BCTCS11, Swansea, 2–5 April, 1995

Symbolic Verification of Hardware Systems

Howard Barringer   Graham Gough   Alan Williams

Department of Computer Science, University of Manchester

Brian Monahan, Harlequin Ltd.

Partners: DRA (Malvern), Harlequin Ltd (Cambridge)
Funded by DTI/SERC

CONTENTS

Background

Our approach has been strongly guided by

> …embedding formal methods within the hardware designer's normal environment …

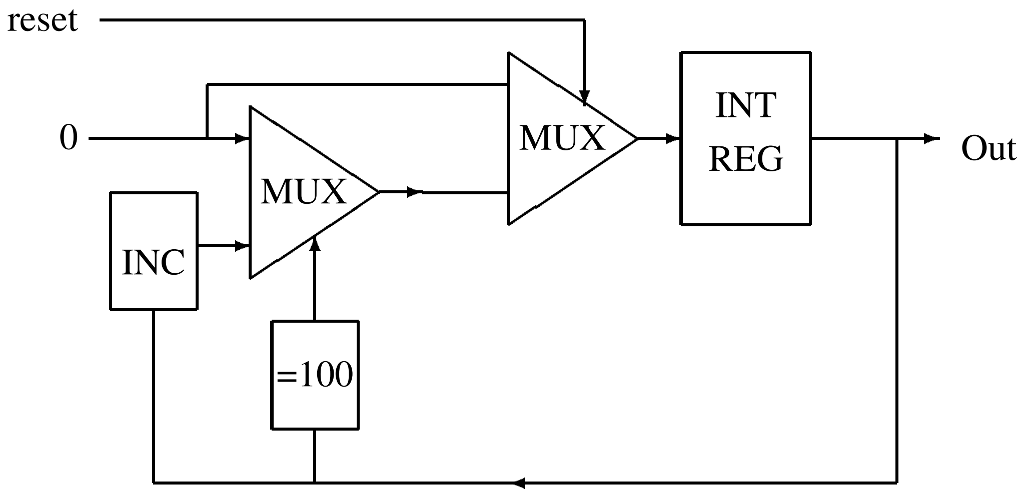
Design representation: ELLA Hardware Description Language

- synchronous communication
- simultaneous input/output

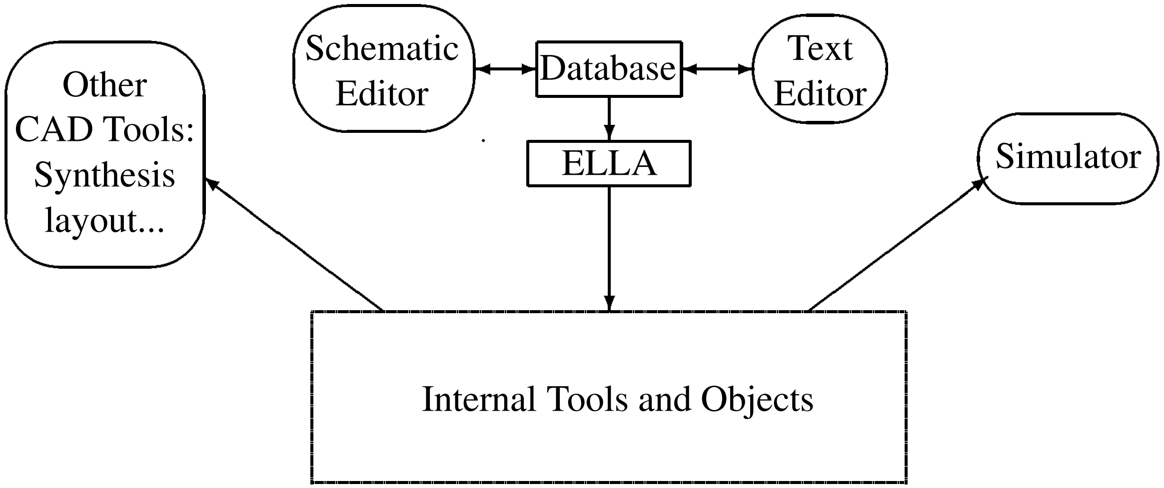
Background (continued)

```
TYPE bool = NEW (t | f).        TYPE data = NEW i/(0..1000).

FN DEL = (data) -> data: IDELAY(i/0,1).
FN INC = (data: x) -> data: ARITH x+1.

FN COUNTER = (bool: reset) -> data:
BEGIN   MAKE DEL: x.
        LET new_x = CASE reset OF
                      t : i/0
                    ELSE CASE x OF
                          i/100 : i/0
                         ELSE INC(x)  ESAC ESAC.
        JOIN new_x -> x.
        OUTPUT x
END.
```

