January 2024

Generating Readable RTL Using a Language Model

Christopher D. Leary

Paul Rigge

Generating Readable RTL Using a Language Model

ABSTRACT

This disclosure describes the use of a language model to improve the readability of generated Register-Transfer Level (RTL) code. The language model is provided with a prompt that includes generated RTL and (optionally), the original hardware description. The model produces as output a readable version of the RTL. Prompts can be iterated upon automatically, e.g. if the code fails to pass a test (e.g., a lint test) or does not compile, or manually if the user prefers the output to be in a particular style. The readable RTL output by the language model can be subjected to a logic equivalence check (LEC) to confirm that the output readable RTL is equivalent to the input RTL to ensure that the behavior of the design is identical. Unlike a traditional code formatter, a language model can perform complex transformations and interpret the ideas in a block of code. A language model can also generate meaningful comments as part of the output.

KEYWORDS

- Code generation
- Code readability
- Code rewrite
- Register-Transfer Level (RTL)
- Design verification
- Physical Design
- Logic equivalence check
- Language model
- Code formatter

BACKGROUND

Programmatically generating code is common. Protocol buffers, parser generators, and generators for language bindings are some examples of programmatic code generation. Programmatically generated code is often challenging for humans to comprehend due to abstraction mismatch or difficulties in generating human-readable code. In many cases, humans do not need to read the generated code and therefore, programmatically generated code being hard to read is not a problem. For example, protobuf [1] is a well-tested and well-understood abstraction that users rarely need to read programmatically generated protobuf code.

However, in the context of generating Register-Transfer Level (RTL) (abstraction level commonly used to describe hardware), human-readable code is useful, but the generated RTL is often difficult to read. Similar to protocol buffer or parser generators, hardware generators take an abstract hardware description as input and produce RTL as output. The generated RTL often needs to be integrated with a substantial amount of manually written RTL. Crucially, RTL undergoes a higher degree of verification than software code since hardware cannot be updated once manufactured. Such verification includes meeting coverage metrics and debugging, both of which require some level of human understanding of the code.

Additionally, physical designers also rely on RTL to understand issues related to clock frequency, area required, etc. For both design verification and physical design (compiling RTL into a manufacturable chip), the readability of RTL is important. This poses a challenge for RTL generation tools that aim to boost developer productivity by raising the abstraction level of hardware design. Code that is not easily readable can undermine the gains in developer productivity by making design verification and physical design tasks difficult.

Code formatters attempt to make code more readable by reformatting code. However, these tools are limited in transformations and do not address most problems with generated RTL.

DESCRIPTION

This disclosure describes the use of a language model to improve the readability of generated Register-Transfer Level (RTL). The language model is provided with a prompt that includes generated RTL and (optionally), the original hardware description. The model produces as output a readable version of the RTL. Prompts can be iterated upon automatically, e.g. if the code fails to pass a test (e.g., a lint test) or does not compile, or manually if the user prefers the output to be in a particular style. The readable RTL output by the language model can be subjected to a logic equivalence check (LEC) to confirm that the output readable RTL is equivalent to the input RTL to ensure that the behavior of the design is identical. Unlike a traditional code formatter, a language model can perform complex transformations and interpret the ideas in a block of code. A language model can also generate meaningful comments as part of the output.

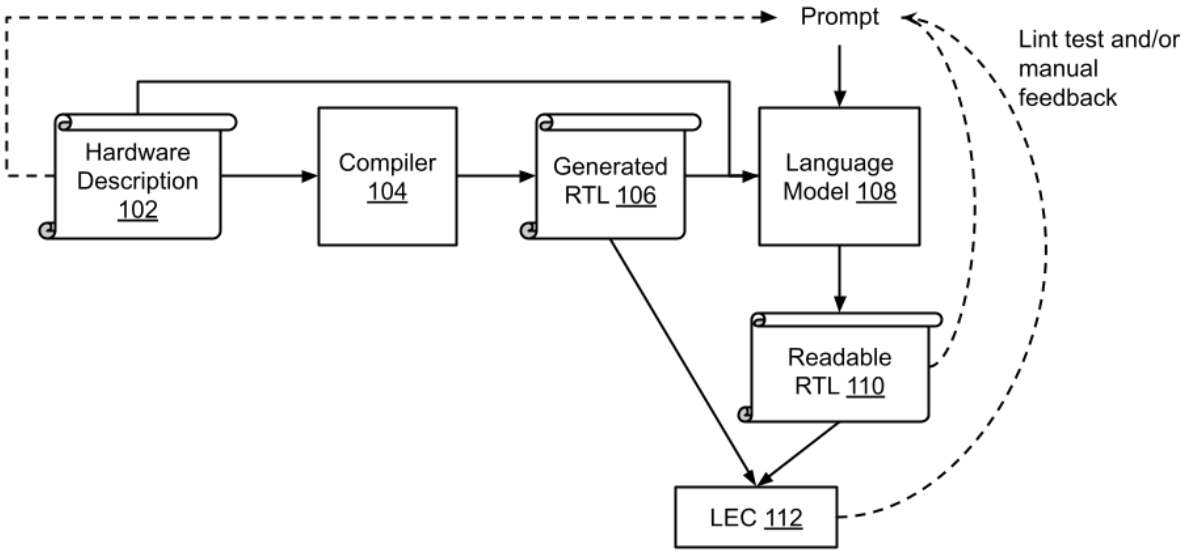

Fig. 1 : Obtaining readable RTL using a language model

Fig. 1 illustrates an example diagram illustrating obtaining readable RTL by using a language model to perform a code rewrite. Hardware description (102) is converted into generated RTL (106) using a compiler (104) using established techniques. The generated RTL and optionally the hardware description are provided as inputs to a language model (108) along with a prompt that instructs the language model to rewrite the RTL for readability. The original hardware description can also be utilized when generating the prompt to augment the context.

The language model (108) generates a readable version of the RTL code (110). The readable RTL output by the language model is subjected to testing (e.g., lint test) and/or manual review. Further, a logic equivalence check (112) can be performed between the generated RTL and the readable RTL. Based on the testing, user review, and LEC, additional prompts can be provided to the language model to generate new versions of readable code until the code passes tests and the user confirms readability.

CONCLUSION

This disclosure describes the use of a language model to improve the readability of generated Register-Transfer Level (RTL) code. The language model is provided with a prompt that includes generated RTL and (optionally), the original hardware description. The model produces as output a readable version of the RTL. Prompts can be iterated upon automatically, e.g. if the code fails to pass a test (e.g., a lint test) or does not compile, or manually if the user prefers the output to be in a particular style. The readable RTL output by the language model can be subjected to a logic equivalence check (LEC) to confirm that the output readable RTL is equivalent to the input RTL to ensure that the behavior of the design is identical. Unlike a traditional code formatter, a language model can perform complex transformations and interpret

the ideas in a block of code. A language model can also generate meaningful comments as part of the output.

REFERENCES

1. "Protocol Buffers" available online at https://protobuf.dev/ accessed Jan 2, 2024.
2. "ANTLR: (ANother Tool for Language Recognition)" available online at https://www.antlr.org/ accessed Jan 2, 2024.
3. "SWIG.org" available online at https://www.swig.org/ accessed Jan 2, 2024.