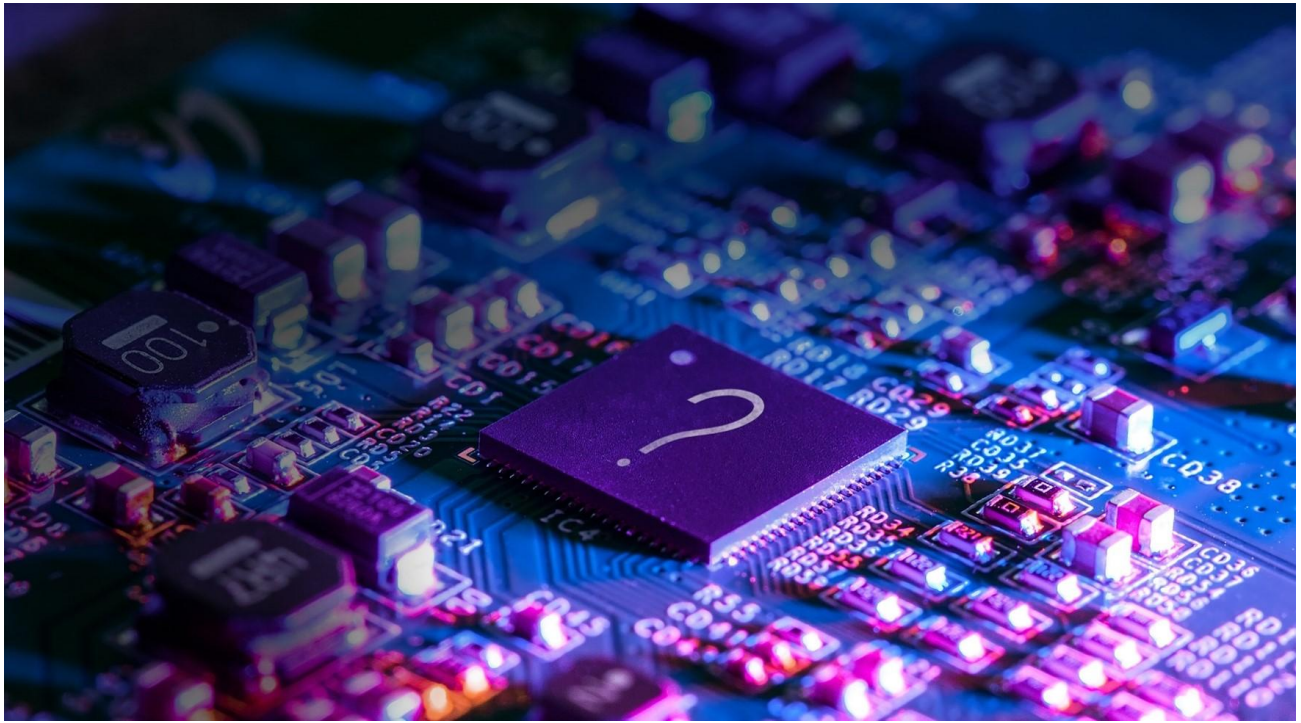
Mining Security Critical Linear Temporal Logic Specifications for Processors

