On the Naturalness of Hardware Descriptions

Jaeseong Lee, **Pengyu Nie**, Junyi Jessy Li, Milos Gligoric

TEXAS
The University of Texas at Austin

ESEC/FSE 2020

Partially supported by: Google

Motivation: Success of Mining Software Repositories

- **Corpora**: software artifacts available in open-source repositories
  - Code, natural language documentations, pull requests, open issues...
- **Observation**: code in programming languages is natural (repetitive and predictable) just like text in natural language
  - ICSE'12 On the Naturalness of Software: Java, C
  - ICSE'19 Natural Software Revisited: Java, C, C#, JavaScript, Python, Ruby, Scala
- **Applications**: statistical and learning-based models for code completion, code repair, code search, code summarization...
- **Limitation**: existing work and applications have focused almost exclusively on general-purpose languages
- **Unexplored**: hardware description languages

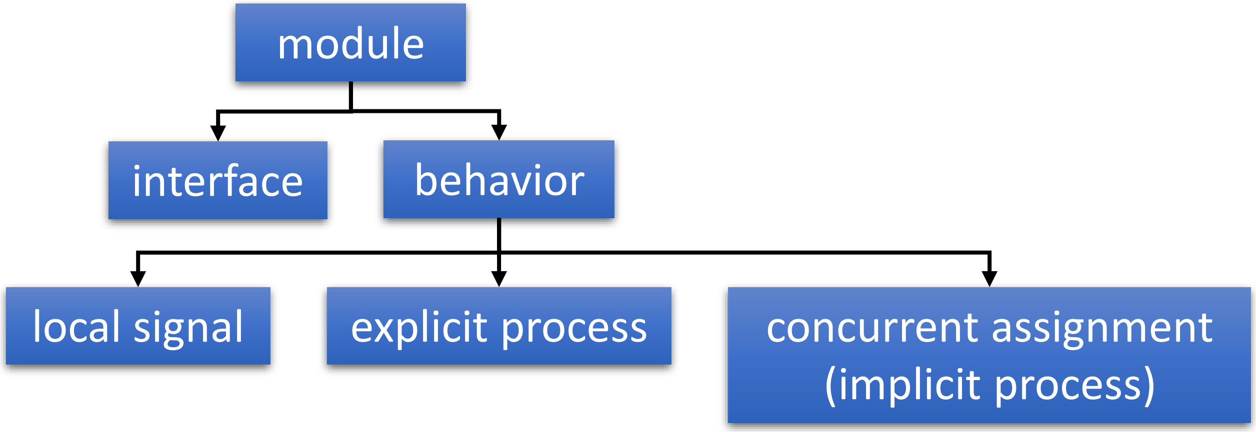
Hardware Description Languages (HDLs)

- **Usage**: describing logic circuits
- **Examples**: VHDL, Verilog, SystemVerilog
- **Key difference**: processes are executed in parallel

```
entity fpga64_sid_iec is
  port(...
    clk32 : in std_logic;
    uart_txd : out std_logic; ...
    uart_dcd_out: out std_logic; );
end fpga64_sid_iec;
architecture rtl of fpga64_sid_iec is ...
  signal cia2_pao: unsigned(7 downto 0);
  signal cia2_pbo: unsigned(7 downto 0);
  signal vicAddr: unsigned(15 downto 0); ...
begin ...
  process(clk32)
  begin ... end process;
  iec_data_o <= cia2_pao(5);
  iec_clk_o <= cia2_pao(4);
  ...
  vicAddr(14) <= (not cia2_pao(0));
  vicAddr(15) <= (not cia2_pao(1));
end architecture;
```

from MiSTer-devel/C64_MiSTer

executed in parallel



Our Contributions

Corpora

- Mined hardware descriptions repositories corpora from GitHub
- 3 popular languages: VHDL, Verilog, SystemVerilog; 8.5M lines of code

Naturalness

- Conducted the first comparative evaluation of the naturalness of hardware descriptions by building language models and reporting standard cross entropy measures
- Compared the naturalness of hardware descriptions written in VHDL, Verilog, SystemVerilog against the naturalness of software written in Java

Assignment Completion Model

- Designed and implemented deep learning models for predicting the right hand side of concurrent assignments in VHDL

Our Contributions

Corpora

- Mined hardware descriptions repositories corpora from GitHub
- 3 popular languages: VHDL, Verilog, SystemVerilog; 8.5M lines of code

Naturalness

- Conducted the first comparative evaluation of the naturalness of hardware descriptions by building language models and reporting standard cross entropy measures
- Compared the naturalness of hardware descriptions written in VHDL, Verilog, SystemVerilog against the naturalness of software written in Java

Assignment Completion Model

- Designed and implemented deep learning models for predicting the right hand side of concurrent assignments in VHDL

Hardware Descriptions Corpora

- Mine 100 top repositories from GitHub (ranked by the number of stars) for VHDL, Verilog, SystemVerilog
- Keep only parsable files (using open-source parsers generated using ANTLR)
- Filter out duplicate files

