Experiences with Two FabScalar-based Chips

Elliott Forbes, Rangeen Basu Roy Chowdhury, Brandon Dwiel, Anil Kannepalli, Vinesh Srinivasan, Zhenqian Zhang, Randy Widialaksono, Thomas Belanger, Steve Lipa, Eric Rotenberg, W. Rhett Davis, Paul D. Franzon
North Carolina State University, Raleigh, NC, 27695, USA {jeforbe2@ncsu.edu}

1. Introduction

This extended abstract previews two on-going projects at NCSU to design, fabricate, test, and evaluate novel processors. Both processors are derived from FabScalar-generated superscalar cores [1]. The technical objective of both processors is to exploit adaptivity: adapting the microarchitecture to varying instruction-level behavior in programs. They represent two different forms of adaptive processors. The H3 processor features two different superscalar core types with fast thread migration between the two. The core is virtually customized to the currently executing program phase by way of migrating the program's execution to the most suitable core. Both 2D and 3D IC implementations are being pursued. The AnyCore processor is a single highly reconfigurable superscalar core. Its superscalar widths and structure sizes are adjustable.

Aside from their technical objectives, these projects are motivated in part by two other personal goals. First, having spearheaded the FabScalar project, we wanted to fulfill its original purpose – streamline the development of single-ISA heterogeneous multi-core processors. A chip-building funding opportunity arose which perfectly matched the capabilities afforded by NCSU's FabScalar toolset, heterogeneous multi-core research program, and 2D/3D IC physical design and fabrication expertise. Second, speaking for the computer architects in the project, another personal goal was to experience the distraction, risk, and reward of building something. Having wandered far outside of our comfort zone of simulation and the grind of the premier conference circuit, we tremendously expanded our knowledge and skills.

The focus of this write-up is on the design, fabrication, and custom test infrastructure of the projects; a timeline for each project to gauge effort and hurdles; and current status. This write-up will not delve into the underlying research, microarchitectures, or performance results.

2. H3 Project

Project Overview

In the H3 project, we are building a 3D Heterogeneous Multi-core Processor ("H3" stands for "heterogeneity in 3D"). The 3D IC will consist of two 3D-stacked superscalar cores. The two cores are a 1-wide superscalar and a 2-wide superscalar, with respect to peak instruction fetch rate. The 1-wide core has smaller ILP-extracting structures (active list, issue queue, LQ/SQ, and physical register file) than the 2-wide core. Both cores have three execution lanes (3-wide issue): simple/complex integer ALU lane, load/store lane, and branch lane.

H3 features Fast Thread Migration (FTM) and Cache-Core Decoupling (CCD). FTM is a bulk swap of the architectural register state of the two cores, enabling fine-grain switching of threads between the two cores. CCD is the ability for a core to access either its caches or the other core's caches. This can be used to avoid migration-induced misses at the expense of higher hit latency due to crossing clock domains. Migrations can be initiated either from an off-chip interrupt signal (referred to as a global migration) or from the program itself via a new migrate instruction (a local migration). FTM and CCD require low-latency high-bandwidth interconnect. 3D face-to-face bonding is the ideal technology to meet this requirement.

More details can be found in our ICCD 2013 paper [2]. That paper was published after the first tapeout (2D version of two-core stack) and before packaging, PCB design, and chip bring-up.

The project has two phases. In the first phase, we built a 2D version of the two-core stack to test the design, compiled memories, *etc*., and to work out the physical design flow. The 2D version also contains a "debug core" as explained below. The first phase is well into chip bring-up. Meanwhile, we have begun the second phase, which is to partition the design for 3D stacking, fix functional bugs and performance bugs found during the first phase, and incorporate new features that did not make it into the first phase.

Design and Verification

Figure 1 summarizes all of the RTL design and physical design tasks performed prior to the May 20, 2013, tapeout of the 2D version, via a timeline (obtained by carefully combing through hundreds of project emails). In the workshop, we will use this timeline and the one in the chip bring-up section to initiate discussions. There are several key takeaways.

- FabScalar accelerated the RTL design in that we did not have to design the superscalar cores, and this leverage applied to two different core types. Instead, as the timeline shows, the RTL design effort focused on implementing caches and their off-chip buses (custom I$ and retooled OpenSparc T2 D$), adapting the Fetch-1 stage and load/store lane for synchronous read/write compiled memories, implementing novel features (FTM, CCD, performance counters), and implementing a third independent low-risk debug core. The RTL design effort took 9 months and 4 students, measured from RTL generation of the two raw FabScalar core types in Jan. 2012 to the RTL freeze in Sep. 2012. Moreover, some students were separately performing research tasks and taking classes in this period.
- Significant attention was paid to mitigating risk. Most notably, we included a third core separate from the two-core-stack called the debug core. It has the same microarchitecture as the 2-wide core in the two-core-stack, except it does not have the new features of FTM and CCD. It has small synthesized scratchpads for instructions and data, instead of complex caches and SRAM macros. It features full scan. Its only signal I/Os are for the five scan chains. Thus, the debug core does not depend on the compiled memories working; it is free from the risk of the highly complex OpenSparc T2 D$; it has a simple scan interface for loading the scratchpads with microbenchmarks; and its scan chains provide total observability and controllability for debug and escaped-bug circumvention. Essentially, the debug core provides a low-risk Plan B to test a pure FabScalar superscalar core. In terms of lowering risk in the two-core-stack, each core has dedicated request and response buses (for servicing miss requests), for four 1-byte buses total. Serializers/deserializers packetize requests and responses into 1-byte packets. Anticipating possible changes in pad allocations to the different chip experiments, the bus width is

parameterized in the RTL -- mitigating risk from a schedule standpoint. This aspect was exploited when the bus width had to decrease from 2 bytes to 1 byte.

