language and Friends. In Proc. 33rd International Conference on Computer Aided Verification (CAV 2021).
[Liu2019] Qiaoyi Liu, Jeff Setter, Kathleen Feng, Xuan Yang, Teguh Hofstee, Mark Horowitz and Priyanka Raina. A Unified Push Memory for Accelerator Generation (Poster). 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO 2019), Columbus, Ohio, USA. https://www.microarch.org/micro52/program/src.html

[Liu2021] Qiaoyi Liu, Dillon Huff, Jeff Setter, Maxwell Strange, Kathleen Feng, Kavya Sreedhar, Ziheng Wang, Keyi Zhang, Mark Horowitz, Priyanka Raina, and Fredrik Kjolstad. Compiling Halide Programs to Push-Memory Accelerators. ArXiv, 2021. https://arxiv.org/abs/2105.12858
[Liu2022] Qiaoyi Liu, Jeff Setter, Dillon Huff, Maxwell Strange, Kathleen Feng, Mark Horowitz, Priyanka Raina, Fredrik Kjolstad. Unified Buffer: Compiling Image Processing and Machine Learning Applications to Push-Memory Accelerators. ACM Transactions on Architecture and Code Optimization (TACO), November 2022. https://doi.org/10.1145/3572908 Presentation at HiPEAC: https://www.hipeac.net/2023/toulouse/#/program/sessions/8059/
[Mann2020] Makai Mann and Clark Barrett. Partial Order Reduction for Deep Bug Finding in Synchronous Hardware. TACAS 2020.
[Mann2021] Makai Mann, Ahmed Irfan, Alberto Griggio, Oded Padon, Clark Barrett. Counterexample-Guided Prophecy for Model Checking Modulo the Theory of Arrays. TACAS 2021.
[Mann2021a] Makai Mann, Ahmed Irfan, Florian Lonsing, Yahan Yang, Hongce Zhang, Kristopher Brown, Aarti Gupta, and Clark Barrett. Pono: A Flexible and Extensible SMT-based Model Checker In Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part II 33, pp. 461-474. Springer International Publishing, 2021.
[Mann2021b] Makai Mann, Amalee Wilson, Yoni Zohar, Lindsey Stuntz, Ahmed Irfan, Kristopher Brown, Caleb Donovick, Allison Guman, Cesare Tinelli, and Clark Barrett. SMT-switch: a solver-agnostic C++ API for SMT solving. In Theory and Applications of Satisfiability Testing–SAT 2021: 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings 24, pp. 377-386. Springer International Publishing, 2021.
[Melchert2022] Jackson Melchert, Keyi Zhang, Yuchen Mei, Mark Horowitz, Christopher Torng, Priyanka Raina. Canal: A Flexible Interconnect Generator for Coarse-Grained Reconfigurable Arrays. Workshop on Democratizing Domain-Specific Accelerators (WDDSA) at MICRO, October 2022. https://ctorng.com/pdfs/kong-dpr-wddsa2022.pdf
[Melchert2023] Jackson Melchert, Kathleen Feng, Caleb Donovick, Ross Daly, Ritvik Sharma, Clark Barrett, Mark Horowitz, Pat Hanrahan, Priyanka Raina. APEX: A Framework for Automated Processing Element Design Space Exploration using Frequent Subgraph Analysis. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), March 2023.
[Misc2019] Attendees: Misc, AHA-affiliated faculty and students, plus corporate representatives from corporate affiliates including Intel, Facebook, Google, Nvidia, and Amazon. AHA Asilomar Retreat. Asilomar Conference Grounds, 800 Asilomar Avenue, Pacific Grove, CA 93950, May 31 – June 1 2019. Multiple presentations, see individual links embedded in agenda at https://aha.stanford.edu/resources/aha-retreat-may-2019
[Misc2019a] Attendees: Misc, AHA-affiliated faculty and students. Intel Annual Review. Intel Corporation Conference Arena, Santa Clara, CA, Sep. 9, 2019 Multiple presentations, see individual links embedded in agenda at https://aha.stanford.edu/resources/intel-annual-review-sep-2019
[Nayak2020] Ankita Nayak, Keyi Zhang, Raj Setaluri, Alex Carsello, Makai Mann, Stephen Richardson, Rick Bahr, Pat Hanrahan, Mark Horowitz and Priyanka Raina. A Framework for Adding Low-Overhead, Fine-Grained Power Domains to CGRAs. Design, Automation and Test in Europe Conference (DATE 2020). https://www.date-conference.com/

