AFRL-RY-WP-TR-2023-0137

OPEN-SOURCE ROOTED DESIGN EXPERTS WITH REPUTE (ORDER)

David Wentzlaff
Princeton University

SEPTEMBER 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-2023-0137 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//Signature//

CHRISTOPHER A. BOZADA
Program Manager
Aerospace Components and Subsystems Division

//Signature//

GENE M. WILKINS, Lt Col, USAF
Deputy Chief
Aerospace Components & Subsystems 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
September 2023	Final	11 June 2018	13 June 2022

4. TITLE AND SUBTITLE
OPEN-SOURCE ROOTED DESIGN EXPERTS WITH REPUTE (ORDER)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8650-18-2-7846	N/A	62716E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
N/A	N/A	Y1TC

6. AUTHOR(S)
David Wentzlaff

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Princeton University
171 Broadmead St
Princeton, NJ 08540

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
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

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RYD

11. SPONSOR/MONITOR'SREPORT NUMBER(S)
AFRL-RY-WP-TR-2023-0137

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

13. SUPPLEMENTARY NOTES
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 FA8650-18-2-7846. 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
This effort has defined requirements for different Electronic Design Automation Computer Aided Design tools, provided and curated example designs and benchmarks leveraging past designs, used the Intelligent Design of Electronic Assets (IDEA) program developed tools, provided feedback and bug reports for the IDEA tools, taped-out two chips using IDEA created tools, and helped with the OpenROAD tape-in using a design provided by the ORDER team to the OpenROAD team.

15. SUBJECT TERMS
open-source chips, using open source EDA tools

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	40

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

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

PREVIOUS EDITION IS OBSOLETE.

Table of Contents

Section **Page**

List of Figures ii
ACKNOWLEDGEMENTS AND DISCLAIMER 1
1 INTRODUCTION 2
2 SUMMARY 3
3 RESULTS 4
3.1 Designs, Benchmarks, Flows, and Tools 4
3.2 OpenPiton Design Benchmarks (OPDB) 4
3.3 BSG Micro Designs 6
3.4 BSG Pipeclean Suite 7
3.5 IDEA Dimensionless Floorplan (IDF) and IDF Tools 9
3.6 BSG Fakeram SRAM Front-End Generator 10
3.7 BSG SV2V - SystemVerilog to Verilog Converter 12
3.8 UW OpenROAD Free45 Reference Flow 13
3.9 Using IDEA tools, Providing Feedback, and Interactions 15
3.9.1 BlackParrot Tape-In Consultation 15
3.10 ParrotPiton 18
3.11 ORDER Digital Chip Tape-out 19
3.12 BSG SSTL Chip 20
3.13 TurboXAUI 22
3.13.1 Non-Uniformly Quantized SAR ADC 23
3.13.2 NUSA with Code-Dependent Weight 24
3.13.3 Sub-Sampling CDR - PI Controller 27
3.13.4 Sub-Sampling CDR - PI 28
3.13.5 Sub-Sampling CDR - Encoder 29
4 DISCUSSION 31
4.1 Impact and Workforce Development 31
4.2 Challenges 31
5 CONCLUSIONS 33
6 REFERENCES 34
LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS 35

List of Figures

Figure **Page**

Figure 1: Different Modules in the OPDB design 4
Figure 2: Graphical Representation of the Module Hierarchy for OPDB Designs 5
Figure 3: Characterization of Different Pre-built Components of the OPDB Design 5
Figure 4: BSG Micro Design Directory Structure 7
Figure 5: Available Designs within BSG Pipeclean Suite and their Size in Approximate Number of Gates 8
Figure 6: Design File Descriptions 9
Figure 7: Two Primary use cases for IDF files to be used for Information Transfer 10
Figure 8: Flow Diagram for BSG Frakeram 11
Figure 9: Image of a BSG Fakeram Macro Integrated and Routed into a Digital Design 12
Figure 10: BSG SV2V Flow Diagram 13
Figure 11: UW OpenROAD Free45 Steps 14
Figure 12: Example Layouts Done using the UW OpenROAD Free45 Reference Flow 14
Figure 13: ParrotPiton Block Diagram 18
Figure 14: Layout of the FPGA Design with a Zoom-in on a Single CLB 19
Figure 15: SSTL Test Chip Schematic 21
Figure 16: SSTL Test Chip Layout 22
Figure 17: Turbo-XAUI block diagram 23
Figure 18: Turbo-XAUI simulation results 23
Figure 19: Transfer Function of a 5-bit Simple NUSA, i.e., NUSA-CIW 24
Figure 20: Threshold Voltage Trees of 4-bit ADCs 25
Figure 21: Transfer Functions of a 3-bit NUSA-CDW and 3-bit NUSA-CIWs 25
Figure 22: Circuit Diagram of a 5-bit NUSA-CDW 26
Figure 23: Performance Summary of NUSA-CDW 26
Figure 24: Transfer Function and BER vs. SNR of the 5-bit NUSA-CDW Circuit and an Ideal 5-bit Uniform ADC 27
Figure 25: (a) Timing Diagram of PI Controller, (b) PI Controller Output Code in XAUI Mode and Turbo-XAUI Mode 28
Figure 26: (a) Complementary Phase Interpolation Cell and (b) Timing Diagram 29
Figure 27: Simulation Result of Complementary PI Cell (a) DNL/INL and (b) Output Duty vs. Input Duty 29

ACKNOWLEDGEMENTS AND DISCLAIMER

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7846. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

We wish to thank the three DARPA MTO program managers Andreas Olofsson, Serge Leef, and James Wilson who provided support and feedback on this work. We also wish to thank the DARPA team and AFRL team who provided support to this work.

1 INTRODUCTION

The DARPA Intelligent Design of Electronic Assets (IDEA) program seeks to reduce the time and cost of creating future computer chips and Systems on Chip (SoC) by developing technologies to take designs and translate those designs into low-level physical chip layout (GDSII) in less than 24 hours with no-human intervention in the loop. By doing so, the program seeks to make the creation of next-generation electronic systems faster and less expensive for the Department of Defense (DoD). The program has funded different performers to create Electronic Design Automation (EDA) tools that are primarily open-source to attempt to accomplish this goal. The IDEA program used innovative integration exercises to encourage the performers to meet, share designs and code, and work together to fix complex issues.

As part of the IDEA program, the ORDER: Open-Source Rooted Design Experts with Repute team has worked to aid the different IDEA EDA tool designers in multiple ways. In particular, the ORDER team has defined requirements for different EDA Computer Aided Design (CAD) tools, provided and curated example designs and benchmarks leveraging past designs, used the IDEA developed tools, provided feedback and bug reports for the IDEA tools, taped-out two chips using IDEA created tools, and significantly helped with the OpenROAD tape-in using a design provided by the ORDER team to the OpenROAD team. In addition, the ORDER team attended the IDEA integration exercises providing support, feedback on tools, and helped different performers use the designs that were curated by the ORDER team.

2 SUMMARY

The ORDER team has worked as an independent team of designers to provide requirements for the IDEA tools, provide helpful infrastructure, advise the IDEA tool performers and use the tools providing feedback. In particular, the highlight achievements of the ORDER team include:

- Provided Requirements, Infrastructure, and Advising for IDEA Tool Performers
- Open-Source release of CAD Benchmarks for EDA Community
- Provided Feedback and Bug reports for IDEA Tools
- Designed a TurboXAUI link
- Open-Source Chip Tape-outs using IDEA Tools including two chips:
    - Open-Source High-Speed DDR3 SSTL I/O Design
    - Open-Source FPGA using OpenROAD tools

In the following sections, this report discusses these main achievements in more detail.

3 RESULTS

3.1 Designs, Benchmarks, Flows, and Tools

Multiple designs were created or curated as part of the ORDER project. These designs were designed to fill the needs of the different ORDER IDEA tools, in particular, there is a mixture of analog and digital designs and the designs vary in size from very small designs all the way up to 100's of processor cores. Many of these designs have been curated and organized into benchmark suites and work has been done to make them easier for the EDA tool user to use and integrate as test cases for their tools. In addition to designs and benchmarks, the ORDER team has created different tool flows and point-tools that have helped facilitate the IDEA program. The following sections describe these different designs, benchmarks, tool flows, and point-tools.

3.2 OpenPiton Design Benchmarks (OPDB)

The ORDER team has built and curated a set of designs based off of OpenPiton integrated together with different accelerators that can enable other researchers and CAD-tool designers to use large designs without needing to understand the design or a makefile flow called the OpenPiton Design Benchmark (OPDB).

Module name	Top module	Macros	X-dim	Y-dim	Topology	L1-I	L1-D	L1.5	L2	Hardware Verified
chip	chip	✓	✓	✓	✓	✓	✓	✓	✓	FPGA/Tapeout
chip_bridge	chip_bridge									FPGA/Tapeout
dynamic_node	dynamic_node_top_wrap				2dmesh					FPGA/Tapeout
	dynamic_node_top_wrap_para				xbar					FPGA
fpga_bridge_rcv_32	fpga_bridge_rcv_32									FPGA
fpu	fpu									FPGA/Tapeout
ifu_esl	sparc_ifu_esl									FPGA/Tapeout
ifu_esl_counter	sparc_ifu_esl_counter									FPGA/Tapeout
ifu_esl_fsm	sparc_ifu_esl_fsm									FPGA/Tapeout
ifu_esl_htsm	sparc_ifu_esl_htsm									FPGA/Tapeout
ifu_esl_lfsr	sparc_ifu_esl_lfsr									FPGA/Tapeout
ifu_esl_rtsm	sparc_ifu_esl_rtsm									FPGA/Tapeout
ifu_esl_shiftreg	sparc_ifu_esl_shiftreg									FPGA/Tapeout
ifu_esl_stsm	sparc_ifu_esl_stsm									FPGA/Tapeout
l15	l15_wrap	✓			✓		✓	✓		FPGA/Tapeout
l2	l2	✓			✓				✓	FPGA/Tapeout
MIAOW (GPGPU)	neko	✓								FPGA/Tapein
pico	picorv32									FPGA/Tapeout
sparc_core	sparc_core	✓				✓	✓			FPGA/Tapeout
sparc_exu	sparc_exu_wrap									FPGA/Tapeout
sparc_ffu	sparc_ffu_nospu_wrap	✓								FPGA/Tapeout
sparc_ifu	sparc_ifu	✓				✓				FPGA/Tapeout
sparc_lsu	lsu	✓					✓			FPGA/Tapeout
sparc_mul	sparc_mul_top_nospu_wrap									FPGA/Tapeout
sparc_tlu	tlu_nospu_wrap									FPGA/Tapeout
tile	tile	✓			✓	✓	✓	✓	✓	FPGA/Tapeout
FFT	fftmain									FPGA
GNG	gng									FPGA/Tapein

Figure 1: Different Modules in the OPDB design
Table from the OPDB TCAD paper [1].

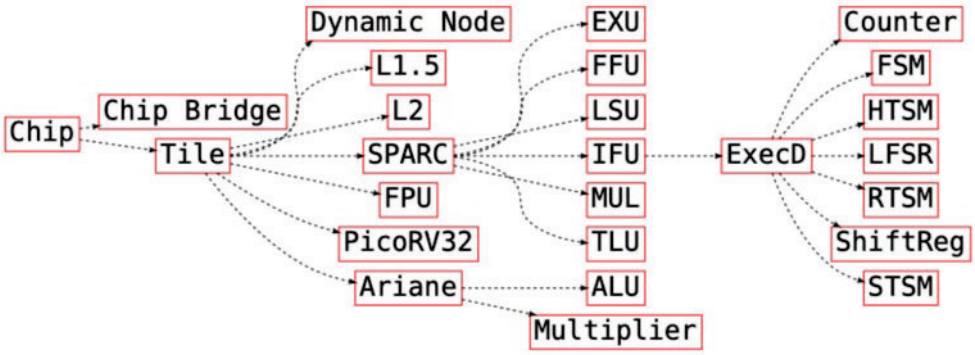


Figure 2: Graphical Representation of the Module Hierarchy for OPDB Designs
Image from the OPDB TCAD paper [1].

