ETH zürich

Pathfinder: Constructing Cycle-accurate Taint Graphs for Analyzing Information Flow Traces

Katharina Ceesay-Seitz,

Flavien Solt, Alexander Klukas, Kaveh Razavi

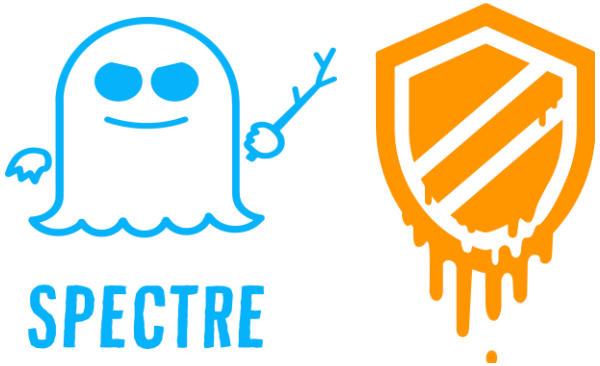
DVClub Cambridge

4th December 2025

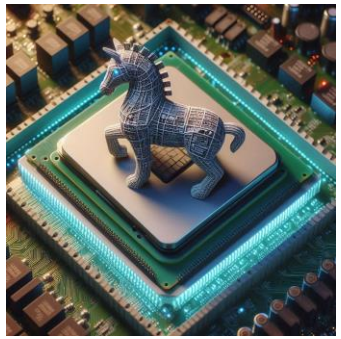
Published at International Conference on Computer-Aided Design (ICCAD'25)

COMSEC

Timing
side channels



Speculative execution
attacks

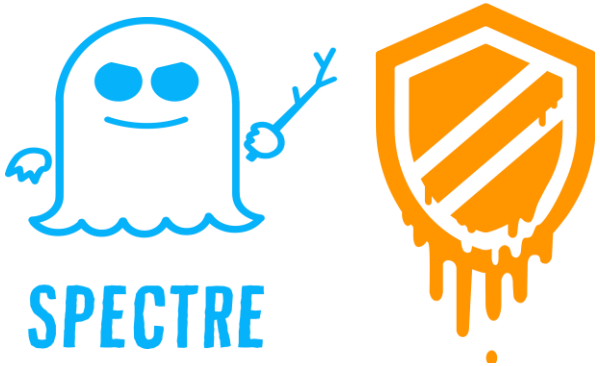


Hardware Trojans

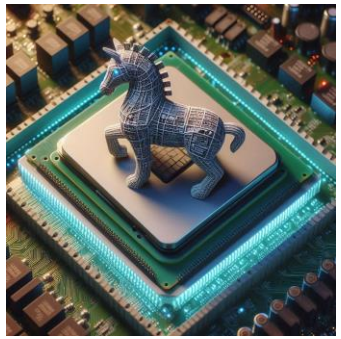
Access control
violations

AI model theft

Timing
side channels



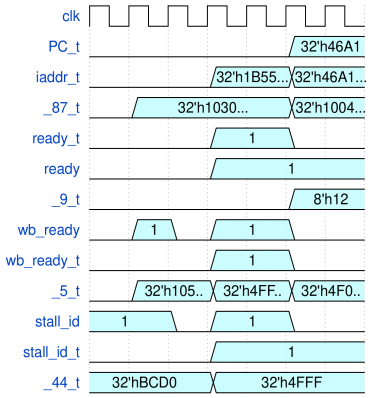
Speculative execution
attacks



Hardware Trojans

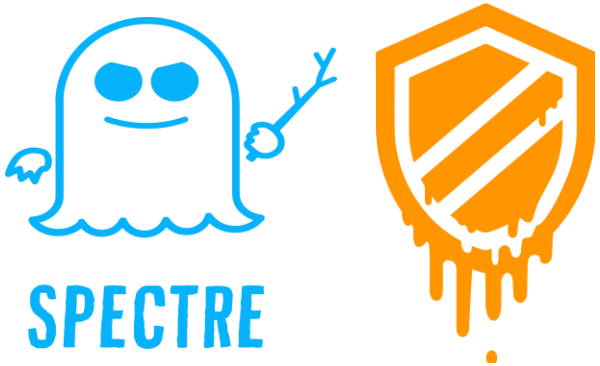
Access control
violations

AI model theft

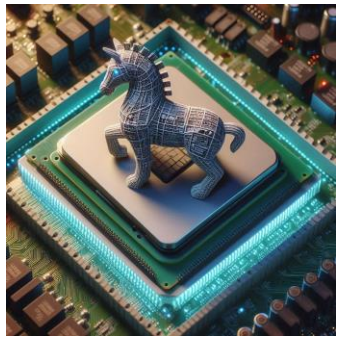


Information
flow (taint)
tracking
violation
trace

Timing
side channels



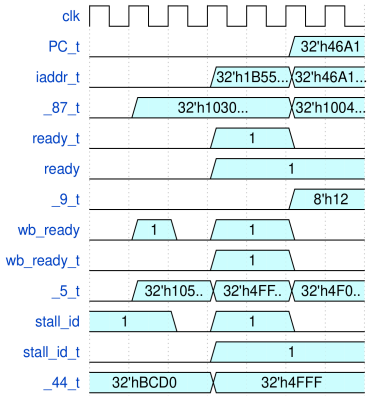
Speculative execution
attacks



Hardware Trojans

Access control
violations

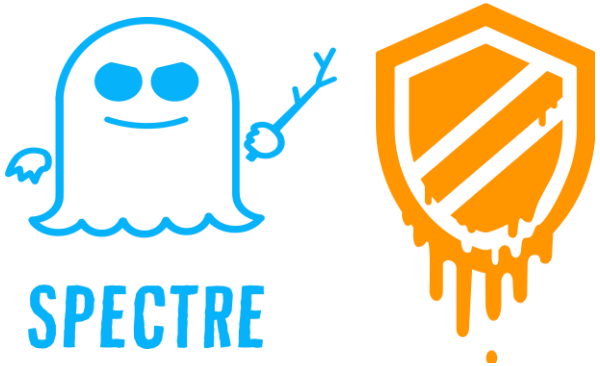
AI model theft



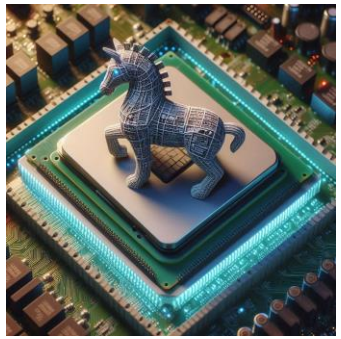
Information
flow (taint)
tracking
violation
trace

BOOM CPU - Spectre:
17 317 tainted signals

Timing
side channels



Speculative execution
attacks

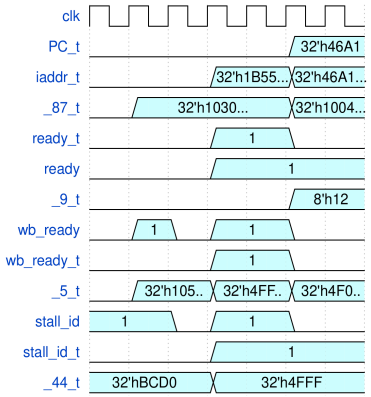


Hardware Trojans

Access control
violations

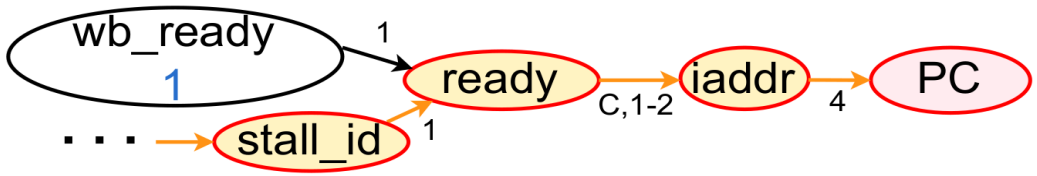
AI model theft

Pathfinder



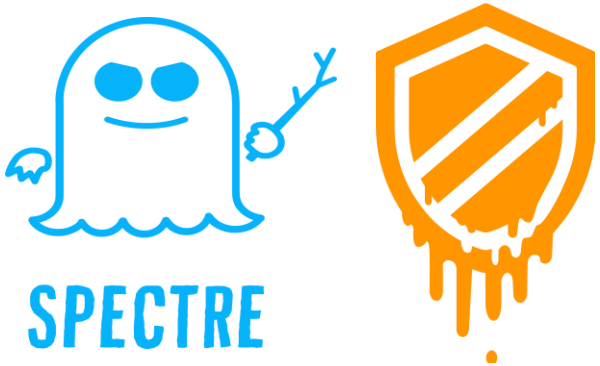
Information
flow (taint)
tracking
violation
trace

Taint graph

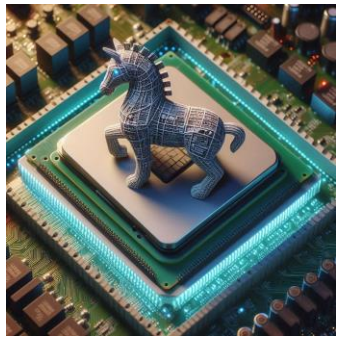


BOOM CPU - Spectre:
17 317 tainted signals

Timing
side channels



Speculative execution
attacks

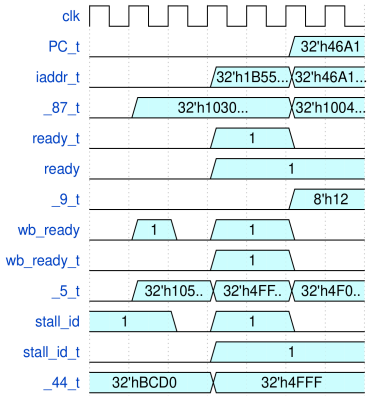


Hardware Trojans

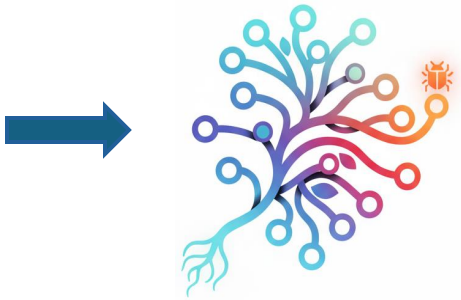
Access control
violations

AI model theft

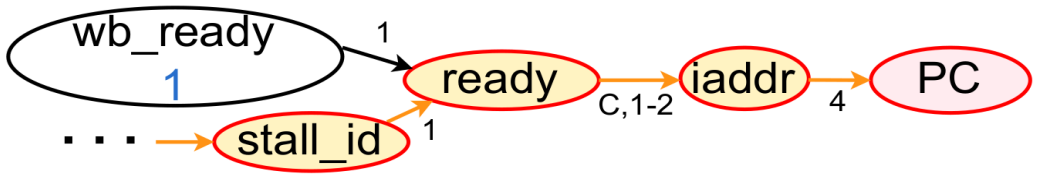
Pathfinder



Information
flow (taint)
tracking
violation
trace



Taint graph

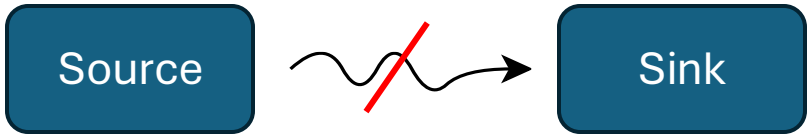


BOOM CPU - Spectre:
17 317 tainted signals

BOOM CPU - Spectre:
213 signals on path

Security Properties

Information flow properties

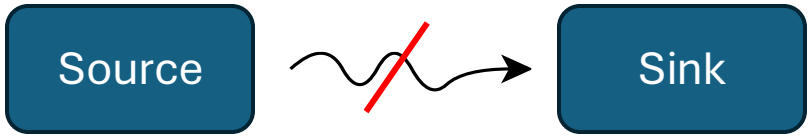


Information =

- data flows
- time & control flows

Security Properties

Information flow properties

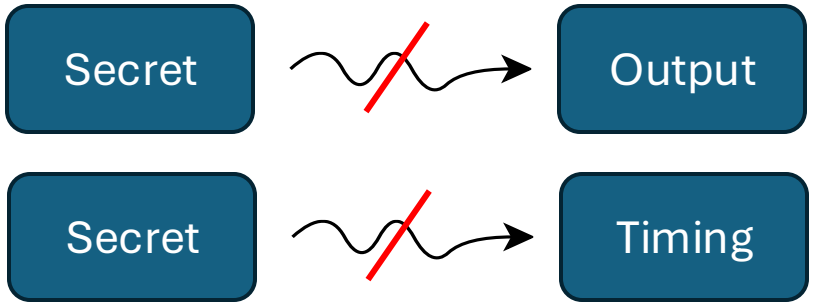


Information =

≠ data flows

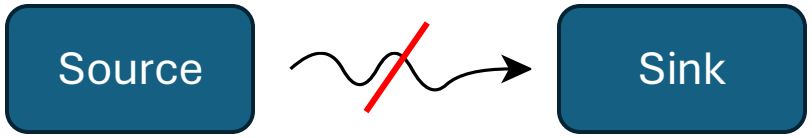
time & control flows

Confidentiality



Security Properties

Information flow properties

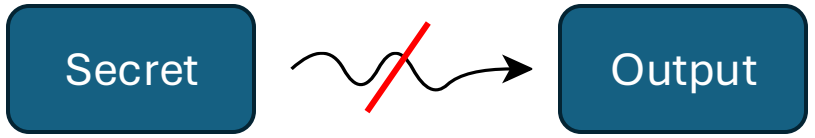


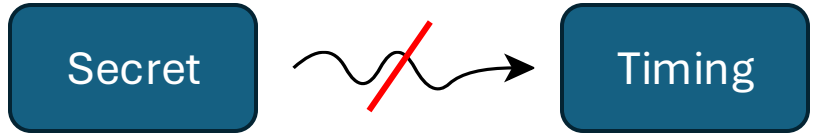
Information =

≠ data flows

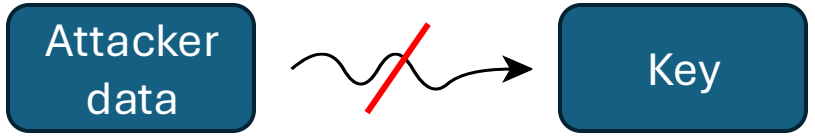
time & control flows

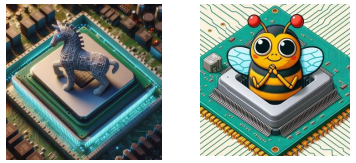
Confidentiality





Integrity



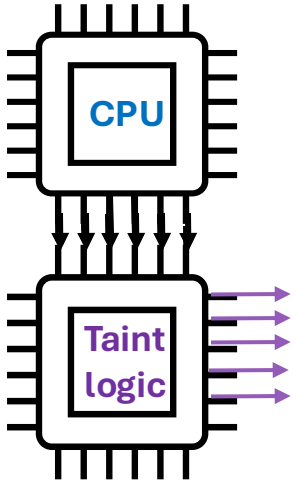


Challenges

- Plethora of information flow tracking methods and tools

Challenges

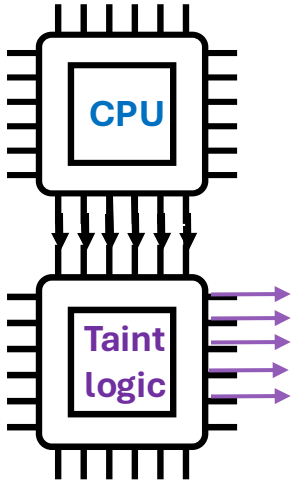
- Plethora of information flow tracking methods and tools



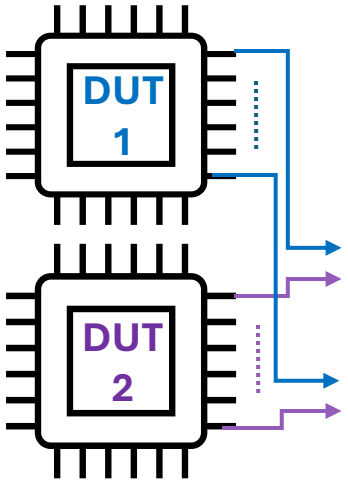
Taint tracking:
GLIFT [8], RTLIFT [9], CellIFT [10], RTL2MμPATH [11], μCFI [12], MileSan [13], …

Challenges

- Plethora of information flow tracking methods and tools

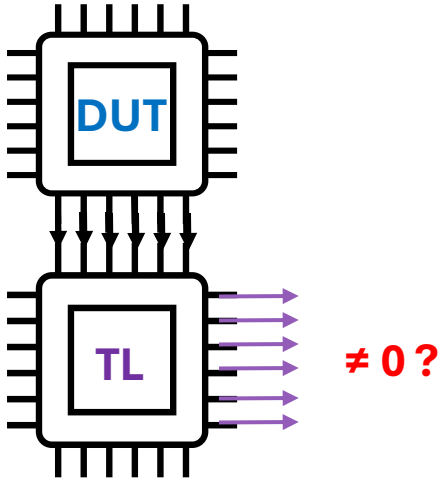


Taint tracking:
GLIFT [8], RTLIFT [9],
CellIFT [10],
RTL2MμPATH [11],
μCFI [12], MileSan [13], …



Self-composition / miter:
UPEC [1-3],
AutoCC [4], LeaVe [5],
H-Houdini [6], SPV [7], …

Verifying Information Flows with **Taint Tracking**

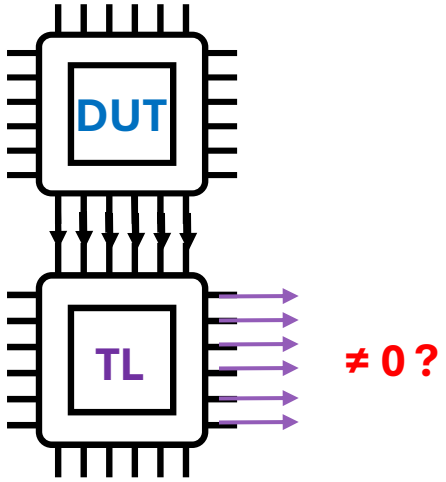


≠ 0 ?

