IP-Coding Style Variants in a Multi-layer Generator Framework

Zhao Han, Keerthikumara Devarajegowda,
Andreas Neumeier, Wolfgang Ecker

accellera SYSTEMS INITIATIVE™

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Unconfigurable IP-Coding Style

- System on Chips (SoCs) need to be customized to perform the required tasks with minimal area and power consumption.
- Hardware generators are therefore introduced to accelerate the design process and aid Intellectual Property (IP) reuse.
- But the IP coding style, generated HDL files, remains unconfigurable to incorporate target design process and to achieve the desired optimization in a synthesizer.
- To address these challenges, we propose to transform platform-independent design models and platform-dependent view models in our novel multi-layer generation flow.
- Design models capture the design intent, where deceptive components are provided to indicate hierarchies and coding styles.
- By fine-tuning the descriptive components, IP coding styles are configured. Furthermore, file caching and hierarchy flattening transformations are provided to further assist flexibility in the back-end. Additionally, transformations to adapt naming convention is provided as well.

OBJECTIVES

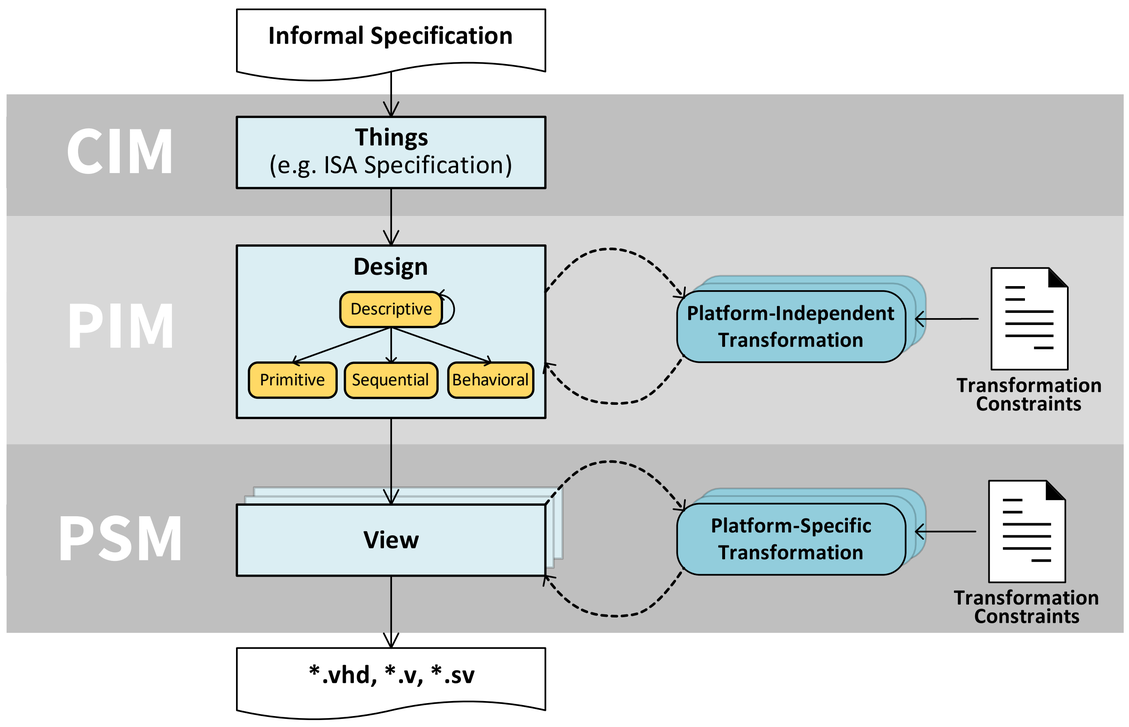
IP Coding-Style

- File Granularity
- Naming Convention
- Hierarchical Granularity
- Code Formatting

↓

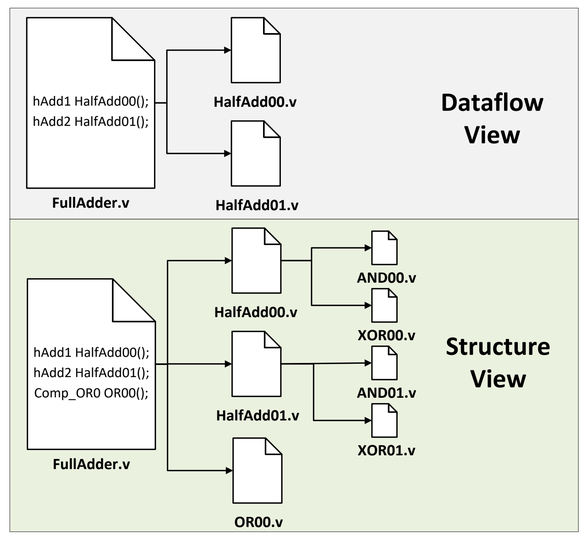
- Readability
- Maintainability
- Synthesis Time
- Code base versioning

Multi-layer Hardware Generation Framework



Generation of IP-Coding Style

Other than directly using descriptive components such as Structure, Dataflow, transformations are developed to further assist the coding style flexibility.



- **File-Cacher** reduces same-function HDL files.
- **Flattener** flattens hierarchies as instructed.
- **Dataflower/Structurer** consistent descriptive components in platform-independent design model.
- **Nomenclaturer** transforms the naming convention as user preferred.

Results

To demonstrate the applicability, the proposed approaches are applied on two industrial RISC-V SoCs.

Structural	Dataflower	File-Cacher	Flattener	Nomenclaturer
86	86	419	174	26

Lines of Code of Transformations

Design	Structural	Dataflower	File-Cacher	Flattener
SoC[1]	1.3k	0.1k	0.4k	0.06k
SoC[2]	12.3k	3.4k	3.8k	0.11k

Coding Style Variety

Results

Design	Structural	Dataflower	File-Cacher	Flattener
SoC[1]	8min	0.8min	2.7min	0.4min
SoC[2]	990.4min	45.5min	46.2min	1.4min

Rational ClearCase® Check-In Time

Design	Structural	Dataflower	File-Cacher	Flattener
SoC[1]	11.5min	5.0min	6.3min	4.9min
SoC[2]	97.1min	40.5min	43.0min	5.5min

Vivado® Synthesize Time

TECHNISCHE UNIVERSITÄT MÜNCHEN

This work has been partly developed in the SAFE4I project which is funded by the German ministry of education and research (BMBF) (reference number: 01IS17032A). Its roots go back to the project SANITAS starting October 2009, which has been partially funded by German ministry of education and research (BMBF) (reference number: 16M30888) as well.

infineon