ASIC Implementation of Neural Network Based Image Compression

K.Venkata Ramanaiah and Cyril Prasanna Raj

***Abstract*—Image data consumes enormous bandwidth and storage space. Neural networks can be used for image compression. Neural network architectures have proven to be more reliable, robust, and programmable and offer better performance when compared with classical techniques. In this paper the main focus is development of new architectures for neural network based image compression optimizing area, power and speed as specific to ASIC implementation, and comparison with FPGA.**

The proposed architecture designs are realized on Spartan IIIE FPGA Using Xilinx ISE, and the ASIC implementation is carried out using Synopsys tools targeting 130nm TSMC Library. The ASIC implementation for 2 input and 16 input neuron with low power techniques adopted such as buffer insertion, clock gating etc,

***Index Terms*—Image compression, neural networks, FPGA, ASIC, CSD**

I. Introduction

The transport of images across communication paths is an expensive process. Image compression provides an option for reducing the number of bits in transmission. This in turn helps increase the volume of data transferred in a space of time, along with reducing the cost required. It has become increasingly important to most computer networks, as the volume of data traffic has begun to exceed their capacity for transmission. Traditional techniques that have already been identified for data compression include: Predictive Coding, Transform coding and Vector Quantization [1, 2]. In brief, predictive coding refers to the decor relation of similar neighboring pixels within an image to remove redundancy. Following the removal of redundant data, a more compressed image or signal may be transmitted [1]. Transform-based compression techniques have also been commonly employed. These techniques execute transformations on images to produce a set of coefficients. A subset of coefficients is chosen that allows good data representation (minimum distortion) while maintaining an adequate amount of compression for transmission. The results achieved with a transform based technique is highly dependent on the choice of transformation used (cosine, wavelet, Karhunen-Loeve etc.)[2]. Finally vector quantization techniques require the development of an appropriate codebook to compress data. Usages of codebooks do not guarantee convergence and hence do not necessarily deliver infallible decoding accuracy. Also the process may be very slow for large codebooks as the process requires extensive searches through the entire codebook [1].

Artificial Neural Networks (ANNs) have been applied to many problems [3], and have demonstrated their superiority over traditional methods when dealing with noisy or incomplete data. One such application is for image compression. Neural Networks seem to be well suited to this particular function, as they have the ability to preprocess input patterns to produce simpler patterns with fewer components [1]. This compressed information (stored in a hidden layer) preserves the full information obtained from the external environment. Not only can ANN based techniques provide sufficient compression rates of the data in question, but security is easily maintained. This occurs because the compressed data that is sent along a communication line is encoded and does not resemble its original form.

The basic architecture for image compression using neural network is shown in figure1.

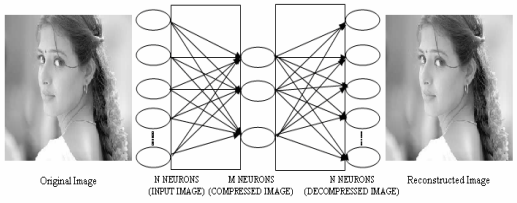


Fig. 1. The basic architecture for image compression

II. FPGA Implementation of 16 & 64 Input Neural Network Architecture

The neural network architecture proposed in this paper is consisting of 64 and 16 input neuron, are modeled using HDL. The network supporting numbers in the range 0 to 1 is taken care by introducing BCSD multipliers for weight multiplication [6]. The HDL code for the proposed network is verified for its functionality using test bench, the design is synthesized on FPGA to estimate the hardware complexity for efficient ASIC implementation. The design is mapped on Spartan III device from Xilinx. The synthesis results are shown in table 1.

TABLE 1: Synthesis Results

Implementation details	16	64
Maximum operating frequency	220 MHZ	242.97MHz
Maximum power at 25 degree Centigrade	81mW	81.37mW
Resource utilization		
Number of Slices out of 4656	25	126
Number of 4 input LUTs out of 9312	39	226
Number of global clock	1	1

Manuscript received June 16, 2010; revised June 20, 2011.

K.Venkata Ramanaiah, Principal, Narayana Engineering College Gudur, Andhra Pradesh, India (ramanaiahkota@gmail.com).

