

AN EFFICIENT H.264 INTRA FRAME CODER SYSTEM DESIGN

Ilker Hamzaoglu, Ozgur Tasdizen, Esra Sahin

Faculty of Engineering and Natural Sciences, Sabanci University

34956, Tuzla, Istanbul, Turkey

hamzaoglu@sabanciuniv.edu, tasdizen@su.sabanciuniv.edu, esra@su.sabanciuniv.edu

ABSTRACT

In this paper, we present an efficient H.264 / MPEG4 Part 10 Intra Frame Coder System. The system achieves real-time performance for portable applications with low hardware cost, and it includes a novel intra prediction hardware design. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code works at 71 MHz in a Xilinx Virtex II FPGA and it code 35 CIF frames (352x288) per second. The system also includes a software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The H.264 Intra Frame Coder hardware and software are demonstrated to work together on an Arm Versatile Platform development board.

1. INTRODUCTION

Video compression systems are used in many commercial products, from consumer electronic devices such as digital camcorders, cellular phones to video teleconferencing systems. These applications make the video compression systems an inevitable part of many commercial products. To improve the performance of video compression systems, recently, H.264 / MPEG4 Part 10 video compression standard, offering significantly better video compression efficiency than previous standards, is developed with the collaboration of ITU and ISO standardization organizations.

The video compression efficiency achieved in H.264 standard is not a result of any single feature but rather a combination of a number of encoding tools. As it is shown in the top-level block diagram of an H.264 encoder in Figure 1, one of these tools is the intra prediction algorithm used in the baseline profile of H.264 standard [1, 2, 3]. Intra prediction algorithm generates a prediction for a Macroblock (MB) based on spatial redundancy. H.264 intra prediction algorithm achieves better coding results than the intra prediction algorithms used in the previous video compression standards. However, this coding gain comes with an increase in encoding complexity which makes it an exciting challenge to have a real-time implementation of H.264 intra prediction algorithm.

H.264 Intra Frame Coder is a video encoder which uses H.264 intra prediction algorithm for generating predictions for each MB [4, 5]. H.264 intra frame coder is a competitive alternative to JPEG2000 for still image compression, in terms of both coding efficiency and computational complexity. H.264 intra frame coder is also shown to be superior to Motion-JPEG2000, especially at lower resolutions, for motion picture production, editing and archiving, where video frames are coded as I-frames only to allow for random access to each individual picture.

In this paper, we present an efficient H.264 Intra Frame Coder System. The system achieves real-time performance for portable applications with low hardware cost, and it includes a novel intra

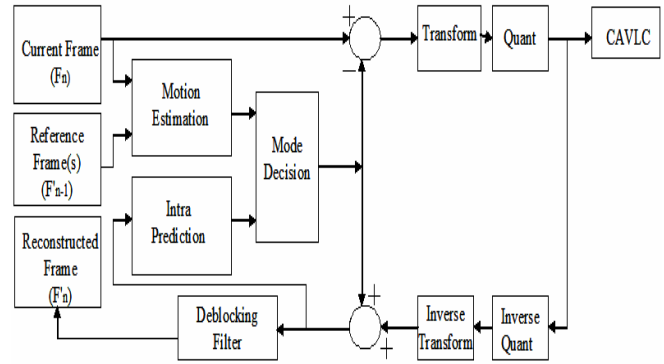


Figure 1. H.264 Encoder Block Diagram

prediction hardware design. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code works at 71 MHz in a Xilinx Virtex II FPGA and it code 35 CIF frames (352x288) per second. The system also includes a software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The H.264 Intra Frame Coder hardware and software are demonstrated to work together on an Arm Versatile Platform PB926EJ-S development board.

An H.264 Intra Frame Coder hardware is presented in [4, 5]. This hardware achieves higher performance than our hardware design at the expense of a much higher hardware cost. Our hardware design is a more cost-effective solution for portable applications. They use four reconfigurable datapaths, which include 12 adders, 16 multiplexers, 4 shifters and 4 clippers, in their intra prediction hardware design. They use additional adders and multiplexers for preprocessing in 16x16 plane mode and 8x8 plane mode. On the other hand, we use three reconfigurable datapaths, which include 6 adders, 12 multiplexers, 6 shifters and 2 clippers, in our intra prediction hardware design. We don't use any additional hardware resources for 16x16 plane mode and 8x8 plane mode.

The rest of the paper is organized as follows. Section 2 explains the H.264 intra frame coder algorithm. Section 3 describes the proposed architecture in detail. The implementation results are given in Section 4. Finally, Section 5 presents the conclusions.

