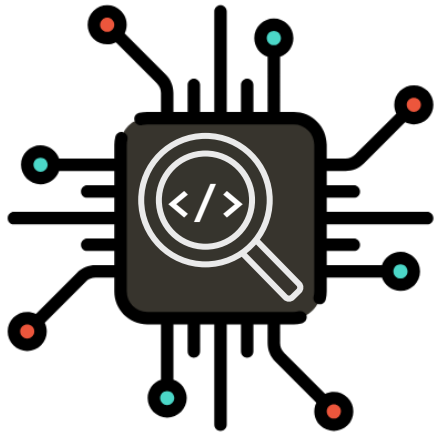


ChiSA: Static Analysis for Lightweight Chisel Verification

Jiacai Cui, Qinlin Chen, Zhongsheng Zhan, Tian Tan*, and Yue Li*

PASCAL

NANJING UNIVERSITY

POPL 2026

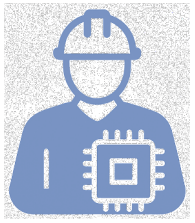
le Couvent des Jacobins

Chisel

DAC'12, 1.5K+ Citations

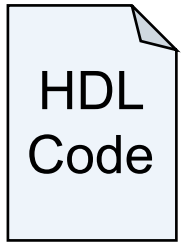
(Constructing Hardware in a Scala Embedded Language)

- A novel **hardware description language (HDL)** that enables agile chip development by leveraging modern PL features for productive design.

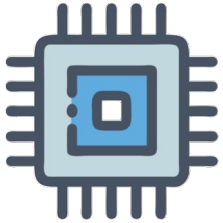


Hardware Engineers

Write

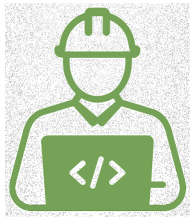


Synthesized



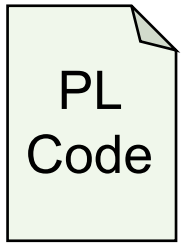
Netlist & Layout

Manufactured



Software Engineers

Write



Compiled

Executable

Run

Chisel

DAC'12, 1.5K+ Citations

(Constructing Hardware in a Scala Embedded Language)

- A novel **hardware description language (HDL)** that enables **agile chip development** by leveraging modern PL features for productive design.

> *"**Agile Chip Development**: ... Small teams should be able to design chips, tailored for a specific domain or application. This will require that hardware design become much **more efficient**, and **more like modern software** design."*
>
> — John Hennessy & David Patterson
>
> Lecture for 2017 Turing Award

Chisel

DAC'12, 1.5K+ Citations

(Constructing Hardware in a Scala Embedded Language)

- A novel **hardware description language (HDL)** that enables **agile chip development** by leveraging modern PL features for **productive design**.

> *"**Agile Chip Development**: ... Small teams should be able to design chips, tailored for a specific domain or application. This will require that hardware design become much **more efficient**, and **more like modern software design**."*
>
> — John Hennessy & David Patterson
>
> Lecture for 2017 Turing Award

Academic Practice (Open-Source)

(Rocket, Boom, etc.)

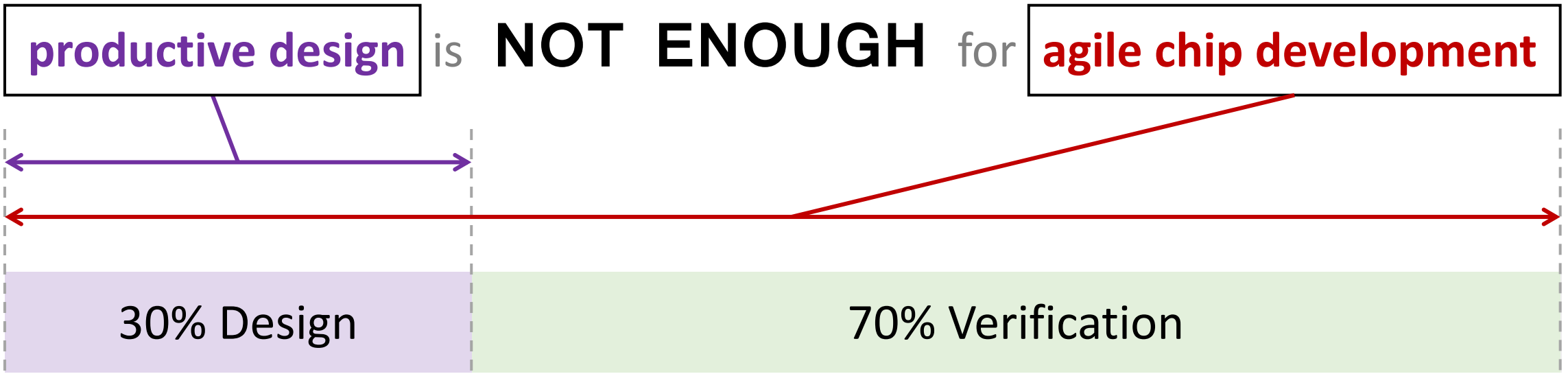
... ...

Industrial Practice (Commercial)

... ...

productive design is **NOT ENOUGH** for **agile chip development**

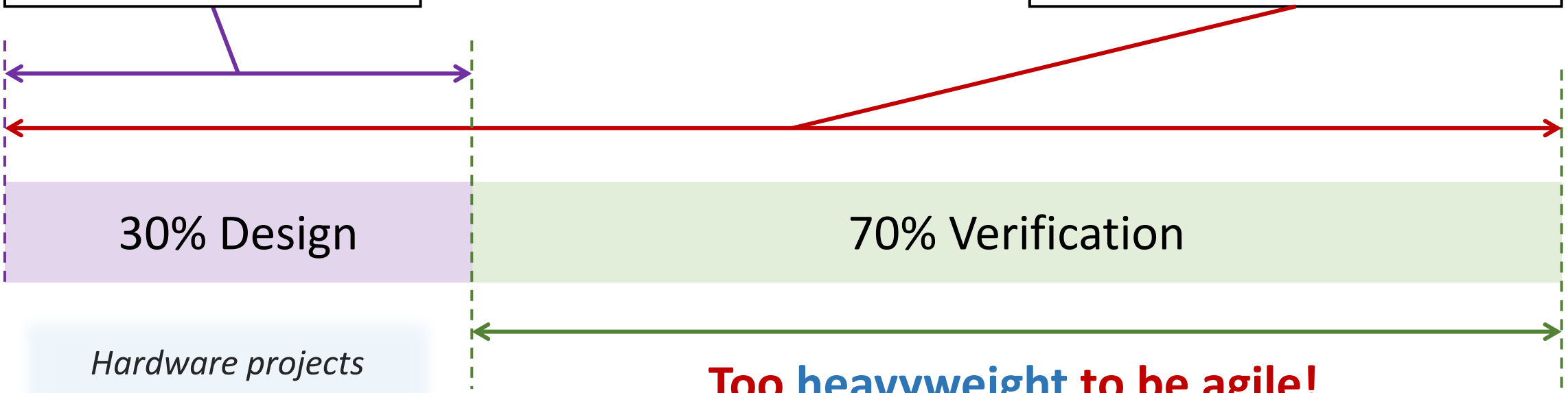
productive design is **NOT ENOUGH** for **agile chip development**



*Hardware projects often employ **more verification engineers** **than design engineers**, and even require **designers** to devote nearly **half of their time to verification** tasks.*

— according to [Siemens EDA's 2024 Global Industrial Study]

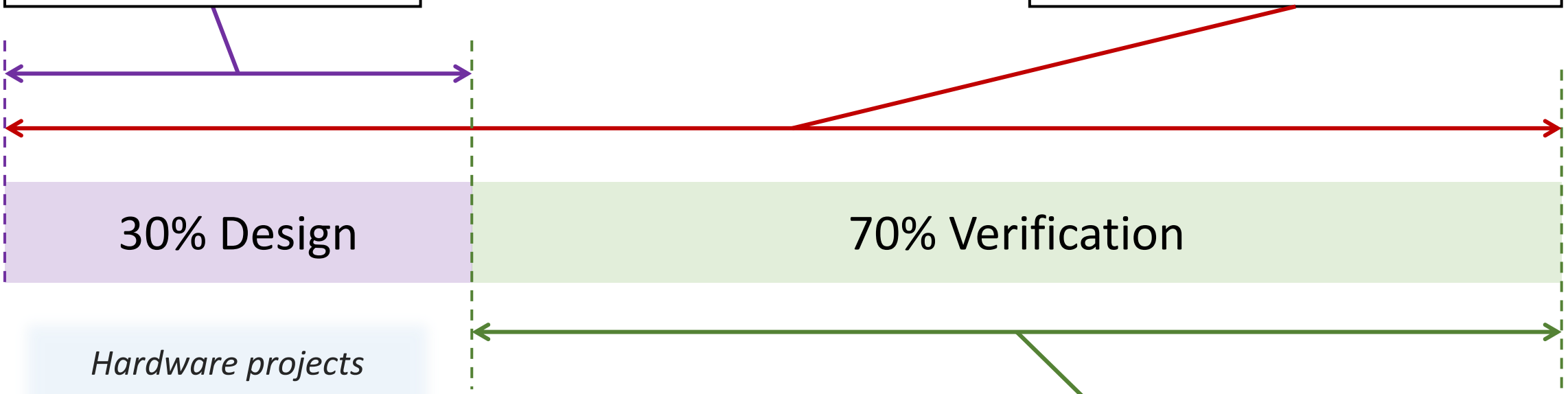
productive design is **NOT ENOUGH** for **agile chip development**



