An Efficient H.264 Intra Frame Coder System
İlker Hamzaoglu, Member, IEEE, Ozgur Tasdizen and Esra Sahin

Abstract — In this paper, we present an efficient H.264 intra frame coder system that achieves real-time performance for portable consumer electronics applications with low hardware cost. The system includes a low cost intra prediction hardware design that implements all intra prediction modes used in H.264 video coding standard based on a novel organization of the intra prediction equations. The proposed hardware is implemented in Verilog HDL. The Verilog RTL code works at 71 MHz in a Xilinx Virtex II FPGA and it can code 35 CIF (352x288) frames per second. The system also includes software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The H.264 intra frame coder system is demonstrated to work correctly on an Arm Versatile Platform development board and it is verified to be compliant with H.264 standard.

Index Terms — H.264, Video Coding, Intra Frame Coder, Intra Prediction, Hardware Implementation, FPGA.

I. 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 Fig. 1, one of these tools is the intra prediction algorithm used in the baseline profile of H.264 standard [1, 2]. Intra prediction algorithm generates a prediction for a Macroblock (MB) based on spatial redundancy. H.264 intra prediction algorithms achieve coding results than the intra prediction algorithms used in the previous video compression standards. However, this coding gain comes with an increasing 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 [1, 2]. It is a competitive alternative to Motion-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 that achieves real-time performance for portable consumer electronics applications with low hardware cost. The system includes a low cost intra prediction hardware design that implements all intra prediction modes used in H.264 standard based on a novel organization of the intra prediction equations [3, 4], a low cost forward and inverse transform and quantization hardware design [5] and a low cost context-adaptive variable length coding hardware design [6].

The proposed hardware is implemented in Verilog HDL. The Verilog RTL code works at 71 MHz in a Xilinx Virtex II FPGA and it can code 35 CIF (352x288) frames per second. The system also includes software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The system is demonstrated to work correctly on an Arm Versatile Platform development board and it is verified to be compliant with H.264 standard.

The H.264 intra frame coder hardware presented in [7] achieves higher performance than our hardware design at the expense of a higher hardware cost. The pipelined architecture presented in [8] for H.264 intra frame coding achieves higher performance than our hardware design, but it degrades the compression efficiency since it excludes some of the intra prediction modes for achieving pipelined execution. The fast intra mode decision algorithm presented in [9] reduces the amount of computation for H.264 intra frame coding by using local edge information. However, it degrades the compression efficiency since it only tries the best intra prediction modes based on the local edge direction rather than trying all intra prediction modes.

1 This research was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK).

I. Hamzaoglu, O. Tasdizen and E. Sahin are with Department of Electronics Engineering, Sabanci University, Tuzla 34956, Istanbul, Turkey (e-mail: hamzaoglu@sabanciuniv.edu, tasdizen@sabanciuniv.edu, esra@sabanciuniv.edu).

Contributed Paper
Manuscript received September 21, 2008

0098-3063/08/$20.00 © 2008 IEEE
Our H.264 intra frame coder hardware tries all intra prediction modes and implements the Lagrangian mode decision algorithm (SATD+λR) used in the H.264 Joint Model (JM) reference software encoder [10].

The rest of the paper is organized as follows. Section II explains the H.264 intra frame coder algorithm. Section III describes the proposed intra prediction hardware and Section IV describes the proposed intra frame coder hardware in detail. The implementation results are given in Section V. Finally, Section VI presents the conclusions.

II. H.264 INTRA FRAME CODER ALGORITHM

The top-level block diagram of an H.264 intra frame coder is shown in Fig. 2. An H.264 intra frame coder has a forward path and a reconstruction path [1, 2]. 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 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]. 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 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 Fig. 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. DC mode is always used regardless of the availability of neighboring pixels. However, it is adopted based on which neighboring pixels A-M are available. The other prediction modes can only be used if all of the required neighboring pixels are available.

The prediction equations used in 4x4 Diagonal Down-Left prediction mode are shown in Fig. 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. Each 16x16 luma prediction mode generates 256 predicted pixel values using some or all of the upper and left-hand neighboring pixels. Vertical, Horizontal and DC modes are similar to 4x4 prediction modes. Plane mode is an approximation of bilinear transform with only integer arithmetic. DC mode is always used regardless of the availability of the neighboring pixels. However, it is adopted based on which neighboring pixels are available. The other prediction modes can only be used if all of the required neighboring pixels are available.