C. P. Raj, Course Manger M.S. Ramaiah School of Advanced Studies Bangalore, Karnataka, India

As the design is mapped on FPGA, it supports Reconfigurability. Reconfigurability can be achieved by changing the weight matrix and the input layer for better compression [8].

The design proposed consists of matrix multiplication of two matrices, one is the input image samples, and the second is the weight matrix obtained after training. This multiplied output is passed through the nonlinear transfer function to obtain the compressed output that gets transmitted or stored in compressed format. On the decompression side, the compressed data in matrix form is multiplied with the weight matrix to get back the original image. The image quality of the decompressed image depends on the weight matrix. The image data of size 16x16 is multiplied by the weight matrix of 4x16 to get a compressed output of 4x16. On the decompress or side 4x16 input matrix (compressed image) is multiplied with the weight matrix of size 16x4 reproduces the original image. In order to achieve better compression nonlinear functions are used both at the transmitter and receiver section. The HDL code modeled for FPGA implementation is modified for ASIC implementation. The general coding styles are adopted for building optimized RTL code for ASIC implementation. The results obtained from ASIC synthesis to physical implementation are compared with

FPGA implementation. The architecture synthesized is optimized for power area and speed.

The design is synthesized using TSMC 130 nanometer technology and library files. The design is synthesized using Synopsys DC, the timing analysis is carried out using Prime Time. The proposed neural network architecture is implemented on FPGA as well as ASIC. The HDL model developed for the entire network supporting 64-4-64 and 16-4-16 is verified for its functionality, the input image pixels of size 64 x 1 represented in integer form is stored in the RAM and is fed into the network for processing. Simulation results for 16 inputs Neuron is shown in Fig. 2 and 3. Matrix [A] represents the pixel values of the image subset considered. Matrix [B] represents the weight matrix obtained after training. Matrix [C] represents the compressed image.

MATLAB results for 16 input neuron.

Image Matrix A =

```
[ 102  102  105   94  103   99  101  111   98  100   93  106  102  102   87   97;
   90  104   93  106   96   87  108  110  106  102   94  100  101  111   87  123;
   98  103   96   95  102   98  115   89  114  100   91   91  104  109   99   92;
  102  106   89  106   93   98   98  109   97   94   99   90  101   95   96   91;
  101  106  118  145  149  130  103  106   96  104   97   96  102   89   99   92;
  157  164  172  168  168  173  170  156  125  114  102  101  101   97  109  102;
  171  170  168  168  174  171  172  178  179  165  124  108  101  114   92  112;
  179  172  171  167  169  176  174  174  180  186  189  164  115  125   94  106;
  186  182  173  177  173  174  180  180  182  188  193  195  185  139   98  103;
  172  176  182  176  175  178  174  176  184  191  195  196  194  190  164  113;
  180  179  176  179  180  179  181  183  181  185  193  196  194  194  190  180;
  145  178  180  178  178  183  185  187  190  189  190  193  195  195  193  187;
  103  115  164  181  179  178  183  188  192  192  195  195  192  191  191  187;
  103  100  109  128  167  183  184  183  187  194  194  196  195  189  184  182;
  101   96  109  105  111  130  170  186  183  187  190  196  196  195  189  173;
  101  104  102  101  101   99  102  135  173  183  184  189  193  192  188  186;]
```

Weight Matrix B =

Columns 1 through 8

```
  0.6670   0.4805   0.1365  -0.1067   0.4565   0.3721   0.2078   0.1189
  0.4004   0.2488  -0.0576  -0.3003  -0.0237  -0.0930  -0.1882  -0.2195
  0.9998   0.9998   0.6091   0.2117   0.6836   0.6758   0.5840   0.5327
  0.0247   0.0989   0.1956   0.2620   0.0022   0.0105   0.0342   0.0547
```

Columns 9 through 16

```
  0.1294   0.1965   0.3000   0.3682  -0.1443   0.0642   0.3870   0.6064
 -0.5791  -0.5000  -0.2754   0.1350  -1.0000  -0.8149  -0.3364  -0.0198
 -0.1431   0.1287   0.5894   0.8423  -0.8484  -0.2983   0.5886   0.9998
 -0.0144  -0.0779  -0.0779  -0.1995  -0.0537  -0.1704  -0.3269  -0.4285
```

C = output matrix from the neuron.