Hardware projects often employ ***more verification engineers than design engineers****, and even require* ***designers*** *to devote nearly* ***half of their time to verification*** *tasks.*

Too heavyweight to be agile!

productive design is **NOT ENOUGH** for **agile chip development**



*Hardware projects often employ **more** **verification engineers** **than** **design engineers**, and even require **designers** to devote nearly **half of their time** **to** **verification** tasks.*

lightweight verification

POPL'26 ChiSA: Static Analysis for Lightweight Chisel Verification

ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

time

Heavyweight

manual effort

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

ChiSA: Static Analysis for Lightweight Chisel Verification

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

time

Heavyweight

manual effort

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

ChiSA: Static Analysis for Lightweight Chisel Verification

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

time

Heavyweight

manual effort

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

ChiSA: Static Analysis for Lightweight Chisel Verification

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

time

Heavyweight

manual effort

ChiSA: Static Analysis for Lightweight Chisel Verification

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

time

Heavyweight

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

manual effort

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

ChiSA: Static Analysis for Lightweight Chisel Verification

static analysis is more **efficient**

time

Lightweight

million-line codebases

Heavyweight

manual effort

static analysis is more **automated**

Existing Work

Simulation

hours of simulation for seconds of chip behavior

DAC'20 ISCA'16 ...

Bounded Model Checking

well-known state explosion problem

DATE'22 WOSET'21 ...

Theorem Proving

labor-intensive construction of formal proofs

DAC'24 ...

Secure Type Systems

prohibitive annotation burden

HASP'19 CCS'18 ...

ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

3 Applications

Chisel Bug Detection
(RQ1: ChiSA vs. Bounded Model Checking)

Chisel Security Analysis
(RQ2: ChiSA vs. Secure Type Systems)

ChiSA
chisel static analyzer

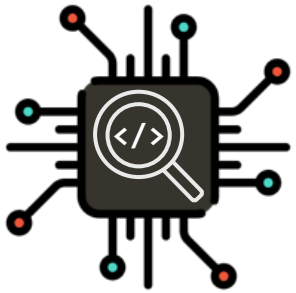
HVFAs
Framework and Instances

Chisel Analysis Infrastructures
(Reusable: front-end, IR, manager, etc.)

Proof of Concept Implementation 2

---

1 Theoretical Foundation



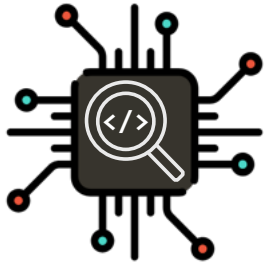
λ_C
the essence of Chisel

circuit structure	circuit behavior	circuit characteristics
λ_C syntax	λ_C semantics	λ_C properties

HVFA
hardware value flow analysis

Inspired by Software	Customized for Hardware
mathematical roots (lattice and fixed-point theory)	hardware-specificity (synchronous, clock-driven, etc)

Properties about: Soundness | Precision | Efficiency



ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

Evaluation Summary

Applications

Proof of Concept Implementation

Hardware-Specific Intuition

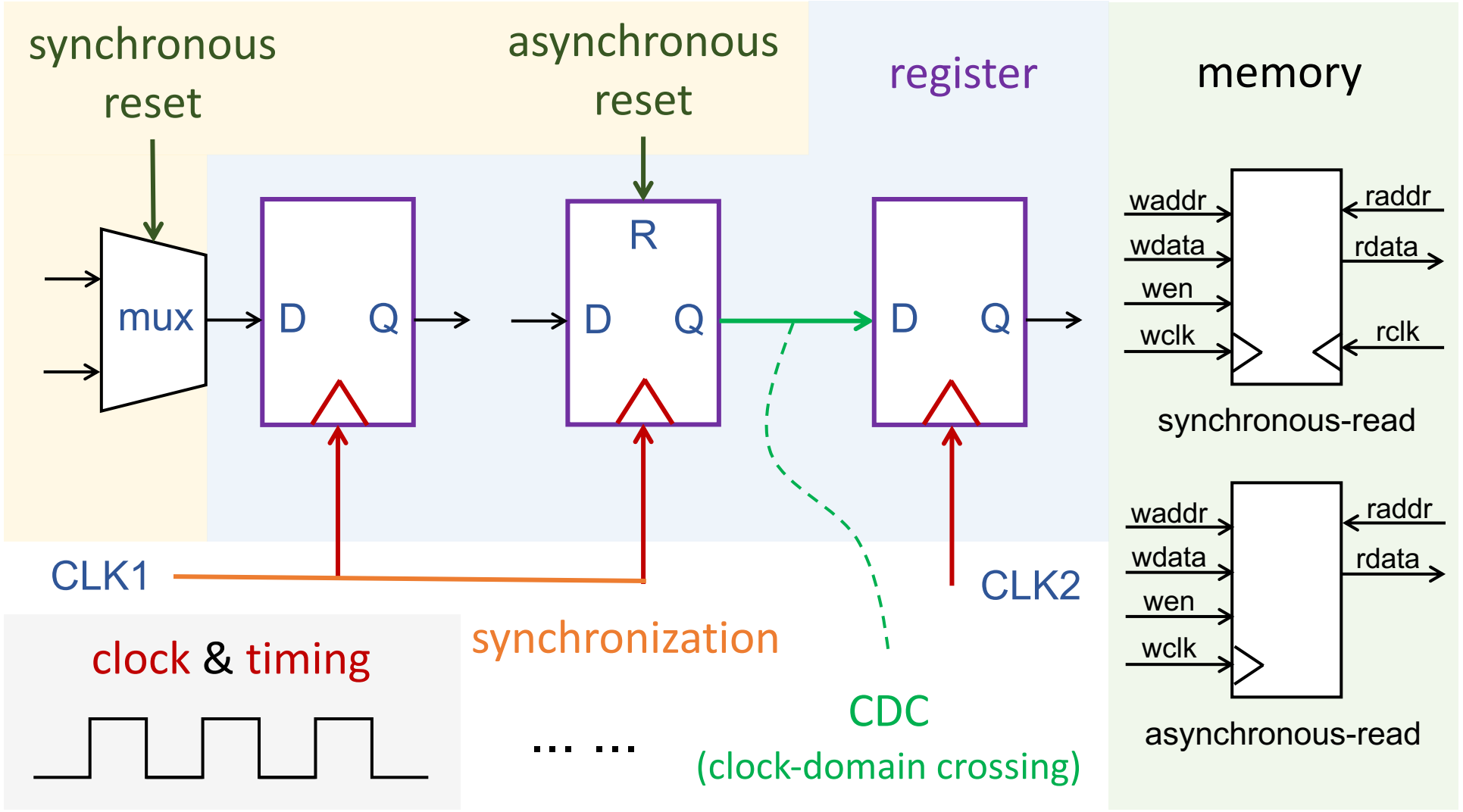
Theoretical Foundation

Brief Overview

What is special about hardware programs?

