International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 07 | July 2020 www.irjet.net p-ISSN: 2395-0072

Verification of Convolution Neural Network using Universal Verification Methodology

Padmashree B N[1], Ashwini V[2]

[1]*Padmashree B N, MTech student[VLSI], Dept. of ECE, BMS College of Engineering, Karnataka, India*
[2]*Ashwini V, Assistant Professor, Dept. of ECE, BMS College of Engineering, Karnataka, India*

Abstract - *With the passage of time, the semiconductor industry has seen a drastic change in the design and production of functional chips. From the development of a simple MOSFET to the current age where a whole functionality is integrated into tiniest of chips available. In order to test these complex designs, the former, testbench verification methods using the verilog language is no longer sufficient. Hence the verification methodology such as "systemverilog" was formed. This paper was made to understand the way of verifying a design using an object oriented Programming based Universal Verification Methodology which is a standard way of writing testbenches using Systemverilog Language. The design in question is a machine learning algorithm: "The Convolution Neural network" which was implemented in the verilog language. Hence the project also encompasses the part where a study on the working of this particular algorithm was made.*

Key Words: **Convolution Neural Network, SystemVerilog, Universal Verification Methodology, Neural Networks, Verification Methodology, AlexNet.**

1. INTRODUCTION

Verification of designs is an important part of developing a SoC (System on chip). In this age the technology is so advanced and this results in the increase in the complexity of the designed hardware. These' hardware are not only complex but are really expensive to be fabricated in real time. Aspects like turnaround time, density and development cost plays a crucial role in fabrication of such complex designs. Any errors seen the design after the fabrication would cost dearly in order to be rectified since fabrication process is expensive. This increases the demand for much advanced verification methodologies to be adapted in order to generate a functionally sound chip. This led to the development of verification methodologies with multiple features that enabled the verification process to be thorough. One such language that was developed for the purpose was SystemVerilog.

While system verilog is a language similar to verilog using which design and verification can be written, UVM is a framework of systemverilog classes. This framework can be used to build testbenches that are fully functional. The only requirement to learn the Universal Verification Methodology is to know system verilog since all of UVM testbenches are written in systemverilog using its concepts and jargons.

In this paper the concepts of UVM are utilized to verify the functionality of a convolution neural network with reference to a working model written in python language.

Yann LeCun proposed the Convolution Neural Network in the year 1988. It was developed based on a study on how human minds learn to recognize certain objects over repetitive exposure toward it.

Convolution Neural networks are used to train machines or computers into learning features of objects so as to recognize them by detecting the features of the said object. A general architecture is followed in training a machine to do this. This architecture consists of layers of certain repetitive operations that are performed on the input (say image). The different layers and the operation performed by them are as follows:

1. Input: The image pixel of the image that is to be tested is given to the convents. The size of the image depends on the architecture in used

2. Convolution layer: This layer is used to help the machine learn the features of a particular image. This is done by convolving the images with certain filters and highlighting the peculiar aspects of an image that helps in learning it distinctively. Importance is given to certain aspects of the image by assigning weights to it in this layer.

3. Relu layer: Since Convolution involves in linear operations like multiplications, many aspects of the image could be missed due to this nature, and hence nonlinearity is added to the image to highlight the distinctive features of the images that might be otherwise missed if it were a linear operation. This is the place where Relu comes into picture. Relu is a function that adds non linearity to the system.

4. Pooling layer: This layer helps in reducing the size of the image by choosing only the aspects of the image that matter to the image detection. This greatly helps in reducing the processing time required to make an image prediction.

5. Fully Connected Layers: This layer accumulates all the features that are learnt in the previous stages and perform a

mathematical operation on these values to arrive at a prediction of what the image might be.

These are the fundamental layers involved in the development of Convolution Neural Network architecture. Multiple of these layers are used in succession to form different CNN architecture. The architecture that is used for the verification in this paper is called the AlexNet.

2. LITERATURE SURVEY

On reference of paper [1] I was able to obtain a fairly basic idea about the convolution neural networks, the different layers in the CNN and their importance. The author also mentions the available architecture in CNN like Alexnet Zefnet etc,. This came in handy to obtain further knowledge on the Convolution Neural Networks. The paper introduces the terms like dropout and gradient descent which are used to increase the computation time and learning rates respectively. This paper suggests Convolution neural networks with a recognition rates lesser than the already available architecture but they are a simpler architecture compared to the existing ones. This quality of the paper helped in obtaining a basic knowledge on the working of a CNN to kick-start the journey towards the motive of the project.

While the previous paper had given a basic idea of the components of a neural network, the reference of this paper [2] gave a fair knowledge on the exact uses of the different stages in the convolution Neural Networks. In this paper the terms pertaining neural networks such as were familiarized. Fairly detailed information of the computations involved in the networks was understood upon referring this paper.

