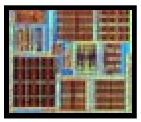
Assume-Guarantee Validation for STE Properties within an SVA Environment

Tom Melham
Oxford University

Zurab Khasidashvili & Gavirel Gavrielov
Intel Israel Ltd.

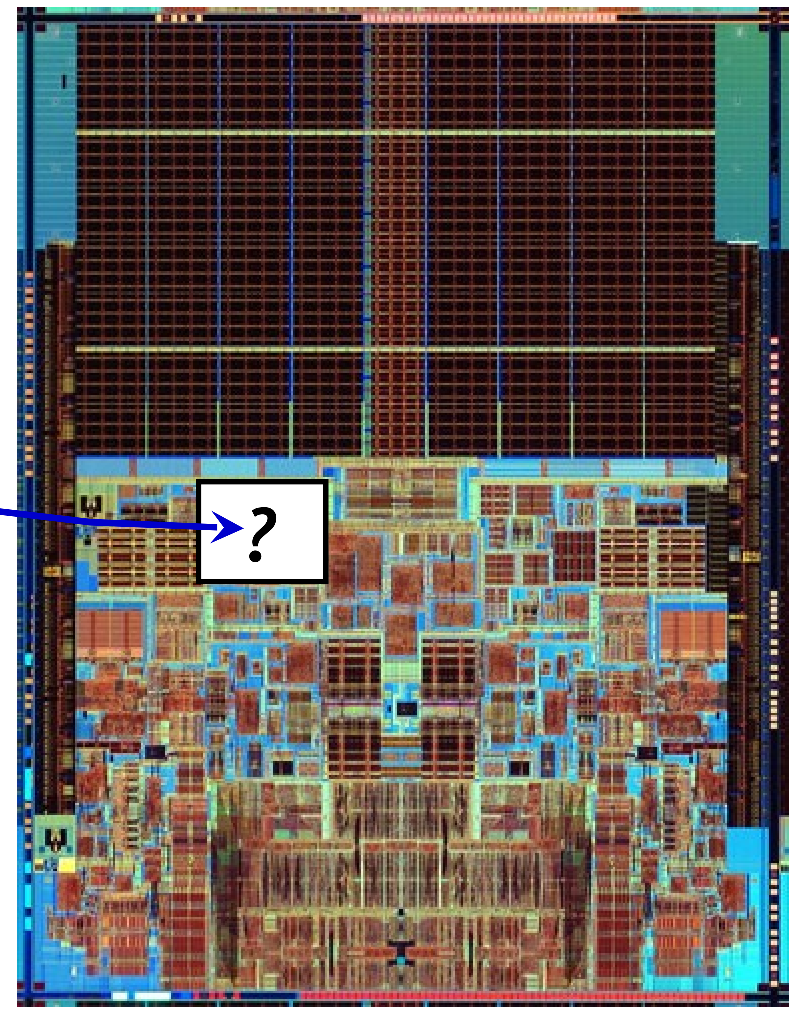
Validation of STE Verification Environment

- Assume (STE)



$$, \; \underbrace{P \models A}_{} \Rightarrow C$$

- Guarantee (SVA)



Big processor EXE proofs

- improve assumptions
- catch environment bugs

Symbolic Trajectory Evaluation

f := n is 0

| n is 1

| f_1 and f_2

| N f

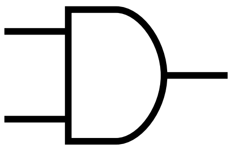
| $P \rightarrow f$

stimulus

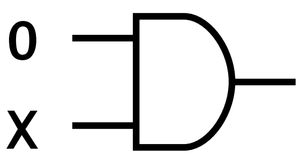
$A \Rightarrow C$

response

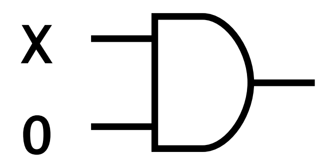
Example



$v \rightarrow$ a is 0 and $\bar{v} \rightarrow$ b is 0 $\Rightarrow$ out is 0



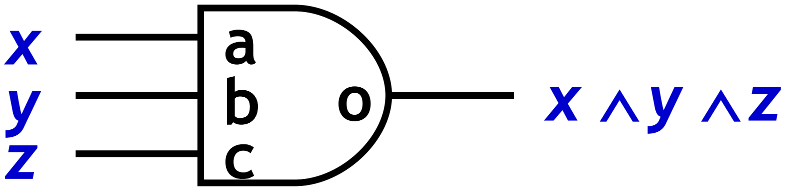
a is 0 $\Rightarrow$ out is 0



b is 0 $\Rightarrow$ out is 0

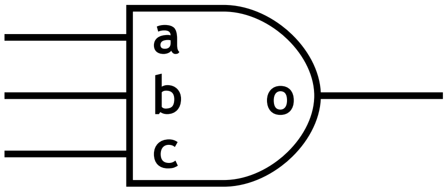
Symbolic Simulation

$$n \text{ is } E \quad = \quad E \rightarrow (n \text{ is } 1) \text{ and } \overline{E} \rightarrow (n \text{ is } 0)$$



$$(\text{a is } x) \text{ and } (\text{b is } y) \text{ and } (\text{c is } z) \quad \Rightarrow \quad \text{o is } x \wedge y \wedge z$$

