NTNU – Trondheim
Norwegian University of
Science and Technology

Design of an 8x8 Intra Prediction Module

Kim Trønnes

Electronics System Design and Innovation
Submission date: June 2014
Supervisor: Kjetil Svarstad, IET
Co-supervisor: Milica Orlandic, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Title:	Design of an 8×8 Intra Prediction Module
Student:	Kim Trønnes

Problem description:

Based on an existing design of a 4×4 luminance intra prediction module, an 8×8 luminance intra prediction module shall be designed and tested. Additionally, the design should be extended to an 8×8 chrominance intra prediction module.

The design will be in VHDL as the existing design is. It should be tested for FPGA implementation.

Supervisor:	Kjetil Svarstad, IET
Co-supervisor:	Milica Orlandić, IET

Abstract

In this thesis, a proposed hardware architecture of an H.264/AVC 8×8 luminance intra prediction module is designed and realized in VHDL to be used on an FPGA. The module is a part of a MPEG-2 to H.264/AVC transcoder and its implementation is based on an existing design of a 4×4 luminance intra prediction module used in the same transcoder.

Intra prediction is characterized by high data dependency between input video frames which makes it hard to achieve a high throughput. This is solved by processing 16 image samples in parallel and by implementing a partial pipeline to increase efficiency.

The design is implemented and synthesized on the *Kintex-7 XC7K325T* board with a maximum clock frequency of 129.34 MHz, which gives a throughput of 456.32 Mpixels/s. This is enough to encode 2160p (4k UHD) video frames at 30 frames per second in real time.

Sammendrag

Denne masteroppgaven viser en foreslått hardwarearkitektur for en H.264/AVC 8×8 intraprediksjonsmodul for luminans, skrevet i VHDL og realisert på en FPGA. Modulen er en del av en MPEG-2-H.264/AVC-transkoder, og implementasjonen er basert på et eksisterende design av en 4×4 intraprediksjonsmodul for luminans som skal brukes i den samme transkoderen.

Intraprediksjon kjennetegnes ved høy dataavhengighet mellom nærliggende videorammer, noe som gjør at høy hastighet er vanskelig å oppnå i implementasjonen. Dette er løst ved å prosessere 16 bildeelementer samtidig og ved implementasjon av en pipeline for å øke effektiviteten.

Designet er implementert og syntetisert på en *Kintex-7 XC7K325T*-FPGA med en maksimal klokkefrekvens på 129.34 MHz. Dette gjør at systemet er i stand til å prosessere 456.32 megapiksler i sekundet, noe som er nok til å kode video med en oppløsning på 2160p (4k UHD), med en bildefrekvens på 30 Hz, i sanntid.

Preface

This master's thesis was completed at the Department of Electronics and Telecommunications at the Norwegian University of Science and Technology during the spring of 2014. It was written as the final part of the master program at NTNU and marks the end of a five year long study period to achieve a Master of Science degree in Electronics System Design and Innovation.

I would like to thank my supervisor Kjetil Svarstad for meetings on a regular basis to provide helpful advice during this thesis. Special thanks to my co-supervisor Milica Orlandic who works on the H.264 transcoder in which this thesis' design should be a part of, and who provided valuable advice throughout the design process and provided me with the source code of an already implemented 4×4 intra prediction module, which the work in theis thesis is based on.

Kim Trønnes
Trondheim, June 2014

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In recent years, the way we use digital media has begun to change. Motion Picture Experts Group 2 (MPEG-2) is still a widely used video encoding format in broadcasting. With a tendency of higher video resolution in recent media applications with portable devices, more efficient solutions in video compression might be needed to satisfy the increased bandwidth demands. The H.264 Advanced Video Coding (AVC) standard had its inception in 2003 and is now widely used in video coding. It offers higher efficiency in compression compared to earlier standards. [1]

1.1 This Thesis

This thesis focuses on the design and development of an 8×8 luminance intra prediction module as a part of a MPEG-2 to H.264/AVC intra-frame transcoder. It will be written in VHSIC Hardware Description Language (VHDL), and should be synthesizable on a field-programmable gate array (FPGA). The implementation is based on an already existing design of a 4×4 luminance intra prediction module as seen in [5]. In addition, the design should be expanded to include an 8×8 chrominance intra prediction module.

The biggest challenge of designing an intra prediction module is the data dependency between the current block being processed and the previous block due to a reconstruction loop where output data from the current block must be processed and used to make predictions for the next block. This is referred to as bubbles and limits the throughput of the process because an increased amount of clock cycles are needed to complete adjacent 8×8 blocks.

