ML-Based Verification and Regression Automation

Abhishek Chauhan, Asif Ahmad

Agnisys, abhishek.ch@agnisys.com;

Agnisys, asif@agnisys.com

ABSTRACT

Verification is a labour-intensive work. Machine Learning (ML) algorithms can be used to eliminate the manual work. This paper shows ML techniques used to reduce or eliminate manual verification work which includes validating waveforms for the written design checks and manual regression in the context of verifying complex hardware designs. These techniques are based on the fundamental problem of looking for similar "difference patterns" and grouping them. When the differences are grouped together, it reduces or eliminates the need to manually verify them. These techniques are not specific to design verification but can be used for any similar problem.

INTRODUCTION

With the continued increasing complexity of chip designs, the process of verification and validation through regression framework takes up a vast amount of time. Ideally, self-checking tests are written in a Universal Verification Methodology (UVM) environment and the regression process ensures that there is a constant forward march towards code coverage and functional coverage. However, to get to that point, often verification engineers rely on waveforms to manually debug and isolate the errors. These errors then need to be manually verified with each test in their waveform. What we bring here is our current state of experiment and development to describe how one can easily handle their differences, both text and waveform based, by grouping the errors and then validating them in a collective way. The objective of this approach is to categorize each unique "diff" between the original simulation and the revised simulation, which could be because of the changes to the design or the testbench environment. So, when design changes are detected, this approach aims to reduce the tedious task of verifying each one of them manually.

MACHINE LEARNING SPACE USED

AUTOENCODER

An autoencoder neural network is an unsupervised machine learning algorithm that applies backpropagation, i.e., it looks for the minimum value of the error function in weight space by setting the target values equal to the inputs. Basically, they are used to reduce the size of the inputs into smaller representation. Thus, to learn the representation of inputs as the first step, we used autoencoder networks. The network produced the reconstruction loss of the inputs and attempt was made to reconstruct the inputs at the output. This seems to be the best way of learning weights to produce intermediate representation of inputs. The reconstruction loss is calculated using predictable outputs and inputs.

$$\textit{Reconstruction loss} = ||x_{input} - x_{output}||^2$$

Figure 1:-Reconstruction loss though predicted inputs and outputs

SIAMESE NETWORK

In this era of deep learning, neural networks rely mostly on abundance of data to perform well. But in some cases, data collection seems to be an issue. Here, siamese network is introduced which uses lesser amount of data to get better predictions. This ability to learn from limited data has made siamese network popular in the recent years. We too have drawn inspiration from such model to achieve our objective. In this model, the weights are shared across multiple inputs, and the vectors are computed for each input using the same set of weights to produce embedding for each input. The model is a feed-forward network (wherein connections between the nodes do not form a cycle), comprises of convolution layers (to analyse visual imagery), similarity matrix and a fully connected layer with dimension of 100.

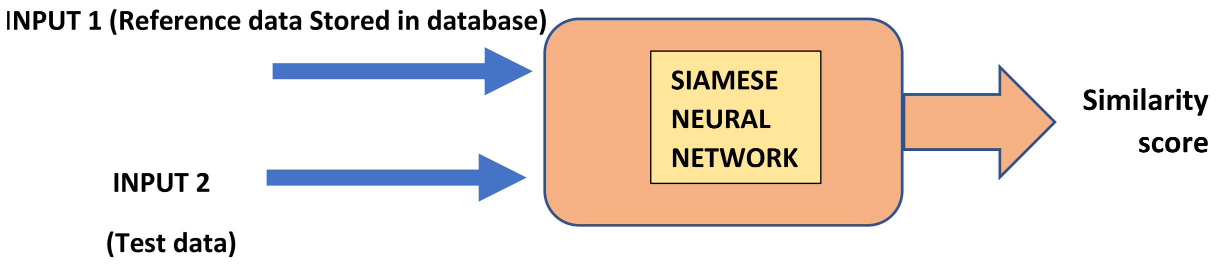


Figure 2:- A Siamese network

DBSCAN (Clustering Algorithm)

Density Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm groups a set of data points in a way such that similar data points are grouped together in an unsupervised learning method. This clustering technique is much more efficient to create an arbitrary shaped clusters and to detect outliers. We have used DBSCAN to form the groups in between vectors wherever they exist. This algorithm has two following parameters -

- ***min_radius* -** Minimum radius or distance between two data points. A data point is part of group, if its distance is less than or equal to minimum radius.
- ***min_neigbours-*** To form a group, data point should be associated with minimum number of neighbouring data point.