Taint Logic (TL)
tracks information flows

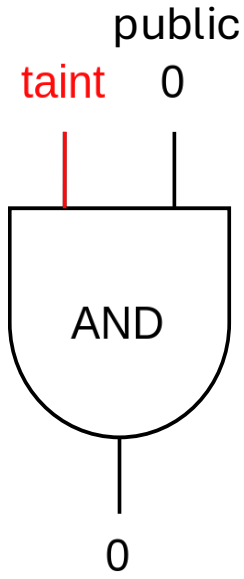
DUT = Design Under Test

Verifying Information Flows with **Taint Tracking**



Taint Logic (TL)
tracks information flows

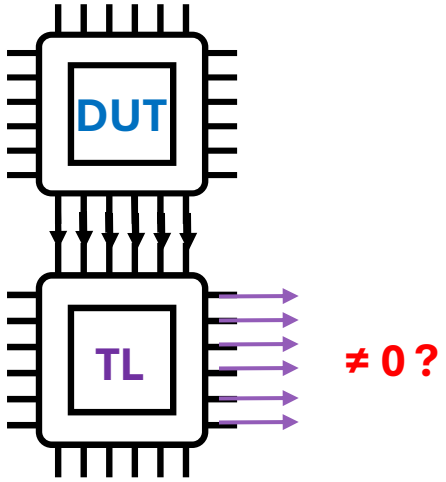
Information flow tracking with taint logic



taint = **secret** or **attacker-controlled**

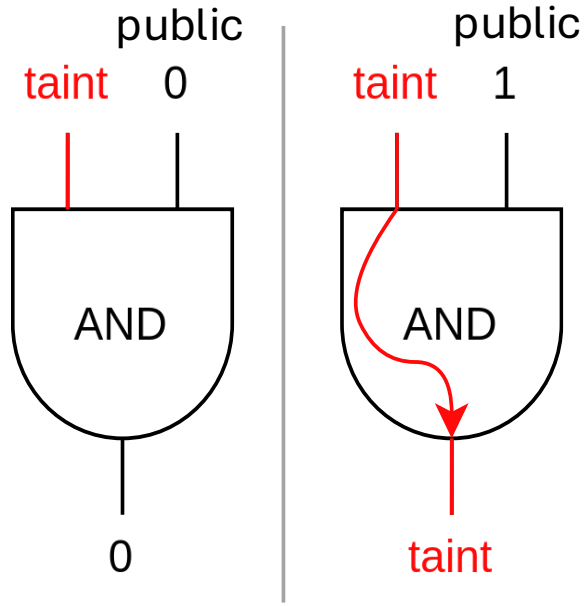
DUT = Design Under Test

Verifying Information Flows with **Taint Tracking**



Taint Logic (TL)
tracks information flows

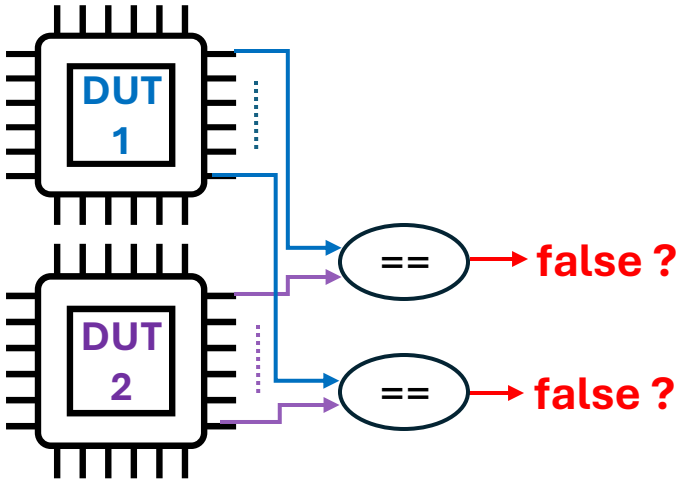
Information flow tracking with taint logic



taint = **secret** or **attacker-controlled**

DUT = Design Under Test

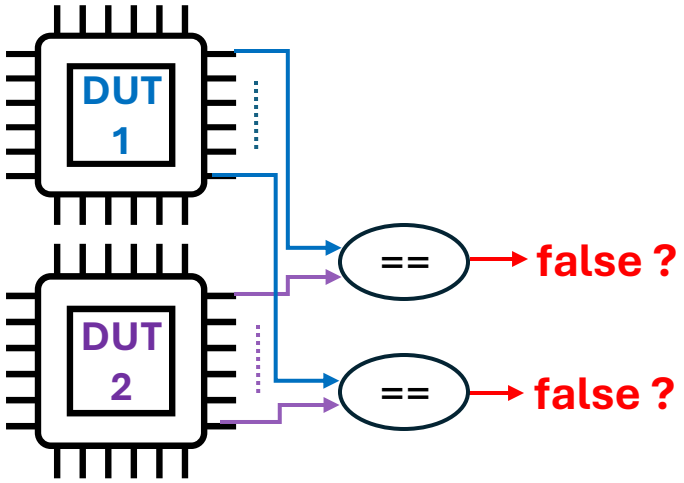
Verifying Information Flows with **Self-composition**



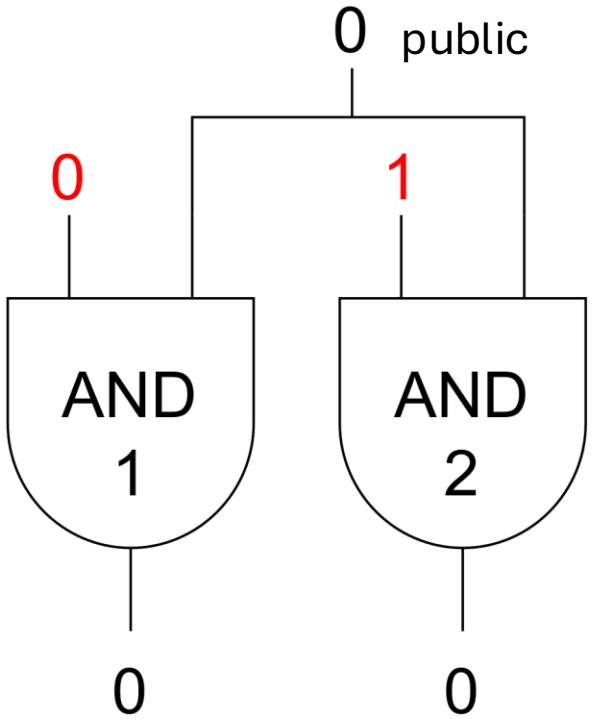
Difference in values
between the two DUT copies
tracks information flows

DUT = Design Under Test

Verifying Information Flows with **Self-composition**



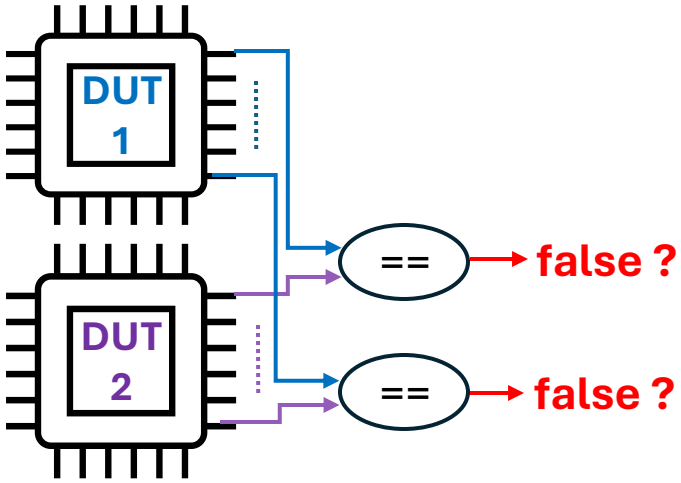
Difference in values
between the two DUT copies
tracks information flows



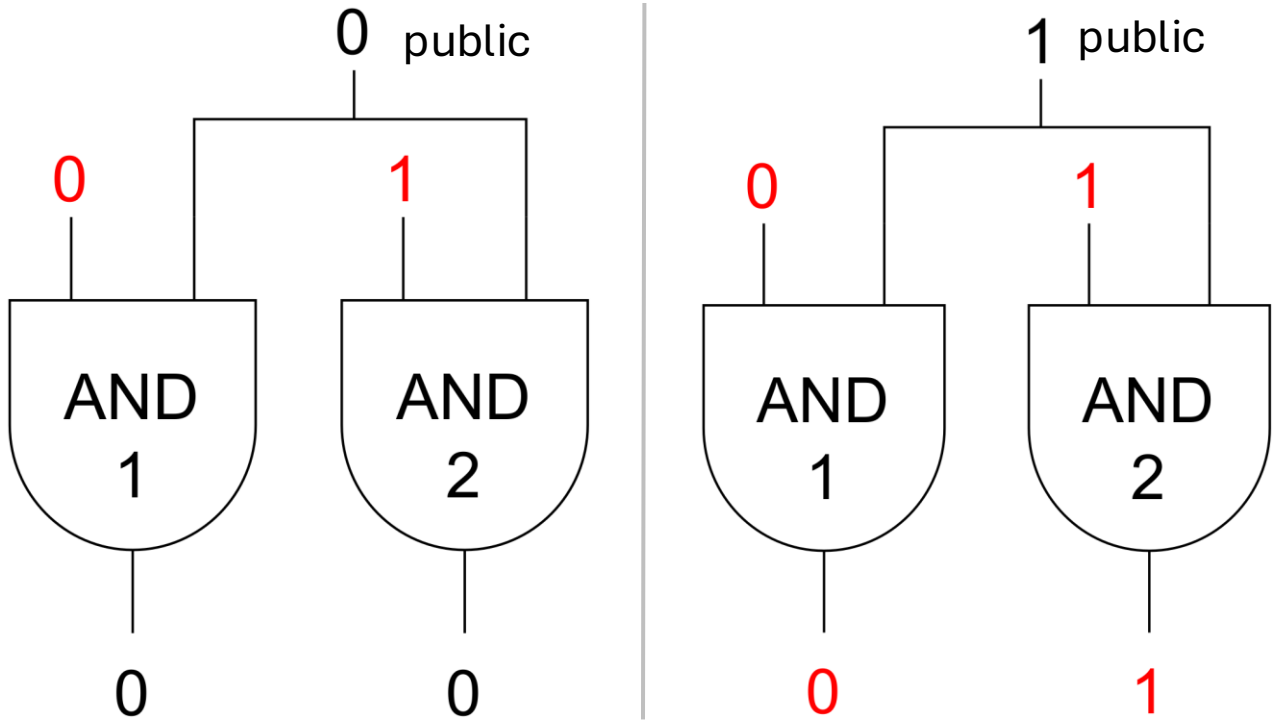
difference = **secret** or **attacker-controlled**

DUT = Design Under Test

Verifying Information Flows with **Self-composition**



Difference in values
between the two DUT copies
tracks information flows

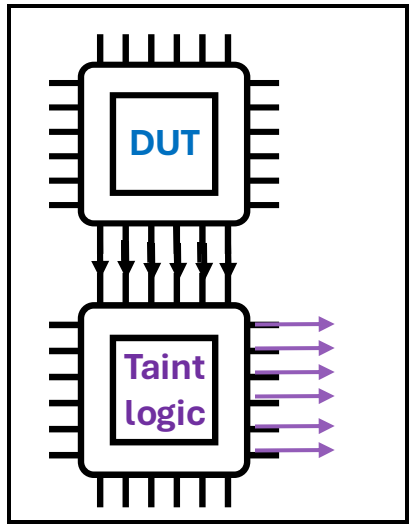


difference = **secret** or **attacker-controlled**

DUT = Design Under Test

Formal Verification of Information Flows

Taint logic

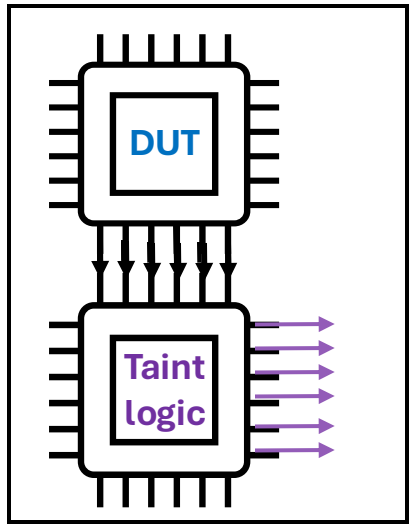


Formal SVA properties

SVA = SystemVerilog Assertions
DUT = Design Under Test

Formal Verification of Information Flows

Taint logic



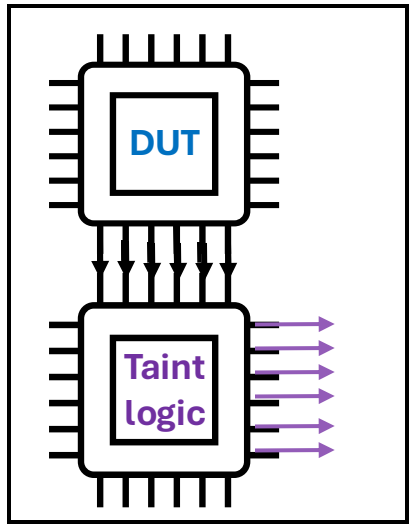
Formal SVA properties

Formal model checker

SVA = SystemVerilog Assertions
DUT = Design Under Test

Formal Verification of Information Flows

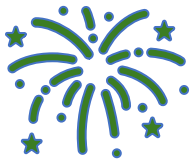
Taint logic



Formal SVA properties

Formal model checker

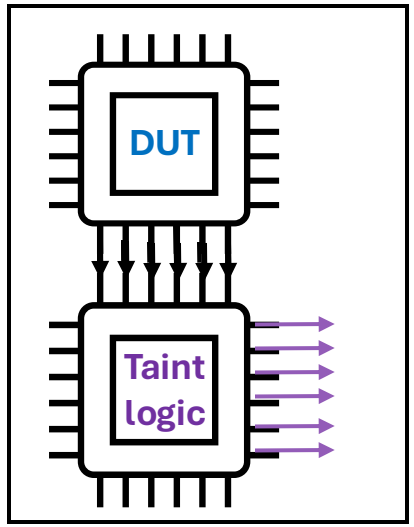
Formal proof



SVA = SystemVerilog Assertions
DUT = Design Under Test

Formal Verification of Information Flows

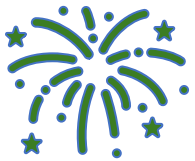
Taint logic



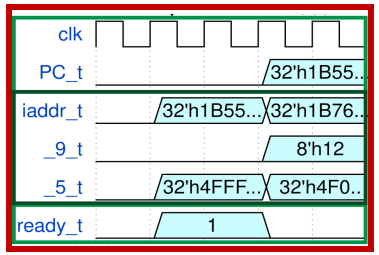
Formal SVA properties

Formal model checker

Formal proof



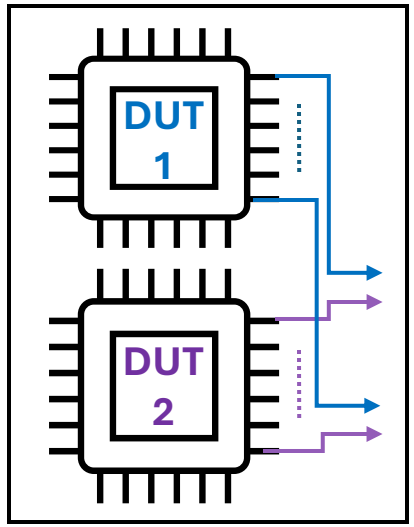
Counter example



Waveform,
e.g., Value Change Dump (VCD)

SVA = SystemVerilog Assertions
DUT = Design Under Test

Formal Verification of Information Flows



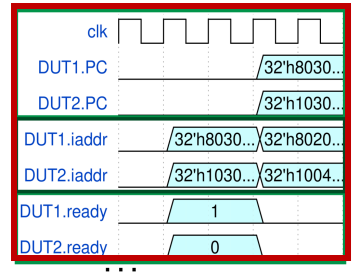
Self-composition / miter

Formal SVA properties

Formal model checker

Formal proof

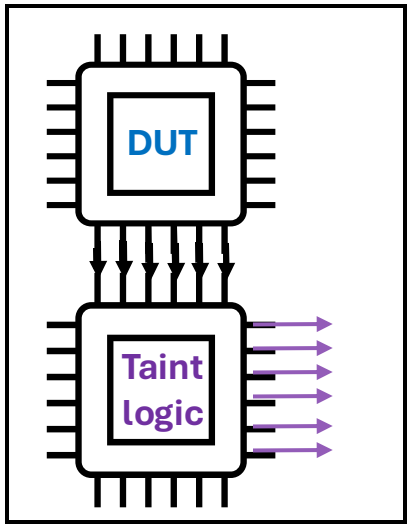
Counter example



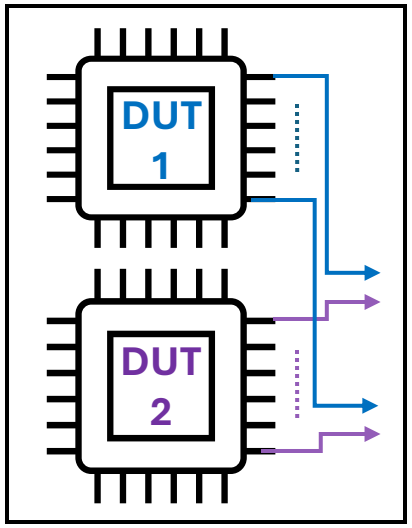
Waveform, e.g., Value Change Dump (VCD)