- Interestingly, the 2D version of H3 was not originally planned. Both phases were supposed to be 3D. The Nov. 19, 2012, tapeout using the Tezzaron process was deferred until Mar. 2013, and then indefinitely. We decided to tapeout a 2D version with an IBM process and corresponding ARM IP which NCSU already had licensed, which in hindsight is very useful anyway for testing the design, flow, *etc*. In Dec. 2012, we unpacked the IBM 8RF PDK and ARM standard cells, pads, and memory compilers. We missed the first tapeout opportunity (Feb. 2013) and made the second one (May 2013). Figure 2, Table 1, and Table 2, show the final layout, physical design data, and flow execution time, respectively.

Our RTL verification plan was not up to commercial standards but we felt it was adequate for the project goals. We primarily used 100 million instruction SimPoints of SPEC benchmarks to stress the cores. FTM and CCD were tested using these same benchmarks, with either global migration interrupts asserted by the pin-accurate testbench or local migrate instructions inserted into the benchmark. Different clock domains and different frequencies were tested.

A serious concern was the complexity and lack of time in getting post-synthesis/post-layout netlist simulation going. We only got it going just after tapeout. Indeed, post-synthesis netlist simulation caught a serious, but not fatal, bug caused by a misplaced `ifdef. The verilog testbench defined it but the synthesis script did not. It is described in the errata Table 3 (issue #1), and will be discussed at the workshop as two key lessons. First, simulation-related instrumentation should be avoided in the synthesizable RTL altogether and consolidated in the testbench. Then there is no chance of misplaced instrumentation. Second, leave enough time for post-synthesis and post-layout netlist simulation. Gate-level simulation is challenging to get initiated, slow, and painstaking to debug.

Post-synthesis/post-layout netlist simulation would have also caught probable hold-time violations in the OpenSparc T2 D$, which we encountered in chip bring-up (Table 3, issue #5). As to whether or not these were flagged by static timing analysis in Design Compiler, Prime Time, and/or Encounter, we would have to revisit and audit the timing reports. In any case, timing violations in netlist simulation are persuasive as they manifest as functional bugs (*e.g.*, propagation of x's). The fact that netlist simulation models delays is also one of the reasons it is challenging to get started (correctly annotating the netlist with Standard Delay Format (.sdf)).

***About second phase (3D version of H3)*:**

We initially planned to use a Tezzaron 3D process with (1) face-to-face bonding for FTM and CCD between stacked cores and (2) through-silicon-vias (TSVs) to stacked DRAM. Plans with Tezzaron have not materialized so we are scaling back the 3D aspect to only implement face-to-face bonding. Without stacked DRAM, the cores need pads for their memory buses, as before. Conventional pads are out because the top metal layers of the two chips are bonded to each other. We will use a Ziptronix 3D process with face-to-face bonding (recommended 8μm bond pitch) and pads implemented in the Metal-1 layer of the top (flipped) chip. The pads are exposed by thinning the top chip and back-etching through its substrate to the pads. The pads will then be wirebonded to a conventional package.

We are working towards an August 2015 tapeout. A major change in the design is replacing the OpenSparc T2 D$ with a D$ designed in-house (the D$ from the AnyCore project). The open-source D$ was ill-fated due to its pervasive use of latch-based design and its highly structural style. Latches are explicitly instantiated in the indexing paths of all RAM instances (among other places), presumably due to the use of custom RAMs with complementary latches embedded somewhere within the RAM. We kept the leading latches and adjusted the clocks to our compiled RAMs. Issue #5 in Table 3 describes a load that never retires due to repeatedly missing in the cache (the core replays a cache-missed load until it hits). Netlist simulation shows the tag not being written to the correct set (set index goes to x's), owing to a hold-time violation in the set index for the tag fill. (Table 3 discusses the workaround.)

Packaging, PCB Design, and Chip Bring-up

Figure 3 shows a timeline for three post-silicon tasks: packaging, printed circuit board (PCB) design, and chip bring-up. This "physical" part of the project is one of the most fascinating experiences and we will try to convey all of its facets at the workshop. Moreover, we took different approaches in the H3 and AnyCore projects (the latter leveraging lessons from the former), providing interesting contrasts. Here are some highlights from the H3 project:

- The 2D version of H3 has 400 pads divided into four experiments (see Table 5), each experiment being allocated 100 pads. A 400+ pin package is too sophisticated for an academic project (in our opinion). We opted for a 128-pin gull-wing Quad Flat Package (QFP). This shifted the complexity to wirebond diagrams and PCBs: we needed four separate wirebond configurations and PCB designs. This effort was supported by research staff on the project.
- We opted for two routes with the debug core experiment: packaged (not pictured) and chip-on-board (Figure 4). We learned some hard lessons about lead forming and PCB assembly from the former. The latter, done in-house with a wire bonder in our lab, was successful in performing basic liveness tests (no short circuit between Vdd/V+ and Gnd, scan out == scan in, *etc*.) and ultimately retiring millions of instructions of microbenchmarks. In-house wirebonding was practical thanks to the debug core having only a dozen scan pads to wirebond (and living on the edge with only a few power and ground pads wirebonded).
- After initial success with the debug core, we moved on to testing the two-core stack. Figure 5 shows its package surface-mounted onto its 4-layer PCB. On the underside of the PCB is an FMC LPC connector for attaching the PCB as a mezzanine card to a Xilinx ML605 FPGA board. All signal I/Os go to both the LPC connector and headers, the latter allowing for debugging independent of the FPGA board. Arbitrary testbenches are synthesized to the FPGA and it also services L1 instruction and data cache misses from the cores. Elliott developed a convenient GUI and software backend on the host PC for downloading and running testbenches. Moreover, he also developed a compiler for compiling a language that lies somewhere between C and assembly, so that we can strictly control the types and ordering of instructions in benchmark programs without literally writing assembly. Using this setup, we found several issues with the setup itself (current meter causing flaky processor reset) and bugs in the chip. These are documented in the errata Table 3. Results of several

microbenchmarks are shown in Table 4.