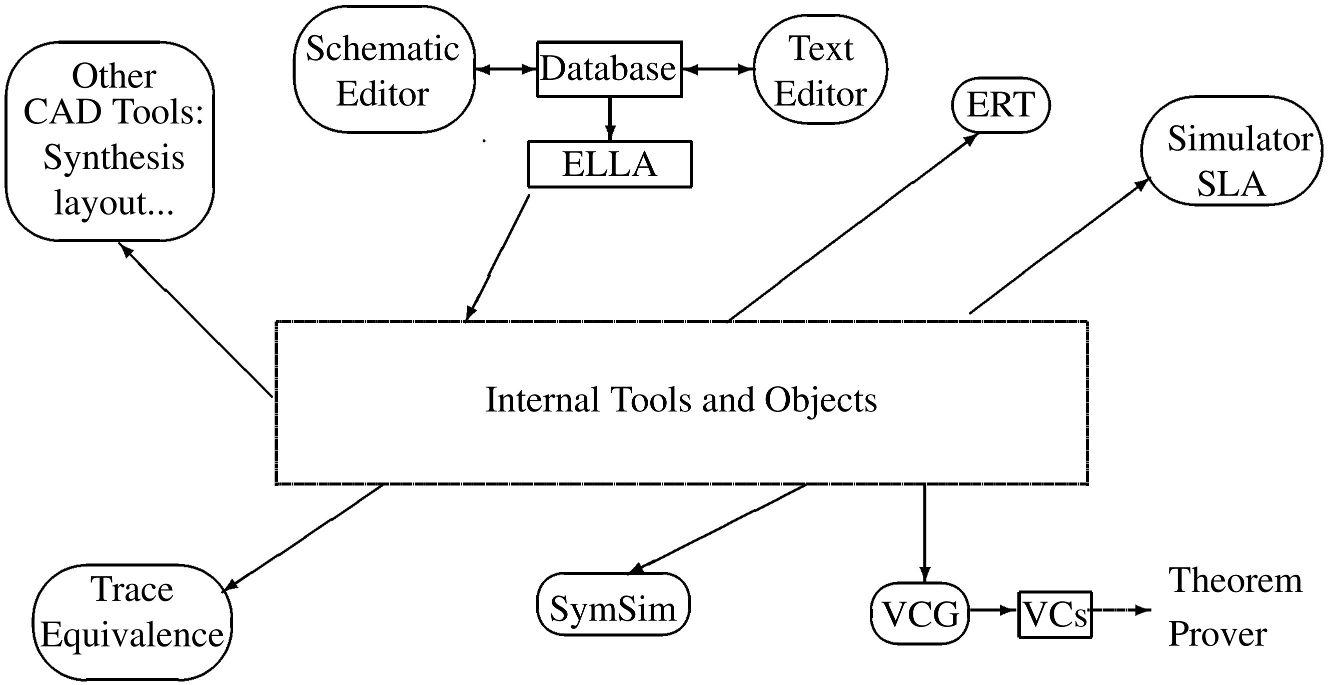
Background (continued)