---

The prediction equations used in 4x4 Diagonal Down-Left prediction mode are shown in Fig. 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 \( [y,x] \).
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 which are similar to 16x16 luma prediction modes. A mode decision algorithm is used to compare the 8x8 predictions and select the best prediction mode for each chroma component of the MB. Both chroma components of a MB always use the same prediction mode.

The predicted MB is subtracted from the current MB to generate the residual MB. Residual MB is transformed using forward transform algorithm. 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. 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. 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.

III. PROPOSED INTRA PREDICTION HARDWARE

The proposed hardware architecture for intra prediction is shown in Fig. 5 [3, 4]. The proposed hardware generates the predicted pixels for both luma and chroma components of a MB using available prediction modes. In the proposed hardware, there are two parts operating in parallel in order to perform intra prediction faster.

The upper part is used for generating the predicted pixels for the luma component of a MB using available 16x16 luma prediction modes and for generating the predicted pixels for the chroma components of a MB using available 8x8 chroma prediction modes. The size of register files that are used for the current MB and the prediction buffer is 384x8, because they are used for storing both luma and chroma components of the current and predicted MB respectively.

The lower part is used for generating the predicted pixels for each 4x4 block in the luma component of a MB using available 4x4 luma prediction modes. The lower part is more computationally demanding and it is the bottleneck in the intra prediction hardware. The size of the current MB register file is 256x8, because it is used for storing only luma components of the current MB. The size of the prediction buffer is 16x8 since it is used for storing the predicted pixels for a 4x4 luma block.

Two local neighboring buffers, local vertical register file and local horizontal register file, are used to store the neighboring pixels in the previously coded and reconstructed left-hand and upper neighboring 4x4 luma blocks in the current MB respectively. After a 4x4 luma block in the current MB is coded and reconstructed, the neighboring pixels in this block are stored in the corresponding local register files. The proposed hardware uses this data to determine the neighboring pixels in the left-hand and upper previously coded neighboring 4x4 luma blocks in the current MB.

Six global neighboring buffers, three global vertical neighboring buffers and three global horizontal neighboring buffers, are used to store the neighboring pixels in the previously coded and reconstructed neighboring MBs of the current MB. The 16x16 luma components of the MBs in a frame and the 4x4 luma blocks in them are shown in Fig. 6.

Global luma vertical register file is used to store the neighboring pixels in 4x4 luma blocks 5, 7, 13 and 15 of the previously coded MB. The proposed hardware uses this data to determine the neighboring pixels in the left-hand previously coded neighboring MB of 4x4 luma blocks 0, 2, 8, and 10 in the current MB. Global Cb and Cr vertical register files are used for Cb and Cr components of the MBs.

Fig. 5. Intra Prediction Hardware.

Fig. 6. 16x16 and 4x4 Luma Blocks in a Frame.
Global luma horizontal register file is used to store the neighboring pixels in 4x4 luma blocks 10, 11, 14, and 15 of the previously coded MBs in the previously coded MB row of the frame. The proposed hardware uses this data to determine the neighboring pixels in the upper previously coded neighboring MB of 4x4 luma blocks 0, 1, 4, and 5 in the current MB. Global Cb and Cr horizontal register files are used for Cb and Cr components of the MBs.

Instead of using one large external SRAM, we have used 8 internal register files to store the neighboring reconstructed pixels in order to reduce power consumption by accessing a small register file for storing and reading a reconstructed pixel instead of accessing a large external SRAM. In addition, we have disabled the register files when they are not accessed in order to reduce power consumption.

A. Proposed Hardware for 4x4 Luma Prediction Modes

After a careful analysis of the equations used in 4x4 luma prediction modes, it is observed that there are common parts in the equations and some of the equations are identical. The intra prediction equations are organized for exploiting these observations to reduce both the number of memory accesses and computation time required for generating the predicted pixels. The organized prediction equations for Diagonal Down-Left and Diagonal Down-Right 4x4 luma prediction modes are shown in Fig. 7. As it can be seen from the figure, 

\[(A + B), (B + C), (C + D), (D + E), (E + F), (F + G), (G + H), (J + K), (I + J), (M + I)\] and \[(M + A)\] are common in two or more equations, and some of the prediction equations (e.g. \[(A + B) + (B + C) + 2\]) are identical.

