FPGA Implementation of 4×4 Luminance Intra Prediction

Mahesh Jayakar[1],
[1]M.Tech(Electronics) Student,
BMSCE, Bangalore, India.

Ashwini.V[2]
[2]Assistant Professor, E&C Dept.,BMSCE.
BMSCE, Bangalore, India.

***Abstract*- This paper proposes an efficient, fast and parallel processing of 4×4 luminance intra prediction implemented on FPGA. H.264[1] advanced video coding is a present generation video compression algorithm which can achieve high compression ratio without degrading the video quality. To achieve high compression H.264 uses various algorithms. Intra prediction [2,3] is one such algorithm to exploit the spatial redundancy in video frames. As we know there is a high correlation between pixels in a video frame. Intra prediction eliminates the correlation between pixels in a same frame to achieve compression. In this paper, luminance part of the image or video frame is taken and intra prediction algorithm is applied. In H.264[1] the luminance part of a video frame is divided into 4×4 or 16×16 block size depending on user application on image quality. Here we use 4×4 block size intra prediction. This has nine modes [1,2,3] to predict an image using reconstructed neighbor pixels of adjacent blocks. Processing of nine modes of 4×4 luminance intra prediction requires huge computations and has high latency. So we propose a method to reduce computations and processing all modes in parallel to achieve lower latency. This project is designed in Verilog using Xilinx ISE, simulated using Xilinx ISIM, synthesized and implemented on Virtex-6(Device: xc6vcx130T, Package: ff484, Speed:-2) FPGA. Project is implemented on FPGA's because present generation FPGA's are very much faster and has more resources and design time and time to market is less compared to ASIC's. The design is synthesized using Xilinx XST and power consumption is calculated using XPower Analyzer. Results achieved is analyzed and compared to check the design works perfectly. This project is very useful for video application which needs faster processing and can be used as intra prediction IP (Intellectual Property).**

Keywords:H.264, Intra Predictions, mode, luminance.

I. INTRODUCTION

Video related applications have become a necessary part of our day to day life. These demands on application have resulted in various researches in video compression. These researches led to the development of H.264 advanced video coding (AVC). H.264[1] was developed by ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1Moving Picture Experts Group (MPEG). The project partnership effort is known as the Joint Video Team (JVT). H.264/AVC block diagram is shown in Fig. 1. In H.264 video frame is converted into YCbCr format. H.264 has two predictions known as intra and inter-prediction. Intra prediction is used to eliminate spatial redundancy, i.e to eliminate correlation between pixels in a single frame. Inter prediction is used to eliminate temporal redundancy, i.e. to eliminate redundancy between neighbouring frames. In intra prediction luminance part Y is divided into macroblock of size 16×16 and processed by subdividing into sixteen 4×4 subblocks or processed as one16×16 block.

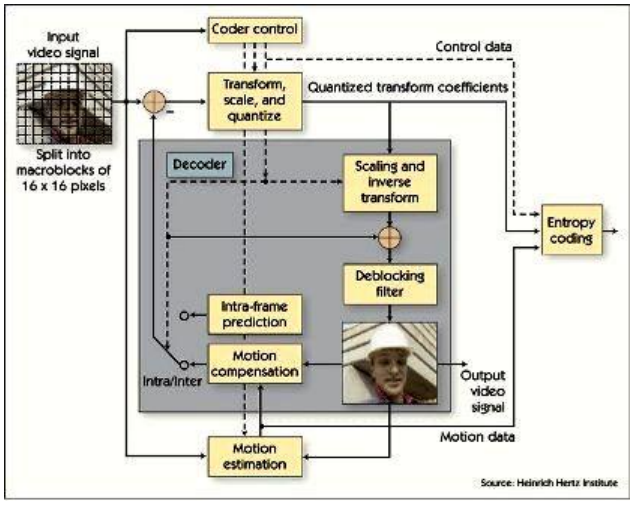

Fig.1. Basic Coding Structure for H.264/AVC.