2. H.264 INTRA FRAME CODER ALGORITHM

The top-level block diagram of an H.264 Intra Frame Coder is shown in Figure 2. An H.264 Intra Frame Coder has a forward path and a reconstruction path [1, 2, 3, 4, 5]. The forward path is used to encode a video frame and create the bitstream. The reconstruction path is used to decode the encoded frame and reconstruct the decoded frame. Since a decoder never gets original images, but

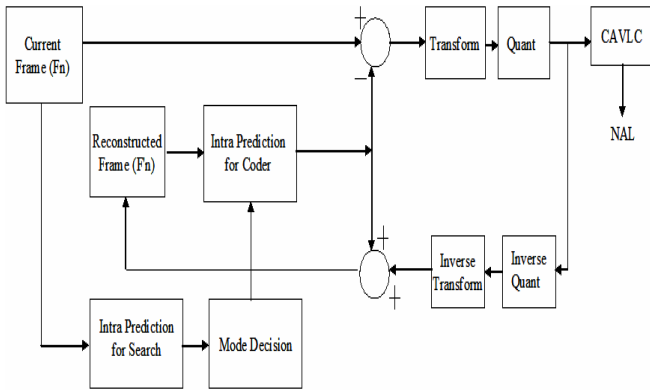


Figure 2. H.264 Intra Frame Coder Block Diagram

rather works on decoded frames, reconstruction path in the encoder ensures that both encoder and decoder use identical reference frames for intra prediction. This avoids possible encoder – decoder mismatches.

Intra prediction algorithm predicts the pixels in a MB using the pixels in the available neighboring blocks [1, 2, 3]. For the luma component of a MB, a 16x16 predicted luma block is formed by performing intra predictions for each 4x4 luma block in the MB and by performing intra prediction for the 16x16 MB. There are nine prediction modes for each 4x4 luma block and four prediction modes for a 16x16 luma block. A mode decision algorithm is then used to compare the 4x4 and 16x16 predictions and select the best luma prediction mode for the MB. 4x4 prediction modes are generally selected for highly textured regions while 16x16 prediction modes are selected for flat regions.

There are nine 4x4 luma prediction modes designed in a directional manner. Each 4x4 luma prediction mode generates 16 predicted pixel values using some or all of the neighboring pixels A to M as shown in Figure 3. The pixels A to M belong to the neighboring blocks and are assumed to be already encoded and reconstructed and are therefore available in the encoder and decoder to generate a prediction for the current block. The arrows indicate the direction of prediction in each mode. The predicted pixels are calculated by a weighted average of the neighboring pixels A-M for each mode except Vertical, Horizontal and DC modes. The prediction equations used in 4x4 Diagonal Down-Left prediction mode are shown in Figure 4 where [y,x] denotes the position of the pixel in a 4x4 block (the top left, top right, bottom left, and bottom right positions of a 4x4 block are denoted as [0, 0], [0, 3], [3, 0], and [3, 3], respectively) and $\text{pred}[y,x]$ is the prediction for the pixel in the position [y,x].

There are four 16x16 luma prediction modes designed in a directional manner. Vertical, Horizontal and DC modes are similar to 4x4 luma prediction modes. Plane mode is an approximation of bilinear transform with only integer arithmetic.

For the chroma components of a MB, a predicted 8x8 chroma block is formed for each 8x8 chroma component by performing intra prediction for the MB. There are four 8x8 chroma prediction modes. Vertical, Horizontal, DC and Plane modes are similar to 16x16 luma prediction modes. A mode decision algorithm is used to compare the 8x8 predictions and select the best chroma prediction mode for chroma components of the MB. Both chroma components of a MB always use the same prediction mode.

