2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 28 - MARCH 3, 2022

Automatic Translation of Natural Language to SystemVerilog Assertions

Author :

Abhishek Chauhan

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

accellera
SYSTEMS INITIATIVE

MACHINE LEARNING IN DESIGN VERIFICATION AND VALIDATION

"When All You Have Is a Hammer, Everything Looks Like a Nail"

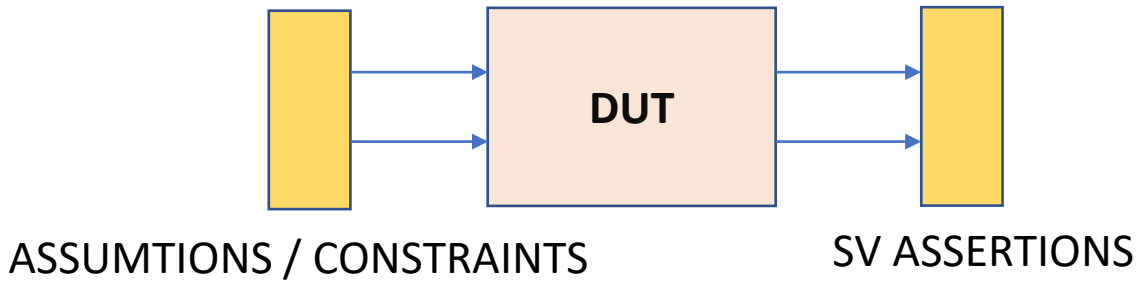
- Verification of designs or changes in them requires a lot of manual work
- There is need to automate this as much as possible
- ML can be used to great extent for automation in this regard

SystemVerilog Assertions (SVA) in Verification and Validation

- Primarily used to validate the behavior of a design against its original specification.
- Used to provide information on functional coverage for a design.
- May also be used as a formal specification language, making the requirements clear and unambiguous to automate validation of the design against its given specification.

SystemVerilog Assertion complexities

- Writing accurate yet compact SVAs is a difficult task to achieve.
- Often the actual written SVAs may not be equivalent to the intended assertion/check, leading to missed design bugs as well lower functional coverage.



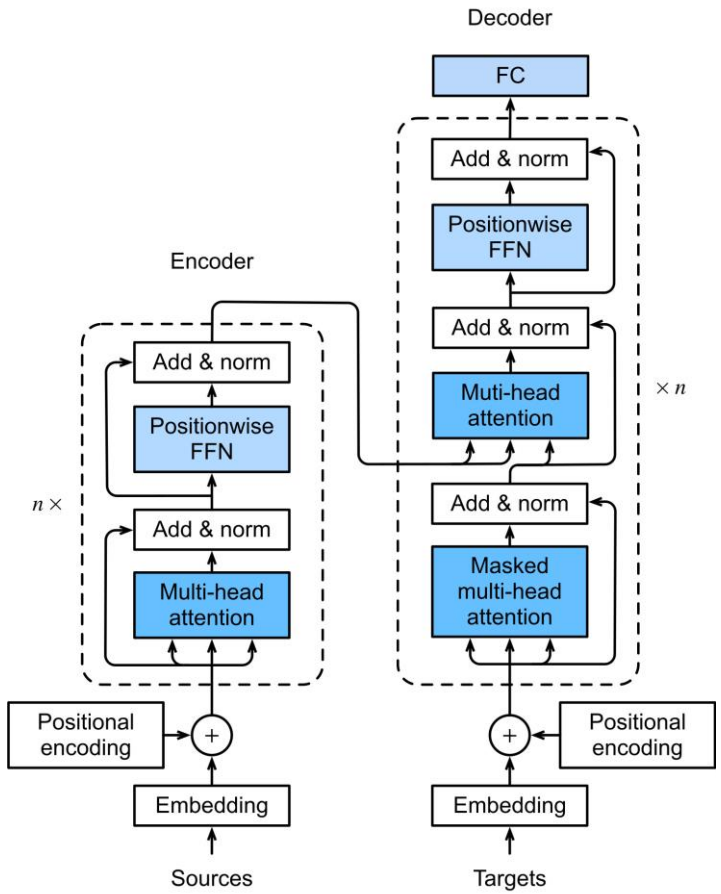
Objective

- Solving the problem of writing sophisticated SystemVerilog Assertions through machine learning.
- To bridge the gap for verification engineers who do not have prior knowledge of writing SVA to functionally validate a design.
- To create a pipeline or connectivity to understand the user's natural language description of assertions and generate SVA, and vice-versa.
- Help users to conceptualize their intended assertions into actual SVA.

Deep learning space

- Transformer neural network

  - State-of-art network in NLP
  - Can out-perform all NLP datasets such as earlier models like SQUAD, BABL etc.
  - Architecture with self attention-based encoder-decoder
  - Used in computer vision applications
  - Much more promising results as compared to previous neural nets
  - Used to perform both Machine Translation and Named Entity Recognition



Transformer neural architecture

❑ Tokenizer

- A method to convert sentences into multiple tokens or sub-words
- Lesser data may make it difficult to build a new tokenizer
- Pre-trained Bert Tokenizer used to deal with Out-Of-Vocabulary words
- Pre-trained word embeddings also used which is associated with the tokenizer
- Excludes multiple spaces and considers punctuation as separate token

Data Mining

- This task involved collecting domain specific cases
- As very limited availability of information related to System Verilog Assertions, Multiple sources considered for initial samples including industry standard samples to focus more on actual SVA practices
- Additional data was generated using rules and syntaxes of writing System Verilog Assertions
- Samples were generated using phrase-based dictionaries
- Data generator involve complex mapping of rules to adhere to the rules of target language through python scripts
- Steps were done to build an initial model, and it was generalized further using augmentation techniques along with fine-tuning using new available data

Data generalization

- More data was required apart from the collected samples
- Augmentation techniques like back-translation was used to generate more samples
  - This data needed to be monitored, in order to avoid distorted sentences
- Training of the model was also done using fine-tuning to remove data uncertainty.

