

An Efficient Intra Prediction Hardware Architecture for H.264 Video Decoding

Esra Sahin and Ilker Hamzaoglu

*Faculty of Engineering and Natural Sciences, Sabanci University
34956, Tuzla, Istanbul, Turkey*

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

Abstract

In this paper, we present an efficient hardware architecture for real-time implementation of intra prediction algorithm used in H.264 / MPEG4 Part 10 video coding standard. The hardware design is based on a novel organization of the intra prediction equations. This hardware architecture is designed to be used as part of a H.264 video decoder for portable applications. The proposed architecture is implemented in Verilog HDL. The Verilog RTL is verified to work at 70 MHz in a Xilinx II FPGA. The FPGA implementation can process a VGA frame (640x480) in the worst case in 9.85 msec.

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 video compression 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 decoder in Figure 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 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 and decoding complexity which makes it an exciting challenge to have a real-time implementation of H.264 intra prediction algorithm.

The hardware architecture we presented in [3] for real-time implementation of H.264 intra prediction algorithm is designed to be used as part of a H.264 video encoder for portable applications. The intra prediction hardware used in a H.264 video encoder has to find the intra predictions for all available prediction modes of a MB. However, the intra prediction hardware used in a H.264 video decoder has to find the intra prediction only for the prediction mode selected by the encoder. Therefore, the hardware presented in [3] is optimized for finding the intra predictions for all available prediction modes of a MB.

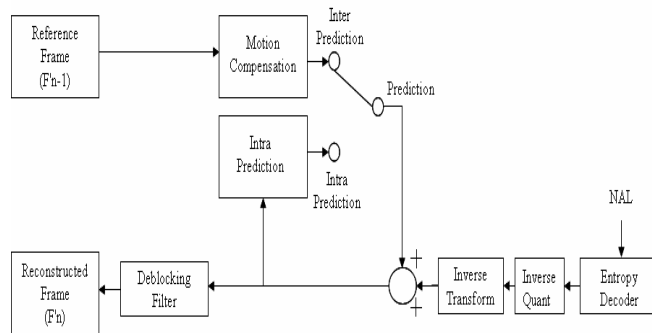


Figure 1. H.264 Decoder Block Diagram

In this paper, we also present an efficient hardware architecture for real-time implementation of H.264 intra prediction algorithm. However, this hardware is designed to be used as part of a H.264 video decoder for portable applications. The hardware design is based on a novel organization of the intra prediction equations. Since this hardware is optimized for finding the intra prediction only for a given prediction mode, it achieves higher performance for H.264 video decoding than the hardware architecture presented in [3]. The proposed architecture is implemented in Verilog HDL. The Verilog RTL code is verified to work at 70 MHz in a Xilinx Virtex II FPGA. The FPGA implementation can process a VGA frame (640x480) in the worst case in 9.85 msec.

A hardware architecture for real-time implementation of H.264 intra prediction algorithm is presented in [4]. 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 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 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 prediction algorithm. Section 3 describes the proposed architecture in detail. The implementation results are given in Section 4. Finally, Section 5 presents the conclusions.

2. Overview of H.264 Intra Prediction Algorithm

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 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 2. 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 the 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 Figure 3 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.

3. Proposed Hardware Architecture

The block diagram of the proposed intra prediction hardware architecture for H.264 video decoding is shown in Figure 4. The proposed hardware first generates the predicted pixels for the luma component of a MB using the luma prediction mode selected by the encoder, and then it generates the predicted pixels for the chroma components of a MB using the chroma prediction mode selected by the encoder.

The datapath and controller for 4x4 luma prediction modes are used for generating the predicted pixels for each 4x4 block in the luma component of a MB using the given 4x4 luma prediction modes for each 4x4 luma block if 4x4 intra prediction is selected for the luma component of a MB by the encoder. The proposed datapath and controller designs for 4x4 luma prediction modes are explained in the following section. This hardware design is based on a novel organization of the 4x4 intra prediction equations. Since this hardware is optimized for finding the intra prediction only for a given 4x4 luma prediction mode, it achieves higher performance for H.264 video decoding than the hardware architecture presented in [3] which is optimized for finding the intra predictions for all available 4x4 luma prediction modes of a MB.

