CHARME'95, Frankfurt

Formal Support for the ELLA Hardware Description Language

Howard Barringer   Graham Gough   Brian Monahan*   Alan Williams

Department of Computer Science, University of Manchester

Project Partners: Harlequin Ltd., DRA (Malvern)

Funded by: DTI/SERC (UK)

3rd October, 1995

THE UNIVERSITY
of MANCHESTER

*now with Harlequin Ltd.

Contents

- Overview
- Design Representation
- Formal Semantics I
- Formal Semantics II
- EPA
- Design Tools
- Implementation

Overview

Full semantics for commercial HDL:

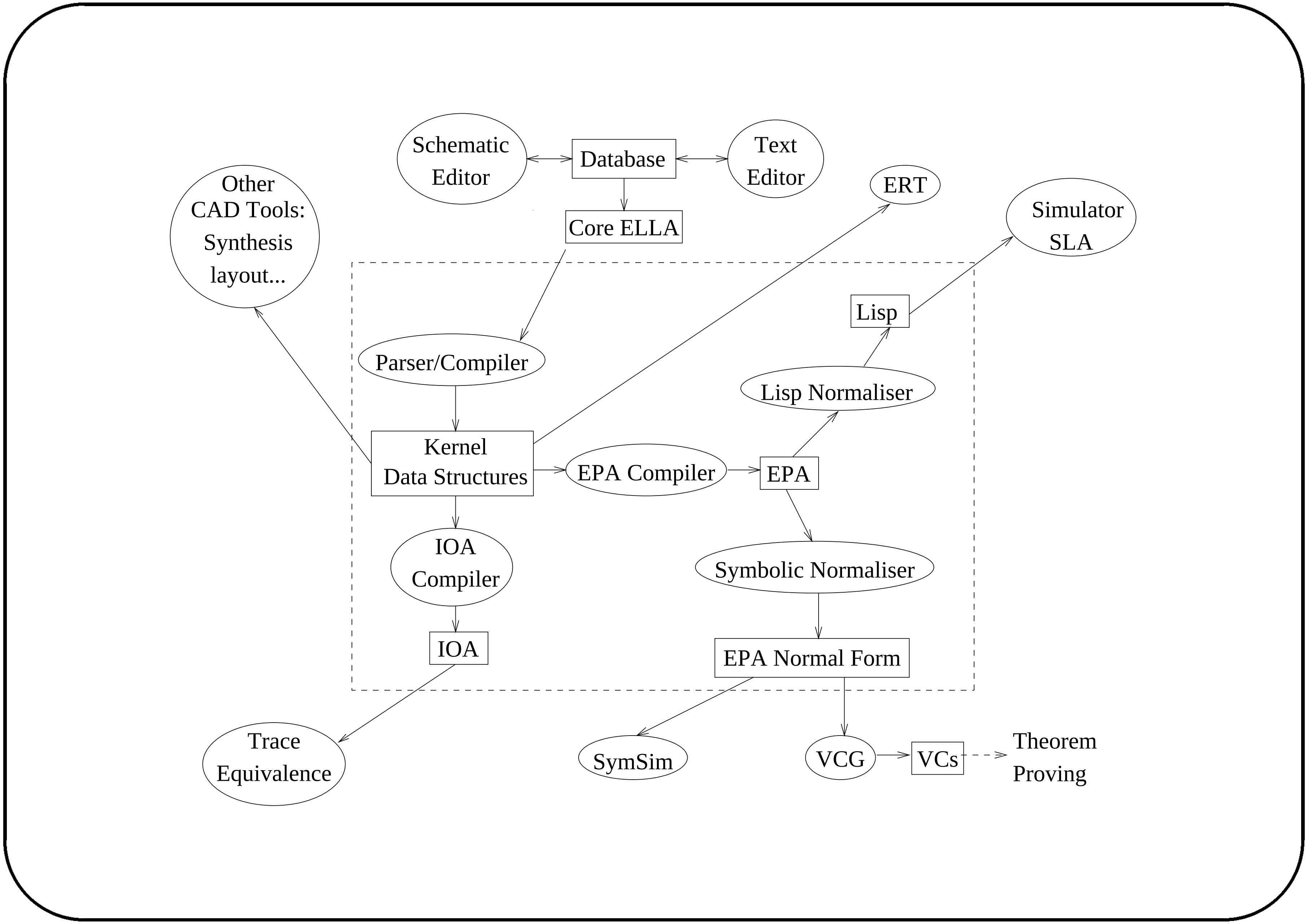
– design tools based on underlying formal semantic definition