Columns 1 through 8

```
  532.4542   553.3630   556.0333   559.4684   571.0199   565.9154   584.1573   611.6194
 -508.1305  -508.0672  -561.8866  -599.0243  -625.3295  -650.9471  -665.8167  -674.5208
  909.6719   950.4613   909.8816   903.7885   906.2319   893.3486   929.2258   965.3972
  -83.5810   -87.0468  -101.9572   -98.7255  -109.6476  -119.0124   -28.8570  -150.7306
```

Columns 9 through 16

```
  614.4376   620.6669   600.1038   613.4345   608.5498   603.1723   563.5169   565.3058
 -688.1480  -699.9430  -708.3724  -696.1829  -678.3096  -643.3851  -601.7479  -556.8632
  970.8134   964.5284   915.4445   920.5861   910.2671   930.4870   863.6388   887.2100
 -166.9798  -178.6281  -184.1987  -192.9532  -191.7607  -187.6392  -186.5236  -171.6240
```

Results obtained from the Modelsim simulation for the 1st row of the neuron it is shown in Fig. 2, these results exactly matches with the first column in C matrix obtained from MATLAB.

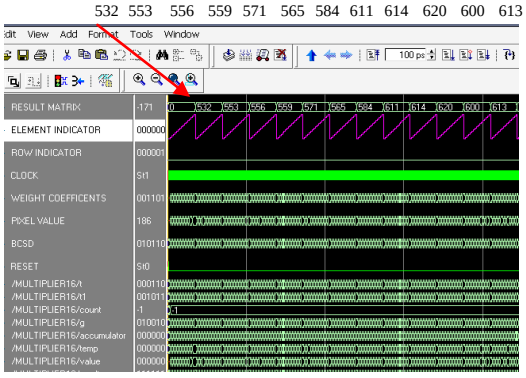
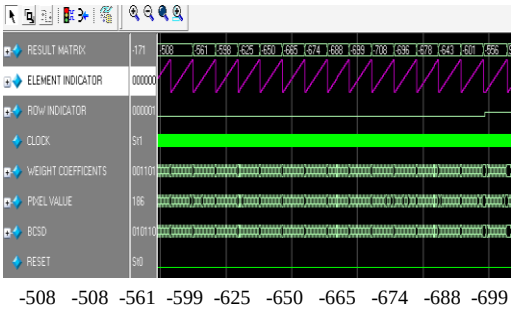
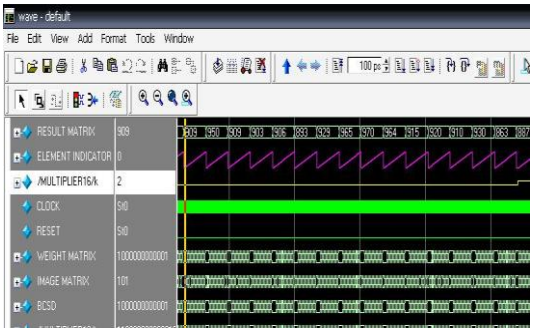


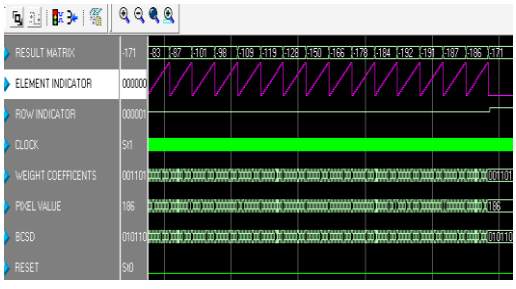
Fig. 2. Simulation results for 16 input neuron.



-508 -508 -561 -599 -625 -650 -665 -674 -688 -699



950 909 903 906 893 929 965 970 964 915 920 910



-83 -87 -101 -98 -109 -119 -128 150 -166 -178 -184

Fig. 3. Simulation results for 16 input

Fig. 3 highlights the remaining rows after compression. These results are matching with Mat lab results. As the complexity in the RTL code increases the area should increase. However, due to the efficient coding style adopted

optimizes the total hardware required. In the RTL model, weight matrix obtained is optimized where ever required by finding redundancy. This drastically reduces the hardware complexities. It is observed that by setting proper constraints, the macros available on FPGAs are forcibly used to exploit the architecture resources. This saves the hardware utilization.

