Madeo: A CAD tool for reconfigurable architecture

Product Name	Madeo
Owner	Lab-STICC, UBO
Developers	Loic Lagadec and Pottier, Gouyen, Goubier, Llopis, Villelas, Lépicier, Dezan, Fabiani Several students (UBO)
Smalltalk Dialect	Cincom VisualWorks
URL	http://as.univ-brest.fr/~llagadec/ESUG/
Keywords	CAD, FPGA, High level synthesis
Licence	Research product.

Forewords

Reconfigurable architectures are electronic circuits whose behavior can be altered on demand by changing their configuration (bitstream, locates in an embedded memory).
These circuits offer the promise of software flexibility (writing to memory is enough to change/refactor the applicative circuit) to circuit designer while preserving high performances compared to purely software execution. This has favored a kind of agile circuit development with short cycle, allowing iteration rate to the final circuit delivery far ahead of those of competing solutions.

Another feature that made the FPGAs success possible is the high reuse of the target through multiple applications, that allowed to recoup (very high) non recurring costs and made unattractive designing dedicated circuits (ASICs) in many cases.

At this time, FPGAs vendors offer huge circuit, embedding several microprocessor cores. The main question is now «how to program these ? . On the other side, smaller companies sell reconfigurable cores to be located within an heterogeneous System On Chip.
Again the problem appears to deliver well suited tools to allow application synthesis. The wide spread solutions come from outdated former ones , VLSI oriented (strongly typed code such as VHDL to private bitstream format, through vendor specific tools), hence bring a productivity gap.

Overview

Madeo is a framework that addresses application synthesis over reconfigurable architectures (FPGAs).
Madeo is composed of Madeo-Bet (target definition) and Madeo-Fet (application specification).
Madeo-Bet allows to model the FPGA target, relying on element composition, topology information and computing element specification. This model supports a set of tools to floorplan, place and route application, appearing as register transfer level (RTL) textual formats, and to output resource allocation (required for bitstream generation, i.e. configuration production).

Madeo supports architectural prospection and very fast FPGA prototyping. Several FPGA, including some commercial ones (e.g Xilinx Virtex family) and prospective ones (STMicro LPPGA) have been modeled using Madeo. Also, Madeo-Bet has been used to address some reconfigurable datapath, and an emerging nano-technology from Umass (NASIC architecture).

Madeo-Fet is used to produce RTL description based on pieces of Smalltalk code on one hand, and external context (i.e. values set for each variable) on the other hand. Madeo relies on enumeration to build up truth tables that are converted to a binary format (PLA) that is post-processed using third party tools (SIS, from UC. Berkeley). The result is a hierarchical graph of optimized logic modules, that Madeo-Bet takes as input.
Madeo-Lite is a alternative version of Madeo-Fet that promotes the use of execution traces to collect values set, rather than relying on the designer to provide the context. This has been used to generate very efficient turbo decoders circuits.

A third level, out of the scope of this presentation is Madeo+ that is dedicated to architectural synthesis (i.e implementing control structures such as loops, multi process handshakes, ...). Madeo+ addresses building up a concurrent threads-to-FPGA path. Madeo-Bet is used to generate missing operators (operators that are not in library) and custom primitives (e.g. loop termination). Madeo+ has been partly developed in the scope of the Morpheus EU Project.

Application Modeling

This work was initiated in the mid 90s. The goal of the first version of Madeo-Fet (revisiting smalltalk-80 blocks) was to generate some small circuits based on a simple Smalltalk block to ensure efficient development. The block parameters were inputs of the circuit, while some variables, external to the block, acted as states, hence allowed to design sequential circuit as well. The circuit performances could be raised up by automatically inserting flip-flops, hence increazing the pipeline.