Symbolic Indexing



a	b	c
0	X	X
X	0	X
X	X	0
1	1	1

$\neg p \wedge \neg q \rightarrow$ (a is 0) and

$\neg p \wedge q \rightarrow$ (b is 0) and

$p \wedge \neg q \rightarrow$ (c is 0) and

$p \wedge q \rightarrow$ (a is 1) and (b is 1) and (c is 1)

$\Rightarrow$

$\neg(p \wedge q) \rightarrow$ (o is 0) $\wedge$

$(p \wedge q) \rightarrow$ (o is 1)

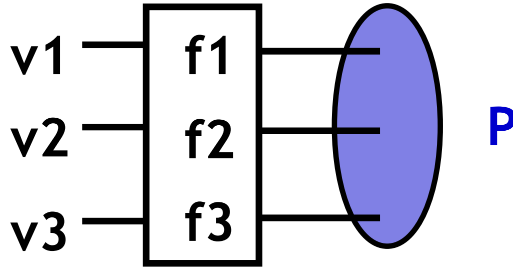
Environmental Constraints

- Conditional verification

$$P[xs] \models A[xs] \Rightarrow C[xs]$$

- Parametric representation

$$fs[vs] := \text{param}(xs, P[xs])$$



- Efficient verification

$$A[fs[vs]] \Rightarrow C[fs[vs]]$$

Translation to SVA?

- Easy case

$$x \vee y \models \text{ a is } x \text{ and b is } y \Rightarrow \ldots \qquad \text{a || b}$$

- Harder…

$$R[z] \models P \rightarrow (\text{a is } z) \text{ and } Q \rightarrow (\text{b is } z) \Rightarrow \ldots$$

Machine Representation - 5 Tuples

$$(guard,\ node,\ value,\ start,\ end)$$

f := n is 0

| n is 1

| f_1 and f_2

| N f

| $P \rightarrow f$

($P \rightarrow$ a is x) and ($P \rightarrow$ N(a is x))

(P, a, x, 0, 2)

STE Proof Environment - SVA Guarantee

$P \models A \Rightarrow C$

	restrictions	***assumptions***
timed	ignore signals	how inputs driven
global	ignore behaviours	input constraints

not trigger or checker

Methodology Restrictions For Boolean Variables

- For each *x* need at least one:

  (P, n, *x*, s, e)

- Variable dependency

  (Q[*y*], _, *z*, _, _)

  (P[*x*], _, *y*, _, _)

is a strict partial order.

Finding a Representative Name

$T(x,g) = \{(g_1, \_, x, \_, \_), \ldots, (g_n, \_, x, \_, \_)\}$

$g_1 \supset g \quad \ldots \quad g_n \supset g$

s = earliest start time

n = node with earliest start time

f = future reference time

node(x,g) = $past(n,f-s)

Translating Boolean Constraints

P - support = $\{x_1, \ldots, x_n\}$

θ = choose one node(x_i,g_i) for each x_i.

exp(P,θ) = (g_1 θ && … && g_j θ) <= P θ

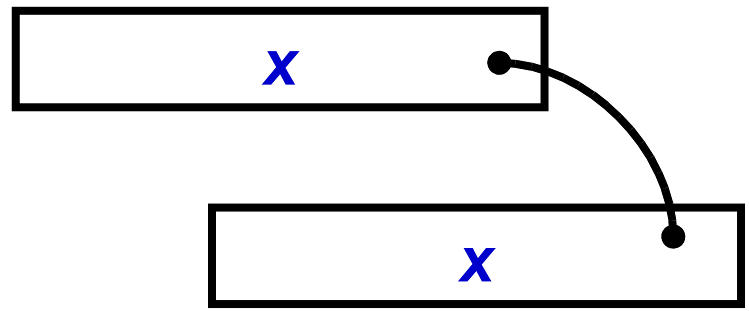
Exp(P) = (exp(P,θ_1) && … && (exp(P,θ_k))