III. ASICS IMPLEMENTATION RESULTS FOR 16-INPUT NEURON

The design supporting higher order multiplication modeled using HDL, developed to an RTL model using efficient coding styles for ASIC Synthesis using Synopsys Design Compiler Version 2007.03. The timing analysis is carried out using Synopsys Prime Time. The optimization techniques are:

1) Resource Sharing
2) If Statements Replaced Using Case Statements
3) Clock Gating and power gating
4) Setting Maximum Transition
5) Setting Max Capacitance
6) Setting Hold Time And Setup Time
7) Setting Minimum Path Delay
8) Gate Sizing
9) Inserting Clock buffers and super buffers
10) Weight Optimization

The above techniques are set using the tool options and changing the coding styles. Optimization 1 refers to the results obtained with initial settings; these results reflect the performance of the design without any optimization techniques incorporated. Optimization 2 refers to use of inbuilt constraints available on the tool Design Compiler (DC). This involves setting up of constraints on clock, area, power, time arrivals, load capacitance, gate sizing, clock buffers and insertion buffers. Optimization 3 refers to modifying coding styles by insertion of clock gating, power gating, resource sharing and memory sharing techniques. A graph shown in Fig. 5 to 8 discusses the variation in performance from the initial design to the final design based on optimization techniques mentioned. Optimization 4, 5, 6, refers to the techniques adopted by the tool based on constrains set during the flow. Fig. 4 shows the synthesized RTL net list obtained using Synopsys design compiler. The green cells represents the logic gates from TSMC 130nm technology library, blue lines represents interconnects and the red line represents the interconnect that takes the maximum delay (critical path).

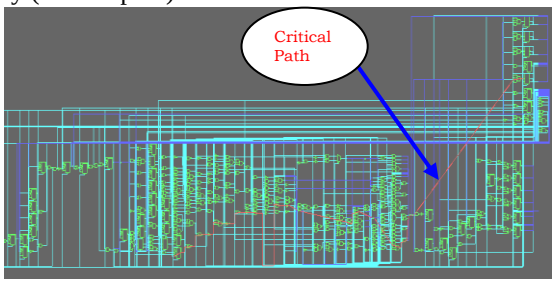


Fig. 4. synthesized net list highlighting the critical path

Fig. 5 represents the timing report for the sixteen input neuron. Timing report is generated using prime time from Synopsys for different optimization techniques. The parameters are expressed in terms of slack.

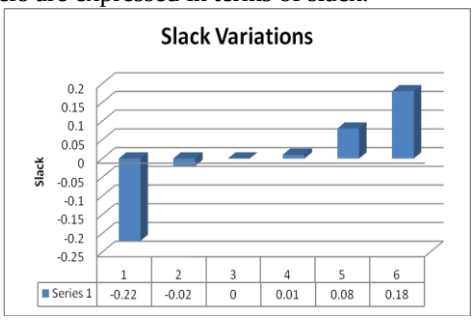


Fig. 5 Slack variations for 16 input neuron.

Slack represents whether the data is available at the right time to be latched on to next stage. Slack should always be positive. Positive slack implies that the design meets the timing requirements. Without optimization the design is verified for the slack. From Fig. 5 it is found that the slack that was negative has been converted to positive slack of 0.18. At slack 0.0 the design is just ideal, with optimization techniques, by setting constraints on clock the slack is made positive. The results are obtained using Design Compiler, a signoff tool from Synopsys. It is recommended that a slack of 0.08 is optimum.

In order to achieve positive slack, the buffers get introduced on the critical path; even the gate sizes are increased to achieve higher currents to drive the load. This drastically increases the cell area. However with change in coding style the percentage increase in cell area is minimized by adopting coding guidelines. The number of nets also increases as the layers are increased to more than three. Fig. 6 shows that in order to achieve positive slack, the cell count is increased to 486 from 353. It is also observed that the number of nets has been increased to 581 from 440. The cell count and nets reaches saturation after certain limits. This demonstrates the idealities of the synthesis tool.