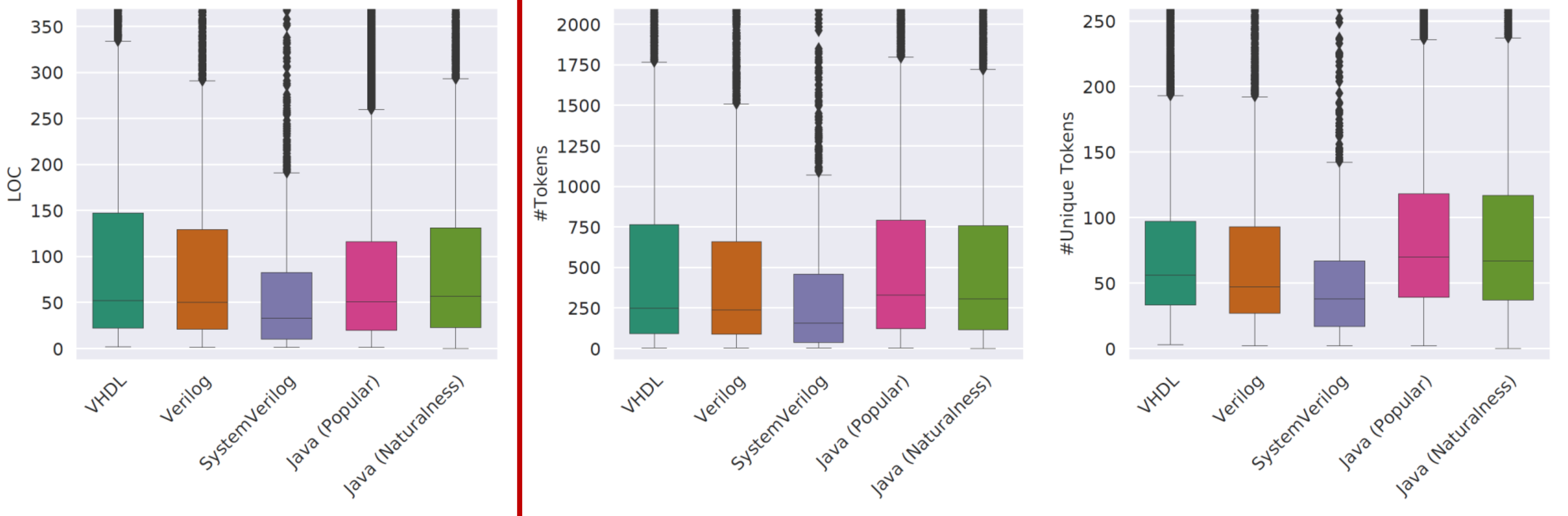
Corpus	#Repos	#Parsable Files	%Duplicate Files	#Unique Files	LOC	#Tokens	Vocab. Size
VHDL	100	13,554	15.5%	11,459	4,759,308	14,572,639	227,117
Verilog	100	7,219	4.8%	6,869	3,433,764	8,238,560	273,893
SystemVerilog	100	2,021	6.5%	1,890	317,886	925,656	28,693

Java Corpora

- Java (Popular): top 10 repositories from GitHub at the time of our study
- Java (Naturalness): 10 repositories from ICSE’12 On the Naturalness of Software

Corpus	#Repos	#Parsable Files	%Duplicate Files	#Unique Files	LOC	#Tokens	Vocab. Size
VHDL	100	13,554	15.5%	11,459	4,759,308	14,572,639	227,117
Verilog	100	7,219	4.8%	6,869	3,433,764	8,238,560	273,893
SystemVerilog	100	2,021	6.5%	1,890	317,886	925,656	28,693
Java (Popular)	10	32,294	3.2%	31,264	6,672,160	23,502,694	387,812
Java (Naturalness)	10	9,886	2.6%	9,630	2,457,854	6,926,953	147,682

Corpora Statistics



(a) LOC (b) Number of all tokens (c) Number of unique tokens

- Hardware descriptions in VHDL are more verbose than Verilog
- Hardware descriptions in SystemVerilog is shorter
- #Tokens and #Unique Tokens are higher in Java repositories than HDL repositories
- #Tokens and #Unique Tokens are smaller in SystemVerilog than VHDL and Verilog

Our Contributions

Corpora

- Mined hardware descriptions repositories corpora from GitHub
- 3 popular languages: VHDL, Verilog, SystemVerilog; 8.5M lines of code

Naturalness

- Conducted the first comparative evaluation of the naturalness of hardware descriptions by building language models and reporting standard cross entropy measures
- Compared the naturalness of hardware descriptions written in VHDL, Verilog, SystemVerilog against the naturalness of software written in Java

Assignment Completion Model

- Designed and implemented deep learning models for predicting the right hand side of concurrent assignments in VHDL

