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)

\[ P \vdash A \Rightarrow C \]

• Guarantee (SVA)

Big processor EXE proofs
• improve assumptions
• catch environment bugs
Symbolic Trajectory Evaluation

\[ f := \begin{cases} n \text{ is } 0 \\ n \text{ is } 1 \\ f_1 \text{ and } f_2 \\ N f \\ P \rightarrow f \end{cases} \]

stimulus \[ \Rightarrow \]

response

\[ A \Rightarrow C \]
Example

\[ \overline{a} \rightarrow \text{a is 0 and } \overline{b} \rightarrow \text{b is 0 } \Rightarrow \text{ out is 0} \]

\[ \text{a is 0 } \Rightarrow \text{ out is 0} \]

\[ \text{b is 0 } \Rightarrow \text{ out is 0} \]
Symbolic Simulation

\[ n \text{ is } E = E \rightarrow (n \text{ is 1}) \text{ and } \overline{E} \rightarrow (n \text{ is 0}) \]

\[ \begin{align*}
\text{(a is } x) \text{ and (b is } y) \text{ and (c is } z) & \Rightarrow o \text{ is } x \land y \land z
\end{align*} \]
Symbolic Indexing

\[ \neg p \land \neg q \rightarrow (a \text{ is } 0) \text{ and} \]
\[ \neg p \land q \rightarrow (b \text{ is } 0) \text{ and} \]
\[ p \land \neg q \rightarrow (c \text{ is } 0) \text{ and} \]
\[ p \land q \rightarrow (a \text{ is } 1) \text{ and } (b \text{ is } 1) \text{ and } (c \text{ is } 1) \]
\[ \Rightarrow \]
\[ \neg(p \land q) \rightarrow (o \text{ is } 0) \land \]
\[ (p \land q) \rightarrow (o \text{ is } 1) \]
Environmental Constraints

• Conditional verification

\[ \text{P}[\text{xs}] \models \text{A}[\text{xs}] \Rightarrow \text{C}[\text{xs}] \]

• Parametric representation

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

• Efficient verification

\[ \text{A}[\text{fs}[\text{vs}]] \Rightarrow \text{C}[\text{fs}[\text{vs}]] \]
Translation to SVA?

• Easy case

\[ x \lor y \models a \text{ is } x \text{ and } b \text{ is } y \Rightarrow \ldots \quad a \mid \mid \ b \]

• Harder...

\[ R[z] \models P \rightarrow (a \text{ is } z) \text{ and } Q \rightarrow (b \text{ is } z) \Rightarrow \ldots \]
Machine Representation - 5 Tuples

\[(\text{guard, node, value, start, end})\]

\[
f := n \text{ is 0}
\]

| \( n \text{ is 1} \) & \((P \rightarrow a \text{ is } x) \text{ and } (P \rightarrow N(a \text{ is } x))\) \\
| \( f_1 \text{ and } f_2 \) & \((P, a, x, 0, 2)\) \\
| \( Nf \) & \n
| \( P \rightarrow f \) & \n
STE Proof Environment - SVA Guarantee

\[ P \models A \Rightarrow C \]

timed

global

restrictions
ignore signals
ignore behaviours
not trigger or checker

assumptions
how inputs driven
input constraints
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

\[ \text{node}(x,g) = \$\text{past}(n,f-s) \]
Translating Boolean Constraints

\[ P \ - \ support = \{x_1, \ldots, x_n\} \]

\[ \theta = \text{choose one node}(x_i, g_i) \text{ for each } x_i. \]

\[ \exp(P, \theta) = (g_1 \theta \&\& \ldots \&\& g_j \theta) \leq P \theta \]

\[ \text{Exp}(P) = (\exp(P, \theta_1) \&\& \ldots \&\& (\exp(P, \theta_k)) \]

\[ \text{Seq}(P) = \#\#f \text{ Exp}(P, \theta) \]
Implicit Equality Constraints

\[(g_1, n_1, x, s_1, e_1)\]
\[(g_2, n_2, x, s_2, e_2)\]

##f \quad \text{Exp}(g_1 \land g_2) \leq \text{past}(n_1, f-e_1) \equiv \text{past}(n_2, f-e_2)
Per-Tuple Stability Constraints

\[(g, n, x, s, e)\]
\[\neg(\text{Seq}(g)) \text{ or } (\#\#s+1(\text{stable}(n))[*e-s-1])\]

\[(g, n, E, s, e)\]
\[\neg(\text{Seq}(g)) \text{ or } \#\#f (\text{past}(n,f-s) == \text{Exp } E)\]
Use of Reflection

Normal evaluation

Reflective overloading

antecedent

[ ... ( , n, , 1, 30) ... ]

[ ... ( , n, , 1, 30) ... ]
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

Runtime (sec) per SVA property

UOP group
Thank You