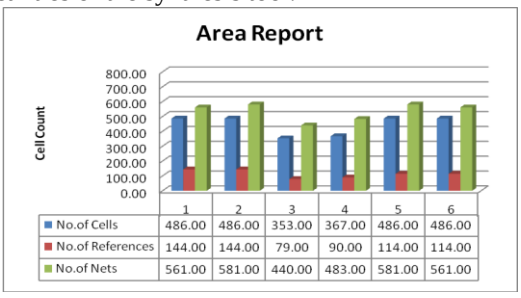


Fig. 6. Area report for16 input neuron.

In the Fig. 7 it is observed that the total cell area has increased to 8662 Sq µm from 6319 Sq µm, this is done in order to achieve low power and higher speeds.

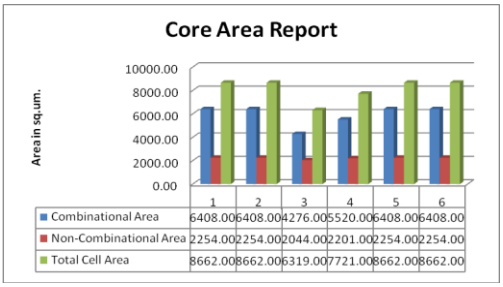
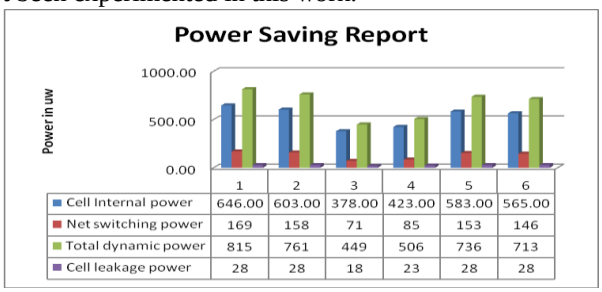


Fig. 7. Core area report for 16 input neuron

The Fig. 8 demonstrates the variation of power with different optimization techniques considered in this work and

labeled as 1, 2, 3, 4, 5 and 6. The dynamic power varies from 449 µW to 815 µW. It is found frequency variation affects power. Increase in power is brought into control by constraining the design during synthesis by adopting suitable power saving techniques. Hence, we find that during the optimization the gradual increase in power is limited to 713 µW which is better than the previous results as shown in Figure. 8. This is achieved by using the power saving techniques. However we find that the leakage power has increased from 18 µW to 28 µW, this is due to the fact that the power saving techniques incorporated such as clock gating, power gating concepts adopted introduces additional cells that enables the required logic only when required, this keeps most of the logic in standby mode by disabling them from the clock network and power network being connected. Hence there is increase in leakage power. In order to reduce the leakage power the High Vt Library cells can be used. This has not been experimented in this work.



	1	2	3	4	5	6
Cell Internal power	646.00	603.00	378.00	423.00	583.00	565.00
Net switching power	169	158	71	85	153	146
Total dynamic power	815	761	449	506	736	713
Cell leakage power	28	28	18	23	28	28

Fig. 8. Power saving report for16 input neuron

IV. The Physical Design Flow

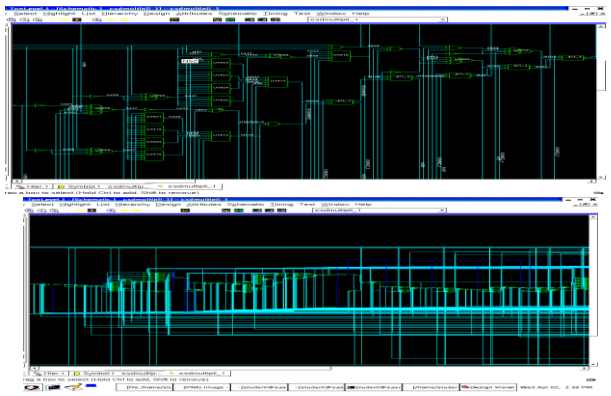


Fig. 9. Schematic gate level net list.