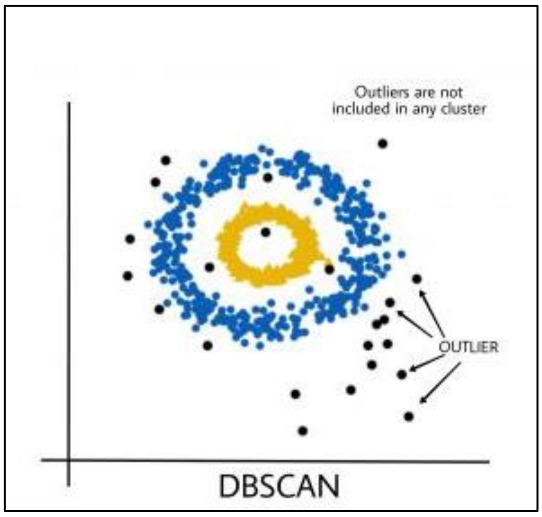


Figure 3:-Groups and outliers formed using DBSCAN

Embedding

Embedding is a low-dimensional learned continuous vector representation of discrete variables and it plays an important role in neural networks by mapping tokens to a fixed vector and makes model easier to use. They can either be learned during training or can be copied from another source since various sources are readily available for pre-trained embeddings. To skip the learning and time taken in this process, we copied a pre-trained embedding of sub-word in one of trained GPT2 model. Using this pre-trained embedding, our new model converged faster.

Back-propagation

Back propagation is a widely used algorithm for the training of neural networks. It basically calculates gradient of the given loss function with respect to the weights of the neural network. It updates the weight in such a fashion so as to minimise the loss function value and thus the model is able to converge closer to the given outputs. It is a learning algorithm for neural networks and makes models ready to predict outputs.

Training strategy used

We first trained autoencoder with pre-trained embeddings over a variety of inputs. Training was done with a learning rate of '*1e-5'* with Adam optimizer and root mean square loss function. Later, we copied the trained weights of the representation layer.

A Siamese network was then built where the initial layers were the same as the autoencoder. The copied weights were set for initial layers. The output layer was fully connected layer (dense) with size 100. The training strategy, i.e., the loss function was chosen after evaluating the performance. Training was done in such a way that with every run three pairs were chosen, namely- anchor, positive and negative tag. The anchor and the positive tags formed a pair which was like each other. Similarly, anchor and negative tags formed a pair which was dissimilar. The approach was such that the model handled both type of cases together and thus a loss was calculated based on the obtained scores. The distance between each pair of samples in an n-dimensional space known as the 'Euclidean Distance' was used as the calculating distance.

$$\boxed{score = ||emb_1 - emb_2||^2}$$

Figure 4:-Euclidean distance

To calculate the loss, the difference of distances between the pairs was considered.

$$\boxed{basic\ loss = (score_{positive} - score_{negative})}$$

Figure 5:-Loss function

$$\boxed{loss = max\ (basic\ loss, 0)}$$

Figure 6:-Non-negative loss function

Our strategy was to obtain well margined groups in the vector space. So, we introduced an additional parameter in the loss function, called margin, and specified it as 'm' to add an additional margin. Here 'm' refers to the maximum margin in the loss function. It is a hyperplane that separates two classes of inputs while giving space to hyperplane and nearest member. The 'm' here requires difference of distances between anchor-positive and anchor-negative to be greater than 'm'. The aim of 'm' is to push negative input sample outside the boundary by margin 'm' while keeping positive sample intact and closer. In our case 'm' is set to 0.4. Now the revised loss becomes -

$$\boxed{\textit{basic loss (revised)} = (\textit{score}_{positive} - \textit{score}_{negative}) + m}$$

Figure 7:-Figure 7:-Revised loss function with margin 'm'

$$\boxed{\textit{loss (revised)} = \max(\textit{basic loss}, 0)}$$

Figure 8:- Non-negative loss function with margin 'm'

These values are used in producing well defined margins between two distinct cases. Because of the additional margin parameter ‘m’, clustering algorithm was able to locate groups in the vector space. We observed that the differences alone between the two files were not enough to comment on the similarity and thus grouping could not be done based on the differences alone. It may happen that despite similar differences across logs, there may have a different context. And thus considering the similar context, we figured out that it is must to group these differences also. So, the changes were made to incorporate requisite context too as input in the model. The model was then trained with 100 training steps and a batch size of 32 and ‘m’ with 0.4 in loss function. (Refer figure 7 & 8).