3. AnyCore Project

Project Overview: The released FabScalar toolset generates cores by piecing together pipeline stages of the desired widths/depths from a pipeline stage library. Internally, we have been using a newer version of FabScalar called the "superset core": a single verilog description in which pipeline stage widths are parameterized (structure sizes were already parameterized). The AnyCore design began with the superset core. Its *static configurability* was preserved so that AnyCore processors of different maximum widths and sizes can be synthesized. This aspect proved quite convenient as we had to scale back our original maximally configured AnyCore (from 6-way to 4-way maximum fetch width) to fit comfortably in terms of passing DRC and LVS without heroic physical design effort, and to allow for the addition of small L1 caches. The key functionality added to the superset core for AnyCore was *dynamic configurability*: the ability to dynamically configure superscalar widths and structure sizes. Table 6 shows the adaptive microarchitecture features for the AnyCore design point that was fabricated.

Design and Verification: Please see Figure 6 for tasks and timeline. Notable features for testability: We implemented custom L1 caches, which can be configured in three modes: cache, scratchpad, and BIST. A reset configures for scratchpad mode where the first N rows are preset to a test loop that toggles a pin. On April 9, we saw the BIST succeed! The chip also has a debug interface for directly reading/writing key structures (scratchpads/caches, register file, PC, *etc*.) as well as partial scan for backup.

Packaging, PCB Design, and Chip Bring-up: Learning from the H3 project, we took a different approach. We packaged and lead-formed through MOSIS using a 100-pin gull-wing QFP (CQFP100-CQZ10001). We consulted with Agile Enterprises to find a good socket (FPQ-100-0.5-06A). In parallel with the sockets arriving (6 weeks), we designed a 4-layer PCB to which the socket, LPC connector, headers, DCAPs, and bypassable shunt resistor are assembled. Each step along the way, we checked for potential problems and found none so far: peered inside the package to examine the chip's orientation, bond wires, and downbonds; tested fit and connectivity of the package in the socket; tested bare boards; tested for Vdd/Gnd and V+/Gnd shorts with the package in the socket of assembled board (Figure 9). Finally, we applied power and clock, held scan enables low, and looked for the toggling BIST pin – and it succeeded on April 9, 2015. The socket approach is potentially scalable to other pad-conforming designs, avoids being hamstrung with a potentially faulty chip via easy substitution, and allows testing of variations.

4. Acknowledgments

The H3 project is supported by a grant from Intel. The AnyCore project is supported by NSF grant CCF-1018517. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the National Science Foundation.

5. References

[1] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg. FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template. Proceedings of the *38th IEEE/ACM International Symposium on Computer Architecture (ISCA-38)*, pp. 11-22, June 2011.

[2] E. Rotenberg, B. Dwiel, E. Forbes, Z. Zhang, R. Widialaksono, R. Basu Roy Chowdhury, N. Tshibangu, S. Lipa, W. R. Davis, and P. D. Franzon. Rationale for a 3D Heterogeneous Multi-core Processor. Proceedings of the *31st IEEE International Conference on Computer Design (ICCD-31),* pp. 154-168, October 2013.