The synthesized net list is taken through the VLSI Physical Design Flow to generate the GDSII. In this flow the design is taken through Physical design flow steps such as floor planning, placement, clock tree synthesis, routing, physical verification, parasitic extraction and finally timing verification to sign off the design. This ensures the pre-silicon verification is carried out using industry standard sign off tools and the design is taped off for fabrication, with generation of GDSII format. In this design Synopsys flow is used for physical design and verification, which is one of the standard signoff tools. The Figure. 9 shows the synthesis results of 16 input neuron RTL code synthesized using Synopsys DC, targeting TSMC130nm library.

This design has to be taken through the Physical design flow. Fig. 9 shows the gate level net list, but does not give details of the cell placement, power network, clock network and pin details.

The design which models 16 input neuron developed in this work is considered as major macro for image compression using neural network, this module is followed by quantization, data encoding and storage modules. Hence the design is converted into a macro that can be interfaced with other building blocks. The pin configuration and the sizing of this macro decide the cost function and its performance.

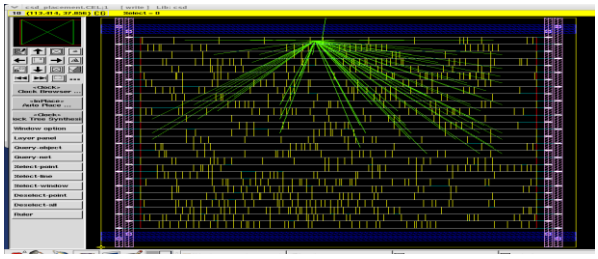


Fig. 10. Floor planned and Clock Tree Structure.

The design is taken through the floor planning, placement and clock tree synthesis stage, in this stage the cell area, I/O area, number of I/O, I/O properties, power network, hot spots, congestion due to interconnects, placement of standard cells, clock tree routing for the cells and total cell area gets identified. This is carried out using floor planning methodology supported by ASTRO tool from Synopsys, the results of which are shown in Fig. 10. The thick dark lines on the perimeter running horizontally and vertically show the power network VDD and VSS, the yellow blocks shows the core area with placed standard cells, the green lines shows the clock distribution required for the design.

The design consumes 500 cells, 2903 pins, 539 clock nets, 30 I/O pins with core area of 9945.07Sq µm (119.82 µm x 99.63 µm). The total chip size with I/O pins is 14334.07 Sq µm (119.82 µm x 119.63 µm). The cell/corn ratio is constrained to 74.5861% and cell/chip ratio is constrained to 51.7483%.

The design is taken through routing phase and the Fig. 11 shows the placed and routed design. As the design has 539 nets to be routed, metal 1 to metal 8 layers are used in order to route the design, with metal layer 6 consisting maximum number of wire length of 6209.5 µm. The total wire length is 12592.2 µm. The design is constrained to minimize congestion. The routed design is shown in Fig. 11 the vertical and horizontal pink line shows the wires connecting the standard cells within the core area. Both local and global routing is performed; the final routing directions are what are highlighted in Fig. 11 which are pink in color.

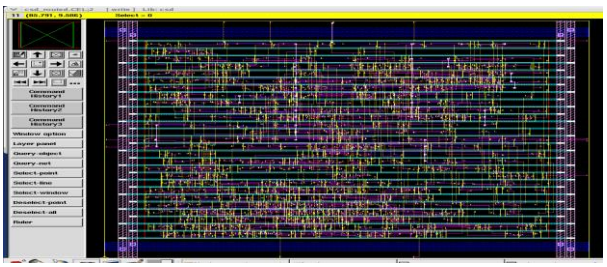


Fig. 11. Place and Route Design.

Fig. 12 shows the placed and routed design with IR analysis.

The color combination highlighted in Fig. 12 shows the IR drop distribution within the die. Red color is the violated and more power consuming region. Green color is the region where there is nominal power consumption. As we see that there are very few areas with high IR drop, this is taken as warning and is neglected. The cell power when compared with the power report obtained during synthesis phase has 97% improvement. This shows that the physical design improves the power performance, as this considers the actual cell placement and pin placement with clock and nets routed.

