CHDL'95, Tokyo

C-9.2: A Design and Verification Environment for ELLA

Howard Barringer Graham Gough Brian Monahan Alan Williams

Department of Computer Science, University of Manchester

Matthew Arcus Andrew Armstrong, Harlequin Ltd.

Mike Hill, DRA (Malvern)

Funded by DTI/SERC (UK)

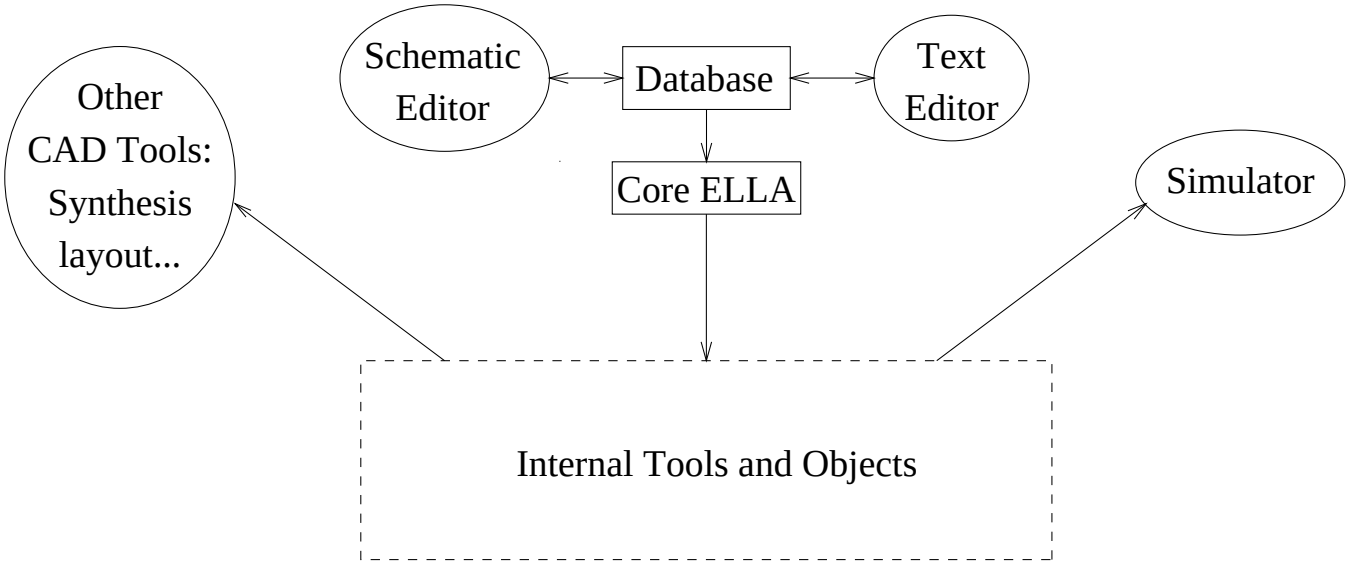
1st September, 1995

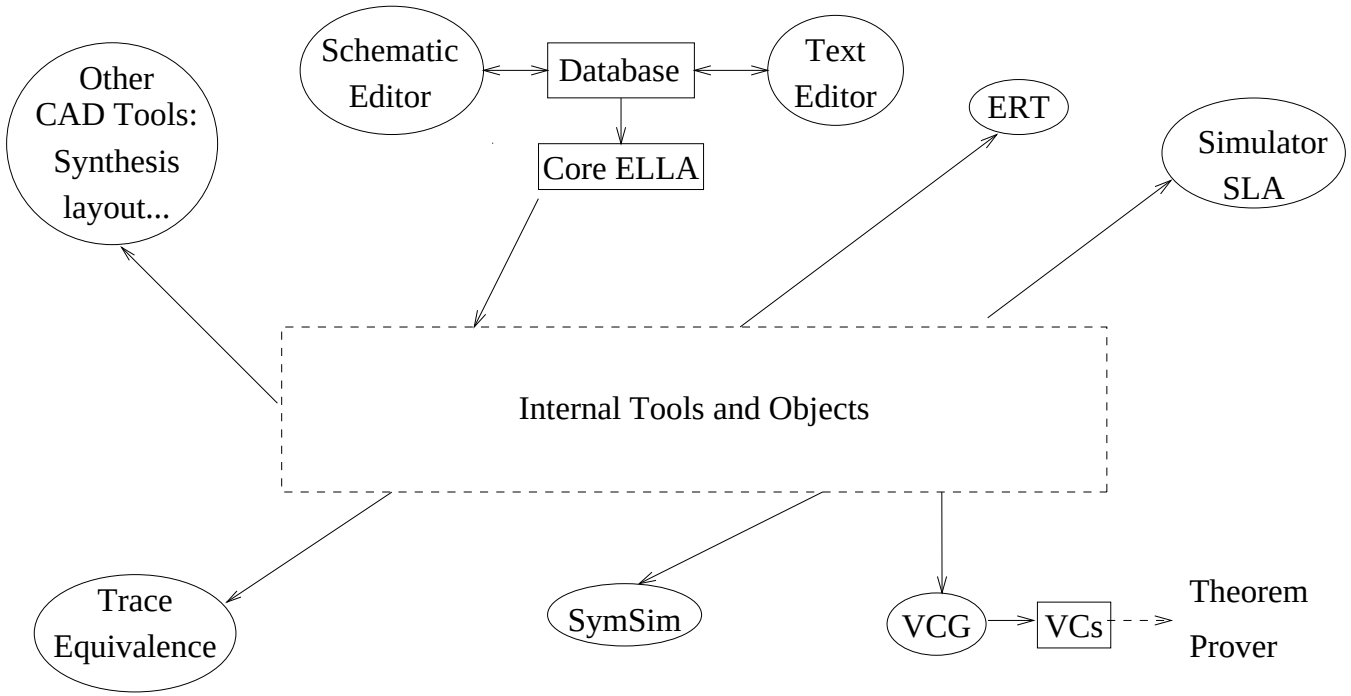
Contents

- Background
- Formal Semantics for ELLA
- Simulation
- Symbolic Simulation
- ELLA Level Design Transformation
- Formal Verification
