

Versatile Hardware Analysis Techniques

From Waveform-based Analysis to Formal Verification

Lucas Klemmer

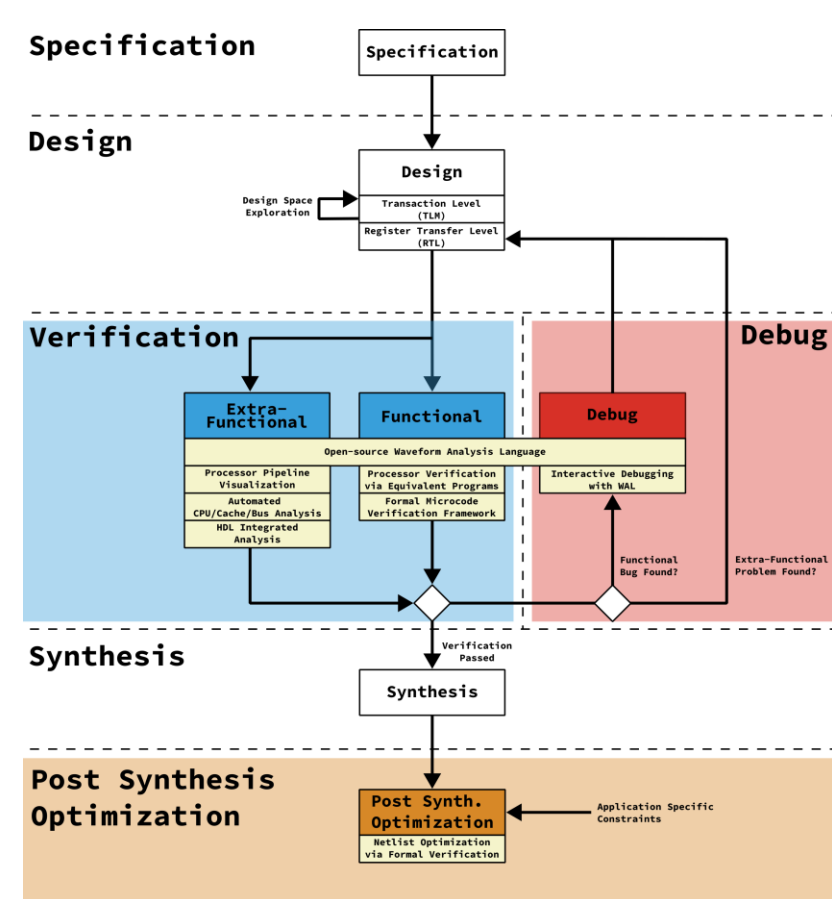
Institute for Complex Systems, Johannes Kepler University Linz

lucas.klemmer@jku.at

Advisor: Prof. Dr. Daniel Große

Introduction

- Hardware analysis important step during most development phases
- Problems:
 - Increasing design sizes
 - Increasing number of custom designs
 - Low supply of domain experts
- **Automation is mandatory**
- This thesis presents HW analysis techniques
 - (Extra-)Functional Verification
 - Debugging
 - Post Synthesis Optimization
- All results are open-source and use open-source tools



Processor Verification [1]

- Equivalent Program EXecution (EPEX)
- Broaden test programs automatically
- Generate new test programs automatically
- For a test program P , formally generates an equivalent program \hat{P} activating different control / data paths