Appendix A: H3 Project

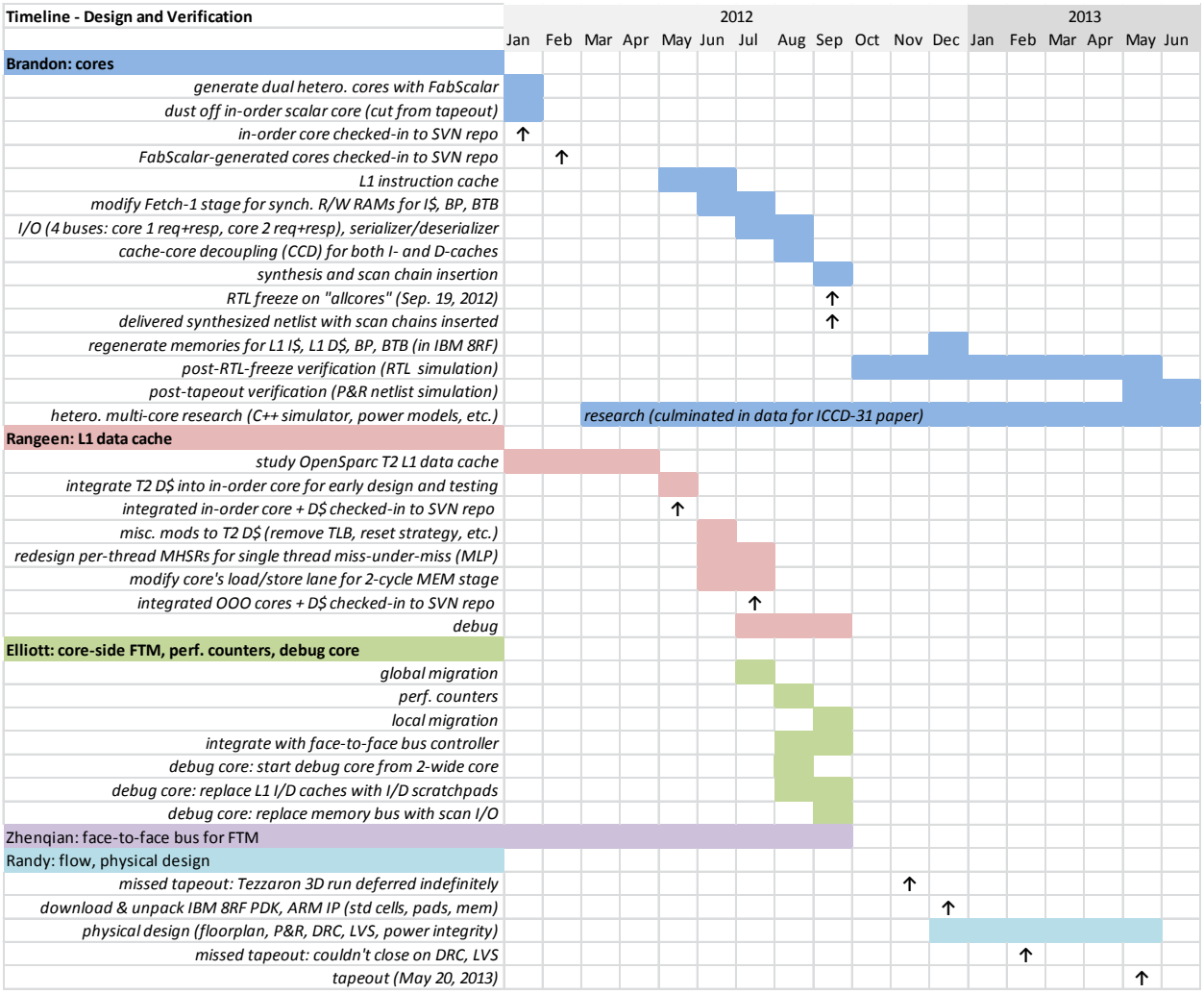
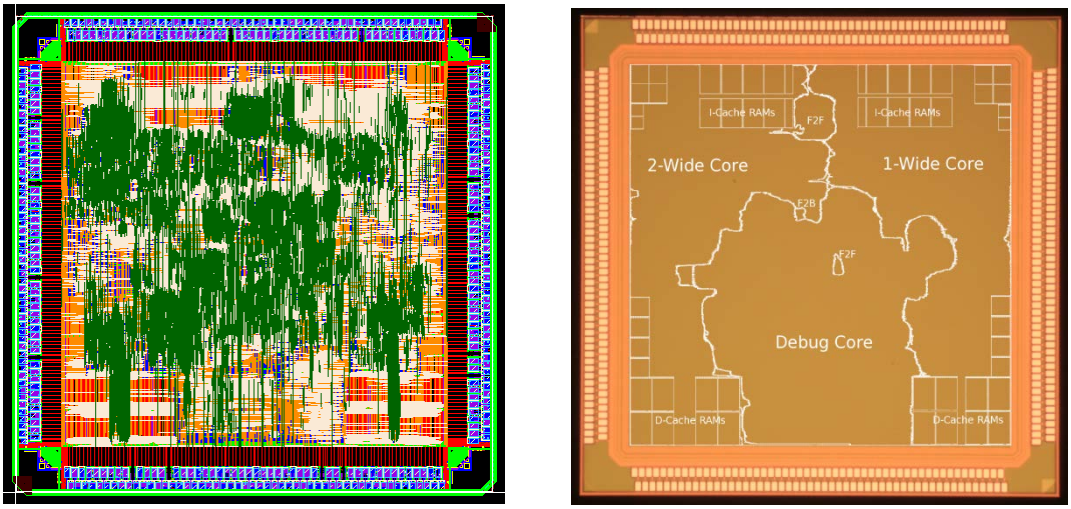


Figure 1. Timeline of RTL design and physical design tasks for May 20, 2013, tapeout of 2D version of H3.



(a) Final layout. (b) Die photo, with floorplan superimposed.

Figure 2. H3 processor (2D version).

Table 1. Physical design data.

Technology	IBM 8RF (130 nm)
Dimensions	5.25 mm x 5.25 mm
Area	27.6 mm^2
Transistors	14.6 Million
Cells	1.1 Million
Nets	721 Thousand
Memory macros	56
Clock domains	10

Table 2. Flow execution time.

Encounter	9 hours
Calibre DRC	4.5 hours
Calibre LVS	2 hours

Table 3. Setup issues and chip bugs found thus far in the H3 project.

	Symptom	**Description**	**Stage**	**Workaround**
1	Instructions that depend on loads will wake up early when the load misses (reading an incorrect value from the register file/bypass).	A misplaced `ifdef guarded key RTL during simulation, but the `ifdef condition was not enabled during synthesis.	post-tapeout	Prefetch blocks such that loads will hit.
2	Source-synchronous clock port errors in FPGA testbench during PAR/mapping.	Clock signals from the cores to the FPGA were not routed to clock-capable pins of the LPC.	post-PCB-assembly	Use a jumper from the clock header of the PCB to a clock-capable pin of the XM105 debug card attached to the HPC connector of the ML605.
3	Non-repeatability in running single threaded test programs.	High precision ammeter used to measure logic current. Current draw of running core caused the ammeter to switch shunt resistance (with a relay) immediately after reset was de-asserted. During the switch, the supply as seen by the core dropped below threshold voltage, causing metastability.	post-silicon testing	Use dedicated shunt resistor, and measure voltage drop across that resistor instead of the in-line ammeter.
4	Repeated I-cache misses to the same block.	- Frequently-executed and mostly-taken branch in the last slot of a cache block. - A BTB update occurs when the branch is being fetched (Fetch-1 stage), and the update is for this branch. Thus, there is a BTB write and a BTB read to the same entry, that of the branch. The BTB read indicates a miss due to the concurrent read and write. This causes Fetch-1 to fetch the next sequential block, generating an I-cache miss.	post-silicon testing	This issue does not impact performance, but does make it difficult to measure the number of actual I-cache misses. It is possible to add NOPs to ensure frequently-executed and mostly-taken branches are not in the last slot of a cache block.