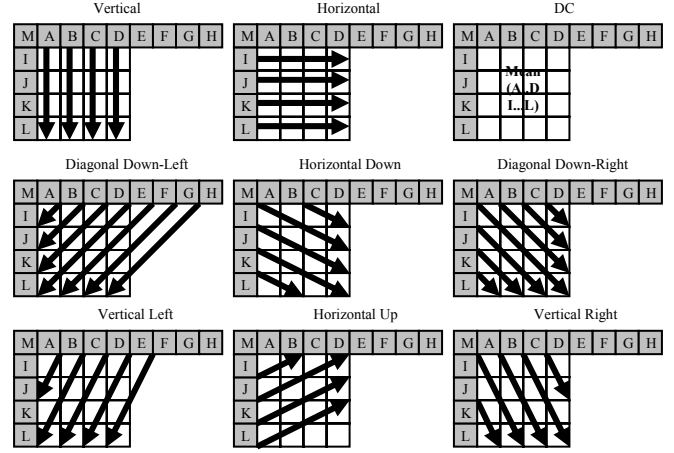


Figure 2. 4x4 Luma Prediction Modes

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

Figure 3. Prediction Equations for 4x4 Diagonal Down-Left Prediction Mode

The datapath and controller for 16x16 luma prediction modes are used for generating the predicted pixels for the luma component of a MB using the given 16x16 luma prediction mode if 16x16 intra prediction is selected for the luma component of a MB by the encoder. The datapath and controller for 8x8 chroma prediction modes are used for generating the predicted pixels for the chroma component of a MB using the given 8x8 chroma prediction mode if intra prediction is selected by the encoder. The proposed datapath and controller designs for 16x16 luma and 8x8 chroma prediction modes shown in Figure 4 are similar to the ones presented in [3]. The difference is that these hardware find the intra prediction only for a given prediction mode, whereas the hardware presented in [3] find the intra predictions for all available prediction modes of a MB.

The 384x8 prediction buffer register file is used for storing both luma and chroma components of the predicted MB.

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 neighboring 4x4 luma blocks in the predicted MB. 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.

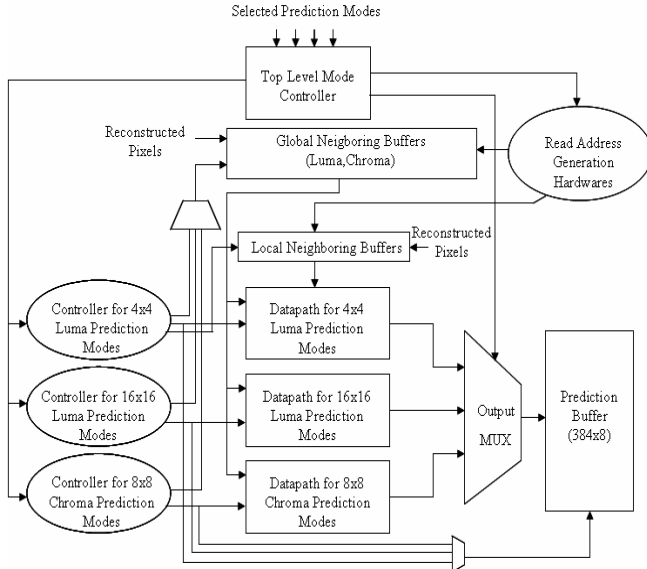


Figure 4. Intra Prediction Hardware

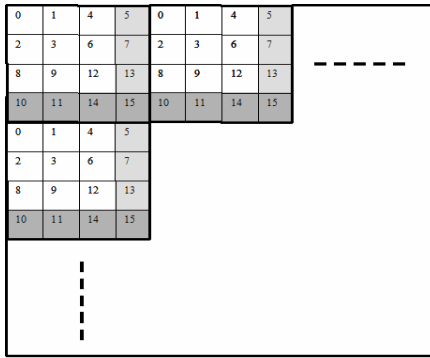


Figure 5. 16x16 and 4x4 Luma Blocks in a Frame

Local vertical register file is used to store the neighboring pixels in the left-hand previously coded and reconstructed neighboring 4x4 luma blocks in the predicted MB. Local horizontal register file is used to store the neighboring pixels in the upper previously coded and reconstructed 4x4 luma blocks in the predicted MB. 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 Figure 5.