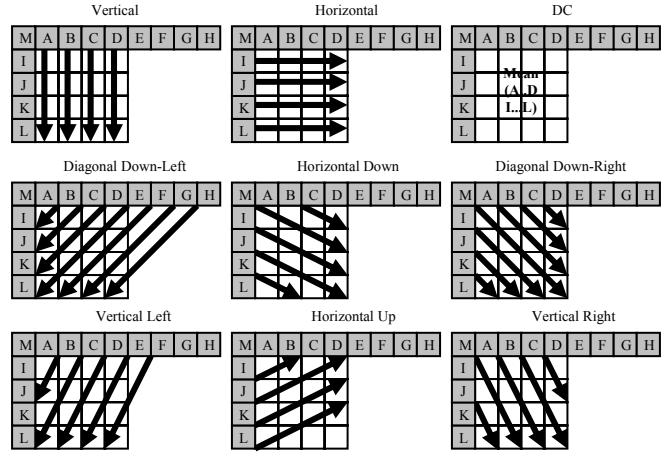


Figure 3. 4x4 Luma Prediction Modes

$$\begin{aligned}
 \text{pred}[0, 0] &= A + 2B + C + 2 \gg 2 \\
 \text{pred}[0, 1] &= B + 2C + D + 2 \gg 2 \\
 \text{pred}[0, 2] &= C + 2D + E + 2 \gg 2 \\
 \text{pred}[0, 3] &= D + 2E + F + 2 \gg 2 \\
 \text{pred}[1, 0] &= B + 2C + D + 2 \gg 2 \\
 \text{pred}[1, 1] &= C + 2D + E + 2 \gg 2 \\
 \text{pred}[1, 2] &= D + 2E + F + 2 \gg 2 \\
 \text{pred}[1, 3] &= E + 2F + G + 2 \gg 2 \\
 \text{pred}[2, 0] &= C + 2D + E + 2 \gg 2 \\
 \text{pred}[2, 1] &= D + 2E + F + 2 \gg 2 \\
 \text{pred}[2, 2] &= E + 2F + G + 2 \gg 2 \\
 \text{pred}[2, 3] &= F + 2G + H + 2 \gg 2 \\
 \text{pred}[3, 0] &= D + 2E + F + 2 \gg 2 \\
 \text{pred}[3, 1] &= E + 2F + G + 2 \gg 2 \\
 \text{pred}[3, 2] &= F + 2G + H + 2 \gg 2 \\
 \text{pred}[3, 3] &= G + 3H + 2 \gg 2
 \end{aligned}$$

Figure 4. Prediction Equations for 4x4 Diagonal Down-Left Mode

The predicted MB is subtracted from the current MB to generate the residual MB. Residual MB is transformed using forward transform algorithm [1, 2, 3]. Transform algorithm is based on a 4x4 integer transform which only uses integer addition and binary shift operations. Transform coefficients are then quantized and re-ordered in a zig-zag scan order [1, 2, 3]. The quantization algorithm uses a non-uniform quantizer and it requires an integer multiplication. Quantization parameter can take a value between 0-51 and an increment of 1 in quantization parameter results in 12.2% increment in quantization step size. The reordered quantized transform coefficients are entropy encoded using context adaptive variable length coding (CAVLC) algorithm [1, 2, 3]. CAVLC uses multiple tables for a syntax element and it adapts to the current context by selecting one of these tables for a given syntax element based on the already transmitted syntax elements.

The quantized transform coefficients are also reconstructed. The quantized transform coefficients are inverse quantized and inverse transformed to generate the reconstructed residual data. Since quantization is a lossy process, inverse quantized and inverse transformed coefficients are not identical to the original residual data. The reconstructed residual data are added to the predicted pixels in order to create the reconstructed frame.

3. PROPOSED HARDWARE ARCHITECTURE

The proposed H.264 intra frame coder hardware, as shown in Figure 5, includes a search & mode decision hardware and a coder hardware that work in a pipelined manner. After the first MB of the input frame is loaded to the input register file, search & mode decision hardware starts to work on determining the best mode for coding this MB. After search & mode decision hardware determines the best mode for the first MB, coder hardware starts to code the first MB using the selected best mode and search & mode decision hardware starts to work on the second MB. The entire frame is processed MB by MB in this order.

This is achieved by performing intra prediction in the search & mode decision hardware using the pixels in the current frame rather than the pixels in the reconstructed frame. However, intra prediction in the coder hardware is performed using the pixels in the reconstructed frame in order to be compliant with H.264 standard. This makes the MB pipelining possible at the expense of a small PSNR loss in the video quality [4, 5].

3.1 Proposed Search & Mode Decision Hardware

The proposed search & mode decision hardware, as shown in Figure 6, includes Intra Prediction, Residue, Hadamard Transform and Mode Decision modules. The efficient intra prediction hardware design presented in [6] is used in the proposed hardware. In the proposed hardware, there are two parts operating in parallel in order to complete the search & mode decision process faster. The upper part is used for finding the best 16x16 luma prediction mode for the luma component of a MB and the best 8x8 chroma prediction mode for the chroma components of a MB. The lower part is used for finding the best 4x4 luma prediction mode for each 4x4 block in the luma component of a MB.

