VHDL Symbolic simulator in OCaml



OCaml Meeting 2009 Grenoble – France 04/02/2009

Florent Ouchet <u>florent.ouchet@imag.fr</u>
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

Model is needed for:

- Model checking: equivalence of IC implementation and specification;
- Other formal methods based on the model: (ANR project FME³) analysis of error consequences using formal methods.

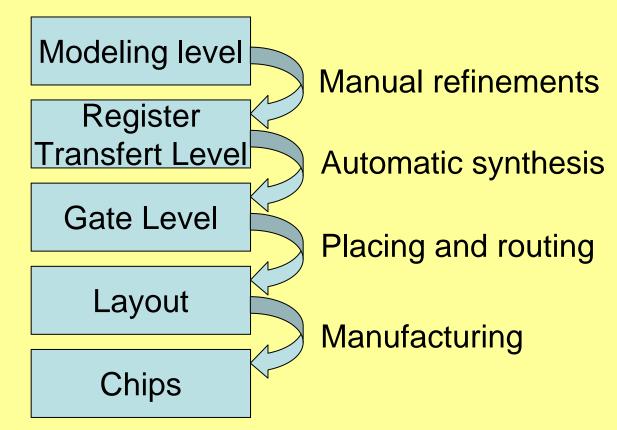
Integrated circuit synthesis flow:

Explicit time, behavioral

FSM, complex types

Boolean type

Drawing of transistors and wires

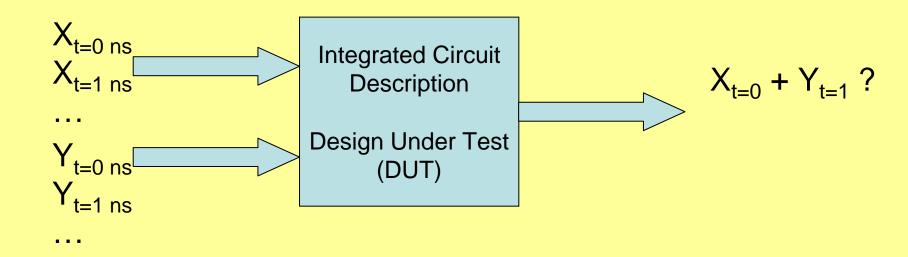


Existing symbolic simulators:

Modeling level Explicit time, behavioral Register Transfert FSM, complex types Level VOSS/Forte Gate Level Boolean type (Intel CAD Labs) Drawing of transistors Layout and wires Chips

Existing symbolic simulators:

Modeling level Explicit time, behavioral Register Transfert **VSYML** FSM, complex types Level Gate Level Boolean type Drawing of transistors Layout and wires Chips



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

Expression patterns:

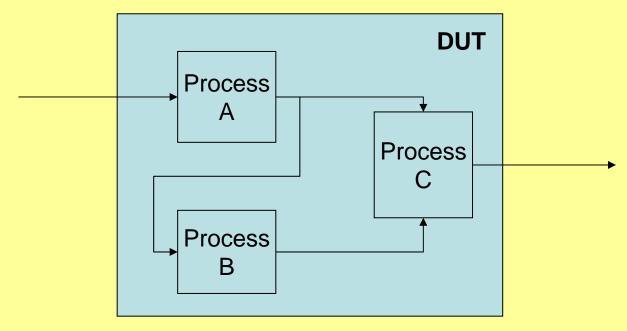
- -X@1ns
- **1**
- unary_operator X (not abs)
- X assoc_operator Y (+ * and or xor xnor)
- $X \text{ binary_operator } Y$ (nand nor $^ = /= < > <= >=$)
- X other_operator Y (mod rem / -)
- ... (total of 28 patterns)

```
type formula =

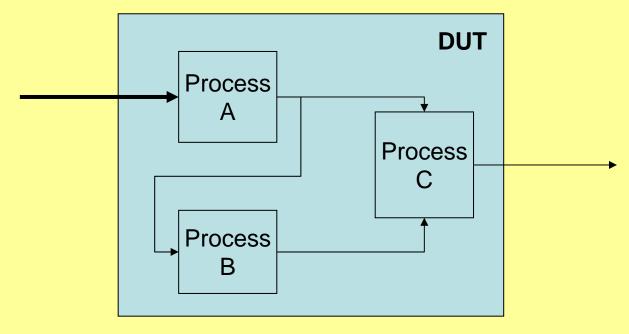
| FormulaSymbol of symbol
| FormulaImmediateInt of big_int
| FormulaUnaryOperator of (unary_operator*formula)
| FormulaAssociativeOperator of (assoc_operator*formula list)
| FormulaBinaryOperator of (binary_operator*formula*formula)
| FormulaOtherOperator of (other_operator*formula*formula list)
| ...
```