Seq(P) = ##f Exp(P,θ)

Implicit Equality Constraints

$(g_1, n_1, \color{blue}{x}, s_1, e_1)$

$(g_2, n_2, \color{blue}{x}, s_2, e_2)$



$$\#\#f \quad Exp(g_1 \wedge g_2) <= \$past(n_1, f\text{-}e_1) == \$past(n_2, f\text{-}e_2)$$

Per-Tuple Stability Constraints

(g, n, x, s, e)

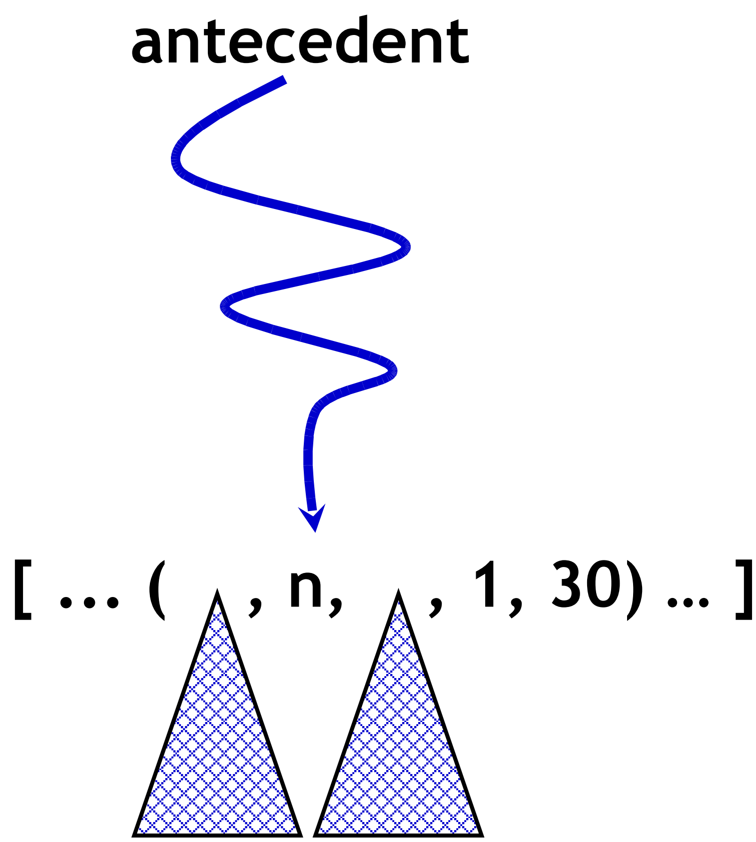
not(Seq(g)) or (##s+1($stable(n))[*e-s-1])

(g, n, E, s, e)

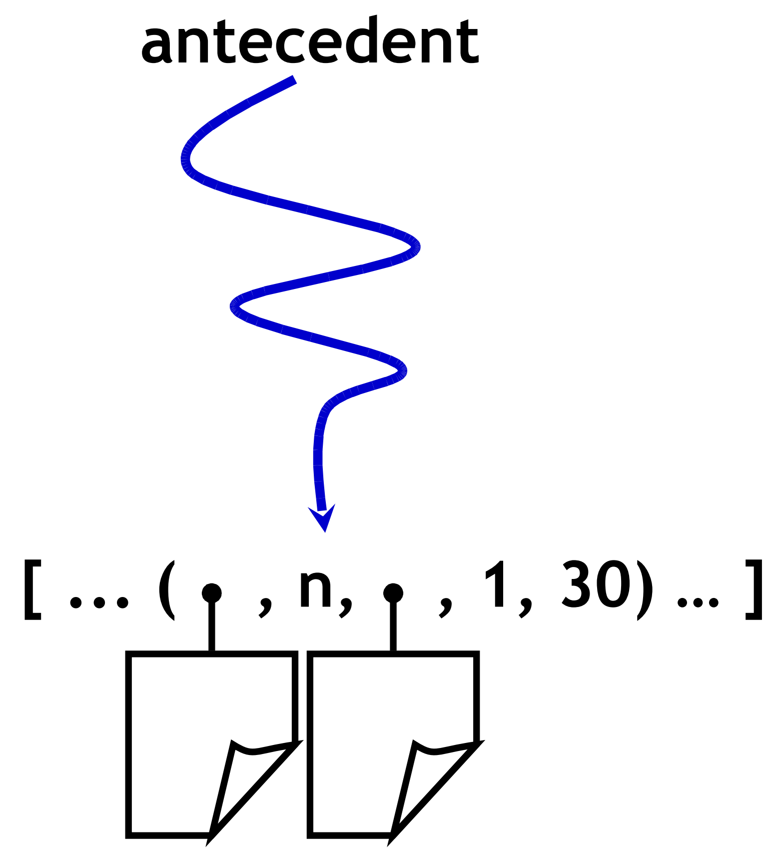
not(Seq(g)) or ##f ($past(n,f-s) == Exp E)