Naturalness: Methodology

- Following the methodology used in ICSE'12 On the Naturalness of Software

1. Randomly partition the corpus into 10 equally sized folds
2. Train a language model on 9 folds, and apply it on the remaining fold
3. Compute the average cross entropy as a measurement of the naturalness

next slide

n-gram language model, n = {1,...,10}

$$P(\mathcal{D}) = P(w_1^m) = \prod_{i=1}^{m} P\left(w_i \middle| w_1^{i-1}\right)$$

$$\approx \prod_{i=1}^{m} P\left(w_i \middle| w_{i-n+1}^{i-1}\right)$$

cross entropy

$$H(\mathcal{D}) = -\frac{1}{\mathrm{len}(\mathcal{D})} \log_2 P(\mathcal{D})$$

Naturalness: Repository Level vs. Language Level

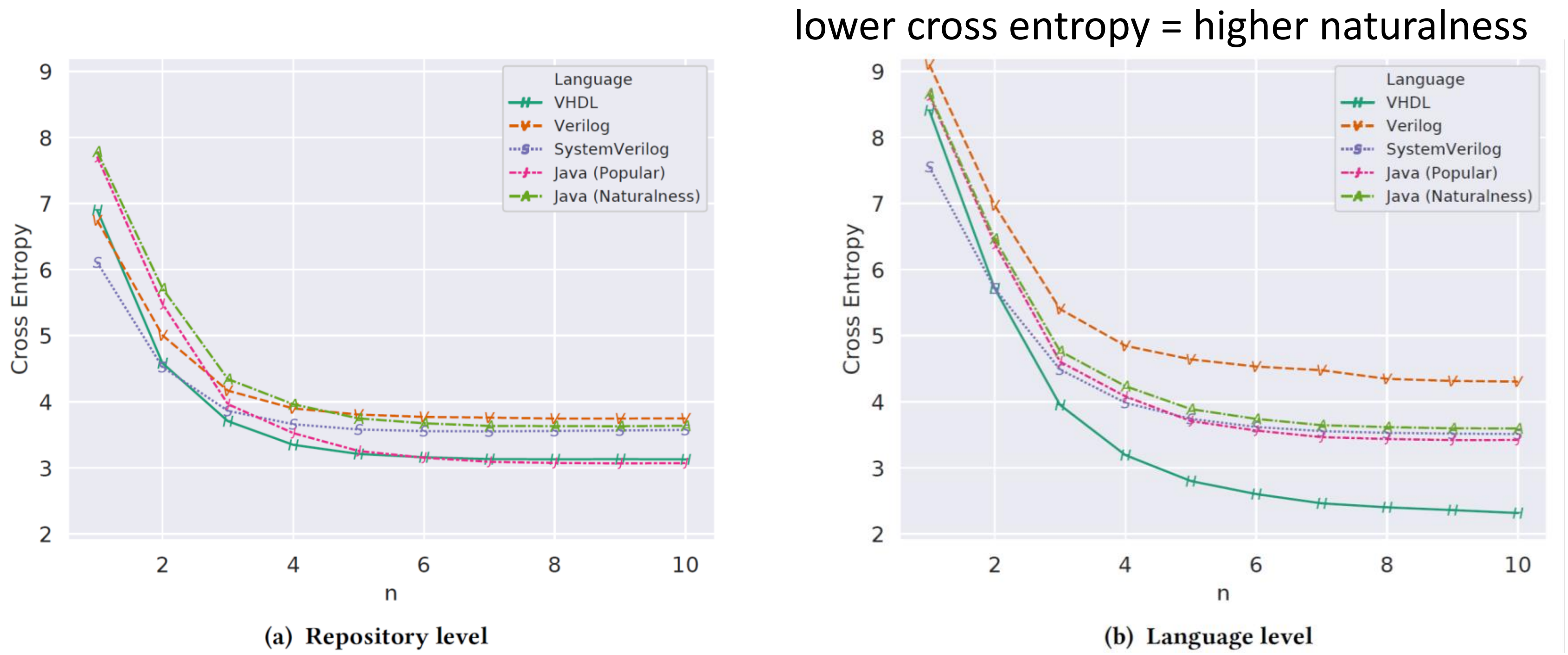
- Repository level (ICSE'12 On the Naturalness of Software)
  - Consider each repository as a (mini) corpus and compute its naturalness
  - Report the average among all repositories

Accounts for variabilities across different repositories

- Language level (ICSE'19 Natural Software Revisited)
  - Consider all repositories of one programming language as a single corpus

Measures the regularity of each language as a whole

Naturalness: Analysis (1/2)

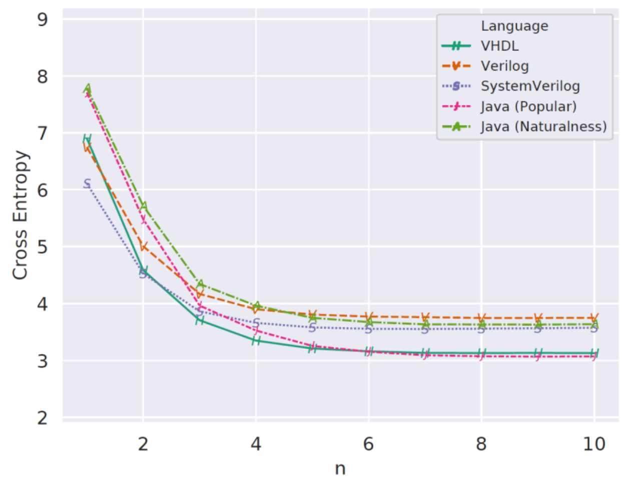


- Cross entropy monotonically drops as n increases
- The decline of cross entropy saturates at around 4-grams for hardware descriptions (similar to ICSE’12 On the Naturalness of Software)