		- The Fetch-1 stage is redirected to the branch's taken target in the next cycle, when the branch is predecoded in the Fetch-2 stage. - Meanwhile, the I-cache miss request for the sequential block is not canceled. Further, the retrieved block is dropped by the core if the Fetch-1 stage is fetching instructions. - End result: The same miss (of the sequential block) is generated repeatedly.		
5	Cache-missed load never retires.	After block is retrieved from memory, netlist simulation shows the tag not being written to the correct set (set index goes to x's), owing to a hold-time violation in the set index for the tag fill. The core replays a cache-missed load until it hits. After the MHSR performs the flawed line-fill, the replayed load will miss again, generating another miss request for the same block. This repeats indefinitely.	post-silicon testing	*Top Core*: Hold-time violation on fill path seems to affect only the tag fill, not the data fill. Further, it appears that uninitialized tags in the tag SRAM are, by chance, 0. Thus, limiting load and store addresses to have tags of 0 masks the flawed tag fill. The workaround is not guaranteed but does seem very stable. *Bottom Core*: Not only does the load's tag need to be 0, but we also had to lower Vdd to slow circuit paths. With these two measures, the replayed load hits and retires. However, the replayed load does not appear to get the correct data, presumably due to the data fill path being compromised (conjecture).
6	Inability to read/write TRF registers.	Attempted read/write of TRF registers in a single-thread experiment. The reset of the F2F controller is fed through clock-synchronizing flip-flop pairs, but the clock of the F2F controller was not running (since it was a single thread experiment).	post-silicon testing	Ensure F2F controller is clocked when either core is clocked, even when migrations are not being tested.
7	Execution deadlock after a thread migration.	The BTB is implemented in SRAM, including its valid bits, so valid bits are not reset after a migration. This caused BTB hits on non-control instructions. False BTB hits are detected and recovered in Fetch-2 stage, except CTIQ is still pushed. Those instructions would allocate entries in the CTIQ, but during retirement, the CTIQ entry was not de-allocated. The CTIQ would fill, and fetch would stall indefinitely.	post-silicon testing	Carefully craft test programs such that instruction addresses do not overlap. Hard reset tends to work, but not guaranteed.
8	Repeated thread migrations only work for fewer than 33 migrations.	The MIGRATE instruction is in a loop. If the loop-ending branch is fetched before the MIGRATE instruction retires, then a CTIQ entry will be allocated for the	post-silicon testing	Add instructions (NOPs if necessary) after a MIGRATE instruction to guarantee no branches can be fetched before the MIGRATE retires.

		branch. However, after the migration, the CTIQ is not reset, so the allocated entries are not de-allocated, eventually filling the CTIQ, blocking forward progress.		
9	Unable to repeatedly migrate from core-to-core when CCD is enabled.	Debugging in progress	post-silicon testing	Unknown

Table 4. Microbenchmark results.

Microbenchmark	Core	Static instr.	Dynamic instr.	Cycles	IPC	Current (mA)	Energy (mJ)	Comments
PRNG *pseudo-random number generator*	1-wide	28	67.1M	67.1M	1.00	8.53†	59.6	Peak IPC achieved.
	2-wide			64.4M	1.04	5.78	46.5	IPC>1 due to branches. IPC not much greater than 1 because only one simple/complex ALU lane, and load/store lane unused.
PRIMES *prime number generator*	1-wide	28	88.2M	100.4M	0.88	8.35†	87.3	OOO tolerates 14-cycle integer divide instruction well.
	2-wide			301.8M	0.29	5.70	215.0	Hypothesis: larger IQ and non-age-based scheduler exacerbates priority inversion, stalling retirement more.
ARRAY-SUM *sum elements of an array*	1-wide	35	41.9M	Did not finish				See Table 3, issue #5.
	2-wide			20.9M	2.00	6.54	17.1	Peak IPC achieved.
BUBBLE *bubble sort, list initially reverse sorted*	1-wide	73	2.5M	Did not finish				See Table 3, issue #5.
	2-wide			8.5M	0.29	5.24	5.57	Early diagnosis: SQ stalls dominate other resource stalls. Hypothesis: Limited store buffer size of write-through D$ causing back-pressure. Frequent swaps imply many consecutive stores. Write-through latency to FPGA is high.
MIGRATE *Migrate instr. in loop*								Completes 1 million consecutive thread migrations correctly. Cores at different frequencies. ~1 migration per 1K cycles.

† 1-wide core has higher current (and power) than 2-wide core because it has full scan.

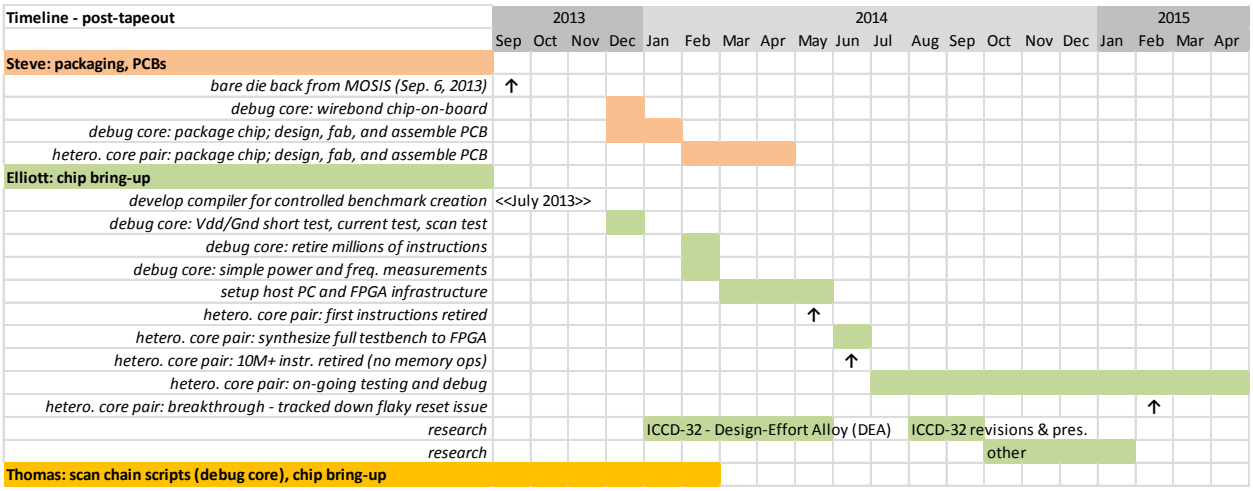


Figure 3. Timeline of packaging, PCB design, and chip bring-up, for 2D version of H3.