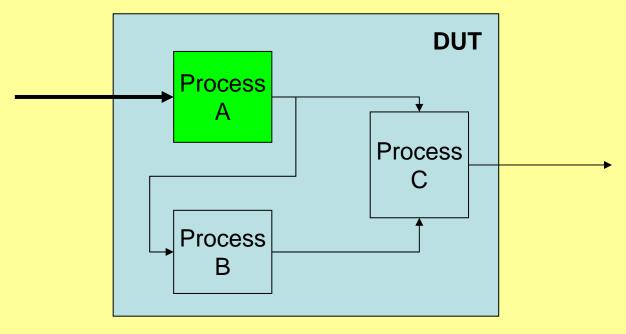
 Known expression patterns are rewritten (numeric evaluation);

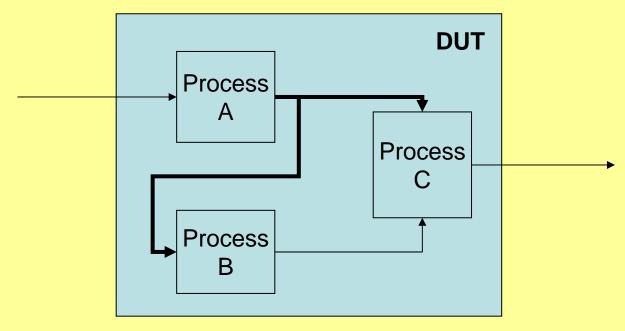
 As in the standard, the simulation algorithm is intrinsically free of side effects.

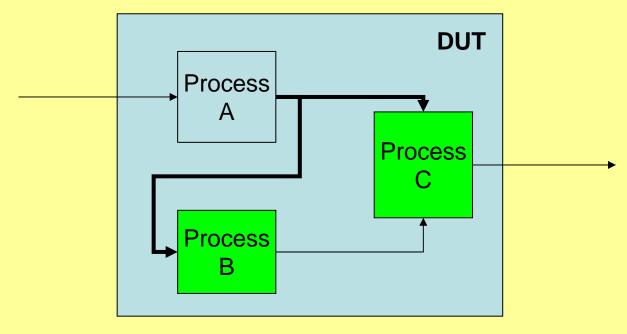


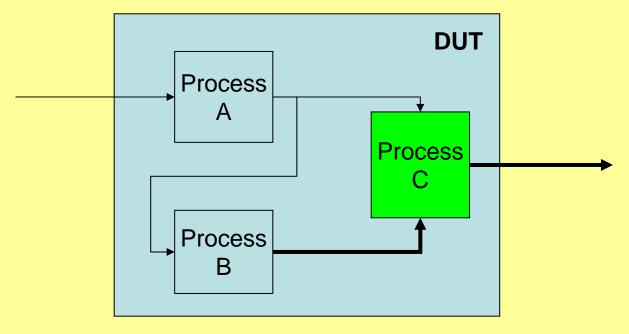
The next value of each signal is computed from current values.

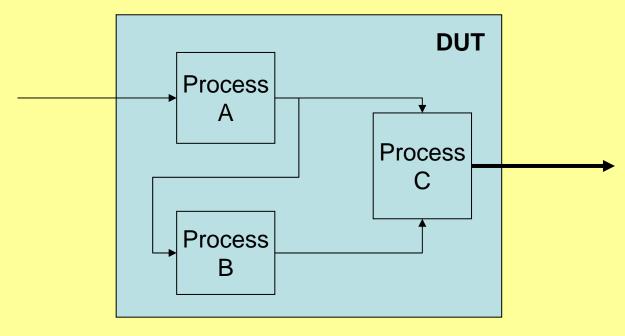








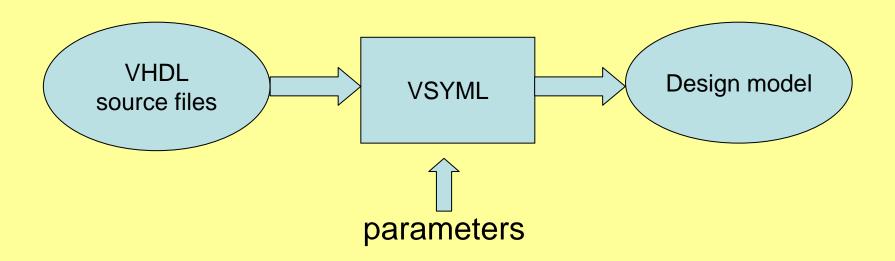




• Strong type checking (compared to C, Pascal...): confidence in computations;

- Lexer/parser generator;
- « Free » software, its ancestor was written in Mathematica;

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



Standalone and automated application.

Some facts:

- Over 26500 lines of code
- 24 dedicated modules (types, basics, lexer, parser, resources, evaluators, converters, volume extrusion, top-level...)