- Implementation

Background

- utilisation of formal methods in design cycle:
  - embedding formal methods in design environment
  - design entry into verification tools
- tractable verification techniques
- feedback to designer


Schematic Editor
Database
Text Editor
Other CAD Tools: Synthesis layout...
Core ELLA
Simulator
Internal Tools and Objects



ERT: ELLA-level Rewrite Tool **Symsim**: Symbolic Simulator

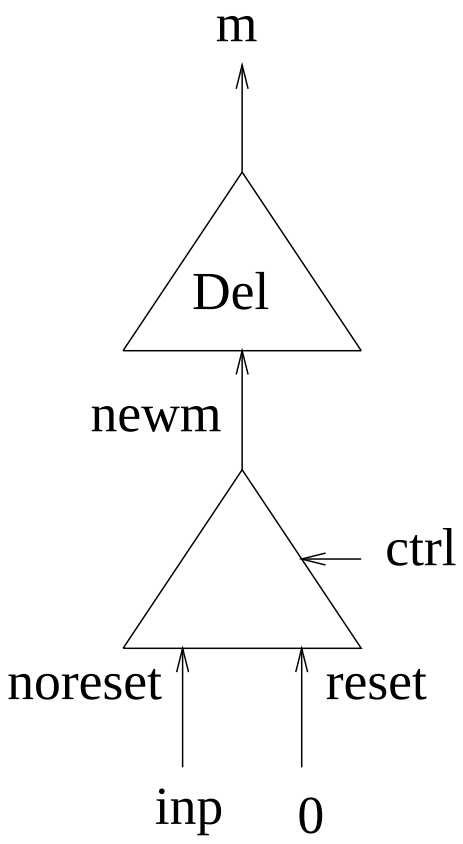
SLA: Soft Logic Analyser **VCG**: Verification Condition Generator

