VHDL Symbolic simulator in OCaml

OCaml Meeting 2009
Grenoble – France
04/02/2009

Florent Ouchet florent.ouchet@imag.fr
TIMA Labs – GINP – UJF – CNRS – VDS group
Outline

VSYML – Vhdl Symbolic Simulator in ocaML:

• Symbolic simulation goals
• Why OCaml?
• Application structure and limitations of OCaml in this context
• Perspectives
• Conclusion
Symbolic simulation goals

Model is needed for:

- Model checking: equivalence of IC implementation and specification;

- Other formal methods based on the model: (ANR project FME$^3$) analysis of error consequences using formal methods.
Symbolic simulation goals

Integrated circuit synthesis flow:

- **Modeling level**
  - FSM, complex types
  - Boolean type
  - Drawing of transistors and wires

- **Register Transfer Level**
  - Explicit time, behavioral

- **Gate Level**
  - Manual refinements
  - Automatic synthesis
  - Placing and routing

- **Layout**
  - Manufacturing

- **Chips**
Symbolic simulation goals

Existing symbolic simulators:

- Explicit time, behavioral
- FSM, complex types
- Boolean type
- Drawing of transistors and wires

Modeling level
Register Transfer Level
Gate Level
Layout
Chips

VOSS/Forte (Intel CAD Labs)
Symbolic simulation goals

Existing symbolic simulators:

- Explicit time, behavioral
- FSM, complex types
- Boolean type
- Drawing of transistors and wires

Modeling level
Register Transfert Level
Gate Level
Layout
Chips

VSYML
Symbolic simulation goals

Description written in VHDL (IEEE Std 1076.1™ – 2007)
Symbolic values depend on the time.

Florent Ouchet – OCaml meeting 2009 – Grenoble – France
Why OCaml?

Expression patterns:

- $X@1ns$
- 1
- unary_operator $X$ (not – abs)
- $X$ assoc_operator $Y$ (+ * and or xor xnor)
- $X$ binary_operator $Y$ (nand nor $\wedge = /= < > <= >=$)
- $X$ other_operator $Y$ (mod rem / -)
- … (total of 28 patterns)
Why OCaml?

type formula =
  | FormulaSymbol of symbol
  | FormulaImmediateInt of big_int
  | FormulaUnaryOperator of (unary_operator*formula)
  | FormulaAssociativeOperator of (assoc_operator*formula list)
  | FormulaBinaryOperator of (binary_operator*formula*formula)
  | FormulaOtherOperator of (other_operator*formula*formula list)
  | …
Why OCaml?

• Known expression patterns are rewritten (numeric evaluation);

• As in the standard, the simulation algorithm is intrinsically free of side effects.
Why OCaml?

The next value of each signal is computed from current values.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

The processes are executed until the design is stabilized.
Why OCaml?

• Strong type checking (compared to C, Pascal…): confidence in computations;

• Lexer/parser generator;

• « Free » software, its ancestor was written in Mathematica;

• Lightweight and easy to redist (Java, C#, F#).
Application structure

VHDL source files -> VSYML -> Design model

parameters

Standalone and automated application.
Application structure

• Some facts:
  – Over 26500 lines of code
  – 24 dedicated modules (types, basics, lexer, parser, resources, evaluators, converters, volume extrusion, top-level…)
Application structure

- Straightforward conversion from BNF rules (language spec)
- No syntax for optional tokens
- Debugging conflicts is a nightmare

VHDL source files (thousands lines)

Lexer
(59 states, 1058 transitions)

Parser
(695 grammar rules, 1318 states)

Abstract syntax tree
Application structure

- Some basic functions are missing in OCaml standard libraries
- Issue 4662 closed as « won’t fix »
Application structure

• A function to map a list of functions to the same parameter:
  \((\forall a \to b) \text{ list } \to a \to b \text{ list}\)

• A function to remove the n first list elements:
  \(a \text{ list } \to \text{ int } \to a \text{ list}\)
Application structure

- Unbounded integers in VHDL (32 bits at least - "Time" requires 64 bits)
- Simulation of design on n bits
- → Big_int
Application structure

• Big_int is an abstract type

• (=) (<) (>) on Big_int raise runtime exceptions

• Associative list functions based on (=)
Application structure

- Many executions of the functional structure with different evaluation contexts
Application structure

• Signature of process simulation function:
  process -> structure_context ->
  evaluation_context -> changed_object list
Application structure

• Prototype of process simulation function:
  process -> structure_context ->
  evaluation_context -> changed_object list

• Functors for efficiency (+50%)
Application structure

- Prototype of process simulation function:
  process -> structure_context -> evaluation_context -> changed_signal list

- evaluation_context = {
  simulation_time: Big_int;
  currentvalues: (signal*formula) list; … }”

- Test with array, hashtbl and references
Application structure

- Some timings: FIR filter simulation, 1000 clock cycles.

<table>
<thead>
<tr>
<th></th>
<th>Parsing + elab.</th>
<th>Simulation</th>
<th>Output</th>
<th>Process memory</th>
<th>GC bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>assoc</td>
<td>40 ms</td>
<td>9.60 s</td>
<td>1.20 s</td>
<td>2.4 Mo</td>
<td>1189 Mo</td>
</tr>
<tr>
<td>array</td>
<td>40 ms</td>
<td>1.82 s</td>
<td>1.20 s</td>
<td>2.4 Mo</td>
<td>910 Mo</td>
</tr>
<tr>
<td>hashtbl</td>
<td>40 ms</td>
<td>2.00 s</td>
<td>1.17 s</td>
<td>2.3 Mo</td>
<td>921 Mo</td>
</tr>
<tr>
<td>ref</td>
<td>40 ms</td>
<td>1.76 s</td>
<td>1.25 s</td>
<td>2.3 Mo</td>
<td>899 Mo</td>
</tr>
</tbody>
</table>
Load averaging for a multi-thread simulation. Communication efficiency for a distributed simulation.
Perspectives

• Spread the simulation over cores and computers: an integrated circuit is a fixed finite set of concurrent process

• concurrent simulation but:
  – lots of rendez-vous and communications
  – Irregular load

• Mathematics: loop invariants…
Conclusion

• Development for project FME$^3$ is now finished.

• Research prototype for parallel computations?

• Release to the public? CeCill Licence?
Questions?

Thanks for your attention