Calvin Deutschbein, Cynthia Sturton
University of North Carolina at Chapel Hill

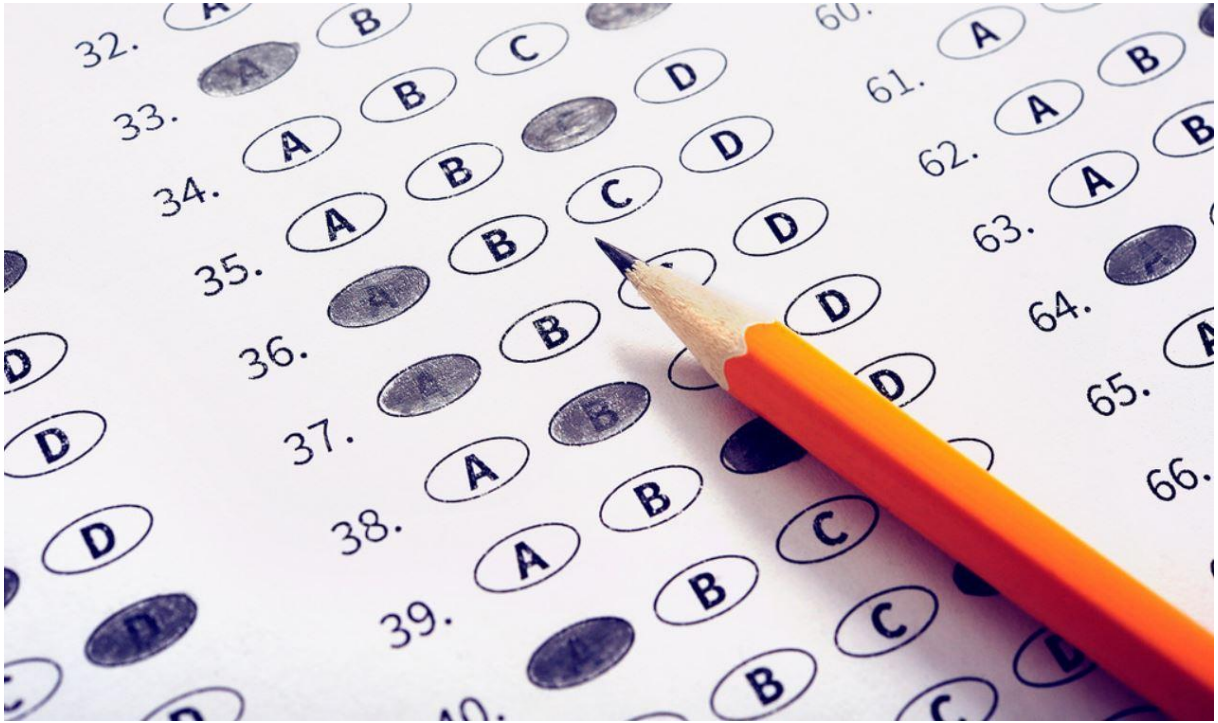
Hardware Security @ UNC

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Hardware May Contain Security Bugs



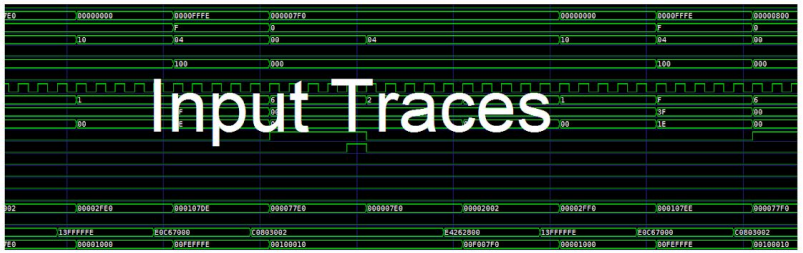
Testing Works – But What To Test For?



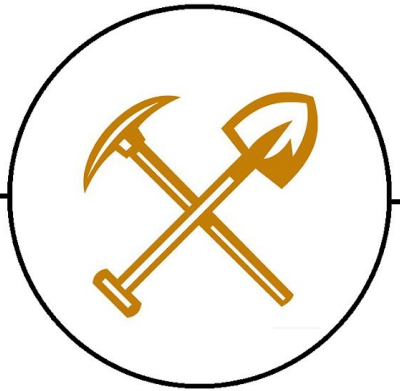
Research Question

How can we find security critical properties for hardware verification and testing?