Design + Verification Environments



Design + Verification Environments (continued)



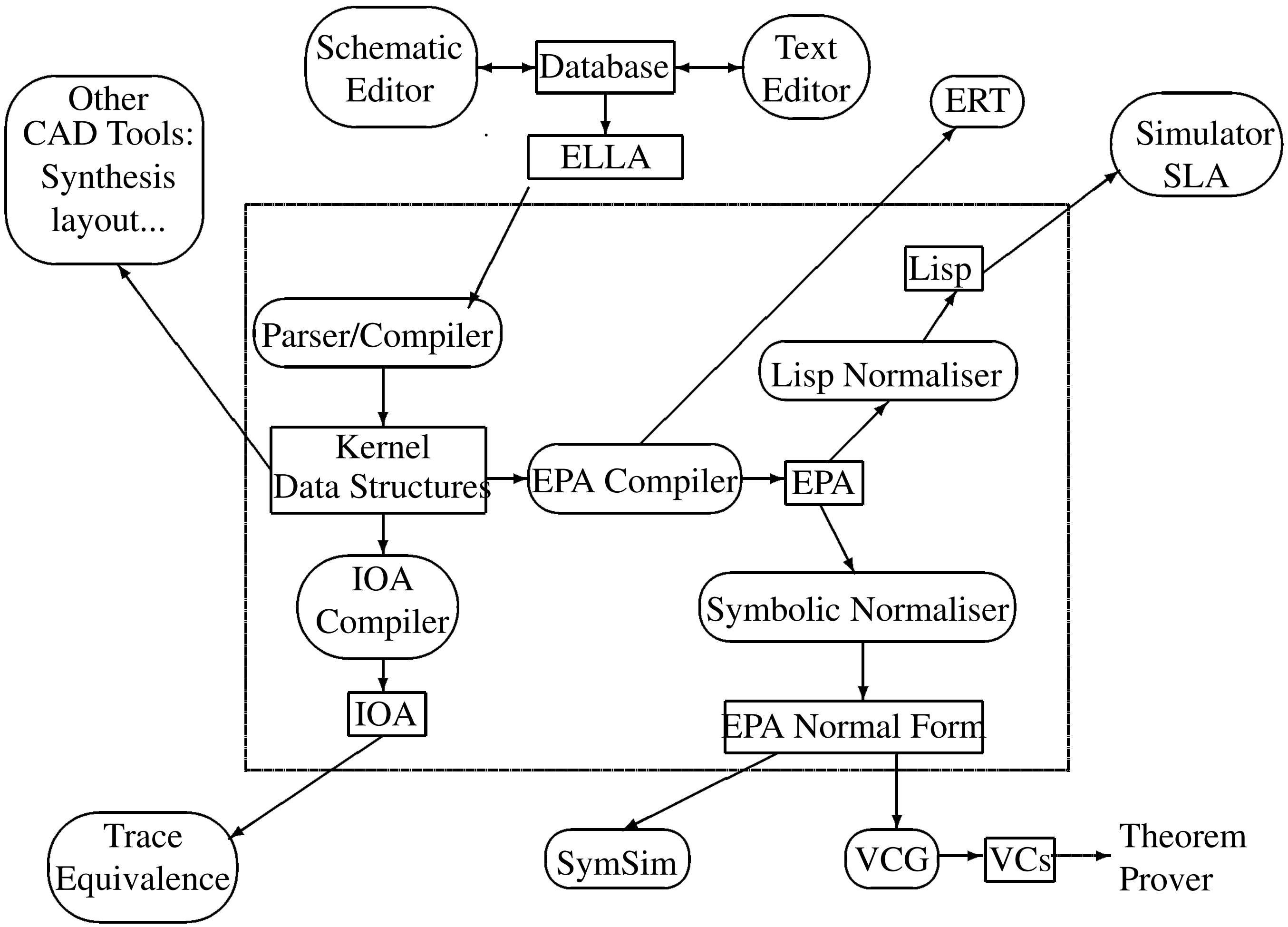
ERT: ELLA-level Rewrite Tool

Symsim: Symbolic Simulator

SLA: Soft Logic Analyser

VCG: Verification Condition Generator

Design + Verification Environments (continued)