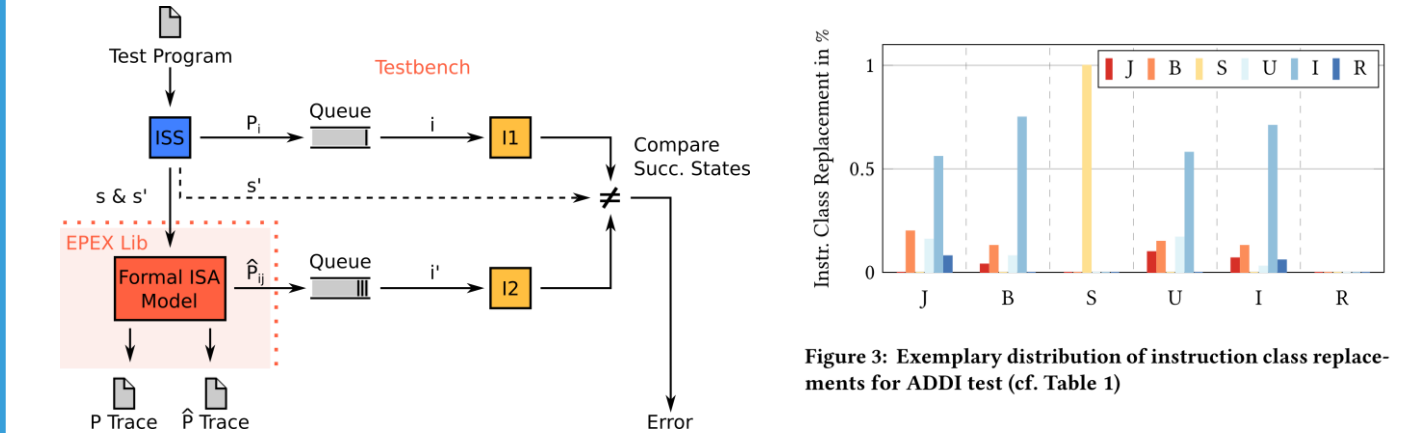
```

0996: ...
# assume t1 = 2, t2 = 2046, t3 = 0
1000: add t3, t1, t2 # t3 = t1 + t2 = 2048
1004: ...
(a) Excerpt of test program P

0996: ...
# assume t1 = 2, t2 = 2046, t3 = 0
1000: slll t3, t1, 10 # shift left t1 by 10
1004: ...
(b) Excerpt of equivalent test program P-hat

0996: ...
# assume t1 = 2, t2 = 2046, t3 = 0
1000: j 1044 # set pc to 2044
1004: ...
(c) Excerpt of equivalent test program P-hat

2044: jal t3, -1040 # jump back and link t3 =
2044 + 4 = 2048
    
```



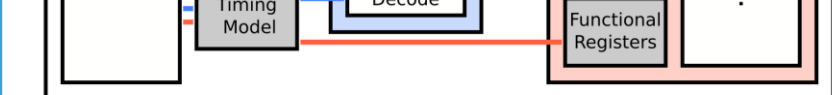
Formal Microcode Verification [2,3]