The routed design is taken through IR drop analysis to find out the impact of power dissipation due to parasitic extracted from the nets, and the cells. The parasitic report suggests that the design has 803 internal nodes, 8 boundary nodes, 881 resistors and 500 current sources. This data enables to find the power dissipation. It is found that the total power is 0.0202574mW and the I/O net switching power is 0.000152925mw.

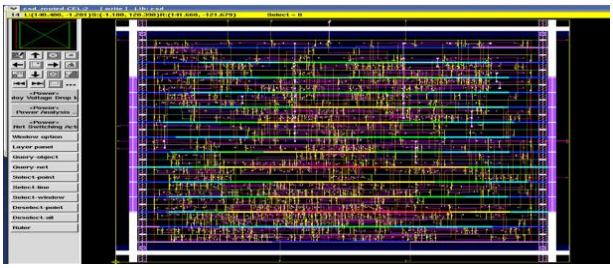


Fig. 12. IR drop analyses for 16 input neuron.

The Fig. 13 shows the comparison of the hardware implementation details focusing the major parameters like area and power.

The result clearly shows the performance metrics of each implementation. FPGA occupying more space and power, the only advantage is Reconfigurability and time to market. ASIC results are found to be better than the FPGA results in terms of area and power. They consume less power and space, hence suitable for low cost and reliable Hardware implementation.

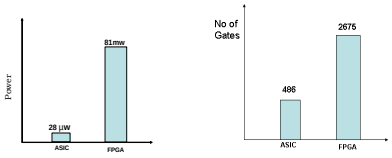


Fig. 13 Comparison of ASIC with FPGA in terms of Area &Power.

V. Conclusion

ASIC implementation of neural network architecture for image compression has been successfully implemented implemented using 130nm technology. Input image is compressed and decompressed using the two layered neural network architecture. The network trained using backpropagation algorithm is realized using multipliers and adders. The trainied weights are stored in memory and is used to compress and decompress image. Low power tehcniques have been used to reduce power dissipation of the complex architecture. Power reduction can be further achieved by replacing multipliers and adders using low power arithmetic units.

The techniques proposed in this work are modeled, designed and validated as per the Hardware requirements for ASIC and FPGA implementation. Suitable techniques have been incorporated to optimize area, power and speed.

References

[1] Dony, R.D., and Haykin, S., Neural Network Approaches to Image Compression, Proceedings of the IEEE, (1995), Vol.23, No.2, pp 289–303.

[2] Namphol, A. et al., Image Compression with a Hierarchical Neural Network, IEEE Transactions on Aerospace and Electronic Systems,(1996), Vol.32, No.1, pp.327–337.

[3] Blumenstein, M., The Recognition of Printed and Handwritten Postal Address using Artificial Neural Networks, (1996), Dissertation, Griffith University, Australia.

[4] Jiang, J., A Neural Network Design for Image Compression and Indexing. International Conference on Artificial Intelligent Expert Systems and Neural Networks, (1996), Hawaii, USA, pp 296–299.

[5] J.Robinson and V. Kecman, "Combining Support Vector Machine Learning With the Discrete Cosine Transform in Image Compression," IEEE Transactions on Neural Networks, Vol. 14, No. 4,IEEE, July 2003, pp.950-958.

[6] Daniele Lo Iacono and Marco Ronchi.,Binary "Canonic Signed Digit Multiplier for High-speed Digital Signal Processing." The 47th IEEE International Midwest Symposium on Circuits and Systems. 0-7803-8346-X/04/$20.00 02004 IEEE I1 -205 to I1 -208.

[7] Ivan Vilvovic, "An experience in Image compression using neural networks', 48th International Symposium ELMAR-2006, June 2006, Zadar, Croatia.

[8] K.Venkata Ramanaiah, Dr K.Lal Kishore, and Dr P.Gopal Reddy "Power Efficient Multilayer Neural Network for Image Compression" Information Technology journal 6(8): 1252–1257 ISSN1812-5638.@ 2007 Asian Network for Scientific Information

[9] K.Venkata Ramanaiah, Dr K.Lal Kishore, and Dr P.Gopal Reddy "New Architecture for NN based Image Compression for optimized Power, Area and Speed". i-managere's Journal on Electrical Engineering. Vol. 1. No. 3. ISSN-0973-8835. Jan-March -2008.