Simulation of Information Flows

Taint logic

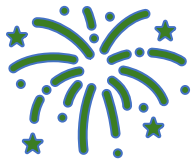


Self-composition / miter

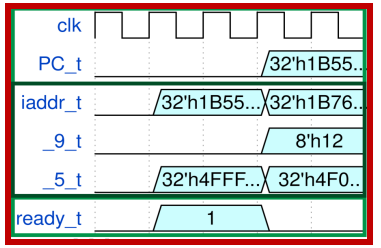


Simulation test benches

Test passes



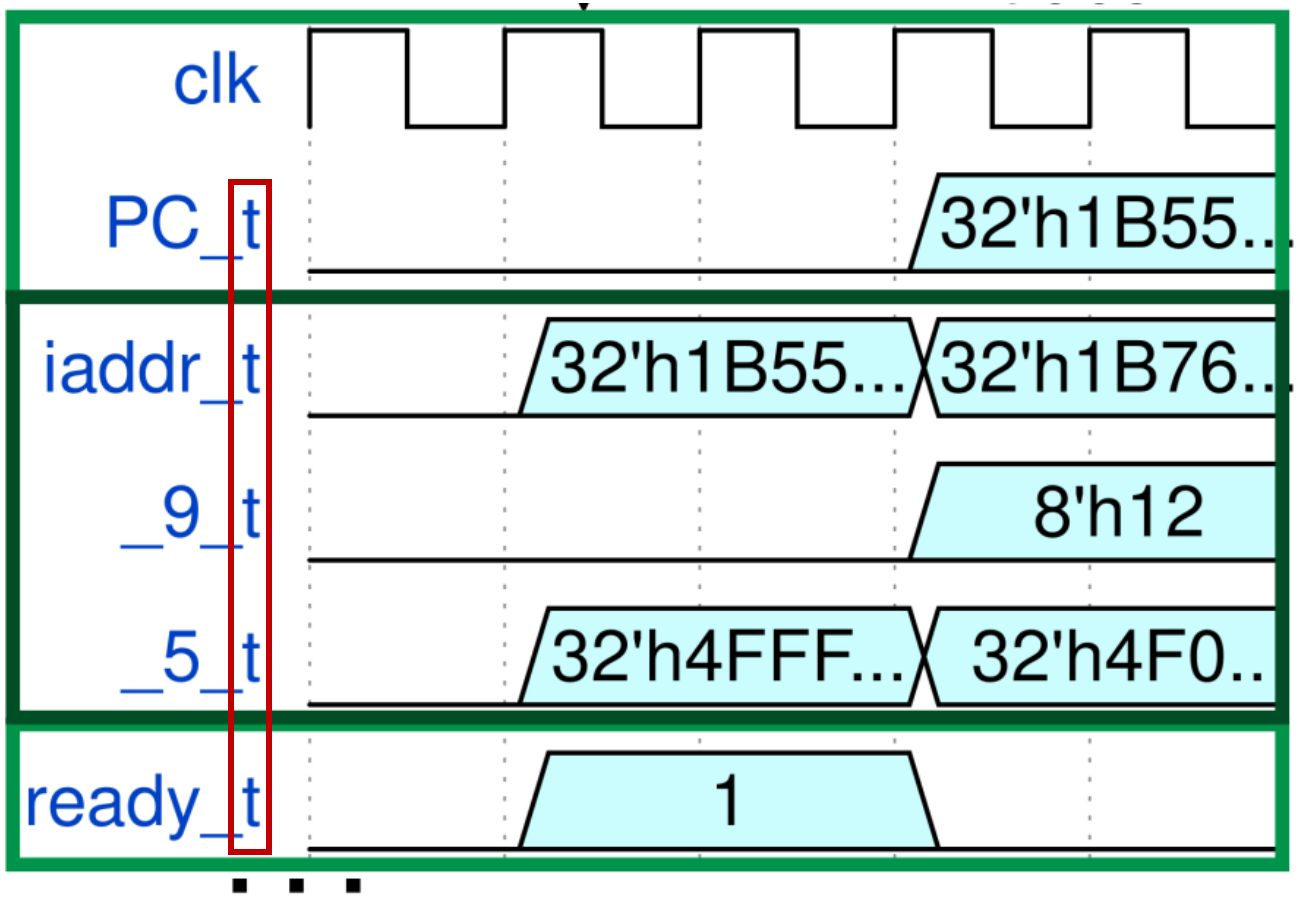
Test fails



Waveform,
e.g., Value Change Dump (VCD)

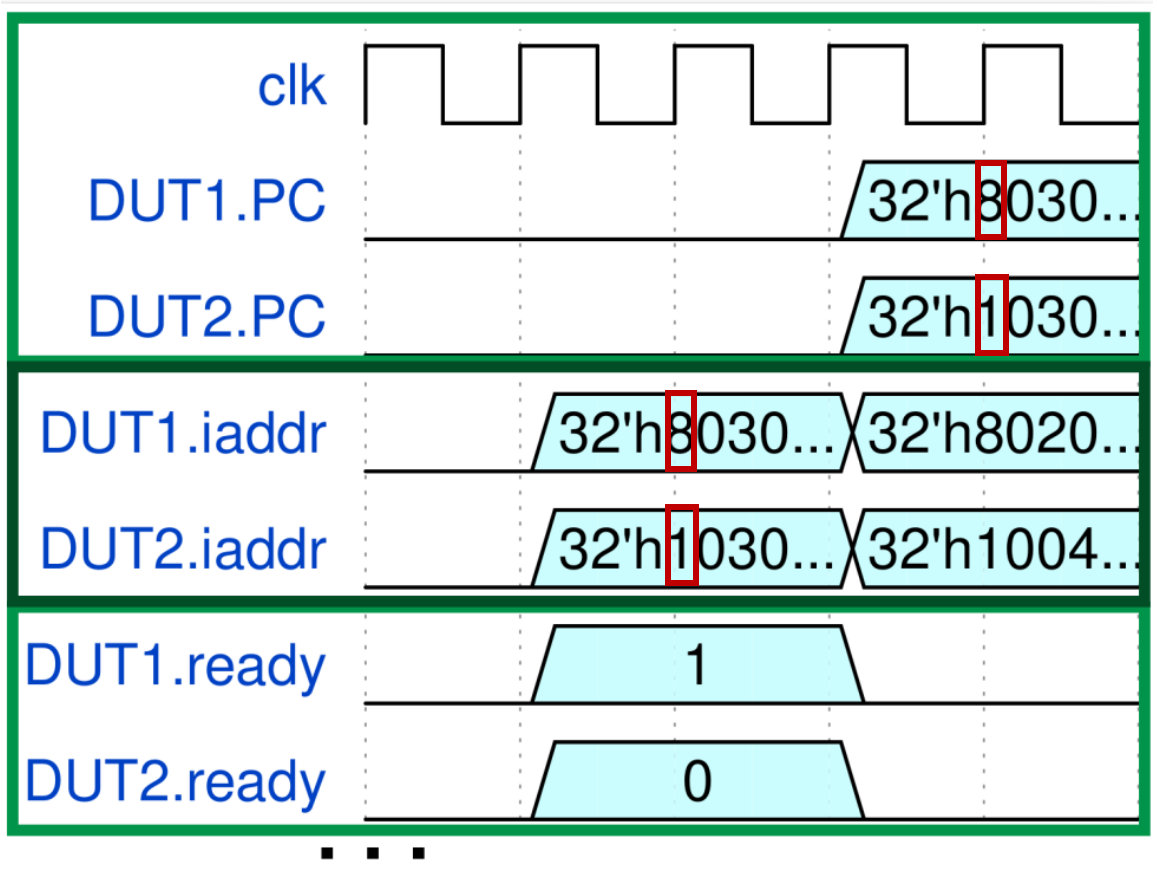
Taint Logic Waveform Debugging

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

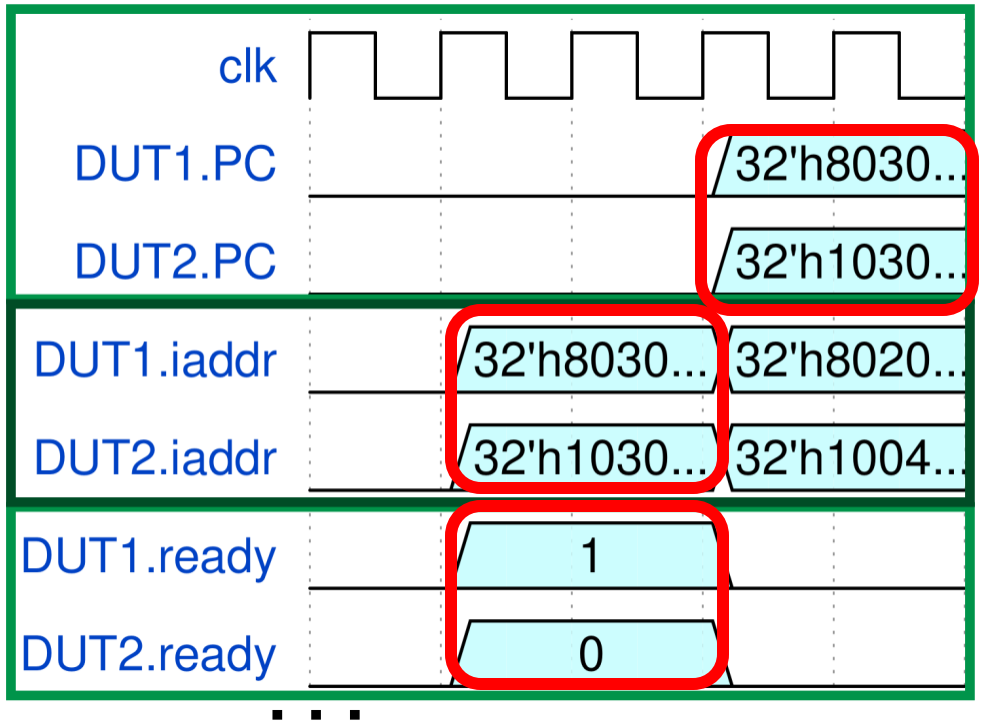


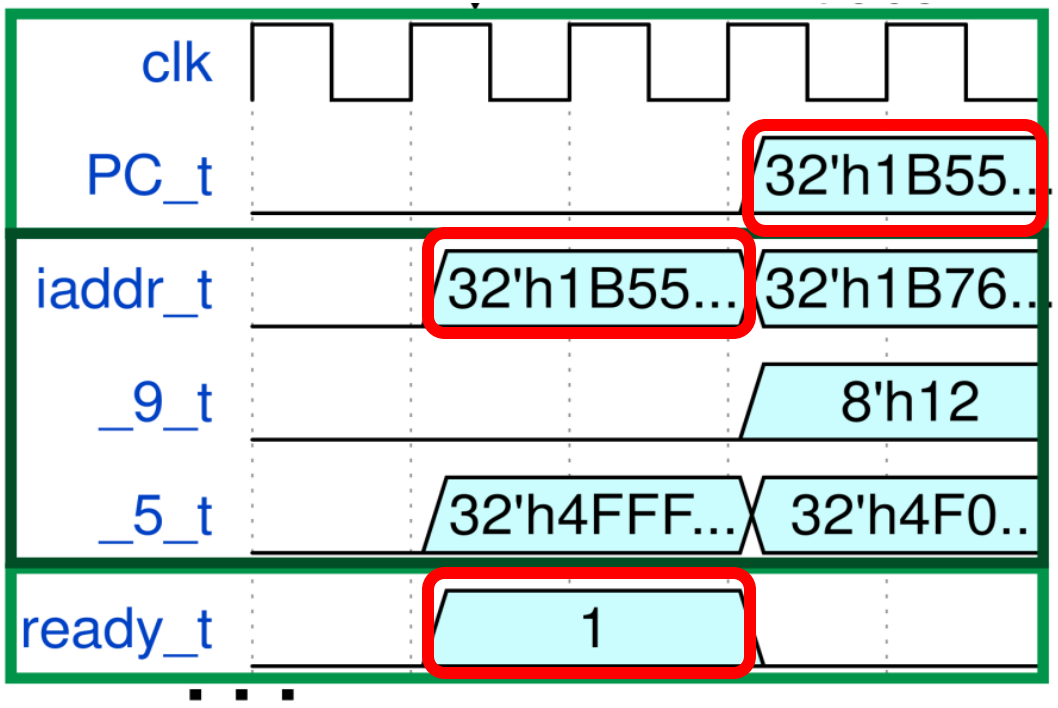
Self-composition Waveform Debugging

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```



Proof of unification of self-composition and taint tracking





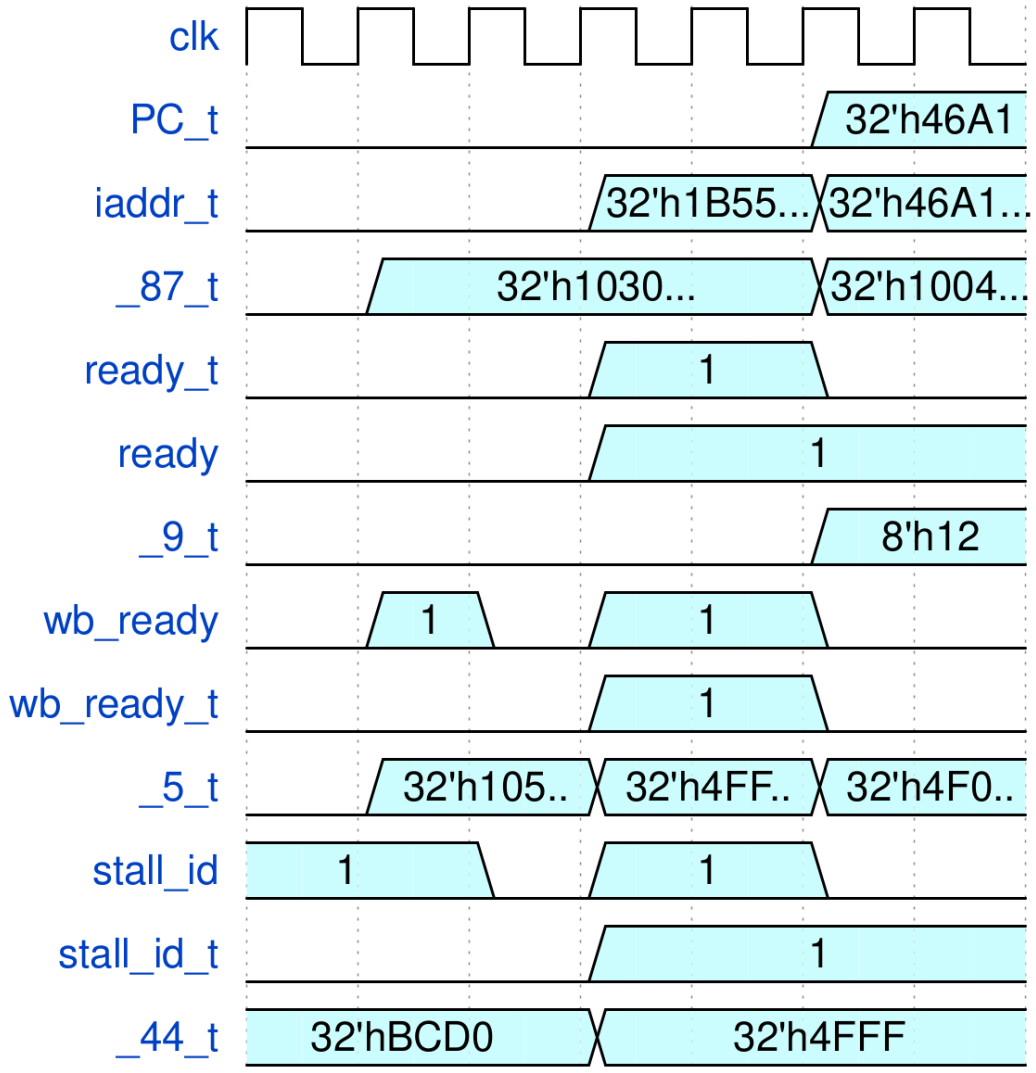
self-composition flow ➡ taint tracking flow