- The ultimate form of RISC is only one instruction
- SUBLEQ**: subtract and branch if less or equal 0
- Programming in SUBLEQ is hard → **RISC-V layer on top**
- Developed **fully compliant SUBLEQ microcode** implementing **RV32I**
- Built **VP** and **formally verified** all RV32I instructions
- Best paper award** at FDL'22

```

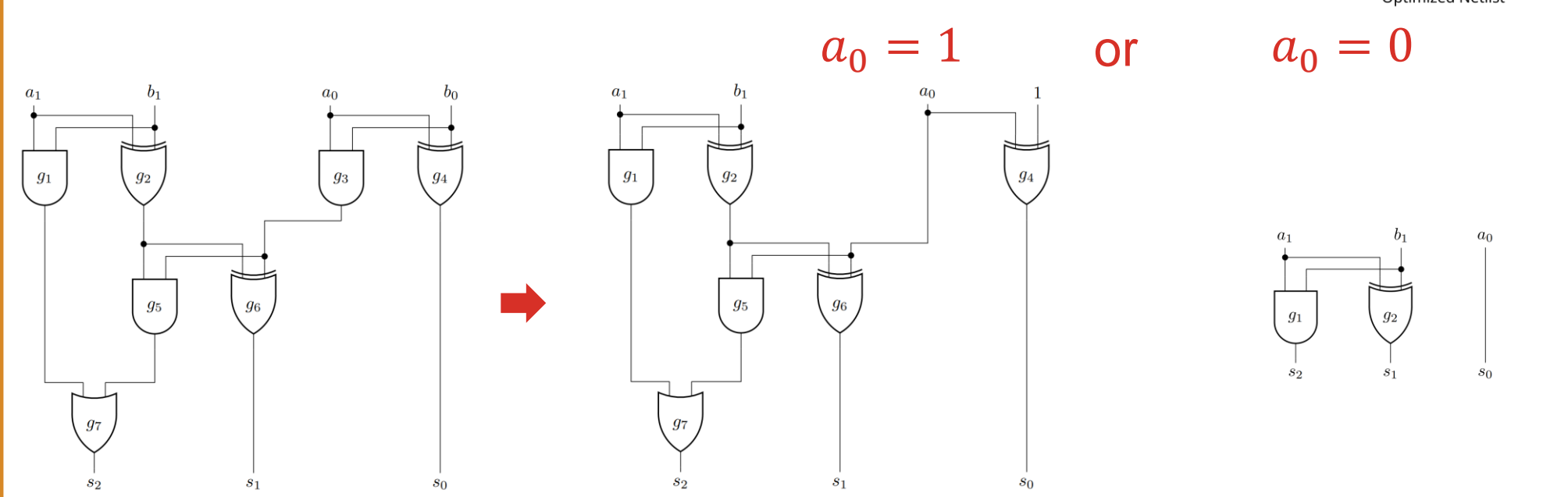
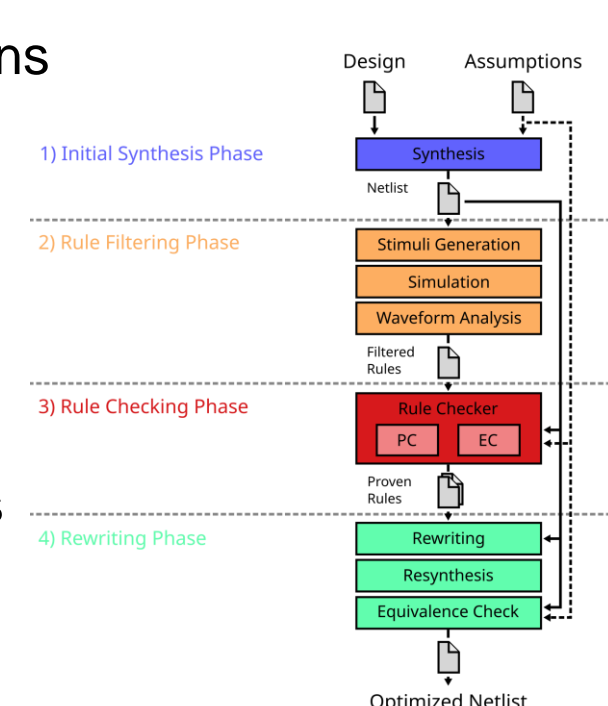
SRC1 TMP0 1      # TMP0 = -SRC1
TMP0 SRC2 1      # SRC2 = SRC2 - (-SRC1)
TMP0 TMP0 1      # TMP0 = 0

(rv-verify
  #name "JAL"
  #init-pc JAL-PC
  #fuel 20
  #microcode microcode
  #solver (boolector)
  #spec
  (lambda (res)
    (and (eq? (list-ref res REG-RVPC)
              (bvadd val-rvpc val-immi))
         (eq? (list-ref res REG-RSLT)
              (bvadd val-rvpc (bv 4 XLEN))))))
  #assumptions
  (lambda (res)
    (assume (eq? (bv 0 2)
                 (extract 1 0 (list-ref res 1))))))
    
```



Formal Netlist Optimization [4, 5]

- External Don't Cares** allow post-synthesis optimizations
 - Reduced instruction set
 - Limited input value range
- Each gate is checked using formal methods
- We propose **FSYN**
 - XDC optimization tool
 - Fully built on open-source technologies
 - Property Checking/Equivalence Checking backends
 - Distributed



Waveform Analysis Language [6,7,8,9]

- How do high-level configuration options impact performance?
- How can development tools support today's diverse designs?
- Staring at waveforms is too much manual work
- Extending testbenches not very flexible and reusable
- Programmable Waveform Analysis** can solve this problem
 - All information is inside waveforms
- Waveform analysis programs should be:
 - Generic and reusable
 - Easy to integrate into existing flows and tools
 - Written in convenient language and not duct-taped to other systems