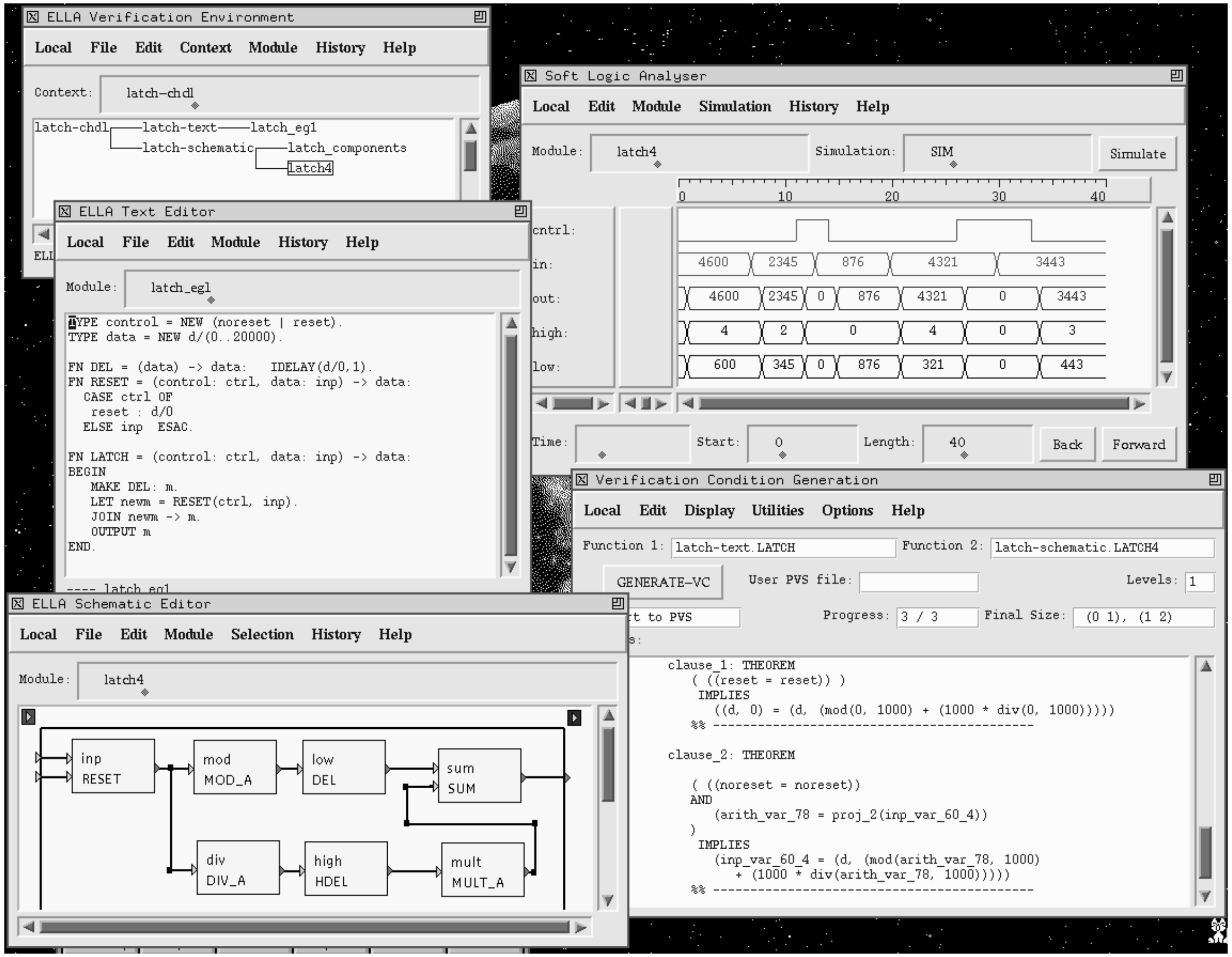
Embedded verification tools:

– semi-automated approach

Integrated design environment for ELLA:

– designer interacts with environment via HDL


Schematic Editor
Database
Text Editor
Other CAD Tools: Synthesis layout...
Core ELLA
ERT
Simulator SLA
Lisp
Parser/Compiler
Lisp Normaliser
Kernel Data Structures
EPA Compiler
EPA
IOA Compiler
Symbolic Normaliser
IOA
EPA Normal Form
Trace Equivalence
SymSim
VCG
VCs
Theorem Proving