SystemVerilog-based verification environment which consists of a layered testbench and SystemC design unit is described in the paper [7]. The multiple inheritances in OOP are useful for creating class types that combine the properties of two or more class types. More about the use of the system verilog language was grasped here which forms the basics of UVM.

In the paper [8] it is shown that The UVM methodology absorbed the advantages of OVM, VMM and other mainstream methodology. The UVM makes verification more efficient and maximum reusability. Based on the study of the UVM typical verification platform and UVM methodology to achieve reusable method, this paper builds a reusable verification platform of I2C.

3. PROPOSED WORK

3.1 The AlexNet Model

This model uses an input of 225X225 RGB image. Any image up for prediction needs to be brought to this format. This module has 5 convolution layers two fully connected layers and a softmax layer where the final prediction occurs.

The class library used for this model is called the caffe and it has 1000 classes of classified members under it. The alexnet model is trained to identify any of the classes from these thousand categories. The inference model identifies these classes.

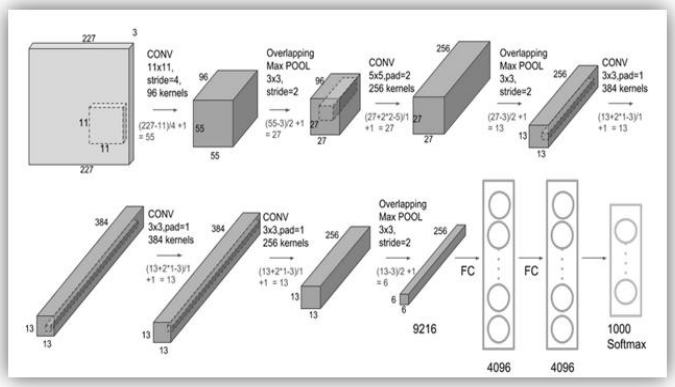

Fig -1: The AlexNet architecture

3.2 Universal Verification Methodology

The Universal Verification Methodology is a framework that is used for verification of System on chips. It is based off of the systemverilog language. UVM provides a standard framework to build a verification testbench so it would be easy to develop the testbench in the format that in universally used.

The UVM has components that have certain set of specific operations for which it is designed. They are:

1. The Sequence: It holds the input stimulus that needs to be sent to the verification of the Design under Test.

2. The Sequence item: It is the set of sequences that is sent to the Design under Test.

3. The Sequencer: It assists in sending the sequence item to the driver.4. The driver: It consists of the logic and time scheduling operations according to which the inputs are to be sent to the Design under Test.

5. The Monitor: It helps in monitoring the status of each component of the testbench. It is a passive component.

6. The Scoreboard: It compares the obtained results from the DuT with the expected results and asserts if the verification test is a pass or fail.

The UVM uses the method of Transaction Level Modeling for communication. In this method a group of information is bundled into transactions and is transmitted by the sender to the receiver. This eliminates the complexity of dealing

with pin level interfaces. This increases the level of abstraction.

3.3 Implementation Details

The following steps were followed in order to implement the verification of CNN using UVM:

1. A trained working model in any language(c/python) is obtained.

2. Obtaining the dataset used for this model. They are the weights, biases and kernels used for them.

3. Using these inputs to send to the Design under Test via the sequence, sequence item and the sequencer.

4. Obtaining the output from the DuT via the monitor and checking for its correctness in a scorebored.

5. A look up table comparison method is employed for the comparison of the output.

The Alexnet model uses Caffe classes as the dataset which means the model is trained to recognize the images belonging to a class in caffe dataset. The Dataset consists of a thousand classes; therefore the model can recognize any image that belongs to these categories. The images of these classes are used as a lookup table in the scoreboard for comparison. The output of the DUT is a number corresponding to the classification of images in the class.

The input output pin level details and specification of the Design under test is as follows:

1. Clk: Used as a reference of time for the transactions.

2. Rst: Used to reset the DUT to default stage.
3. Data_Valid: Used as an enable to signal to send the Image pixel Values. It is of 1 bit size.
4. Input_pixel_r: The image pixels of r channel are sent one pixel per clock cycle to the DUT via this port. It is of 8 bit size
5. Input_pixel_g: The image pixels of g channel are sent one pixel per clock cycle to the DUT via this port. It is of 8 bit size
6. Input_pixel_b: The image pixels of b channel are sent one pixel per clock cycle to the DUT via this port. It is of 8 bit size
7. Kernal_request: This acts as an enable pin to the Kernel values to be sent. The kernel is read by the DUT only when this signal is high.
8. Data_in: This is a control bit which is the input to the decoder. This bit is present to channel the kernel values to their respective convolution layers. This pin is of 3 bits size
9. Kernel_input_r: This is the port to send the kernel input values to DUT for R layer. This bit is of 8 bit size
10. Kernel_input_g: This is the port to send the kernel input values to DUT for G layer. This bit is of 8 bit size.
11. Kernel_input_b: This is the port to send the kernel input values to DUT for B layer. This bit is of 8 bit size