Use of Reflection

Normal evaluation



Reflective overloading



Experimental Results

36 µop groups

1,035 µops

3,161 SVA checkers

global assumptions = **3,061**

constant tuples = **471**

equality constraints = **84**

173 cluster-level tests

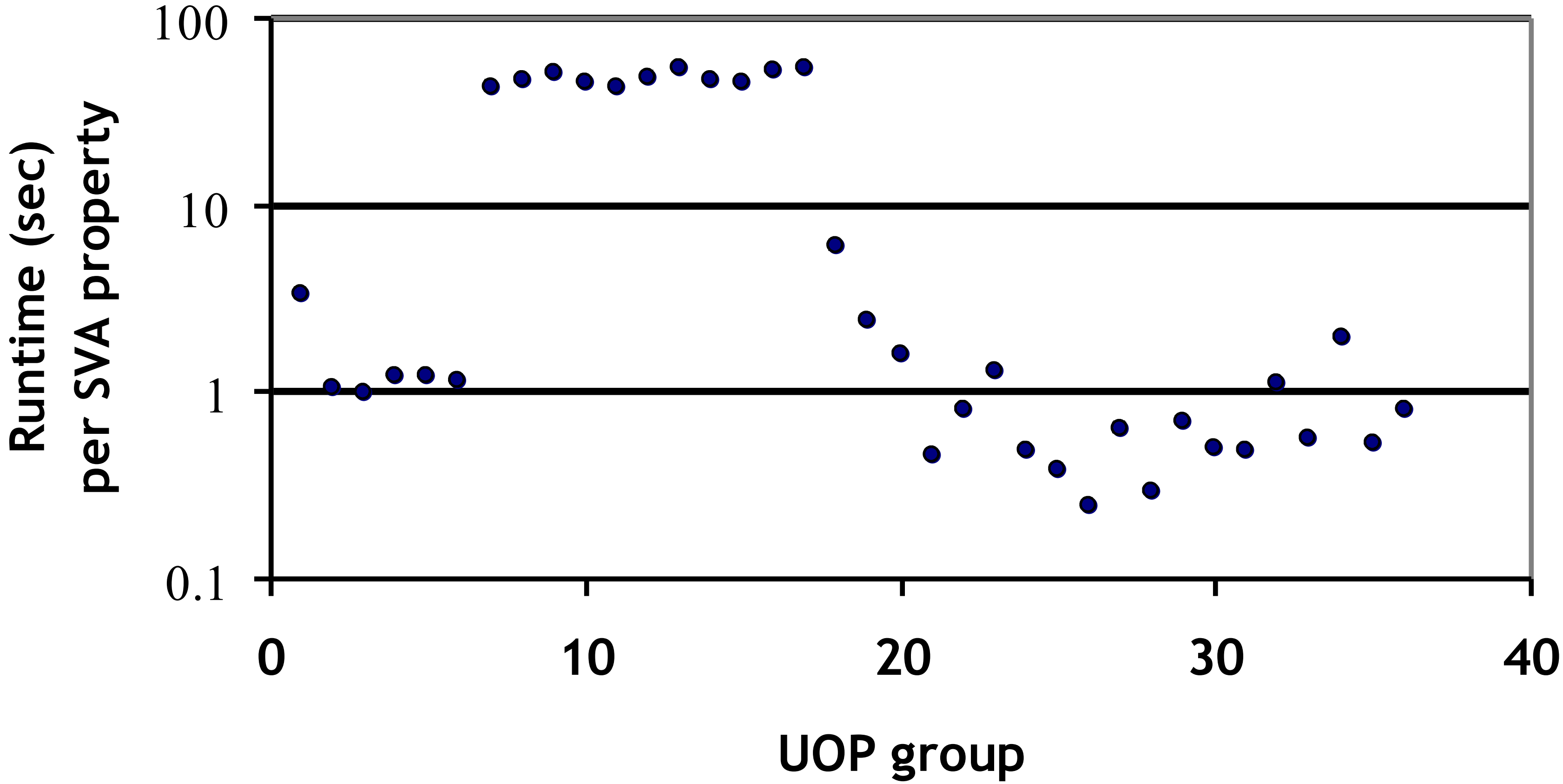
unused variables = **10s**

wrong assumptions = **10s**

1,100 core-level tests

bugs (microcode) = **2**

Runtimes




Thank You