Specification Mining



LTL Input Templates:
G(a -> b), a U G(b),
G((a & b) -> c)



Properties:
sr=0->rst=0
rst=0 U sr=*x*

Difficulties Find Security Properties

Too Many Properties

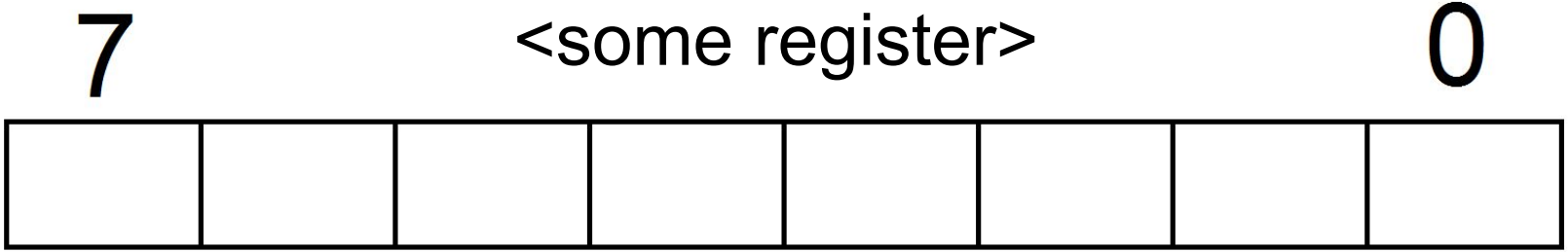
Properties Not Security Related

Properties Not Of Correct Form

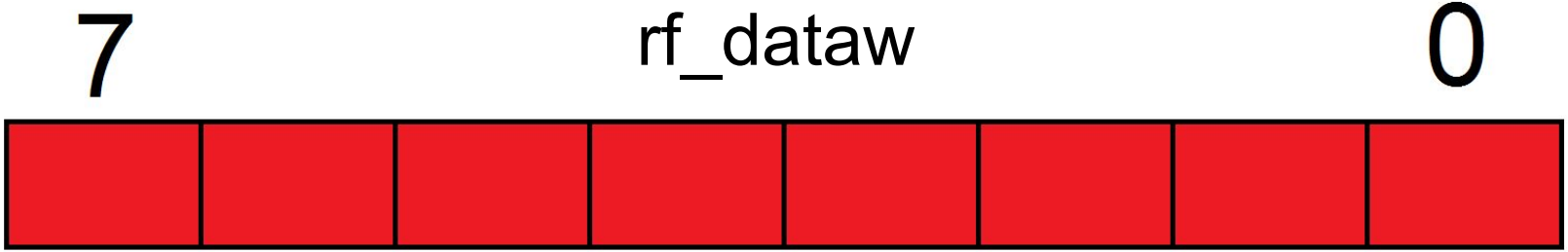
Research Contribution

We type processor events and create a library of security-oriented typed templates for mining

What can be encoded in a register?

