Enhanced Performance Verification

Naveen Kumar Bangalore Ramaiah and Naresh Khatokar
Department of Electrical Engineering, San Jose State University, San Jose, California 95192

Introduction

Most critical part of hardware design cycle is verification. Studies have shown that verification almost consumes 53% of total effort spent for developing a hardware from its initial stage. [1]

Since verification phase of a project is so critical, it is really helpful if one can reduce verification effort and time.

A verification block developed by taking throughput (frame/second) as performance metric for JPEG encoder. JPEG Encoder has multiple encoding phases which has lot of scope for performance tuning. The throughput of JPEG encoder varies with input pixel values. This is determined by building a UVM verification environment to measure throughput of JPEG Encoder.

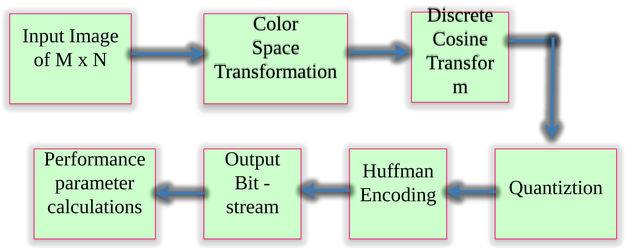
Overview Of JPEG Encoder

JPEG is a lossy image compression method. It uses a transform coding method using the “Discrete Cosine Transform” - DCT.

An Image can be represented as a function of i and j (which can be more generally referred as X and Y direction) in the spatial Domain.

The 2D DCT is used to generate a frequency response in just one step, which is a function F (u,v). It is generally indexed by two integers - u and v [5].

Modeling



Input image is divided into 8*8 blocks and given to following blocks as input

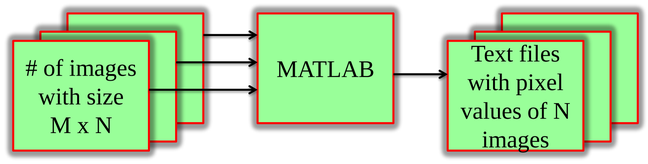
- Color Space Transform : RGB is converted to Y, Cr, and Cb [3]
- DCT : Perform Discrete Cosine transform
- Quantizer : Reduce the amount of information by dividing each component in the frequency domain by constant and then rounding to the nearest integer.
- Huffman Encoding : Perform encoding operation using Huffman encoding techniques[4].

Design Approach

Key Points

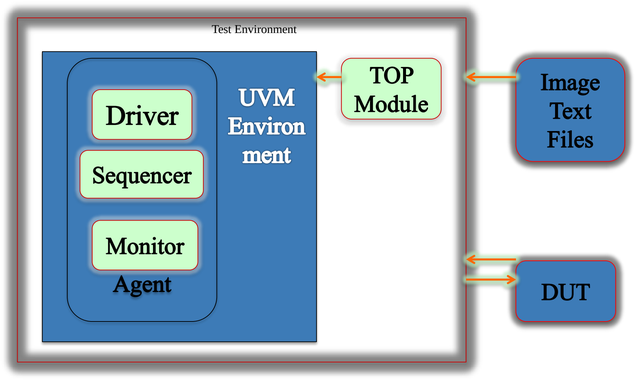
- UVM
- JPEG encoder
- Performance Verification

Image to Text Conversion using Matlab

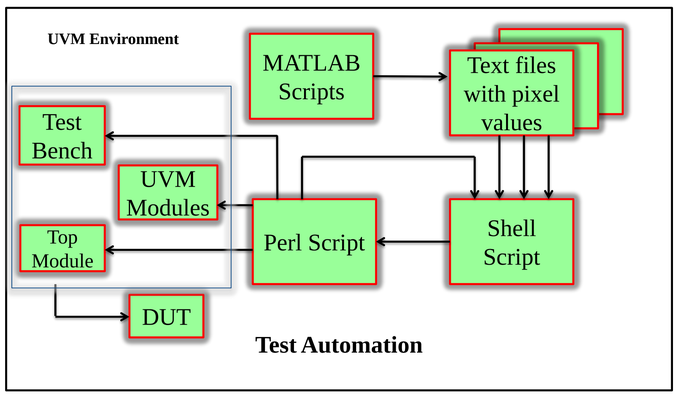


N Images will be given to MATLAB Script which will convert pixel values into binary format and store them into text file.