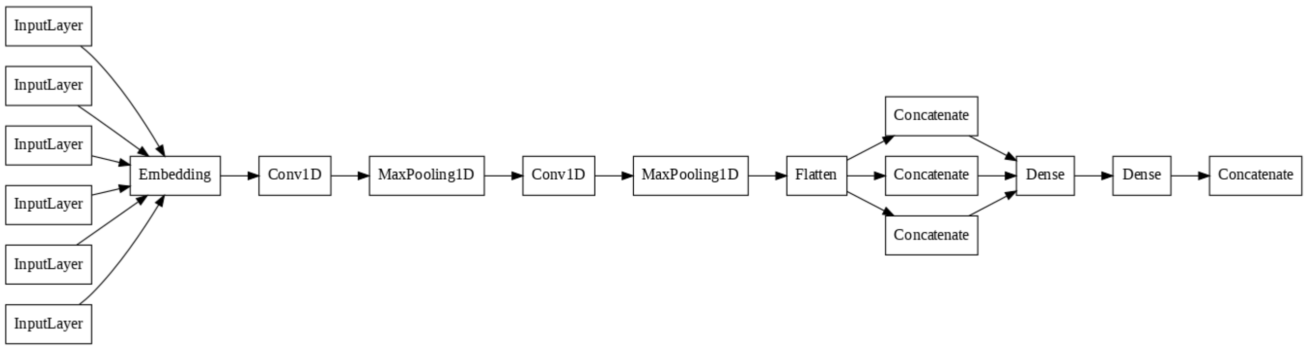


Figure 9:-Siamese neural network (used architecture)

Some of the working codes: -

Loss construction code

```
def triplet_loss (y_true, y_pred, m = 0.4):
    """Implementation of the triplet loss function
    Arguments:
    y_true -- true labels, required when you define a loss in Keras,
                you don't need it in this function.
    y_pred -- python list containing three objects:
    loss -- real number, value of the loss """
    total_length = y_pred.shape.as_list()[-1]
    anchor = y_pred[:,0:int(total_length*1/3)]
    positive = y_pred[:,int(total_length*1/3):int(total_lenght*2/3)]
    negative = y_pred[:,int(total_length*2/3):int(total_lenght*3/3)]

    # distance between the anchor and the positive
    pos_dist = K.sum(K.square(anchor-positive),axis=1)
    # distance between the anchor and the negative
    neg_dist = K.sum(K.square(anchor-negative),axis=1)
    # compute loss
    basic_loss = pos_dist-neg_dist+ m
    loss = K.maximum(basic_loss,0.0)
    return losss
```

Model construction code

```
ids_input_anchor        = Input(shape=[max_length_dl,], name='ids_input_anc
hor',dtype='int32')
gold_input_anchor       = Input(shape=[max_length_dl,], name='gold_input_an
chor',dtype='int32')

ids_input_positive      = Input(shape=[max_length_dl,], name='ids_input_pos
itive',dtype='int32')
gold_input_positive     = Input(shape=[max_length_dl,], name='gold_input_po
sitive',dtype='int32')

ids_input_negative      = Input(shape=[max_length_dl,], name='ids_input_neg
ative',dtype='int32')
gold_input_negative     = Input(shape=[max_length_dl,], name='gold_input_ne
gative',dtype='int32')
```

Embedding code

```
embedding = Embedding(input_dim=num_words, output_dim = 768,name="embedding
",weights=[word_embeddings],trainable=False)
```

Anchors code

```
anchor   = concatenate([flat_ids_anchor,flat_gold_anchor], axis=-
1) #fs_ids_gold_anchor
positive = concatenate([flat_ids_positive,flat_gold_positive], axis=-
1) #fs_ids_gold_positive
negative = concatenate([flat_ids_negative,flat_gold_negative], axis=-
1) #fs_ids_gold_negative
```

EVALUATION

Embedding for differences were then produced using the model. We visualized the test data in 3D and converted the 100-dimension data into 3 components. It was observed that there were many groups formed in the vector space and they also had well defined margins. Clustering of vector data was then carried out along with appropriately setting the parameters, *min_radius* and *min_neigbours*. A number of groups were then found. The DBSCAN feature, to put all data points in one group, was used on those data-points that failed to form groups and were rather considered as outlier.