Design Representation

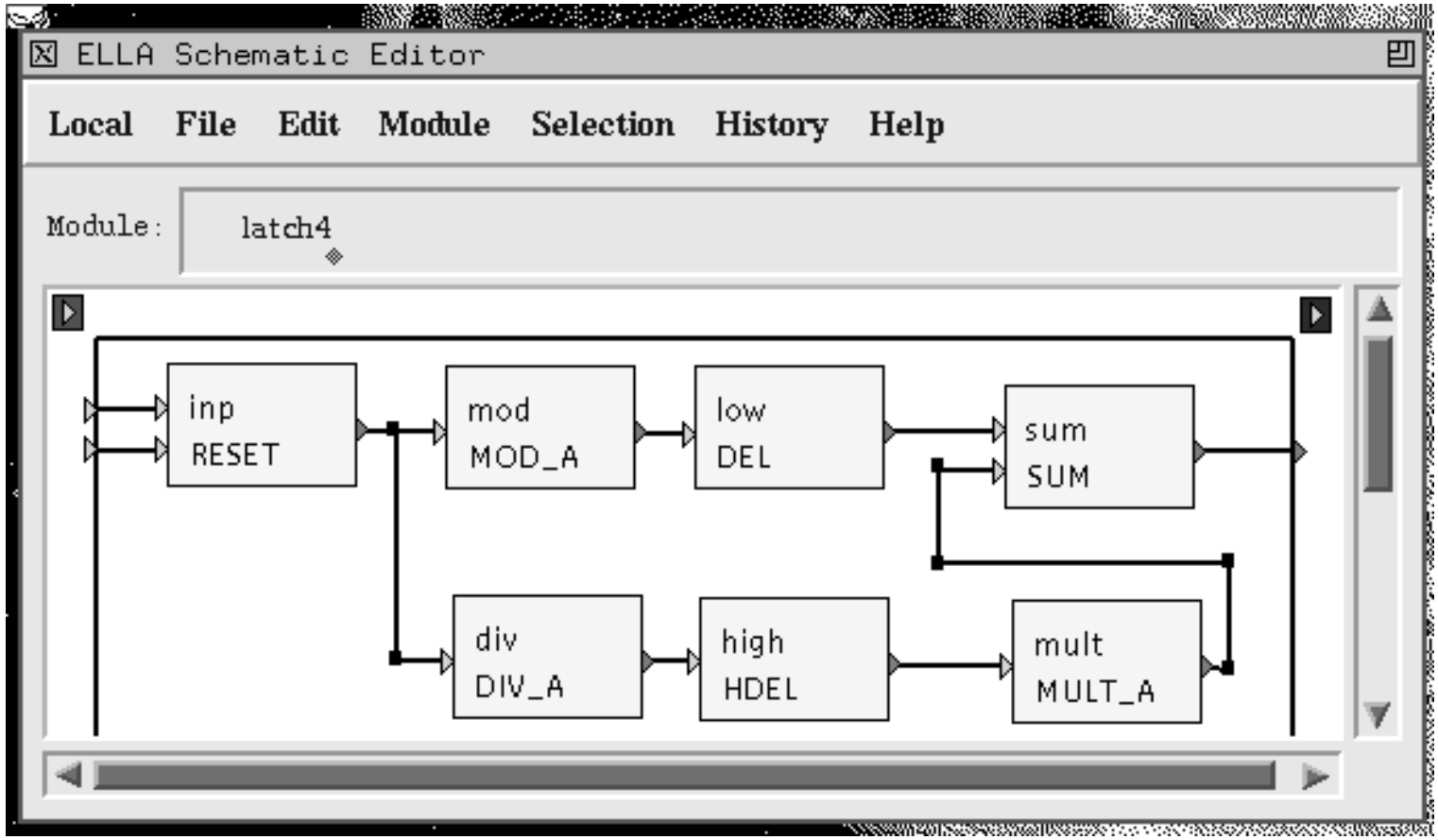
ELLA Hardware Description Language

- 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 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.
```



Alternative Implementation: Schematic



...and in VHDL

```
PACKAGE type_defs IS
 TYPE control IS (noreset, reset);
 SUBTYPE data IS integer range 0 to 20000;
END type_defs;

ENTITY fn_reset IS PORT (a: IN control; b: IN data; y: OUT data); END fn_reset;