Chrominance[1,2] part Cb and Cr is divided into fixed size of 8×8 macroblock and processed separately. Various intra prediction modes are used to predict pixels for current block to eliminate spatial redundancy. Intra prediction modes are explained in later sections. In inter prediction [1], motion estimation and motion compensation[1,2] is used to match the block for current frame in previous or future frames. The predicted frame block is subtracted by original frame block

and transformed using discrete cosine transform[1] (DCT). After DCT, it is quantized and entropy coded to compress the video. In encoder the quantized block, is inverse quantized, inverse transformed and reconstructed block is stored in picture buffer so that it can be used for intra prediction by using reconstructed neighbouring pixels and in inter prediction the reconstructed frame or block is used to match block from previous reconstructed frame for current frame to eliminate temporal redundancy. Deblocking filter is used to reduce blocking artifacts created during intra or inter prediction.

In H.264 prediction process is very complex and time consuming. So there is a need to develop an improvement in prediction process. So in this project luminance 4×4 block size intra prediction part of H.264/AVC[1] is taken for development. 4×4 block size produces good image reconstruction but take more time for processing. So a parallel and efficient approach to predict an image is developed. There are many research carried out for this problem. This project is implemented on FPGA[3] due to its ease of implementation. And present generation FPGA's are faster and has more area, so that complex design can be implemented on FPGA's. Advantages of FPGA's are also one of the factors to implement design on FPGA. FPGA's design and test time is less and time to market is less. So many researchers and industry now a day's prefer FPGA's rather than ASIC's. The rest of the paper discusses the development and design of efficient approach of 4×4 luminance intra prediction.

The rest of the paper is organized as follows. In Section II, H.264 intra prediction is reviewed. The proposed design is discussed in section III. Section IV discusses about simulation and synthesis results of our design. In the end conclusion and references used are highlighted.

II. 4×4 LUMINANCE INTRA PREDICTION

Intra prediction[1, 2, 3] algorithm is a process to eliminate correlation between neighbouring pixels in a single frame of a video sequence. Neighbouring pixels of reconstructed block in current frame is used for prediction of current block. Luminance component[2] of video is more important than chrominance part because luminance is more sensitive to human visual system (HVS-eye). In 4×4 block size luminance intra prediction a frame is divided into macroblock of 16×16 size. The 16×16 macroblock is subdivided into 4×4 block size and processed from top left block. There are nine modes [2, 3] of luminance intra prediction for 4×4 block size:

- Mode-0: Vertical (VER).
- Mode-1: Horizontal (HOR).
- Mode-2: DC (DC).
- Mode-3: Diagonal Down-Left (DDL).
- Mode-4: Diagonal Down-Right (DDR).
- Mode-5: Vertical-Right (VR).
- Mode-6: Horizontal-Down (HD).
- Mode-7: Vertical-Left (VL).
- Mode-8: Horizontal-Up (HU).

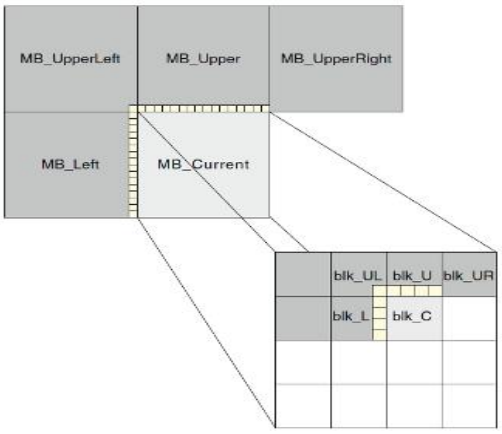

Fig.2. Reference pixels of a Luminance macroblock.

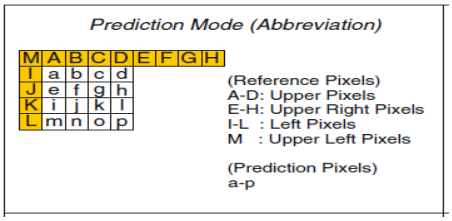

Fig.3. Reference Pixels and Prediction Pixels of 4*4 block.