Hardware-Specific Intuition

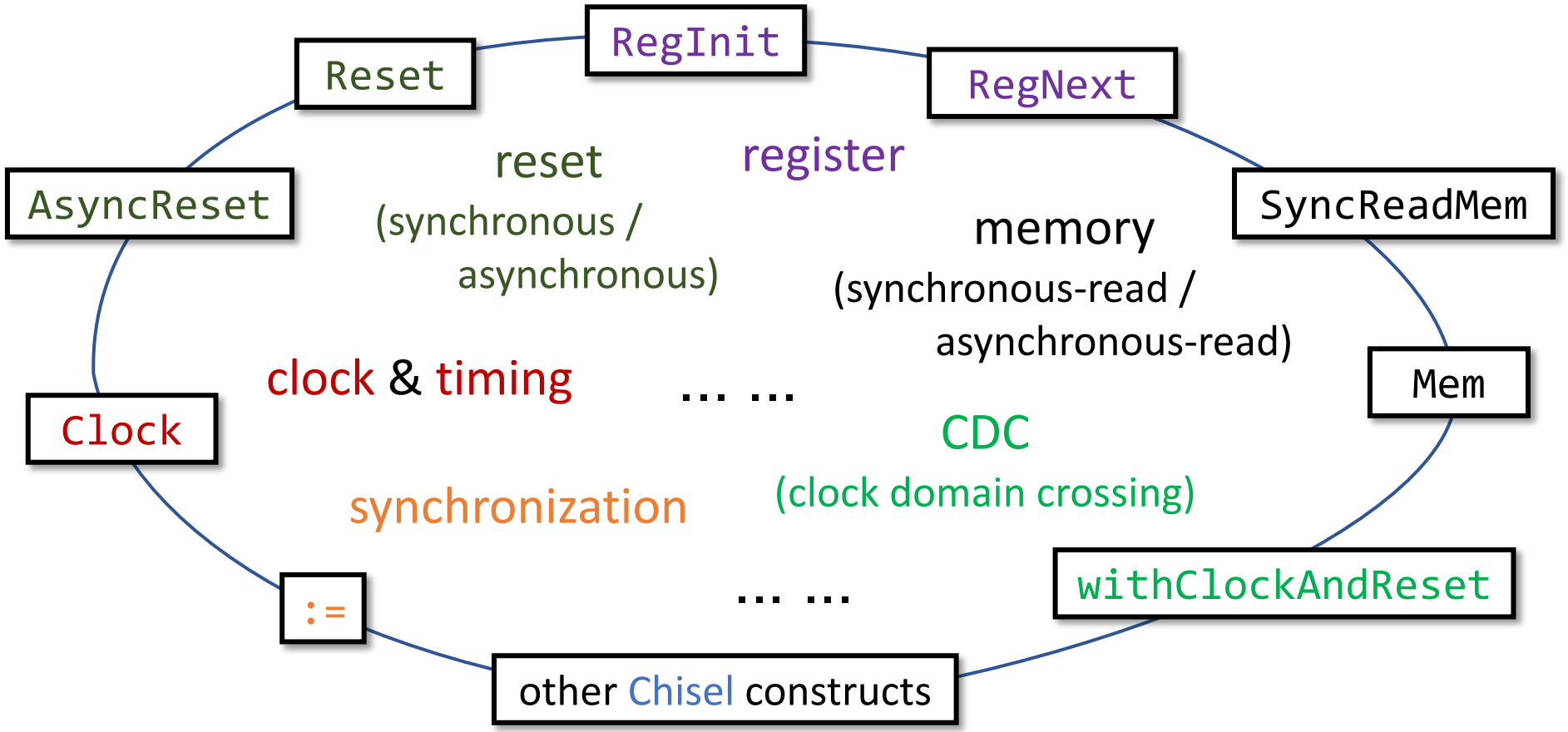
Theoretical Foundation



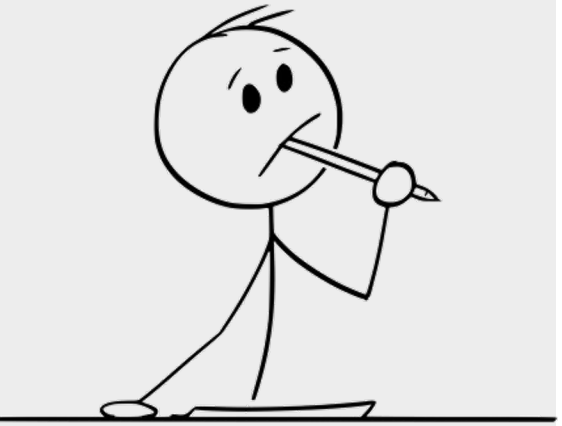
Hardware-Specific Intuition

Theoretical Foundation

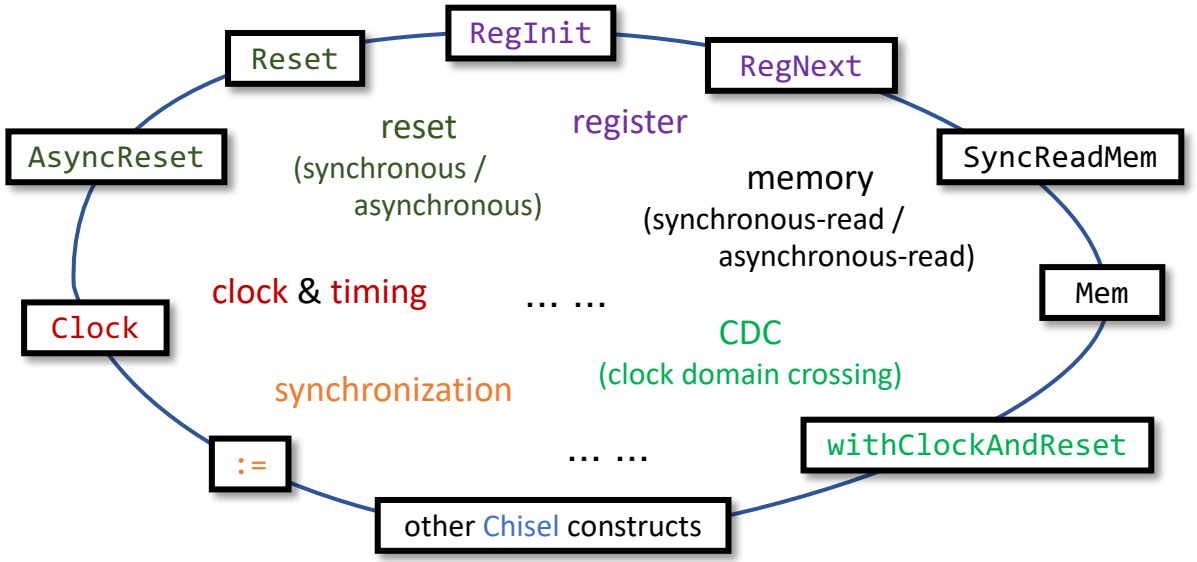
What is special about hardware programs?



Full of hardware-specific constructs uncommon in software.



Full of
hardware-specific constructs
uncommon in software.



Hardware-Specific Intuition

How to characterize dynamic hardware behavior?

We introduce λ_C, the first calculus to capture the essence of Chisel, and prove meta-theorems about λ_C that faithfully reflect the physical reality.

How to statically over-approximate hardware behavior?

Based on λ_C, we define and formalize HVFA that takes care of hardware-specific semantics, and prove analysis-essential properties about it.

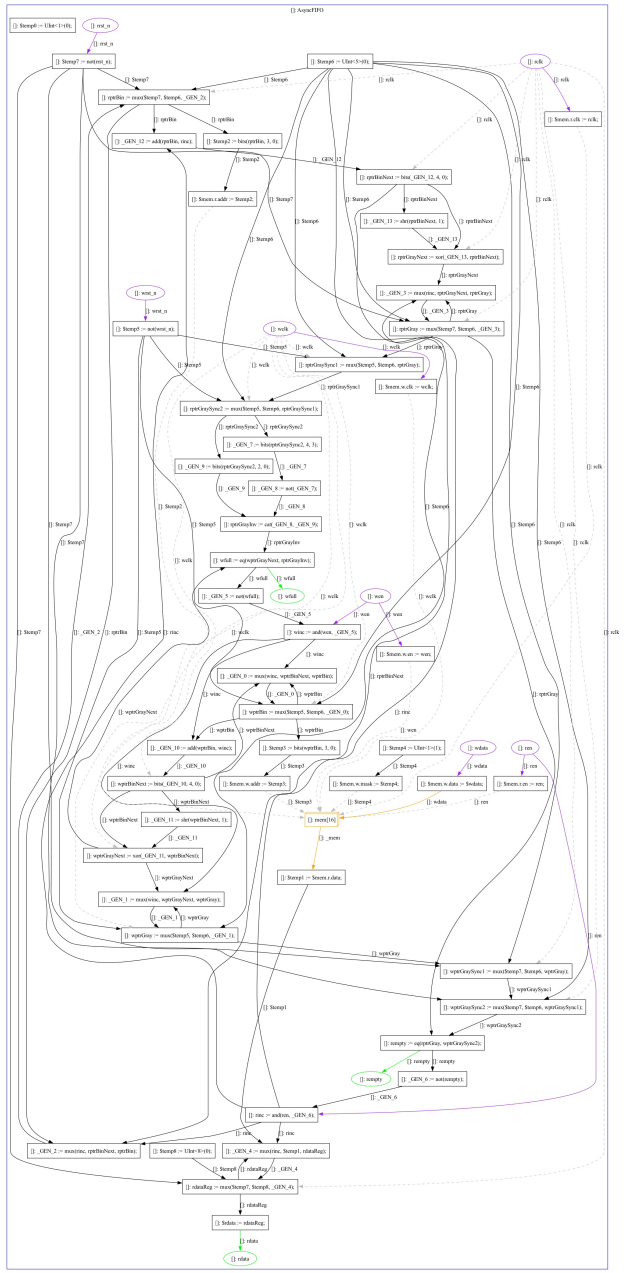
1

Theoretical Foundation

Theoretical Foundation: 14 (paper) + 6 (supplementary) pages of formal discussion.

Hardware-Specific Intuition

Theoretical Foundation



e.g., HVFG of an AsyncFIFO

Hardware Value Flow Graph (HVFG)

Nodes

Ports (In/Out)
- data
- clock
- reset

Statements

Connections
- port
- wire
- register

Monitors
- inspection
- verification

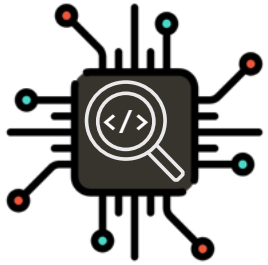
Mocking Entities
- memory
- black box
- ... ...

Edges

Value Flows
- intra-module flow
- inter-module flow

Control
- synchronization
- ...
- reset
- mux-switching

as a bonus!



ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

Evaluation Summary

3

Applications

Hardware-Specific Intuition

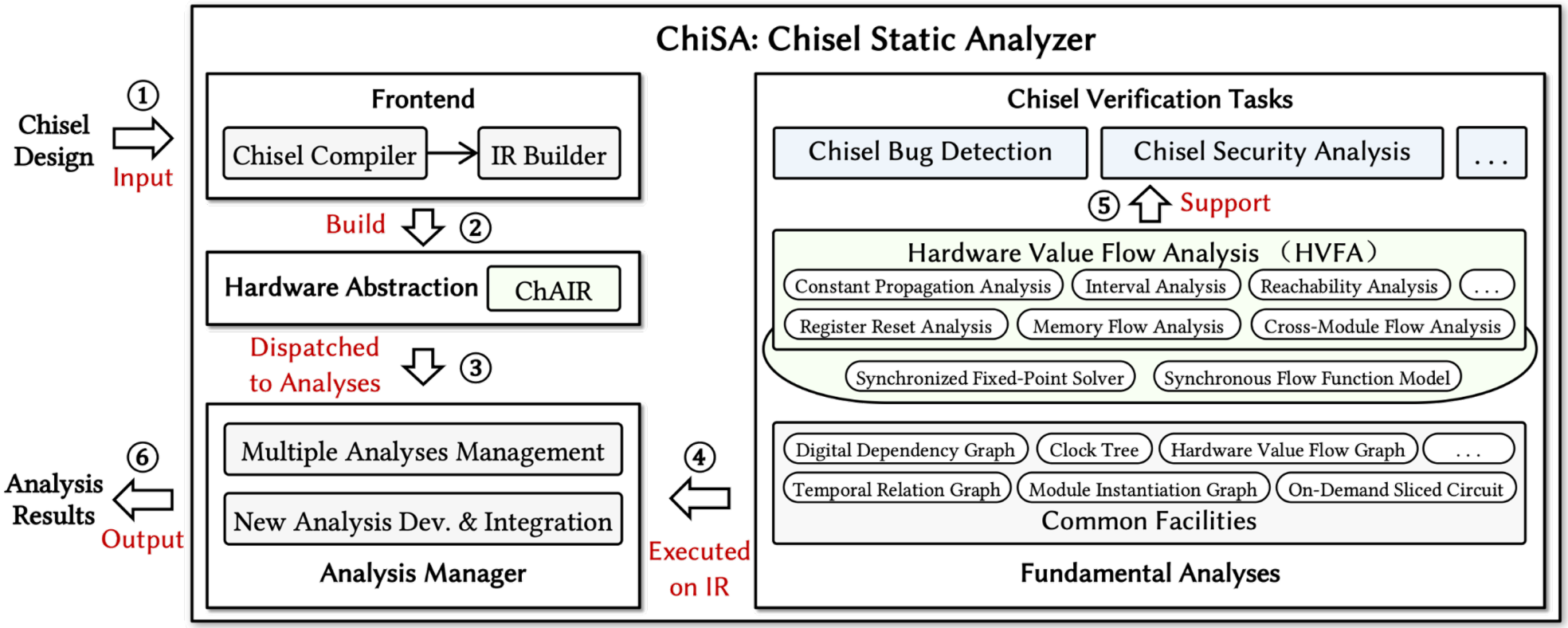
1

Theoretical Foundation

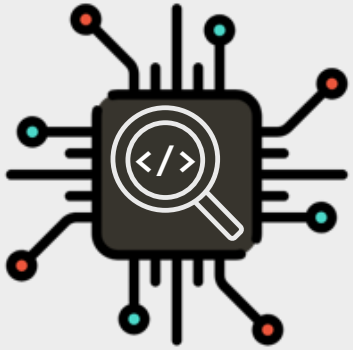
2

Proof of Concept Implementation

Brief Overview



The architecture and end-to-end workflow of **ChiSA**.