Top level scheduling for the upper part of the search & mode decision hardware for 16x16 luma predictions is shown in Figure 7. First, the neighboring buffers in the intra prediction hardware are loaded with the corresponding neighboring pixels from the current MB register. Then, the intra prediction hardware generates the pixel predictions for the luma component of the current MB using the first available 16x16 luma mode and writes the predicted pixels to the prediction buffer. The Residue hardware, then, calculates the difference between the corresponding luma pixels in the current MB and the predicted MB. As the residue data associated with the first pixel position in a MB is calculated, Hadamard Transform module starts to calculate the Sum of Absolute Transformed Difference (SATD) for that mode using the residue data. So, Residue and Hadamard Transform modules are overlapped.

Hadamard transform for SATD calculations of 16x16 luma prediction modes requires storing DC coefficients in a register, because Hadamard transform has to be applied to these coefficients again. The multiplexer before Hadamard Transform module selects between DC coefficients and coefficients from the residue block.

After Hadamard Transform module finishes calculating SATD for an available 16x16 luma prediction mode of a MB, it decides whether it is the mode with lowest cost or not. After each available 16x16 luma prediction mode for a MB is searched, the prediction mode with the lowest cost and its cost information are sent to the Top Level Mode Decision hardware.

When the upper part of the search & mode decision hardware finishes with available 16x16 luma modes of a MB for luma samples, it starts to work with 8x8 chroma modes of the same MB for chroma samples. Top level scheduling for chroma samples is similar to that of luma samples.



Figure 5. H.264 Intra Frame Coder Block Diagram

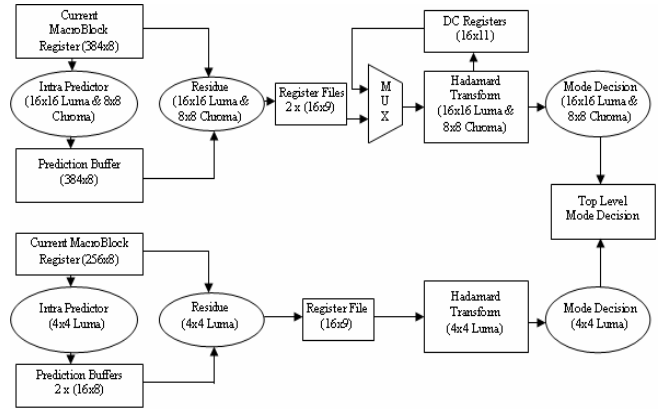


Figure 6. Search & Mode Decision Hardware Block Diagram

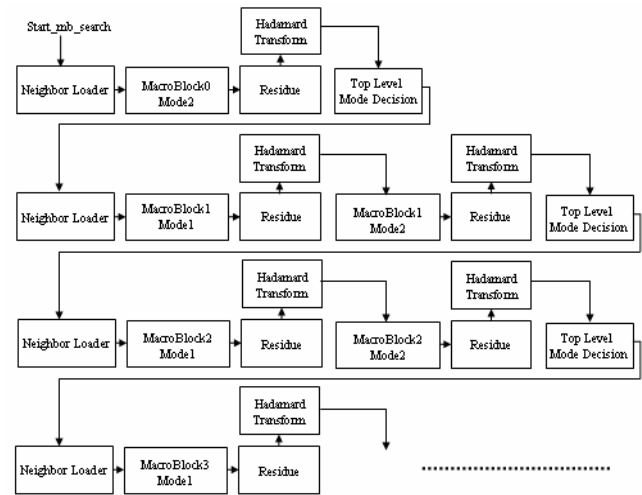


Figure 7. Schedule for 16x16 Luma Prediction Modes

Table 1. Latencies of the Modules in the Upper Part of the Search & Mode Decision Hardware

Module	Latency
Neighbor Loader	256 clock cycles
Hadamard Transform	288 clock cycles
Residue Module	256 clock cycles
Intra Prediction – Mode0	257 clock cycles
Intra Prediction – Mode1	257 clock cycles
Intra Prediction – Mode2	273 clock cycles
Intra Prediction – Mode3	340 clock cycles

The latencies of the modules in the upper part of the search & mode decision hardware are given in Table 1. In the worst case, when all 16x16 prediction modes are available, intra search for a MB takes 256*4 (Neighbor Loader) + 1127 (Intra Prediction) + 288*4 (Hadamard Transform) + 1*4 (Top Level Mode Decision) = 3307 clock cycles.

Top level scheduling for the lower part of the search & mode decision hardware is shown in Figure 8. Before intra prediction hardware for the first available mode of a 4x4 luma block starts, the corresponding entries of the neighboring buffers for that 4x4 block in the prediction hardware are loaded with the neighboring pixels from the current MB register file. After generating pixel predictions of a 4x4 luma block using an available 4x4 luma prediction mode, the difference (residue) between the current 4x4 luma block and the predicted 4x4 luma block is calculated by Residue module. When the Residue module finishes the calculation of residue data for a 4x4 luma block for the current available 4x4 luma prediction mode, Hadamard Transform module starts to calculate SATD for that mode using the residue data. After Hadamard Transform finishes to calculate SATD for a 4x4 luma prediction mode, mode decision hardware for 4x4 luma blocks determines whether this prediction mode is the mode with lowest cost or not.