100% precise taint tracking flow ➡ self-composition flow

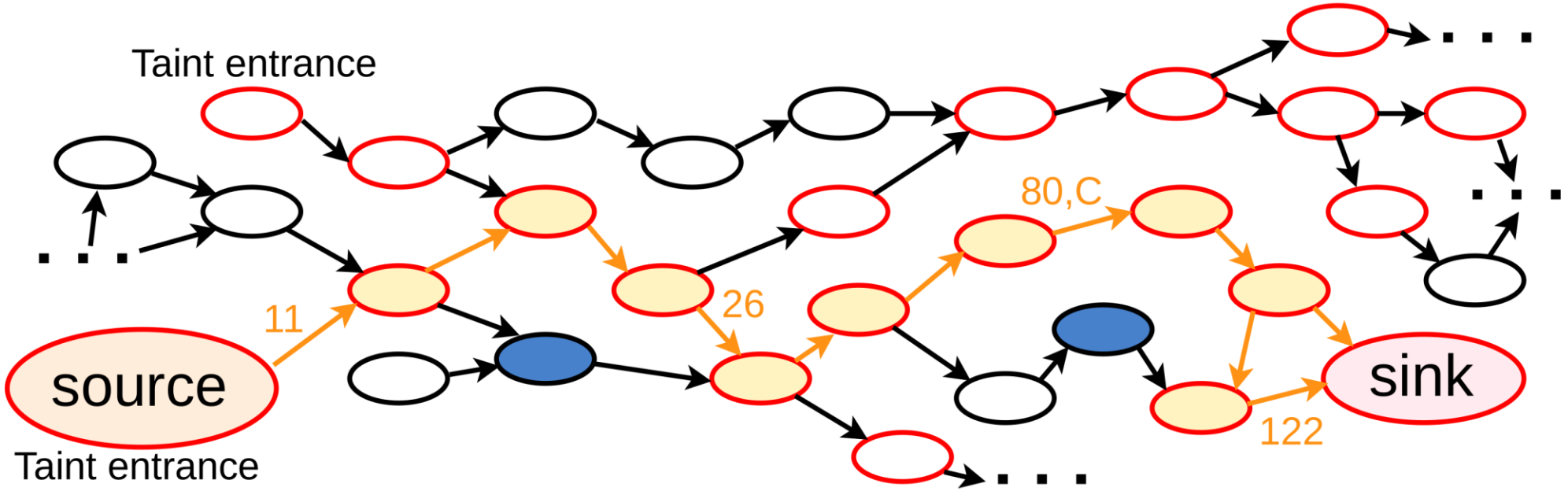
Challenges

- Plethora of information flow tracking methods and tools

- **Functional verification debugging tools show too much information**



Pathfinder

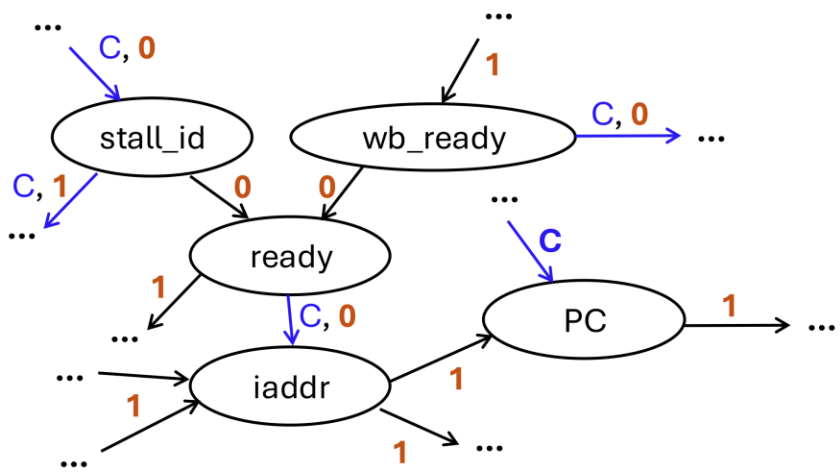


- **Where**, **when** and **why** does information flow?
- Visualizes clock-cycle-accurate propagation of information

Pathfinder - Overview

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

Hardware Description Language (HDL)

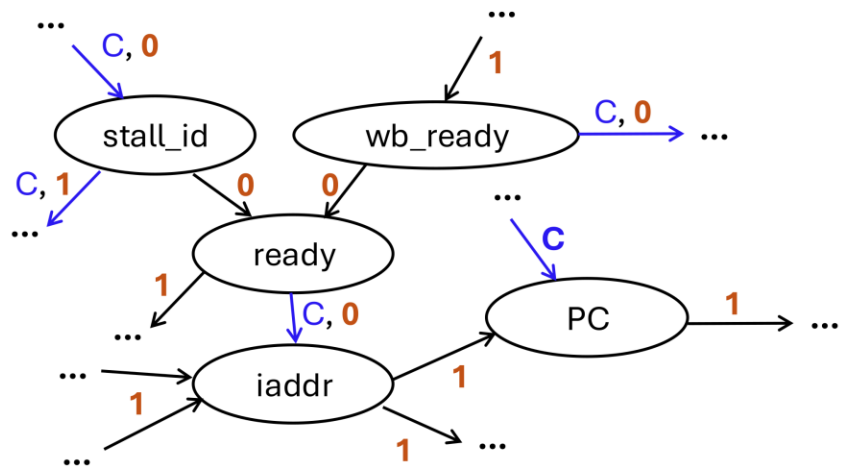


Temporal Information Flow Graph (TIFG)

Pathfinder - Overview

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

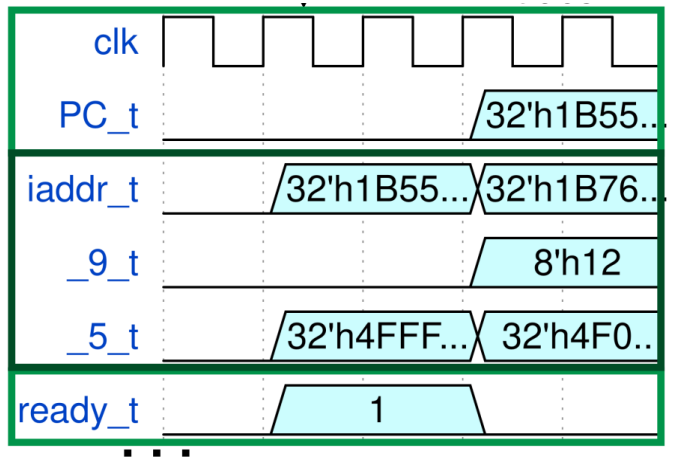
Hardware Description Language (HDL)



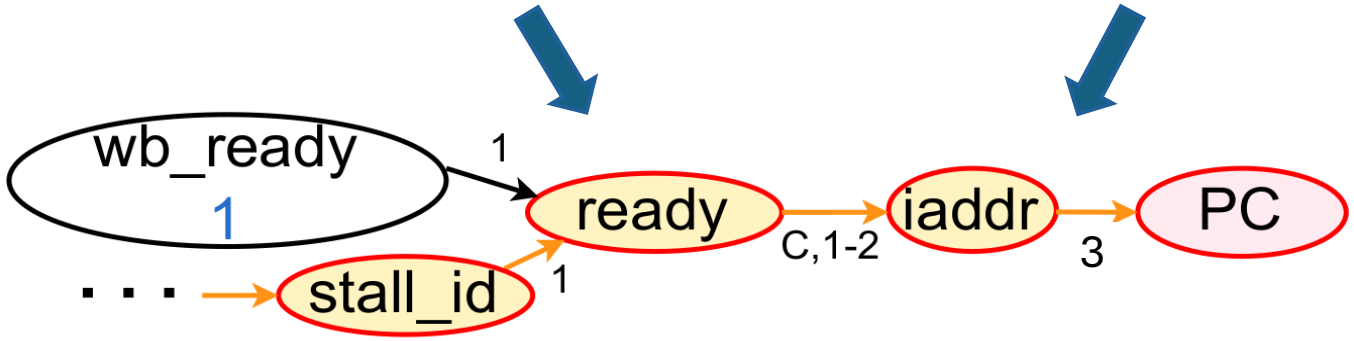
Temporal Information Flow Graph (TIFG)

Formal model checker

Simulation test benches



Waveform (VCD)



Taint Graph (TG)

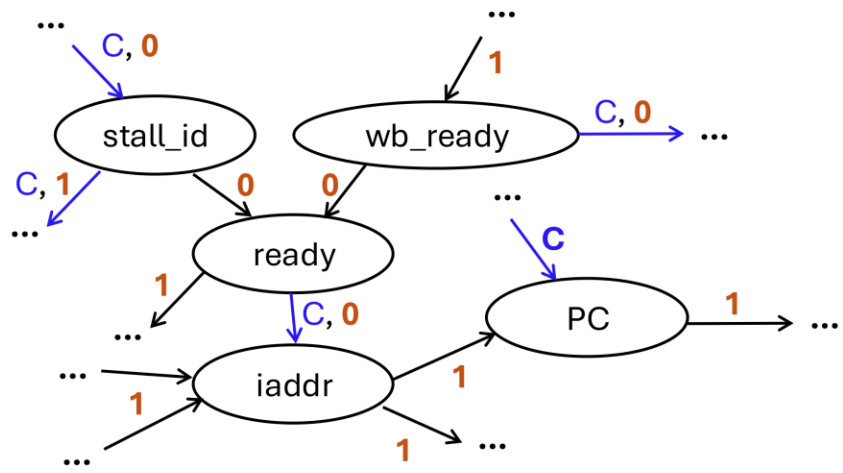
Pathfinder - Overview

Formal model checker

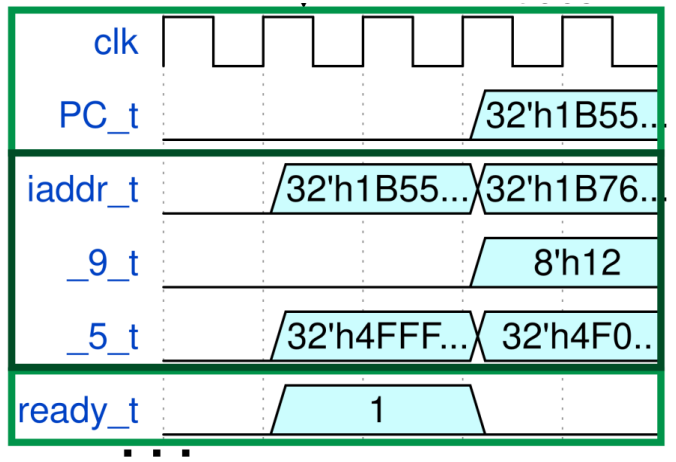
Simulation test benches

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

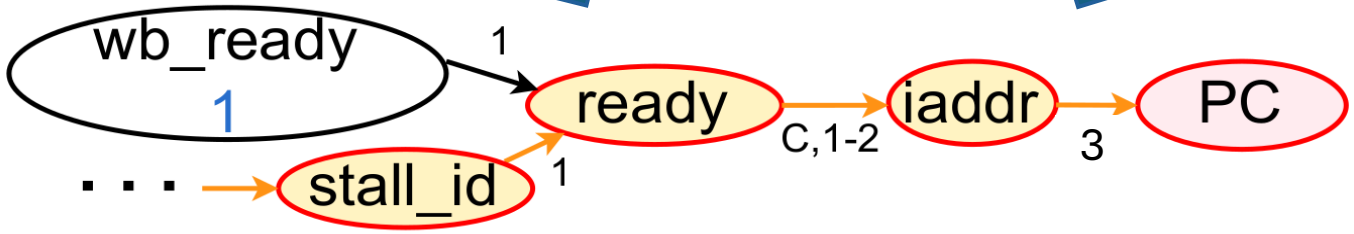
Hardware Description Language (HDL)



Temporal Information Flow Graph (TIFG)



Waveform (VCD)



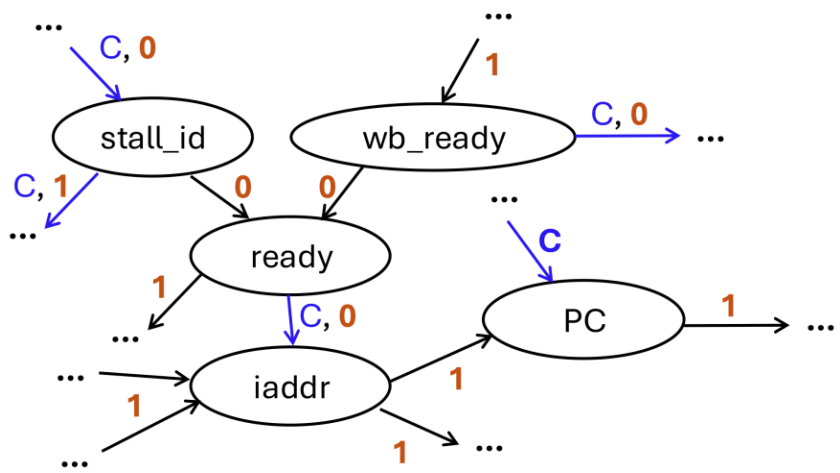
Taint Graph (TG)

Pathfinder - TIFG

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

Hardware Description Language (HDL)

TIFG Yosys pass

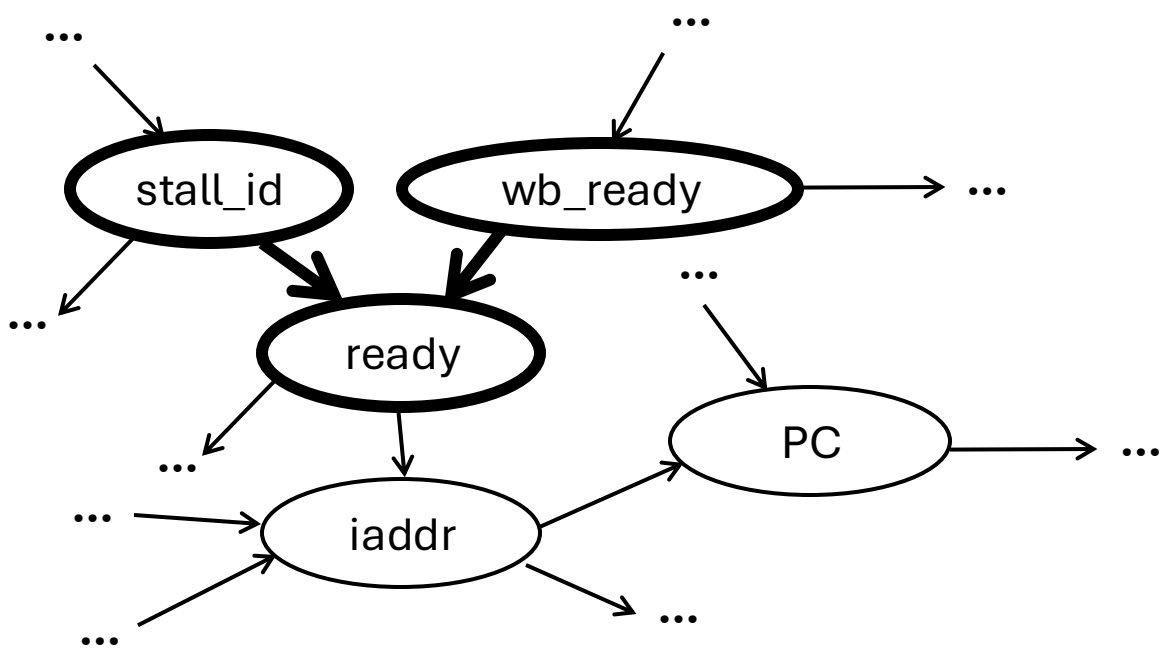


Temporal Information Flow Graph (TIFG)

[1] Yosys Open SYnthesis Suite - https://github.com/YosysHQ/yosys

Temporal Information Flow Graph (TIFG)

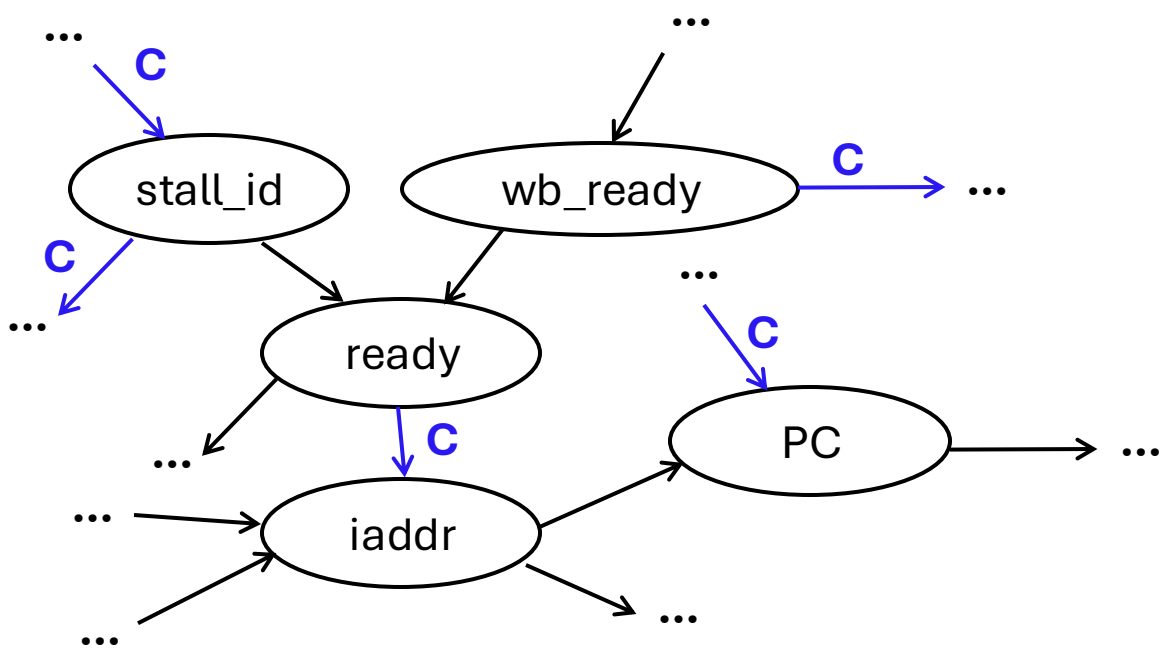
```
assign ready =
    ~stall_id & wb_ready;

assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```



encodes **potential information flows**

Temporal Information Flow Graph (TIFG)