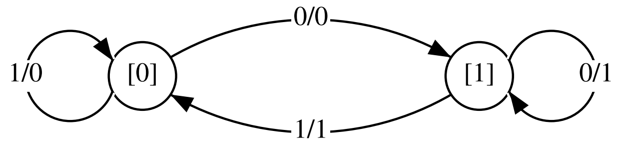


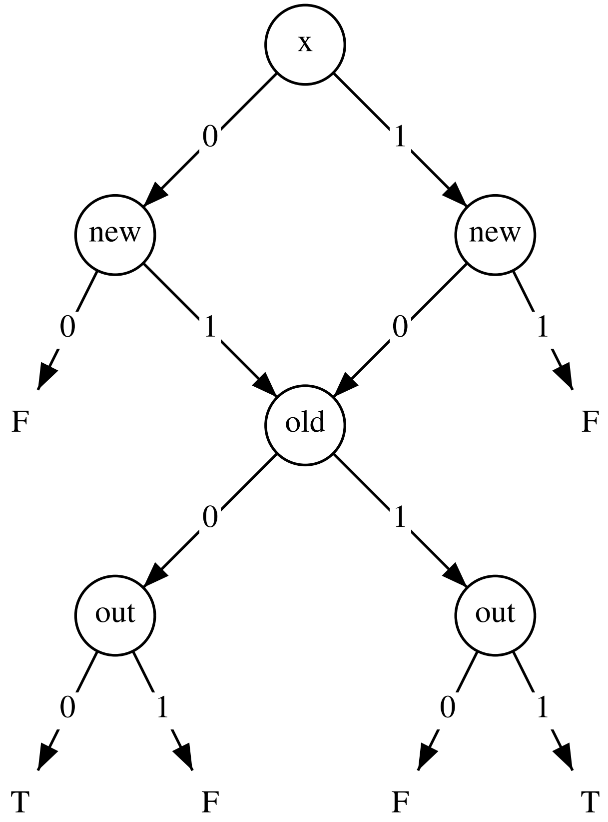
Semantics for ELLA #I

Use Input/Output Automaton (IOA)

- i/o maps as transition labels
- special IOA product operation
- compositional semantics for Core ELLA
- state-based trace-equivalence/containment
- also BDD encoding

Semantics for ELLA #I (continued)





Semantics for ELLA #II

Higher-level algebraic representation of IOAs:

ELLA Process Algebra (EPA):

- inspired by Milner + SCCS
- ELLA action algebra: "rich" actions
- specialised composition operator
- value-passing process calls

EPA

$pnm(xs_1, \ldots, xs_k)$	$\triangleq$	P_1	Process Definition
$P \in Proc$	$::=$	$P_1 \mid P_2$	Specialised Product
	$\mid$	$\Box\, xs \bullet P$	Process Choice
	$\mid$	$P \backslash nm$	Hiding
	$\mid$	$\alpha :: P$	Action Prefix
	$\mid$	$pnm(x_1, \ldots, x_k)$	Process Call

$\alpha \in IOAct$, $xs = x\colon S$ for some set S.

$$\left(pnm_1(args_1) \mid \ldots \mid pnm_k(args_k) \right) \backslash \{nm_1, \ldots, nm_m\}$$

Deterministic Machine

$P(v\colon \mathbb{Z}\ ;\ reset\colon\mathbb{B}\ /\ out\colon\ \mathbb{Z})\ \underline{\triangle}$

$\quad \square\, a\colon\ \mathbb{B}\ \bullet\ reset(a)\ /\ out(r_P(a,v)) :: P(e_P(a,v)\ ;\ reset\ /\ out)$

output expression: $r_P(a,v) = a$

state expression: $e_P(a,v) =$ **case** a :