In particular, a key feature of this which makes it easy for other researchers and IDEA performers to use is that OPDB provides pre-built system designs that all fit in a single Verilog file. Therefore, others who want to use these designs to test their CAD tools do not need to import more than a single file. Also, the main OPDB designs solely use Verilog and therefore a working SystemVerilog flow is not needed. Figure 1 shows the different modules and their features that can be hierarchically used from OPDB and Figure 2 shows how the hierarchy fits together. By having different designs, a CAD tool designer can start out testing their tools with a simple OPDB design and build up to more complex designs.

Module	Cells	Nets	Ports	Area (kGE)	Delay (FO4)	Runtime	Memory Macros
ExecD multiple sub-components	<1000	<1000	<1000	<10	<20	short	0
OST1 Instruction Fetch Unit (IFU)	1,900	8,500	4,500	<10	<50	short	5
Ariane Arithmetic and Logic Unit (ALU)	2,900	3,000	270	<10	<40	short	0
ExecD Execution Drafting	3,000	3,400	400	<10	<30	short	0
OST1 Load/Store Unit (LSU)	3,600	14,000	8,000	<10	<50	short	5
GNG Gaussian Noise Generator	6,800	7,000	24	14	<40	short	0
OST1 Floating-point Frontend Unit (FFU)	5,300	5,700	560	33	<40	short	1
OpenPiton Dynamic Node	8,300	8,800	890	24	<40	short	0
OST1 Multiplier Unit (MUL)	11,000	11,000	200	26	<20	short	0
PicoRV32 RISC-V Core	11,000	11,000	620	27	<20	short	0
Ariane Multiplier	14,000	15,000	210	44	<30	short	0
OpenPiton Chip Bridge	24,000	25,000	1,700	87	<20	short	0
OST1 Floating-Point Unit (FPU)	26,000	28,000	1,100	75	<50	short	0
OST1 Trap Logic Unit (TLU)	31,000	33,000	3,100	110	<30	short	0
OpenPiton L1.5 cache	34,000	40,000	7,700	260	<50	short	7
OpenPiton L2 cache	42,000	47,000	6,700	820	<70	short	8
FFT Fast Fourier Transform	80,000	83,000	4,000	285	<30	short	0
OST1 EXecution Unit (EXU)	75,000	76,000	2,300	260	<40	short	0
Ariane RISC-V Core	130,000	150,000	14,000	860	<50	short	32
OpenSPARC T1 Core (OST1)	180,000	190,000	6,300	990	<60	medium	11
OpenPiton OST1 Tile	310,000	420,000*	23,000	2200	<70	short	26
MIAOW GPGPU	658,000	808,000	152,000	1800	<50	medium	212
OpenPiton OST1 Chip 3x3 (9 Tiles)	2,800,000	4,500,000*	210,000	20000	<70	short	234
OpenPiton OST1 Chip 5x5 (25 Tiles)	7,800,000	12,000,000*	570,000	56000	<70	medium	650
OpenPiton OST1 Chip 10x10 (100 Tiles)	31,000,000	50,000,000*	2,300,000	220000	<70	medium	2600
OpenPiton OST1 Chip 15x15 (225 Tiles)	70,000,000	110,000,000*	5,100,000	500000	<70	long	5850
OpenPiton OST1 Chip 19x19 (361 Tiles)	110,000,000	180,000,000*	8,200,000	810000	<70	long	9386

Figure 3: Characterization of Different Pre-built Components of the OPDB Design
Table taken from the OPDB TCAD paper.

In order to create the multitude of designs and in order to provide scale, OPDB uses two important tools/techniques. First, the ORDER team used the FuseSoC framework and its Core API (CAPI) description format to describe different components of the OPDB design. Then using FuseSoC it was possible to generate the appropriate hierarchy containing just the Verilog code needed. Once all of the code was identified, we used the Tursi framework which takes the needed files, runs them through a Python pre-processor (PyHP) in order to configure the design (things like cache sizes, number of cores, etc). And then finally, OPDB used the open source tool iverilog in a mode which allows the files to be "pickled" into a single Verilog file. Figure 3 shows the different flattened designs that OPDB contains by default and has designs all the way from a small portion of a processor's execution unit all the way up to a 361-processor design which is likely a greater than 5 billion transistor design in most processes. The hierarchical ORDER approach enables the creation of even larger designs. One other important portion of the ORDER approach is that it does not only address processor cores, but it also includes other accelerators including a Gaussian Noise Generator and a Fast Fourier Transform (FFT) accelerator as well as the MIAO GP-GPU.

The OPDB public repository includes example scripts for converting the files from Verilog to BLIF and AIGER formats. Furthermore, a skeleton for performing gate-level verification of the dynamic node modules using open-source CAD tools and the Nangate 45 library was included in the external repository to make it easier for others to use this work. OPDB is open source and can be found at the following location https://github.com/PrincetonUniversity/OPDB.

A paper with more details was published on OPDB in the June 2022 print issue of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems entitled, "OPDB: A Scalable and Modular Design Benchmark" [1].

Multiple other performers in the IDEA program have used OPDB as part of their CAD tool testing. In particular, the LSOracle and OpenROAD teams have used designs from OPDB as test cases for their tools being developed.

3.3 BSG Micro Designs

(https://github.com/bsg-idea/bsg_micro_designs)

To better support the development of open-source hardware CAD tools, the ORDER team has created BSG Micro Designs - a repository of small, unit-test style designs based around Basejump STL components. These designs cover a wide variety of common hardware components such as counters, FIFOs, memory devices, on-chip networks, IO systems, clock-domain-crossing, and more. Each component covers a diverse range of circuits which will stress different implementation and optimization engines within a traditional CAD flow, however the average runtime for the designs is between 10-30 minutes making them ideal for unit testing.

Figure 4: BSG Micro Design Directory Structure

At the top level are design categories (e.g. bsg_misc) followed by the component name (e.g. bsg_abs). Each component has several parameterizations creating a single design (e.g. bsg_abs_width_p_128) which has a single Verilog file (e.g. top.v) and a variety of timing constraint modes (e.g. sdc/vclk_30_FO4.constraints.sdc).

BSG Micro Designs has 185 different components. Each component has 1 or more specific parameterization of the component which creates the design. These parameterizations are representative of real parameterizations that might exist within a system and can be used to test the scalability of the CAD tools. There are 582 different component-parameterization pairings. Each design has a single pickled Verilog file called top.v that has been converted using bsg_sv2v to maximize the compatibility between the hardware description and the CAD tool front-end parser. Each design also contains several SDC timing constraints, however each run through the CAD tool should only source a single SDC file. These timing constraints represent different modes to evaluate the design from aggressive constraints that will require more area and power to more relaxed constraints that leave more opportunities for area and power optimizations. The timing constraints use fan-out-of-four (FO4) as the base time unit to make them as process independent as possible. By running the design through the CAD tool with varying modes, the CAD tool's performance can be evaluated on its ability to perform PPA optimizations based on varying constraints. There are 5984 different design-mode pairs within BSG Micro Designs.

3.4 BSG Pipeclean Suite

(https://github.com/bespoke-silicon-group/bsg_pipeclean_suite)

The ORDER team created BSG Pipeclean Suite - a repository containing designs used to stress test new CAD flows. These designs are intended to stress different aspects of the CAD flow with increasing complexity. The structure of each design is consistent to make it easy for CAD flows to find and link all required files for the design.

These designs make use of the IDEA Dimensionless Floorplan (IDF) specification. Floorplan and timing constraint files are dimensionless meaning that all units are expressed in terms of

properties of the process rather than concrete units (such as microns or picoseconds). In-order to use these designs, you will need access to basic PDK information such as the metal layer stack properties and unit tile sizes as well as the FO4 value. For easy integration of the floorplanning file into existing CAD flows, the IDF floorplanning files can be converted to DEF files using the IDF-to-DEF converter provided by the BSG IDF Tools.

Design Name	Description	# of Gates
small_comb	small sized combinational only design	~300
medium_comb	medium sized combinational only design	~3K
large_comb	large sized combinational only design	~30K
black_parrot_fe_only_2019_03_11	Front-end of a 64-bit RISC-V Core with Cache Coherence Directory	~20K
black_parrot_be_only_2019_03_11	Back-end of a 64-bit RISC-V Core with Cache Coherence Directory	~35K
black_parrot_2019_03_11	Full 64-bit RISC-V Core with Cache Coherence Directory	~125K
black_parrot_2019_03_28	Full 64-bit RISC-V Core with Cache Coherence Directory	~65K

Figure 5: Available Designs within BSG Pipeclean Suite and their Size in Approximate Number of Gates

Designs are structured in a consistent manner to make it easy for CAD flows to find all required files. The following table shows the files found within each design and what each file contains.

File	Description
Makefile	Makefile to facilitate in running macro_generate.py and IDF-to-DEF converter utility script.
constraints.FO4.sdc	SDC timing constraints which depend on the FO4 value for the target process.
floorplan.idf.json	IDF Floorplan constraints file.
harden_me.v	Additional design file that contains blocks that are intended to be hardened for correct quality-of-results (requires additional setup).
macro_generate.py	Used to call macro generators for the target process (requires additional setup).
pickled.v	Main design RTL file that has been pickled into a single .v file.

Figure 6: Design File Descriptions

These designs are dimensionless and process agnostic, therefore to properly integrate them into a CAD flow and get the expected quality-of-result (QoR) we need to re-dimensionalize the designs. Inside the IDF Floorplan file is a "harden" section which describes components in the design that are either macro blocks or pre-synthesized gate netlists. For macro generation, there is a python script inside each design called macro_generate.py. The intention of this script is to read the IDF Floorplan file and use that info to call macro generators. However, macro generators are process specific therefore the user is required to set up the script to shell out the correct commands. A small amount of framework has been implemented in macro_generate.py however it is up to the user to get the generators operational.

In-order to instantiate hardened blocks they must be instantiated properly in the RTL. This will be done inside harden_me.v. This file contains a collection of modules with synthesizable RTL models that are intended to be replaced with the hardened instance (either a macro or pre-synthesized netlist). The user should use the RTL model to ensure that the semantics of the hardened block match what the design expects.

The easiest way to integrate the IDF Floorplan into a cad flow is to first convert it into a design exchange format (DEF) file. DEF files are standardized floorplan files. By converting the IDF Floorplan to a DEF file, you will also be re-dimensionalizing the floorplan.

3.5 IDEA Dimensionless Floorplan (IDF) and IDF Tools

(https://github.com/bespoke-silicon-group/bsg_idf_tools)
(https://docs.google.com/document/d/1rIB81hEOSNs2F1pIF6WbIwSNe-8HbfEtZRlaMFKdhKk)

One challenge that the open-source hardware community faces is the fear of sharing physical design constraints that violate NDAs by revealing sensitive information about process

geometries. To address this issue, the ORDER team created the IDEA Dimensionless Format (IDF) - a simple, coherent coarse floorplan specification allowing designers to open-source physical design constraints without fear of NDA violations. This is done by converting traditional constraints into *dimensionless* units. These are PDK agnostic values that reveal no sensitive information. The IDF specification is built on-top of JSON which allows the file to be both human read-writeable as well as machine read-writeable.