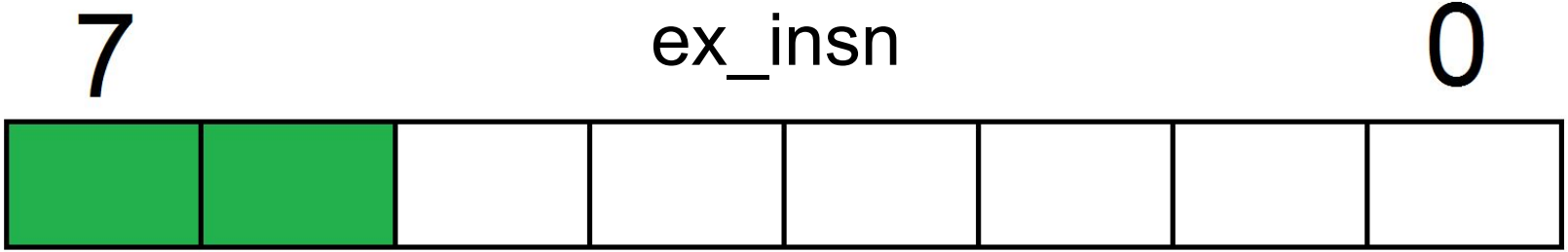


Register Type: Registers simply holds some value



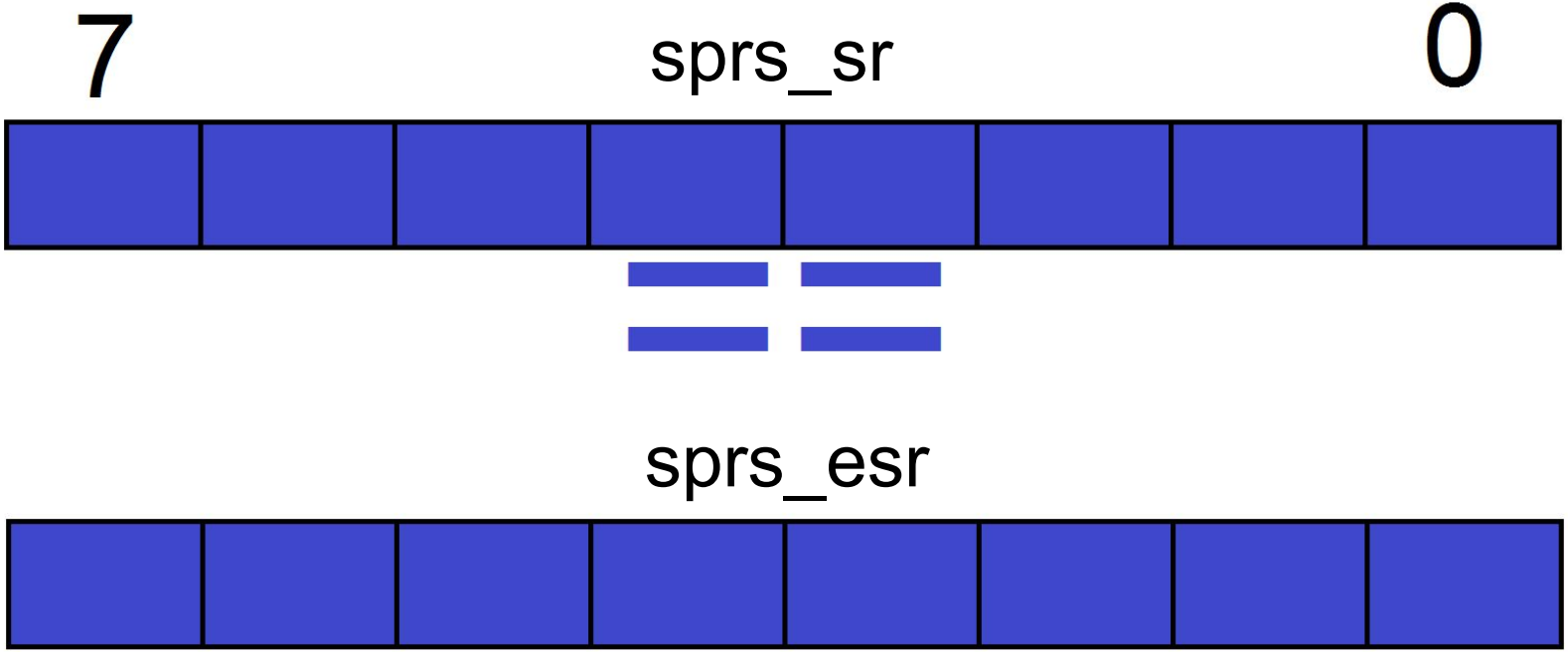
This gives the value to be written to a register.

Slice-Register: Internal Bits Have Semantic Meaning



This gives the opcode with no operands.

Register-Register: Two Registers Must Be Equal



This shows if status is saved during exceptions.