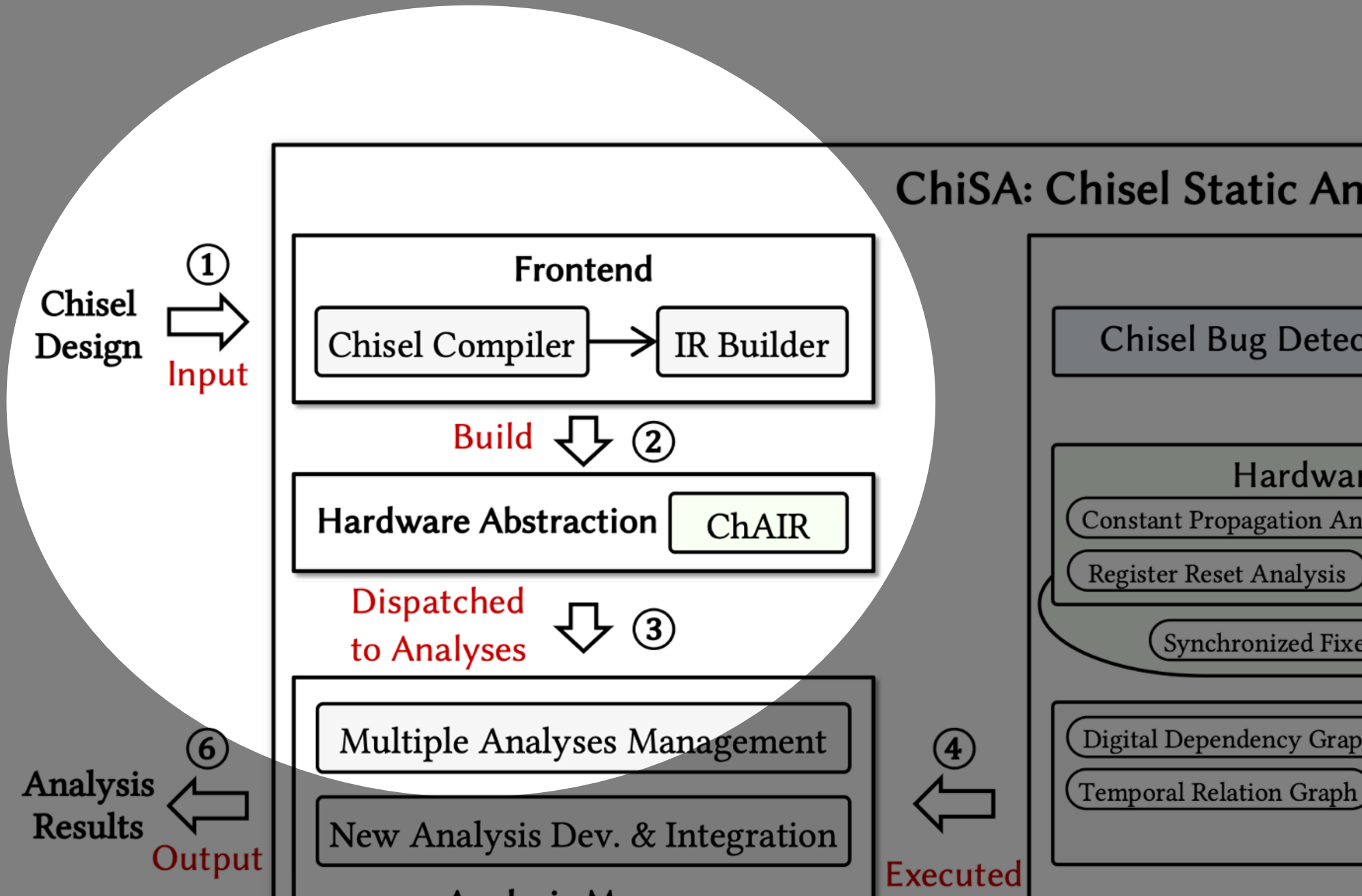


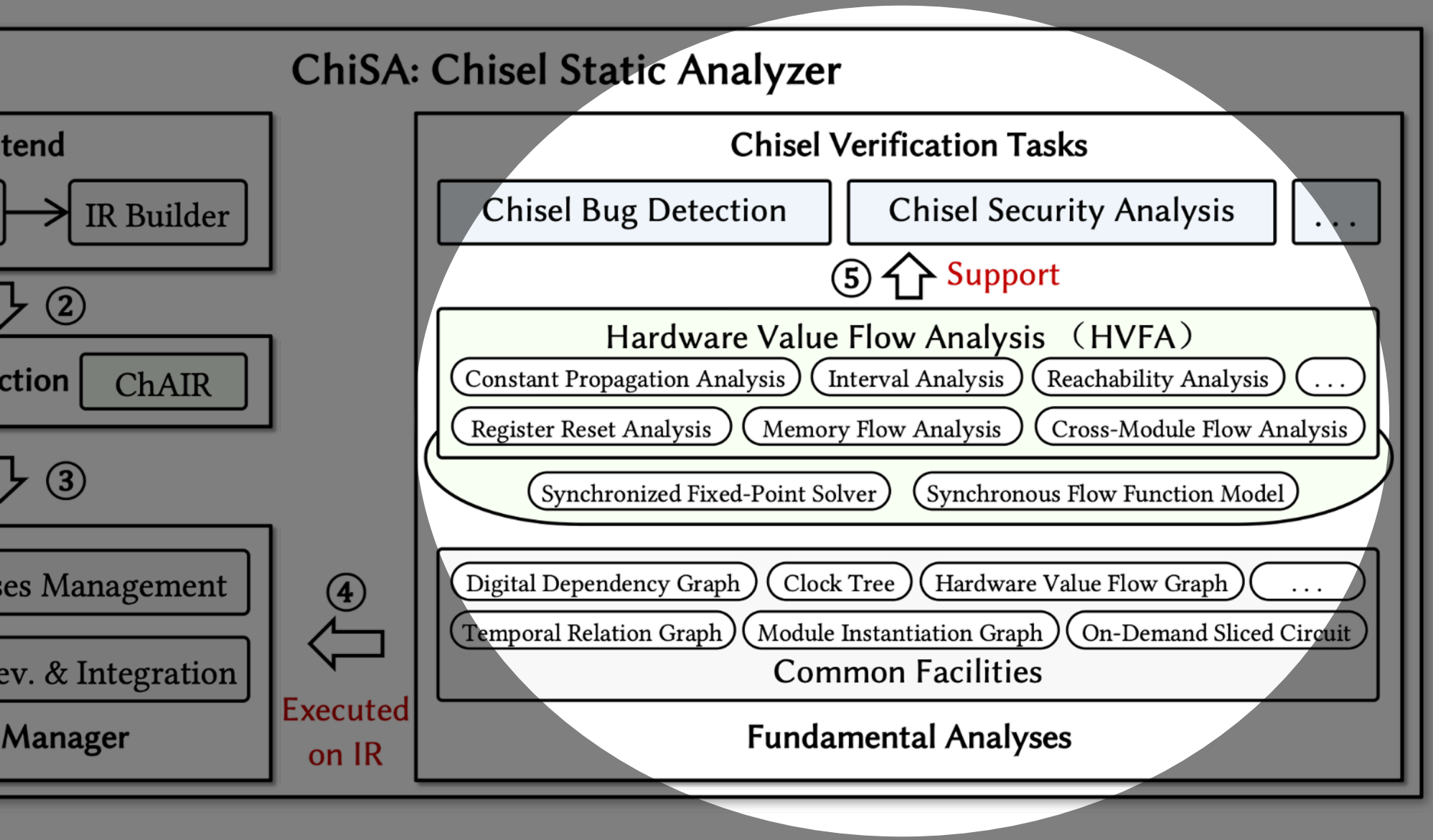
ChiSA

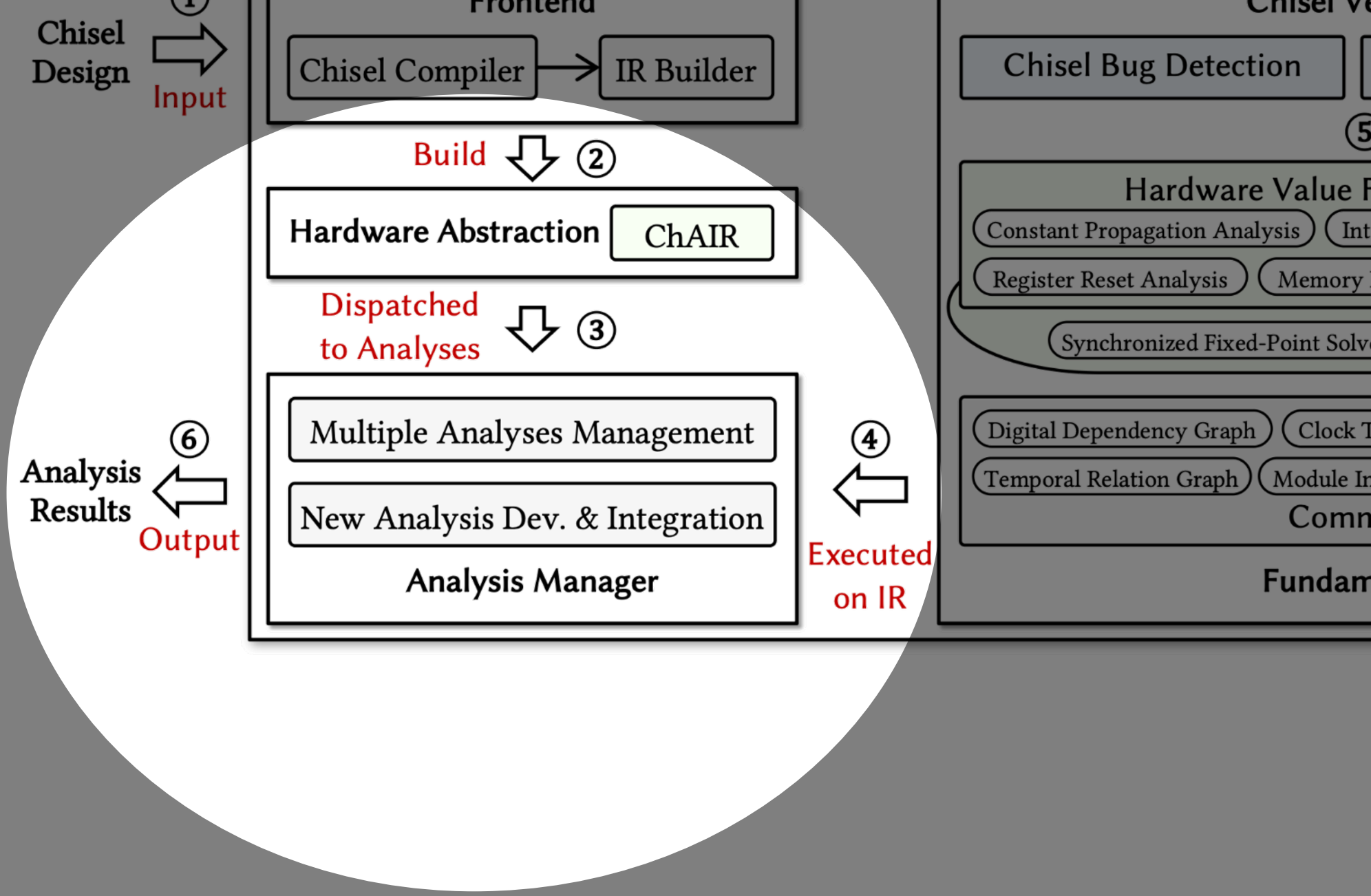
(Chisel Static Analyzer)

Brief Overview

Proof of Concept Implementation


ChiSA: Chisel Static An
Chisel Design
①
Input
Frontend
Chisel Compiler
IR Builder
Build
②
Hardware Abstraction
ChAIR
Dispatched to Analyses
③
Multiple Analyses Management
New Analysis Dev. & Integration
④
Executed
Chisel Bug Detec
Hardwar
Constant Propagation An
Register Reset Analysis
Synchronized Fixe
Digital Dependency Grap
Temporal Relation Graph
⑥
Analysis Results
Output


ChiSA: Chisel Static Analyzer
tend
IR Builder
②
ction
ChAIR
③
ses Management
ev. & Integration
Manager
④
Executed on IR
Chisel Verification Tasks
Chisel Bug Detection
Chisel Security Analysis
. . .
⑤
Support
Hardware Value Flow Analysis （HVFA）
Constant Propagation Analysis
Interval Analysis
Reachability Analysis
. . .
Register Reset Analysis
Memory Flow Analysis
Cross-Module Flow Analysis
Synchronized Fixed-Point Solver
Synchronous Flow Function Model
Digital Dependency Graph
Clock Tree
Hardware Value Flow Graph
. . .
Temporal Relation Graph
Module Instantiation Graph
On-Demand Sliced Circuit
Common Facilities
Fundamental Analyses