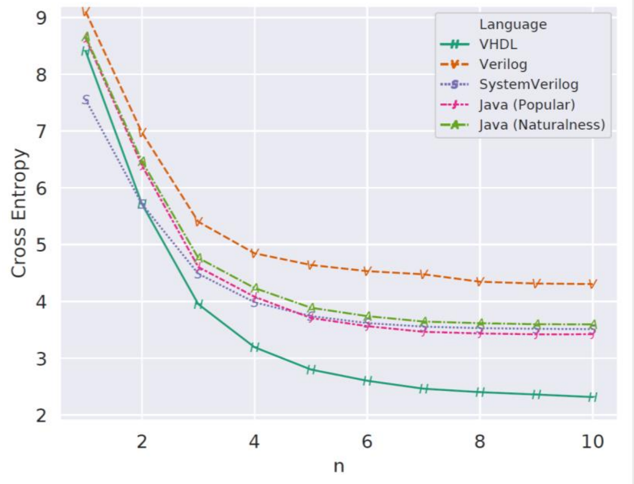
lower cross entropy = higher naturalness

Naturalness: Analysis (2/2)

- Comparisons of cross entropies of different corpora:
  - Repository level, lower n:
    VHDL ≈ Verilog ≈ SystemVerilog < Java(Popular) ≈ Java(Naturalness)
  - Repository level, higher n:
    VHDL ≈ Java(Popular) < Verilog ≈ SystemVerilog ≈ Java(Naturalness)
  - Language level:
    VHDL < SystemVerilog ≈ Java(Popular) ≈ Java(Naturalness) < Verilog
- Hardware descriptions show clear properties of naturalness
- VHDL code has the highest naturalness among the 3 HDLs, and is higher than that of Java software at the repository level



(a) Repository level



(b) Language level

Our Contributions

Corpora

- Mined hardware descriptions repositories corpora from GitHub
- 3 popular languages: VHDL, Verilog, SystemVerilog; 8.5M lines of code

Naturalness

- Conducted the first comparative evaluation of the naturalness of hardware descriptions by building language models and reporting standard cross entropy measures
- Compared the naturalness of hardware descriptions written in VHDL, Verilog, SystemVerilog against the naturalness of software written in Java

Assignment Completion Model

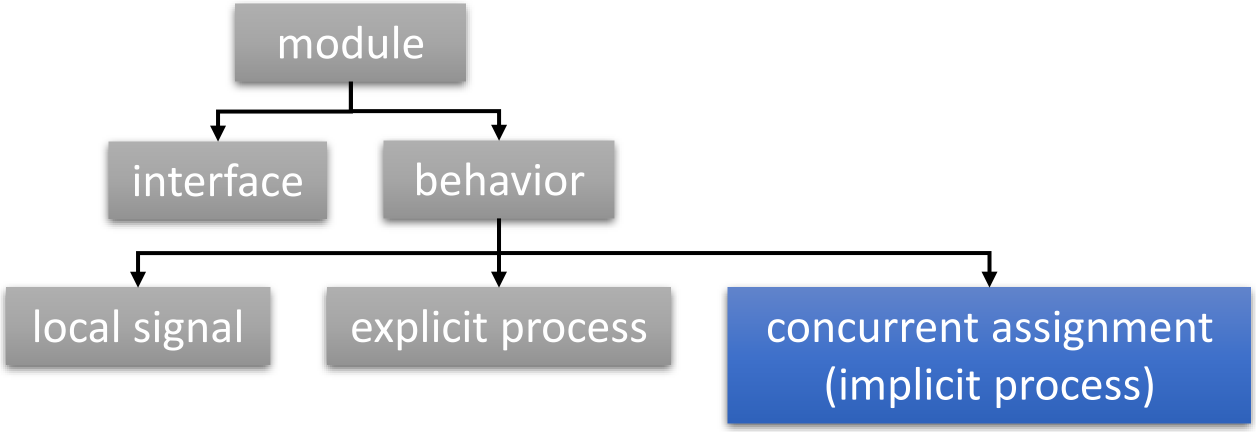
- Designed and implemented deep learning models for predicting the right hand side of concurrent assignments in VHDL

Assignment Completion: Task

```
entity fpga64_sid_iec is
  port(...
    clk32 : in std_logic;
    uart_txd : out std_logic; ...
    uart_dcd_out: out std_logic; );
end fpga64_sid_iec;
architecture rtl of fpga64_sid_iec is ...
  signal cia2_pao: unsigned(7 downto 0);
  signal cia2_pbo: unsigned(7 downto 0);
  signal vicAddr: unsigned(15 downto 0); ...
begin ...
  process(clk32)
  begin ... end process;
  iec_data_o <= cia2_pao(5);
  iec_clk_o <= cia2_pao(4);
  ...
  vicAddr(14) <= (not cia2_pao(0));
  vicAddr(15) <= (not cia2_pao(1));
end architecture;
```