Intra prediction module is overlapped with Residue and Hadamard Transform modules. As the Residue and Hadamard Transform modules are working on the current available 4x4 luma prediction mode for a 4x4 luma block, intra prediction module starts to generate the prediction for the next available 4x4 luma prediction mode for the same 4x4 luma block if the current available 4x4 luma prediction mode is not the last available mode for the current 4x4 luma block.

If the current available 4x4 prediction mode is the last available 4x4 luma prediction mode for the current 4x4 luma block, Neighbor Loader module starts to load the corresponding neighboring pixels for the next 4x4 luma block from the current MB register file as the Residue module is working on the current 4x4 luma block. The Residue module is again followed by Hadamard Transform module. After Neighbor Loader finishes loading the neighboring pixels of the next 4x4 luma block, intra prediction module starts to generate the prediction for the first available 4x4 luma prediction mode for the next 4x4 luma block. All the 4x4 luma blocks in a MB are processed in this order.

After intra prediction for all 4x4 blocks in a MB is finished, most probable mode calculation module determines the number of selected modes which are not the most probable mode for each 4x4 block in a MB and uses this information to calculate the cost of using intra 4x4 prediction for a MB (for each 4x4 block, $Cost_{4x4} = SATD + 4\lambda R$, where $R=0$ when selected mode is the most probable mode and $R=1$ otherwise). Most probable mode calculation module has vertical and horizontal buffers that are used for storing the most probable mode information of the 4x4 blocks in the MB boundaries.

Finally, Top Level Mode Decision module uses the results produced by the individual mode decision modules of the lower and upper parts of search & mode decision hardware to determine the prediction modes with lowest cost for a MB (one mode for luma samples and one mode for chroma samples) and sends this information to the coder hardware. The mode decision algorithm implemented in the proposed mode decision hardware is the same as the algorithm implemented in the H.264 Joint Model (JM) reference software encoder when there is no Rate-Distortion optimization.

In order to complete the SATD operations faster, the high speed Hadamard transform hardware shown in Figure 9 is designed. The proposed hardware finishes SATD operations of a 4x4 block in 18 clock cycles.

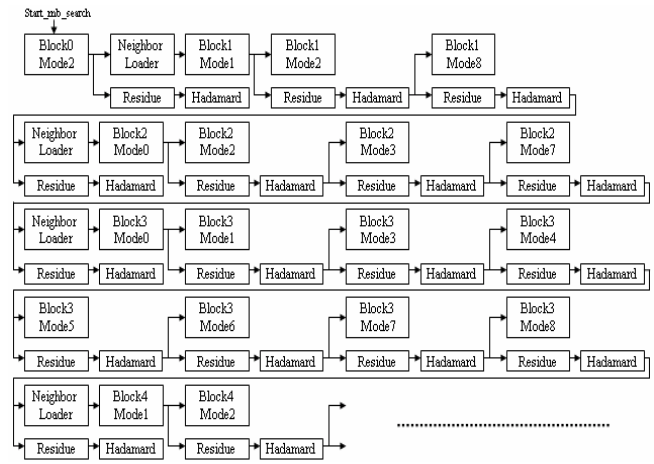


Figure 8. Schedule for 4x4 Luma Prediction Modes

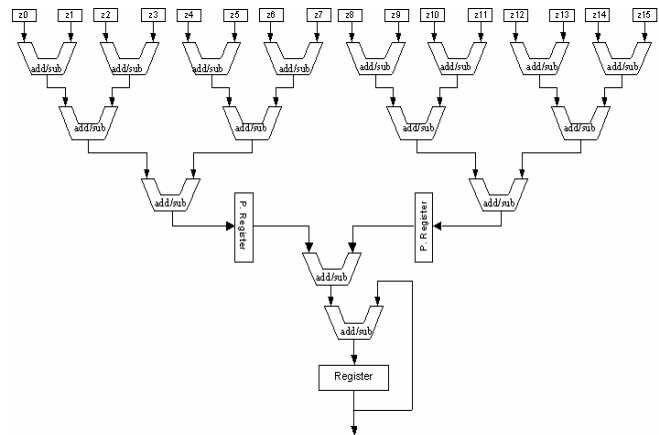


Figure 9. Hadamard Transform Hardware

Table 2. Latencies of the Modules in the Lower Part of the Search & Mode Decision Hardware