As a block based structure was not enough to build composite circuit, the Madeo-Fet project started in 2000. Madeo-Fet inherits from the former work its ability to generate/simplify RTL description, and extends it by supporting Smalltalk code parsing. The AST is used to generate a composite of messages sends. The application domain is represented by a class. For every method of the class, the AST can push forward the composite pattern or considers it as an atomic action; this depends on either tree depth, or can consider the method's category for decision. Also manual labeling remains possible so that the designer keeps a total control over the process. For method external to the class, the call appears always as an atomic action.

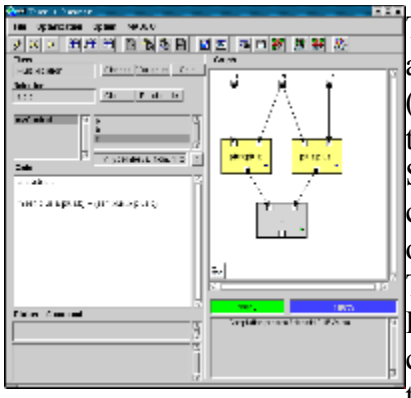

Figure 1: Graphical Tool: Code/Types/AST

The first step is to build up the AST (see figure 1), then by applying the context to the inputs and performing type propagation (on demand, depth first,...) the AST is converted to a graph of truth-tables.
Some classical compiler optimizations take place at this level (dead code removal, code fusion, ...) as well as some revisited one (e.g. op removal).
The context can refer to some non special types such as Galois-Field, with no impact on the application specification. In comparison, migrating from a float to GF typing may require up to a full redesign of the circuit with standard (traditional VLSI) approach.

The truth tables are converted to binary representation with a set of translators (2 bits, index based, type based), then logic synthesis takes place to output a minimized RTL description.

The code example is a floating point multiplier, operating on a couple of three values (sign, significand and exponent).

Typing relies on a set of classes for radix coding, set/interval, Galls Fields, scientific notation, etc.

```
sign: signA significand: significandA exponent: exponentA sign: signB significant:
significandB exponent: exponentB
        | sign exp significand normalize |
        sign := self computeSignFor: signA and: signB.
        significand := self computeSignificandFor: significandA and: significandB.
        exp := self computeExponentFor: exponentA and: exponentB.
        normalize := self normalizeSignificand: significand.
        ^Array
                with: sign
                with: (normalize at: 1)
                with: exp + (normalize at: 2)
```

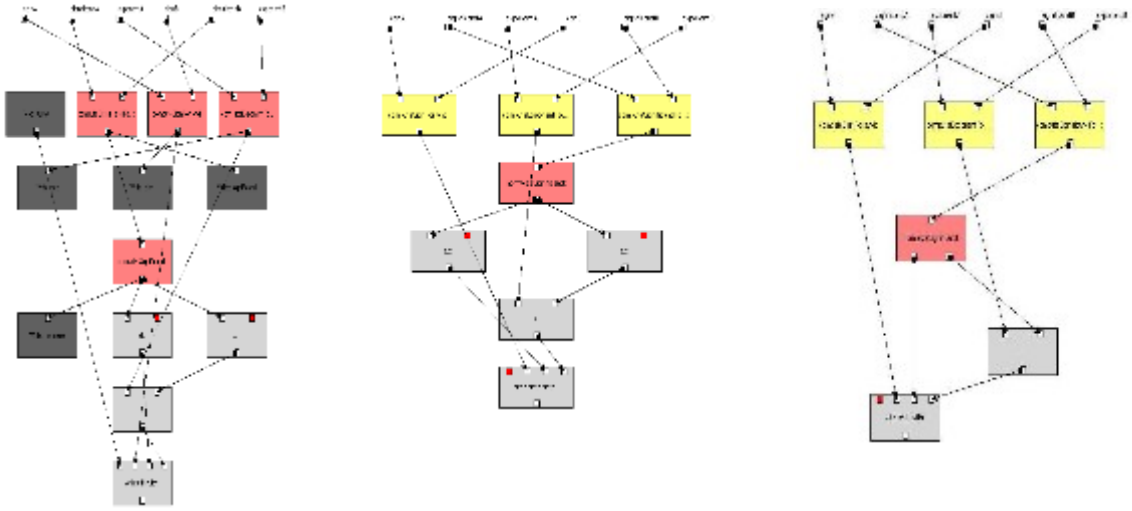

Figure 2: a Floating point multiplier at several optimization steps. red boxes are leaves within the domain(eg. #normalizeSignificand;), yellow ones are composite nodes and light grey boxes are external leaves (e.g. #+).