Table 5. The H3 processor (2D version) has 400 pads and supports four separate experiments each with 100 dedicated pads. With only a 128-pin package, each experiment has a different wirebond configuration (A through D).

Wirebond Configuration	Experiment	Signal/Supply Pads
A	Heterogeneous core pair	63/37
B	Debug core	13/87
C	Isolated F2F bus	62/38
D	Isolated F2B bus	49/51

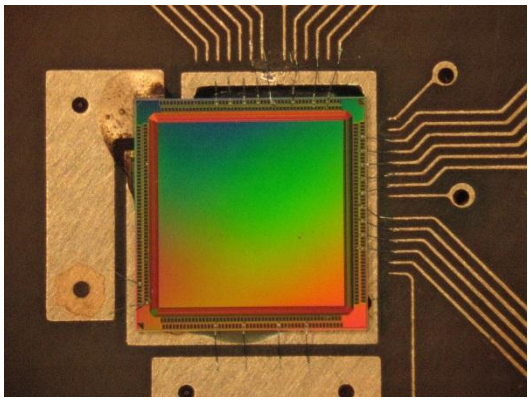


Figure 4. Chip-on-board setup for testing H3's debug core. Wirebonding was done at NCSU due to feasible number of bonds.

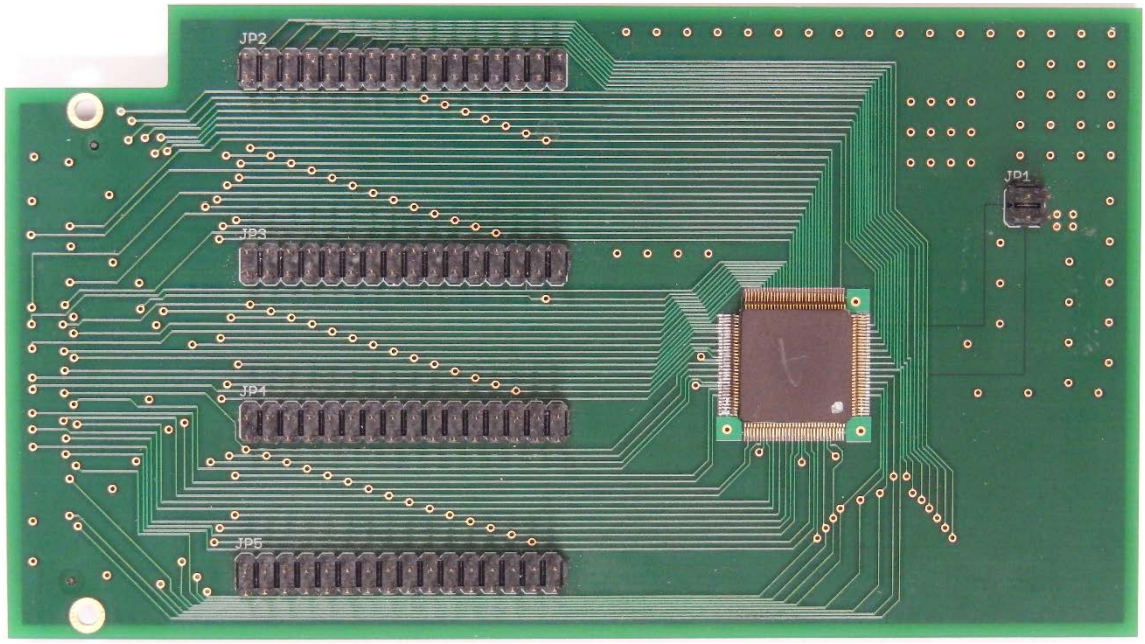


Figure 5. Package for H3's heterogeneous core pair, surface-mounted onto its 4-layer PCB. Underside of PCB has the FMC LPC connector for attaching the PCB as a mezzanine card to the Xilinx ML605 FPGA board.