The proposed hardware first calculates the results of the common parts in all the 4x4 luma prediction modes and stores them in temporary registers. It, then, calculates the results of the prediction equations using the values stored in these temporary registers. If both the left and top neighboring blocks of a 4x4 luma block are available, 12 common parts are calculated in the preprocessing step and this takes 8 clock cycles. The neighboring buffers are only accessed during this preprocessing. Therefore, they are disabled after the preprocessing for reducing power consumption.

The proposed datapath for generating predicted pixels for a 4x4 luma block using all 4x4 luma prediction modes is shown in Fig. 8. Level0 (L0) registers are used to store the results of the common parts in the equations of all the 4x4 luma prediction modes. Level1 (L1) registers are used to store the results of the identical prediction equations used in all the 4x4 luma prediction modes. If both the left and top neighboring blocks of a 4x4 luma block are available, it takes 165 clock cycles to generate the predicted pixels for that 4x4 block using available 4x4 luma prediction modes.

Since the order of the equations used in a 4x4 luma prediction mode is not important for functional correctness, the equations are ordered to keep the inputs of the adders the same for as many consecutive clock cycles as possible. This avoids unnecessary switching activity and reduces the power consumption.
B. Proposed Hardware for 16x16 Luma Prediction Modes and 8x8 Chroma Prediction Modes

After a careful analysis of the equations used in 16x16 luma prediction modes, it is observed that Vertical, Horizontal and DC mode equations can directly be implemented using adders and shifters, however the equations used in Plane mode can be organized to avoid using a multiplier and to reduce computation time required for generating the predicted pixels. The organized Plane mode prediction equations for block 0 in a MB are shown in Fig. 9 where \( p \) represents the neighboring pixel values and Clip1 is to clip the result between 0 and 255.

The proposed datapath for implementing all 16x16 luma prediction modes is similar to the proposed datapath for 4x4 luma prediction modes. It first calculates the common parts \( C_0, (C_0 + b), (C_0 + 2b), \) and \( (C_0 + 3b) \) and stores them in temporary registers. It, then, generates the predicted pixels in the first row by using the values stored in these temporary registers. It, then, adds \( c \) to the values stored in the temporary registers and stores the resulting values in the same temporary registers. It, then, generates the predicted pixels in the second row by using the values stored in these temporary registers. It repeats this process until all the predicted pixels for the current MB are generated.

If both the left and top neighboring MBs of a 16x16 luma block are available, it takes 1127 clock cycles to generate the predicted pixels for that 16x16 luma block using available 16x16 luma prediction modes. Plane mode is the most computationally demanding 16x16 luma prediction mode. The predicted pixels for a 16x16 luma block are generated in 340 clock cycles using Plane mode.

Since 8x8 chroma prediction modes are similar to 16x16 luma prediction modes, the proposed hardware for 8x8 chroma prediction modes is also similar to the proposed hardware for 16x16 luma prediction modes. If both the left and top neighboring MBs of an 8x8 chroma block are available, it takes 302 clock cycles to generate the predicted pixels for that 8x8 chroma block using available 8x8 chroma prediction modes. Plane mode is also the most computationally demanding 8x8 chroma prediction mode. The predicted pixels for an 8x8 chroma block are generated in 95 clock cycles using Plane mode.

IV. PROPOSED INTRA FRAME CODER HARDWARE

The proposed H.264 intra frame coder hardware includes a search & mode decision hardware and 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 at the expense of a small PSNR loss in the video quality. 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.

A. Proposed Search & Mode Decision Hardware

The proposed search & mode decision hardware, as shown in Fig. 10, includes Intra Prediction, Residue, Hadamard Transform (HT) and Mode Decision modules. 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 Fig. 11. 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, HT module starts to calculate the Sum of Absolute Transformed Difference (SATD) for that mode using the residue data. So, Residue and HT modules are overlapped.

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

After HT 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.

The latencies of the modules in the upper part of the search & mode decision hardware are given in Table I. 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 (HT) + 1*4 (Top Level Mode Decision) = 3307 clock cycles.