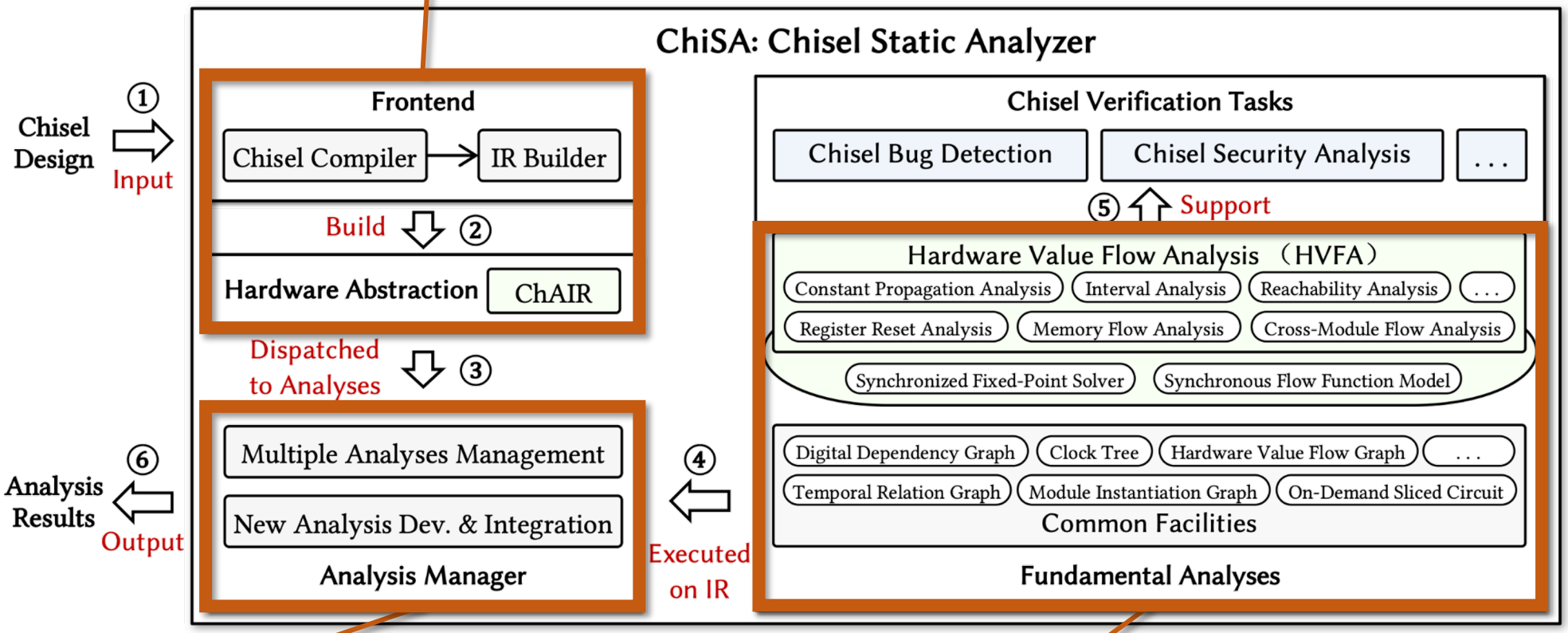

Chisel Design
Input
Frontend
Chisel Compiler
IR Builder
Build
②
Hardware Abstraction
ChAIR
Dispatched to Analyses
③
Multiple Analyses Management
New Analysis Dev. & Integration
Analysis Manager
⑥
Analysis Results
Output
④
Executed on IR
Chisel Bug Detection
Hardware Value
Constant Propagation Analysis
Register Reset Analysis
Synchronized Fixed-Point
Digital Dependency Graph
Temporal Relation Graph
Fundam

ChAIR

(Chisel Analysis Intermediate Representation)

- Structurally Simple (3AC, SSA, Linear)
- Semantically Expressive (for Chisel)

Reusable Infrastructures



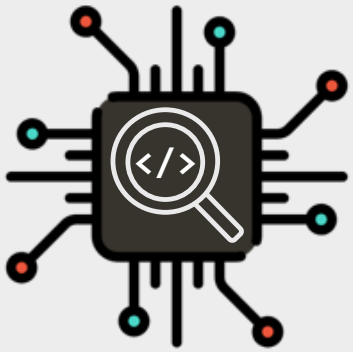
Analysis Manager

(ease development and extension)

- Orchestrate existing analyses
- Integrate new analyses

Fundamental Analyses

- a reusable HVFA framework
- some general-purpose instances (e.g. intervals)
- graph representations for Chisel programs

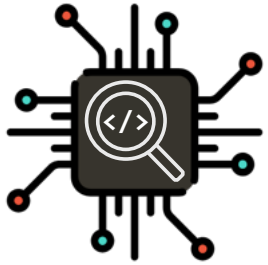


ChiSA

(Chisel Static Analyzer)

Brief Overview

Proof of Concept Implementation



ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

Evaluation Summary

3

Applications

Hardware-Specific Intuition

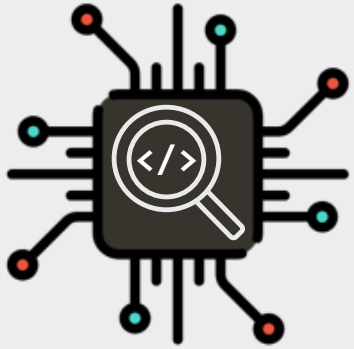
1

Theoretical Foundation

2

Proof of Concept Implementation

Brief Overview



In One Word

Evaluation Summary

Applications

ChiSA offers

produce helpful results

in terms of time and manual effort

an effective and significantly more lightweight approach

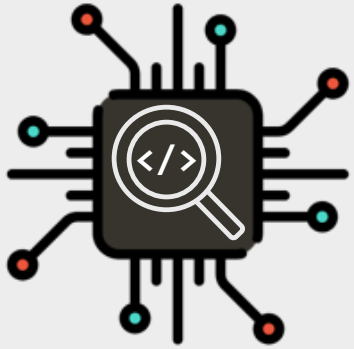
RQ1: hardware bug detection (e.g., identify violable assertions)

for representative Chisel verification tasks,

RQ2: hardware security analysis (e.g., detect unintended information flows)

especially on large and complex real-world designs.

ChiSABench: Chisel Static Analysis Benchmark



ChiSABench

Chisel
(Static Analysis)
Benchmark

Evaluation Summary

Applications

Official Toolchain Tests

System on Chips

Deep Neural Network Accelerator

Chisel3 (225K / 877)

Quasar (159K / 1)

RiscvMini (2K / 1)

Gemmini (632K / 1)

XiangShan (7.2M / 1)

Rocket (560K / 2)

BOOM (550K / 1)

Constellation (5K / 1)

TrustHub (1.1M / 25)

ChiselFlow (657 / 18)

Sodor (21K / 5)

IceNet (237K / 1)

Hwacha (553K / 1)

Security Benchmarks

Network on Chips

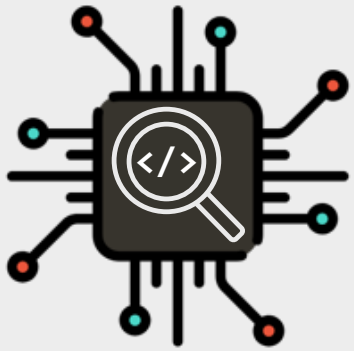
Vector Co-Processor

ChiSABench
(LoC = 11.3M / #designs = 935)

diversity
(purpose, feature, scale)

out-of-the-box accessibility
(all pre-elaborated into standalone files)

real-world
(mostly popular projects)



RQ1

Hardware Bug Detection

Evaluation Summary

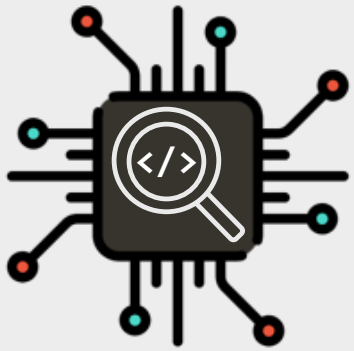
Applications

ChiSA vs. Bounded Model Checking

Static Assertion Analysis
(Insight: *approximate assertion violation conditions* with interval and constant HVFA)

BMC SOTA
ChiselTest-BMC [WOSET'21]

Feature	Benchmark	LoC	ChiSA			ChiselTest-BMC		
			#Violable (#P-Validated)	#Crashes	Time (s)	#Violable (#P-Validated)	#Crashes	Time (s)
Small-Scale	Chisel3 (877 designs)	256 (on average)	25 (24)	0	**3.1**	139 (139)	72	**2776.3**
Real-World	XiangShan	7,176,167	28 (23)	0	145.3	Assumption Errors		
	Gemmini	632,327	8 (7)	0	10.7	Internal Errors		
	Rocket	560,405	13 (13)	0	9.6	Incomplete Errors & Internal Errors		
	Hwacha	553,087	7 (7)	0	10.8	Internal Errors		
	Boom	550,147	7 (3)	0	17.6	Incomplete Errors		
	IceNet	236,506	0 (0)	0	3.8	Incomplete Errors		
	Constellation	5,389	6 (3)	0	0.1	Incomplete Errors		
	Total:	9,714,028	69 (56)	0	197.9	0 (0)	8	—



RQ1

Hardware Bug Detection

Evaluation Summary

Applications

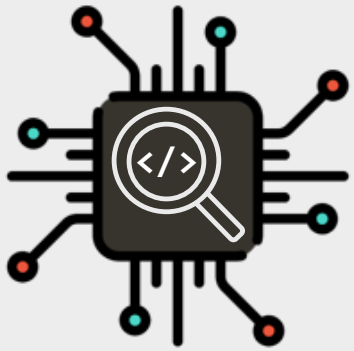
ChiSA vs. Bounded Model Checking

Static Assertion Analysis
(Insight: *approximate assertion violation conditions* with interval and constant HVFA)

BMC SOTA
ChiselTest-BMC [WOSET'21]

Feature	Benchmark	LoC	ChiSA			ChiselTest-BMC		
			#Violable (#P-Validated)	#Crashes	Time (s)	#Violable (#P-Validated)	#Crashes	Time (s)
Small-Scale	Chisel3 (877 designs)	256 (on average)	25 (24)	0	**3.1**	139 (139)	72	**2776.3**
Real-World	Xian…	7,176,167	28 (28)	0	145.2	Assumption Errors		
	Gem…					Internal Errors		
	Ro…					Incomplete Errors & Internal Errors		
	Hw…					Internal Errors		
	Bo…					Incomplete Errors		
	Ice…					Incomplete Errors		
	Constellation	5,389	6 (3)	0	0.1	Incomplete Errors		
	Total:	9,714,028	69 (56)	0	197.9	0 (0)	8	—

ChiSA is **Effective**
(eight were recognized by developers and scheduled for future fixes)



RQ1

Hardware Bug Detection

Evaluation Summary

Applications

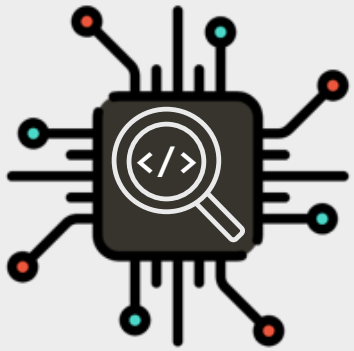
ChiSA vs. Bounded Model Checking

Static Assertion Analysis
(Insight: *approximate assertion violation conditions* with interval and constant HVFA)