The accuracy of forming groups was found to be around 60%.

Some group labels predicted on evaluation (each colour represents a group in 3D space) –

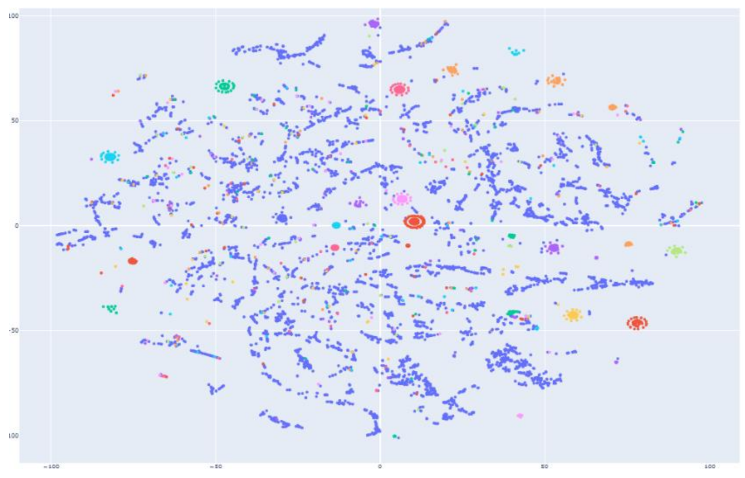


Figure 10:- Groups with outliers

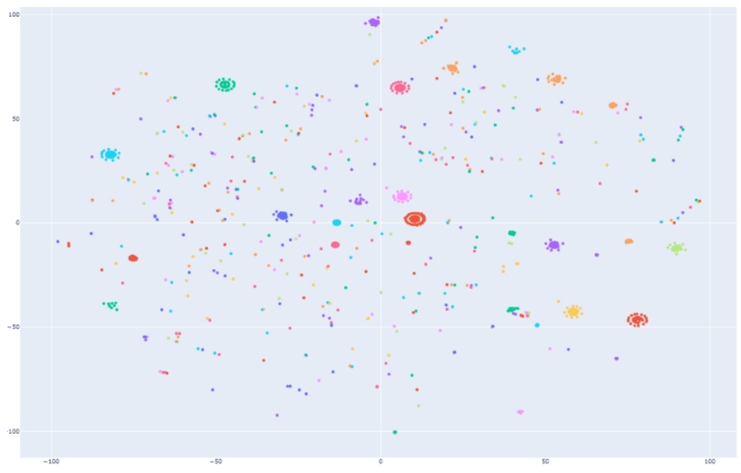


Figure 11:- Groups without outliers

IMPLEMENTATION

(i) Text based regression mechanism

This initial work involved grouping and then categorising text-based data in the regression flow to monitor changes for faster and more precise validation in a continuous integration and continuous development cycle (CICD).

DATA

A sample of around 1500 scenarios and their text-based output were taken for the initial breakthrough. We considered those initial output as the golden output. The CICD program, involving codes changes, led to changes in output of those scenarios. We then compared the existing golden output with the new generated output in an aligned way. The differences were extracted between the two files along with their context.

METHOD

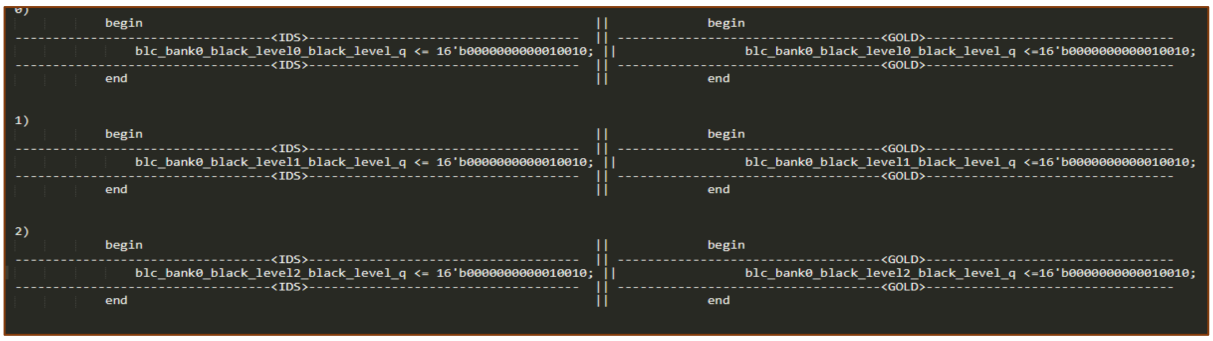
On this initial experiment data, we covered all golden files of the sample regression and aligned their diffs when compared to the newly generated files and dumped the changes in another created diff file for processing.