from MiSTer-devel/C64_MiSTer

executed in parallel

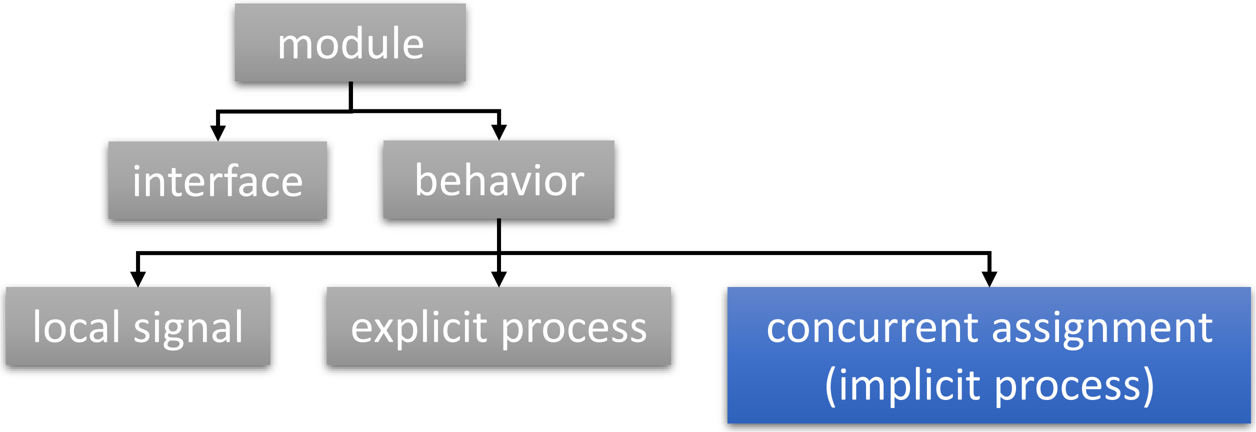


Assignment Completion: Task

- Given the left hand side of a concurrent assignment, predict the value on the right hand side to be assigned

```
entity fpga64_sid_iec is
  port(...
    clk32 : in std_logic;
    uart_txd : out std_logic; ...
    uart_dcd_out: out std_logic; );
end fpga64_sid_iec;
architecture rtl of fpga64_sid_iec is ...
  signal cia2_pao: unsigned(7 downto 0);
  signal cia2_pbo: unsigned(7 downto 0);
  signal vicAddr: unsigned(15 downto 0); ...
begin ...
  process(clk32)
  begin ... end process;
  iec_data_o <= cia2_pao(5);
  iec_clk_o <= cia2_pao(4);
  ...
  vicAddr(14) <= (not cia2_pao(0));
  vicAddr(15) <=         ?         ;
end architecture;
```

from MiSTer-devel/C64_MiSTer



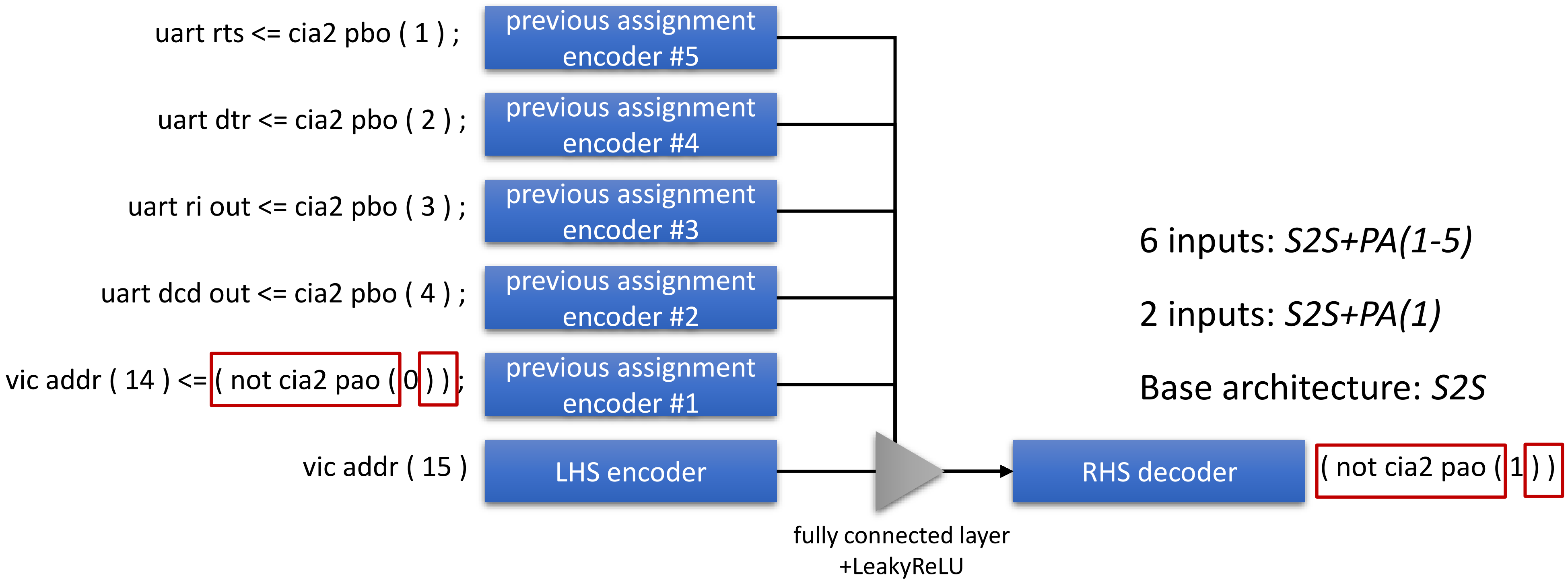
Neural Model Architecture

- Underlying framework: sequence-to-sequence architecture
  - Encodes a sequence into a deep representation, and predicts a target sequence
- Novel architectures to capture HDL-specific characteristics
  1. Multi-source architectures to encode more previous assignments context
  2. Utilizing the types of signals
  3. Ensembling multiple sequence-to-sequence models to capture the parallel nature of HDLs