BMC SOTA
ChiselTest-BMC [WOSET'21]

Feature	Benchmark	LoC	ChiSA #Violable (#P-Validated)	ChiSA #Crashes	ChiSA Time (s)	ChiselTest-BMC #Violable (#P-Validated)	ChiselTest-BMC #Crashes	ChiselTest-BMC Time (s)
Small-Scale	Chisel3 (877 designs)	256 (on average)	25 (24)	0	**3.1**	139 (139)	72	**2776.3**
Real-World	XiangS						Assumption Errors	
	Gemm						Internal Errors	
	Rock						te Errors & Internal Errors	
	Hwac						Internal Errors	
	Boom	550,147	7 (3)	0	17.6		Incomplete Errors	
	IceNet	236,506	0 (0)	0	3.8		Incomplete Errors	
	Constellation	5,389	6 (3)	0	0.1		Incomplete Errors	
	Total:	9,714,028	69 (56)	0	197.9	0 (0)	8	—

ChiSA is **Lightweight**
(finishes analysis for 9.7M+ LoC within 200s)



RQ1

Hardware Bug Detection

Evaluation Summary

Applications

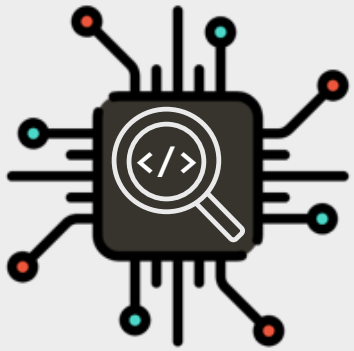
ChiSA vs. Bounded Model Checking

Static Assertion Analysis
(Insight: *approximate assertion violation conditions* with interval and constant HVFA)

BMC SOTA
ChiselTest-BMC [WOSET'21]

Feature	Benchmark	LoC	ChiSA #Violable (#P-Validated)	ChiSA #Crashes	ChiSA Time (s)	ChiselTest-BMC #Violable (#P-Validated)	ChiselTest-BMC #Crashes	ChiselTest-BMC Time (s)
Small-Scale	Chisel3 (877 designs)	256 (on average)	25 (24)	0	**3.1**	139 (139)	72	**2776.3**
Real-World	XiangShan	7,176,167	28 (23)	0	145.3	Assumption Errors		
	Gemmini	632,327	8 (7)	0	10.7	Internal Errors		
	Rocket	560,405	13 (13)	0	9.6	Incomplete Errors & Internal Errors		
	Hwacha	553,087	7 (7)	0	10.8	Internal Errors		
	Boom	550,147	7 (3)	0	17.6	Incomplete Errors		
						Incomplete Errors		
						Incomplete Errors		
						0 (0)	8	—

ChiselTest-BMC is **inapplicable** on large and complex real-world designs.
(all *crashed* with various errors)



RQ1

Hardware Bug Detection

Evaluation Summary

Applications

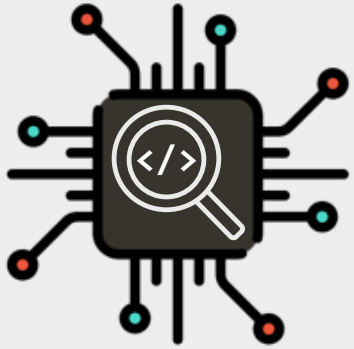
ChiSA vs. Bounded Model Checking

Static Assertion Analysis
(Insight: *approximate assertion violation conditions* with interval and constant HVFA)

BMC SOTA
ChiselTest-BMC [WOSET'21]

Feature	Benchmark	LoC	ChiSA #Violable (#P-Validated)	ChiSA #Crashes	ChiSA Time (s)	ChiselTest-BMC #Violable (#P-Validated)	ChiselTest-BMC #Crashes	ChiselTest-BMC Time (s)
Small-Scale	Chisel3 (877 designs)	256 (on average)	25 (24)	0	**3.1**	139 (139)	72	**2776.3**
Real-World	XiangShan	7,176,167	28 (23)	0	145.3	Assumption Errors		
	Gemmini	632,327	8 (7)	0	10.7	Internal Errors		
	Rocket	560,4						
	Hwacha	553,0						
	Boom	550,1						
	IceNet	236,5						
	Constellation	5,38				Incomplete Errors		
	Total:	9,714,028	69 (56)	0	197.9	0 (0)	8	—

ChiSA is significantly more lightweight than BMC.
(in terms of time: 3.1s vs 2776.3s)



RQ2

Hardware Security Analysis

Evaluation Summary

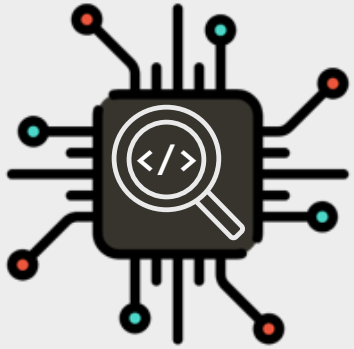
Applications

ChiSA vs. Secure Type Systems

Taint Analysis
(directly use the taint HVFA)

STS SOTA
ChiselFlow [CCS'18]

Benchmark	#Designs × Function	LoC	ChiSA			ChiselFlow	
			#Vulnerabilities (#FP / #FN)	#Annotations (#Sources / #Sinks)	Time (s)	#Annotations (Type Labels)	Time (s)
ChiselFlow	18 × *	655	19 (1 / 0)	44 (25 / 19)	0.006	228	14.475
TrustHub	19 × AES [85]	1,004,180	54 (0 / 0)	73 (19 / 54)	0.175	—	
	3 × ISCAS89 [21]	143,440	3 (0 / 0)	6 (3 / 3)	0.435		
	1 × PIC16F84 [65]	5,932	1 (0 / 0)	2 (1 / 1)	0.017		
	2 × RSA [83]	2,302	2 (0 / 0)	4 (2 / 2)	0.005		
	Total:	1,155,854	60 (0 / 0)	85 (25 / 60)	0.632		



RQ2

Hardware Security Analysis

Evaluation Summary

Applications

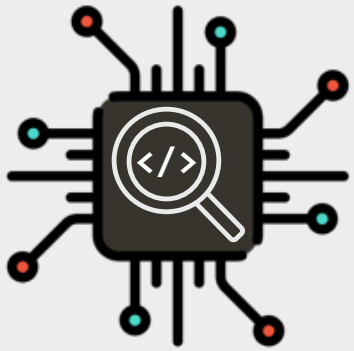
ChiSA vs. Secure Type Systems

Taint Analysis
(directly use the taint HVFA)

STS SOTA
ChiselFlow [CCS'18]

Benchmark	#Designs × Function	LoC	ChiSA #Vulnerabilities (#FP / #FN)	ChiSA #Annotations (#Sources / #Sinks)	ChiSA Time (s)	ChiselFlow #Annotations (Type Labels)	ChiselFlow Time (s)
ChiselFlow	18 × *	655	19 (1 / 0)	44 (25 / 19)	0.006	228	14.475
TrustHub							–
	Total:	1,155,854	60 (0 / 0)	85 (25 / 60)	0.632		

ChiSA is **Effective**
(identified all vulnerabilities with only 1 false positive)



RQ2

Hardware Security Analysis

Evaluation Summary

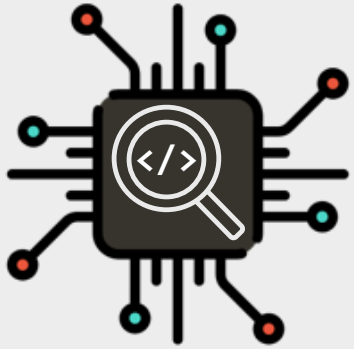
Applications

ChiSA vs. Secure Type Systems

Taint Analysis (directly use the taint HVFA)		STS SOTA ChiselFlow [CCS'18]

Benchmark	#Designs × Function	LoC	ChiSA #Vulnerabilities (#FP / #FN)	ChiSA #Annotations (#Sources / #Sinks)	ChiSA Time (s)	ChiselFlow #Annotations (Type Labels)	ChiselFlow Time (s)
ChiselFlow							475
TrustHub	19 ×					—	
	3 ×						
	1 × PIC16F84 [63]	5,352	1 (0 / 0)	2 (1 / 1)	0.017		
	2 × RSA [83]	2,302	2 (0 / 0)	4 (2 / 2)	0.005		
	Total:	1,155,854	60 (0 / 0)	85 (25 / 60)	0.632		

ChiSA is **Lightweight**
(finishes analysis for 1.1M LoC within 85 annotations and 0.632s)



RQ2

Hardware Security Analysis

Evaluation Summary

Applications

ChiSA vs. Secure Type Systems

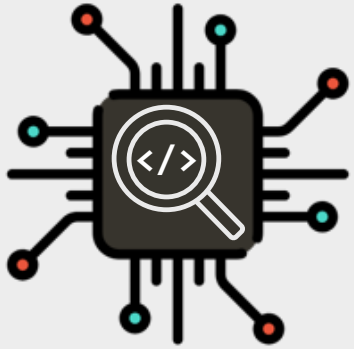
Taint Analysis
(directly use the taint HVFA)

STS SOTA
ChiselFlow [CCS'18]

Benchmark	#Designs × Function	LoC	ChiSA			ChiselFlow	
			#Vulnerabilities (#FP / #FN)	#Annotations (#Sources / #Sinks)	Time (s)	#Annotations (Type Labels)	Time (s)
ChiselFlow	18 × *	655	19 (1 / 0)	44 (25 / 19)	0.006	228	14.475
TrustHub	19 × AES [85]	1,004,180					
	3 × ISCAS89 [21]	143,440					
	1 × PIC16F84 [65]	5,932					
	2 × RSA [83]	2,302					
	Total:	1,155,854					

Inapplicable
It is prohibitive to *retrofit a million-line-scale codebase* with an annotation-intensive STS.

