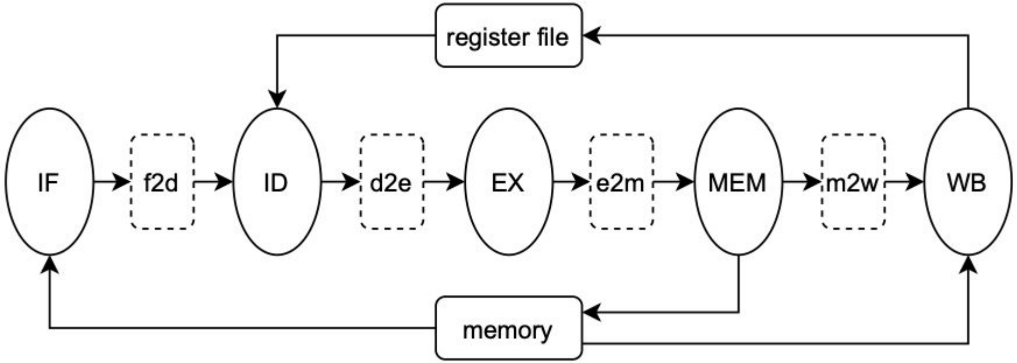
A Correct (and secure) Silver Pipelined Processor

Ning Dong, Roberto Guanciale, Mads Dam, Andreas Lööw

Method for correctness

- Silver ISA model
  - Simple RISC processor
- **Processor design modeled as list of stage functions**
  - $c^2 = f_i(c^1, e^1)$
- **Verification of f_1 … f_n w.r.t. ISA**
- Verilog HOL4 library
  - Library generates a Verilog design and proves a simulation theorem:
    - if $(c^1,e^1) \rightarrow (c^2,e^2)$ then $c^2 = f_1 \ldots f_n(c^1, e^1)$
  - Library exports Verilog design

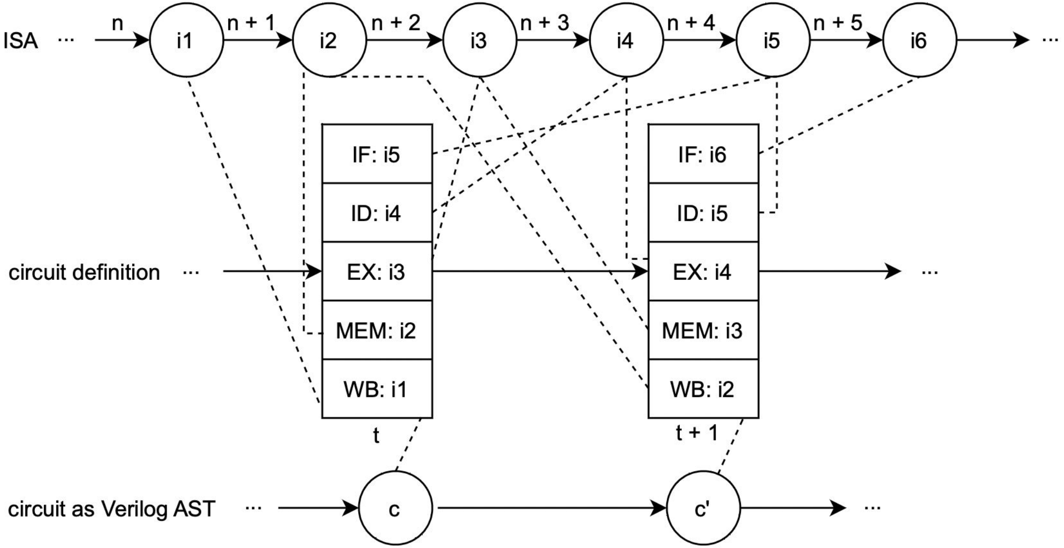
Processor Design



- Data hazard
- Mispredicted PC
  - Target addresses computed by EX
- Self modifying code
  - SW must take countermeasures

When a processor is correct?

- Usage of a scheduling function: I(j)(stage) = n

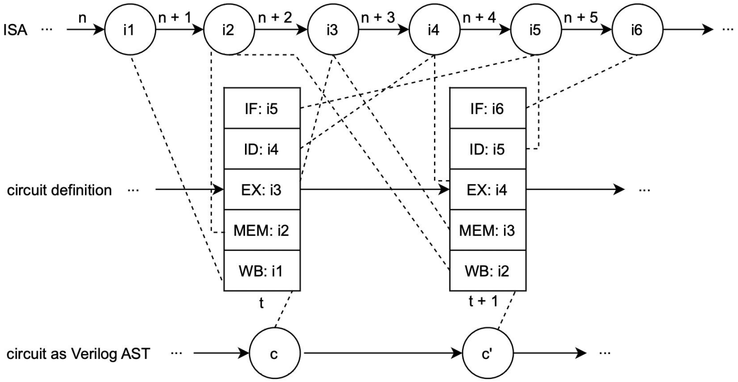


When a processor is correct?

