



Assertion-Gen

Multi-LLM Assertion Generation for Formal Verification Applications

Introduction

Why use LLMs in Formal Verification ?

Formal Verification is hard!

Formal Verification, though very useful is hard for verification engineers to implement due to its differences with widely used simulation based verification.

Complex Designs require advanced convergence techniques.

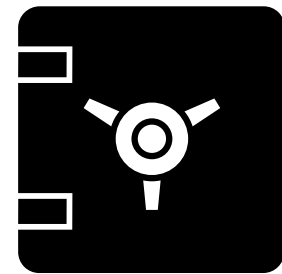
Complexity for formal verification increases exponentially with register/logic counts. Advanced techniques are needed to create formal friendly assertions.

Aggressive verification timelines for IP level verifications.

With yearly tapeouts, verification timelines are pretty aggressive. If possible, automation in formal verification testbenches will save crucial engineer hours.

Challenges

Privacy and other concerns with Large Language Models



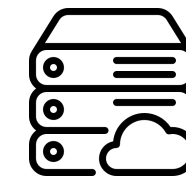
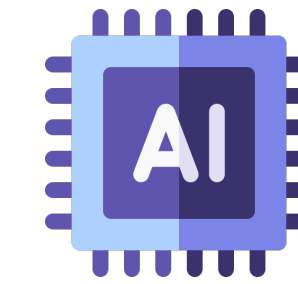
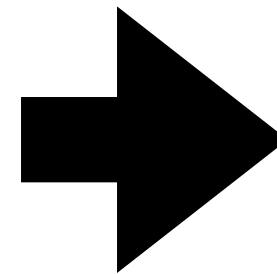
IP security and
Confidentiality is paramount!



Lack of Open Source
or available training data.



Formal methods require
high reasoning effort due
to mathematical nature.



Cloud-based
Frontier Models

Frontier Models
have good
accuracy.

But, are not secure
and private.



Local Open-Weight
Models

Local Open-Weights
Models are free
and open source.

But, are not accurate with
System Verilog Generation

Proposed Solution

Mix use of Local and Frontier Models

Stage 1

Specification to English
Language Assertion
Statements



Anthropic/
Claude-3-Opus

- Only specification.md is shared.
- No RTL specific data is shared to external servers.

Stage 2

English Language
Assertion Statements to
SystemVerilog Assertions



Anthropic/
Claude-3-Opus

- Only Stage 1 output is shared.
- No RTL specific data is shared to external servers.
- Syntactically correct but generic assertions are generated.

Stage 3

Refining raw assertions
with RTL knowledge
and create testbench with
aux-code.



Meta/
Llama-7b-rtlcoder-verilog

- RTL code + Stage 2 output is shared.
- DSPy is used to optimize prompts and use examples for few-shot learning
- Feedback loop with Verilator (Open source verilog simulator) to lint and check for syntax issues in refined testbench.

Stage 4

Generating run scripts in
TCL for JasperGold.



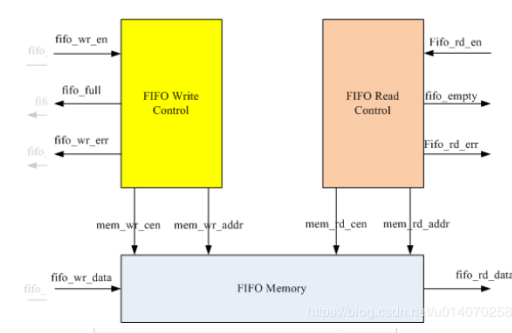
Meta/
Llama-7b-rtlcoder-verilog

- RTL Code + Stage 3 output is shared.
- DSPy is used to optimize prompts using TCL examples.
- Working JasperGold FPV tcl is generated for a complete solution.

Using  DSPy for prompt optimization
and few-shot learning

Experiments and Results

First-In-First-Out (FIFO) and Round Robin Arbiter



Source: programmingsought.com

FIFO

Generated Example:

```
assert property (@(posedge clk) (full
&& wr_en) |-> overflow);
```

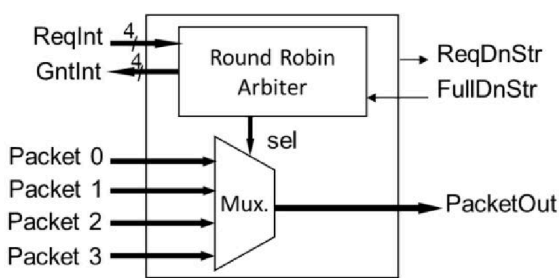
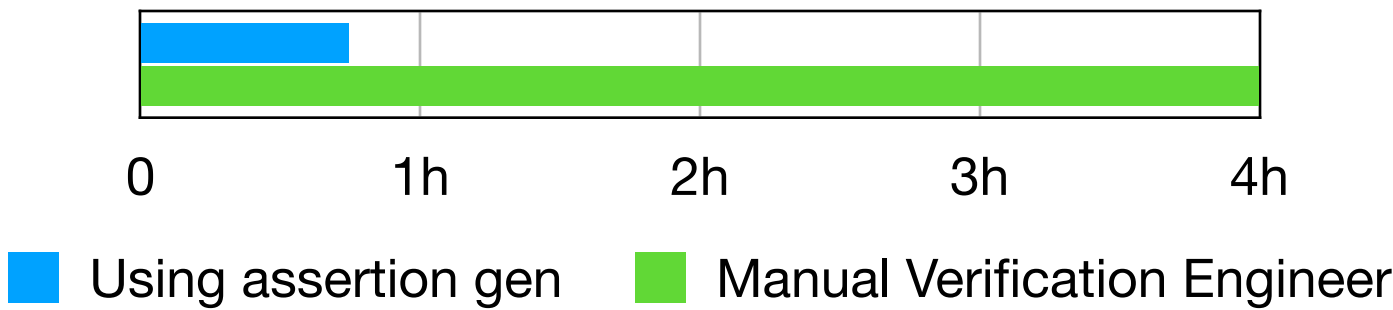
Generated Assertions:

11

Human Review:

7 high quality, 3 medium quality, 1 low quality

Testbench
Development



Source: semanticscholar.org

Round Robin Arbiter

Generated Example:

```
as_req_implies_gnt: assert property
(@ (posedge clk) disable iff (!rst_n) (!
req) |-> (!gnt));
```

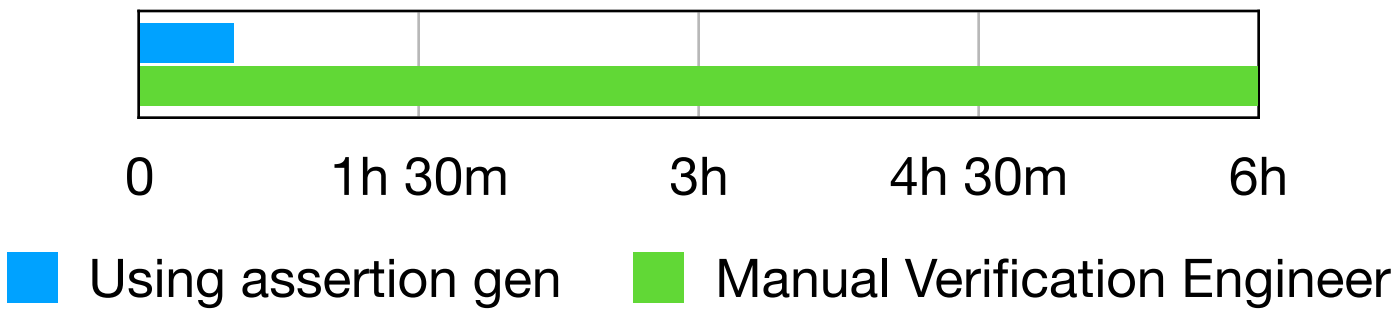
Generated Assertions:

6

Human Review:

3 high quality, 2 medium quality, 1 low quality

Testbench
Development



JasperGold Outputs

Using TAMU - Olympus Server

JasperGold GUI

jaspergold_fpv.tcl (session_0) - Jasper Apps (...ssertion_gen/jgproject) - Main

File Edit View Design Application Window Help

Formal Property V... 0/(0)

File Design Setup Task Setup Formal Verification Search

Design Hierarchy

Property Table

rr_arbiter tb (rr_arbiter tb)

DUT (rr_arbiter)

Type	Name	Engine	Bound	Traces	Time	Task
Assert	rr_arbiter_tb_assert_1	PRE	Infinite	0	0.0	<embedded>
Cover (related)	rr_arbiter_tb_assert_1:witness1	PRE	1	1	0.0	<embedded>
Cover (related)	rr_arbiter_tb_assert_1:precondition1	PRE	1	1	0.0	<embedded>
Assert	rr_arbiter_tb.genblk1[0].as_gnt_implies_req	Hp (1)	Infinite	0	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[0].as_gnt_implies_req...	PRE	1	1	0.0	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[0].as_gnt_implies_req...	PRE	1	1	0.0	<embedded>
Assert	rr_arbiter_tb.genblk1[1].as_gnt_implies_req	Hp (1)	Infinite	0	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[1].as_gnt_implies_req...	Hp	1	1	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[1].as_gnt_implies_req...	Hp	1	1	<0.1	<embedded>
Assert	rr_arbiter_tb.genblk1[2].as_gnt_implies_req	Hp (1)	Infinite	0	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[2].as_gnt_implies_req...	Hp	1	1	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[2].as_gnt_implies_req...	Hp	1	1	<0.1	<embedded>
Assert	rr_arbiter_tb.genblk1[3].as_gnt_implies_req	Hp (1)	Infinite	0	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[3].as_gnt_implies_req...	Hp	1	1	0.1	<embedded>
Cover (related)	rr_arbiter_tb.genblk1[3].as_gnt_implies_req...	Hp	1	1	0.1	<embedded>
Assert	rr_arbiter_tb.as_req_implies_gnt	Hp (1)	Infinite	0	<0.1	<embedded>
Cover (related)	rr_arbiter_tb.as_req_implies_gnt:witness1	PRE	1	1	0.0	<embedded>
Cover (related)	rr_arbiter_tb.as_req_implies_gnt:preconditio...	PRE	1	1	0.0	<embedded>
Assert	rr_arbiter_tb.genblk2[0].as_round_robin_fair...	PRE	1	1	0.0	<embedded>

Design Hierarchy Task Tree

Total: 36 Filtered: 36 Selected: 0

Validity: 29:7:0:0 Run: 0:0:0:36

session_0

[36] rr_arbiter_tb_assert_6:precondition1 covered Hp 1 0.003 s

ASSUMPTIONS

Name	Expression	Location	Status	Dependencies
------	------------	----------	--------	--------------

[<embedded>] %

Console Lint Messages Warnings / Errors Proof Messages

No proofs running Console input ready

Demo