Base architecture: *S2S*

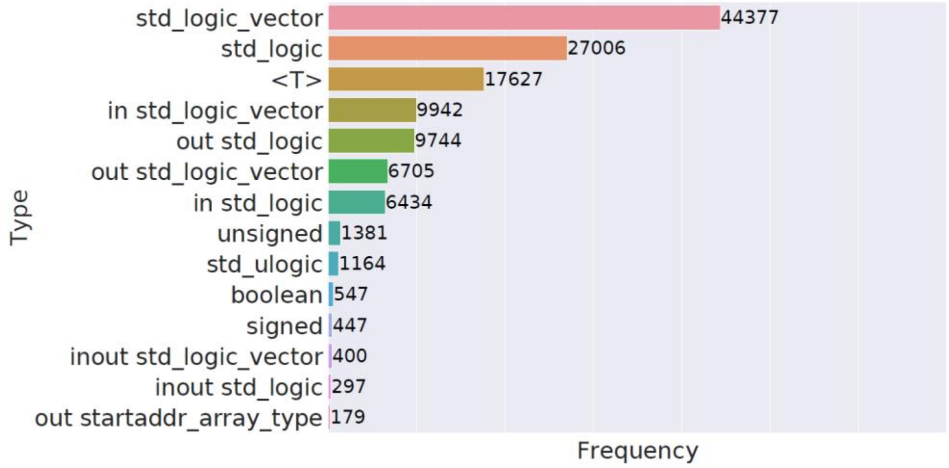
Our Architecture (1/3)

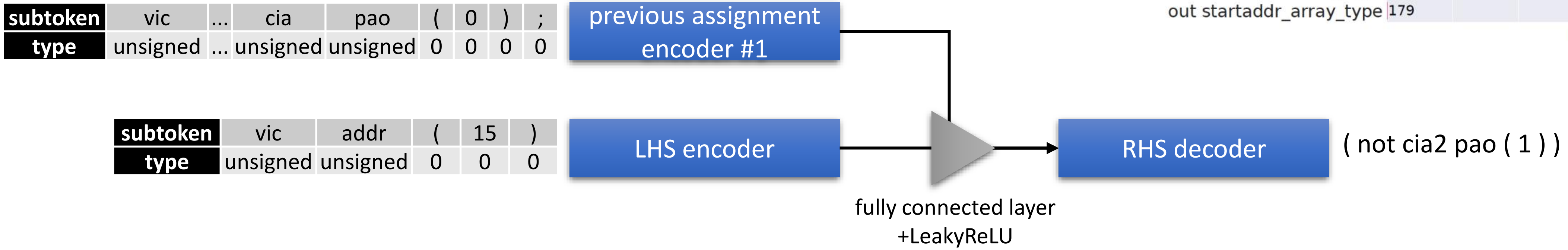
- Multi-source architectures to encode more previous assignments context



Our Architecture (2/3)

- Utilizing the types of signals
  - 14 types: 13 popular types + <T> representing all other types
  - Encode each type as a one-hot embedding
  - Concatenate type embeddings to word embeddings