```
assign ready =
    ~stall_id & wb_ready;

assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```



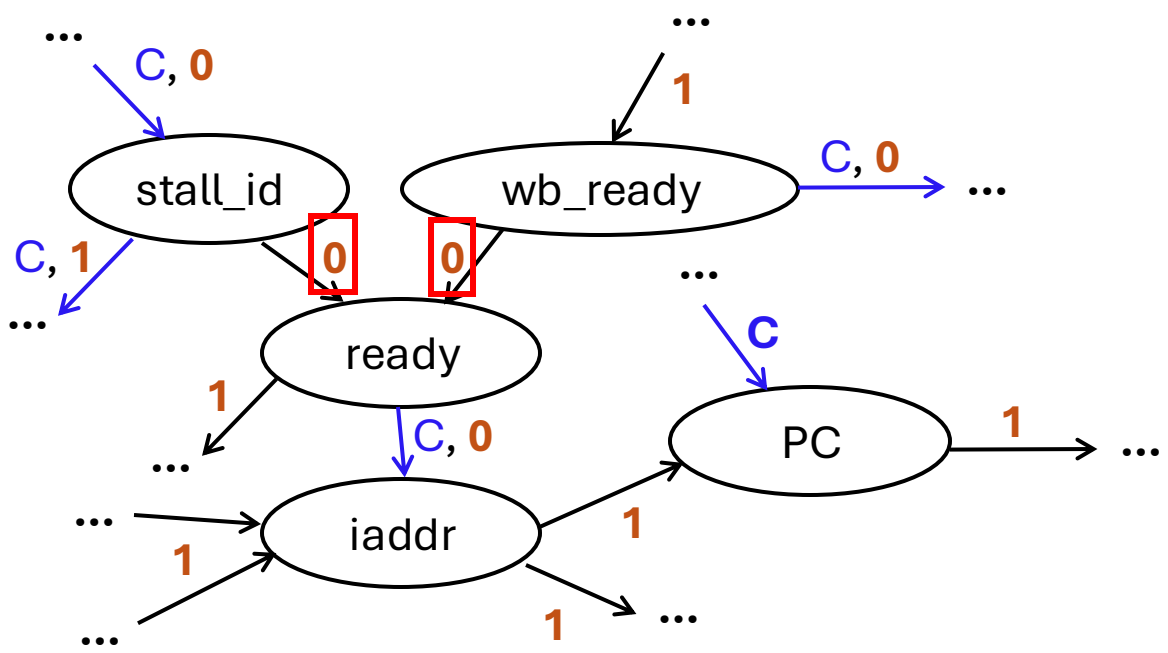
encodes **potential information flows**

C (control) = **implicit** information flow

Temporal Information Flow Graph (TIFG)

```
assign ready =
    ~stall_id & wb_ready;
```

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```



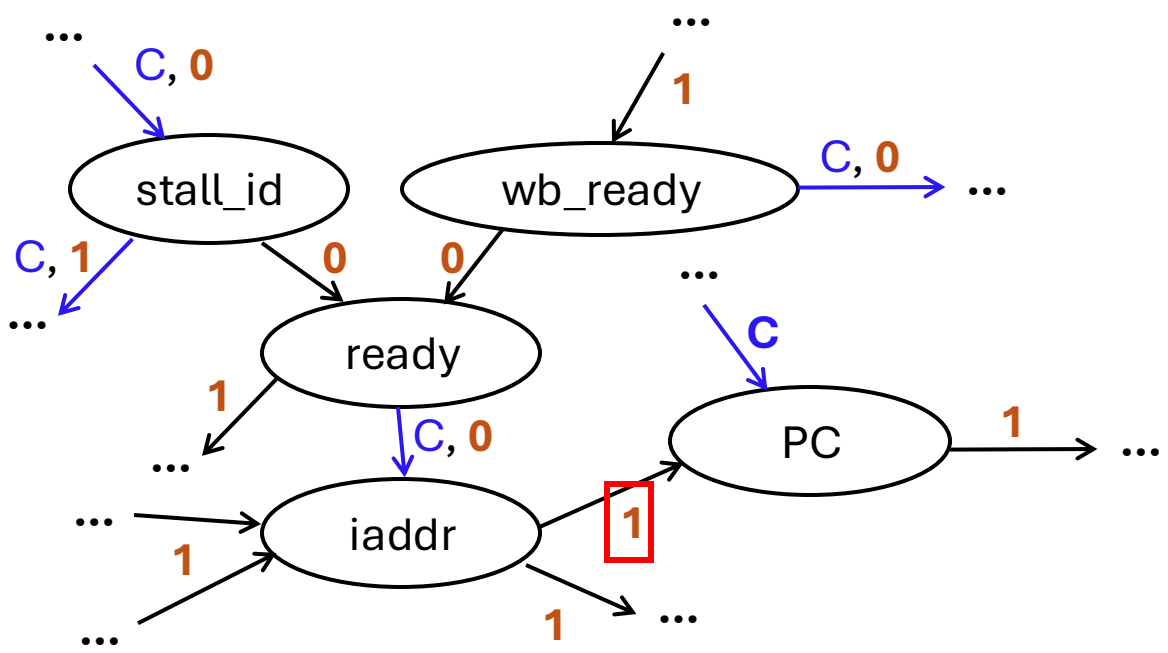
encodes **potential information flows**

C (control) = **implicit** information flow

Nr. = clock cycle **delay**

Temporal Information Flow Graph (TIFG)

```
assign ready =
    ~stall_id & wb_ready;

assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```



encodes **potential information flows**

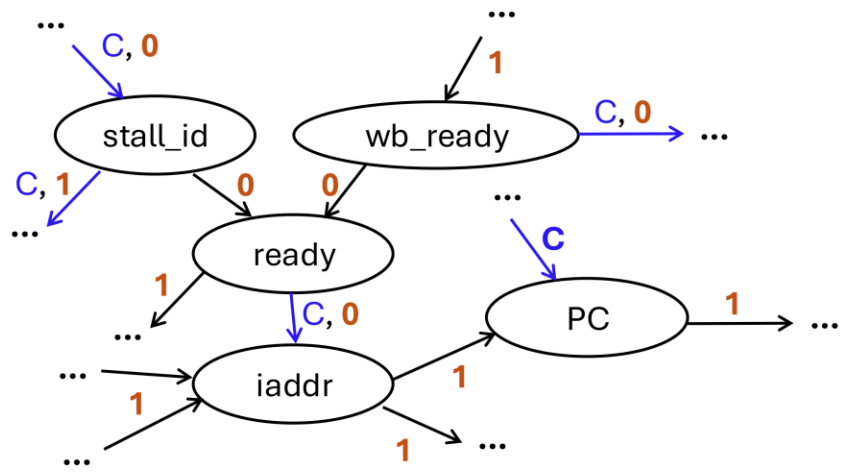
C (control) = **implicit** information flow

Nr. = clock cycle **delay**

Pathfinder - Overview

```
assign iaddr =
    ready ? ...  : ... ;