Module	Latency
Neighbor Loader	16 clock cycles
Hadamard Transform	18 clock cycles
Residue	18 clock cycles
Intra Prediction – Preprocessing	8 clock cycles
Intra Prediction – Mode0	17 clock cycles
Intra Prediction – Mode1	17 clock cycles
Intra Prediction – Mode2	19 clock cycles
Intra Prediction – Mode3	18 clock cycles
Intra Prediction – Mode4	18 clock cycles
Intra Prediction – Mode5	17 clock cycles
Intra Prediction – Mode6	17 clock cycles
Intra Prediction – Mode7	17 clock cycles
Intra Prediction – Mode8	17 clock cycles

The latencies of the modules in the lower part of the search & mode decision hardware are given in Table 2. After the prediction for each mode is generated (hadamard transform is overlapped), it takes 16 cycles for the Residue module to generate the residue block for that mode (loading neighbors is overlapped). 1 extra cycle is required after the Hadamard Transform module before starting intra prediction for the next available mode of the same 4x4 block. So, in the worst case when all 4x4 modes are available, it takes 165 (Intra Prediction) + 16*9 (Residue) + 1*9 = 318 clock cycles for performing intra search for a 4x4 luma block. After intra search for all 4x4 luma blocks in a MB is done, total cost for the selected modes for each 4x4 luma block in a MB is calculated in 18 clock cycles. Most probable mode calculation for 4x4 blocks in a MB is, then, started and this calculation takes 36 clock cycles. Finally, cost comparison between 16x16 and 4x4 intra search is initiated and it takes 9 clock cycles. Since the upper part of the search & mode decision hardware always finishes before the lower part, the lower part is the bottleneck. Therefore, intra search for a MB takes $(16*318) + 18 + 36 + 9 = 5151$ clock cycles.

3.2 Proposed Coder Hardware

The proposed coder hardware, as shown in Figure 10, includes Intra Prediction, Residue, Transform, Quant, Inverse Transform, Inverse Quant, Hadamard Transform, Reconstruction, and Entropy Coder modules. The efficient intra prediction, forward and inverse transform, forward and inverse quantization, and context-adaptive variable length coding hardware designs presented in [6, 7, 8] are used in the proposed hardware.

After the search & mode decision hardware determines the best modes for luma and chroma components of a MB, the MB is loaded to the current MB register file in the coder hardware. As soon as this loading operation finishes, intra prediction hardware generates the predicted MB using the selected best mode. Then, the Residue module creates the residual data by taking the difference between the current MB and the predicted MB and it loads the residual data to the input register file of the Transform-Quant hardware. Reconstruction module adds the results of Inverse Transform module which is stored in a 16x16 register file and the corresponding intra predicted data from the predicted MB register and clips the result to the [0-255] range. The results obtained from the reconstruction process are loaded to the neighboring pixel buffers in the intra prediction hardware and the reconstructed MB register file.

The scheduling of the Coder Hardware for a MB that will be coded with 4x4 luma prediction modes is shown in Figure 11. In the worst case, it takes 2676 clock cycles to code a MB that will be coded with 4x4 luma prediction modes. First, intra prediction hardware generates all pixel predictions for a MB based on the selected mode information for each 4x4 luma block and writes these results to the predicted MB register file. Then, the Residue block subtracts the predicted MB from the current MB. When the residual data for the first 4x4 luma block is available, Transform-Quant module starts to generate the quantized transform coefficients and loads these coefficients to the input register file of CAVLC hardware. After the quantized transform coefficients of the first 4x4 block are loaded, CAVLC and inverse Transform – Quant modules start to work. The bitstream generated by CAVLC module is stored in the output register file of CAVLC hardware. After Transform – Quant module finishes inverse quant and inverse transform operations for the first 4x4 block, reconstruction block starts to work. After the first 4x4 block of a MB is coded and reconstructed, the coder hardware starts to work on the second 4x4 block. In this way, all 4x4 blocks in a MB are coded and reconstructed.

The scheduling of the Coder Hardware for a MB that will be coded with a 16x16 luma prediction mode is shown in Figure 12. In the worst case, it takes 3680 clock cycles to code a MB that will be coded with a 16x16 luma prediction mode. Hadamard Transform has to be applied to DC coefficients after 4x4 integer transforms. Therefore, inverse quant, inverse transform, CAVLC and reconstruction operations for the MB can only start after the Hadamard transform finishes.