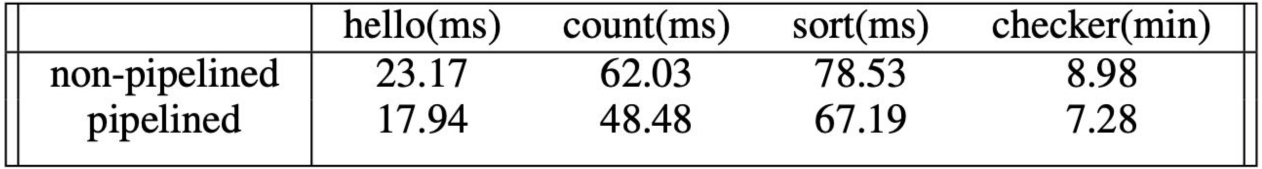
- Scheduling function is defined inductively, e.g.
  - if t^j.EX.enabled then I(j)(EX) = I(j-1)(ID)
  - if t^j.EX.disabled then I(j)(EX) = I(j-1)(EX)
- Pipeline may be processing (in IF and ID) wrong instructions due to JMP
  - I(t^j.EX) = n and s_n takes a JMP then I(j)(IF) = ⊥
  - I(t^j.ID) = n and s_n takes a JMP then I(j)(IF) = ⊥
- two traces are related
  - $(t^1,e^1) \to (t^2,e^2) \to \dots (t^j,e^j) \sim^I s^1 \to s^2 \to \dots s^n$
  - iff for every cycle j, and every circuit field f of each stage
    - if I(j)(stage) = ⊥ then t^j.stage.f = t^{j-1}.stage..f
    - if I(j)(stage) = i then t^j.stage.f = F(s^i)

When a processor is correct?

- Proof establishes that
  - if (c1,e1) → (c2,e2) → … (cj,ej) and
  - if s^i writes in address a then for $i < k < i+5$: $s^k.pc \neq a$
  - then $(c^1,e^1) \rightarrow (c^2,e^2) \rightarrow \dots (c^j,e^j) \sim^I s^1 \rightarrow s^2 \rightarrow \dots s^n$
- Proof done by induction



Evaluation and Future work



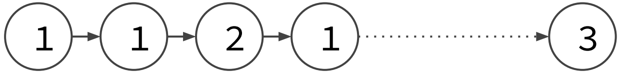
	hello(ms)	count(ms)	sort(ms)	checker(min)
non-pipelined	23.17	62.03	78.53	8.98
pipelined	17.94	48.48	67.19	7.28

- 45 vs 65 Mhz :Main bottleneck I-cache design
- Future work
  - (done) Data Forward
  - Integration with CakeML for full stack correctness (security)
  - Mechanised proof of security

When a processor is secure?

Information flow specification

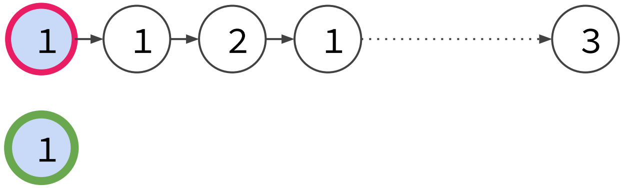
- There is no standard specification of information flows



Information flow specification

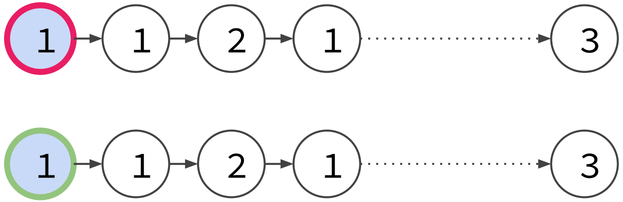
- We extend ISA with observations that should overapproximate information leakage: obs(s)
  - Execution path: s.pc
  - Executed instruction: s.mem[s.pc]
  - Memory operation and address:
    - op(s.mem[s.pc]), addr(s.mem[s.pc], s.regs)