always (@posedge clk)
   if (...)
    PC <= iaddr;
```

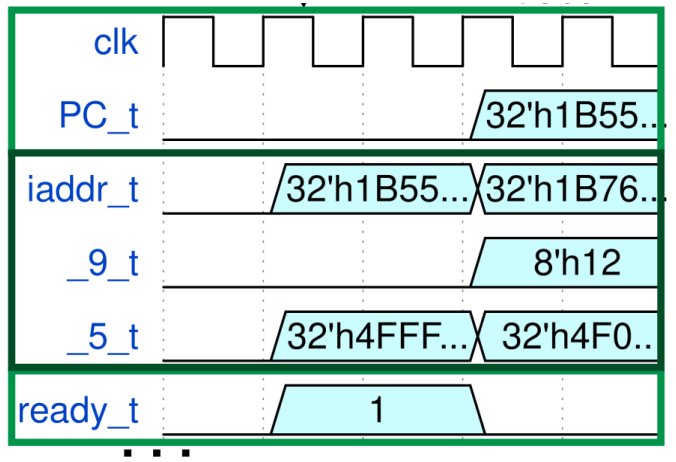
Hardware Description Language (HDL)



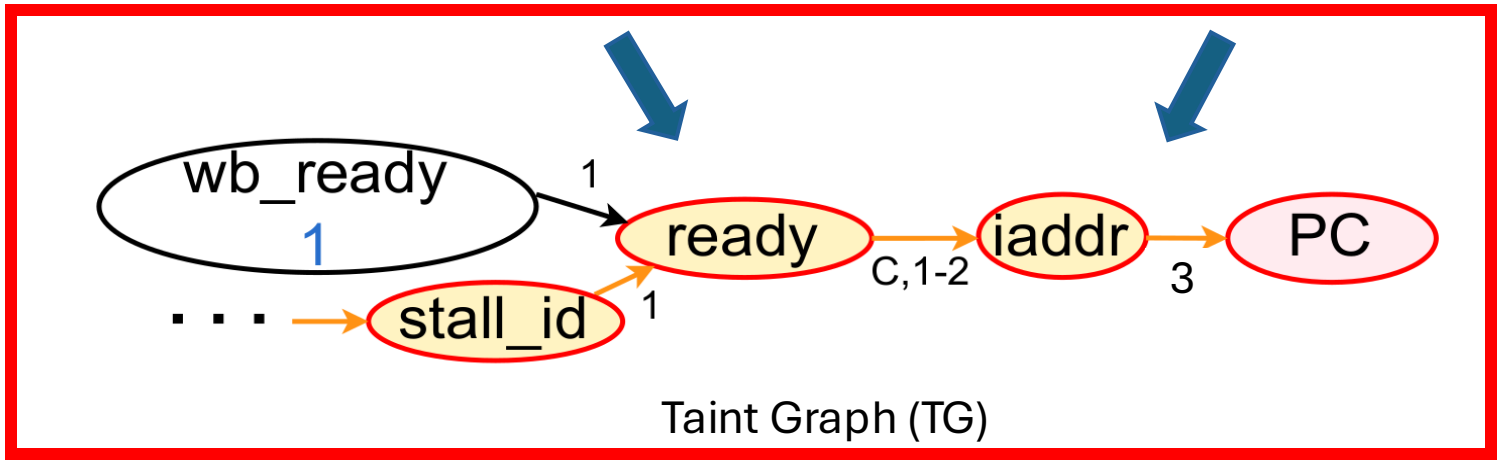
Temporal Information Flow Graph (TIFG)

Formal model checker

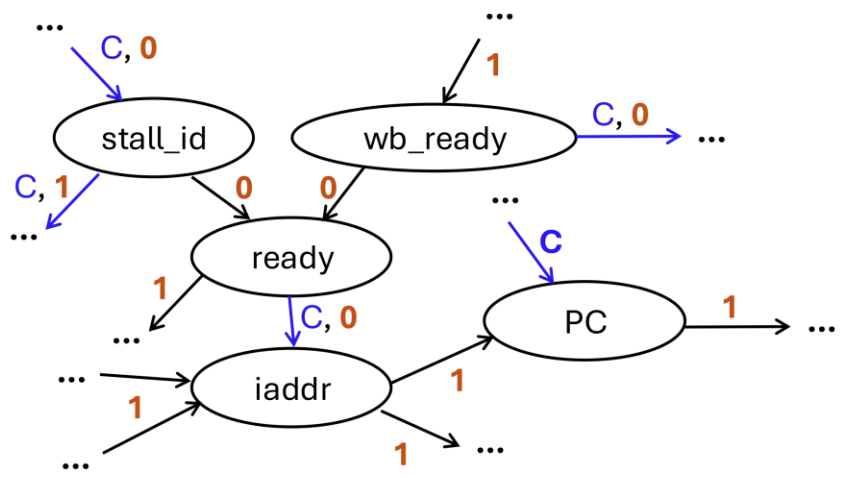
Simulation test benches



Waveform (VCD)



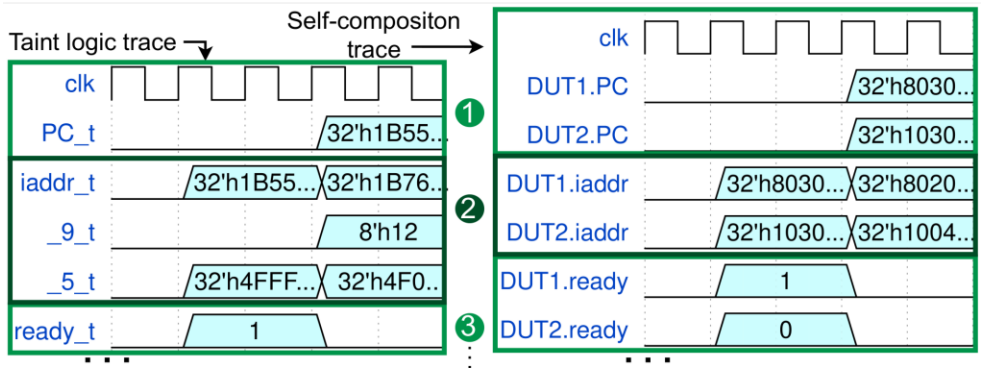
Taint Graph (TG)



Temporal Information Flow Graph (TIFG)

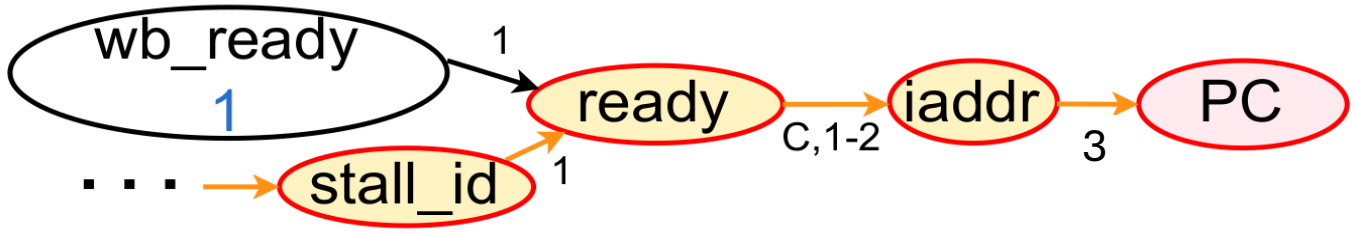
Formal model checker

Simulation test benches

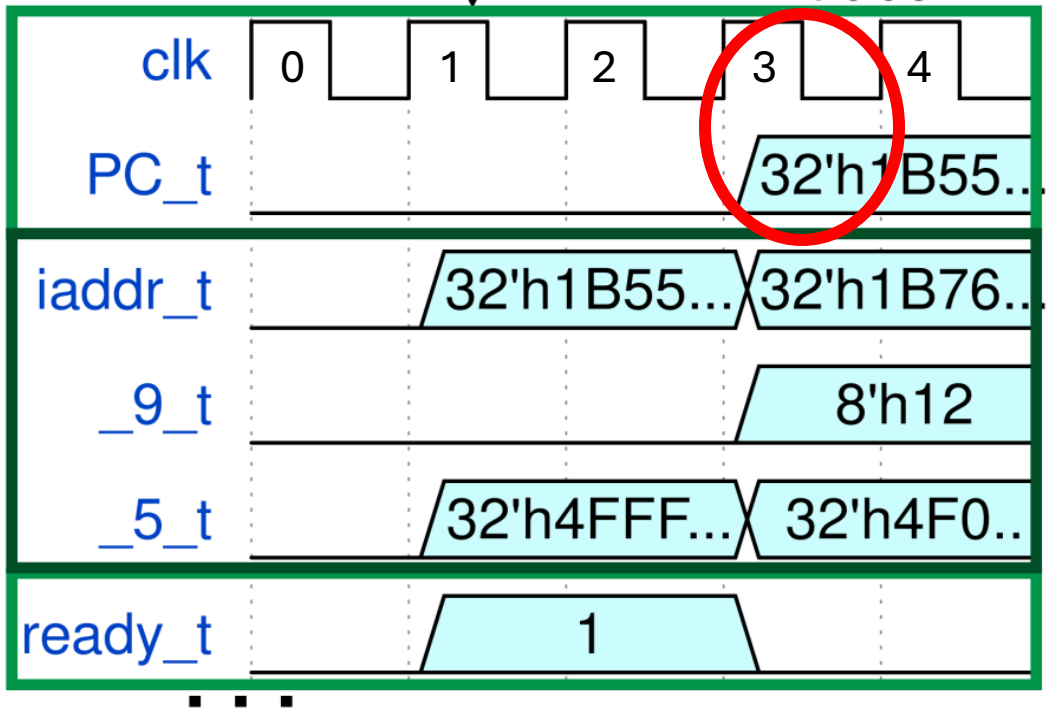


Pathfinder derives **taint** values from **differences** in self-composed design

Pathfinder



Taint Graph (TG)



Pathfinder

Taint Graph construction

Input: waveform, TIFG, taint source/sinks

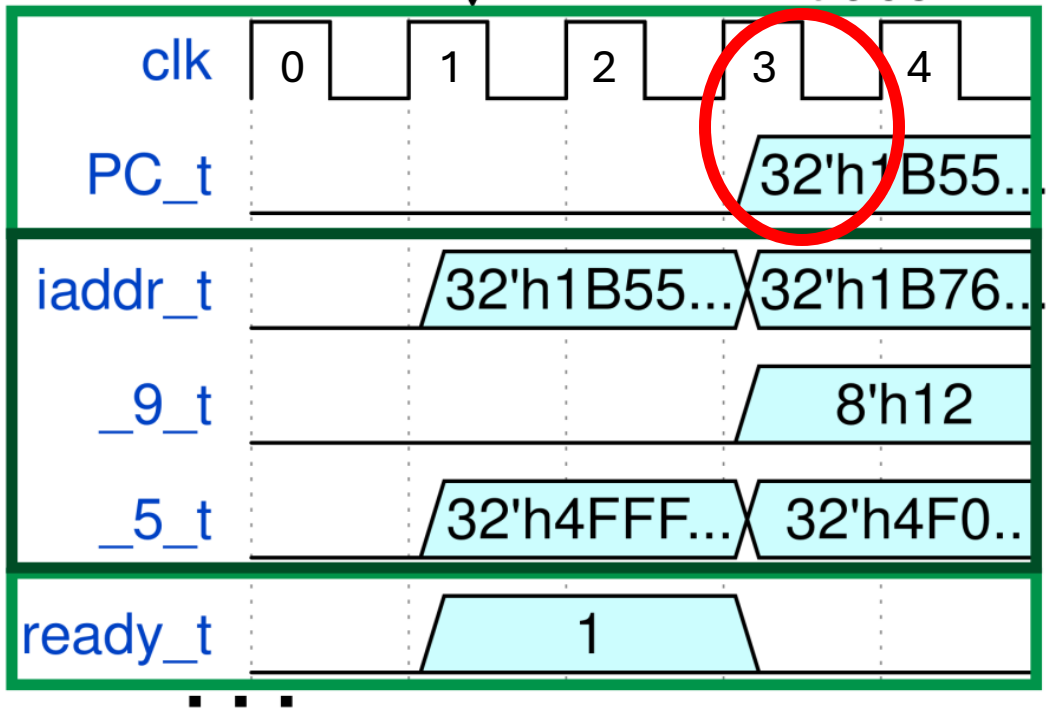
taint sink

PC

Signal 'PC' is first tainted in clock cycle 3

Temporal Information Flow Graph (TIFG)

Pathfinder



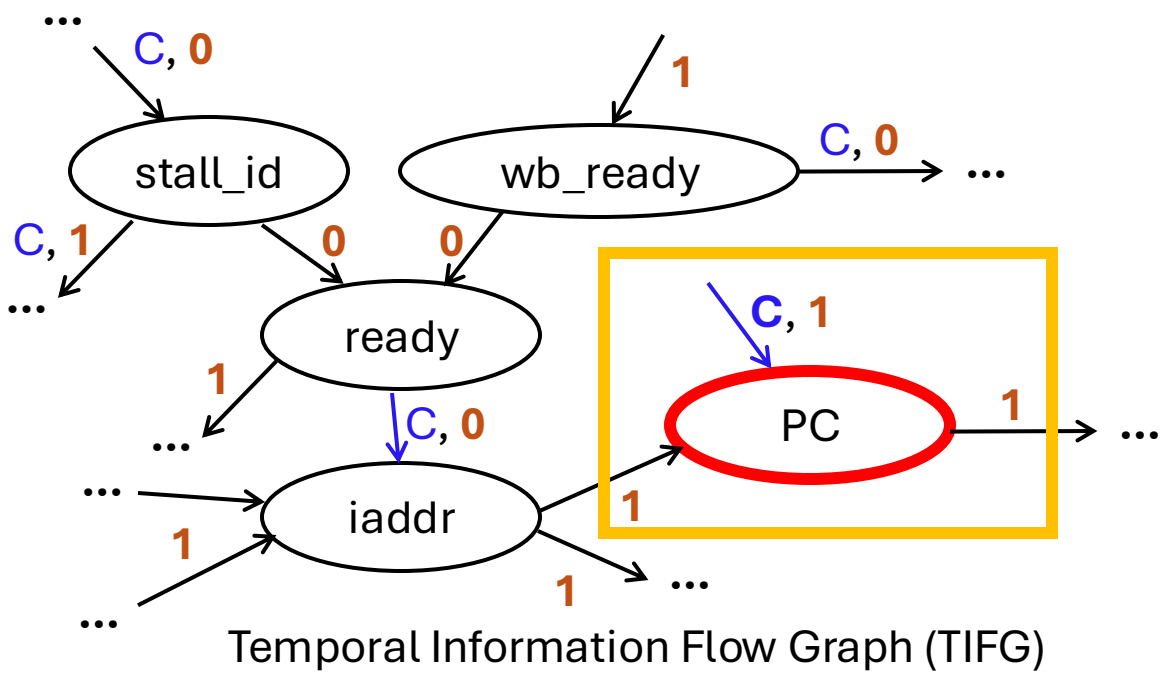
Taint Graph construction

Drivers with zero delay?

No.

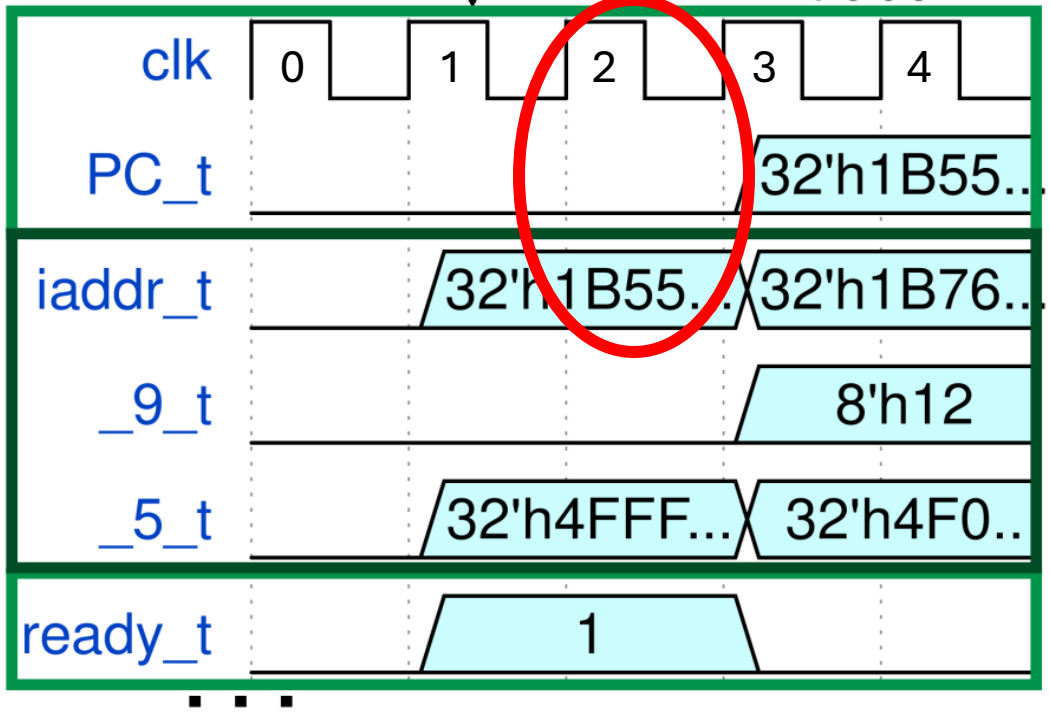
taint sink

PC



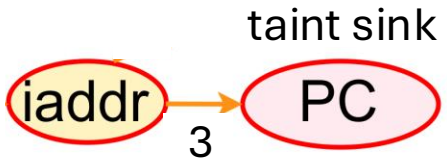
Temporal Information Flow Graph (TIFG)

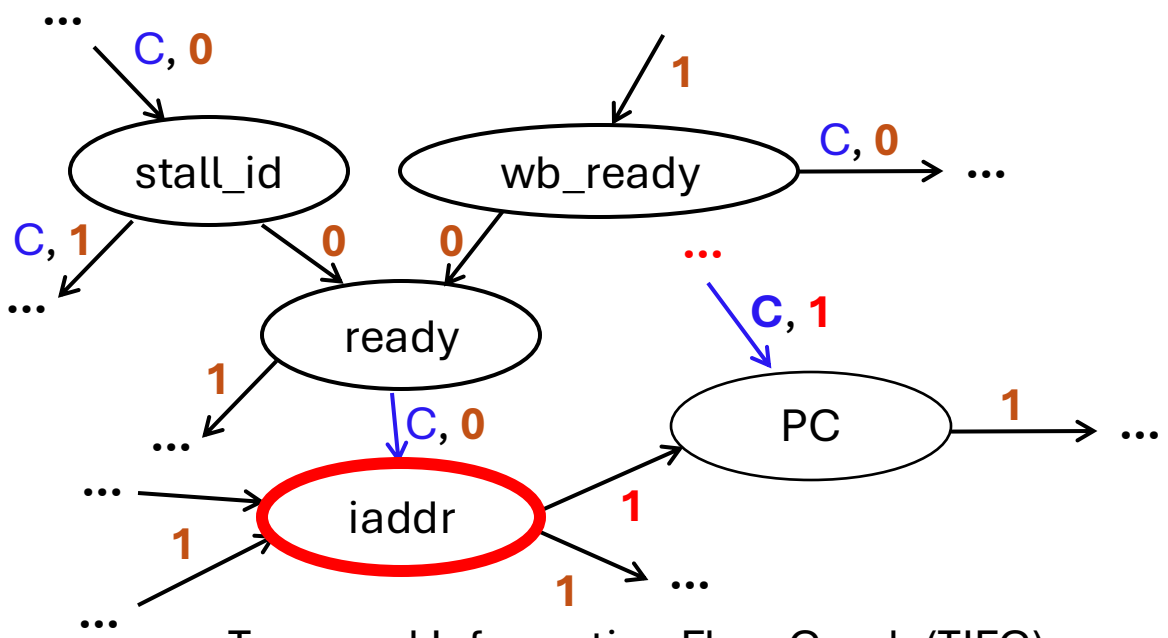
Pathfinder



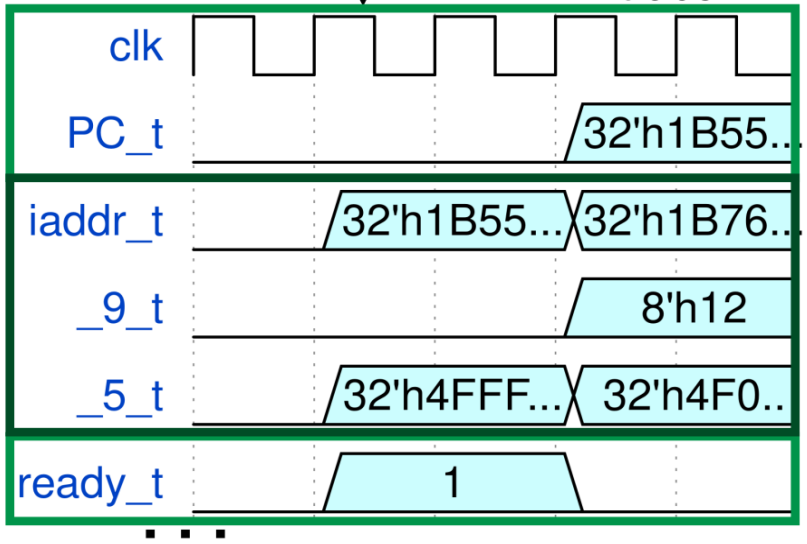
Taint Graph construction

Drivers with 1 cycle delay?
Yes.



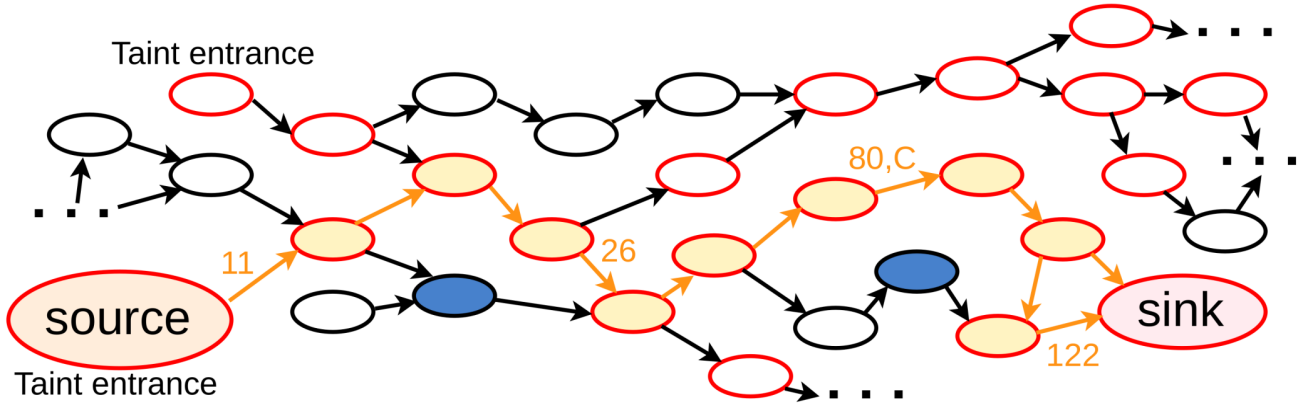


Temporal Information Flow Graph (TIFG)

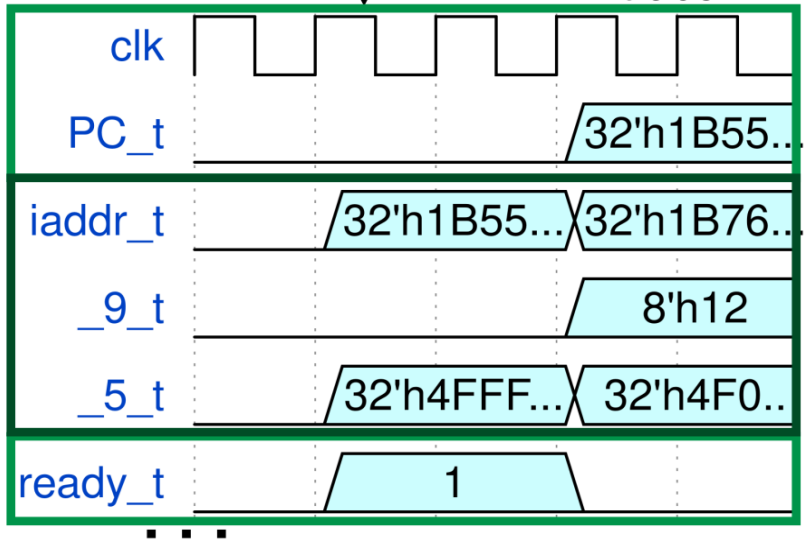


Taint Graph construction

Repeat from source, intersect

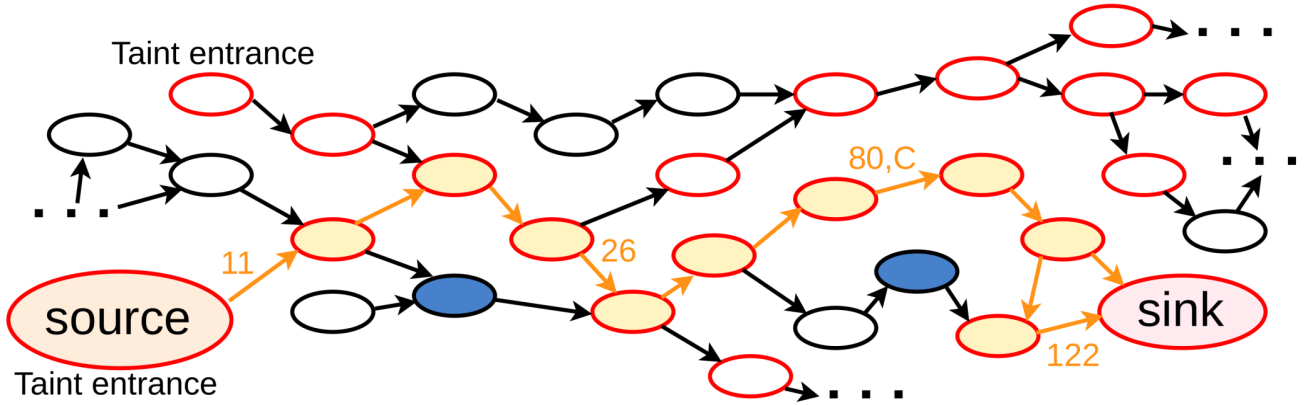


Pathfinder



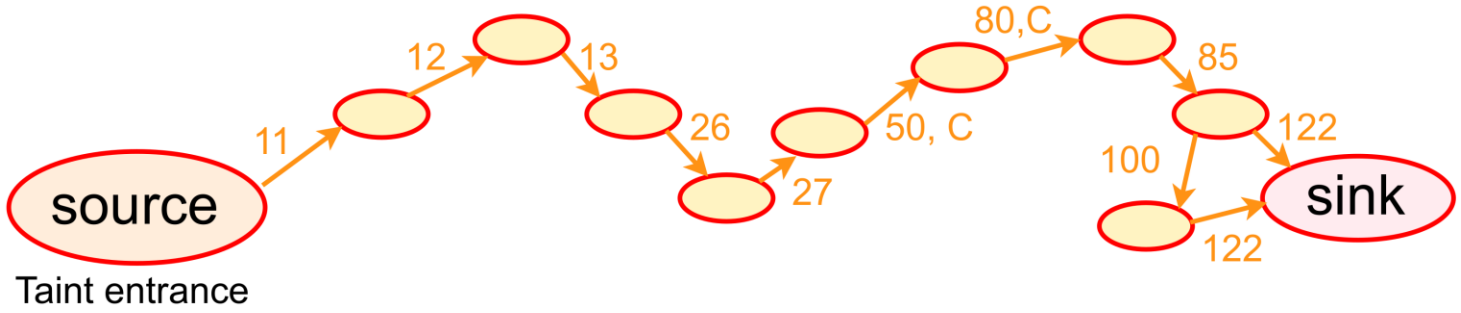
Taint Graph construction

Repeat from source, intersect



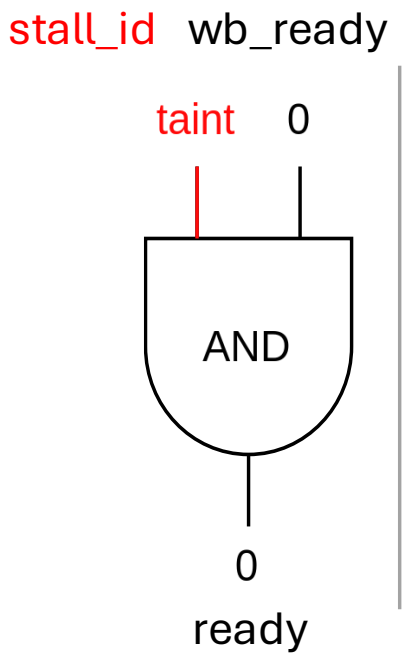
Isolated Taint Graph

Where/when did information flow?



Flow Control Signals