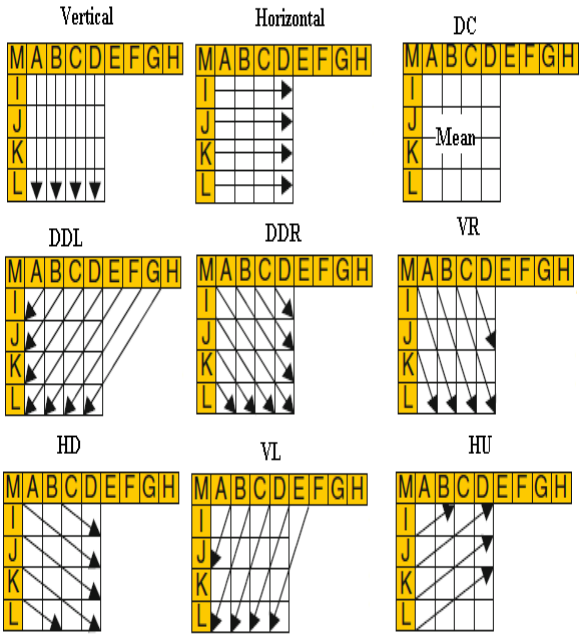

Fig.4. Processing order of Nine 4×4 Luminance Intra Modes.

Reference neighbouring pixels for blocks is shown in Fig.2 and Fig.3. Nine luminance intra predication[1,2,3] modes are calculated using neighbouring pixels shown in Fig.4. Equations for nine modes are:

Mode-0: Vertical

In vertical mode[3] the upper reconstructed pixels "A to D" shown in Fig.3 is used to predicted the block. The upper samples A, B, C and D are extrapolated vertically to predict "a to p" pixels as shown in Fig.4.

Mode-1: Horizontal

In horizontal mode[3] the left block reconstructed adjacent pixels "I to L" shown in Fig. 3 is used to predicted the block. The left samples I, J, K and L are extrapolated horizontally to predict "a to p" pixels as shown in Fig. 4.

Mode-2: DC

In DC prediction[3] the pixels " a to p" are predicted using "A to D" or "I to L" or both " A to D and I to L" neighbouring reconstructed block adjacent pixels. The block position decides which neighbouring pixels to be used for prediction and is explained below.

If "A to D and I to L" adjacent pixels are available then "a to p" pixels are calculated by the following formula.

a-p=(SH+SV+4)/8 (1)

SH=Sum of left adjacent pixels I to L.

SV= Sum of top adjacent pixels A to D.

Else if only "I to L" adjacent pixels are available then "a to p" pixels are calculated by the following formula.

a-p=(SH+2)/4. (2)

SH=Sum of left adjacent pixels I to L.

Else if only "A to D" adjacent pixels are available then "a to p" pixels are calculated by the following formula.

a-p=(SV+2)/4. (3)

SV=Sum of left adjacent pixels A to D.

Else if there is no adjacent pixels for a particular block then mean 128 is the value for prediction pixels.

Equations for **mode-3 to mode-8[3]** are given in table.1. Fig.4 gives the representation of mode3 to mode 8 which shows which neighbouring pixels should be available to process these modes. These equations are use to predict "a to p" prediction pixels.

By using Mode-0 to Mode-8 equation efficient and parallel design of luminance 4×4 block size intra prediction is implemented on FPGA.

Table.I. Equation for Mode-3 to Mode-8

DDL	**DDR**	**VR**	**HD**	**VL**	**HU**
a= (A+2B+C+2)>>2 b=e= (B+2C+D+2)>>2 c=f=i= (C+2D+E+2)>>2 d=g=j=m= (D+2E+F+2)>>2 h=k=n= (E+2F+G+2)>>2 l=o= (F+2G+H+2)>>2 p=(G+3H+2)>>2	m= (L+2K+J+2)>>2 i=n= (K+2J+I+2)>>2 e=j=o= (J+2I+M+2)>>2 a=f=k=p= (I+2M+A+2)>>2 b=g=l= (M+2A+B+2)>>2 c=h= (A+2B+C+2)>>2 d= (B+2C+D+2)>>2	a=j=(M+A+1)>>1 b=k=(A+B+1)>>1 c=l=(B+C+1)>>1 d=(C+D+1)>>1 m= (K+2J+I+2)>>2 i= (J+2I+M+2)>>2 e=n= (I+2M+A+2)>>2 f=o= M+2A+B+2)>>2 g=p= (A+2B+C+2)>>2 h= (B+2C+D+2)>>2	m=(L+K+1)>>1 i=o=(K+J+1)>>1 e=k=(J+I+1)>>1 a=g=(I+M+1)>>1 n= (L+2K+J+2)>>2 j=p= (K+2J+I+2)>>2 f=l= (J+2I+M+2)>>2 b=h= (I+2M+A+2)>>2 c= (M+2A+B+2)>>2 d= A+2B+C+2)>>2	a=(A+B+1)>>1 b=i=(B+C+1)>>1 c=j=(C+D+1)>>1 d=k=(D+E+1)>>1 l=(E+F+1)>>1 e= (A+2B+C+2)>>2 f=m= (B+2C+D+2)>>2 g=n= (C+2D+E+2)>>2 h=o= (D+2E+F+2)>>2 p= (E+2F+G+2)>>2	k=l=m=o=p=L g=i= (L+K+1)>>1 c=e= (K+J+1)>>1 a=(J+I+1)>>1 h=j= (3L+J+2)>>2 d=f= (L+2K+J+2)>>2 b= (K+2J+I+2)>>2