UVM Environment



- Agent: It will initiates and connect Monitor, Driver and Sequencer together.
- Driver: Takes the sequences and Drives DUT.
- Sequencer: Prepares the image pixel sequence, which needs to given to DUT for pixel Encoding.
- Monitors: It monitors Output_bit_stream and Data_ready signals and stores the time whenever it changes [2].

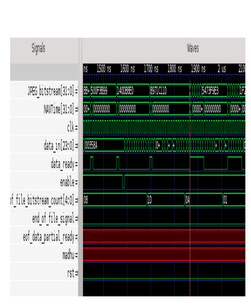


- Perl Scripts generate TOP module, interface and UVM components to start the verification process.
- Top module invokes DUT and UVM components and initiate the performance verification.
- At the end of verification mean and Std. Deviation will be calculated from the values stored I monitors[2].

Results

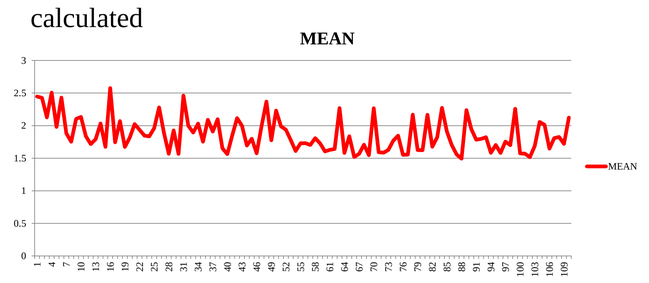
VCS Simulation

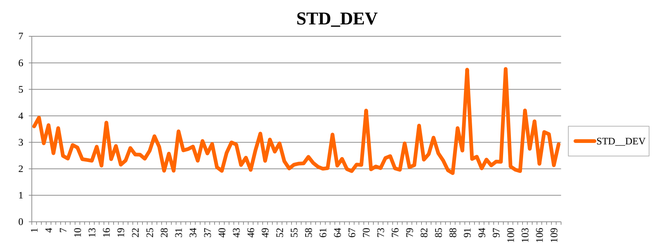
In the above we can see VCS simulation result of DUT with UVM environment. IN the above figure, dataIn which are passed to DUT through driver and bitstream which are calculated and it will be transferred to top module.



Statistical Analysis

Performance of JPEG encoder is verified using, VCS simulation results. By using data_ready signals mean and standard Deviation of latency will be calculated





Verification is done for more than 100 images and few images samples are shown below.

Conclusions

We have presented a enhanced performance verification technique to verify JPEG Encoder using time constraints. Performance of the device under test –JPEG Encoder was determined in terms of latency. More than one hundred different images were tested to determine performance. The plot of Mean and Standard Deviation of latency was plotted, by using these results we can decide whether performance tuning can be done or not.

Key References

[1] *www.blogs.mentor.com*

[2] *UniversalVerificationMethodology(UVM)1.1User’sGuide.*

[3] *MathWorks http://www.mathworks.com/help/images/understanding-color-spaces-and-color-space-conversion.html*

[4] *JPEG Encoder IP Core, Daniel klun opencores.com*

[5] *Image and Video Compression Standards: Algorithms and Architec- tures, 2nd ed. by Bhaskaren and Konstantinides*

Acknowledgments

We would like to thank Prof. Morris Jones for his guidance which helped us to complete the project.

For further information

Please contact *brnaveen.sjsu@gmail.com* or *nareshkhatokar@gmail.com*.

MATLAB code, simulation files, and Perl scripts files are available upon request.