```
assign ready =
    ~stall_id & wb_ready;
```



Why did information flow?

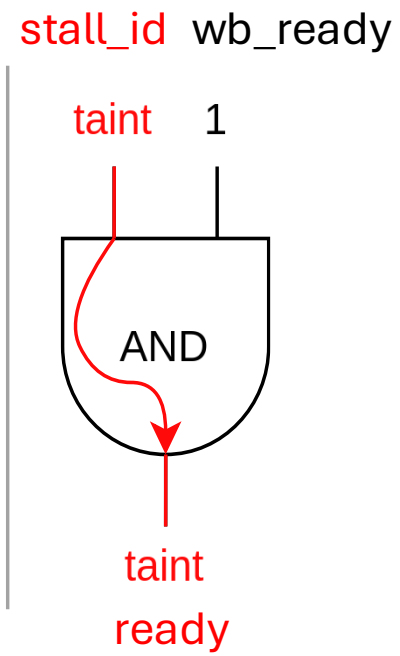
Flow control signal: wb_ready

Flow control condition (wb_ready == 0) blocks information flow

3

Flow Control Signals

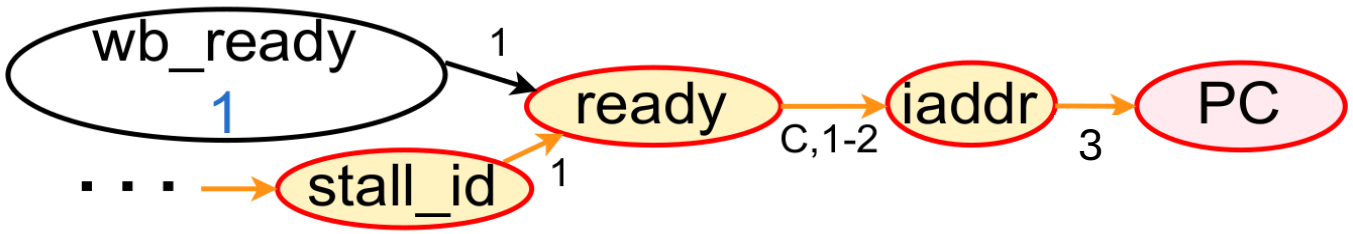
```
assign ready =
    ~stall_id & wb_ready;
```



Why did information flow?

Flow control signal: wb_ready

Flow control condition (wb_ready == 0) is false



Results

Design	IFT	Analysis type	Graph type	Tainted signals	Signals on path	Reduct. factor	Clock cycles	Gen. time [s]
AES-400 [15]	TT	Simulation	Path	15	3	5.0	1	6.03
Kronos [5]	TT	Formal	Path	48	19	2.5	9	0.20
Ibex custom [5]	TT	Formal	Path	330	76	4.3	18	0.59
Ibex small [5]	TT	Formal	Path	164	89	1.8	12	0.78
CVA6 (new trace)	TT	Simulation	Path	82,920	1187	769.9	144	604.0
CVA6 [19]	SC	Formal	Bottom	900	562	1.6	76	4.47
BOOM [17]	TT	Simulation	Path	17,316	213	81.3	988	100.93

Pathfinder supports waves of

- self-composition
- taint tracking
- simulation
- formal verification

IFT = Information Flow Tracking, TT = Taint Tracking, SC = Self-Composition

Results

Design	IFT	Analysis type	Graph type	Tainted signals	Signals on path	Reduct. factor	Clock cycles	Gen. time [s]
AES-400 [15]	TT	Simulation	Path	15	3	5.0	1	6.03
Kronos [5]	TT	Formal	Path	48	19	2.5	9	0.20
Ibex custom [5]	TT	Formal	Path	330	76	4.3	18	0.59
Ibex small [5]	TT	Formal	Path	164	89	1.8	12	0.78
CVA6 (new trace)	TT	Simulation	Path	82,920	1187	769.9	144	604.0
CVA6 [19]	SC	Formal	Bottom	900	562	1.6	76	4.47
BOOM [17]	TT	Simulation	Path	17,316	213	81.3	988	100.93

Number of signals that need to be manually analyzed is reduced by up to ~700x

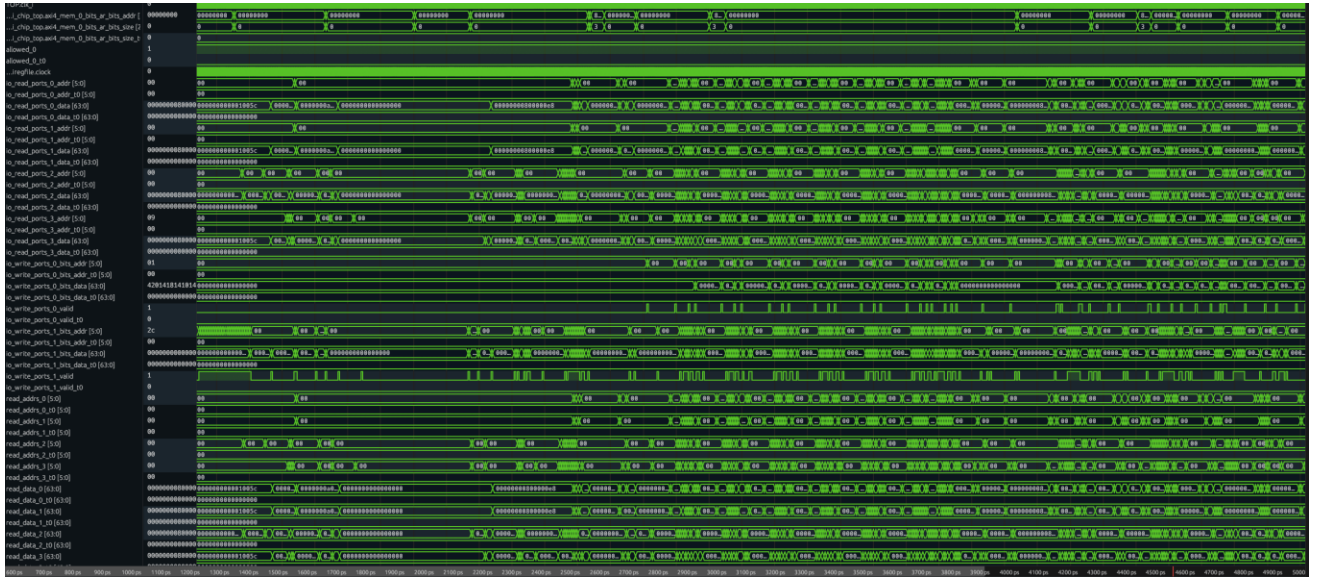
Results

Design	IFT	Analysis type	Graph type	Tainted signals	Signals on path	Reduct. factor	Clock cycles	Gen. time [s]
AES-400 [15]	TT	Simulation	Path	15	3	5.0	1	6.03
Kronos [5]	TT	Formal	Path	48	19	2.5	9	0.20
Ibex custom [5]	TT	Formal	Path	330	76	4.3	18	0.59
Ibex small [5]	TT	Formal	Path	164	89	1.8	12	0.78
CVA6 (new trace)	TT	Simulation	Path	82,920	1187	769.9	144	604.0
CVA6 [19]	SC	Formal	Bottom	900	562	1.6	76	4.47
BOOM [17]	TT	Simulation	Path	17,316	213	81.3	988	100.93

Scalable:
Graph generation in seconds - minutes

Results

BOOM – Spectre:
A speculative load leaves a trace in the cache
and leaks through the
Miss Status Holding Register (MSHR)



17 317 tainted signals

Pathfinder

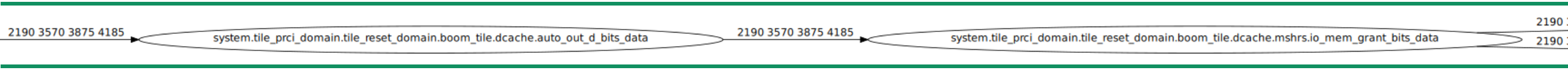
Taint Graph

213 nodes

...

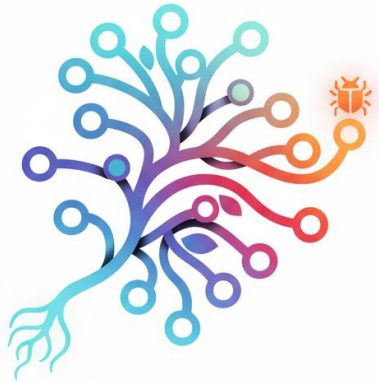
...

SPECTRE



Conclusion

- **Unified** taint tracking and self-composition



- Introduced **Pathfinder**
  - **Graphical aid** for **faster** information flow violation **debugging**
  - Reduces signals that need to be manually examined by **up to 700x**

Thank you! Questions?

Code:

comsec-group/pathfinder

Information:

https://comsec.ethz.ch/research/hardware-design-security/pathfinder/

Contact:

https://www.linkedin.com/in/katharina-ceesay-seitz-ba521087/

kceesay@ethz.ch, flavien.solt@eecs.berkeley.edu

References

[1] M. R. Fadiheh et al., "Processor Hardware Security Vulnerabilities and their Detection by Unique Program Execution Checking", DATE 2019

[2] L. Deutschmann et al., "A Scalable Formal Verification Methodology for Data-Oblivious Hardware", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

[3] M. S. Ali, "Security Risks in AI Accelerators: Detecting RTL Vulnerabilities to Model Theft with Formal Verification", ETS 2025