ARCHITECTURE behavioural OF fn_reset IS
BEGIN
PROCESS
   BEGIN
    IF (a = reset)
    THEN y <= 0;
    ELSE y <= b;
    END IF;
  END PROCESS;
END behavioural;

ENTITY buf_gate IS PORT (a: IN data; y: OUT data); END buf_gate;

ARCHITECTURE behavioural OF buf_gate IS
BEGIN
PROCESS
  BEGIN
   y <= a AFTER 1 ns;
  END PROCESS;
END behavioural;

ENTITY latch IS  PORT (ctrl: IN control; inp: IN data; m: OUT data); END latch;

ARCHITECTURE structural OF latch IS
COMPONENT fn_reset
   PORT (a: IN control; b: IN data; y: OUT data); END COMPONENT;
COMPONENT buf_gate
   PORT (a: IN data; y: OUT data); END COMPONENT;
SIGNAL newm: data;
BEGIN
   c1: fn_reset PORT MAP(ctrl, inp, newm);
   c2: buf_gate PORT MAP(newm,m);
END structural;
```

Formal Semantics for ELLA

Our design tools based on underlying formal semantics:

- 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

Development of process calculus, specifically for IO automata model.

Higher-level algebraic representation of IOAs:

ELLA Process Algebra (EPA):

inspired by Milner's SCCS

specialised composition operator

value-passing process calls

ELLA action algebra: "rich" actions

– reduces amount of encoding required