We extracted those aligned differences with contexts in the created diff file. The differences were then fed to 'siamese model' to produce a vector of 1x100 dimension. Same was performed for other differences and then fed in the batches. Clustering algorithm was then applied to form groups and those groups were dumped in the form of text.

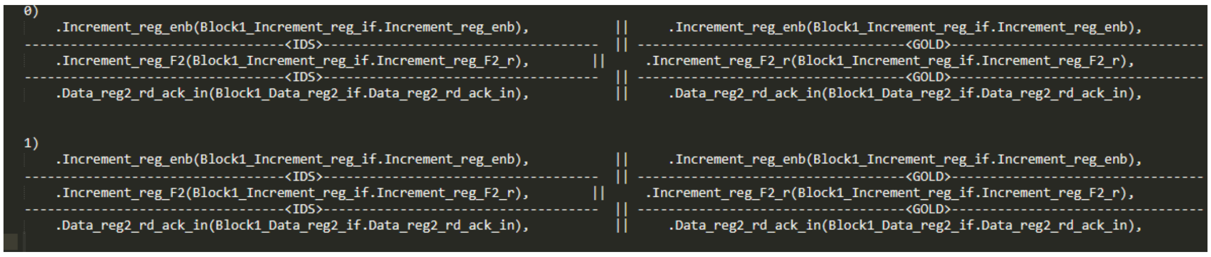
EVALUATION

When cross-verified with our regression environment for validating differences, good and appreciable number of groups with almost same/ identical differences were being created. Each group had a sufficient margin of difference w.r.t other groups. To conclude, we observed that 50% of the total diffs were able to form groups where each group had at-least 5 or more member diffs because of the chosen parameter '***min_neigbours=5***' in clustering. This helped us save time by validating group of diffs across all regression files instead of individual diff in a particular file. Similar scenarios could now be validated in a single go to either accept or reject a newly found difference w.r.t the golden content.