Information flow specification



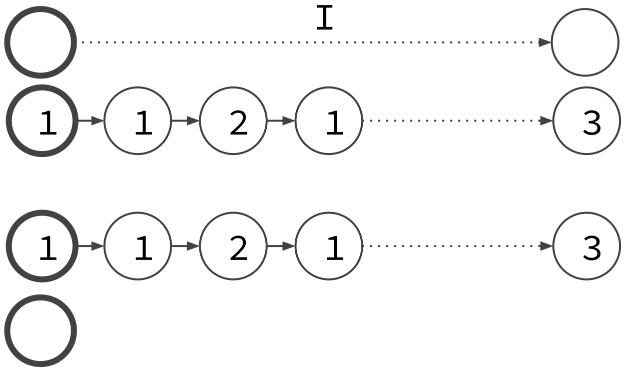
- We extend ISA with observations that should overapproximate information leakage: obs(s)
  - Execution path: s.pc
  - Executed instruction: s.mem[s.pc]
  - Memory operation and address:
    - op(s.mem[s.pc]), addr(s.mem[s.pc], s.regs)
- obs are used to define when a SW is non-interferent
  - if $a^1 \rightarrow a^2 \rightarrow \dots a^n, b^1 \rightarrow b^2 \rightarrow \dots b^n$, and $a^1 =^{low} b^1$
  - then $obs(a^i) = obs(b^i)$

Information flow specification



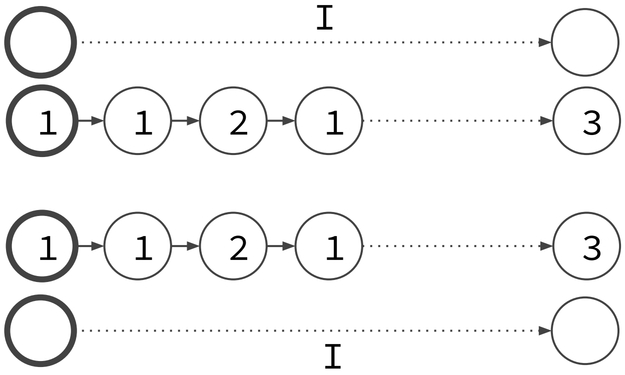
- We extend ISA with observations that should overapproximate information leakage: obs(s)
  - Execution path: s.pc
  - Executed instruction: s.mem[s.pc]
  - Memory operation and address:
    - op(s.mem[s.pc]), addr(s.mem[s.pc], s.regs)
- obs are used to define when a SW is non-interferent
  - if $a^1 \rightarrow a^2 \rightarrow \ldots a^n, b^1 \rightarrow b^2 \rightarrow \ldots b^n$, and $a^1 =^{low} b^1$
  - then $obs(a^i) = obs(b^i)$

Is the processor secure?



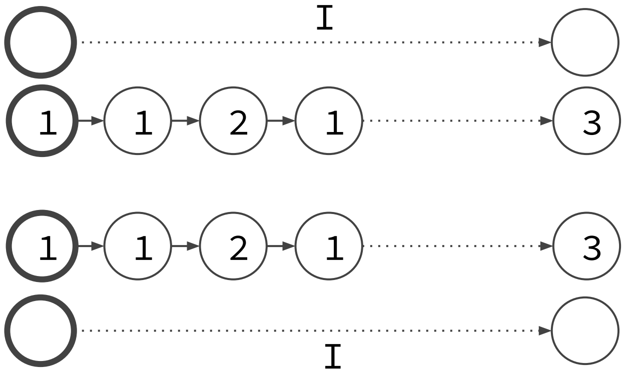
- Conditional non-interference
  - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^{I} \sigma^1$, and ϕ^2[0] $\sim^{I}$ σ^1[0]
  - then $\phi^2 \sim^{I} \sigma^2$

Is the processor secure?



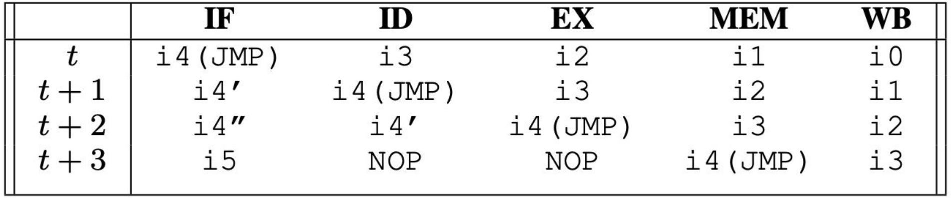
- Conditional non-interference
    - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^{\mathbf{I}} \sigma^1$, and $\phi^2[0] \sim^{I} \sigma^1[0]$
    - then $\phi^2 \sim^{\mathbf{I}} \sigma^2$

Is the processor secure?