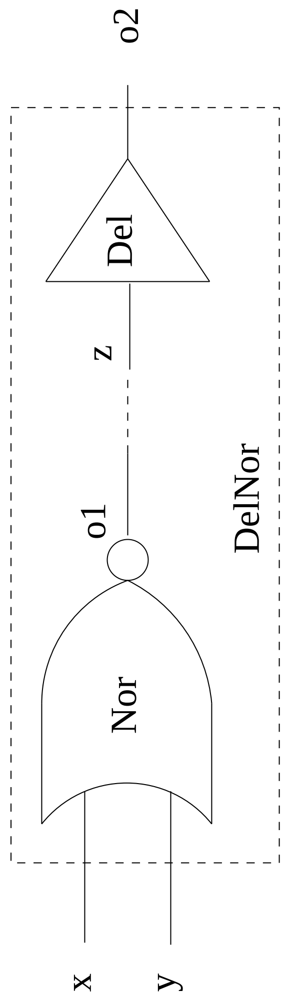

ELLA Verification Environment
Local File Edit Context Module History Help
Context: latch-chdl
latch-chdl latch-text latch_eg1
latch-schematic latch_components
latch4
ELLA Text Editor
Local File Edit Module History Help
Module: latch_eg1
TYPE control = NEW (noreset | reset).
TYPE data = NEW d/(0..20000).
FN DEL = (data) -> data: IDELAY(d/0,1).
FN RESET = (control: ctrl, data: inp) -> data:
CASE ctrl OF
reset : d/0
ELSE inp ESAC.
FN LATCH = (control: ctrl, data: inp) -> data:
BEGIN
MAKE DEL: m.
LET newm = RESET(ctrl, inp).
JOIN newm -> m.
OUTPUT m
END.
Soft Logic Analyser
Local Edit Module Simulation History Help
Module: latch4
Simulation: SIM
Simulate
0 10 20 30 40
cntrl:
in:
out:
high:
low:
4600 2345 876 4321 3443
4600 2345 0 876 4321 0 3443
4 2 0 4 0 3
600 345 0 876 321 0 443
Time:
Start: 0
Length: 40
Back
Forward
Verification Condition Generation
Local Edit Display Utilities Options Help
Function 1: latch-text.LATCH
Function 2: latch-schematic.LATCH4
GENERATE-VC
User PVS file:
Levels: 1
Progress: 3 / 3
Final Size: (0 1), (1 2)
clause_1: THEOREM
(((reset = reset)))
IMPLIES
((d, 0) = (d, (mod(0, 1000) + (1000 * div(0, 1000)))))
clause_2: THEOREM
(((noreset = noreset))
AND
(arith_var_78 = proj_2(inp_var_60_4))
)
IMPLIES
(inp_var_60_4 = (d, (mod(arith_var_78, 1000)
+ (1000 * div(arith_var_78, 1000)))))
ELLA Schematic Editor
Local File Edit Module Selection History Help
Module: latch4
inp RESET
mod MOD_A
low DEL
sum SUM
div DIV_A
high HDEL
mult MULT_A

Design Representation

ELLA Hardware Description Language

- developed for commercial design:
  - existing CAD tools for ELLA (e.g. synthesis)
- synchronous communication
- simultaneous input/output
- simple timing model
- small covering subset: Core ELLA