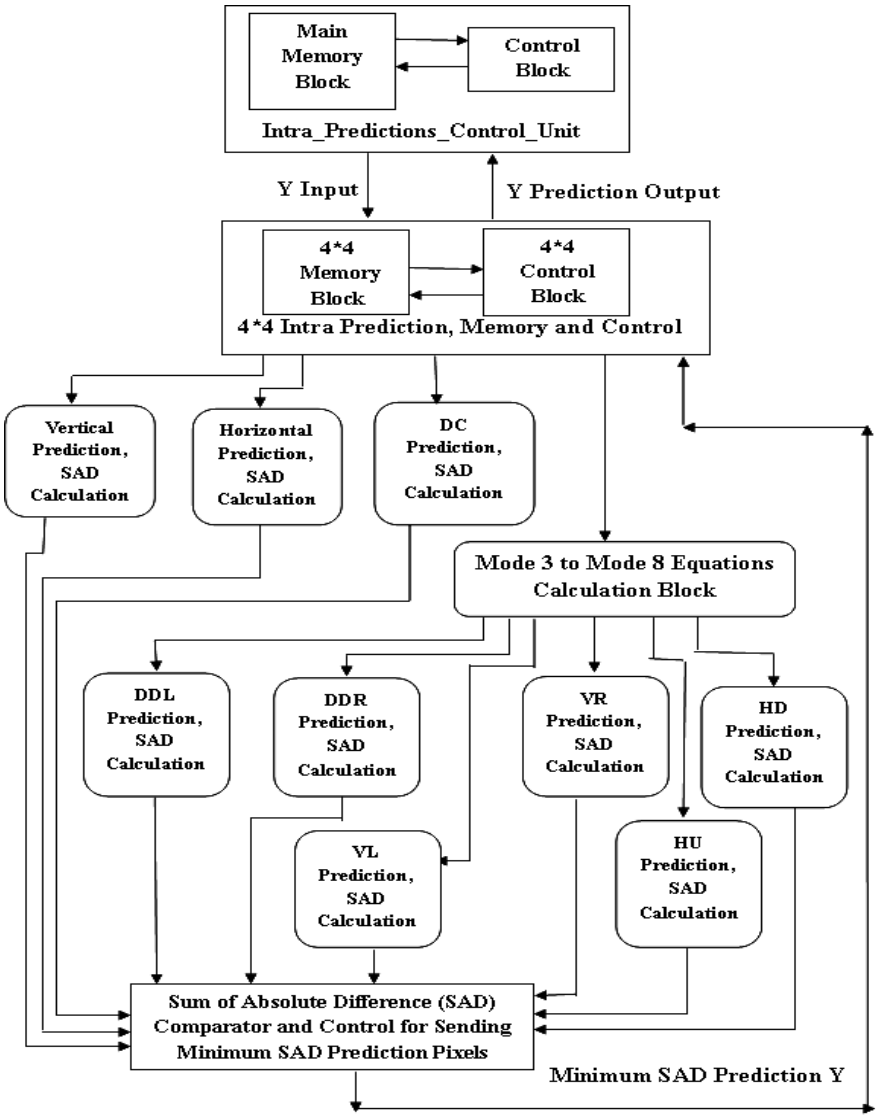

Fig.5. Architecture of Implementation of 4×4 Size Luminance Intra Prediction.