- Conditional non-interference
  - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^{I} \sigma^1$, and $\phi^2[0] \sim^{I} \sigma^1[0]$
  - then $\phi^2 \sim^{I} \sigma^2$
- To verify this we need assumption on the environment:
  - if for add i<j
    - ϕ^1(i).t.cmd = ϕ^2(i).t.cmd
    - ϕ^1(i).t.out_pc = ϕ^2(i).t.out_pc
    - ϕ^1(i).t.cmd = ld/st then
      - ϕ^1(i).t.adr = ϕ^2(i).t.adr
  - then
    - ϕ^1(j).e.rdy = ϕ^2(j).e.rdy
- Memory subsystem
  - can have caches, different access times, preloads, different replacement policies
  - but if it receives the same sequence of commands for the same addresses must reply in at the same time

Are all processor secure?

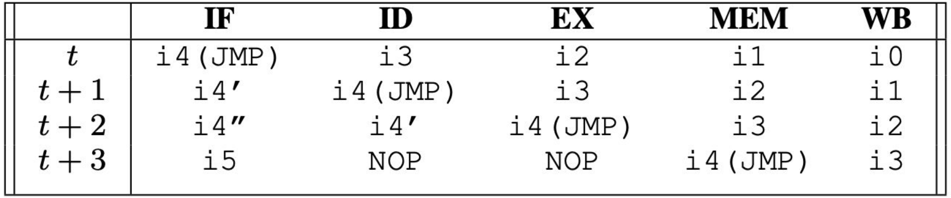


	IF	ID	EX	MEM	WB
t	i4(JMP)	i3	i2	i1	i0
$t+1$	i4′	i4(JMP)	i3	i2	i1
$t+2$	i4″	i4′	i4(JMP)	i3	i2
$t+3$	i5	NOP	NOP	i4(JMP)	i3

- Conditional noninterference
  - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^I \sigma^1$, and $\phi^2[0] \sim^I \sigma^1[0]$
  - then $\phi^2 \sim^I \sigma^2$

```
i3:     R0:=0;

i4:     JMP I5;

i4':      ?

i4'':    ?

i5:     INTR;
```

Are all processor secure?

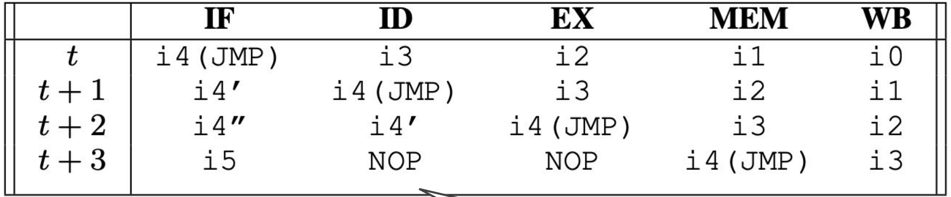


	IF	ID	EX	MEM	WB
t	i4(JMP)	i3	i2	i1	i0
$t+1$	i4′	i4(JMP)	i3	i2	i1
$t+2$	i4″	i4′	i4(JMP)	i3	i2
$t+3$	i5	NOP	NOP	i4(JMP)	i3

- Conditional noninterference
  - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^I \sigma^1$, and $\phi^2[0] \sim^I \sigma^1[0]$
  - then $\phi^2 \sim^I \sigma^2$

```
i3:     R0:=0;

i4:     JMP I5;

i4’:      ?

i4’’:  R2:=R0+1 or R2:=0

i5:     INTR;
```

Are all processor secure?



	IF	ID	EX	MEM	WB
t	i4(JMP)	i3	i2	i1	i0
$t+1$	i4′	i4(JMP)	i3	i2	i1
$t+2$	i4″	i4′	i4(JMP)	i3	i2
$t+3$	i5	NOP	NOP	i4(JMP)	i3

NOP:R0+1
NOP:0

- Conditional noninterference
  - if obs(σ^1) = obs(σ^2), $\phi^1 \sim^I \sigma^1$, and $\phi^2[0] \sim^I \sigma^1[0]$
  - then $\phi^2 \sim^I \sigma^2$

```
i3:     R0:=0;

i4:     JMP I5;

i4’:      ?

i4’’:   R2:=R0+1 or R2:=0

i5:     INTR;
```

Questions?