```
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.
```

EPA Term:

$$(Sel1(inp\ /\ ch1)\ \mid\ Sel2(inp\ /\ ch3)\ \mid\ Const0(\ /\ ch2)\ \mid$$
$$Chooser_{[reset]}(ch1, ch2, ch3\ /\ ch4)\ \mid\ Udel(0\ ;\ ch4\ /\ out)$$
$$)\setminus\{ch1, ch2, ch3, ch4\}$$

Process Evolution

$$(Sel1(inp \,/\, ch1) \mid \cdots \mid Udel(22\,;\, ch4 \,/\, out)) \setminus \{ch1, ch2, ch3, ch4\}$$

$$\xrightarrow{inp(\langle noreset, 33\rangle) \,/\, out(22)}$$

$$(Sel1(inp \,/\, ch1) \mid \cdots \mid Udel(33\,;\, ch4 \,/\, out)) \setminus \{ch1, ch2, ch3, ch4\}$$

$$\xrightarrow{inp(\langle reset, 44\rangle) \,/\, out(33)}$$

$$(Sel1(inp \,/\, ch1) \mid \cdots \mid Udel(0\,;\, ch4 \,/\, out)) \setminus \{ch1, ch2, ch3, ch4\}$$

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: $r_P(\langle cntrl, in\rangle, m) = m$

state expression: $e_P(\langle cntrl, in\rangle, m) =$ **case** $cntrl$:

$reset \rightarrow 0$

else in

end

$control = \{reset, noreset\}$ $data = \{0, \ldots, 2000\}$

Input domain: $I = control \times data$ Output domain: $O = data$

$St_P = data$ Initial value: $m_0 = 0$

Simulation

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

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

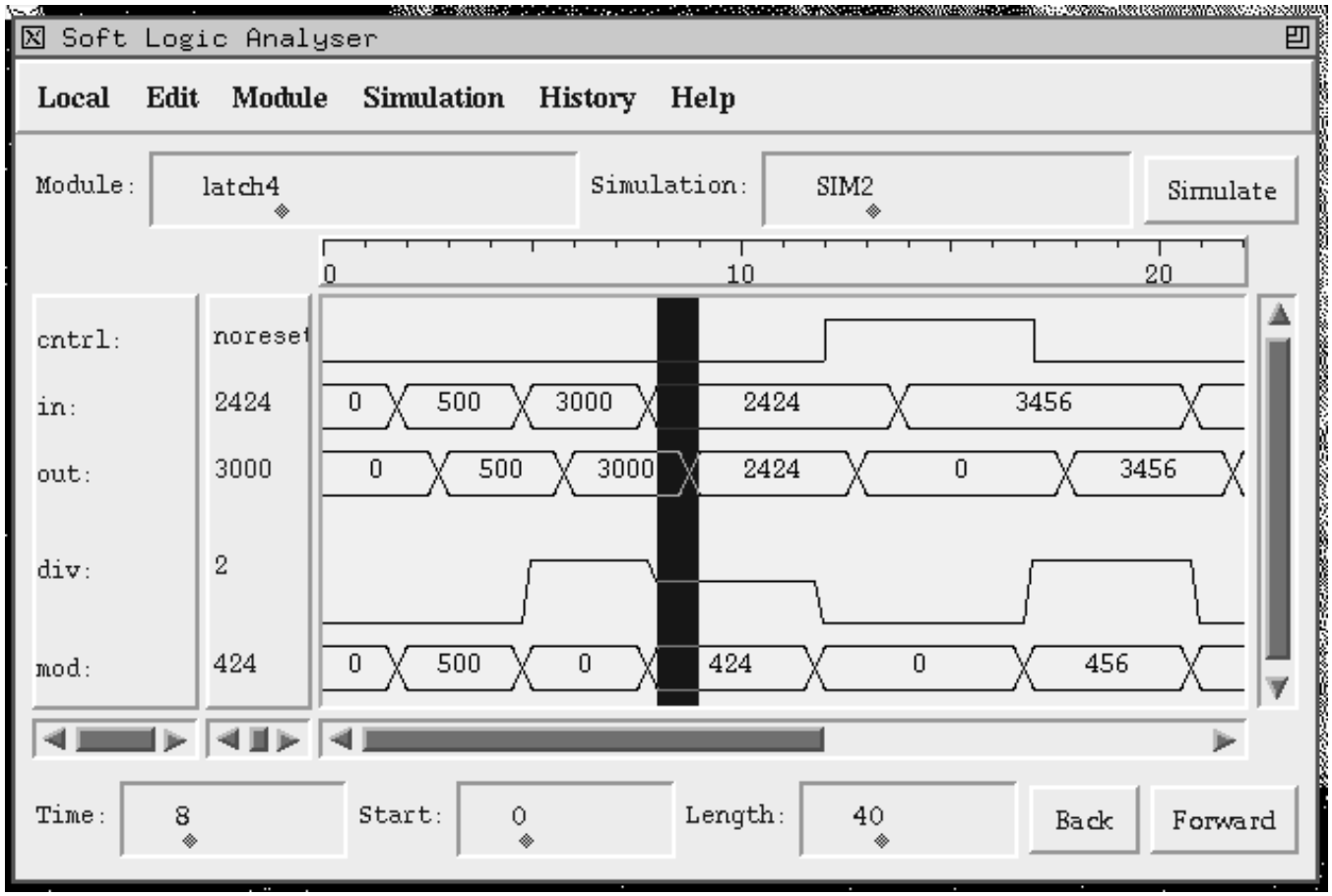
$$\longrightarrow \text{Lisp Functions} \xrightarrow{\text{Lisp Compiler}} \text{Compiled code}$$

Process Call: $P(22\ ;\ inp\ /\ out)$