III. DESIGN AND IMPLEMENTATION OF 4×4 LUMINANCE INTRA PREDICTION

This section explains about implementation of 4×4 block size luminance intra prediction modes on FPGA. Code is designed in Verilog using Xilinx ISE tool. The architecture of 4×4 luminance intra prediction is shown in Fig.5. Intra Predictions control unit controls all operations. It stores original and reconstructed pixels in memory. It divides original frame into blocks and searches for corresponding neighbouring pixels in reconstructed block memory and sends the current original block and neighbouring pixels into 4×4 memory and control unit. 4×4 control unit depending on neighbours [1, 3] "A to D, E to H, I to L or M" available decides which modes should be executed.

If no neighbours available the 4*4 intra process control unit will not predict any pixels and sends mean 128 values as "a to p" prediction pixels.

If only "I to L" pixels are available the 4*4 control unit sends control signals to Horizontal (HOR), DC (DC) and Horizontal Up (HU) prediction and sum of absolute differences (SAD) calculation unit. The SAD comparator compares SAD of the entire three units and sends prediction pixels to the main control unit depending which intra predicted pixels have minimum SAD.

If only "A to D and E to H" pixels are available then 4*4 control unit shown in Fig.11 sends control signals to

Vertical(VER), DC(DC), Diagonal Down Left(DDL) and Vertical-Left (VL) prediction and SAD calculation unit.

After all these four modes are calculated the SAD comparator finds minimum SAD modes and sends that mode predicted pixels to the main block.

If all "A to D, E to H, I to L and M" neighbouring pixels are available then all nine modes explained in section 2 is calculated using neighbouring pixels. After all nine modes are available then SAD comparator compares to find minimum SAD and then sends out the predicted pixels related to minimum SAD mode.

If only "A to D, I to L and M" neighbouring pixels are available then 4*4 intra prediction control unit sends control signals to Vertical, Horizontal, DC, DDR, VR, HD, HU unit to predict pixels using neighbouring pixels. After all these modes are predicted minimum SAD mode pixels are sent to main block.

To reduce prediction time between mode 3 to mode 8, i.e. for DDL, DDR, VR, HD, VL, HU a new approach[4, 5, 6] is used.

If we compare the equations used to calculate prediction pixels given in Table.1 for mode 3 to mode 8, we find that most of the equations in different modes are similar. So we can calculate all similar equation at once and assign the value of the equation to the particular mode where it is needed. So we can reduce the time which was taken to calculate similar equations separately for mode 3 to mode 8. In Fig.5 a separate block is shown to calculate mode 3 to mode 8 equations. All the equations are calculated using shift and add operation. There is no multiplication architecture used because it uses more area and time than shift and add operations. This approach reduces the FPGA area utilization.

The 4×4 block processing order of our design is shown in Fig.6 (b)

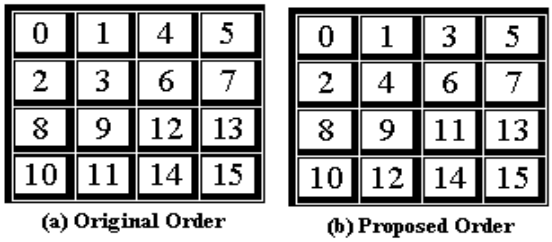

Fig.6. Proposed Order.

This order is useful in finding minimum error prediction pixels and efficient use of reconstructed pixels memory. In our proposed order after processing block 2 we process block 3 in row one and later process block 4 in row 2 than we can process all nine modes for block 4 in our proposed order. But in original order if we process block 3 after block 2 in row 2 since neighbouring pixels "E to H" are not available for block 3 in original order so only some modes are processed and result in more error.

Our design uses parallel approach to predict pixels. AS shown in Fig.5 all the modes are executed in parallel. Vertical, Horizontal and DC modes are calculated in parallel and sum of absolute differences (SAD) is calculated separately for all modes. Mode 3 to mode 8 after calculating equation, values are assigned parallel and SAD is calculated separately for mode 3 to mode 8. After calculating all modes and SAD depending on neighbours available minimum SAD is calculated. The prediction pixels related to minimum SAD is sent to main unit and stored in memory.