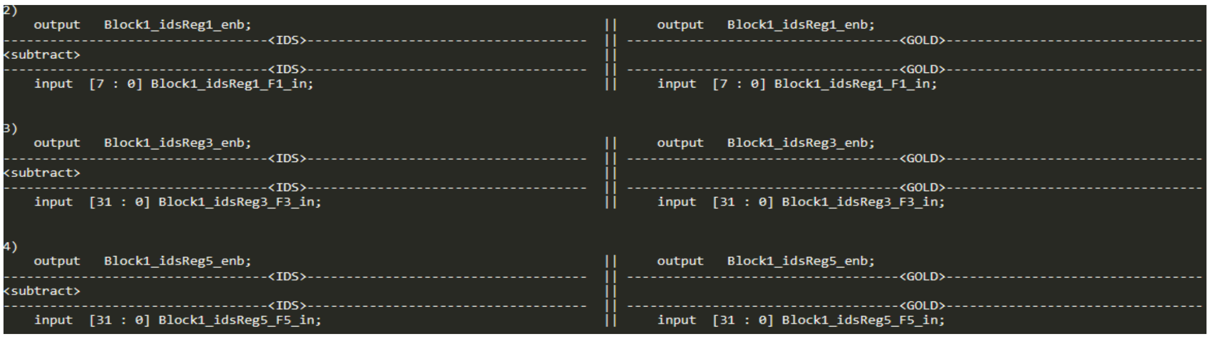
Some example diff groups created-



```
0)
            begin                                              ||            begin
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
                blc_bank0_black_level0_black_level_q <= 16'b0000000000010010; ||              blc_bank0_black_level0_black_level_q <=16'b0000000000010010;
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
            end                                                ||            end


1)
            begin                                              ||            begin
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
                blc_bank0_black_level1_black_level_q <= 16'b0000000000010010; ||              blc_bank0_black_level1_black_level_q <=16'b0000000000010010;
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
            end                                                ||            end


2)
            begin                                              ||            begin
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
                blc_bank0_black_level2_black_level_q <= 16'b0000000000010010; ||              blc_bank0_black_level2_black_level_q <=16'b0000000000010010;
----------------------------------<IDS>-------------------------------------  || ------------------------------------<GOLD>-----------------------------------
            end                                                ||            end
```



```
 0)
    .Increment_reg_enb(Block1_Increment_reg_if.Increment_reg_enb),          ||     .Increment_reg_enb(Block1_Increment_reg_if.Increment_reg_enb),
---------------------------------<IDS>----------------------------------------  || ----------------------------------<GOLD>-----------------------------------------
    .Increment_reg_F2(Block1_Increment_reg_if.Increment_reg_F2_r),       ||     .Increment_reg_F2_r(Block1_Increment_reg_if.Increment_reg_F2_r),
---------------------------------<IDS>----------------------------------------  || ----------------------------------<GOLD>-----------------------------------------
    .Data_reg2_rd_ack_in(Block1_Data_reg2_if.Data_reg2_rd_ack_in),          ||     .Data_reg2_rd_ack_in(Block1_Data_reg2_if.Data_reg2_rd_ack_in),


 1)
    .Increment_reg_enb(Block1_Increment_reg_if.Increment_reg_enb),          ||     .Increment_reg_enb(Block1_Increment_reg_if.Increment_reg_enb),
---------------------------------<IDS>----------------------------------------  || ----------------------------------<GOLD>-----------------------------------------
    .Increment_reg_F2(Block1_Increment_reg_if.Increment_reg_F2_r),       ||     .Increment_reg_F2_r(Block1_Increment_reg_if.Increment_reg_F2_r),
---------------------------------<IDS>----------------------------------------  || ----------------------------------<GOLD>-----------------------------------------
    .Data_reg2_rd_ack_in(Block1_Data_reg2_if.Data_reg2_rd_ack_in),          ||     .Data_reg2_rd_ack_in(Block1_Data_reg2_if.Data_reg2_rd_ack_in),
```



```
2)
    output   Block1_idsReg1_enb;                                     ||     output   Block1_idsReg1_enb;
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
<subtract>                                                           ||
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
    input  [7 : 0] Block1_idsReg1_F1_in;                             ||     input  [7 : 0] Block1_idsReg1_F1_in;


3)
    output   Block1_idsReg3_enb;                                     ||     output   Block1_idsReg3_enb;
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
<subtract>                                                           ||
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
    input  [31 : 0] Block1_idsReg3_F3_in;                            ||     input  [31 : 0] Block1_idsReg3_F3_in;


4)
    output   Block1_idsReg5_enb;                                     ||     output   Block1_idsReg5_enb;
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
<subtract>                                                           ||
----------------------------------<IDS>--------------------------------------  || ------------------------------------<GOLD>--------------------------------------
    input  [31 : 0] Block1_idsReg5_F5_in;                            ||     input  [31 : 0] Block1_idsReg5_F5_in;
```

Figure 12:- some created diff groups

LIMITATION

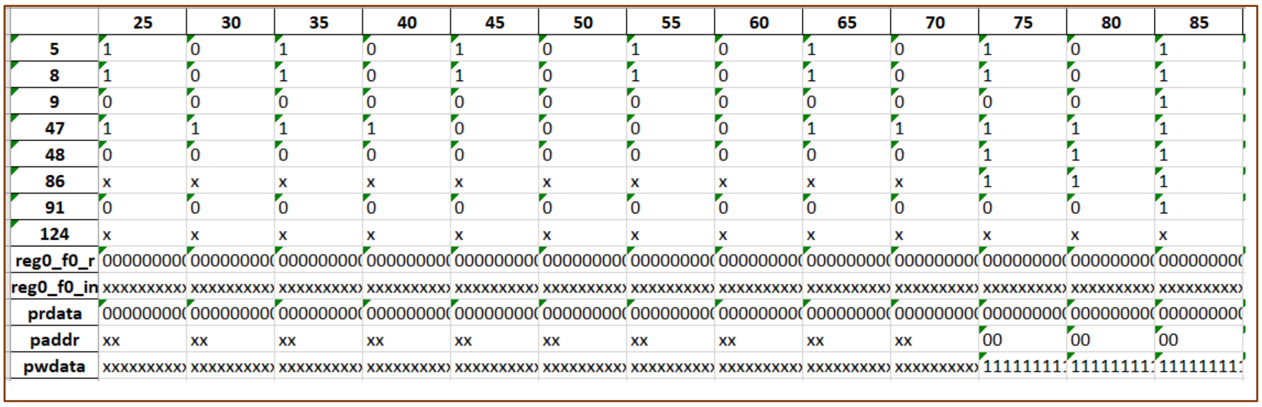
- Absence of structural information of the generated output text.
- Drastic improper alignment of files (golden v/s new) at times.
- Lack of GUI interface for efficient and visually pleasing mechanism and to create a relation with the source code with difference in the comparison files.