<table>
<thead>
<tr>
<th>Module</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbor Loader</td>
<td>256</td>
</tr>
<tr>
<td>Hadamard Transform</td>
<td>288</td>
</tr>
<tr>
<td>Residue Module</td>
<td>256</td>
</tr>
<tr>
<td>Intra Pred. – Mode0 (Vertical)</td>
<td>257</td>
</tr>
<tr>
<td>Intra Pred. – Mode1</td>
<td>257</td>
</tr>
<tr>
<td>Intra Pred. – Mode2 (DC)</td>
<td>273</td>
</tr>
<tr>
<td>Intra Pred. – Mode3 (Plane)</td>
<td>340</td>
</tr>
</tbody>
</table>

Top level scheduling for the lower part of the search & mode decision hardware is shown in Fig. 12. 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, HT module starts to calculate SATD for that mode using the residue data. After HT 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 HT modules. As the Residue and HT 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 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 HT 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 + 4xR, 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 Lagrangian mode decision algorithm (SATD+λR) implemented in the H.264 Joint Model (JM) reference software encoder [10]. The hardware presented in [3] is used to compute SATD. This hardware computes SATD of a 4x4 block in 18 clock cycles.

The latencies of the modules in the lower part of the search & mode decision hardware are given in Table II. After the prediction for each mode is generated (HT 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 HT 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.

<table>
<thead>
<tr>
<th>Module</th>
<th>Latency (clock cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbor Loader</td>
<td>16</td>
</tr>
<tr>
<td>Hadamard Transform</td>
<td>18</td>
</tr>
<tr>
<td>Residue</td>
<td>18</td>
</tr>
<tr>
<td>Intra Pred. – Preprocessing</td>
<td>8</td>
</tr>
<tr>
<td>Intra Pred. – Mode0 (Vertical)</td>
<td>17</td>
</tr>
<tr>
<td>Intra Pred. – Mode1 (Horizontal)</td>
<td>17</td>
</tr>
<tr>
<td>Intra Pred. – Mode2 (DC)</td>
<td>19</td>
</tr>
<tr>
<td>Intra Pred. – Mode3 (Diagonal Down-)</td>
<td>18</td>
</tr>
<tr>
<td>Intra Pred. – Mode4 (Diagonal Down-)</td>
<td>18</td>
</tr>
<tr>
<td>Intra Pred. – Mode5 (Vertical-Right)</td>
<td>17</td>
</tr>
<tr>
<td>Intra Pred. – Mode6 (Horizontal-Down)</td>
<td>17</td>
</tr>
<tr>
<td>Intra Pred. – Mode7 (Vertical-Left)</td>
<td>17</td>
</tr>
<tr>
<td>Intra Pred. – Mode8 (Horizontal-Up)</td>
<td>17</td>
</tr>
</tbody>
</table>

Table II: Latencies of the Modules in the Lower Part of the Search & Mode Decision Hardware

B. Proposed Coder Hardware

The proposed coder hardware, as shown in Fig. 13, includes Intra Prediction, Residue, Transform, Quant, Inverse Transform, Inverse Quant, HT, Reconstruction, and Entropy Coder modules. The low cost forward and inverse transform, forward and inverse quantization and CAVLC hardware designs presented in [5, 6] 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 (TQ) 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 Fig. 14. 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, TQ 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 TQ modules start to work. The bitstream generated by CAVLC module is stored in the output register file of CAVLC hardware. After TQ 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 Fig. 15. In the worst case, it takes 3680 clock cycles to code a MB that will be coded with a 16x16 luma prediction mode. HT 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 HT finishes.

V. IMPLEMENTATION RESULTS

The proposed hardware 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 an 8-million-gate Xilinx Virtex II FPGA on an ARM Versatile PB926EJ-S development board.

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 MB * 5151 clock cycles per MB * 14 ns clock cycle = 28.5 msec. Therefore, it can process 1000/28.5 = 35 CIF (352x288) frames per second.