One of the leading file formats for representing physical layout is the Design Exchange Format (DEF). Nearly all commercial and open-source layout CAD tools support DEF files to some degree. Therefore, to integrate IDF files into existing CAD flows we have developed the IDF-to-DEF converter script which exists within the IDF-tools repository. The DEF file resulting from this conversion is no longer dimensionless and will contain NDA sensitive information. This information comes from the PDK technology file which is an input to the IDF-to-DEF converter script.

Along with allowing physical constraints to be published publicly without fear of NDA violations, the IDF file can also be used to port physical constraints between process nodes. Since the constants are dimensionless, they are by definition process agnostic and when implemented into a CAD flow can be re-dimensionalized using a different set of dimensions than it was originally created with. This process is not perfect due to physical dimensions not being perfectly consistent across process nodes. However, with some post-processing scripting intelligence, a reasonable transfer of floorplanning constraints between process nodes can be achieved.

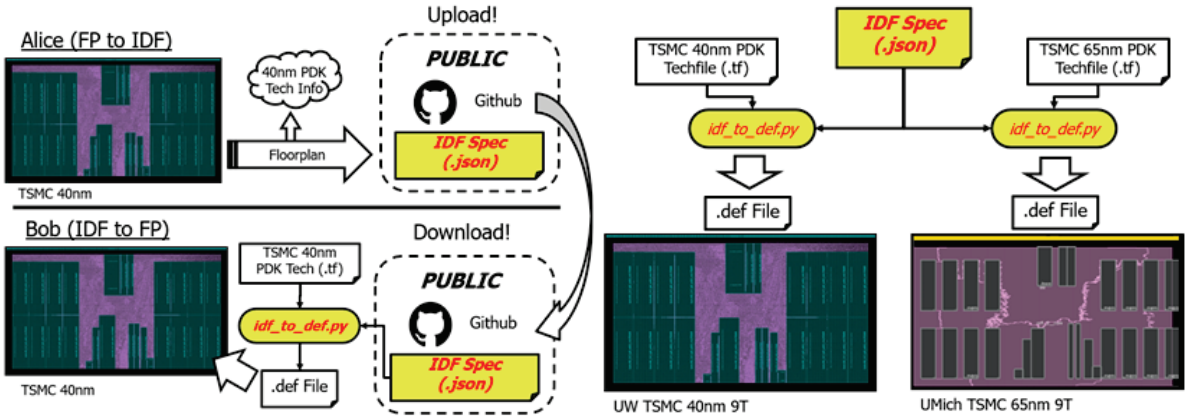


Figure 7: Two Primary use cases for IDF files to be used for Information Transfer
The first (left) is to use IDF files to publish floorplan constraints that can be used to replicate a design if the same process node and PDK is used. The second (right) is used a as an initial floorplan when porting a design from one process node to another.

3.6 BSG Fakeram SRAM Front-End Generator

(https://github.com/bespoke-silicon-group/bsg_fakeram)

One challenge open-source CAD tool developers face is the availability of SRAMs. Open-source PDKs have been available for a few years, but without SRAMs, comparing a design's PPA is

very difficult. To address this challenge, the ORDER team created BSG Fakeram - a black-boxed SRAM front-end view generator for use with CAD flows where SRAM generators do not exist or are not available. BSG Fakeram uses CACTI [2] to generate PPA models for various SRAM memory configurations. CACTI is an open-source analytical model for caches and memories. By using CACTI for PPA modeling, we avoid all NDA complications that can arise when generating memories.

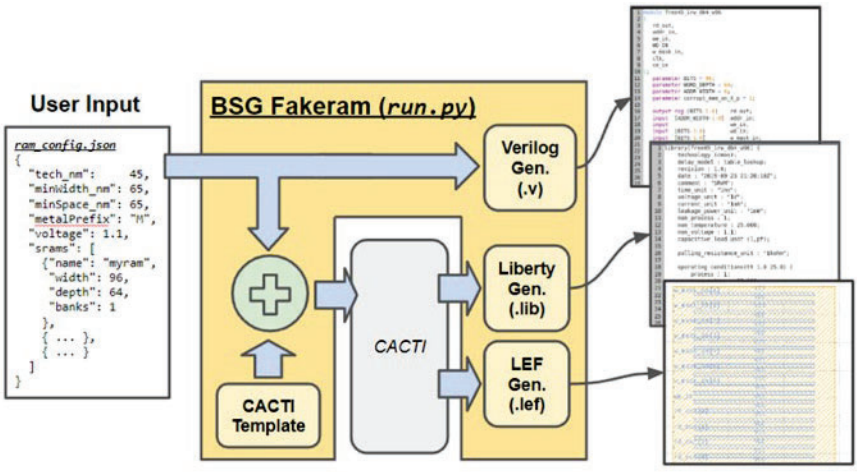


Figure 8: Flow Diagram for BSG Frakeram

The user creates a configuration file which contains basic information about the target process as well as a list of SRAM models to generate. For each SRAM model, BSG Fakeram will query CACTI for PPA information regarding the specific parameterization of the memory and then generate 3 front-end views: a behavior model (Verilog file format), and timing-power model (Liberty file format), and a abstract physical model (Library Exchange Format (LEF) file format).

Even the abstract physical view of SRAMs can vary significantly between process nodes and IP vendors. Furthermore, the metal layer used by the SRAM as well as the pin layers can also change. BSG Fakeram has several configuration options the user can set to tweak the physical view generator to best match the target process. Currently BSG Fakeram includes a configuration file template for generating SRAMs for both Nangate45 and SKY130 processes.

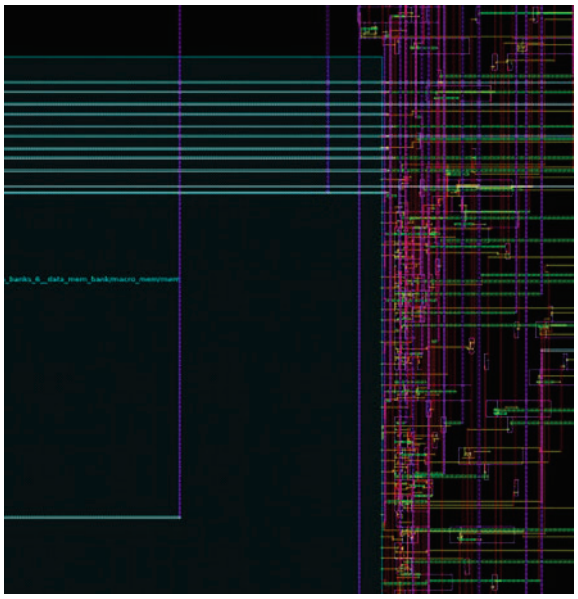


Figure 9: Image of a BSG Fakeram Macro Integrated and Routed into a Digital Design
The blue region is the generated SRAM with ports on the right side of the macro with wire connections running to the ports and over the SRAM.

3.7 BSG SV2V - SystemVerilog to Verilog Converter

(https://github.com/bespoke-silicon-group/bsg_sv2v)

A challenge that the open-source hardware community has faced for a long time is SystemVerilog support in open-source CAD tools. This has limited the number of designs available to implement using these open-source CAD tools and discouraged cutting edge hardware designers who rely on the features provided by SystemVerilog from considering open-source CAD tools as a viable option. To address this, the ORDER team created BSG SV2V - a SystemVerilog to Verilog 2005 converter that takes a collection of SystemVerilog RTL files and outputs a single Verilog RTL file that represented the same hardware but using only Verilog 2005 language features and has pickled several files into just a single file.

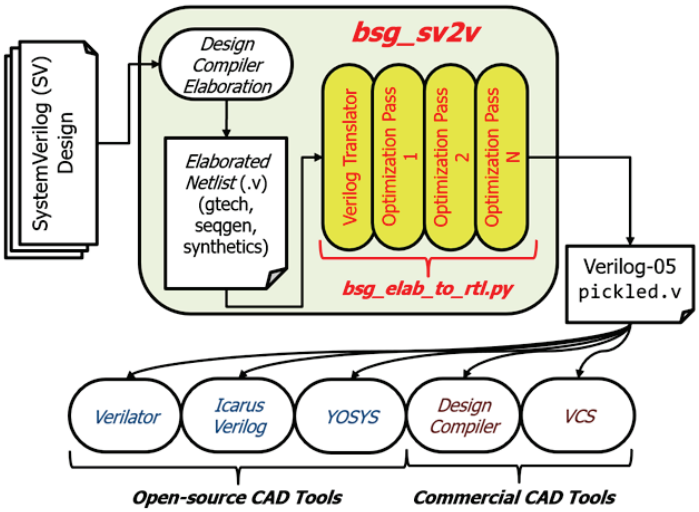


Figure 10: BSG SV2V Flow Diagram

3.8 UW OpenROAD Free45 Reference Flow

(https://github.com/bsg-idea/uw_openroad_free45)

When starting out with a commercial CAD tool, EDA vendors and even the PDK will include several reference scripts from ASIC implementations. These scripts are then modified by the designers to fix and optimize them for the specific design being implemented. This takes months, if not years before a tape-out is complete. Furthermore, since these scripts may be under NDA, everyone is replicating a lot of this work. One of the major advantages of open-source CAD tools and open-source PDKs is the ability to share these flow scripts publicly thus cutting years of development time and allowing designs to focus on optimizing design capabilities rather than flow fixes and optimizations.

The UW OpenROAD Free45 reference flow is an RTL-to-GDS flow that links several tools together to form a cohesive ASIC implementation flow. The flow uses BSG Fakeram for SRAM modeling as Free45 has no available SRAM generators. If the input design is written in SystemVerilog, BSG SV2V can be used to convert the RTL into Verilog 2005 RTL which is more compatible with the open-source CAD tool offerings. OpenROAD is used for synthesis and automatic place-and-route.

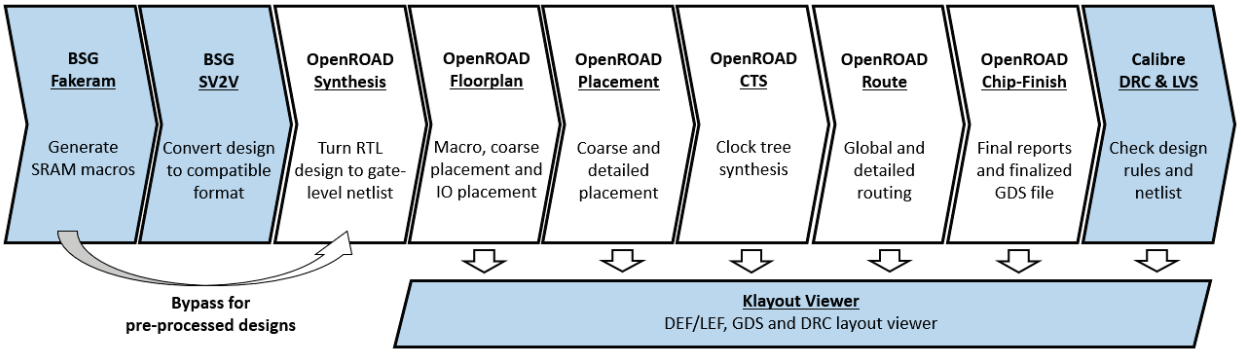


Figure 11: UW OpenROAD Free45 Steps

Steps can be run one at a time or all at once controlled via a simple Makefile.