$t \rightarrow 0$

else case v :

$100 \rightarrow 0$

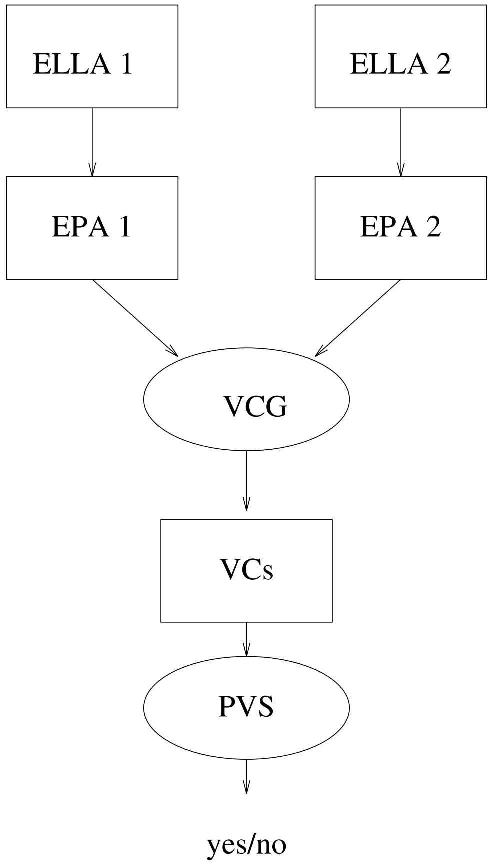
else $inc(v)$

end

end

Output domain: $O = \mathbb{Z}$ Initial value: $v = 0$

Verification Approaches



Verification Approaches (continued)

State-based:

- automated (+ve)
- combinatorial explosion ($-$ve)

Theorem-proving:

- high-level structured representation (+ve)
- proof obligation extraction/formulation ($-$ve)
- verification of VIPER: (e.g)
- case analysis + theorem-proving

Combined approach:

- “merge” approaches
- use knowledge of application domain
- extract (simple) proof obligations

Bisimulation

(Park and Milner)

$S \subseteq Proc \times Proc$ is a (strong) bisimulation if $(P,\ Q) \in S$ implies, for all $\alpha \in IOAct$:

Whenever $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in S$
Whenever $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \xrightarrow{\alpha} P'$ and $(P', Q') \in S$

P and Q are strongly bisimilar — $P \sim Q$ — if $(P,\ Q) \in S$ for some strong bisimulation S.

State Evolution

or... how to check for bisimulation...

$$\boxed{\text{St-Ev}}\ \frac{\begin{array}{l}\text{For some fixed } N \in \mathbb{N} : \\ \quad \text{(a)} \quad \Gamma \vdash AR_N(i_P, i_Q) \\ \quad \text{(b)} \quad \Gamma \vdash \forall\, \sigma_P,\ \sigma_Q\ \centerdot \\ \qquad\qquad AR_N(\sigma_P, \sigma_Q) \ \Rightarrow\ R_{N+1}(\sigma_P, \sigma_Q)\end{array}}{\Gamma \vdash P \sim Q}$$

How to find N?

Can N always be found?

State Evolution (continued)