The sequence logic: Upon Rst the system is set to default with all the signals initialized to zero. The Data_valid and the Kernal_request signals are set to high to enable continuous streaming of input signals to the DUT (based on clock cycle). The input pixel signals (Input_pixel_r, Input_pixel_g and Input_pixel_b) and the kernal values (Kernel_input_r, Kernel_input_g and Kernel_input_b) are sent parallel.

The input kernels need to be sent to different convolution layers based on the readiness of the convolution stages. Same three channels are tapped to different layers. These layers are controlled by a multiplexer. For example, if the kernels need to be input to the first convolution layer, first the decoder input (Data_in) is set to 000. This disables all the layers except the first layer. Once the inputs to the first layer are exhausted, a certain time delay is inserted after which the Data_in input is changed to 001 and the set of inputs corresponding to the next layer is sent, which is received by the decoder enabled second layer.

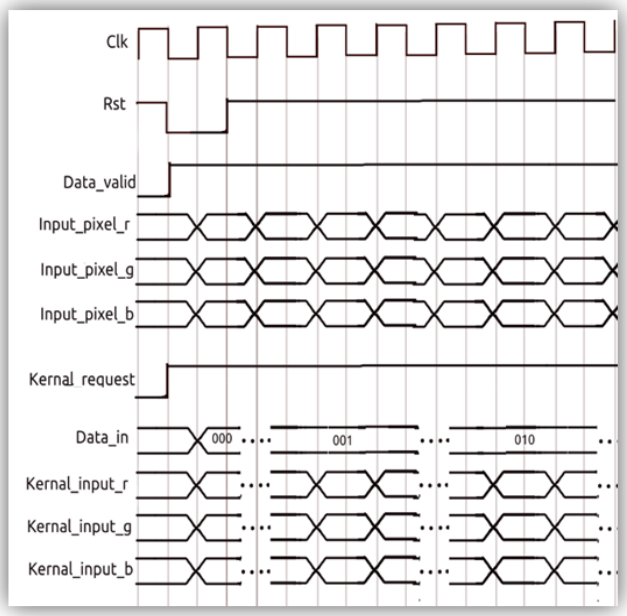

Fig -2: Input pin level specification

4. RESULTS AND SIMULATION

The sending of image pixels of R, G and B channels are seen in the Fig-3. It is seen that the data valid is high and the reset is low. Since the kernel input and the image pixel input uses the same channel for data input, the kernel_enable is set to 0 until the input of image pixels is done

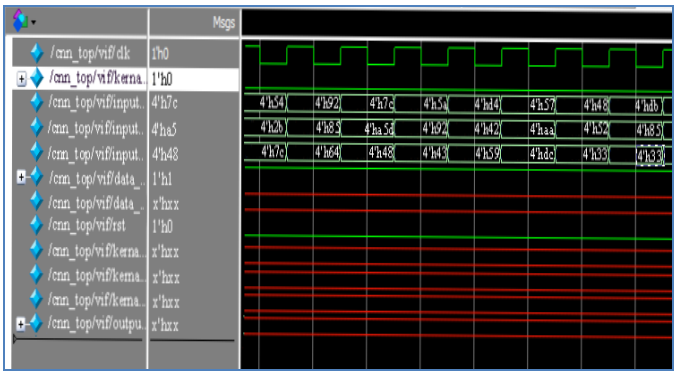

Fig -3: Waveform of sending image pixel

The Fig -4 shows the waveforms of the kernel data being sent to the RTL. Here the kernel_enable bit is set to high to signal the input of kernels. The Data_in value is the input to the multiplexer. The value of Data_in determines to which convolution layer the kernel values are sent to. Here in the figure, the kernel values are sent to the third convolution layer.