–



RQ2

Hardware Security Analysis

Evaluation Summary

Applications

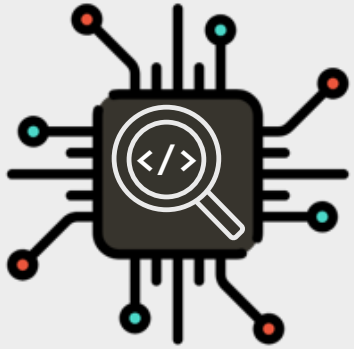
ChiSA vs. Secure Type Systems

Taint Analysis
(directly use the taint HVFA)

STS SOTA
ChiselFlow [CCS'18]

Benchmark	#Designs × Function	LoC	ChiSA			ChiselFlow	
			#Vulnerabilities (#FP / #FN)	#Annotations (#Sources / #Sinks)	Time (s)	#Annotations (Type Labels)	Time (s)
ChiselFlow	18 × *	655	19 (1 / 0)	44 (25 / 19)	0.006	228	14.475
TrustHub	19 × AES [85] 3 × ISCAS89 [21] 1 × PIC16F84 [65] 2 × RSA [83]	1,004,180	54 (0 / 0)	73 (19 / 54)	0.175		
	Total:	1,155,854	60 (0 / 0)	85 (25 / 60)	0.632		

ChiSA is significantly more lightweight than STS.
(in terms of manual effort: 44 vs 228 annotations)



In One Word

Evaluation Summary

Applications

ChiSA offers

produce helpful results

in terms of time and manual effort

an effective and significantly more lightweight approach

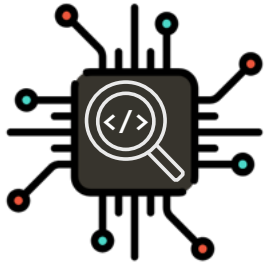
RQ1: hardware bug detection (e.g., identify violable assertions)

for representative Chisel verification tasks,

RQ2: hardware security analysis (e.g., detect unintended information flows)

especially on large and complex real-world designs.

ChiSABench: Chisel Static Analysis Benchmark



ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

Evaluation Summary

3

Applications

Hardware-Specific Intuition

1

Theoretical Foundation

2

Proof of Concept Implementation

Brief Overview

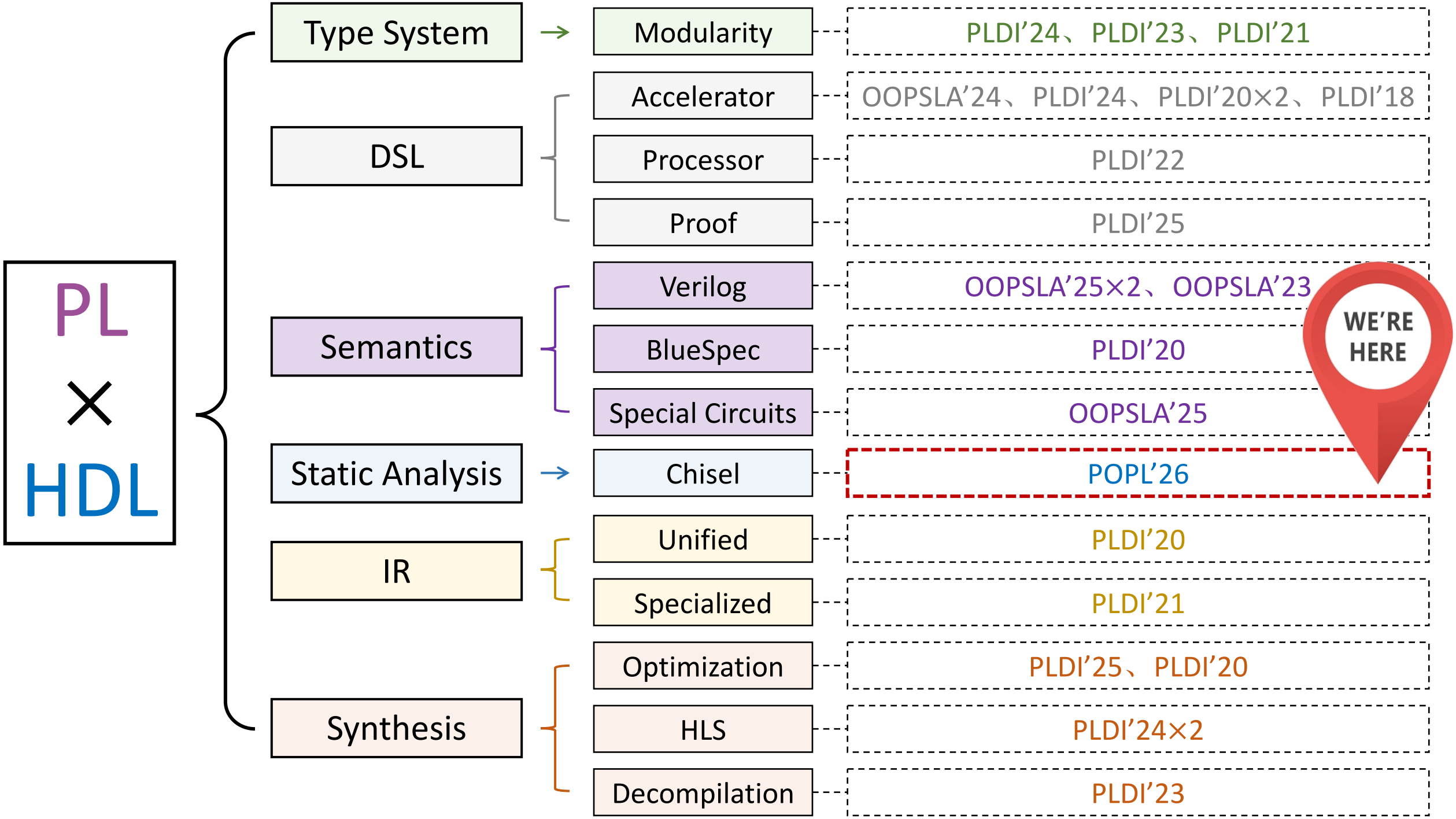
Research Opportunities

PL × HDL

TODO :)

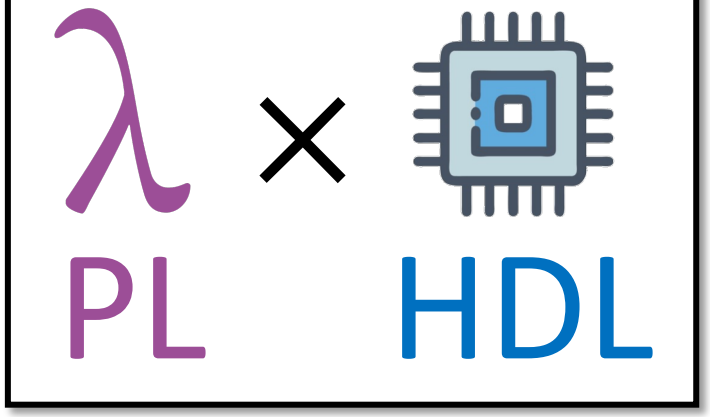
TODO :)

ChiSA: **Static Analysis** for Lightweight **Chisel Verification**

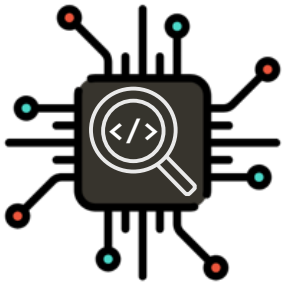

PL
×
HDL
Type System
DSL
Semantics
Static Analysis
IR
Synthesis
Modularity
Accelerator
Processor
Proof
Verilog
BlueSpec
Special Circuits
Chisel
Unified
Specialized
Optimization
HLS
Decompilation
PLDI'24、PLDI'23、PLDI'21
OOPSLA'24、PLDI'24、PLDI'20×2、PLDI'18
PLDI'22
PLDI'25
OOPSLA'25×2、OOPSLA'23
PLDI'20
OOPSLA'25
POPL'26
PLDI'20
PLDI'21
PLDI'25、PLDI'20
PLDI'24×2
PLDI'23
WE'RE HERE

Research Opportunities

Jiacai Cui @ NJU, China



TODO :)



ChiSA: **Static Analysis** for Lightweight **Chisel Verification**