SAD formula is given below:

$$SAD = \sum_{i=0}^{N} \sum_{j=0}^{N} C_{i,j} - R_{i,j}$$

Where $C_{i,j}$ are the original pixels and Ri, j are the prediction pixels for the particular 4*4 block. If we implement SAD in conventional way on FPGA it will take more time to calculate SAD. So in this project a new approach is used to implement SAD. As we know original pixels and prediction pixels are in the range of 0 to 255. The 8 bit original and prediction pixel are assigned to two 9 bit registers In SAD equation, we 2's complement 9 bit $R_{i,j}$ value. And add it with Ci; j. If 9th bit of the result is 1 then 2's complement of result is done to get result. If 9th bit is 0 then there is no 2's complement for the result.

It is explained using example given below.

Consider $C_{1;1}$=198 and $R_{1;1}$=213. Then to calculate sum of absolute difference we assign these two values to 9 bit register

C1[8 : 0] = 198 = 011000110;

R1[8 : 0] = 213 = 011010101;

Take 2's complement of R1 and add to C1.

Sum[8:0]= [198-213] =C1+~R1+1= 011000110+100101011 = 111110001;

If we consider above result 9th bit is 1.So it implies the result is negative. So we take 2's complement of result. If 9^{th} bit is zero than there is no need of 2's complement to Sum and the Sum result is final answer.

111110001=2'scomplement is 000001110+1=1111=15.
[|198-213|=15].
SAD is calculated to get error between original block and predicted block. Using SAD value best mode is selected and prediction pixels of that mode are sent to main unit.

IV. SIMULATION AND SYNTHESIS RESULTS

Intra 4×4 luminance intra prediction is designed in verilog language using Xilinx ISE and simulates using Xilinx ISIM. The video frame data required for simulation is stored in text file using matlab and given as input by reading text file for simulation using verilog test bench. Simulation result is verified mathematically. Test image used for simulation is shown in Fig.7.

Fig.7: Test Image.

Fig.8: Y Component

Luminance component of image is shown Fig.8. Intra Predicted Y component is shown in Fig.9.

Fig.9: Intra Predicted YComponent.

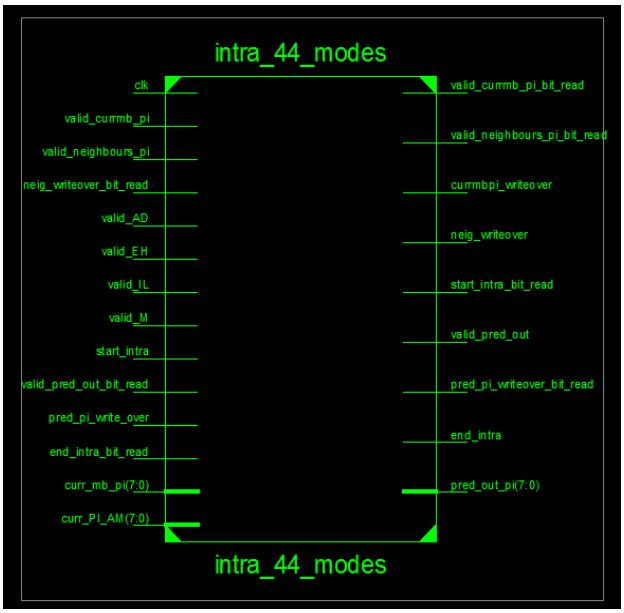

Fig.10: RTL Schematic for 4×4 Luminance Intra Prediction.

Table.II: Device Utilization Summary for 4×4 Intra prediction Module.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	836	1,60,000	1%
Number used as Flip Flops	797		
Number used as AND/OR logics	39		
Number of Slice LUTs	1,872	80,000	2%
Number used as logic	1,787	80,000	2%
Number used as Memory	80	27,840	1%
Number used as Dual Port RAM	80		
Number of occupied Slices	678	20,000	3%
Number of LUT Flip Flop pairs used	2,041		
Number with an unused Flip Flop	1,266	2,041	62%
Number with an unused LUT	169	2,041	8%
Number of fully used LUT-FF pairs	606	2,041	29%
Number of unique control sets	67		
Number of slice register sites los to control set restrictions	303	1,60,000	1%
Number of bonded IOBs	44	240	18%
Average Fanout of Non-Clock Nets	3.93		

This design is implemented on Xilinx Virtex-6 (Device: xc6vcx130T, Package: ff484, Speed:-2) FPGA. The RYL schematic of 4×4 intra prediction modes is shown in Fig.10. Table.2 gives the design summary which gives information about FPGA device utilization.