The problem description given initially was changed throughout the development and design process of the thesis work, in consultation with the supervisors of this project. Because the chrominance intra prediction module is very different compared to the luminance module, expansion into including a chrominance module was not

explored. Instead, more time was allocated towards modelling and designing a reconstruction loop to reduce the bubbles of the luminance intra prediction module.

1.1.1 Specification and Guidelines

The following requirements and specifications were provided as guidelines when developing an 8×8 luminance intra prediction module.

- Input size are one 4×4 matrix per clock cycle (it will take four clock cycles to get one 8×8 block).
- Output size should be 16 pixels per clock cycle.
- System frequency should be above 100 MHz.
- Use as few logic elements as possible.

1.1.2 Contribution

This thesis contains a proposed hardware architecture and implementation of an 8×8 luminance intra prediction module. The module is reconfigurable, to allow changes to input video frame resolutions, and supports the calculation of 16 input pixels in parallel. A reconstruction loop is modelled and accounted for in the prediction module, but the modules that are needed in a reconstruction loop are not implemented.

1.2 Outline

The thesis is organized as follows:

- Chapter 2 gives an introduction to necessary background theory such as video encoding concepts and specifics in the H.264/AVC specification.
- Chapter 3 presents the hardware architecture of the designed 8×8 luminance intra prediction module.
- Chapter 4 describes how the design is implemented on an FPGA based on its hardware architecture.
- Chapter 5 shows key results from the verification stage of the implemented design. This includes testbench simulation and synthesis.
- Chapter 6 discusses the results and compares the performance of the implemented design to its specification and hardware architecture.
- Chapter 7 contains concluding remarks of the design in addition to suggestions to further improve the 8×8 intra prediction module.

Chapter 2
Background

This chapter introduces necessary background theory about video encoding and the H.264/AVC standard.

2.1 Video Encoding Basics

Where a natural visual scene is temporally and spatially continuous, a digital video is temporally and spatially discrete. A digital video is a collection of 2-D images, usually sampled in a rectangular matrix (spatially), and are sampled with a fixed frequency (temporally). This is illustrated in Figure 2.1.

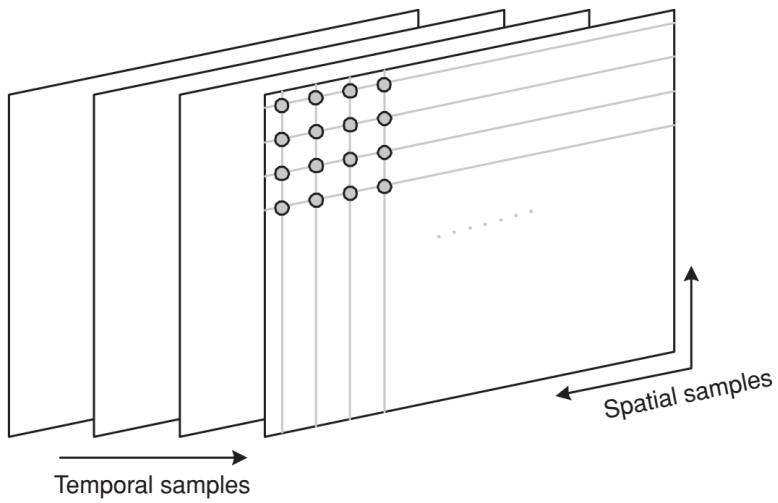

Figure 2.1: Spatial and temporal sampling of a video sequence. [1]

2.1.1 Color Spaces

The smallest unit in a digital video is the picture element (pixel), which is a spatio-temporal image sample and are represented by numbers indicating brightness and color. For a monochrome image, only one number is needed to represent the brightness or luminance (luma) of each pixel. For color images, at least three numbers are needed to indicate color. There are several different ways to do this, and the chosen method of representing brightness and color is called a color space. [1]

RGB

In the RGB color space, three numbers are used and correspond to the proportions of the three additive primary colors of light: red, green and blue. By changing the combination of these proportions, any color can be made.

YCrCb

The luminance/red chrominance/blue chrominance (YCrCb) color space is widely used in digital video because of its efficiency compared to the RGB color space. Here, Y is the luma component given by a weighted average of red, green and blue:

$$Y = k_r R + k_g G + k_b B, \tag{2.1}$$

where k are weighting factors.