Target Modeling

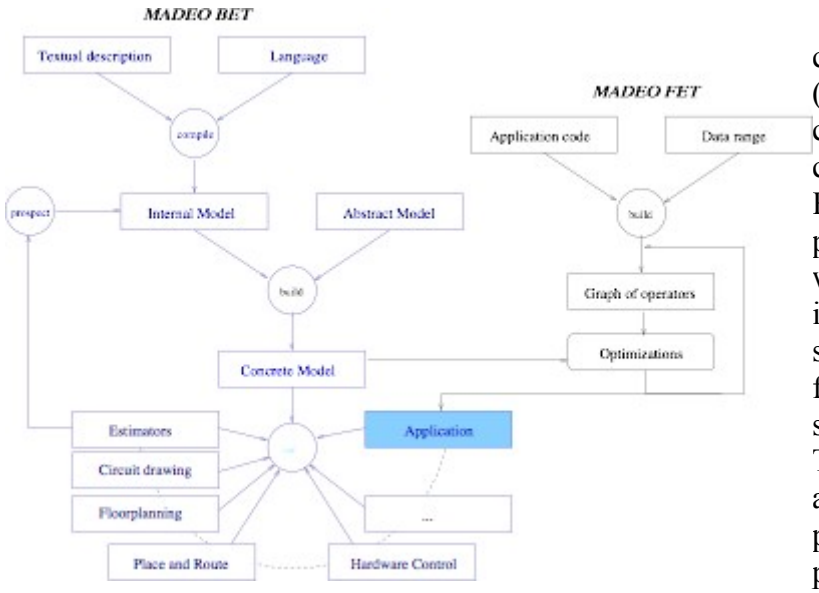

Figure 3: Global flow.

Madeo-Bet describes a set of classes for elementary elements (wires, transistors, ...) and composition mechanisms (matrix, composite).

Each target is described as a proprietary HDL code (figure 4), whose compilation is used to instantiate the model, and that supports parameterized variation to favor domain exploration (see left side of figure 3).

The instance of the target model is accessed through a set of tools to perform editing, floorplaning, place/route, ... that are required to program the target.

Also, Madeo-Bet supports generating bitstream and controlling third parties tools. As an example, figure 5 shows a circuit in Madeo-Bet and its counterpart in the Xilinx tools (fine resource allocation vs color based routing density information).
Figure 6 shows the layout of the floating point multiplier (the figure 2), on a LPPGA FPGA.
As the application remains executable , generating characterization tests is straight forward.
Also, assuming the internal state of the FPGA is accessible (e.g. XC6200 family from Xilinx), the tests can be extended to in-situ execution, taking advantage of Smalltalk polymorphism to hide hardware/software nature of the execution..