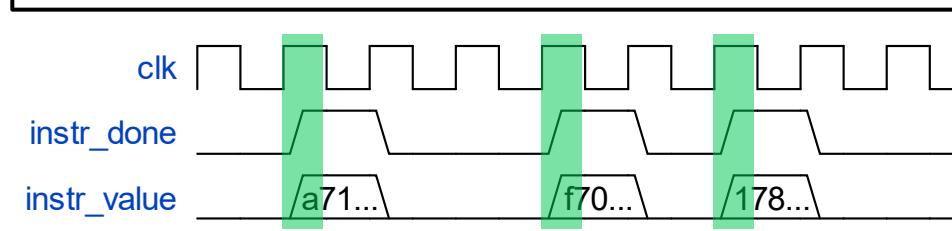


wal-lang.org

```

(count (&& (rising clk) instr_done))

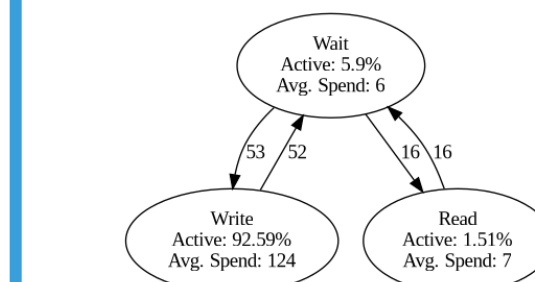
(whenever (= instr_value[6:0] 111)
  (print "JAL"))
    
```



Verification

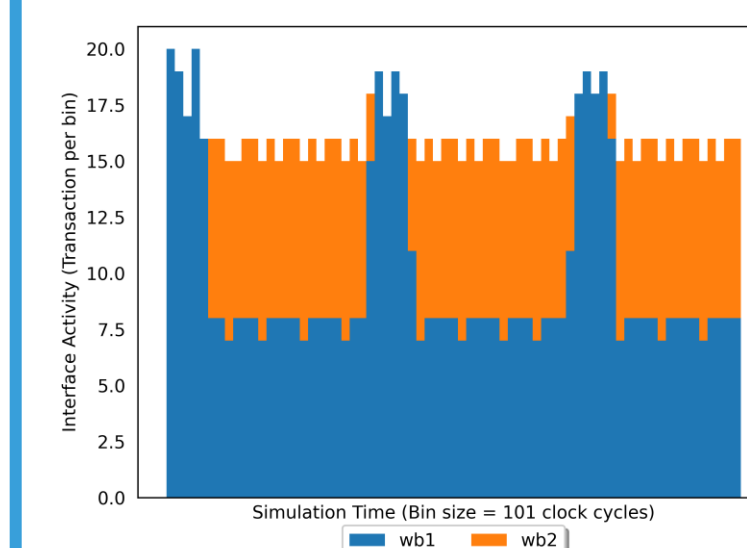
- Reusable analysis programs**
- Generic core algorithm + little design-specific code
- Performance analysis of**
 - Processors
 - Busses
 - Caches
- Use as a **compilation target**
- Check **SVA** on waveforms
- Embeddable into HDL design
- Add analysis to **waveform viewers**

Core	Configuration	IPC	Stalled Cycles
SERV	servant	0.02	not pipelined
PicoRV32	default	0.24	not pipelined
VexRiscv	microNoCsr	0.33	63%
VexRiscv	smallest	0.33	66%
VexRiscv	smallAndProductive	0.42	54%
VexRiscv	smallAndProductiveCache	0.47	51%
VexRiscv	twoThreeStage	0.47	48%
VexRiscv	secure	0.57	42%
VexRiscv	linux	0.59	38%
VexRiscv	full	0.57	35%
VexRiscv	fullNoMmuMaxPerf	0.63	33%
IBEX	default	0.63	48%
IBEX	icache	0.89	19%
TGC	3-Stage	0.61	65%
TGC	4-Stage v1	0.72	49%
TGC	4-Stage v2	0.70	45%
TGC	4-Stage v3	0.70	44%
TGC	4-Stage v4	0.68	43%
TGC	5-Stage	0.78	40%



```

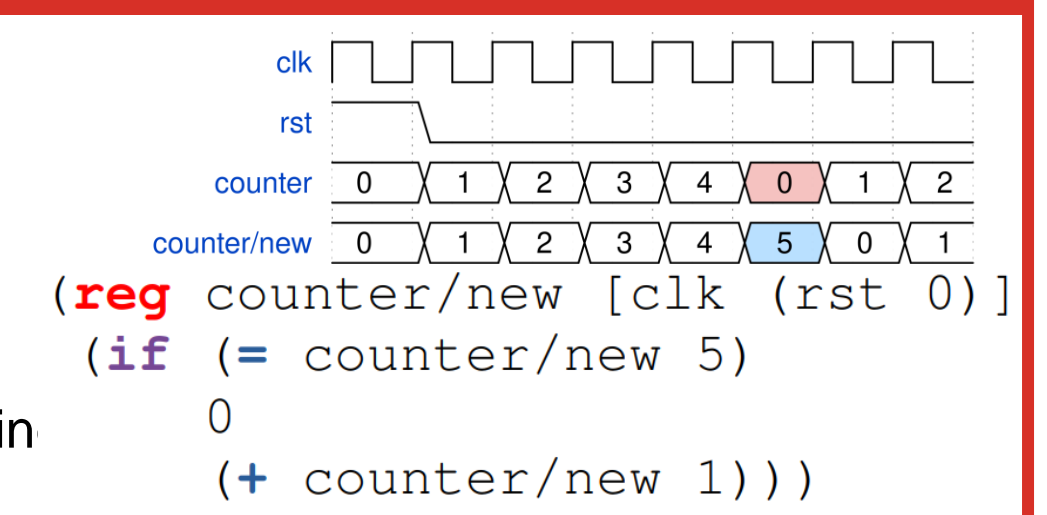
set_cnt: assert property(
  @(posedge clk)
  disable iff(rst)
  start | => (cnt == value));
    
```