C_R and C_B are color differences or chrominance (chroma), where they are the red-difference and blue-difference to the luminance Y:

$$\begin{aligned} C_R &= R - Y \\ C_B &= B - Y \end{aligned} \tag{2.2}$$

By looking at the RGB color space one would expect a green-difference component, $C_G = G - Y$, as well. Only two chroma components are needed because $C_R + C_B + C_G$ is a constant and the third chroma component can therefore be calculated from the other two.

The YCrCb color space has one advantage compared to the RGB color space in that the chroma components (C_R and C_B) can be stored with lower resolution than the luminance Y. This is because the human visual system (HVS) is less sensitive to color than to brightness (luminance). By lowering the resolution of the chroma components, the amount of data required to transmit YCrCb images are reduced with little or no loss of observed visual quality. Figure 2.2 shows an image in its original form (top-left), luminance (top-right), blue chroma (bottom-left) and red chroma (bottom-right). The Y component looks like a grey-scale version of the image

Figure 2.2: Y, C_R and C_B components of an image.

while the C_R and C_B elements are more blurry, and can be seen as an example of why these components are downscaled in image compression due to the lack of details compared to the luminance component.

2.2 Video Coding

Compression is a fundamental part of digital video coding, where the goal is to process a digital video into a format used for storage or transmission while keeping the number of bits low and the perceived quality high. Compression of a video consists of a pair of systems: A compressor (encoder) and a decompressor (decoder). Together, they are called a COder/DECoder pair (CODEC). This is illustrated in Figure 2.3.

There are two types of data compression: Lossy and lossless. The former is based on removing subjective redundancy, usually both in the spatial and temporal domain. Lossy compression gives the highest amount of compression, but the decompressed data will not be identical to the source data. Lossless compression are based on

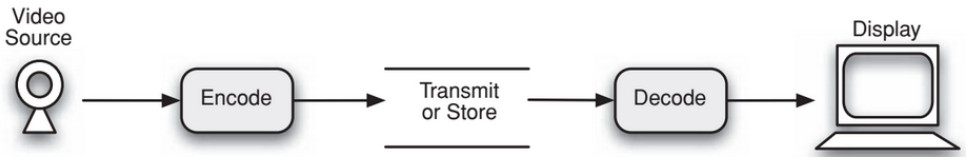

Figure 2.3: Encoder decoder pair. [1]

removing statistical redundancy. Decompressed data from lossless compression will be a perfect copy of the source data, but the amount of compression achieved will normally be lower than lossy compression.

Many modern video coding methods use lossy compression because there is often a high correlation between video frames that are close to each other in the temporal domain, especially if the video is captured with a high frame rate. In addition, there is often a high correlation between pixels in a frame that are nearby in the spatial domain. There is usually a tradeoff between bitrate (level of compression) and video quality. Methods to both lower the bitrate of compression and keeping a high perceived quality to a certain degree exist, but those often come at the price of increased computational complexity.

2.3 Video Codec Model

The H.264/AVC standard utilizes both lossy and lossless video coding methods to achieve efficient compression results. By looking into other widely used video coding formats such as MPEG-2, MPEG-4 and H.263, all of these are based on the video CODEC model seen in Figure 2.4. They use prediction-based or block-based motion compensation in a prediction model, a transform and quantization in a spatial model, and entropy encoding to get a final compressed bitstream of a video. These three processes will be looked into in greater detail in the following sections, where the prediction model is the area of focus in this thesis.

2.3.1 Prediction Model

The prediction model is where current data samples from a video frame are input and processed. This step exploits similarities between neighboring video frames and neighboring pixels in the same video frame to reduce redundancy, and are called *temporal prediction* and *spatial prediction* respectively. In both domains, the formed prediction is being subtracted from the current video data to form a set of residual (difference) samples, which is the output of the process. In addition, a set of prediction parameters are output. These can indicate the prediction type used or

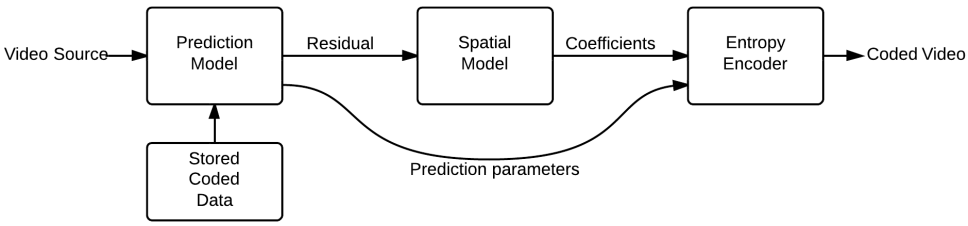

Figure 2.4: Video encoder block diagram. [1]

how motion was compensated. The more accurate the prediction is, the less amount of energy is in the residual, which leads to a better compressed coded video. The prediction model is a lossless step, where the residual that is encoded can be sent to a decoder that will create an identical prediction to reconstruct the original video frame. To make this possible, it is important that the encoder only uses data that is available to the decoder: Data that is already coded and transmitted.