Known Properties + Expertise = Typed Templates

1	$\text{Register-Register}_a \ U \ G(\text{Register}_b)$
2	$G(\text{Slice-Register}_a \rightarrow \text{Register-Register}_b)$
3	$G((\text{Slice-Register}_a \wedge \text{Slice-Register}_b) \rightarrow \text{Register-Register}_c)$

Without Typing There Are Many Properties

Sample Trace

reg_a==1

reg_b==1

reg_c==0

reg_d==0

reg_a==reg_b

reg_c==reg_d

Mined 30 G(x → y)

reg_a==1→ reg_b==1

...

reg_a==1→ reg_c==reg_d

…

reg_c==reg_d→ reg_a==reg_b

Typing Events Reduces The Number

Sample Trace

reg_a==1

reg_b==1

reg_c==0

reg_d==0

reg_a==reg_b

reg_c==reg_d

Mined 8 G($R_b \rightarrow R\text{-}R_b$)

reg_a==1→ reg_a==reg_b

reg_a==1→ reg_c==reg_d

reg_b==1→ reg_a==reg_b

…

reg_f==0→ reg_c==reg_d

Type reg_a,b Register, reg_c,d Register-Register

Sample Trace

reg_a==1

reg_b==1

reg_c==0

reg_d==0

reg_a==reg_b

reg_c==reg_d

Mined 2 G($R_b \rightarrow R\text{-}R_b$)

reg_a==1→ reg_c==reg_d

reg_b==1→ reg_c==reg_d

Register Slices Uncover Sematic Meaning

Sample Trace

reg_a==7

#tick

reg_a==3

#tick

reg_a==5

...

Mining G(a)

<no properties>

Register Slices Uncover Sematic Meaning

Sample Trace

reg_a[0]==1

reg_a[1]==1

reg_a[2]==1

#tick

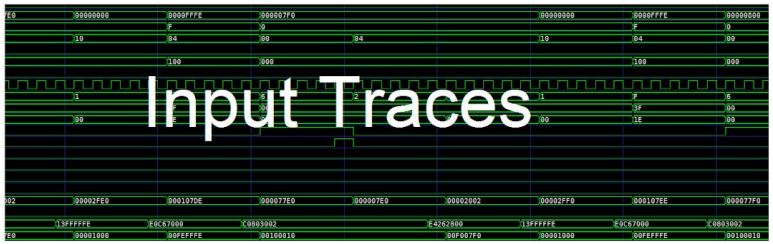
reg_a[0]==1

...

Mining G(a)

G(reg_a[0]==1)

UNDINE Implements Typed Mining



Input Typed LTL Templates

RR_a U G(R_b),G($SR_a \rightarrow RR_b$)

G(($SR_a \wedge SR_b$) $\rightarrow$ R-R_c)

Input Type Information:

type(rst) = R, type(insn) = SR

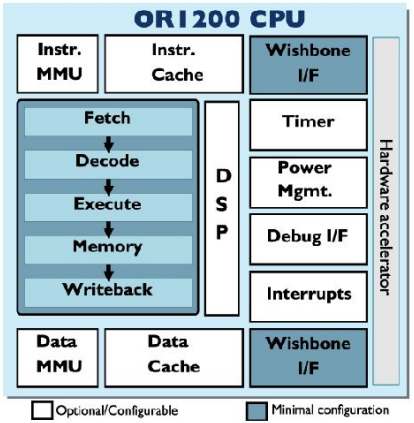
UNDINE

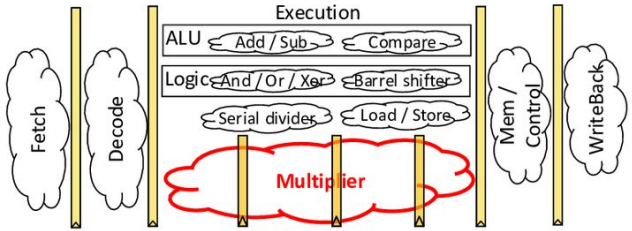
Security Properties:

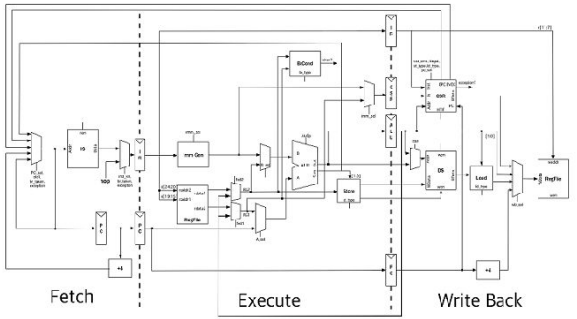
G(GPRO==0)

esr==sr U rst

Tested on 3 Processors

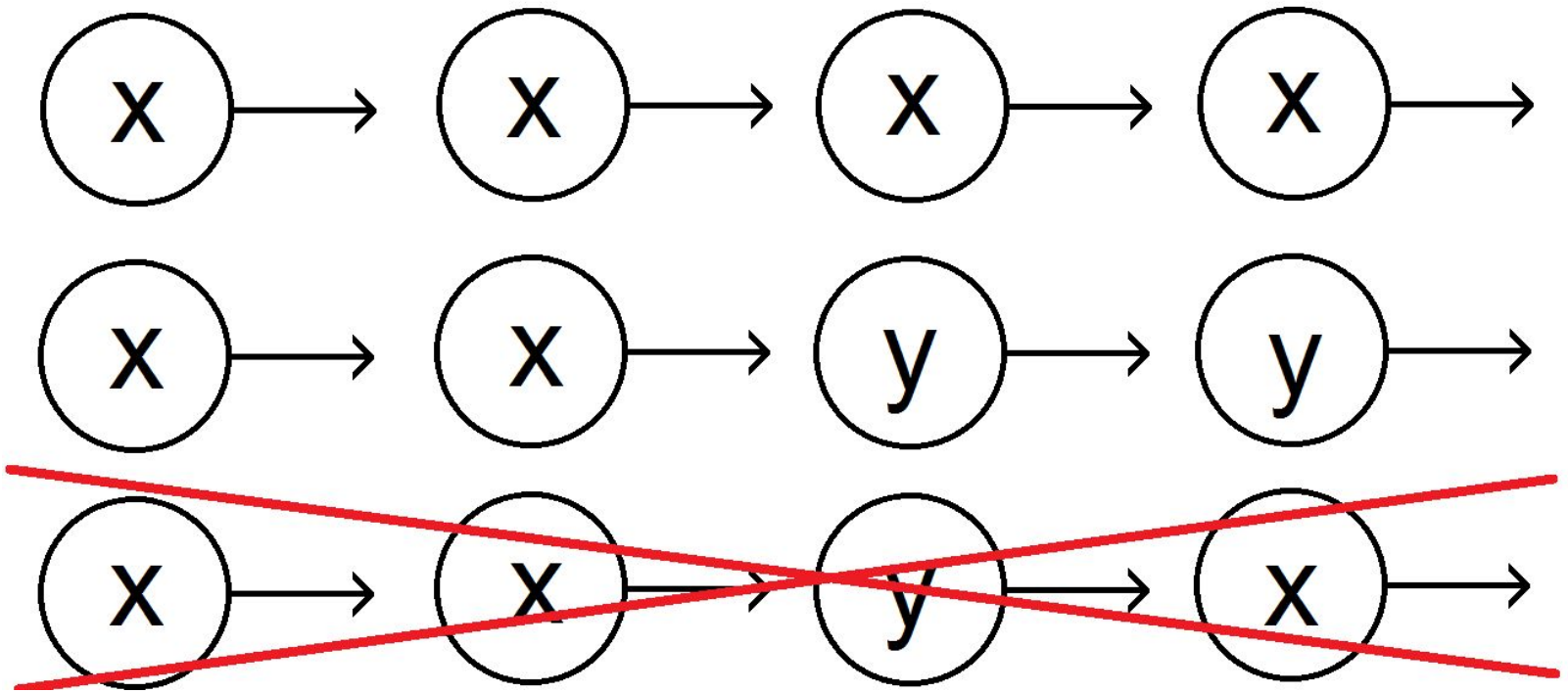




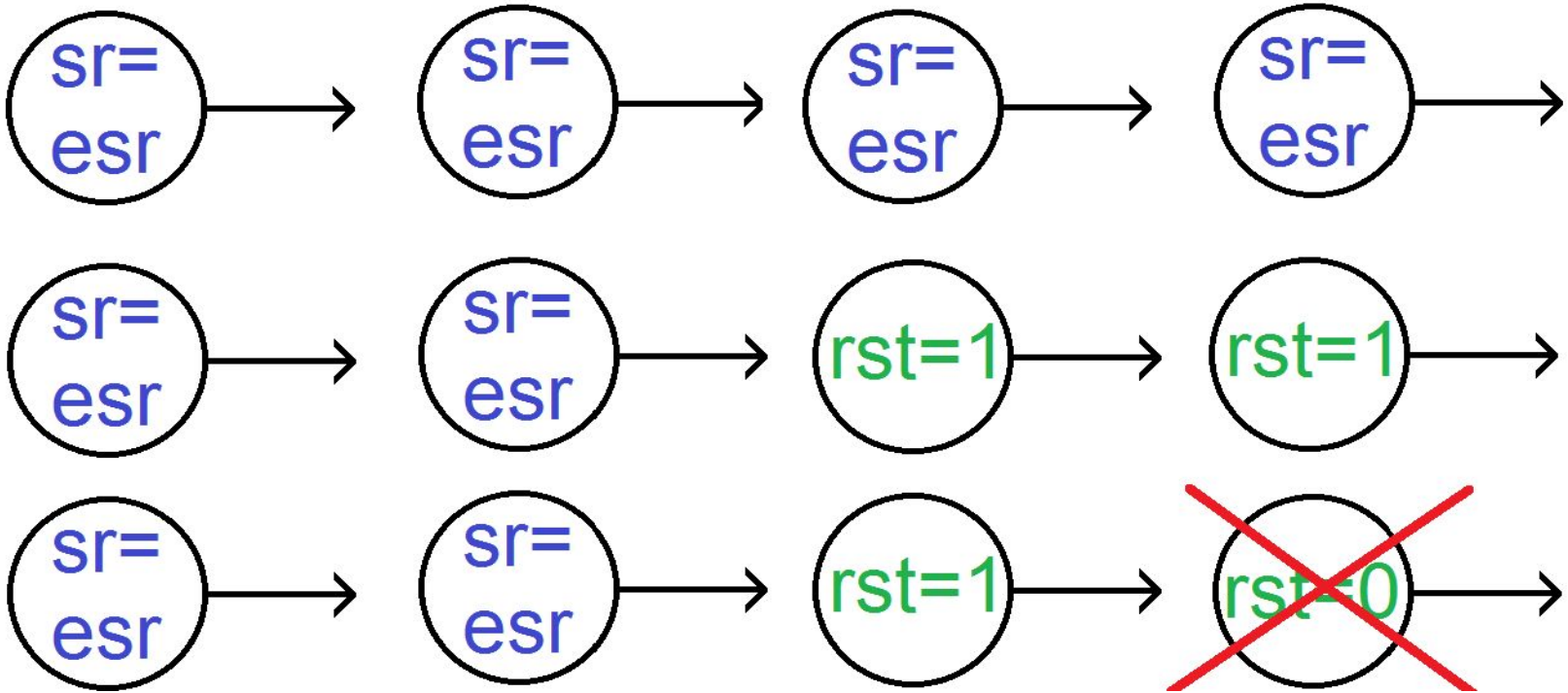


OR1200 mor1kx RISC-V

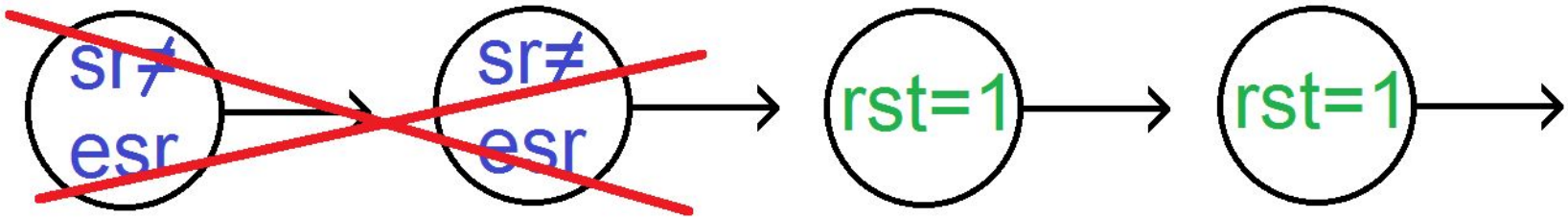