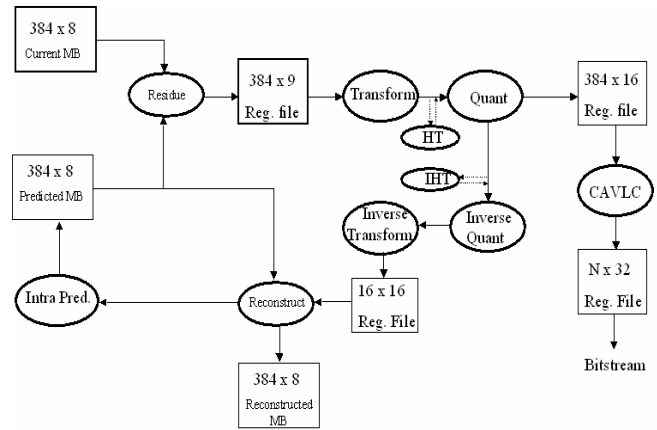


Figure 10. Coder Hardware Block Diagram

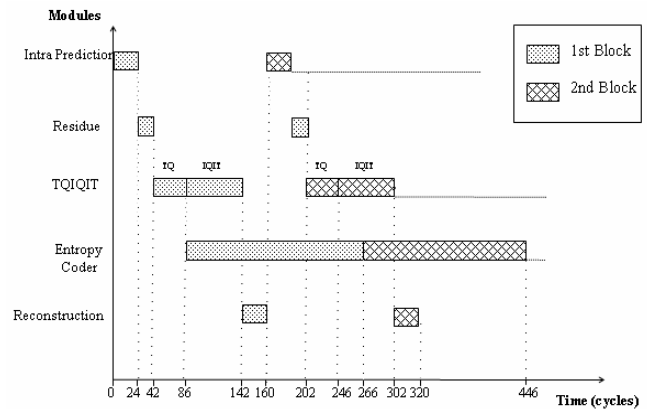


Figure 11. Coder Hardware Schedule for 4x4 Intra Modes

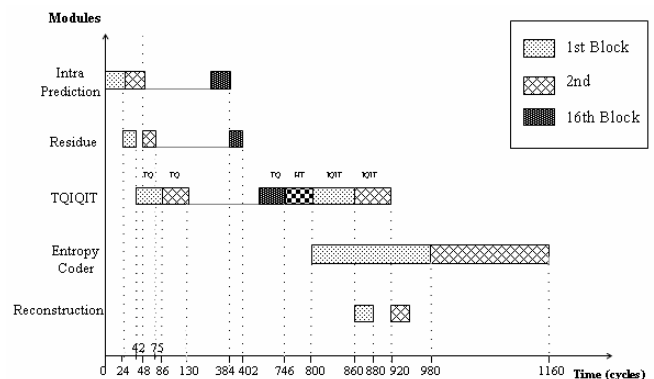


Figure 12. Coder Hardware Schedule for 16x16 Intra Modes

4. IMPLEMENTATION RESULTS

The proposed architecture is implemented in Verilog HDL. The implementation is verified with RTL simulations using Mentor Graphics ModelSim SE. The Verilog RTL is then synthesized to a 2V8000ff1152 Xilinx Virtex II FPGA with speed grade 5 using Mentor Graphics Leonardo Spectrum. The resulting netlist is placed and routed to the same FPGA using Xilinx ISE Series 7.1i. The FPGA implementation is verified to work at 71 MHz on a Xilinx Virtex II FPGA on an ARM Versatile PB926EJ-S development board shown in Figure 13.

As shown in Figure 14, an AHB bus Master interface is designed and integrated into H.264 intra frame coder hardware in order to communicate with ARM processor and SRAM through AHB bus and the H.264 intra frame coder hardware is integrated into the Xilinx Virtex II FPGA on the logic tile of the ARM Versatile PB926EJ-S development board as a master of the AHB S bus. The H.264 intra frame coder hardware is verified to work correctly on this board. The verification includes first capturing an RGB image, converting it into YCbCr format, partitioning it into MBs and writing it into an SRAM using the software running on ARM9EJ-S processor. Then, the intra frame coder hardware mapped to the Xilinx Virtex II FPGA reads the input image from the SRAM using AHB bus protocol, encodes the image and reconstructs it, and writes the reconstructed image to the SRAM using the AHB bus protocol. The conversion of reconstructed image into raster scan order and RGB color domain is then performed by software running on ARM9EJ-S processor. The reconstructed image is then displayed on a color LCD panel for visual verification.