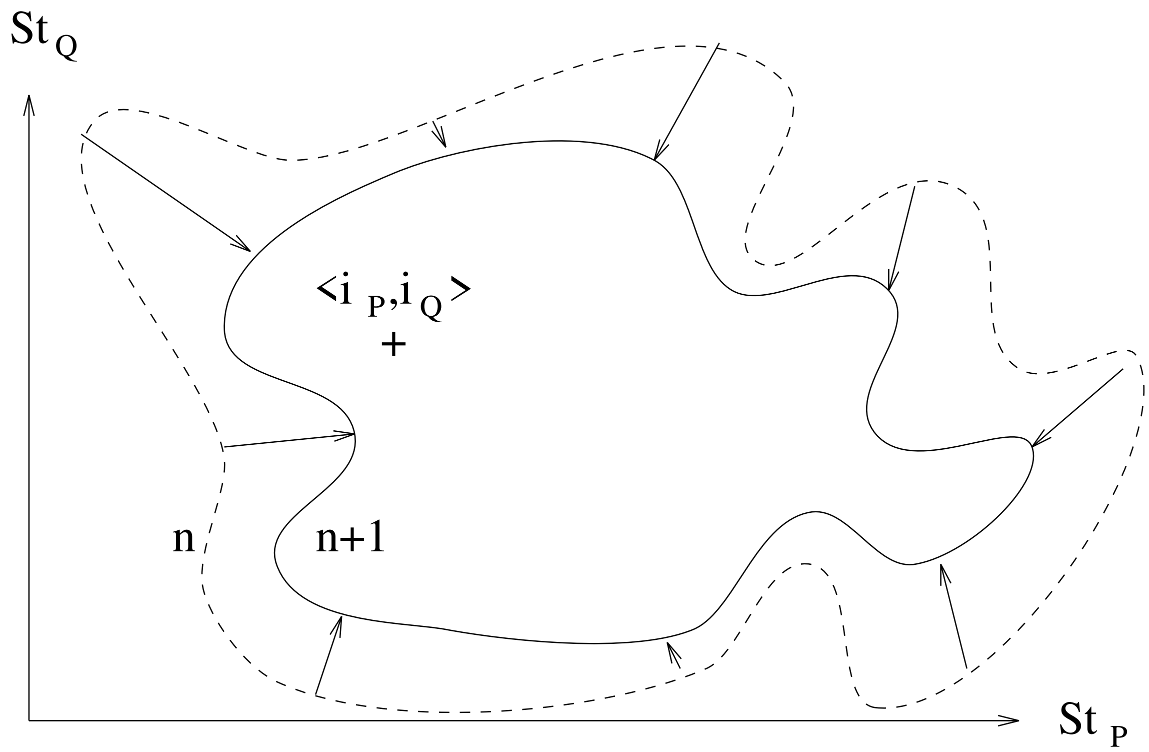
$$R_i(\sigma_P, \sigma_Q) \;\Leftrightarrow$$
$$\forall\, x_i,\;\cdots\; x_1 \cdot r_P(x_i,\, sp_i) = r_Q(x_i,\, sq_i)$$

where

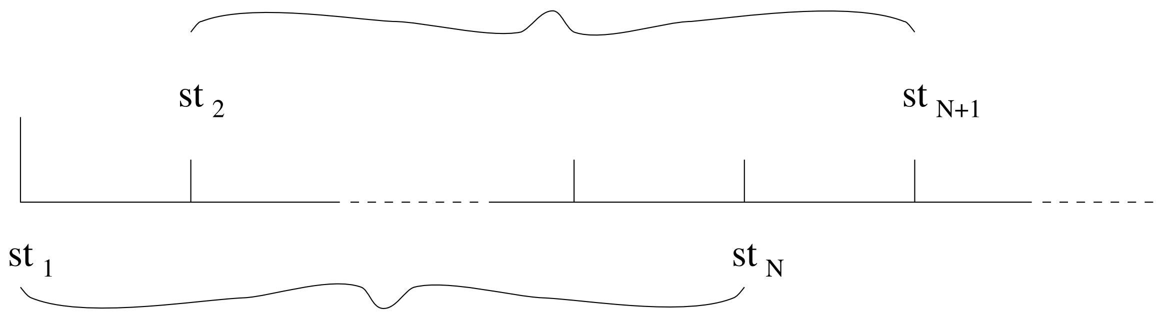
$$sp_i = e_P(x_{i\text{-}1}, sp_{i\text{-}1})$$
$$sq_i = e_Q(x_{i\text{-}1}, sq_{i\text{-}1})$$

$$AR_i(\sigma_P, \sigma_Q) \triangleq \bigwedge_{j=1}^{i} R_j(\sigma_P, \sigma_Q)$$