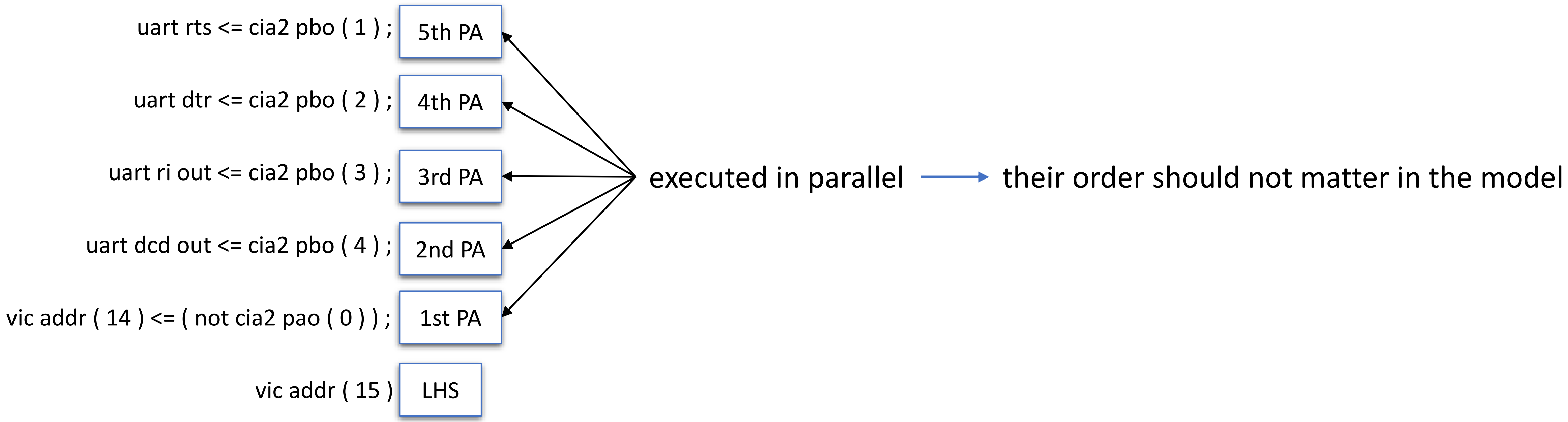


2 inputs, without type: *S2S+PA(1)*

2 inputs, with type: *S2S+PA(1)+Type*

Our Architecture (3/3)

- Ensembling multiple sequence-to-sequence models to capture the parallel nature of HDLs

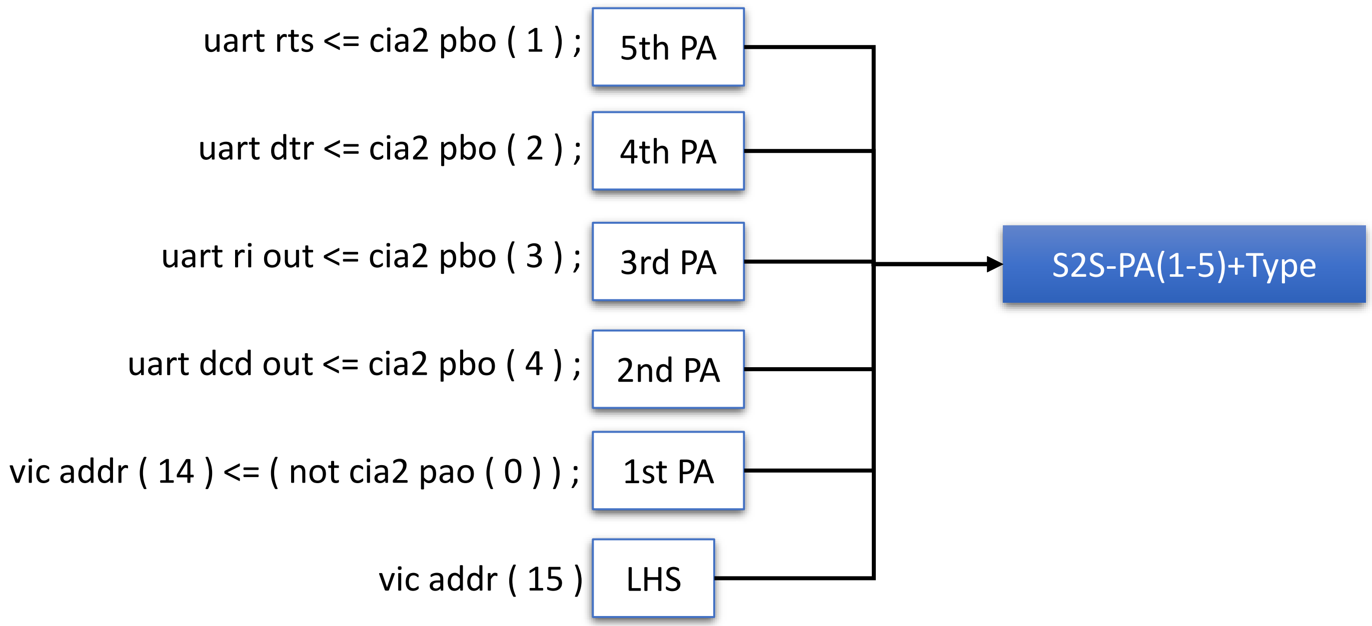


expected RHS: (not cia2 pao (1))

Our Architecture (3/3)

- Ensembling multiple sequence-to-sequence models to capture the parallel nature of HDLs