Inter Prediction

As mentioned, there are two approaches to a prediction model. Temporal prediction is also called *inter prediction* and exploits the correlation between the current frame and past or future frames in time (reference frames) to form a prediction. As a simple example, a temporal prediction can be the result of using the previous frame as a predictor of the current frame. A residual frame can then be formed by subtracting the previous frame from the current one, to get the difference between them. This method has a few problems since much of the energy stored in the residual can be caused by movement between the frames, and this movement could be compensated for to get a better prediction. Methods such as block-based motion estimation and compensation exists to decrease the energy stored in the residual, but this is outside the scope of this thesis.

Intra Prediction

Intra prediction is the spatial prediction where previously coded blocks in one frame are used to predict a current block, utilizing a concept called spatial extrapolation. This is illustrated in Figure 2.5 and is used in H.264/AVC with different block sizes of 4×4, 8×8 and 16×16 pixels for the luminance component, and with a 8×8 block size for both chrominance components. [1]

Since adjacent pixels have the highest correlation between each other, only the pixels at the top edge and left edge are being used to form a prediction block. Similar to the inter prediction, the predicted block is then subtracted with the

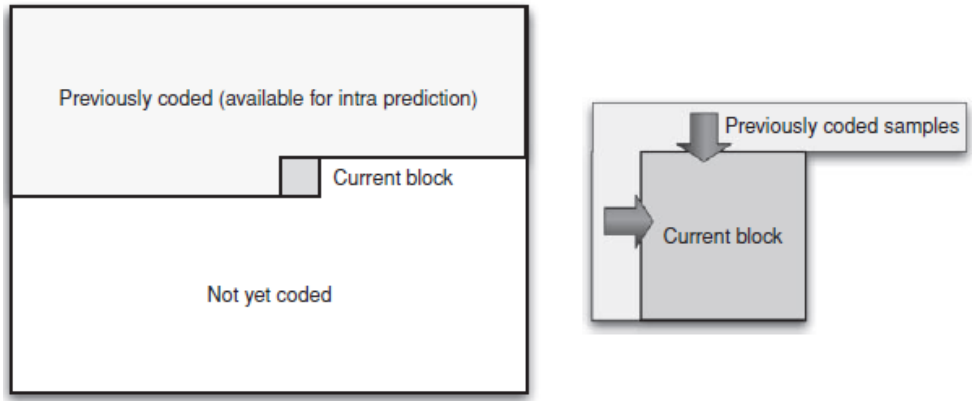

Figure 2.5: Intra prediction: available samples and spatial extrapolation. [1]

current block to form a residual which is then encoded and transformed in the spatial model. Figure 2.6 illustrates the predicted pixels and the residual from running intra prediction on the luminance component of a color picture using a block size of 8×8 pixels. Mid-grey sections of the residual represents zero or low differences, which indicates a small amount of energy. Light and dark grey sections represents positive and negative differences respectively, which corresponds to more energy or information stored in the residual. The compression rate of a frame is inversely proportional to the amount of energy in the residual.

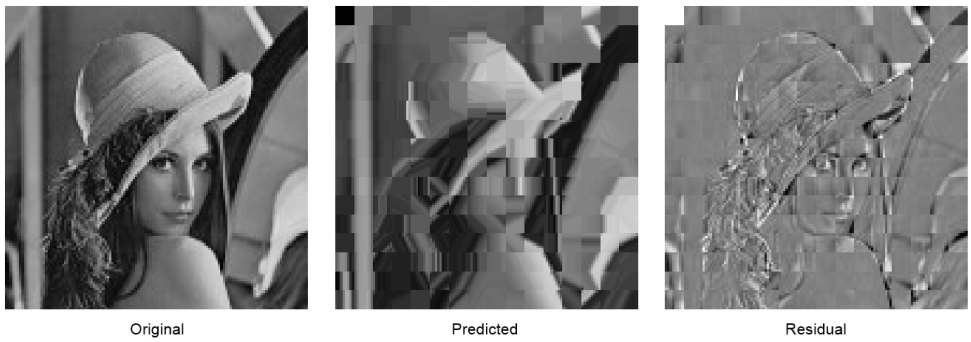

Figure 2.6: 8×8 luminance intra prediction of an image.