$$P(22\ ;\ inp\ /\ out) \xrightarrow{inp(\langle noreset, 33\rangle)\ /\ out(22)} P(33\ ;\ inp\ /\ out)$$

$$\xrightarrow{inp(\langle reset, 44\rangle)\ /\ out(33)} P(0\ ;\ inp\ /\ out)$$

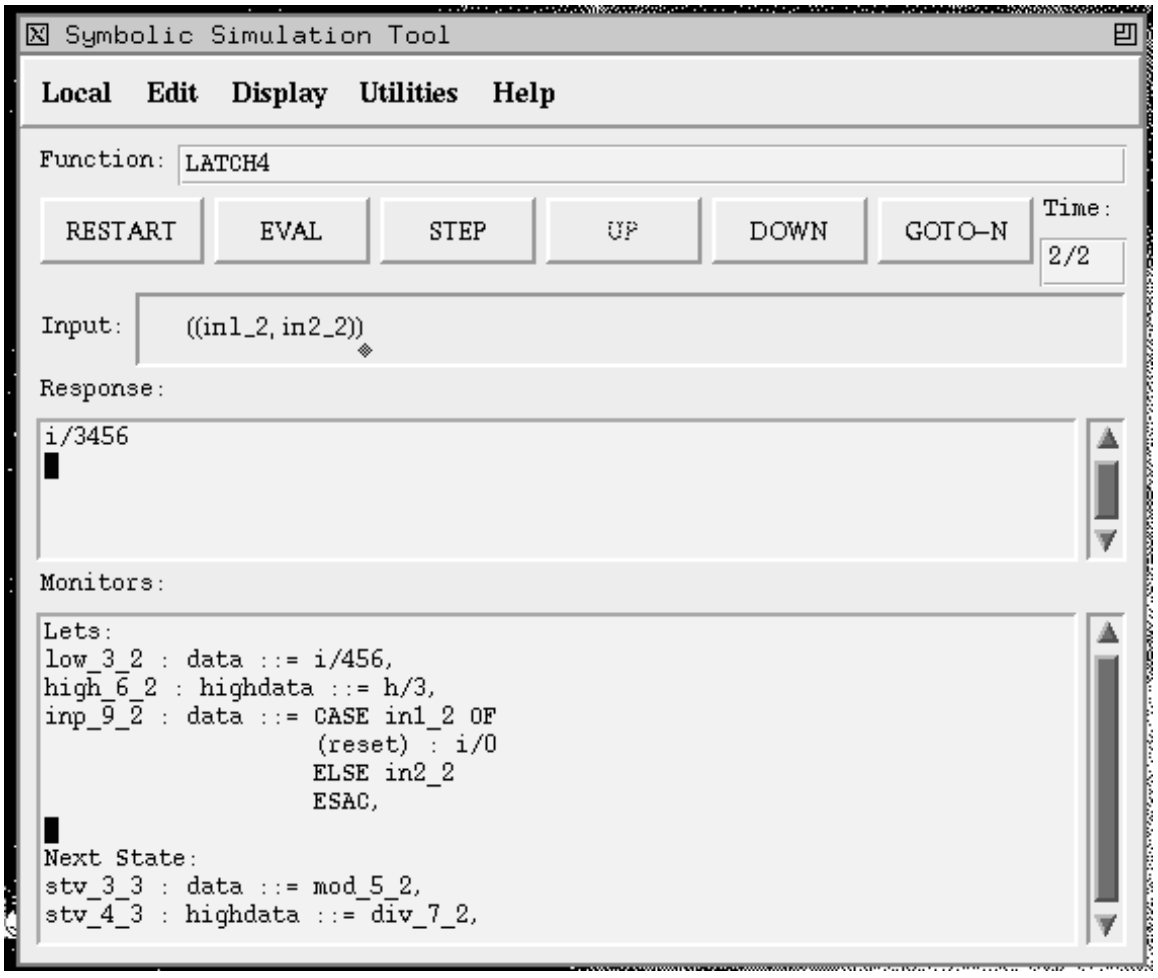
Soft Logic Analyser: Simulation of Integer Latch



Symbolic Simulation

- non-ground terms as inputs (e.g. variables)
- use EPA Deterministic Machine Form
- apply behaviour-preserving EPA rewrite rule
- 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

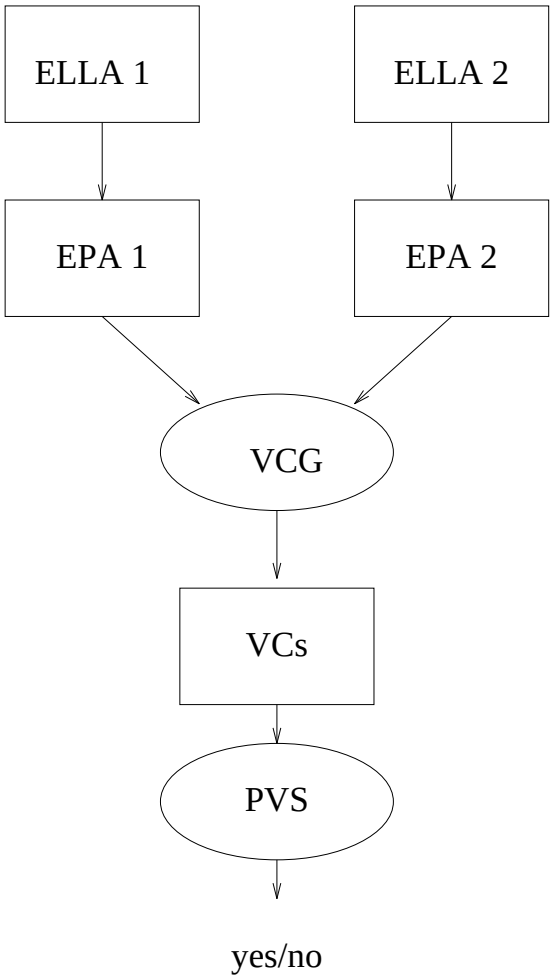
Symbolic Simulation



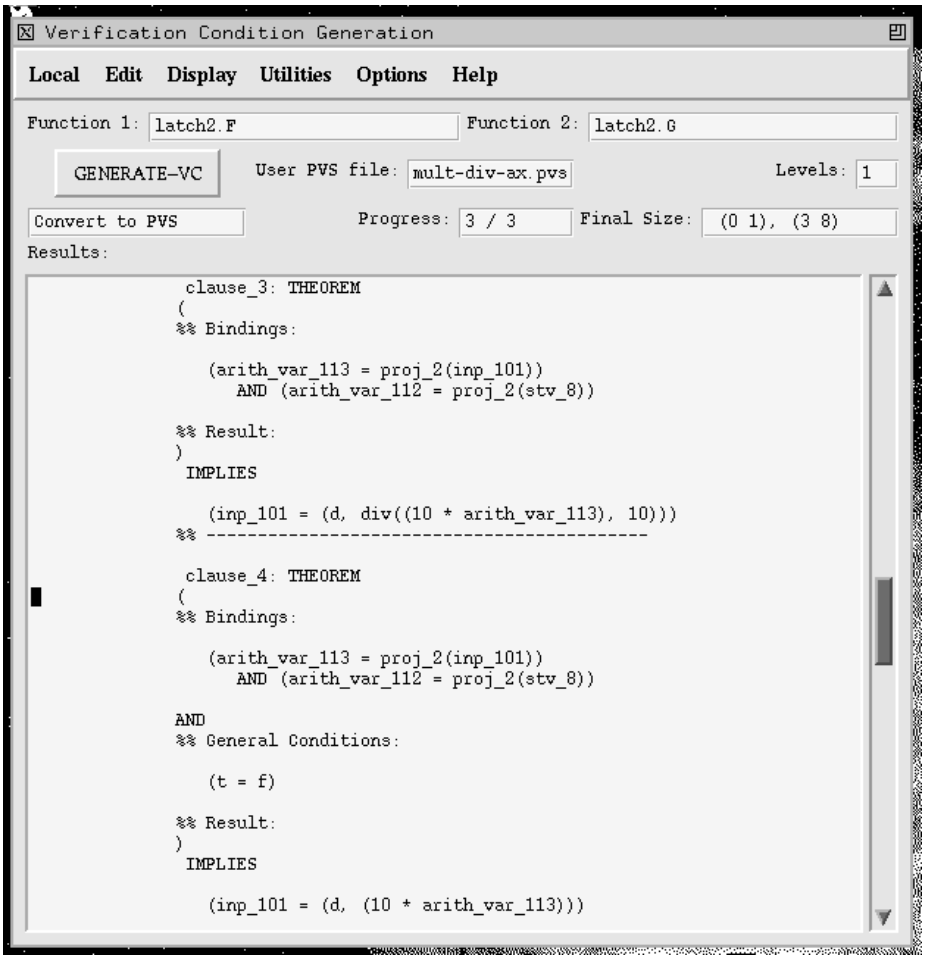
ELLA Level Design Transformation

- ELLA-level rewrite rules
- rules preserve behaviour, according to formal semantics

Formal Verification