FPGA resource usages of intra prediction hardware and intra frame coder hardware including input, output and internal register files are shown in Table III. All register files are implemented as Distributed SelectRAMs.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Intra Prediction Hardware</th>
<th>Intra Frame Coder Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLB Slices</td>
<td>1001 (% 2.15)</td>
<td>9795 (% 21.02)</td>
</tr>
<tr>
<td>Function Generators</td>
<td>2002 (% 2.15)</td>
<td>19589 (% 21.02)</td>
</tr>
<tr>
<td>DFFs</td>
<td>518 (% 0.54)</td>
<td>3698 (% 3.83)</td>
</tr>
<tr>
<td>Block Multipliers</td>
<td>0</td>
<td>1 (% 0.6)</td>
</tr>
</tbody>
</table>

The H.264 intra frame coder system is verified to work correctly in the ARM Versatile PB926EJ-S development environment shown in Fig. 16. The development environment consists of a PC connected to ARM Versatile PB926EJ-S board through ARM Multi-ICE, a logic tile mounted on the Versatile PB926EJ-S baseboard and a color LCD panel [11].
An AHB bus Master interface is designed and integrated into H.264 intra frame coder hardware in order to communicate with ARM processor and external SRAM through AHB bus and the H.264 intra frame coder hardware is integrated into the Xilinx Virtex II FPGA on the logic tile as a master of the AHB S bus.

The H.264 intra frame coder is verified by first loading an RGB video frame into SRAM located on the board from PC using software. Then, the software running on ARM9EJ-S processor converts it into YCbCr format, partitions it into MBs and writes it into the SRAM. Then, the intra frame coder hardware mapped to the Xilinx Virtex II FPGA reads the input video frame from the SRAM using AHB bus protocol, encodes and reconstructs it, and writes the reconstructed video frame into the SRAM using the AHB bus protocol. The conversion of reconstructed video frame into rastor scan order and RGB color domain is then performed by software running on ARM9EJ-S processor. The reconstructed video frame is then displayed on the 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 H.264 intra frame coder hardware presented in [7] achieves higher performance than our hardware design at the expense of a higher hardware cost. They use four datapaths, which include 12 adders, 16 multiplexers, 4 shifters and 4 clippers, in their intra prediction hardware. They use additional adders and multiplexers for preprocessing in 16x16 plane mode and 8x8 plane mode. However, we use three datapaths, which include 6 adders, 12 multiplexers, 6 shifters and 2 clippers, in our intra prediction hardware. We don’t use any additional hardware resources for 16x16 plane mode and 8x8 plane mode.

They use 96x32 buffers in order to access 4 pixels in a clock cycle. However, we use 384x8 register files since we access 1 pixel in a clock cycle. They use 4 subtractors in their diff datapath. However, we use only 1 subtractor in our residue datapath. They use 16 adders and 16 internal register files in their transform / inverse transform datapath. However, we use 3 adders and 6 internal register files in our transform / inverse transform datapath.

VI. CONCLUSION

In this paper, we presented an efficient H.264 intra frame coder system that achieves real-time performance for portable consumer electronics applications with low hardware cost. The system includes a low cost intra prediction hardware design that implements all intra prediction modes used in H.264 standard based on a novel organization of the intra prediction equations, a low cost transform and quantization hardware design and a low cost CAVLC hardware design. The proposed hardware works at 71 MHz in a Xilinx Virtex II FPGA and it codes 35 CIF (352x288) frames per second. The system also includes software running on an Arm926EJS processor for implementing pre-processing and post-processing functions. The H.264 intra frame coder system is demonstrated to work correctly on an Arm Versatile Platform development board.

REFERENCES


Iker Hamzaoglu (M’90) received B.Sc. and M.Sc. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey in 1991 and 1993 respectively. He received Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign, IL, USA in 1999. He worked as a Senior and Principle Staff Engineer at Multimedia Architecture Lab, Motorola Inc. in Schaumburg, IL, USA between August 1999 and August 2003. He is working as an Assistant Professor at Sabanci University, Istanbul, Turkey since September 2003. His research interests include SoC ASIC and FPGA design for digital image and video processing and coding, low power digital SoC design, digital SoC verification and testing.

Özgür Tasdizen received B.S. degree in Electronics and Communication Engineering from Istanbul Technical University, Istanbul, Turkey in 2003. He received M.S. degree in Electronics Engineering from Sabanci University, Istanbul, Turkey in 2005 where he is currently working towards a PhD degree. His research interests include low power digital hardware design for digital video processing and compression.

Esra Sahin received B.S. and M.S. degrees in Electronics Engineering from Sabanci University, Istanbul, Turkey in 2004 and 2006 respectively. She is currently working as a design engineer at STMicroelectronics Istanbul Design Center. Her research interests include low power digital hardware design for digital video processing and compression.