Unlike commercial EDA tool flows which have a predefined set of steps to realize the finalized ASIC, a successful reference flow for open-source CAD tools needs to be able to adapt to new and emerging tools to best support the state-of-the-art in a rapidly changing landscape. This was taken into consideration when architecting the flow and so the flow has been built in a modular fashion with each step of the flow being "containerized" where it is run in individual directories so that all generated reports and results are easy to find and passed off to the next step. The scripting for each step has also been organized into individual directories and linked together inside the root Makefile making it easy to add, upgrade, modify and remove steps from the workflow allowing for easy exploration and experimentation with new tools and capabilities.

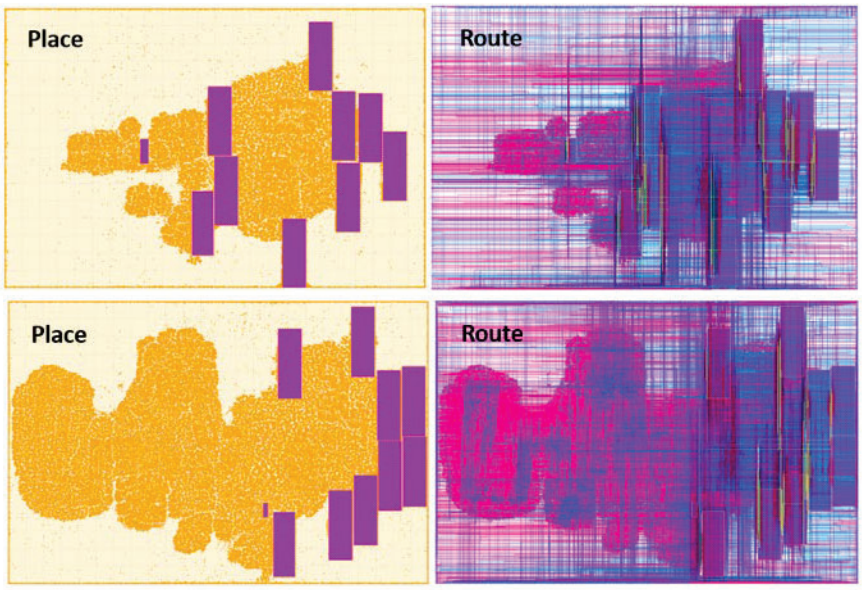


Figure 12: Example Layouts Done using the UW OpenROAD Free45 Reference Flow

The top layout is a BlackParrot Front-end design with approximately 40K gates and 10 SRAM macros. The bottom layout is a BlackParrot Back-end design with approximately 80K gates and 10 m macros. Both designs come from the BSG Pipeclean Suite and the SRAM macros were generated using BSG Fakeram.

3.9 Using IDEA tools, Providing Feedback, and Interactions

Throughout the IDEA program, the ORDER team has worked to use the IDEA tools (analog and digital) and provide insightful feedback, bug reports, and interactions with the IDEA EDA tool designer teams. While there were many interactions and feedback provided, this subsection highlights some example interactions.

The ORDER team provided designs and consultation with the OpenROAD team using a design provided by the ORDER team which will be described in more depth below (BlackParrot Tape-In Consultation). Another great example of the ORDER team's interactions include when The University of Utah reached out looking for help running simulations on the black_parrot design found in the OpenROAD-flow as a test case. The ORDER team supported the University of Utah's use of the BlackParrot design. The ORDER team generated a testing repository for OpenROAD-flow's black_parrot design and helped the LSOracle team debug post-synthesis simulation flow setup. Another notable interaction is when the Stanford POSH team requested designs to test out their ILA work (Prof. Malik's (Princeton) group (funded under Stanford DARPA POSH team)). The ORDER team provided an advanced sub-word serial SIMD design as a benchmark design for their ILA work and provided feedback on their bug finding abilities. An example analog team interaction was when the ORDER team provided the Sandia Xyce team with a test case of analog assertion. The ORDER team provided curated PCB designs to the Purdue team (Prof. Jiao's Team) for Electromagnetic Analysis.

3.9.1 BlackParrot Tape-In Consultation

https://github.com/bsg-idea/bsg_designs_openroad_one
https://github.com/bsg-idea/bsg_openroad_flow_tests

The first issues to tackle at the beginning of the OpenROAD BlackParrot tape in collaboration was simply having a set of incrementally more challenging designs to work towards to ensure the OpenROAD team was never stalled waiting for sufficiently challenging designs. We handed off 4 designs of increasing complexity. These included a design that just had BlackParrot's off-chip IO and three designs with 1, 2 and 4 BlackParrot cores respectively. These designs came from previous tapeout designs that we had experience with, however there were tool compatibility issues. To address issues with RTL compatibility, we created a bsg_sv2v (https://github.com/bespoke-silicon-group/bsg_sv2v) which could convert and pickle these designs into easy to digest pure Verilog designs. We then rewrote the IO pad ring constraints and timing constraints in formats that were compatible with OpenROAD. To facilitate in tracking the QoR of the OpenROAD tape in throughout development, we generated PPA reports for all 4 designs that the OpenROAD team could use as the main comparison point with commercial offerings for these designs (https://docs.google.com/spreadsheets/d/1a-gtwkOzAgLHSOCCAuyB1Pye7-IN3o20nnwG8jhxrkc/edit#gid=0).

Throughout the development of the OpenROAD + BlackParrot tape in, issues were brought up during weekly meetings and email communication that we assisted on resolving. Issues came in two main varieties: lower-level implementation issues and higher-level evaluation issues. Where possible we assisted with both types of issues.

For lower-level implementation issues, the OpenROAD team would send over resulting files, reports, and screenshots of the current status of design implementations. We would then analyze these files and compare them with our understanding of the design as implemented by our commercial CAD flows. From there we would advise and iterate with the OpenROAD team on solutions or potential workarounds. Some specific examples of lower-level implementation issues that we helped resolve include:

- Clock net annotations being dropped causing runtime and memory usage issues during high fanout max-capacitance design rule violation fixing. We were able to check reports and mid-implementation timing constraints and found that certain clock nets had lost their annotations.

- Automatic macro placer creating unnecessarily challenging timing paths. We analyzed the floorplan being generated by OpenROAD's automatic macro placer and found that all macros were being optimized towards a single corner of the design. By examining data paths to those macros we found that spreading the macros more evenly around the edge of the design would alleviate the rest of the place-and-route implementation.

- GPIO cell pin connection DRCs. OpenROAD's router was creating routes with DRC errors for connections specifically with the GPIO cell's core side pins. We advised on keep-out margin distances that we specify with commercial tools to avoid similar issues.

- Antenna rule fixing. OpenROAD implementation was running into several antenna rule violations. We evaluated our prior tapeout designs to discuss the possibility of using net buffering to fix antenna rules rather than implementing diode insertion.

- Seal ring insertion appeared to be causing several DRCs (particularly for latchup checks). We provided the OpenROAD team several checks to try and eliminate some trivial bugs that arise from seal ring generation and merging. This included specific distances between all of the geometries around the edge of the ASIC to manually verify that all geometry offsets were correct.

- Timing constraint adjustments and timing path evaluations. Throughout development, we had to make several adjustments to the timing constraints to accommodate the current status of OpenROAD's capabilities. The source-synchronous IO and clock-domain-crossing timing constraints were of particular importance as these are less trivial to adjust than standard system clock's and traditional IO paths. We provided guidance for how to properly adjust the constraints to fix timing issues at various stages of the tape in, as well as checking constraint changes made by the OpenROAD team directly. When timing paths failed, we discussed and implemented timing constraint changes if appropriate as well as any physical design changes that we performed on our commercial flows that might help the underlying algorithms being implemented by OpenROAD.

For high-level evaluation issues, we were made aware of specific tasks or evaluation capabilities that the OpenROAD team needed that we had knowledge and experience with. This usually involved creating custom infrastructure or writing documentation that was made available to the

OpenROAD team allowing them to focus on core OpenROAD development. Some specific examples of higher-level evaluation issues that we helped resolve include:

- Gate-level netlist evaluation. Each design that we delivered to the OpenROAD team included a testing directory with a set of tests that could be executed. This testing directory was also able to take netlists that were generated throughout the flow. Throughout the tape in, the OpenROAD team would periodically handoff netlists that we ran functional simulations on to ensure there were no logical issues with the generated netlists. To overcome x-pessimistic netlists being generated from synthesis, we ran a sweep of constrained random register initializations for each netlist. We spent a considerable amount of time working with the OpenROAD team trying to get commercial formal equivalency checkers to work on the generated netlists but were generally not able to get any strong guarantees.

- For maintenance, the OpenROAD team was using open-source PDKs such as nangate45 and asap7 for their continuous integration (CI) efforts. For these process nodes, modifications to the Black-Parrot single, dual and quad core designs were required to accommodate the lack of IO cells found in those PDKs which we made and delivered. Furthermore, the CI infrastructure was leveraging bsg_fakeram (https://github.com/bespoke-silicon-group/bsg_fakeram) for SRAM macro generation and bsg_sv2v (https://github.com/bespoke-silicon-group/bsg_sv2v) for RTL canonicalization. The OpenROAD team provided github issues and feature requests throughout the development of OpenROAD which were then fixed or added. Below are some of the specific GitHub issue links.
  - https://github.com/bespoke-silicon-group/bsg_fakeram/issues/8
  - https://github.com/bespoke-silicon-group/bsg_fakeram/issues/4
  - https://github.com/bespoke-silicon-group/bsg_fakeram/issues/3
  - https://github.com/bespoke-silicon-group/bsg_sv2v/issues/13
  - https://github.com/bespoke-silicon-group/bsg_sv2v/issues/12
  - https://github.com/bespoke-silicon-group/bsg_sv2v/issues/11

- The OpenROAD team needed to generate a seal ring for the finalized GDS and toplevel DRC testing. The generation of the seal ring was out of scope for the OpenROAD tool however the OpenROAD team still needed to generate and merge the GDS of a seal ring. Due to NDA restrictions we could not provide the seal ring so we provided detailed directions on exactly how to generate a seal ring using the GF14 PDK.

- The OpenROAD team was looking to run commercial signoff parasitic extraction (PEX) and static-timing analysis (STA) of the GF14 ASICs being generated by OpenROAD to help mitigate their standard post-place-and-route STA solution from being a limiter on the ability to sign off on a finalized design. Our commercial infrastructure was built up around proprietary file formats for the commercial tools we were using so we created a brand new commercial CAD flow infrastructure that was designed to handle file formats being generated by OpenROAD (LEF, DEF, Verilog, and SDC) and perform signoff quality parasitic extraction and static timing analysis using Synopsys StarRC and PrimeTime ADV including multi-corner and signal integrity analysis.

3.10 ParrotPiton

In order to create very large designs, the ORDER team embarked on integrating OpenPiton and BlackParrot to create a large modern manycore design. The ParrotPiton project was an extension of OpenPiton's "Bring Your Own Core" initiative (BYOC) that also extended BlackParrot's own community outreach. This initiative is designed to give hardware developers a framework by which they can leverage an existing infrastructure to cross compare designs from RTL simulations, FPGA implementations, and ASIC tape outs. The ORDER team was involved with helping the BlackParrot team integrate BlackParrot cores with OpenPiton's P-Mesh, an on-chip network that implements a high-speed, directory-based MESI coherence protocol.