[Nayak2022] Ankita Nayak, Keyi Zhang, Raj Setaluri, Alex Carsello, Makai Mann, Christopher Torng, Stephen Richardson, Rick Bahr, Pat Hanrahan, Mark Horowitz, Priyanka Raina. Improving Energy Efficiency of CGRAs with Low-Overhead Fine-Grained Power Domains. Transactions on Reconfigurable Technology and Systems (TRETS), August 2022. https://dl.acm.org/doi/10.1145/3558394
[Niemetz2020] Aina Niemetz and Mathias Preiner. Ternary Propagation-Based Local Search for More Bit-Precise Reasoning. 20th International Conference on Formal Methods in Computer Aided Design (FMCAD 2020). https://cs.stanford.edu/~niemetz/publications.html#fmcad20 https://fmcad.forsyte.at/FMCAD20/
[Niemetz2021] Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark Barrett, Cesare Tinelli. Syntax-Guided Quantifier Instantiation. In Proc. 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2021).
[Niemetz2021a] Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark Barret, Cesare Tinelli. On Solving Quantified Bit-Vector Constraints using Invertibility Conditions. Journal on Formal Methods in System Design, 2021.
https://cs.stanford.edu/~niemetz/publications/2021/NiemetzPreinerReynoldsBarrettTinelli-FMSD21.pdf http://dx.doi.org/10.1007/s10703-020-00359-9
[Niemetz2021b] Aina Niemetz, Mathias Preiner, Andrew Reynolds, Yoni Zohar, Clark Barrett, Cesare Tinelli. Towards Satisfiability Modulo Parametric Bit-Vectors. Journal of Automated Reasoning JAR 65 (7): 1001-1035, 2021.
https://cs.stanford.edu/~niemetz/publications/2021/NiemetzPreinerReynoldsZoharBarrettTinelli-JAR21.pdf
[Niemetz2022] Aina Niemetz, Mathias Preiner, Clark Barrett. Murxla: A Modular and Highly Extensible API Fuzzer for SMT Solvers. In Computer Aided Verification (CAV), pages 92-106, 2022.
[Noetzli2022] Andres Noetzli, Haniel Barbosa, Aina Niemetz, Mathias Preiner, Andrew Reynolds, Cesare Tinelli and Clark Barrett. Reconstructing Fine-Grained Proofs of Complex Rewrites Using a Domain-Specific Language. Formal Methods in Computer-Aided Design (FMCAD), 2022.
[Scott2021] Joseph Scott, Aina Niemetz, Mathias Preiner, Saeed Nejati and Vijay Ganesh. MachSMT: A Machine Learning-based Algorithm Selector for SMT Solvers. In Proc. 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2021).
[Truong2019] Lenny Truong and Pat Hanrahan. A Golden Age of Hardware Description Languages: Applying Programming Language Techniques to Improve Design Productivity. 3rd Summit on Advances in Programming Languages (SNAPL 2019).
http://drops.dagstuhl.de/opus/frontdoor.php?source_opus=10550
[Truong2020] Lenny Truong, Steven Herbst, Rajsekhar Setaluri, Makai Mann, Ross Daly, Keyi Zhang, Caleb Donovick, Daniel Stanley, Mark Horowitz, Clark Barrett, Pat Hanrahan. Fault: A Python Embedded Domain-Specific Language For Metaprogramming Portable Hardware Verification Components. 32nd International Conference on Computer-Aided Verification (CAV 2020). http://i-cav.org/2020/
[Tsiskaridze2021] N. Tsiskaridze, M. Strange, M. Mann, K. Sreedhar, Q. Liu, M. Horowitz, and C. Barrett. Automating System Configuration. 2021 Formal Methods in Computer Aided Design (FMCAD). doi: 10.34727/2021/isbn.978-3-85448-046-4_19. Paper presented at FMCAD 2021.

[Yang2020] Xuan Yang, Mingyu Gao, Qiaoyi Liu, Jeff Setter, Jing Pu, Ankita Nayak, Steven Bell, Kaidi Cao, Heonjae Ha, Priyanka Raina, Christos Kozyrakis, Mark Horowitz. Interstellar: Using Halide's Scheduling Language toAnalyze DNN Accelerators. Proc. 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20), Lausanne, Switzerland.
[Zohar2022] Yoni Zohar, Ahmed Irfan, Makai Mann, Aina Niemetz, Andres Nötzli, Mathias Preiner, Andrew Reynolds, Clark Barrett, Cesare Tinelli. Bit-Precise Reasoning via Int-Blasting. In Verification, Model Checking, and Astract Interpretation (VMCAI), pages 496-518, 2022.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

ACRONYM	DESCRIPTION
AR/VR	Augmented Reality / Virtual Reality
ARM	Advanced RISC Machine
CGRA	Coarse-Grained Reconfigurable Architecture
CPU	Central Processing Unit
FPGA	Field Programmable Gate Array
DRAM	Dynamic Random Access Memory
DSL	Domain Specific Languages
EDP	Energy-Delay Product
GCD	Greatest Common Divisor (Euclidean Algorithm)
HLS	High Level Synthesis
I/O	Input/Output
IP	Intellectual Property
PLDI	Programming Language Designed and Implementation (Conference)
RISC	Reduced Instruction Set Computer
SLAM	Simultaneous Localization And Mapping
SMT	Satisfiability Modulo Theories
SoC	System-on-Chip
SVA	System Verilog Assertions
TACO	Tensor Algebra Compiler
VDE	Virtual Development Environment
XGCD	Extended Greatest Common Divisor (Extended Euclidean Algorithm)