The H.264 intra frame coder hardware is also verified to be compliant with H.264 standard. The bitstream generated by the H.264 intra frame coder hardware for an input frame is successfully decoded by H.264 Joint Model (JM) reference software decoder and the decoded frame is displayed using a YUV Player tool for visual verification.

The proposed H.264 intra frame coder hardware includes a search & mode decision hardware and a coder hardware that work in a pipelined manner. Since, in the worst case, the search & mode decision hardware takes 5151 clock cycles for a MB and the coder hardware takes 3680 clock cycles for a MB, the intra frame coder hardware takes 5151 clock cycles for a MB. Therefore, the FPGA implementation can process a CIF frame in $396 \text{ MB} * 5151 \text{ clock cycles per MB} * 14 \text{ ns clock cycle} = 28.5 \text{ msec}$. Therefore, it can process $1000/28.5 = 35 \text{ CIF (352x288) frames per second}$.

The FPGA implementation including input, output and internal RAMs and register files uses the following FPGA resources; 19589 Function Generators, 9795 CLB Slices, 3698 DFFs, and 1 Block Multiplier, i.e. %21.02 of Function Generators, %21.02 of CLB Slices, %3.83 of DFFs, and %0.6 of Block Multipliers.

5. CONCLUSION

In this paper, we presented an efficient H.264 Intra Frame Coder System. The system achieves real-time performance for portable applications with low hardware cost, and it includes a novel intra prediction hardware design. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code works at 71 MHz in a Xilinx Virtex II FPGA and it code 35 CIF frames (352x288) per second. The system also includes a software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The H.264 Intra Frame Coder hardware and software are demonstrated to work together on an Arm Versatile Platform development board.

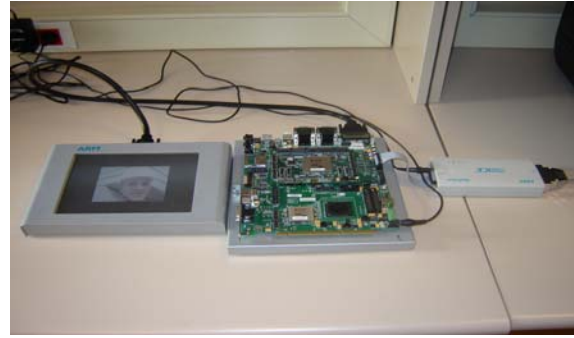


Figure 13. Arm Versatile PB926EJ-S Development Board

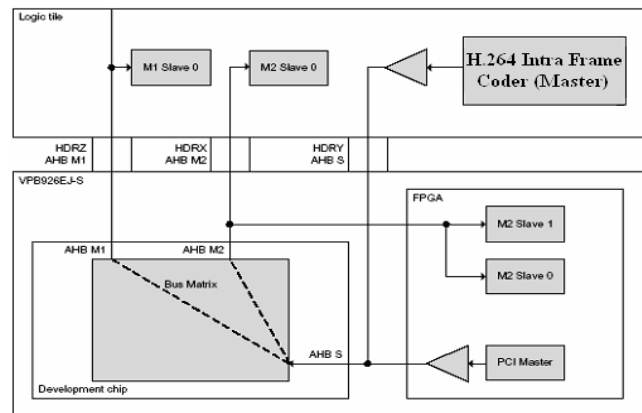


Figure 14. Integration of H.264 Intra Frame Coder Hardware into Arm Versatile PB926EJ-S Development Board

6. REFERENCES

- [1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560–576, July 2003.
- [2] I. G. Richardson, H.264 and MPEG-4 Video Compression, Wiley, 2003.
- [3] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.
- [4] Y. Huang, B. Hsieh, T. Chen, and L. Chen, "Hardware Architecture Design for H.264/AVC Intra Frame Coder", Proc. of IEEE ISCAS, pp. 269-272, April 2004.
- [5] Y. Huang, B. Hsieh, T. Chen, and L. Chen, "Analysis, Fast Algorithm and VLSI Architecture Design for H.264/AVC Intra Frame Coder", IEEE Trans. on CAS for Video Technology, March 2005.
- [6] E. Sahin and I. Hamzaoglu, "An Efficient Hardware Architecture for H.264 Intra Prediction Algorithm", Design, Automation and Test in Europe (DATE) Conference, April 2007.
- [7] O. Tasdizen and I. Hamzaoglu, "A High Performance and Low Cost Hardware Architecture for H.264 Transform and Quantization Algorithms", European Signal Processing Conf., September 2005.
- [8] E. Sahin and I. Hamzaoglu, "A High Performance and Low Power Hardware Architecture for H.264 CAVLC Algorithm", European Signal Processing Conf., September 2005.