Example: x U y



Example: sr==esr U G(rst==1)

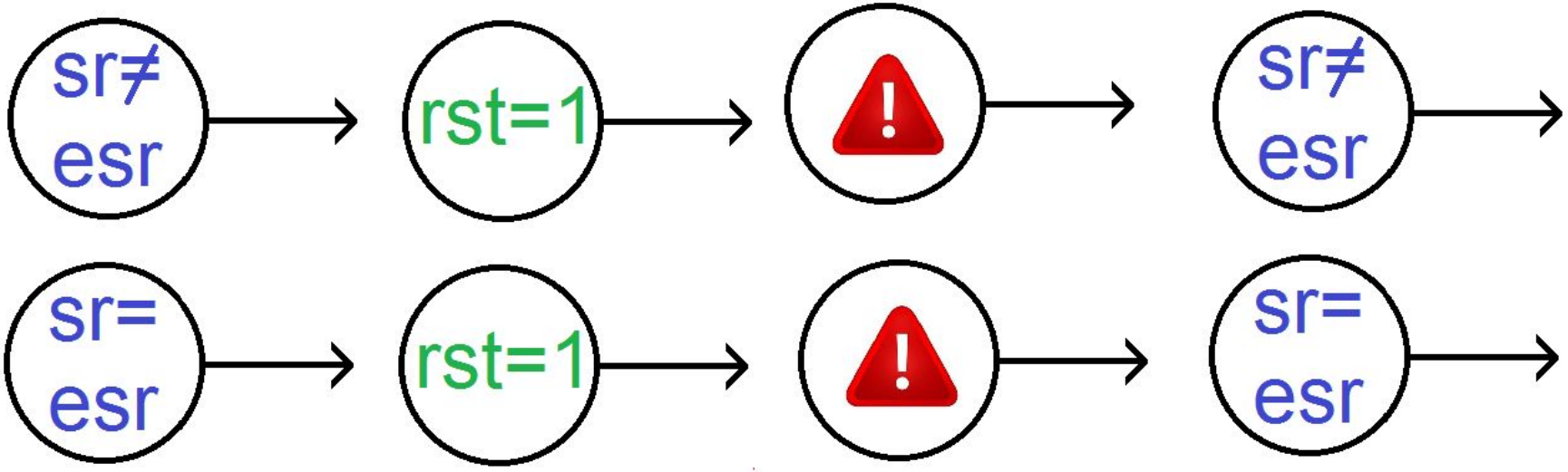


Example: Attack Status Register Initialization



Status and Exception Status Initialized Together

Example: Exploit by Triggering Exception



The Incorrect Value is Saved as Exception Status!

Thank you!

How can we find security critical properties for hardware verification and testing?

We type processor events and create a library of security-oriented typed templates for mining