VHDL source files (thousands lines) Lexer (59 states, 1058 transitions) Parser (695 grammar rules, 1318 states) **Abstract**

syntax tree

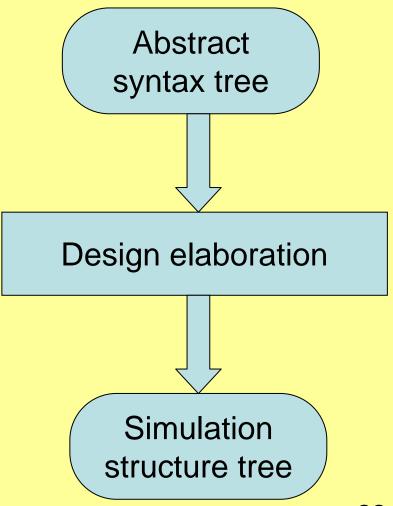
 Straightforward conversion from BNF rules (language spec)

No syntax for optional tokens

 Debugging conflicts is a nightmare

 Some basic functions are missing in OCaml standard libraries

 Issue 4662 closed as « won't fix »



 A function to map a list of functions to the same parameter:

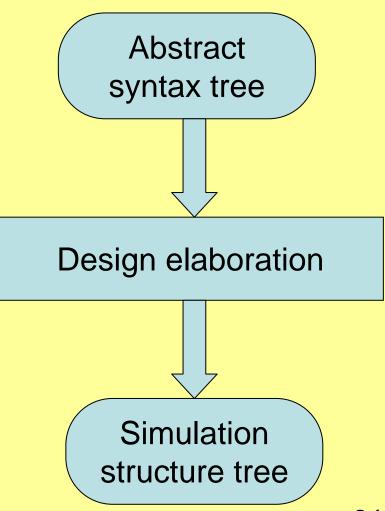
 A function to remove the n first list elements:

'a list -> int -> 'a list

 Unbounded integers in VHDL (32 bits at least -"Time" requires 64 bits)

 Simulation of design on n bits

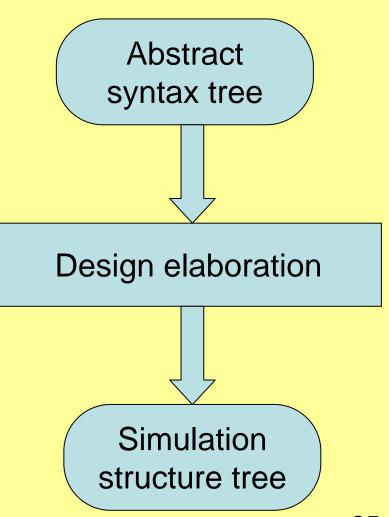
• → Big_int

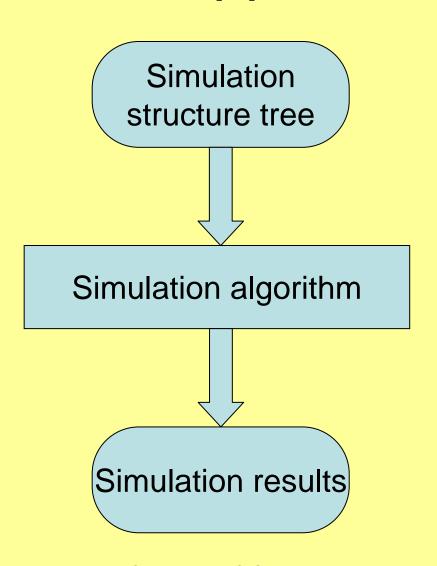


Big_int is an abstract type

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

 Associative list functions based on (=)





 Many executions of the functional structure with different evaluation contexts

Signature of process simulation function:

```
process -> structure_context ->
  evaluation_context -> changed_object list
```

Prototype of process simulation function:

```
process -> structure_context ->
  evaluation_context -> changed_object list
```

Functors for efficiency (+50%)

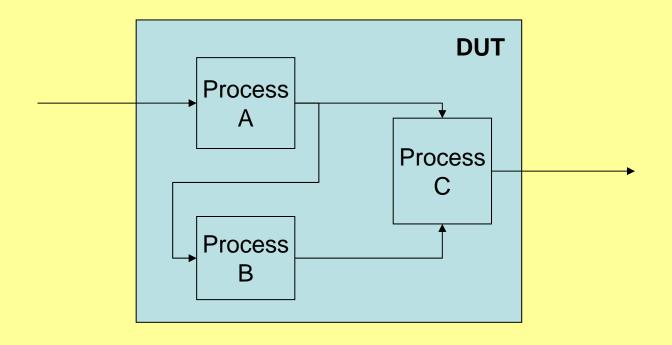
Prototype of process simulation function:
 process -> structure_context ->
 evaluation_context -> changed_signal list

```
    evaluation_context = {
        simulation_time: Big_int;
        currentvalues: (signal*formula) list; ... }
```

Test with array, hashtbl and references

Some timings: FIR filter simulation, 1000 clock cycles.

	Parsing + elab.	Simulation	Output	Process memory	GC bytes
assoc	40 ms	9.60 s	1.20 s	2.4 Mo	1189 Mo
array	40 ms	1.82 s	1.20 s	2.4 Mo	910 Mo
hashtbl	40 ms	2.00 s	1.17 s	2.3 Mo	921 Mo
ref	40 ms	1.76 s	1.25 s	2.3 Mo	899 Mo



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 FME³ is now finished.

Research prototype for parallel computations?

Release to the public? CeCill Licence?

Questions?

Thanks for your attention