Appendix B: AnyCore Project

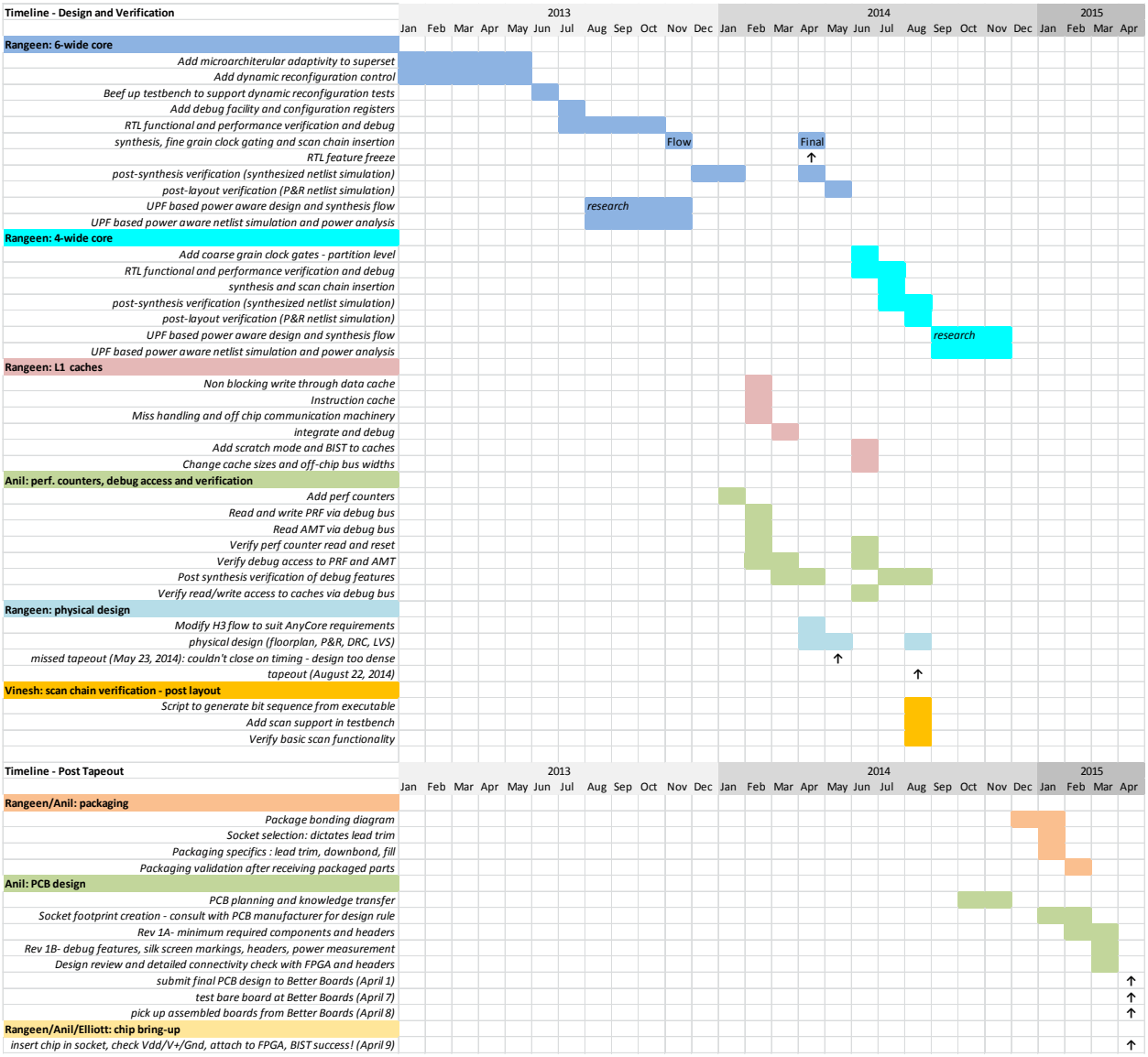


Figure 6. AnyCore project timeline.

Table 6. Microarchitectural adaptivity of AnyCore chip.

Adaptive microarchitecture feature	*Configurations*
fetch/dispatch width (instructions/cycle)	1, 2, 3, 4
issue width (instructions/cycle)	3, 4, 5
physical register file & active list	64, 96, 128
load and store queues (each)	16, 32
issue queue	16, 32, 48, 64

Table 7. Physical design data.

Technology	IBM 8RF (130nm)
Dimensions	5 mm x 5 mm
Area	25 mm^2
Pads (signal, power)	100 (79, 21)
Transistors	3.4 million
Cells	1.5 million
Nets	7.6 million

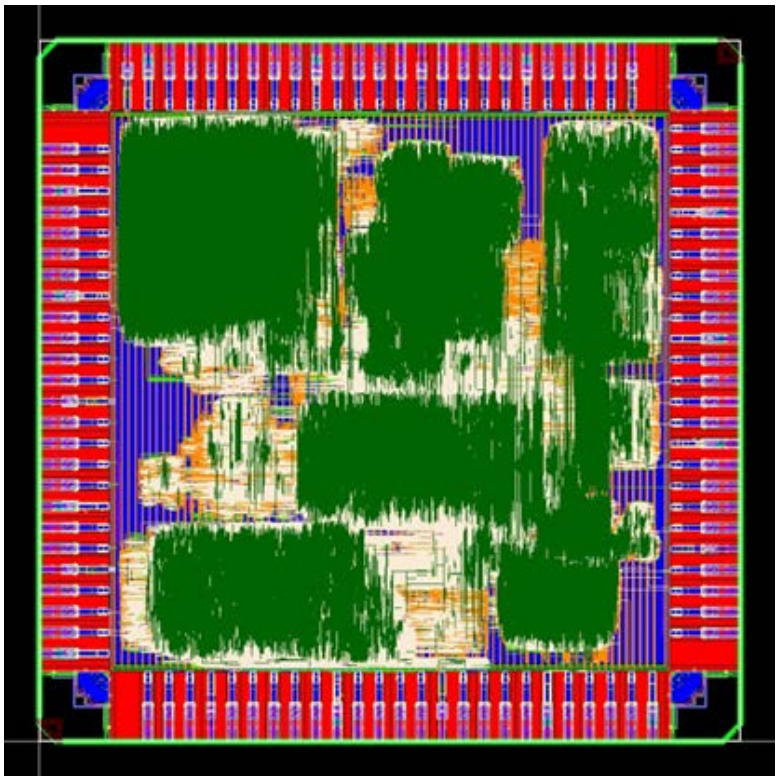


Figure 7. Layout of AnyCore prototype.