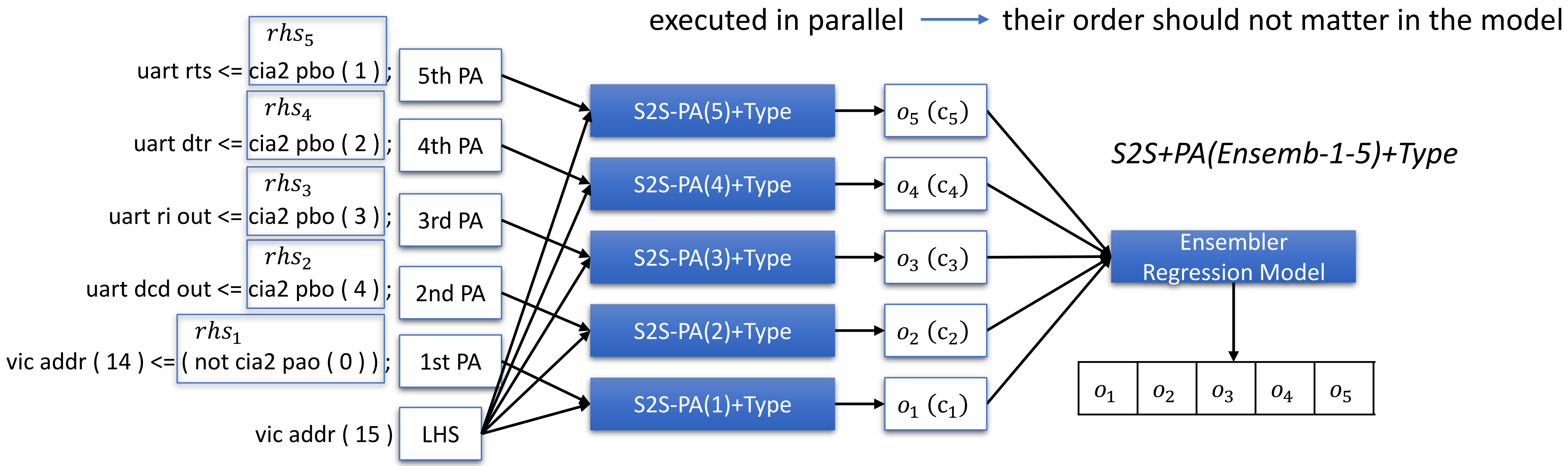
executed in parallel → their order should not matter in the model



expected RHS: (not cia2 pao (1))

Our Architecture (3/3)

- Ensembling multiple sequence-to-sequence models to capture the parallel nature of HDLs



expected RHS: (not cia2 pao (1))

Dataset

- Extract all concurrent assignments from our VHDL corpus
- Split to training, development, and testing sets with a ratio of 80%:10%:10%
    1. Random shuffle all files
    2. Take enough files to obtain ~10% assignments for the testing set
    3. Take enough files to obtain ~10% assignments for the development set
    4. Assignments from other files (~80%) go into the training set

Statistic	All	Training	Development	Testing
#Assignments	49,982	39,986	4,998	4,998
Avg. LHS length	4.10	4.11	4.06	4.10
Avg. RHS length	8.55	8.56	8.51	8.51

Evaluation: Baselines and Models

- Rule-based baseline
  - Copy the RHS of the 1st PA
- Language model baseline
  - RNN language model using LHS + 1st PA as context: *RNNLM+PA(1)*
  - Not good at handling long context: *RNNLM+PA(1)* is better than *RNNLM+PA(1-5)*
- Sequence-to-sequence models
  - Base architecture: *S2S*
  - 2 inputs: *S2S+PA(1)*
  - 2 inputs with type: *S2S+PA(1)+Type*
  - 6 inputs with type: *S2S+PA(1-5)+Type*
  - Ensemble model: *S2S+PA(Ensemb-1-5)+Type*

Evaluation: Metrics

- Compute the similarity between the predicted RHS vs. human-written RHS for each data in testing set, and report the average scores
- Similarity measurements:
  - xMatch: exactly match = 100%, otherwise = 0%
  - Acc: subtoken level accuracy $= \frac{\mathrm{len}(\{i|pred[i]=tgt[i]\})}{\max\left(\mathrm{len}(pred),\mathrm{len}(tgt)\right)}$
  - BLEU: range 0-100, calculates the percentage of n-grams in the predicted RHS that also appear in human-written RHS, averaging across n ∈ $\{1,2,3,4\}$ and using a brevity penalty to eliminate the impact of the number of subtokens predicted

Evaluation: Key Results (1/2)

Model	BLEU	Acc [%]	xMatch [%]
Rule-based Baseline	29.4	38.1	8.8
RNNLM+PA(1)	18.0	22.0	8.2
S2S+PA(Ensemb-1-5)+Type	**37.3**	**48.0**	**19.1**

- The best model is the model that ensembles multi-source sequence-to-sequence models for 5 previous assignments with utilizing types of signals

Evaluation: Key Results (2/2)

Model	BLEU	Acc [%]	xMatch [%]
S2S+PA(Ensemb-1-5)+Type	**37.3**	**48.0**	**19.1**
S2S+PA(1-5)+Type	24.4	28.2	11.4
S2S+PA(1)+Type	25.8	30.4	14.1
S2S+PA(1)	25.4	30.0	14.4
S2S	19.6	21.9	12.3

- Using more previous assignment context and type embedding improved the performance over the base architecture (S2S)
- Ensembling handles the previous assignment context more effectively than only using multi-source architecture

Conclusions

Corpora

- Mined hardware descriptions repositories corpora from GitHub
- 3 popular languages: VHDL, Verilog, SystemVerilog; 8.5M lines of code

Naturalness

- Conducted the first comparative evaluation of the naturalness of hardware descriptions by building language models and reporting standard cross entropy measures
- Compared the naturalness of hardware descriptions written in VHDL, Verilog, SystemVerilog against the naturalness of software written in Java

Assignment Completion Model

- Designed and implemented deep learning models for predicting the right hand side of concurrent assignments in VHDL

https://github.com/EngineeringSoftware/hdlp

Pengyu Nie <pynie@utexas.edu>