Power utilization is analysed using Xpower analyser and results are shown in Fig.11.The 4×4 luminance intra prediction module was implemented as a sub module in intra prediction project shown in Fig. Fig.12. The design in Fig.12 was designed to predict 4×4 and 16×16 luminance(Y), 8×8 chrominance (Cb,Cr) intra prediction modes with RGB to YCbCr and YCbCr to RGB module. The design summary of intra prediction control unit is shown in table.3.

V. CONCLUSION AND FUTURE WORK

Intra prediction for 4×4 luminance component is designed and implemented on FPGA effectively. Our design of 4×4 luminance intra prediction can be used for faster prediction so that overall encoding time can be reduced.
Our design is compared with previous design and our parallel approach to process intra prediction modes results in less latency. The design developed can be redesigned to implement on ASIC's. This project can be used various FPGA based H.264 video encoding and video processing applications

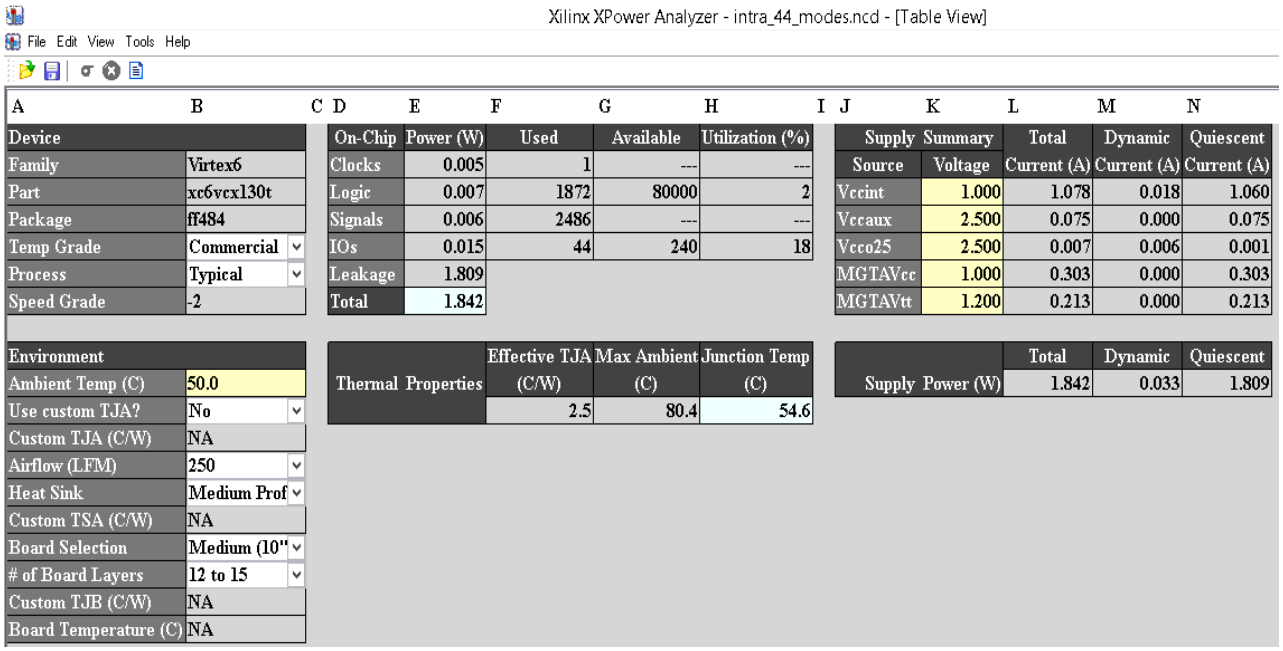

Fig.11: Power Estimation Results for 4×4 Block Size Luminance Intra Prediction.