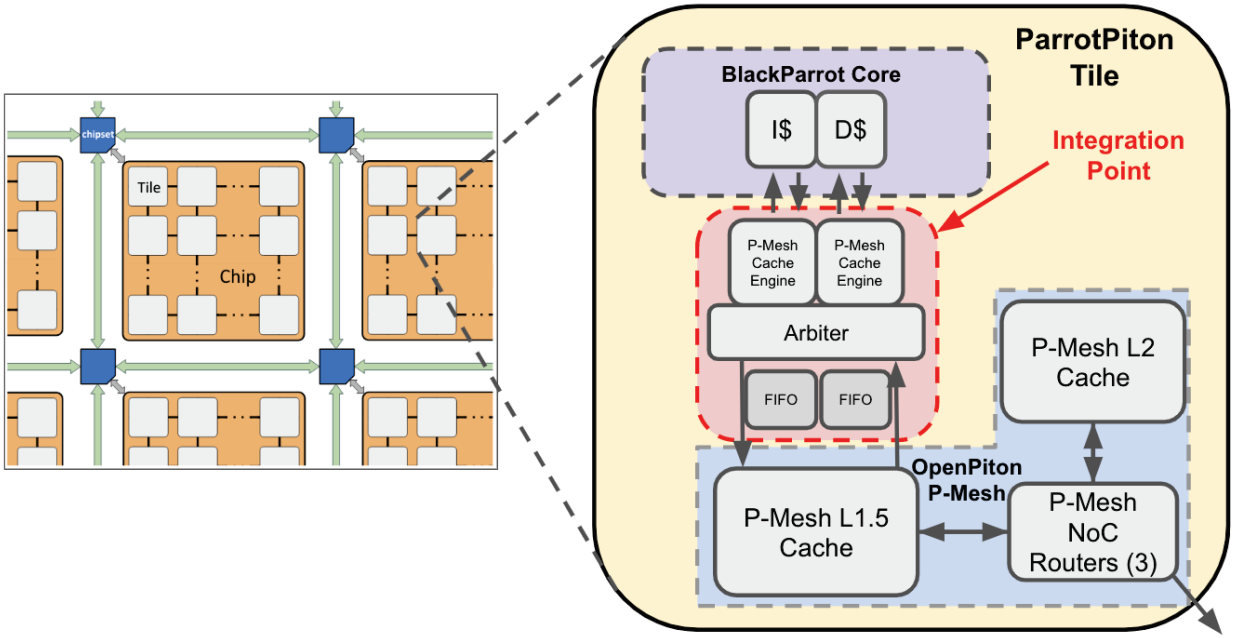


Figure 13: ParrotPiton Block Diagram

The BlackParrot core's instruction-cache and data-cache communicate with the OpenPiton P-Mesh via the transaction-response interface (TRI) via a simple integration point composed of a pair of arbitrated P-Mesh cache engines.

The primary work to integrate BlackParrot cores with the OpenPiton P-Mesh was integrating BlackParrot's instruction-cache and data-cache with the transaction-response interface (TRI) that OpenPiton uses in BYOC designs. To accomplish this, an "integration point" was designed using a pair of arbitrated P-Mesh cache engines. This integration point allows BlackParrot's and OpenPiton's cache systems to work without requiring a redesign and did not compromise the performance of the ParrotPiton system. By leveraging the OpenPiton P-Mesh, this system is scalable up to half-a-million BlackParrot cores which would likely make this the highest performance Linux-capable RISC-V manycore system.

During the integration of BlackParrot with OpenPiton's P-Mesh, several bugs were found and fixed in the BlackParrot core. This includes race conditions for non-idempotent memory operations and multi-core wakeup operations. Furthermore, the BlackParrot project got new

features to their cache system such as remote atomic support for the data-cache as well as support for both write-through and write-back cache modes.

3.11 ORDER Digital Chip Tape-out

For our digital chip, the ORDER team designed the ORDER PRGA chip which is a 512 Look-Up Table (LUT) FPGA design where each LUT takes 4 inputs (LUT4). This design utilizes a 2-level hierarchical design to test out hierarchical design in the OpenROAD design flow. The lower level design block is a Configurable Logic Block (CLB) while the upper level design connects those CLBs together to build an 8x8 array. Each CLB contains eight LUT4's and eight D Flip-Flops (DFFs).

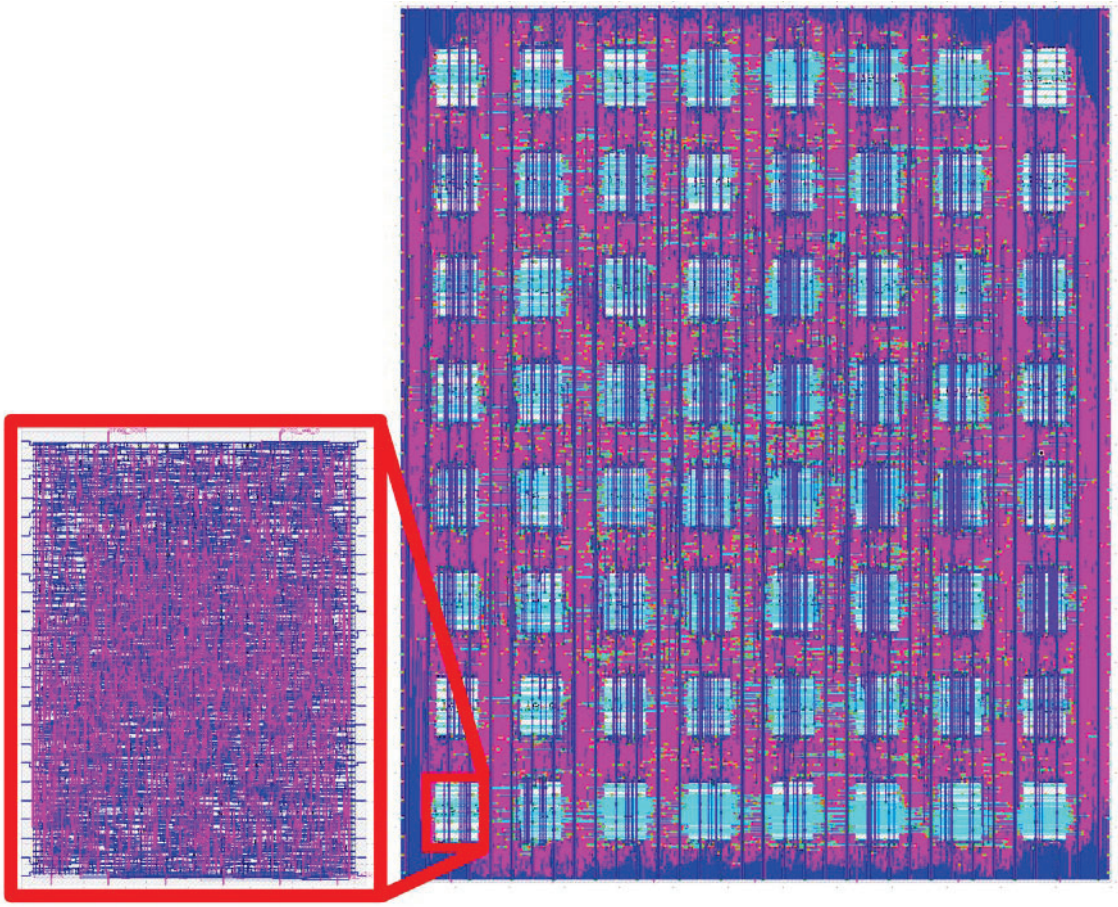


Figure 14: Layout of the FPGA Design with a Zoom-in on a Single CLB

At the CLB level, a dual-clock design was used with one clock for the configuration clock and one clock for the FPGA running clock. A false-path was set between them in the tools. In addition, combinational loops were broken with set_case_analysis during static timing analysis. A customized power-distribution network (PDN) was used inside of the CLB because the blocks were small and the default PDN did not fully cover the CLB. Last, the CLB-level of hierarchy was routed utilizing only three of the five metal layers enabling the top level to route on the top two layers.

At the top level of hierarchy, the CLB macros were manually placed in an 8x8 array. Top-level standard cells were placed in between the CLBs. A customized PDN was used at the top level, but the PDN connection was automated. The top level is able to use all five metal layers of the SkyWater 130nm process. Connecting the different CLBs, this design has a 24-track routing channel with L1 tracks. The types of circuits that can be implemented with this FPGA design were evaluated and found that it should be capable of implementing 16 out of 30 ISCAS'89 circuits.

The programming clock period is set to 1500ns period (667KHz) and the user clock was constrained to 1MHz, but this the switch speed and not the final emulated circuit's speed. The CLB size is 165.6um * 217.6um. The total fabric size is 2.6mm * 3.2mm (8x8 CLBs, each with 8 LUT4s and 8 DFFs).

The FPGA design is wrapped with the Caravel wrapper to provide I/Os for interfacing to the outside world and for bringing in configuration data and clocks. Figure 14 shows the layout of the FPGA chip with a zoom-in on the layout of a CLB. This chip was taped out in SkyWater 130nm on MPW-6 on June 8th, 2022. This chip was made with the OpenROAD tools and is open-sourced at: https://github.com/getziadz/caravel_mpw5_prga

While taping-out the FPGA, a bug was found in the automated antenna fixing algorithm used in OpenROAD/OpenLane. In particular, when diodes are inserted for route-over-macro metal tracks, the diodes may be placed within the blockage around the macro, where no power straps are available which causes issues further down the flow. A temporary solution was applied by forking the OpenROAD repo and directly modifying the C++ source file in the global router (src/grt/src/AntennaRepair.cpp). This bug was reported as a GitHub issue (https://github.com/The-OpenROAD-Project/OpenROAD/issues/1868).

3.12 BSG SSTL Chip

Traditionally, chip building startups and University research groups have struggled to get access to a wide variety of IP that one would expect to find in a high speed ASIC. One example of such IP are the IO cells for communicating off-chip. Typically, research groups and startups are stuck with relatively low performance LV-CMOS IO drivers. However, with the advent of the SKY130 open-source PDK there is an opportunity to design more complex IO drivers, such as SSTL drivers used in high-speed memory communication, and distribute them as open-source.

The goal of this chip is to test the performance of a SSTL driver circuit. The SSTL circuit was designed to meet the DDR3 specification. A signal is fed into the chip (through the pin rx_DQ below). A receiving SSTL serves as configurable termination. The signal passes through a digital buffer, and then transmitted out through a differential pair of output SSTLs.