Time	fetch	decode	execute	memory	writback
23	lw x7, 84(x1)	sub x7, x7, x2	add x7, x4, x5	ll x4, x7, x2	
24	lw x2, 96(x1)	sub x7, x7, x2	add x7, x4, x5	st x4, x7, x2	
25	add x8, x2, x3	lw x2, 96(x1)	sw x7, 84(x3)	sub x7, x4, x5	add x7, x4, x5
26	jal x3, -8	add x8, x2, x5	lw x2, 96(x1)	sw x7, 84(x3)	sub x7, x7, x2
27	jal x3, -8	add x8, x2, x5	lw x2, 96(x1)	sw x7, 84(x3)	sw x7, 84(x3)
28	add x2, x1, 1	jal x3, -8	add x8, x2, x5	lw x2, 96(x1)	lw x2, 96(x1)
29	add x2, x2, x9	add x7, x2, 1	jal x3, -8	add x8, x2, x5	
30	add x2, x2, x9	pc: 108	pc: 2	jal x3, -8	add x9, x2, x5
31	sw x2, 30(x1)	pc: 0	pc: 1	sw x2, 30(x1)	jal x3, -8

Debug

- Virtual Signals**
- Inject new logic into waveforms
- Use-cases
 - Debugging
 - Abstraction
 - Analysis
- Allows highly interactive debugging without need for re-simulation



References

- L. Klemmer and D. Große. 2021. "EPEX: Processor Verification by Equivalent Program Execution." GLSVLSI, 2021, pp. 33–38.
- L. Klemmer and D. Große. "An Exploration Platform for Microcoded RISC-V Cores leveraging the One Instruction Set Computer Principle." ISVLSI, 2022, pp. 38-43
- L. Klemmer, S. Gurtner and D. Große. "Formal Verification of SUBLEQ Microcode implementing the RV32I ISA." FDL, 2022, pp. 1-8
- L. Klemmer, D. Bonora and D. Große. "Large-scale Gatelevel Optimization Leveraging Property Checking." DVCon Europe, 2023, pp. 86-93.
- D. Große, L. Klemmer, and D. Bonora. "Using Formal Verification Methods for Optimization of Circuits under External Constraints." DATE, 2024
- L. Klemmer and D. Große. "WAL: A Novel Waveform Analysis Language for Advanced Design Understanding and Debugging." ASP-DAC, 2022, pp. 358-364
- L. Klemmer and Daniel Große. 2022. "Waveform-based performance analysis of RISC-V processors: late breaking results". DAC, 2022 22
- F. Skarman, L. Klemmer, O. Gustafsson and D. Große. "Enhancing Compiler-Driven HDL Design with Automatic Waveform Analysis." FDL, 2023, pp. 1-8
- L. Klemmer and D. Große. 2024. "Towards a Highly Interactive Design-Debug-Verification Cycle" ASP-DAC, 2024