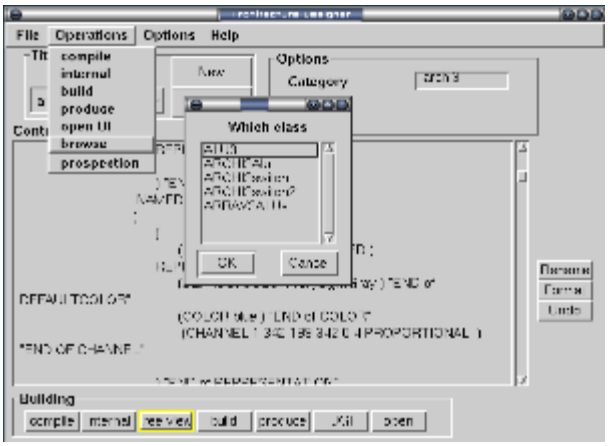

Figure 4: Specifying a target in Madeo-Bet

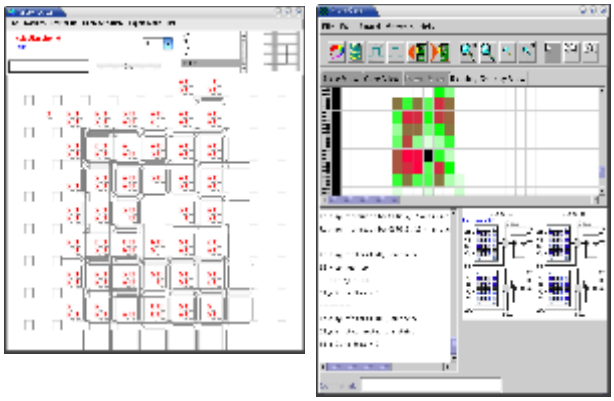

Figure 5: Madeo vs Vendor Tools

Madeo-Bet has also been extended to design circuit on an emerging nano-technology, with as special goal to implement nano-FPGAs. As an example, figure 7 shows a look-up table (elementary element of the FPGA, figure 3) implemented on top of such a technology.

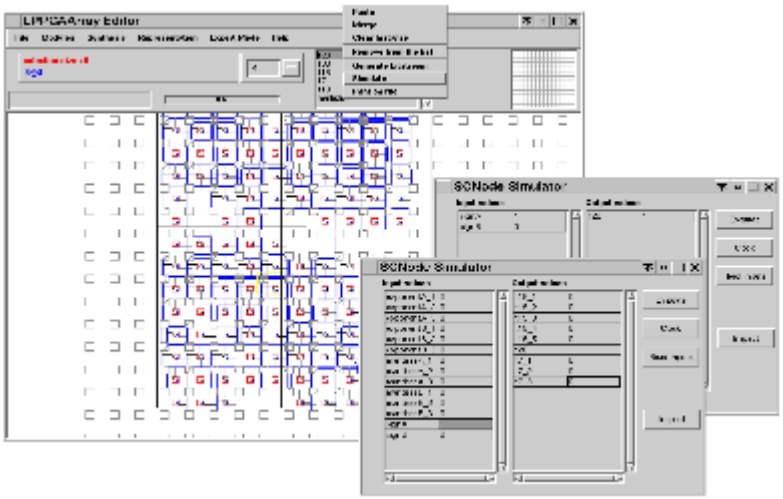

Figure 6: The UGI of Madeo-Bet. A synthesized floating-Point on the LPPGA FPGA, and its simulation results.

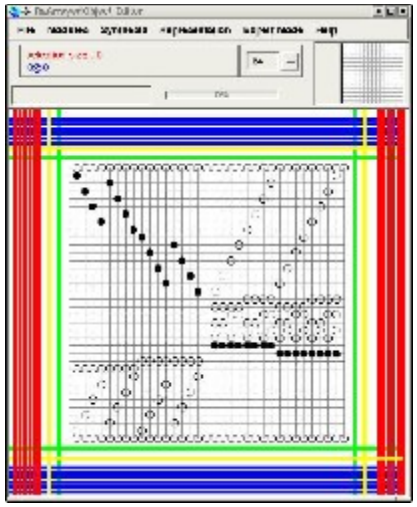

Figure 7: a 3 inputs LUT in NanoMadeo.

Concluding Remarks

Madeo is a Smalltalk framework, that offers an alternative to the use of expensive commercial synthesis/back-end Tools, while exhibiting OO well known characteristics: agile development facilities, high reuse, modularity, etc. Madeo has been being used in several projects (PRIR ValMadeo, RNTL OSGAR, FP6 Morpheus).