Data inconsistency

- Large inconsistencies were observed during dataset building
- Large number of overlapped meanings occurred
- Model sometimes diverged if trained with that inconsistent data
- Debugging of such dataset cases were laborious and time consuming
- Also, parallel meaning didn’t exist between data samples unlike other cases such as German-English or vice-versa, etc.

Model Training

- Model was well trained :
  - With a batch size of 64 and a maximum sentence length of 512
  - For 50 epochs, where each consisted of 300 batches
- Model was observed to have the ability to learn quickly even the complex scenarios
- Training was data centric initially due to absence of generalized dataset and was expected to overfit
- Augmentation was done :
  - To increase model's generalizability
  - Decrease model's overfitting to small dataset

Training with Augmentation

- Along with improved performance on augmented samples, the data samples needed to be monitored to avoid model divergence
- Augmentation performed for both character and word level
- Mechanism used :
  - Back-Translation
  - Random word/character swap
  - Random word/character deletions
  - Random word/character insertions
  - Synonym's replacement (i.e., words with same meaning were exchanged)

English	French	English
signala is high and go low in the next cycle	*signala est élevé et diminue au cycle suivant*	*signala is high and decreases in the next cycle*
signala changes to 1	*le signala passe à 1*	*the signala goes to 1*

Hyperparameter selection

- Hyperparameter were used after trying out few combinations. Such as :
  - Number of layers = 4
  - Attention head = 8
  - Embedding size = 128

- The above-mentioned parameters found out to be best for the considered scenario

- We saw,
  - Increasing number of layers lead to increase of model training time as well as model convergence time
  - Increasing embedding size lead to fluctuation of the accuracy, and also model jumps from one point to another

Some data samples with predictions

Input	Predict
signala is stable, signaln is low after signalg is unknown, after 2 cycles, signalb is low	*$stable (signala) ## 0 $isunknown (signalg) |=> ! signaln ## 2 ! signalb*
ack is stable , req is low after pready is unknown, after 2 cycles, arready is low	*$stable (ack) ## 0 $isunknown (pready) |=> ! req ## 2 ! arready*
After a 5 valid signala, signala goes high until signall arrives	*$rose (signala) |-> ## 5 $rose (signala) throughout ($rose (signall) [- > 1])*
When signala deassert and after 2 clock ticks throughout which signale is low and it is for 10 cycles	*$fell (signala) ## 2 throughout (! signale [* 10])*
Whenever signala goes high, signalb must go high within 2 clocks and go low in the next clock	*$rose (signala) |-> ## [1 : 2] signalb ## 1 ! signalb*
The signala is not equal to signald for 10 or more non consecutive clock cycle	*(signala != signald) [- > 10 : $]*
signala rose to 1	*$rose(signala)*
signala is high and go low in the next cycle	*signala ## 1 ! signala*
The sequence in which signala is high and signald is low and the sequence in which signalc is stable and in the next cycle, signale is low.	*(signala && !signald) and ($stable(signalc) ##1 !signale)*
The sequence in which signala is high and signald is low and the sequence in which signalc is stable and in the next cycle, signale is low. They must end at same clock cycle	*(signala && !signald) intersect ($stable(signalc) ##1 !signale)*
When signala is high and in the same cycle signalc is less than the value of signald before 2 clock ticks and must occur within the sequence signala is high.	*(signala |-> signalc < $past (signald, 2)) within (signala)*

Results

- Resulted in a state of art for machine translation
- Achieved a greater accuracy
- Model was able to learn and translate complex descriptions of assertions
- Achieved an accuracy of around 99 percent on the training dataset
- Achieved an accuracy of around 83-85% on an internal test dataset

Applications

- Helping verification and validation engineers to easily capture accurate SVAs through their specification level descriptions
- Automatically hooking assertions from descriptions to the design for improved functional coverage
- Following application has been already deployed on https://ispec.ai

Future Scope

- Covering more complex scenarios
- Increased model reliability
- Improved generation of sophisticated SVAs efficiently
- Covering and understanding user’s natural language intent

References

- 1800-2017 - IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language
  - https://ieeexplore.ieee.org/document/8299595
- Attention Is All You Need
  - https://arxiv.org/abs/1706.03762
- Transformer: A Novel Neural Network Architecture for Language Understanding
  - https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
- Nlpaug python library
  - https://github.com/makcedward/nlpaug

References

- Incorporating Copying Mechanism in Sequence-to-Sequence Learning
  - https://arxiv.org/abs/1603.06393
- A Survey of Data Augmentation Approaches for NLP
  - https://arxiv.org/abs/2105.03075
- Automating the Translation of Assertions Using Natural Language Processing Techniques
- Generating Formal Verification Properties from Natural Language Hardware Specifications
- https://d2l.ai/chapter_attention-mechanisms/transformer.html

Questions