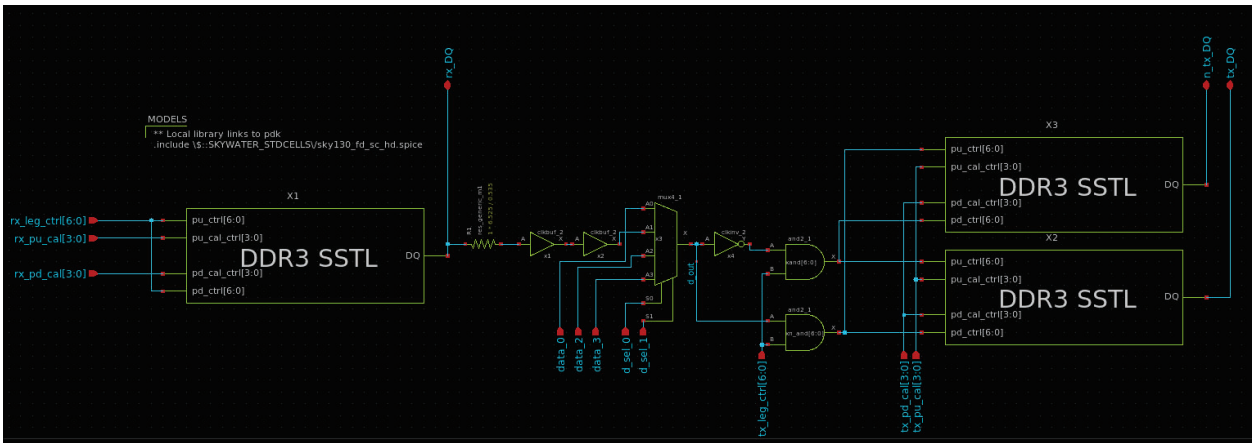


Figure 15: SSTL Test Chip Schematic

Each SSTL driver contains 7 pull-up and 7 pull-down "legs". All 14 legs are in parallel. Each leg can be considered to be a controllable resistor. When enabled, a pullup leg will connect the DQ pin through a 240 ohm resistor to VDD. The legs pull up the DQ pin with 240 ohm impedance. When disabled, a pullup leg disconnects from the DQ pin (the leg makes a high impedance connection). Similarly, when enabled, a pulldown leg will connect the DQ pin through a 240 ohm resistor to VSS. Disabling a pulldown leg disconnects it from DQ.

Enabling and disabling legs is how the driver is switched from transmitting to receiving mode.

When driving DQ, the SSTL has some number of legs enabled on one side (up or down) and none of the legs enabled on the other side. So when driving DQ high, some number of pullup legs are enabled, and none of the pulldown legs are enabled. The specific number of pullup legs enabled controls the output impedance of the signal. When driving DQ low, the only pulldown legs are enabled.

The DDR3 specification requires driving impedances to be configurable to 40 ohms and 34.29 ohms. This is achieved by the SSTL by enabling either 6 or 7 legs respectively.

When receiving, some number of legs are enabled on both the pullup and pulldown sides. This means the SSTL is actually driving DQ to the voltage VDD/2. It is expected the driving circuit on the other end has a much lower impedance so it can effectively control the voltage of DQ.

The DDR3 specification requires termination impedances to be configurable to 120 ohms, 60 ohms, and 40 ohms. This was achieved by SSTL by enabling (1 pullup and 1 pulldown), (2 pullup and 2 pulldown), and (3 pullup and 3 pulldown) legs respectively.

The additional feature an SSTL must have is the ability to fine tune the resistances of every leg to keep the resistance close to the required 240 ohms. The DDR3 spec includes a calibration procedure where the true SSTL drive strength is measured periodically. If it falls out of range, the leg resistances are adjusted. In this particular SSTL implementation, there are several calibration FETs in parallel with the main (polysilicon) resistor in each leg. In the low temperature or high voltage cases (where the leg has reduced resistance) some calibration FETs

are turned off to increase the resistance back in spec. In the high temperature or low voltage cases, more calibration FETs are turned on.

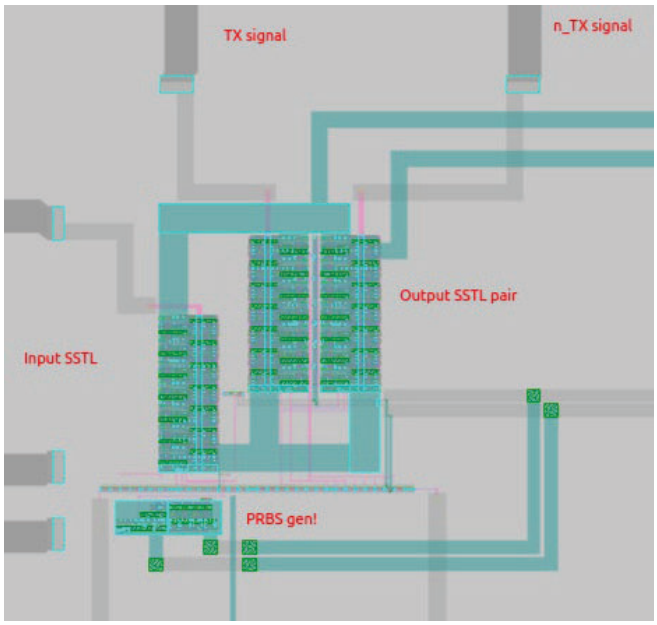


Figure 16: SSTL Test Chip Layout

3.13 TurboXAUI

Turbo-XAUI is a high speed I/O interface that supports the conventional XAUI standard 3.125Gb/s data rate and Turbo mode which runs at quadruple data rate (10Gb/s) of conventional XAUI. The Turbo-XAUI block diagram is illustrated in Figure 17. The transmitter consists of digital data serializers, encoder, pre-emphasis equalizer, pre-driver and output driver. The receiver consists of CTLE, time-interleaved ADC, CDR, and digital parts such as the decoder and the deserializer. ADC-based receiver architecture is used to enable equalization in the digital domain. The MDLL is used to generate a 1.563GHz clock signal for the transmitter and the reference clock signal for the receiver. In the standard XAUI mode, the transmitter works in the quadrature-rate whereas in the Turbo mode, it works in full-rate. The receiver, on the other hand, adopts sub-sampling CDR to achieve the Turbo mode. We extracted the Turbo-XAUI transceiver model and simulated the I/O system in MATLAB. The simulation results are shown in Figure 18. The channel loss at 5GHz is -27dB. Two key circuit blocks, ADC and CDR, are further introduced in the following sections.

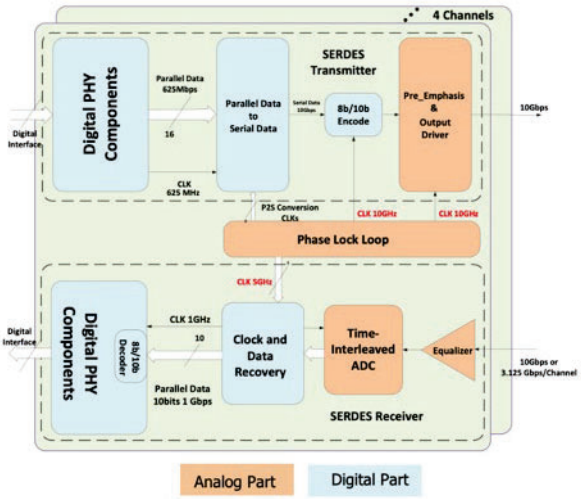


Figure 17: Turbo-XAUI block diagram

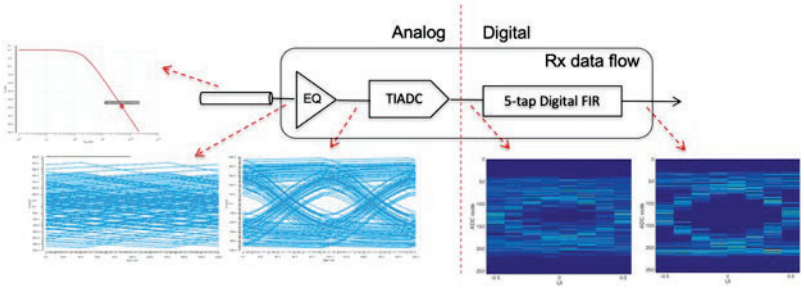


Figure 18: Turbo-XAUI simulation results

3.13.1 Non-Uniformly Quantized SAR ADC

High insertion loss channels and multi-level signaling boost the demands for ADC-based I/O link transceivers. The ADCs in wireline receivers receive input signals with non-uniform distribution mainly due to ISI and noise from the wireline channels. It naturally calls for design of non-uniformly quantized ADCs. The role of the ADCs in turn evolves from signal type conversion to signal type conversion and equalization of ISI. Previous research on non-uniformly quantized ADCs is based on flash ADCs since it is straightforward to control the threshold voltages. A non-uniform flash ADC is introduced to optimize power consumption of receivers. A 3-bit and a 1-to-6-bit non-uniform flash ADCs are proposed to minimize BER. The 3-bit non-uniform flash ADC adjusts the threshold voltages by controlling the reference voltages of comparators. The 1-to-6-bit non-uniform flash ADC shuts down the comparators inputting unwanted threshold voltages. However, SAR ADCs are widely used for mid-resolution (4-8 bits) applications including transceivers and usually outperform flash ADCs. Therefore, we propose to use non-uniform SAR ADCs (NUSA) for transceivers.

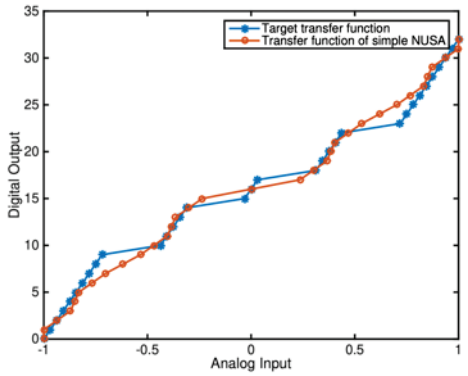


Figure 19: Transfer Function of a 5-bit Simple NUSA, i.e., NUSA-CIW

One simple approach to design NUSAs is to control the transfer function by using non-binary weighted CDAC. To distinguish it from NUSA-CDWs, we name it as non-uniform SAR ADC with code-independent weights (NUSA-CIW). However, such NUSA-CIWs generate a limited amount of transfer functions. An N-bit NUSA-CIW has N-1 switching capacitors and 2^N-2 non-zero threshold voltages. Each threshold voltage is a linear function of N-1 capacitor weights. There are 2^N-2 linear equations and N-1 variables, which is naturally an ill-defined problem. When 2^N-2 > N-1, there is a high chance that there is no solution for the equations. Therefore, many transfer functions cannot be realized by this naive NUSA. The percentage of the transfer functions that can be realized by this NUSA further reduces when the resolution of the ADC increases. Figure 19 shows an example of a 5-bit NUSA-CIW. In this example, the NUSA-CIW cannot generate the target transfer function due to the above reasons.

A configurable NUSA with code-dependent weight (CDW) is proposed to realize non-linear transfer functions. The NUSA-CDWs acquire flexibility for transfer functions by allowing different capacitor weights for different digital output codes. There are three main advantages of NUSA-CDWs. NUSA-CDWs improve BER when used in ADC-based receivers compared to conventional SAR ADCs. NUSA-CDWs are less power-hungry than non-uniform flash ADCs for mid-to-high resolutions since they are based on SAR ADC architecture. NUSA-CDWs achieve a much wider range of transfer functions than NUSA-CIWs.

3.13.2 NUSA with Code-Dependent Weight

To overcome the intrinsic limit of SAR ADCs and generate non-uniform transfer functions, we introduce a NUSA with code-dependent weight. At each conversion step, a NUSA-CDW not only dictates how to switch the capacitor arrays but also dictates the weight of the switching capacitor based on the comparison result. This allows distinct threshold steps and removes correlation between threshold voltages. Figure 20 shows threshold voltages of a 4-bit NUSA-CDW. We define a threshold voltage path as a set of threshold voltages traversed to output a digital code and a path code is the corresponding output digital code.

