Complementing EDA with Meta-Modelling and Code Generation

Ecker Wolfgang
wolfgang.ecker@infineon.com

Michael Velten
michael.velten@infineon.com

Ajay Goyal
ajay.goyal@infineon.com

Leily Zafari
leily.zafari@infineon.com
Outline

- Motivation & Idea
- Concept – Abstraction Levels
- Meta-Modelling Infrastructure
- Flexibility of Infrastructure
- Features and Benefits of Infrastructure
- Competition with EDA and Applications
- Challenges of Technology
- Conclusion & Future Work
Motivation

EDA focus on generic tools.

EDA companies design platforms rather than specific tools.

EDA cost is high for investment in domain specific tools.

But, domain specific tools are needed to make designers effective.

Designer does not have expertise to build tools.
Idea

Provide automation wherever possible in development process

Framework to support homogeneous and well structured code

Provide framework to develop methodology around existing flow and data

Methodology to develop sophisticated domain specific tools

Meta-Modelling!!!
Concept – Abstraction Levels

**Meta-Meta Model**
- Defines Structure of Meta-Model
- Generates structured Meta-model

**Meta-Model**
- Defines Structure of Model
- Generates structured way to access model

**Model**
- Defines content of view language independently
- Generates view in a structured/systematic way

**View**
- Implementation of content
Model Driven Software Technology has been actively investigated by the industry for 10+ years. But none of the related work has been utilized by hardware/design industry.

- **The Eclipse Modelling Framework**
  - EMF is Java based, which implies some overhead in implementation due to Java’s strict object oriented approach.
  - The template engine is less intuitive and user friendly.

- **MetaCase/MetaEdit**
  - In this tool, a graphical domain specific language can be developed.
  - Wide applicability and hence reported use cases are from SW development, e.g. software for fish pond management.
Flexibility of Infrastructure

Same model used for multiple Input & Output formats

- Spec Format 1 - XLS
- Spec Format 2 - XML
- Spec Format 3 - HTML

Meta-Model

Generated Model

- VHDL
- SystemC
- Documentation
Features of Infrastructure

Reusability
- VHDL/Verilog, XML parsers, etc are integrated in infrastructure

Extendibility
- Model Extension easy due to automatic generation

Linkability
- Multiple models can be linked to make complex data structures.

Transferability
- Easier knowledge transfer as data structure has visual look.
Competition with EDA?

Absolutely Not!!!

- No point in re-inventing the technology wheel. Point is to extend and adapt application of EDA tools.

- In fact, the created automation enables designers to use EDA tools more productively.

- Target is to create domain specific tools for particular design formats and specification which are not generic in nature.
IO PAD Area Estimator

- Manually created run-set files
- Spice Simulator
- Spice Netlist
- Meta-Model represented through UML diagram
- User defined DRC Values
- Estimated IO PAD Area

Design Layout without Area Estimation
IO Lib Info Collector

Connectivity & Logical Equations etc. → Formatted Text Files → In-house IO Library Generator → Complete IO Library

Third Party EDA Tools → Schematic → Layout
Can EDA Further Help?

Definitely!!

- Creating more libraries/ parsers for HDL etc which can be integrated with the infra-structure and provided to designers.

- Open access to internal structures/databases → can help to replace today’s language interface (e.g. VHDL, UPF, …)

- XML based tool reports to ease parsing of results.

- Windows support, since automation mostly starts at concept/specification level, and that is Windows domain!
**Benefits for Designers**

- **Generation enables handling late specification changes easier**
  - **LATE SPECIFICATION CHANGES**
  - **CONSISTENCY INCREASE**
  - **PRODUCTIVITY IMPROVEMENT**
  - **EASIER TOOL INTEGRATION**
  - **QUALITY INCREASE**

- **Automation leads to better productivity**
  - **Copy/ Paste errors are eliminated due to automation**
  - **Tools can be designed to reduce entry barriers to EDA tools**
  - **Code, Documentation, etc. are generated from single source**
Challenges of Technology

**Learning Curve**
- Modelling infrastructure needs to be learned.

**Understanding of Domain**
- Good understanding of the design flow required to automate it.

**Management Support**
- It takes some time to learn the methodology and create own automation.

**Language Wars**
- Currently Python used for generation but continuous discussions on using Perl, Tcl etc.

**Excessive Interest**
- Automation eases design flow so much that designer becomes more interested in code generation rather than designing.
Conclusion and Future Work

**Tool Creation**
- Domain specific tools can be easily created.
- Extends EDA and make tool usage more productive.

**Effort**
- Meta-Modelling features assures flexibility, consistency and reusability of the design flow
- Needs initial effort from designer to learn and automate flows

**Future Work**
- Extend infrastructure to generate automatic GUI from the model
- Create utilities to read diagrams and generate code
- Create utilities to DIFF models easily
QUESTIONS?