Global luma vertical register file is used to store the neighboring pixels in the 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 the 4x4 luma blocks 0, 2, 8, and 10 in the current MB. Global Cb vertical register file and global Cr vertical register file are used for the chroma Cb and chroma Cr components of the MBs.

Global luma horizontal register file is used to store the neighboring pixels in the luma blocks 10, 11, 14, and 15 of 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 the 4x4 luma blocks 0, 1, 4, and 5 in the current MB. Global Cb horizontal register file and global Cr horizontal register file are used for the chroma Cb and chroma 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. The power consumption is reduced 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.

3.1 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 in each 4x4 luma prediction mode. 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 Figure 6. As it can be seen from the figure, (B + C), (C + D), (D + E), (E + F), (F + G) and (G + H) are common in two or more Diagonal Down-Left mode prediction equations, and some of the Diagonal Down-Left mode prediction equations (e.g. pred[0, 1] and pred[1, 0]) are identical.

The proposed datapath for generating the predicted pixels for a 4x4 luma block using the selected 4x4 luma prediction mode for that 4x4 luma block is shown in Figure 7. The proposed hardware first calculates the results of the common parts in the prediction equations of the selected 4x4 luma prediction mode for a 4x4 luma block and stores them in temporary registers REG0-REG7, i.e. it performs preprocessing for the selected 4x4 luma prediction mode.

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

(a) 4x4 Diagonal Down-Left Mode

$$\begin{aligned}
 \text{pred}[0, 2] = \text{pred}[1, 3] &= [(A + B) + (B + C) + 2] \gg 2 \\
 \text{pred}[0, 3] &= [(C + D) + (B + C) + 2] \gg 2 \\
 \text{pred}[3, 0] &= [(J + K) + (K + L) + 2] \gg 2 \\
 \text{pred}[2, 0] = \text{pred}[3, 1] &= [(J + K) + (I + J) + 2] \gg 2 \\
 \text{pred}[1, 0] = \text{pred}[2, 1] &= \text{pred}[3, 2] \\
 &= [(M + I) + (I + J) + 2] \gg 2 \\
 \text{pred}[0, 0] = \text{pred}[1, 1] &= \text{pred}[2, 2] \\
 &= \text{pred}[3, 3] = [(M + I) + (M + A) + 2] \gg 2 \\
 \text{pred}[0, 1] = \text{pred}[1, 2] &= \text{pred}[2, 3] \\
 &= [(A + B) + (M + A) + 2] \gg 2
 \end{aligned}$$

(b) 4x4 Diagonal Down-Right Mode

Figure 6. Organized Prediction Equations for 4x4 Luma Prediction Modes

It, then, calculates the results of the prediction equations of the selected 4x4 luma prediction mode for that 4x4 luma block using the values stored in the temporary registers REG0-REG7. The neighboring buffers are only accessed during the preprocessing. Therefore, they are disabled after the preprocessing for reducing power consumption.

Since the hardware architecture presented in [3] is optimized for finding the intra predictions for all available prediction modes of a MB, it first calculates the results of the common parts in the prediction equations of all the 4x4 luma prediction modes for a 4x4 luma block and stores them in temporary registers, i.e. it performs preprocessing for all available 4x4 luma prediction modes. It, then, calculates the results of the prediction equations using the values stored in these temporary registers. Since the hardware architecture proposed in this paper is optimized for finding the intra prediction only for a given prediction mode, it performs preprocessing only for the selected 4x4 luma prediction mode. Therefore, it achieves higher performance for H.264 video decoding than the hardware architecture presented in [3].

The proposed hardware calculates the results of the identical prediction equations of the selected 4x4 luma prediction mode for a 4x4 luma block only once. It, then, writes the predictions of the pixels specified by the identical prediction equations into the corresponding prediction buffer entries. Thus, no extra temporary registers are required for storing the results of the identical prediction equations of the selected 4x4 luma prediction mode for a 4x4 luma block.