VCG embedded in Design Environment



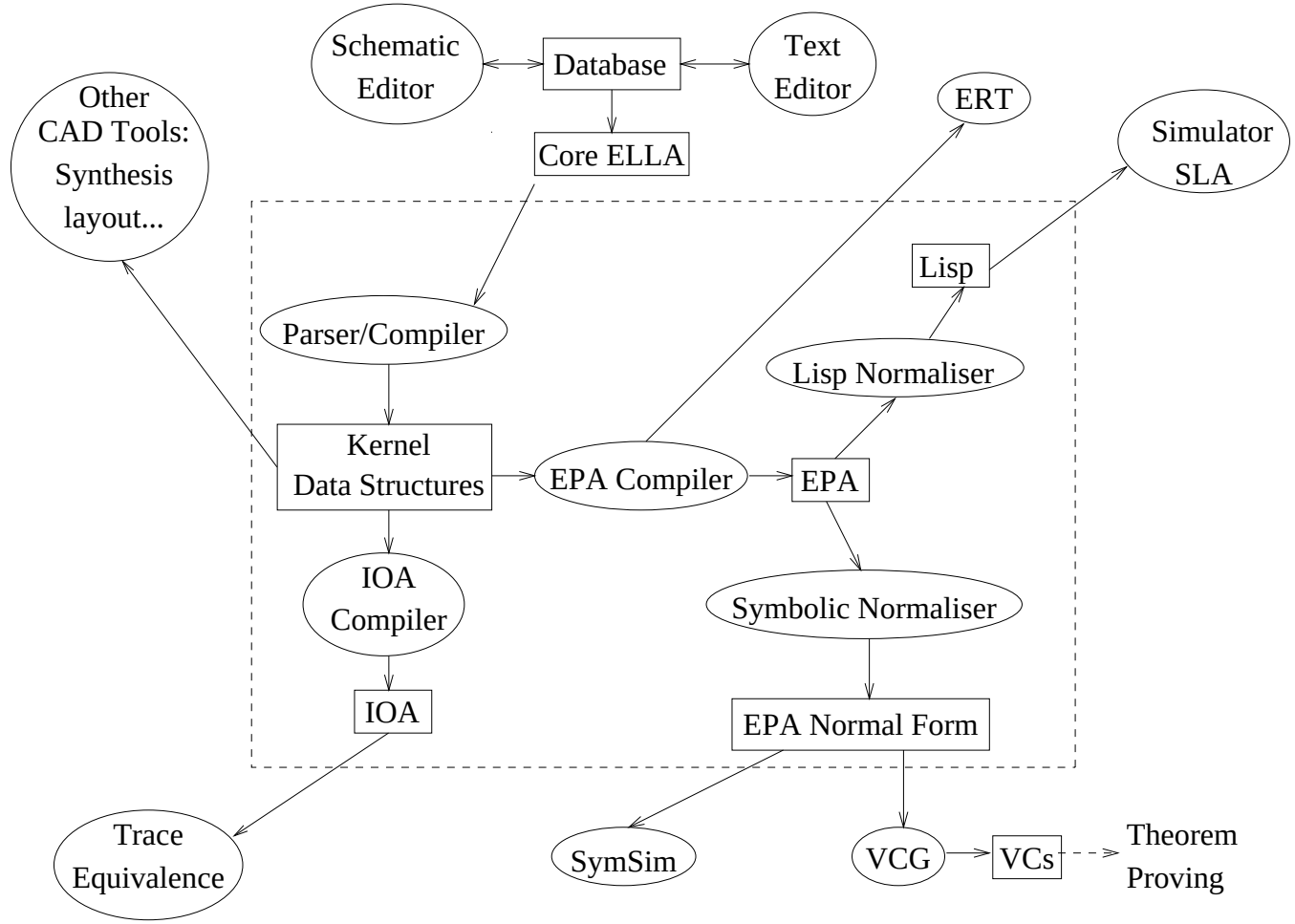
Implementation

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

Examples

- Compilation + Simulation:
  - 6502 Processor
  - 1600 lines of ELLA
  - approx 20 machine cycles/second
- Formal Verification:
  - 128-bit ALU
  - 5-stage FIR Filter: retiming

Summary


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

Integrated design environment for ELLA

Full semantics for commercial HDL

- Tools based on underlying formal semantic definition
- behaviour preserving transformation rules

Embedded verification tools

- Semi-automated approach

Designer interacts with environment via HDL

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