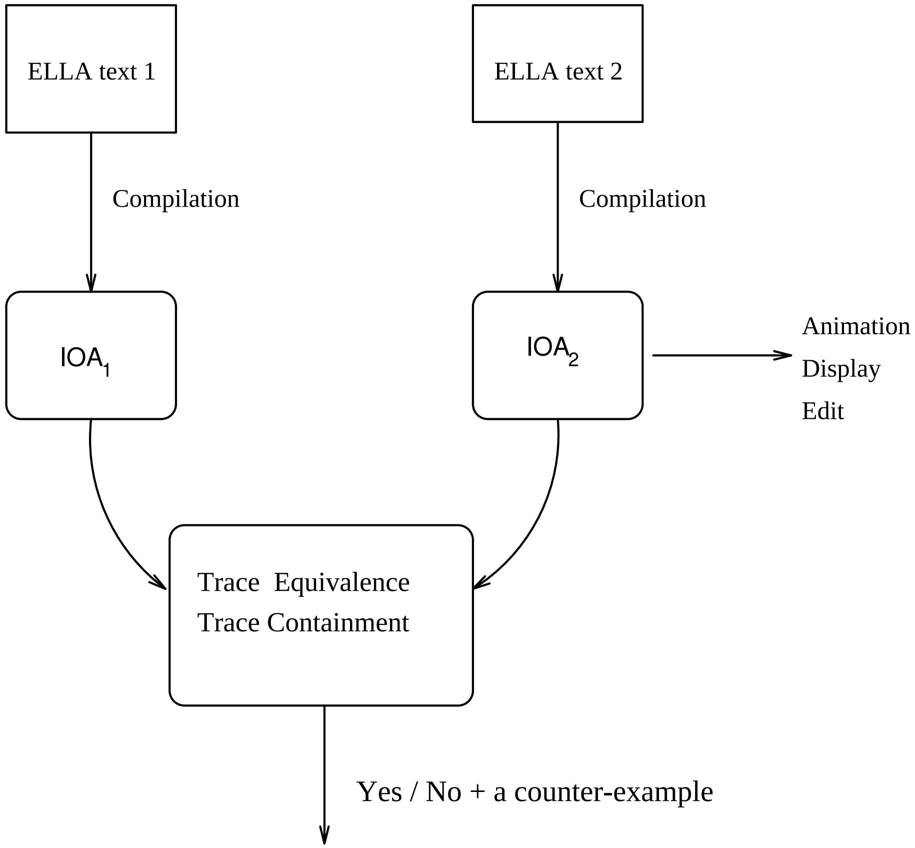
```
TYPE bool = NEW (f | t).

FN NOR = (bool: x, bool: y) -> bool:
  CASE (x,y) OF
     (f,f) : t
  ELSE f  ESAC.

FN DEL = (bool) -> bool: IDELAY(f,1).

FN DELNOR = (bool: x, bool: y)
            -> bool:
BEGIN
  MAKE NOR: o1.
  MAKE DEL: o2.
  JOIN (x,y) -> o1.
  JOIN o1 -> o2.
  OUTPUT o2
END.
```



Formal Semantics I

- set of Input/Output traces as semantic object
- use Input/Output Automata (IOAs)
  - i/o maps as transition labels
  - special IOA product operation
  - compositional semantics for Core ELLA
  - state-based trace-equivalence/containment
  - also BDD encoding
- complete semantics for ELLA

Automata-Theoretic Verification



Problems with State-based Verification

- Works well on small systems — of tens of thousands of states
- Combinatorial state space explosion is inherent
- Large(r) designs can be handled through:
  - BDD encodings
  - problem constraint
  - ELLA source level transformation

Formal Semantics II

Development of ELLA Process Algebra (EPA) for IO automata model:

- inspired by Milner's SCCS
- specialised composition operator
- value-passing process calls
- ELLA action algebra: "rich" actions
  - reduces amount of encoding required
- high-level, structured representation of IOA

EPA

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

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

action structure: $i_1(x_1) \cdots i_m(x_m) \mid o_1(e_1) \cdots o_n(e_n)$

EPA Semantics

Vacuum

$$\frac{}{P \xrightarrow{\boxtimes}_C P'}$$

Hiding

$$\frac{P_1 \xrightarrow{a}_C P_1'}{P_1 \backslash ch\text{-}set \xrightarrow{hide\text{-}act(a,\, ch\text{-}set)}_C P_1' \backslash ch\text{-}set}$$

Prefixing

$$\frac{}{a :: P_1 \xrightarrow{a}_C P_1}$$

ProcChoice

$$\frac{v \in set, \quad P_1 [\, v / x \,] \xrightarrow{a}_C P_1'}{\Box\, x\colon\ set \;\bullet\; P_1 \xrightarrow{a}_C P_1'}$$

ProcDefn

$$\frac{P_1 \underline{\triangle} P_2, \quad P_2 \xrightarrow{a}_C P'}{P_1 \xrightarrow{a}_C P'}$$

Product

$$\frac{P_1 \xrightarrow{a}_C P_1', \quad P_2 \xrightarrow{b}_C P_2'}{P_1 \mid P_2 \xrightarrow{a \star b}_C P_1' \mid P_2'}$$