The number of clock cycles required for generating the predicted pixels for a 4x4 luma block that has both left and top neighboring 4x4 luma blocks using the 4x4 luma prediction modes are given in Table 1. The number of clock cycles given in the table includes the preprocessing in each mode. If both the left and top neighboring 4x4 luma blocks of a 4x4 luma block are available, it takes 24 clock cycles in the worst case to generate the predicted pixels for that 4x4 luma block using the selected 4x4 luma prediction mode.

4. Implementation Results

The proposed intra prediction hardware 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 at 70 MHz under worst-case PVT conditions using Xilinx ISE Series 7.1i. The FPGA implementation is verified to work correctly in the 2V8000ff1152 Xilinx Virtex II FPGA on the logic tile of an ARM Versatile / PB926EJ-S development board.

If 4x4 intra prediction is selected for the luma component of a MB by the encoder, the proposed hardware generates the predicted pixels for the luma component of a MB using the 4x4 luma prediction modes selected by the encoder in the worst case in $24 \times 16 = 384$ clock cycles. If 16x16 intra prediction is selected for the luma component of a MB by the encoder, the proposed hardware generates the predicted pixels for the luma component of a MB using the 16x16 luma prediction mode selected by the encoder in the worst case in 340 clock cycles. Therefore, generating the predicted pixels for the luma component of a MB using the 4x4 luma prediction modes is the bottleneck.

The proposed hardware generates the predicted pixels for the chroma components of a MB using the 8x8 chroma prediction mode selected by the encoder in the worst case in 190 clock cycles. Therefore, the proposed hardware can process a MB in the worst case in $384 + 190 = 574$ clock cycles. Therefore, the FPGA

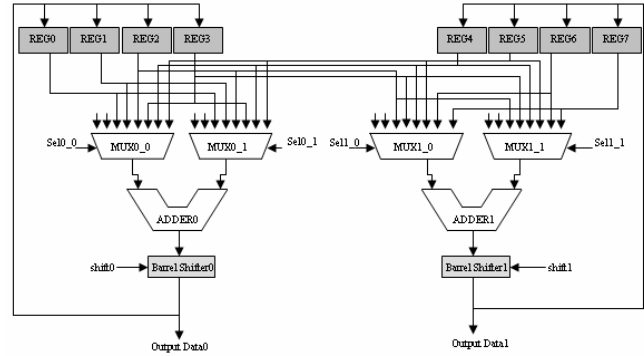


Figure 7. Datapath for 4x4 Luma Prediction Modes

Table 1. Number of Clock Cycles for Performing Intra Prediction Using 4x4 Luma Prediction Modes

Selected Mode	Clock Cycles / 4x4 Luma Block
Vertical	17
Horizontal	17
DC	21
Diagonal Down-Left Mode	24
Diagonal Down-Right Mode	24
Vertical Right	23
Horizontal Down	23
Vertical Left	22
Horizontal Up	20

implementation can process a VGA frame (640x480) in 9.85 msec ($1200 \text{ MB} * 574 \text{ cycles per MB} * 14.3 \text{ ns clock cycle} = 9.85 \text{ msec}$).

The FPGA implementation including input and output register files uses the following FPGA resources; 3034 Function Generators, 1517 CLB Slices, and 342 DFFs, i.e. %3.26 of Function Generators, %3.26 of CLB Slices, and %0.35 of DFFs.

5. Conclusion

In this paper, we presented an efficient hardware architecture for real-time implementation of intra prediction algorithm used in H.264 video coding standard. The hardware design is based on a novel organization of the intra prediction equations. This hardware is designed to be used as part of a H.264 video decoder for portable applications. The proposed architecture is implemented in Verilog HDL. The Verilog RTL code is verified to work at 70 MHz in a Xilinx Virtex II FPGA. The FPGA implementation can process a VGA frame (640x480) in the worst case in 9.85 msec.

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] Esra Sahin and Ilker Hamzaoglu, "An Efficient Hardware Architecture for H.264 Intra Prediction Algorithm", *Design, Automation and Test in Europe Conference*, April 2007.
- [4] 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 Circuits and Systems for Video Technology*, vol. 15, no. 3, March 2005.