(ii) Transaction log-based wave diff mechanism

This second phase of work involved creation of a transaction-based log through some assumptions (or user given data) and then grouping and categorising the transactions with their time stamp to monitor changes in the revised design w.r.t the golden design.

DATA

Sample designs were considered at the very beginning. Simulations were done and their VCD (value change dump) files were dumped along with it. Some tweaks were then made in the same design as adding delays, inserting, or deleting transactions, right shifting, or left shifting transactions w.r.t time stamp, etc. We then processed the VCD files to convert them in signal-time relation table. Then the assumptions that were considered was used to create the intermediate logs. In such intermediate transaction logs, we filled the time gaps with <pad> tokens.

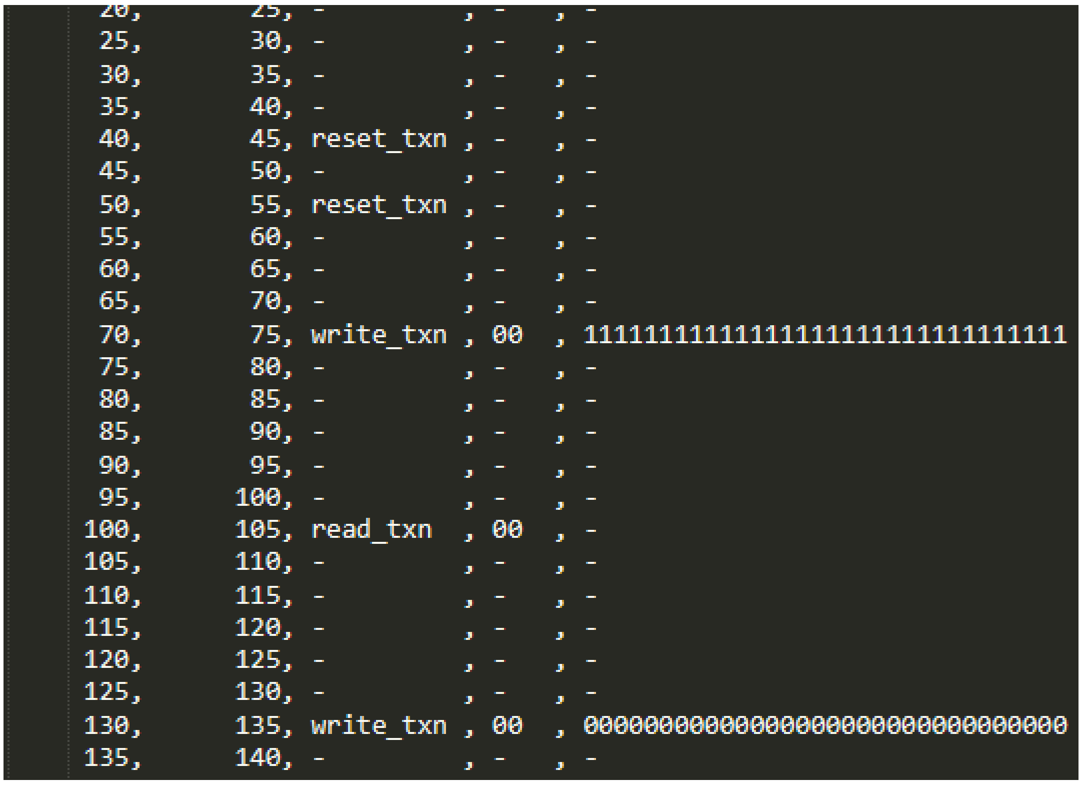


	25	30	35	40	45	50	55	60	65	70	75	80	85
5	1	0	1	0	1	0	1	0	1	0	1	0	1
8	1	0	1	0	1	0	1	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	1
47	1	1	1	1	0	0	0	0	1	1	1	1	1
48	0	0	0	0	0	0	0	0	0	0	1	1	1
86	x	x	x	x	x	x	x	x	x	x	1	1	1
91	0	0	0	0	0	0	0	0	0	0	0	0	1
124	x	x	x	x	x	x	x	x	x	x	x	x	x
reg0_f0_r	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(
reg0_f0_in	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)
prdata	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(	00000000(
paddr	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	00	00	00
pwdata	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	xxxxxxxxx)	11111111	11111111	11111111