Causal Product

Channels w, x, y, z

	Label 1	Label 2	Result
(a)	$w(3)$ / $y(1)$	$y(1)$ / $x(2)$	$w(3)$ / $(y(1), x(2))$
(b)	$w(3)$ / $y(1)$	$y(6)$ / $x(4)$	inconsistent on y

EPA Semantics for ELLA

Unit Delay:

$$\text{process } Del_P(c_1\colon bool \;/\; c_2\colon bool) \;\underline{\triangle}$$

$$udel_{[bool]}(\mathsf{ff} \;;\; c_1 \;/\; c_2)$$

using *generic* process definition :

$$\text{process } udel_{[ty]}(w\colon ty \;;\; c_1\colon ty \;/\; c_2\colon ty) \;\underline{\triangle}$$

$$\Box\, v\colon ty \;\bullet\; c_1(v) \;/\; c_2(w) :: udel_{[ty]}(v \;;\; c_1 \;/\; c_2)$$

```
TYPE bool = NEW (f | t).

FN NOR = (bool: x, bool: y) -> bool:
  CASE (x,y) OF
     (f,f) : t
  ELSE f  ESAC.

FN DEL = (bool) -> bool: IDELAY(f,1).
```

```
FN DELNOR = (bool: x, bool: y)
            -> bool:
BEGIN
  MAKE NOR: o1.
  MAKE DEL: o2.
  JOIN (x,y) -> o1.
  JOIN o1 -> o2.
  OUTPUT o2
END.
```

$$\Big(\; Nor_P(x,y \;/\; o1) \;\mid\; Udel_{[bool]}(v \,;\, o1 \;/\; o2) \; \Big) \backslash \{o1\colon bool\}$$

process $Nor_P(x,y\colon bool \;/\; o1\colon bool) \;\underline{\triangle}$

$$\Box\, a,b\colon bool \;\bullet\; x(a),y(b) \;/\; o1(nor(a,b)) :: Nor_P(x,y \;/\; o1)$$

process $Udel_{[bool]}(w\colon bool \,;\, o1\colon bool \;/\; o2\colon bool) \;\underline{\triangle}$

$$\Box\, v\colon bool \;\bullet\; o1(v) \;/\; o2(w) :: Udel_{[ty]}(v \,;\, o1 \;/\; o2)$$

Process Evolution

$$\Big(\ Nor_P(x,y\ /\ o1)\ \mid\ Udel_{[bool]}(\mathsf{ff}\ ;\ o1\ /\ o2)\ \Big)\backslash\{o1\colon\ bool\}$$

$$\xrightarrow{x(\mathsf{ff}),\,y(\mathsf{ff})\ /\ o2(\mathsf{ff})}$$

$$\Big(\ Nor_P(x,y\ /\ o1)\ \mid\ Udel_{[bool]}(\mathsf{tt}\ ;\ o1\ /\ o2)\ \Big)\backslash\{o1\colon\ bool\}$$

$$\xrightarrow{x(\mathsf{tt}),\,y(\mathsf{ff})\ /\ o2(\mathsf{tt})}$$

$$\Big(\ Nor_P(x,y\ /\ o1)\ \mid\ Udel_{[bool]}(\mathsf{ff}\ ;\ o1\ /\ o2)\ \Big)\backslash\{o1\colon\ bool\}$$

Deterministic Machines

$D = \langle init_P, r_P, e_P \rangle$

where

$init_P$	$\in St_P$	— `initial state`
r_P	$: (I \times St_P) \rightarrow O$	— `response function`
e_P	$: (I \times St_P) \rightarrow St_P$	— `evolution function`

Basis of (symbolic) simulation and verification tools.

Example

output expression: $r_P(\langle x, y \rangle, v) = v$

next-state expression: $e_P(\langle x, y \rangle, v) = nor(x, y)$

Input domain: $I = bool \times bool$ Output domain: $O = bool$

$St_P = bool$ Initial value: $v_0 = \mathsf{ff}$

Design Tools

- Ground simulation
- Symbolic simulation
- Symbolic Verification

Ground Simulation

- Evaluate response function r_P and next state function e_P
- Start from initial state $init_P$

$$\mathrm{ELLA} \xrightarrow{\text{formal semantics}} \mathrm{EPA} \xrightarrow{\text{rewrite}} \text{Deterministic Machine}$$

$$\xrightarrow{translate} \text{Lisp Functions} \xrightarrow{\text{Lisp Compiler}} \text{Compiled code}$$