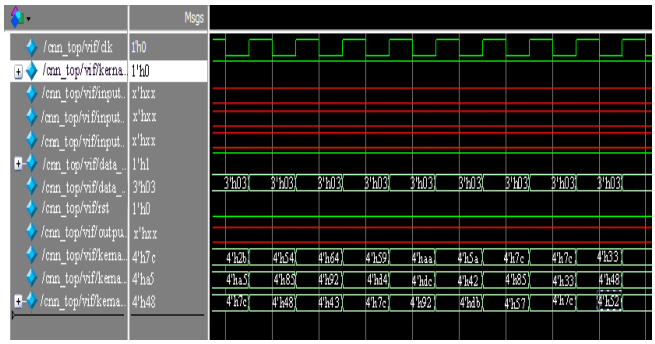

Fig -4: Waveform of sending kernels

In the Fig-5, the output of the image prediction is seen. The output is the number corresponding to the class in the caffe class library. In this case the output prediction is the fifth class of the library containing thousand classes which is the "hammerhead shark". All the bits in the interface are not determined at these points as all the inputs are done being sent to the DUT.

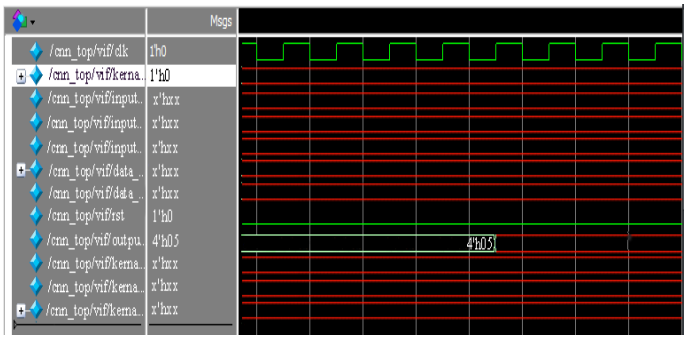

Fig -5: Waveform of final prediction

Fig -6 shows the UVM report summary. In the said code there are 102 INFO printed upon simulation, 2 warnings, 0 errors and 0 fatal issues in the testbench execution.

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  102
# UVM_WARNING :  2
# UVM_ERROR :  0
# UVM_FATAL :  0
# **Report counts by id
# [cnn_driver]   10
# [cnn_monitor]  12
# [cnn_sequence]   10
# [cnn_scoreboard]  7
# Simulation complete via $finish(1) at time 32145 NS + 55
```

Fig -6: UVM Report Summary

5. CONCLUSION

Upon undertaking the project, a practical experience of writing a UVM testbench code for ambitious design such as the Convolution Neural Network was obtained. The nature of the project process was based on research and improvisation. This project not only helped in developing an experience in the field of verification, it also helped in study and research of the state of art fields such as Artificial intelligence, Machine Learning, Deep Learning and Neural Networks.

ACKNOWLEDGEMENT

I am grateful for this opportunity to conduct this project/research under the guidance of Mrs. Ashwini V, Assistant Professor, and Department of ECE. I am also thankful for the college, BMSCE, for providing me this opportunity and support for pursuing this work.

REFERENCES

[1] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(, Beijing, 2017, pp. 721-724.

[2] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6.

[3] H. Shin et al., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1285-1298, May 2016.

[4] Jing Sun, Xibiao Cai, Fuming Sun and J. Zhang, "Scene image classification method based on Alex-Net model,"

2016 3rd International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS), Jinzhou, 2016, pp. 363-367.

[5] Solovyev, Roman A., Alexandr A. Kalinin, Alexander G. Kustov, Dmitry V. Telpukhov and Vladimir S. Ruhlov. “FPGA Implementation of Convolutional Neural Networks with Fixed-Point Calculations.” ArXiv abs/1808.09945 (2018): n. pag.

[6] X. Chen, J. Ji, S. Mei, Y. Zhang, M. Han and Q. Du, "FPGA Based Implementation of Convolutional Neural Network for Hyperspectral Classification," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, 2018, pp. 2451-2454.

[7] M. You and G. Song, "SystemVerilog-based verification environment using SystemC custom hierarchical channel," *2009 IEEE 8th International Conference on ASIC*, Changsha, Hunan, 2009, pp. 1310-1313.

[8] W. Ni and J. Zhang, "Research of reusability based on UVM verification," *2015 IEEE 11th International Conference on ASIC (ASICON)*, Chengdu, 2015, pp. 1-4..

[9] S. Zhao, K. Zhang, J. Fan and H. He, "A Software-Hardware collaboration system for CNN algorithms based on FPGA," 2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), Xi'an, China, 2019, pp. 1-2.

[10] LecunY, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[11] Hu, M.L.a.X., Recurrent Convolutionalal neural network for object recognition. 2015.

[12] N. Jmour, S. Zayen and A. Abdelkrim, "Convolutional neural networks for image classification," *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, Hammamet, 2018, pp. 397-402.

[13] Redmon J, and Angelova A, “Real-time grasp detection using convolutional neural networks”, IEEE International Conference on Robotics and Automation, pp. 1316–1322, 2015.

[14] Howard, A., “Some improvements on deep convolutional neural network based image classi¿cation.” ICLR, 2014