[4] M. Orenes-Vera et al., "AutoCC: Automatic Discovery of Covert Channels in Time-Shared Hardware", MICRO 2023

[5] Z. Wang, "Specification and Verification of Side-channel Security for Open-source Processors via Leakage Contracts", CCS 2023

[6] S. Dinesh, "H-Houdini: Scalable Invariant Learning", ASPLOS 2025

[7] https://www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-verification-platform/security-path-verification-app.html

[8] M. Tiwari et al., “Complete information flow tracking from the gates up,”, ASPLOS 2009

[9] A. Ardeshiricham, "Register transfer level information flow tracking for provably secure hardware design", DATE 2017

[10] F. Solt, B. Gras, K. Razavi, "CELLIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL", USENIX Security 2022

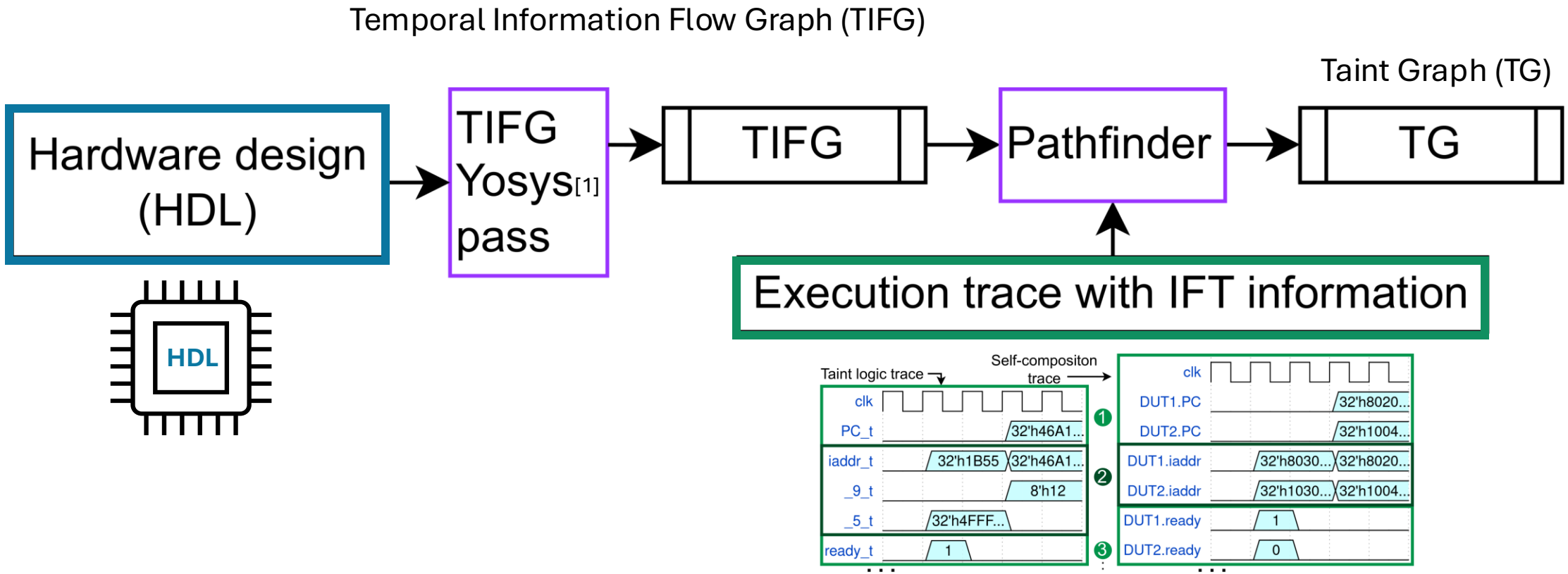
[11] Y. Hsiao et al., "RTL2MμPATH: Multi-μPATH Synthesis with Applications to Hardware Security Verification", MICRO 2024

[12] K. Ceesay-Seitz, F. Solt, K. Razavi, "μCFI: Formal Verification of Microarchitectural Control-flow Integrity", CCS 2024

[13] T. Kovats, F. Solt, K. Ceesay-Seitz, K. Razavi, "MileSan: Detecting Exploitable Microarchitectural Leakage via Differential Hardware-Software Taint Tracking", CCS 2025

BACKUP

Toolflow

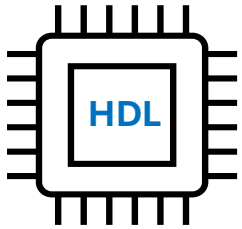


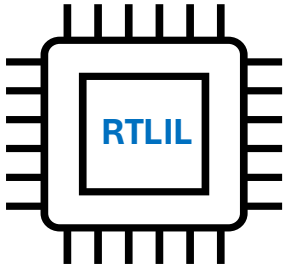
NEW

HDL = Hardware Description Language
IFT = Information Flow Tracking

[1] Yosys Open SYnthesis Suite - https://github.com/YosysHQ/yosys

CellIFT Yosys [1] pass

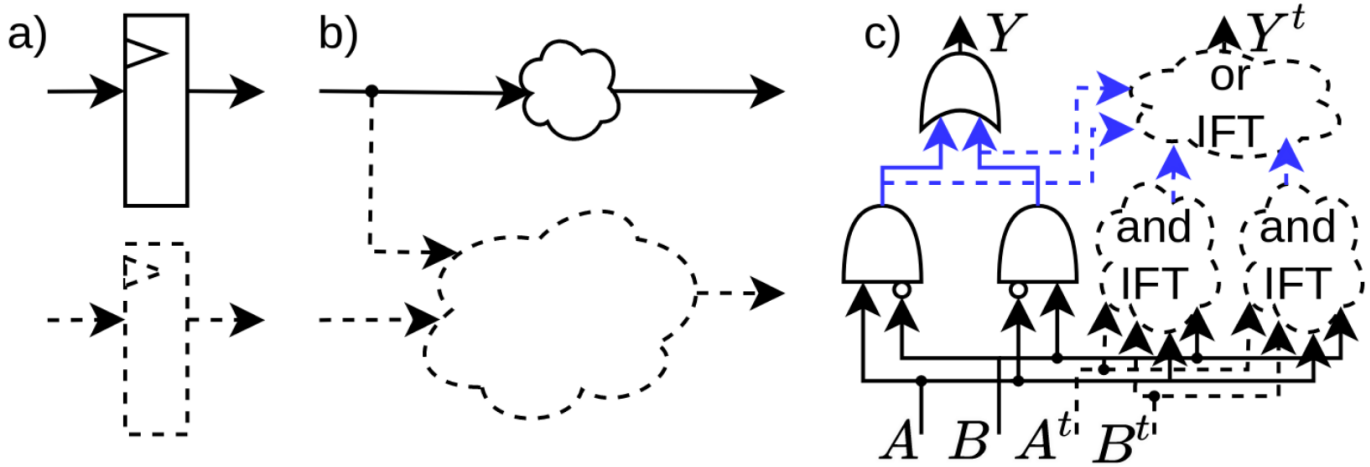


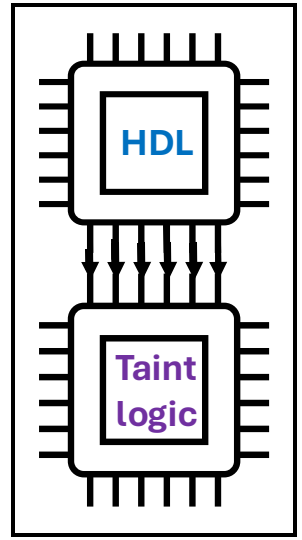


[2]

∀ cells (flip flops, logic cells, ...):

- Duplicate* in-/outputs for taint tracking
- Connect them with cell-type dependent taint tracking logic





a) State-holding cells
b) Combinational block
c) Gate-level output of Yosys

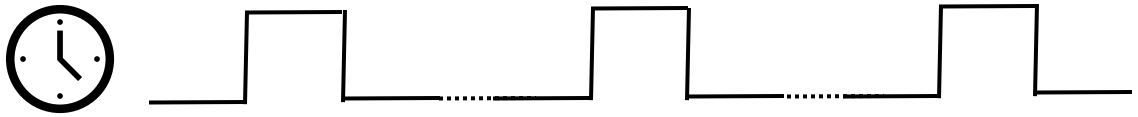
*it is possible to add multiple independent taint instrumentations. Each in-/output gets a taint representation per instrumentation.

[1] Yosys Open SYnthesis Suite - https://github.com/YosysHQ/yosys

[2] F. Solt, B. Gras, K. Razavi, "CELLIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL", USENIX Security 2022

µCFI - Microarchitectural Control-flow Integrity

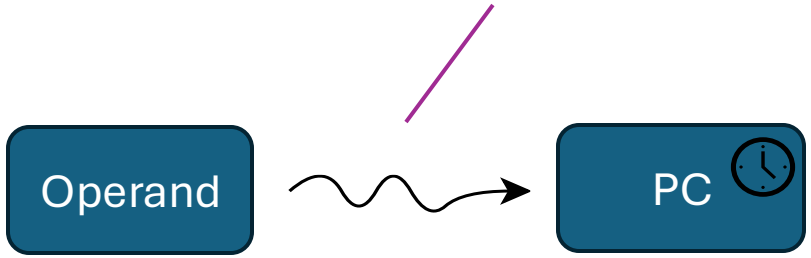
Microarchitectural control flow (µCF)



Program Counter (PC)

0x80000004 0x80000008 0x80001000

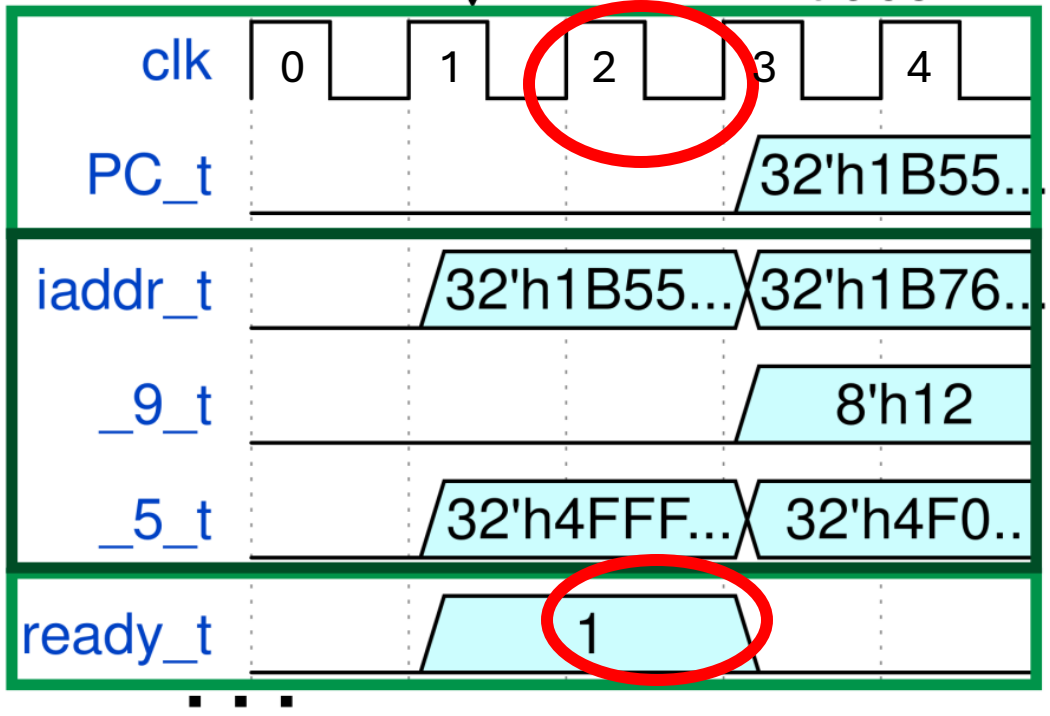
µCFI only allows explicitly ISA specified data dependencies of the µCF



K. Ceesay-Seitz, F. Solt, K. Razavi, "μCFI: Formal Verification of Microarchitectural Control-flow Integrity", CCS'24

ISA = Instruction Set Architecture

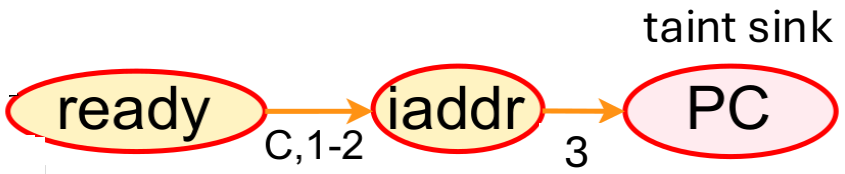
Pathfinder

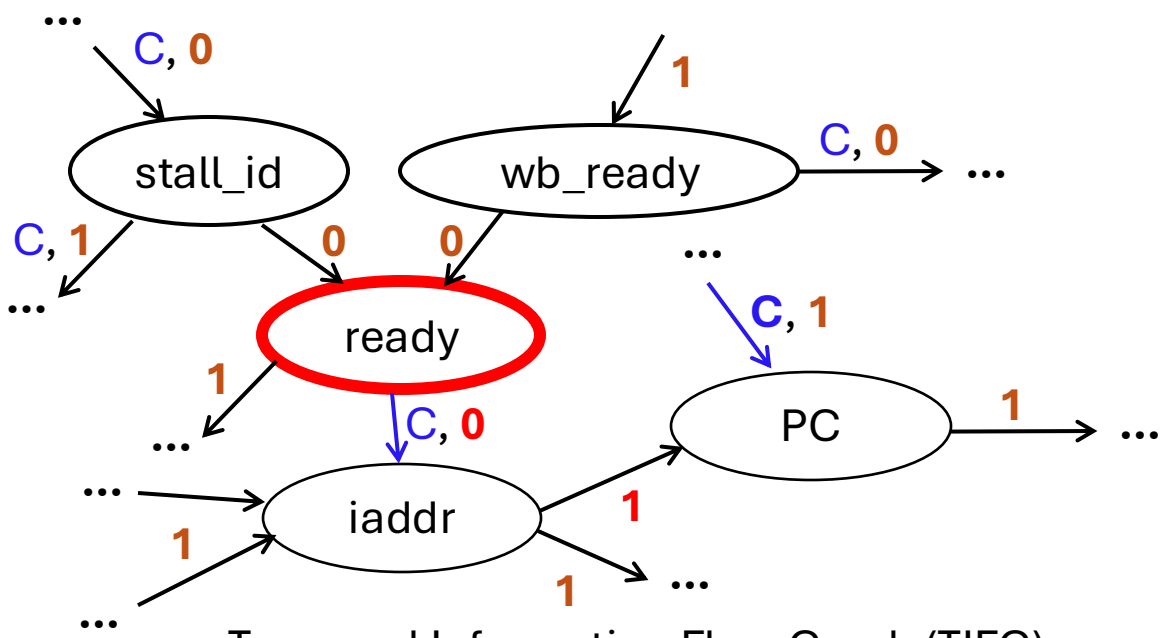


Taint Graph construction

Drivers with zero delay?

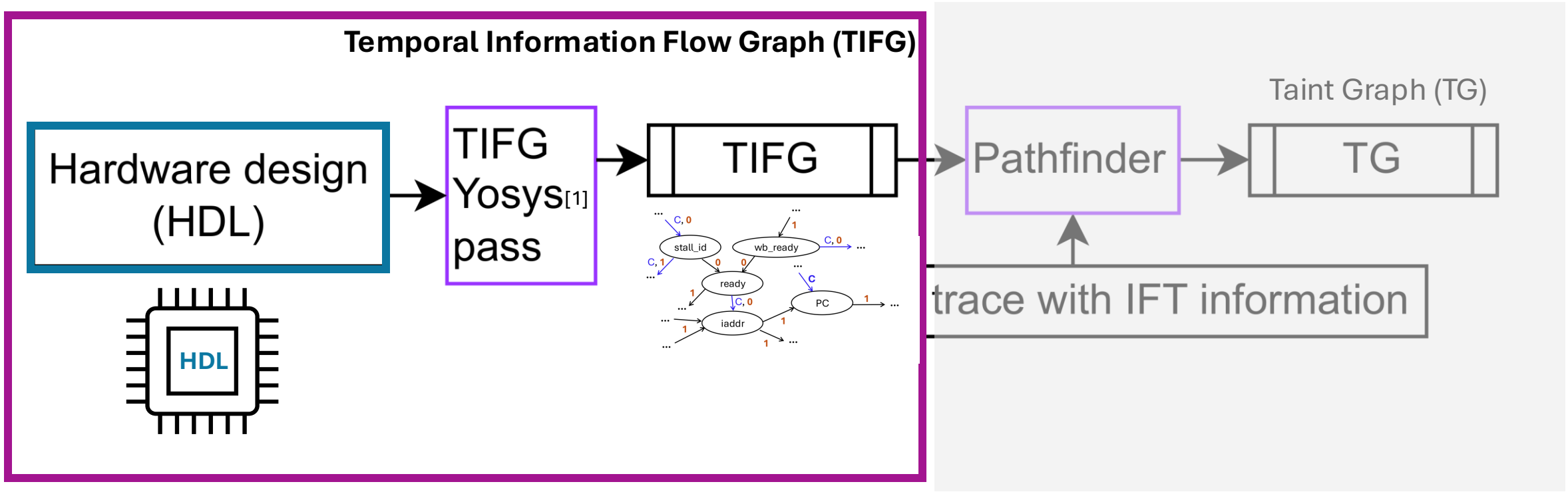
Yes.





Temporal Information Flow Graph (TIFG)

Toolflow



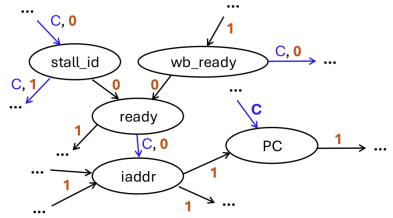
NEW

HDL = Hardware Description Language
IFT = Information Flow Tracking

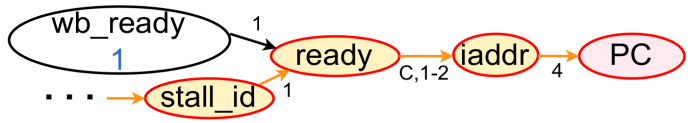
[1] Yosys Open SYnthesis Suite - https://github.com/YosysHQ/yosys

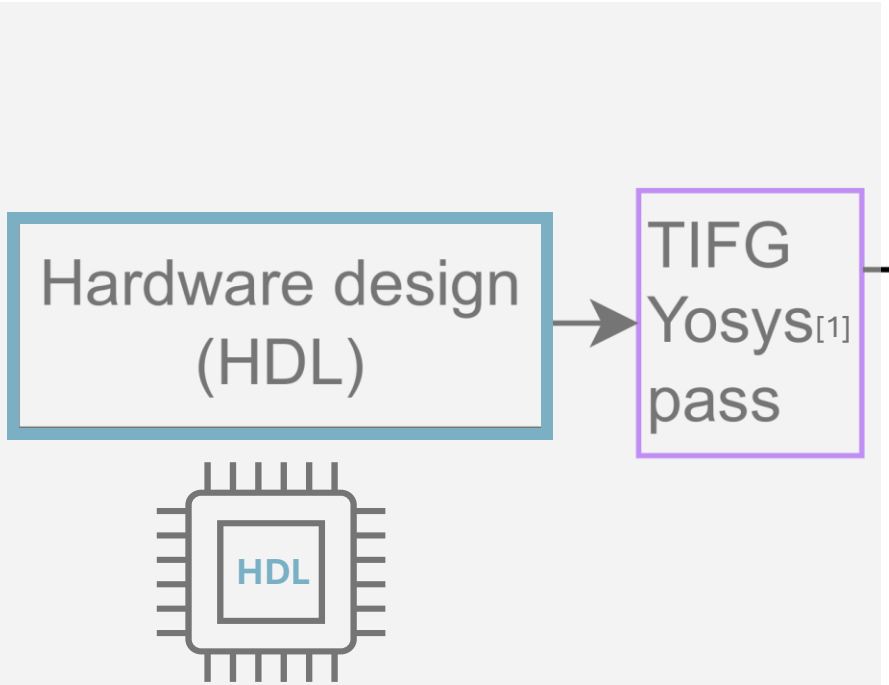
Toolflow

Temporal Information Flow Graph (TIFG)

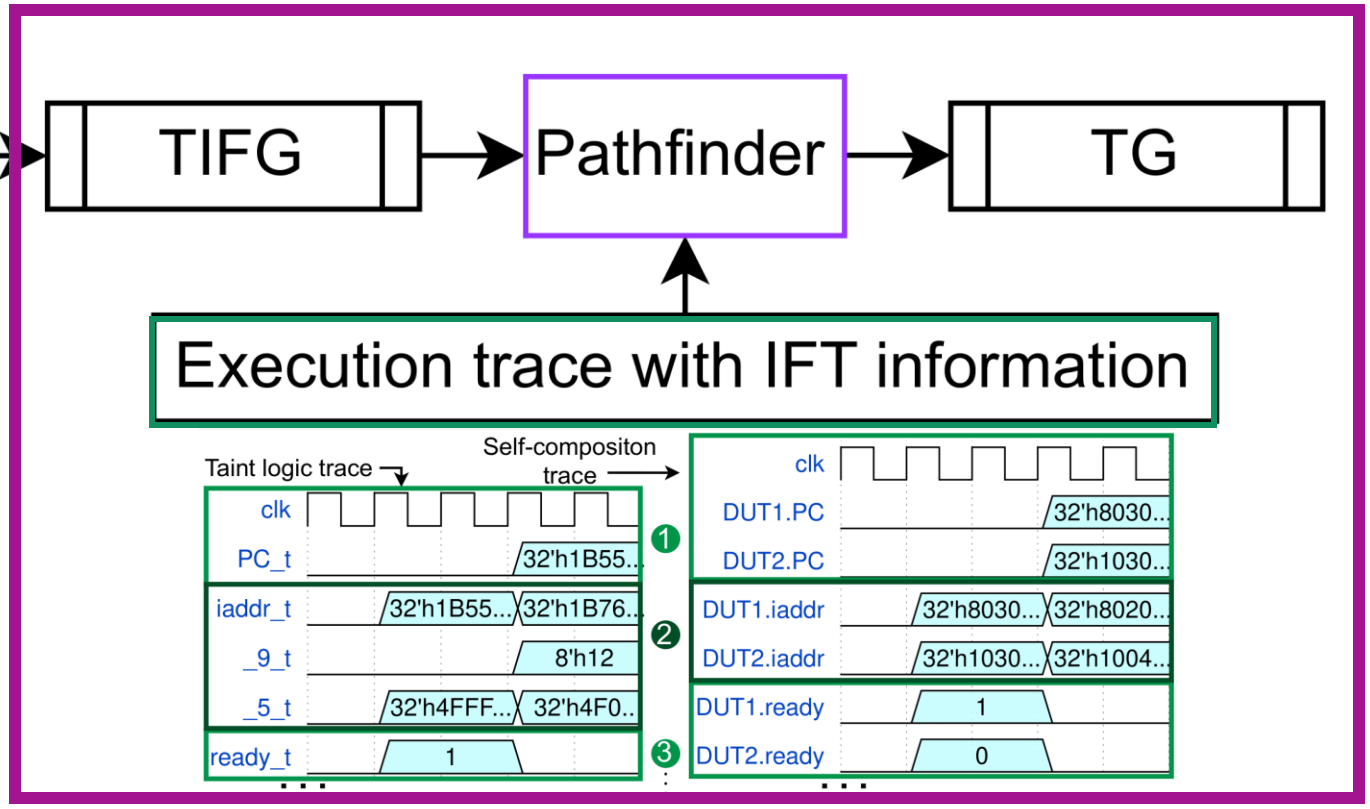


Taint Graph (TG)





TIFG Yosys[1] pass



NEW

HDL = Hardware Description Language
IFT = Information Flow Tracking

Unification: Pathfinder derives **taint** values from **differences** in self-composed design