Example

Process Call: $DelNor_P(\mathsf{ff}\ ;\ x, y\ /\ o2)$

$$DelNor_P(\mathsf{ff}\ ;\ x, y\ /\ o2) \xrightarrow{x(\mathsf{ff}), y(\mathsf{ff})\ /\ o2(\mathsf{ff})} DelNor_P(\mathsf{tt}\ ;\ x, y\ /\ o2)$$

$$\xrightarrow{x(\mathsf{tt}), y(\mathsf{ff})\ /\ o2(\mathsf{tt})} DelNor_P(\mathsf{ff}\ ;\ x, y\ /\ o2)$$

Symbolic Simulation

- non-ground terms as inputs (e.g. variables)
- use EPA Deterministic Machine Form
- apply behaviour-preserving EPA rewrite rules
- simplify/evaluate terms as far as possible
- control growth in term size:
  - only substitute ground terms for variables
  - user controlled substitution of LET and state variables by non-ground terms
- GT term manipulation package for EPA: share common sub-terms in internal representation

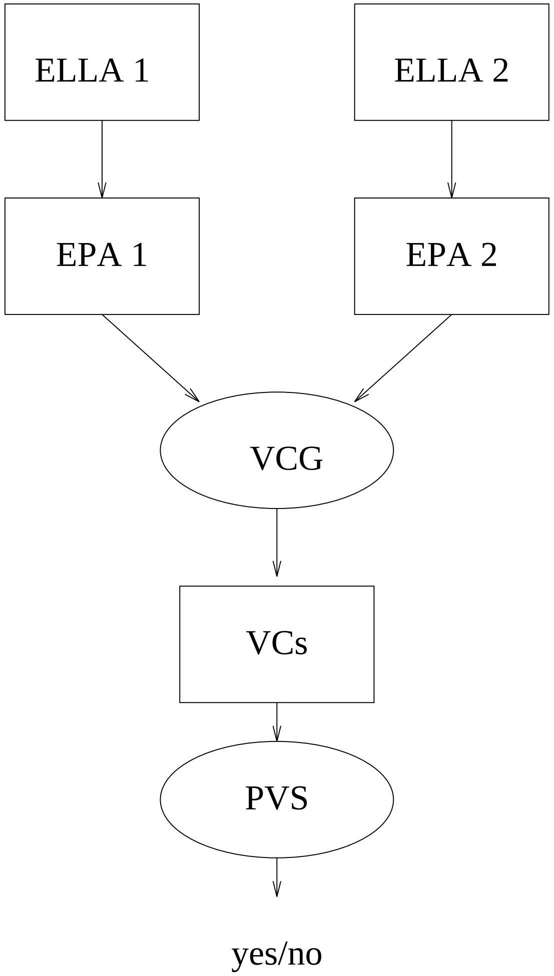
Example

```
TYPE bool = NEW (f|t).
TYPE data = NEW d/(0..10000).
FN DEL = (data) -> data: IDELAY(d/0,1).

FN LATCH = (bool: en, data: x) -> data:
BEGIN
  MAKE DEL: del.
  LET new = CASE en OF
              t : x
            ELSE del ESAC.
  JOIN new -> del.
  OUTPUT del
END.
```

```
CASE in1_6 OF
(t) : in2_6
ELSE CASE in1_5 OF
     (t) : in2_5
     ELSE CASE in1_4 OF
          (t) : in2_4
          ELSE CASE in1_3 OF
               (t) : in2_3
               ELSE CASE in1_2 OF
                    (t) : in2_2
                    ELSE CASE in1_1 OF
                         (t) : in2_1
                         ELSE CASE in1_0 OF
                              (t) : in2_0
                              ELSE d/0
                              ESAC
                         ESAC
                    ESAC
               ESAC
          ESAC
     ESAC
ESAC
```

```
CASE in1_6 OF
(t) : in2_6
ELSE del_3_6
ESAC
```

Formal Verification



Implementation

- LispWorks (Harlequin Ltd, UK) + CLOS + CAPI
- GT (University of Manchester, UK)
- ELLA-to-EPA compiler development facilitated by:
  - formal semantic definition
  - ‘layered’ structure

Summary

Integrated design environment for ELLA:

– designer interacts with environment via HDL

Full semantics for ELLA commercial HDL:

– IOA

– EPA

Design tools based on underlying formal semantic definition:

– compilers

– ground/symbolic simulation

– ELLA-level transformation

– symbolic verification

Embedded (semi-automated) verification tools

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