Table.III: Design Summary of Intra Prediction.

Slice Logic Utilization	Used	Availabl e	Utilizatio n
Number of Slice Registers	5,722	1,60,000	3%
Number used as Flip Flops	4,986		
Number used as AND/OR logics	736		
Number of Slice LUTs	43,554	80,000	54%
Number used as logic	20,641	80,000	25%
Number used as Memory	22,784	27,840	81%
Number used as Dual Port RAM	22,784		
Number of occupied Slices	12,315	20,000	61%
Number of LUT Flip Flop pairs used	43,686		
Number with an unused Flip Flop	38,340	43,686	87%
Number with an unused LUT	132	43,686	1%
Number of fully used LUT-FF pairs	5,214	43,686	11%
Number of unique control sets	1,270		
Number of slice register sites lost to control set restrictions	2,842	1,60,000	1%
Number of bonded IOBs	41	240	17%
Number of RAMB36E1/FIFO36E1s	96	264	36%
Average Fanout of Non-Clock Nets	5.88		

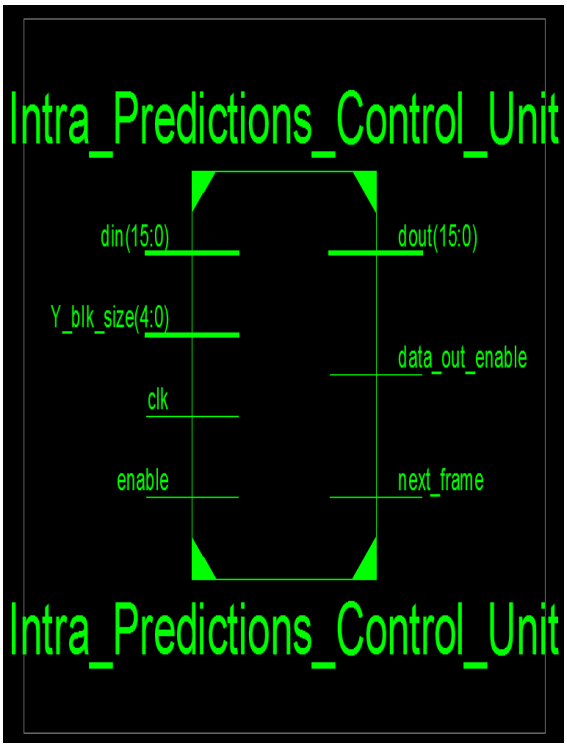

Fig.12: RTL Schematic of Intra Prediction.

REFRENCES

[1].Thomas Wiegand, Gary J. Sullivan, Gisle Bjontega and, and Ajay Luthra, "Overview of the H.264 / AVC Video Coding Standard", IEEE Trans. On Circuits and Systems for Video Technology, July 2003.

[2].I.Richardson, "The H.264 Advanced video Compression Standard", John Wiley & Sons, 2010.

[3].Youn-Long Steve Lin ,Chao-Yang Kao Huang-Chih Ku and Jian-Wen Chen, "VLSI Design for Video Coding- H.264/AVC Encoding from Standard Specification to Chip" Springer Science+Business Media, LLC 2010.

[4].Muhammad Nadeem, Stephan Wong, and Georgi Kuzmanov, "An efficient hardware design for intra-prediction in H.264/AVC decoder", Electronics, communications and Photonics Conference (SIECPC), 2011 Saudi International.

[5].Mikołaj Roszkowski and Grzegorz Pastuszak, "Intra Prediction Hardware Module For High-Profile H.264/AVC Encoder " , Signal Processing Algorithms, Architectures, Arrangements, and Applications Conference Proceedings (SPA), 2010.

[6].A. Ben Atitallah, H. Loukil and N. Masmoudi, "FPGA Design For H.264/AVC Encoder", International Journal of Computer Science, Engineering and Applications (IJCSEA), Vol.1, No.5, October 2011.

[7]. http://iphome.hhi.de/suehring/tml/,"H.264/AVC Software Coordination" JM Reference Software.

[8].http://www.asic-world.com.

[9].http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/xst_v6s6.pdf