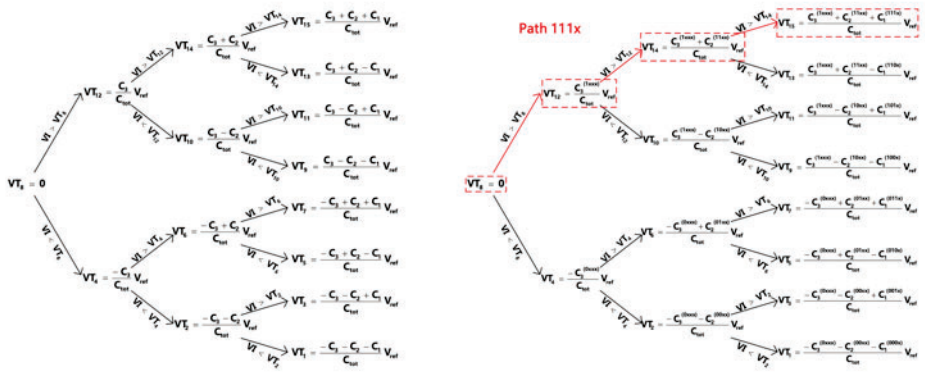


Figure 20: Threshold Voltage Trees of 4-bit ADCs

Figure 21 shows the transfer function of a 3-bit NUSA-CDW and four 3-bit NUSA-CIWs. The capacitor weights of each NUSA-CIW are identical to each path of the NUSA-CDW. In other words, the equivalent transfer function of each path. The transfer function of the NUSA-CDW aligns with the transfer function of each NUSA-CIW when the digital output equals the path code. When the capacitor weight sets of all the paths are identical, a NUSA-CDW degenerates to a NUSA-CIW.

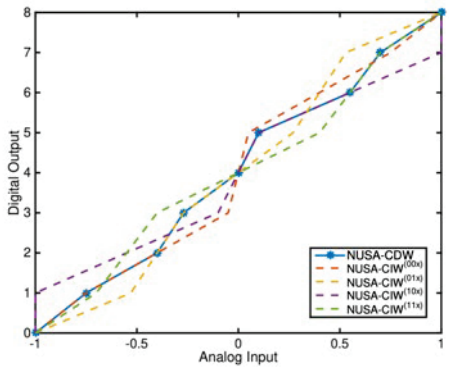


Figure 21: Transfer Functions of a 3-bit NUSA-CDW and 3-bit NUSA-CIWs
The capacitor weights of each NUSA-CIW is identical to each path of the NUSA-CDW.

A configurable 5-bit NUSA-CDW circuit is evaluated (simulation) in 65nm technology. The NUSA-CDW circuit is composed of sample-and-hold circuit, a differential capacitive DAC, 5 comparators, and the NUSA digital logic as illustrated in Figure 22.

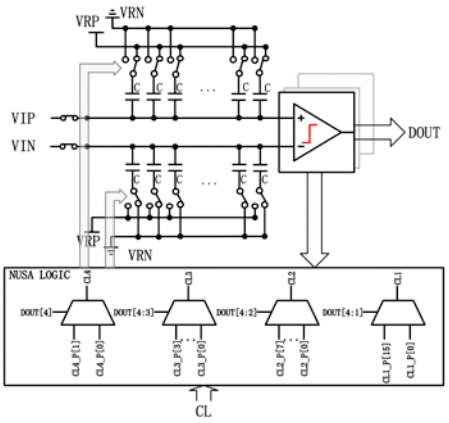


Figure 22: Circuit Diagram of a 5-bit NUSA-CDW

We use a bootstrapped switch to implement the sample-and-hold block. The CDAC consists of binary weighted capacitors. It employs set-and-down switching to improve the speed of the ADC yet maintain a constant common-mode voltage. We adopt one comparator for each conversion step to get rid of the SAR logic and save feedback delay. The NUSA logic mainly composed of multiplexers to select capacitor weights. The inputs of the NUSA logic are 16 sets of the control signals for the capacitive DAC. Each set of the control signals corresponds to one threshold voltage. The transfer function can be easily configured by modifying the control signals.

The NUSA-CDW circuit works as follows. First, we configure the control signals based on the desired transfer function. Then we start to run the ADC circuit. The sample-and-hold circuit captures the input analog signal to the top plate of the capacitive DAC. The first comparator compares the output of the capacitive DAC and generates the first bit of the digital output. Based on this comparison result, the NUSA logic selects one set of the control signals. The selected control signals then switch the capacitor arrays to obtain a new threshold voltage for next conversion step. After five conversion steps, the NUSA-CDW circuit outputs a 5-bit digital code.

Performance Summary

Architecture	NUSA-CDW	Non-uniform flash
Technology	65 nm	90 nm
Resolution	5 b	3 b
Supply voltage	1.2 V	1.2 V
Sampling frequency	512 MS/s	4 GS/s
Differential input range	1.8 V	0.1 V
SNDR@near $F_{Nyquist}$	30.88 dB	
ENOB	4.83 bits	1.4 bits
Power consumption	600.59 μW	
FoM	36.6 fJ/conv	1.42 pJ/conv

Figure 23: Performance Summary of NUSA-CDW

Figure 23 shows the simulation results of the 5-bit NUSA-CDW circuit. The 5-bit NUSA-CDW circuit is simulated at 512MS/s sampling rate. The full-scale input range of the ADC is 1.8 V and the power supply is 1.2 V. We first configured it as a uniform ADC. SNDR and ENOB of the ADC circuit are 30.88dB and 4.83 bits at 244MS/s input frequency. Then we configured it with a non-uniform ADC. The transfer function of the NUSA-CDW circuit is plotted in Figure 24. The

power consumption is 600.59 uW and the figure of merit is 36.6 fJ/conv. We also simulated the NUSA-CDW circuit within a wireline transceiver. The channel response of h[n]=[0.0788, 0.2627, 0.1576, 0.1051, 0.0525, 0.002]. BERs of the receiver using the NUSA-CDW circuit at various SNRs are depicted in Figure 24. When SNR of the channel is 40 dB, the NUSA-CDW improves BER from 4.6e-2 to 1.3e-6 compared to the conventional SAR ADC. BER enhancement of the NUSA-CDW increases with SNR.

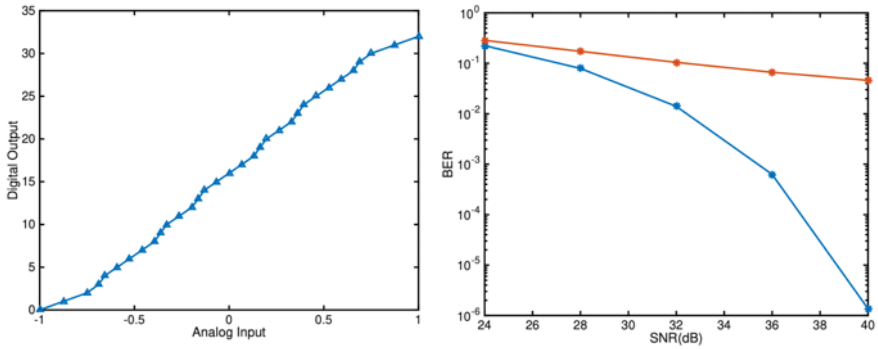
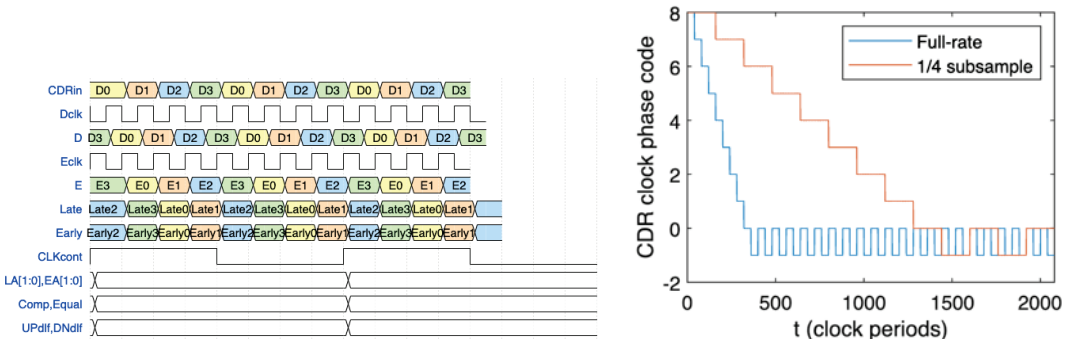


Figure 24: Transfer Function and BER vs. SNR of the 5-bit NUSA-CDW Circuit and an Ideal 5-bit Uniform ADC

3.13.3 Sub-Sampling CDR - PI Controller

The PI controller consists of an early/late (E/L) detector, a majority vote logic (MVL), a digital loop filter (DLF), a 2-1 MUX, and an octant phase controller. The timing diagram of the PI controller is shown in Figure 25 (a). The sampler outputs, D and E, are used as the E/L detector's input. The Early and Late signals are produced by the E/L detector by comparing the output of nearby samplers. By majority vote, the MVL decides whether the sample clocks' phases are ahead of or behind the incoming data stream. The EA[1:0] and LA[1:0] signals are received by the digital loop filter (DLF), which is made up of an encoder and a variable finite-state machine (FSM). It produces the UPDLF/DNDLF signals, which allow the octant phase controller's output signals to be modified. The encoder compares the EA[1:0] and LA[1:0] to produce the Comp signal and the Equal signal. The output Comp signal goes high if EA has more digits than LA, and low if LA has fewer digits than EA has. When E and L both contain equal number of ones, an Equal signal is produced, and the FSM keeps the prior value. The phase selector and the phase interpolator are controlled by the octant phase controller, which is used in conjunction with an up/down counter to generate the control codes. A fully customized design was employed to create the proposed PI controller in order to operate at high speed. In XAUI mode, the PI controller works in full-rate whereas in turbo mode, it adapts to a quarter-rate sub-sampling CDR. The PI controller output codes are shown in Figure 25 (b).

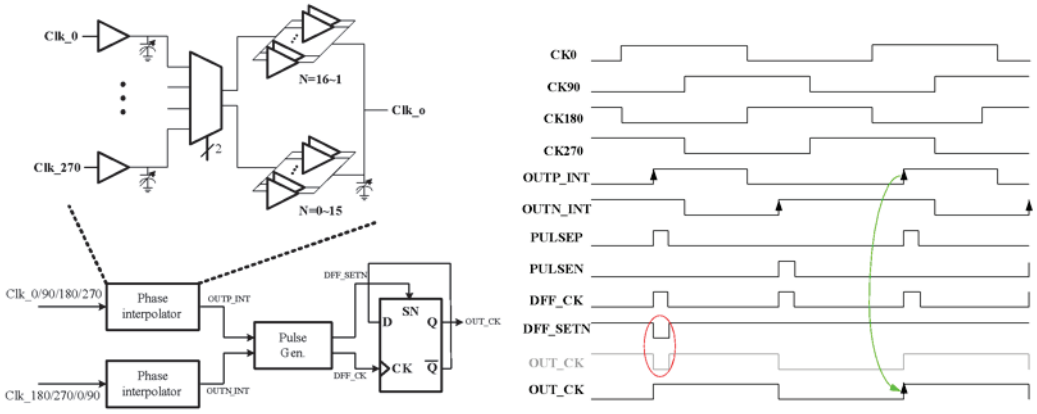


(a) (b)

Figure 25: (a) Timing Diagram of PI Controller, (b) PI Controller Output Code in XAUI Mode and Turbo-XAUI Mode