Figure 13:- A sample processed VCD data after a simulation

INETERMIDIATE LOG

In the process to align the input cases, we created intermediate logs to set a base to run the model. With the earlier taken assumptions, transactions were identified with their occurring time stamp thereby creating a log of sort for the entire simulation time. Those events were then padded taking a reference time unit to create spread view of all occurring events in the log. These were then fed to the model with context to achieve a group-based diff mechanism.



```
 20,        25, -          , -   , -
 25,        30, -          , -   , -
 30,        35, -          , -   , -
 35,        40, -          , -   , -
 40,        45, reset_txn  , -   , -
 45,        50, -          , -   , -
 50,        55, reset_txn  , -   , -
 55,        60, -          , -   , -
 60,        65, -          , -   , -
 65,        70, -          , -   , -
 70,        75, write_txn  , 00  , 11111111111111111111111111111111
 75,        80, -          , -   , -
 80,        85, -          , -   , -
 85,        90, -          , -   , -
 90,        95, -          , -   , -
 95,       100, -          , -   , -
100,       105, read_txn   , 00  , -
105,       110, -          , -   , -
110,       115, -          , -   , -
115,       120, -          , -   , -
120,       125, -          , -   , -
125,       130, -          , -   , -
130,       135, write_txn  , 00  , 00000000000000000000000000000000
135,       140, -          , -   , -
```

Figure 14:- A sample intermediate transaction log created

METHOD

We performed diff on both the transaction logs, and extracted out aligned differences with contexts. The logs were then fed to 'siamese model' to produce an embedding/vector of 1x100 dimension. Same was performed for each difference in sample batches. After that, clustering algorithm was applied to form groups and dump all those groups in the form of text.

EVALUATION

The groups were found w.r.t the created transaction logs, but the absence of context in the simulation data and the dynamic time factor reduced the expected accuracy and usability of this approach.

The alignment of time factor would create a 3-dimension vector space to identify events, like, delays and context would allow to determine better groups with information of contextual transactions data in the compared logs.

LIMITATION

- Absence of good context w.r.t the simulation data or usage of context for transactions.
- Dependency on availability of transaction information (assumptions).
- Lack of GUI interface, to see the exact difference groups in the running wave.
- Inclusion of the time factor in a more dynamic way to align groups.

CONCLUSION

We have been able to handle text-based regression automation with our approach to a good and useful extent by classifying the diffs into defined groups and validating them to either accept or reject the changes.

Verification and validation of changes in a design with the available simulation data and the VCD files using our approach by grouping the occurring changes has been handled to a certain extent reducing a little time and effort to verify waveforms especially in the absence of self-checking tests.

FUTURE SCOPE

- Developing of a proper GUI interface for better debugging and visual appeal to the created groups for validation.
- Researching more about pattern alignment and exploring the identification of a transaction within the simulation data. Dynamic time wrapping algorithm used in searching music, speech recognition, etc., may be useful to align the waves to compare between different simulation.
- Usage of time as a dynamic factor to consider delays and shifts in transactions accordingly.

REFERENCES

- **Siamese Networks for Similar Question Retrieval**
    - https://www.aclweb.org/anthology/P16-1036.pdf

- **A Convolutional Siamese Network for Developing Similarity Knowledge**
    - *http://ceur-ws.org/Vol-2028/paper8.pdf*

- **Facilitating Transactions in VHDL and System Verilog by Rich Edelman, Mentor, A Siemens Business.**
- **Joint Pairwise Learning and Image Clustering Based on a Siamese CNN**

    - https://www.researchgate.net/publication/327995145_Joint_Pairwise_Learning_and_Image_Clustering_Based_on_a_Siamese_CNN

- **A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases**
    - https://www.researchgate.net/publication/44250717_A_Density_Based_Algorithm_for_Discovering_Density_Varied_Clusters_in_Large_Spatial_Databases

- **Reducing the Dimensionality of Data with Neural Networks**
    - DOI: 10.1126/science.1127647

- **Triplet Loss and Online Triplet Mining in TensorFlow**
    - https://omoindrot.github.io/triplet-loss