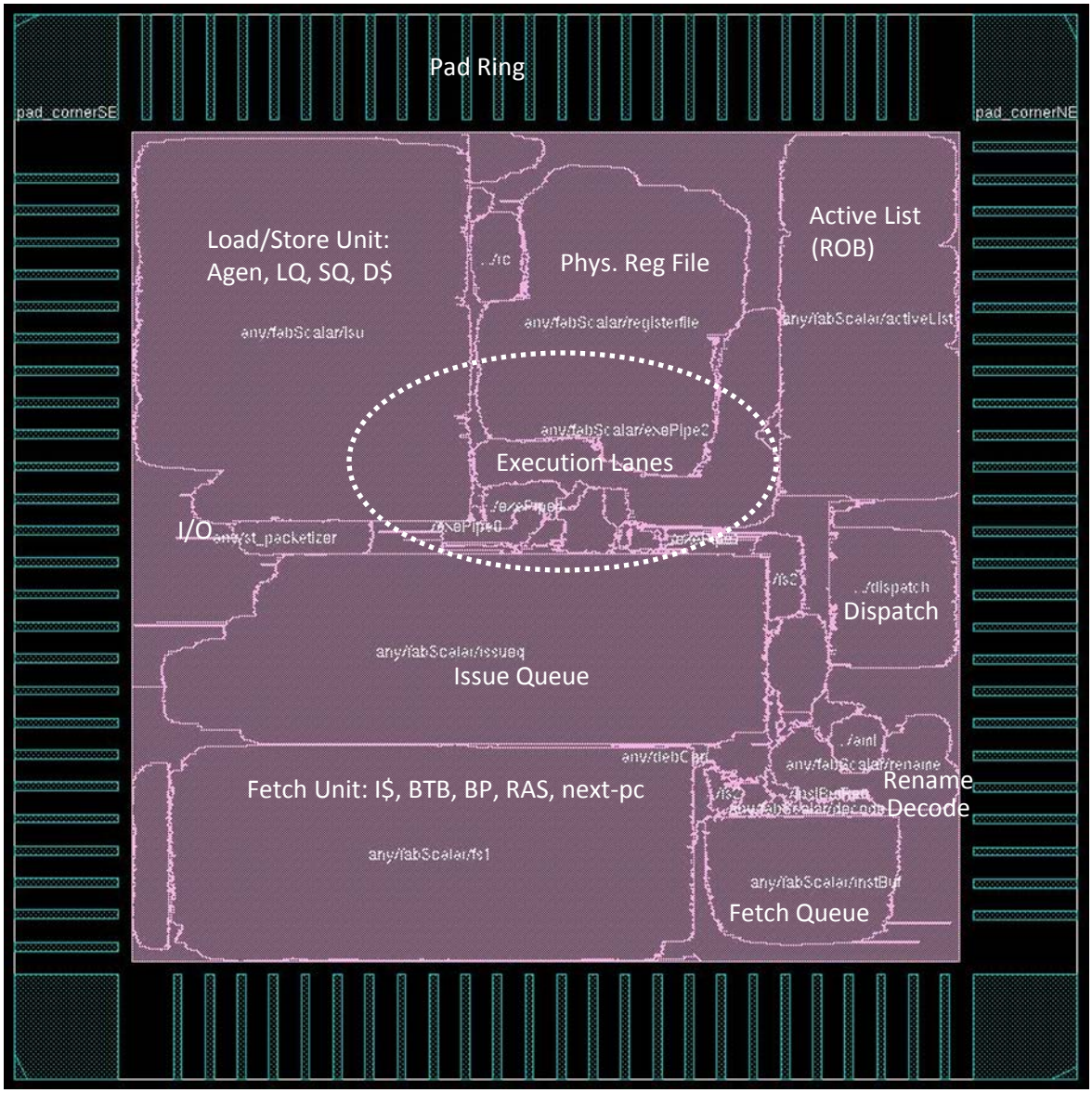


Figure 8. Floorplan of AnyCore prototype.

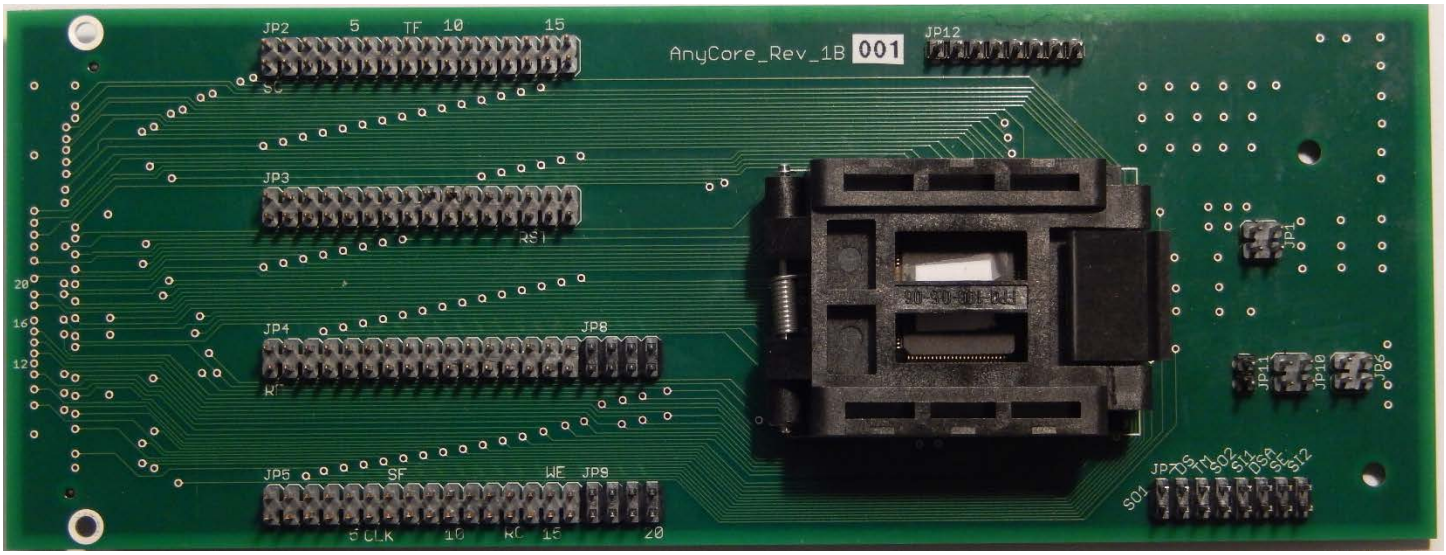


Figure 9. AnyCore package inserted into socket which is assembled on a 4-layer PCB. Underside of PCB has the FMC LPC connector for attaching the PCB as a mezzanine card to the Xilinx ML605 FPGA board. The narrow board width is the same as the LPC connector, so it can be attached to arbitrary Xilinx boards (such as Zynq boards).