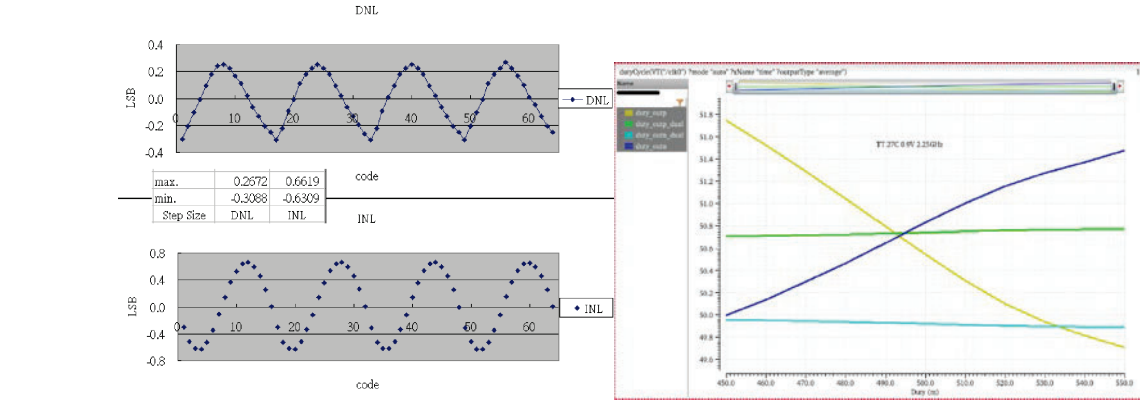
3.13.4 Sub-Sampling CDR - PI

We adopt complementary phase interpolation cells to support the global de-skew as shown in Figure 26(a), is intended to be provided. A clock with a customizable phase and a resolution of 1/64 clock periods can be output by a 6b PI cell. Figure 26(b) displays its timing diagram to ensure proper output phase following division by 2. In order to shape appropriate clock waveforms for linearity concerns, variable capacitance is set at the output nodes of the input clock buffer and the interpolation buffer. Figure 27(a) displays the complementary PI cell's linearity simulation result for the TT corner scenario for both INL and DNL. It is possible to attain DNL of 0.3 LSB and INL of 0.66 LSB. Additionally, two complimentary PI cells with inverted input clocks are employed to ensure an output clock with 50% duty cycle. The results of the duty cycle simulation for various input clock duties are shown in Figure 27(b). We also compare the results with a single PI cell. It is obvious that complementary PI cells maintain constant output clock duty in comparison to single PI cells, although at the expense of somewhat higher power consumption (2mA @ 2.133GHz frequency).



(a) (b)

Figure 26: (a) Complementary Phase Interpolation Cell and (b) Timing Diagram



(a) (b)

Figure 27: Simulation Result of Complementary PI Cell (a) DNL/INL and (b) Output Duty vs. Input Duty

3.13.5 Sub-Sampling CDR - Encoder

Subsampling CDRs relieves circuits from high-speed clock rate and enables digitization of CDR circuits. However, subsampling CDRs suffer from long consecutive identical digits~(CIDs). Conventional CDRs address the long CIDs by using line encoding e.g., 8b/10b encoding. However, because the subsampling CDRs subsample the received data, the bit sequence is different from the outputs of the conventional encoders. Therefore, the conventional 8b/10b encoding cannot solve the CIDs for the subsampling CDRs. A simple interleaved 8b/10b encoding can reduce CIDs. However, it induces DC-balance problem due to interleaving property. We propose a modified 8b/10b encoding dedicated to quarter-rate subsampling CDRs. The proposed encoding has an interleaved architecture and a new code mapping table. The proposed 8b/10b encoding bounds the CIDs to no longer than 5 bits, which is equal to the conventional 8b/10b encoding. The proposed encoding minimizes the code-wise disparity and running disparity to maintain DC-balance. The encoding procedure has four steps: deserialize, select code candidates, select the final code, serialize. These steps are described below.

Deserialize: First, deserialize the input bit stream [A0, B0, C0, D0, ...A7, B7, C7, D7] into four parallel channels [A0, A1, ...A7], [B0, B1, ...B7], [C0, C1, ...C7], [D0, D1, ...D7]. This step is to reorder the bit sequence as the sequence that the subsampling CDR detects. The actual data detected by the subsampling CDR is one of the four parallel channels.

Select Code Candidates: After deserialization, each parallel channel is encoded from an 8-bit code [A0, A1, ...A7] to two 10-bit code candidates [a0', a1', ...a9'] and [a0'', a1'', ...a9''] based on a look-up table. The encoding strategy is to achieve maximum CIDs of 5 bits for each channel. The two code candidates are bit-wisely complementary to have opposite disparity.

Select the Final Code: The next step is to select one code [a0, a1, ...a9] out of the two code candidates [a0', a1', ...a9'] and [a0'', a1'', ...a9'']. The selecting rule is to minimize the disparity of 20 bits after serialization. There are always two bit-wisely complementary code words C{1:40} that have minimal absolute disparity. In order to keep dc-balance, choose the code word that reduces the absolute running disparity.

Serialize: Finally, serialize the four channels of encoded bit streams [a0, a1, ...a9], [b0, b1, ...b9], [c0, c1, ...c9], [d0, d1, ...d9] into one lane [a0, b0, c0, d0, ...a9, b9, c9, d9].

4 DISCUSSION

4.1 Impact and Workforce Development

Overall, the IDEA program has created many exciting tools for designing future computer chips and the ORDER team has curated and built multiple benchmarks to help test these tools. An active community has sprung up around the IDEA open-source EDA tools and this community is expected to grow and stay vibrant. The Integration Exercises which were central to the interactions between teams in the IDEA program proved to be a great idea. In particular, the ORDER team made many great connections at these events and the fact that they were not simply short events or primarily events with a few presentations, but rather engineering-heavy events where engineers from different IDEA and POSH teams could work together and try to solve problems, integrate designs, and code/engineer together in real time. In addition, because the integration exercises lasted for multiple days, it provided the teams time to have deeper interactions than a single day event would have allowed. The IDEA team recommends using Integration Exercises for future DARPA MTO programs. In addition, as a University team, the ORDER team brought graduate student engineers and postdocs to the Integration Exercises. The ORDER students enjoyed the Integration Exercises and had opportunities to increase their professional networks by meeting and working with graduate students from other universities which they would otherwise have been unlikely to experience in their graduate studies. These Integration Exercises and interactions have contributed to workforce development and helped contribute to building and educating an active workforce of new engineers working on EDA tools and semiconductor-related fields. Beyond the Integration Exercises, the ORDER team believes that the overall IDEA and POSH programs and created tools and designs have contributed to workforce development and have increased the excitement around building semiconductor chips and tools to build chips. The excitement around designing chips in the US and workforce development has even trickled down to the undergraduate level. For instance, in our own institutions, we have even seen undergraduate students designing open-source chips using IDEA-funded tooling.

4.2 Challenges

One of the major challenges encountered during the IDEA program by the ORDER team is that of the outbreak of the global COVID-19 pandemic during the middle of the program. This had multiple impacts on the ORDER team. First, the physical labs at Princeton University and the University of Washington were closed for multiple months due to institutional and state orders and the respective state-level work-from-home orders had a negative impact on the productivity and spending of the team. Second, the COVID-19 pandemic made it more difficult to interact with other IDEA team performers who had their own delays and the integration exercises during the pandemic were made virtual which was less productive than in-person integration exercises. Also, the respective travel restrictions by the universities and state made in-person interaction and traveling to in-person events not possible during parts of the pandemic.

The second major challenge faced by the ORDER team is that DARPA at the end of Phase 1 cut (did not fund) teams that were developing PCB (boards), chip package, and SiP design tools and instead focused on funding teams that are developing chip tools (e.g., OpenROAD). By cutting the funding to teams working on these tools, it was not possible for the ORDER team to evaluate

these tools and use these tools to create PCB, chip packages, and SiP as these tools were not fully developed, did not exist, and/or were no longer part of the program. Because of this, further progress was not possible in creating, taping out, and testing open PCB, chip packages, and SiP using the IDEA tools. To help mitigate this challenge and in the interests of supporting open source tools that support package and PCB design, the ORDER team has curated some open source chip package and PCB designs. Also, we have focused more on using the chip-level tools (e.g., OpenROAD) which Phase 2 has focused on.

The third challenge faced by the ORDER team is that the availability (timing) of IDEA EDA tools was delayed relative to the original program plan. Also, the OpenROAD tools have been most well tested on older design nodes than advanced design nodes. These delays delayed the tape-out and subsequent testing of chips created by the ORDER team. To mitigate these challenges, the ORDER team taped-out two chips on the SkyWater 130nm process instead of the originally planned processes (14nm & 7nm). As SkyWater is a larger feature size process than originally planned and the base multi-project-wafer runs on SkyWater 130nm are a fixed size, the designs taped-out as part of this program were significantly smaller than originally planned. The positives of the SkyWater 130nm PDK is that it is fully open source (which has advantages over other processes), enables easy technology transfer, is well supported by OpenROAD, and the full GDSII of our designs are open source which would likely would not have been possible with more advanced nodes.

A technical challenge encountered by multiple teams throughout this program is that sharing information in advanced nodes is challenging due to non-disclosure agreements (NDAs) on the part of the foundries. In order to address and mitigate this issue, the ORDER team proactively created a set of tools and formats that are process independent and dimensionless in order to share relevant end design information and constraints without disclosing proprietary information.

5 CONCLUSIONS

In conclusion, we believe that the DARPA supported IDEA program will have a significant positive impact on the semiconductor and EDA tool community. The ORDER team has provided requirements for the IDEA EDA tool designers, provided early user feedback, provided multiple example designs and benchmarks leveraging past designs, used the IDEA developed tools, provided feedback and bug reports for the IDEA tools, and taped-out two chips using the IDEA tools. The IDEA tools, the benchmarks and feedback the ORDER team has provided will hopefully serve as a foundation for decreasing the time and cost of creating future DoD chips as well as have a positive impact on the semiconductor industry.

We look forward to seeing the tools, benchmarks, designs, and ideas created in the IDEA program having a large practical impact. Even within our own groups we are planning on continuing to use the tools and infrastructures for future research.

6 REFERENCES

[1] G. Tziantzioulis, T.-J. Chang, J. Balkind, J. Tu, F. Gao and D. Wentzlaff, "OPDB: A Scalable and Modular Design Benchmark," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 41, no. 6, pp. 1878-1887, June 2022.

[2] N. Muralimanohar, R. Balasubramonian and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, 2007.

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

ACRONYM	DESCRIPTION
ADC	Analog-to-Digital Converter
AFRL	Air Force Research Laboratory
ASIC	Application Specific Integrated Circuit
BER	Bit Error Rate
BSG	Bespoke Silicon Group
BYOC	Bring Your Own Core
CAD	Computer Aided Design
CDR	Clock Data Recovery
CDW	Code Dependent Weight
CI	Continuous Integration
CID	Consecutive Identical Digits
CLB	Combinational Logic Block
CTLE	Continuous Time Linear Equalizer
DARPA	Defense Advanced Research Agency
DDR	Double Data Rate
DLF	Digital Loop Filter
E/L	Early Late
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
GDS	Graphic Design System
IDEA	Intelligent Design of Electronic Assets
IEEE	Institute of Electrical and Electronics Engineers
ILA	Instruction-Level Abstraction
IO	Input Output
LSB	Least Significant Bit
LUT	Look Up Table
MDLL	Multiplying Delay Lock Loop
MTO	Microsystems Technology Office
MVL	Majority Vote Logic
NDA	Non-Disclosure Agreement
PCB	Printed Circuit Board
PDK	Physical Design Kit
PDN	Power Distribution Network
PI	Phase Interpolator
POSH	Posh Open Source Hardware
PPA	Power Performance Area
RTL	Register Transfer Language
SAR	Successive Approximation Register
SiP	System In Package
SNR	Signal to Noise Ratio
SSTL	Stub Series Terminated Logic
STA	Static Timing Analysis
TT	Typical-Typical