State Evolution (continued)



State Evolution (continued)



fixed N

symbolic representation:
universally-quantified state variables

but ... how to prove:

$$AR_N(i_P, i_Q)$$

$$AR_N(\sigma_P, \sigma_Q) \;\Rightarrow\; R_{N+1}(\sigma_P, \sigma_Q)$$

Example

$$
\begin{aligned}
& (R_1(v_P, v_Q) \quad \Rightarrow \quad R_2(v_P, v_Q)) \\
\Leftrightarrow \quad & \forall\, a_2, a_1 \cdot \\
& \qquad r_P(a_1, v_P) = r_Q(a_1, v_Q) \\
& \qquad\quad \Rightarrow\; r_P(a_2, e_P(a_1, v_P)) = r_Q(a_2, e_Q(a_1, v_Q)) \\
\Leftrightarrow \quad & ((v_P = v_Q) \quad \Rightarrow\; (e_P(a, v_P), e_Q(a, v_Q)))
\end{aligned}
$$

Example (continued)

$$
\begin{aligned}
e_P(a, v_P) = \textbf{case } a :& \\
& t \to 0 \\
& \textbf{else case } v_P : \\
& \quad 100 \to 0 \\
& \quad \textbf{else } inc(v_P) \\
& \quad \textbf{end} \\
& \textbf{end}
\end{aligned}
\qquad
\begin{aligned}
e_Q(a, v_Q) = \textbf{case } a :& \\
& t \to 0 \\
& \textbf{else case } v_Q : \\
& \quad 100 \to 42 \\
& \quad \textbf{else } inc(v_Q) \\
& \quad \textbf{end} \\
& \textbf{end}
\end{aligned}
$$

Verification Condition Generation

$$
\begin{array}{rcl}
(R_1(v_P, v_Q) & \Rightarrow & R_2(v_P, v_Q)) \\
(v_P = v_Q) & \Rightarrow & (\forall\, a \cdot e_P(a, v_P) = e_Q(a, v_Q))
\end{array}
$$

$$
\left(\begin{array}{ll}
 & a_P \in \mathbb{B} - \{t\} \\
\wedge & v_P \in \mathbb{Z} - \{100\} \\
\wedge & a_Q \in \mathbb{B} - \{t\} \\
\Rightarrow & inc(v_P) = inc(v_Q)
\end{array}\right)
\quad
\left(\begin{array}{ll}
 & a_P \in \mathbb{B} - \{t\} \\
\wedge & v_P \in \{100\} \\
\wedge & a_Q \in \mathbb{B} - \{t\} \\
\wedge & v_Q \in \{100\} \\
\Rightarrow & 0 = 42
\end{array}\right)
$$

$$
\left(\begin{array}{ll}
 & a_P \in \{t\} \\
\wedge & a_Q \in \mathbb{B} - \{t\} \\
\wedge & v_Q \in \{100\} \\
\Rightarrow & 0 = 42
\end{array}\right)
\quad
\left(\begin{array}{ll}
 & a_P \in \mathbb{B} - \{t\} \\
\wedge & v_P \in \{100\} \\
\wedge & a_Q \in \mathbb{B} - \{t\} \\
\wedge & v_Q \in \mathbb{Z} - \{100\} \\
\Rightarrow & 0 = v_Q + 1
\end{array}\right)
$$

Summary

Embedded verification tools

Semantics for commercial HDL:

- IOA
- EPA

Bisimulation Equvalence via

- State Evolution
- Verification Conditions

Semi-automated approach:

- automatic theorem generation
- ‘simple’ conditions to prove

WWW:

`http://www.cs.man.ac.uk/fmethods/projects/ella-project.html`