2.3.2 Spatial Model

The spatial model as a part of a video codec model must not be confused with the spatial prediction mentioned in the previous section. A natural image in its original form can be difficult to compress due to high correlation between nearby pixels. The residual as a result of prediction decreases the autocorrelation as seen in Figure 2.7, where a 2-D autocorrelation function of an image and the residual are shown. In the spatial model, the residual comes from the prediction model as input, and a transform is applied with the goal of further decorrelate the residual in order to increase the compression rate in the entropy coder afterwards. There are normally three stages to a spatial model: A transform, quantization which reduces the precision of transformed data, and reordering of the data to group significant values together.

The output from the spatial model are quantized transform coefficients. In the spatial model, the quantization step involves some loss to the perceptible quality of the video due to the reduction of precision.

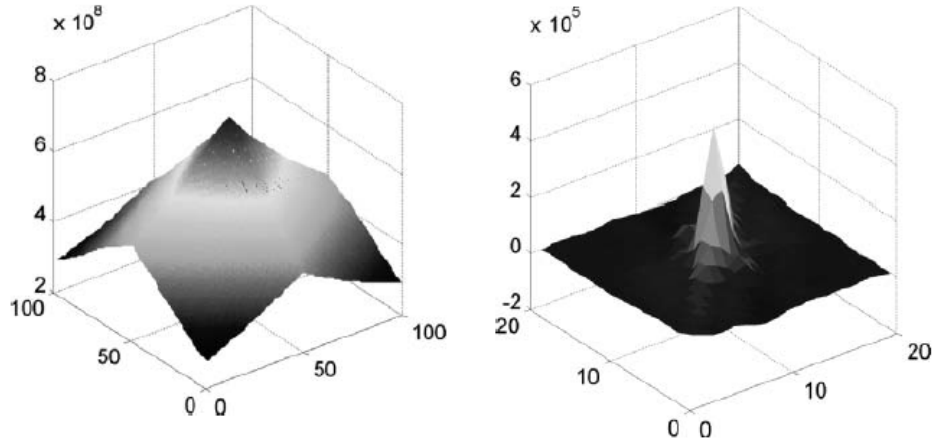

Figure 2.7: 2-D autocorrelation function of original image (left) and residual (right). [1]

Transform Coding

The transform stage in the spatial model can be achieved by different transforms. There are several criteria that could affect the choice of transform: [1]

1. The data in the transform domain should be decorrelated and compact.

2. The transform should be reversible.

3. The transform should be computationally feasible.

DCT

The discrete cosine transform (DCT) is an example of a widely used transform in video coding. It is an operation performed on $\boldsymbol{X}$, an $N \times N$ block of input values, to give $\boldsymbol{Y}$, an $N \times N$ block of coefficients. This is achieved using a transform matrix $\boldsymbol{A}$. The forward DCT of an $N \times N$ sample is defined as:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^{\boldsymbol{T}} \tag{2.3}$$

The inverse DCT (IDCT) is given by:

$$\boldsymbol{X} = \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{Y}\boldsymbol{A} \tag{2.4}$$

The elements of the transform matrix $\boldsymbol{A}$ are:

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N} \text{ where } C_i = \sqrt{\frac{1}{N}}(i=0),\ C_i = \sqrt{\frac{2}{N}}(i>0) \tag{2.5}$$

2.3.3 Entropy Encoder

Prediction parameters and the spatial model coefficients are compressed in the entropy encoder. Statistical redundancy is removed here. The output are a compressed bit stream of the video file. More details on the entropy encoder can be found in [1].

2.4 H.264 Advanced Video Coding

H.264/AVC is a standard for video coding, a format for encoded video and a toolset for video compression. The standard is defined in a document, *Recommendation H.264: Advanced Video Coding*, produced by ITU-T (International Telecommunication Union) and ISO/IEC (International Organisation for Standardisation/International Electrotechnical Commision) [6]. It defines the format and syntax for compressed video and a method for decoding the syntax. Figure 2.8 illustrates the scope of the H.264/AVC standard. The document does not mention how to encode the video, which is left to the manufacturer.

The standard was first organized in three profiles: Baseline, Extended and Main. The profiles are a way of defining which algorithms and coding tool sets that are used to create the coded video. The profile choice depends on target media applications and the first three profiles focused primarily on entertainment quality video. The H.264/AVC standard was extended in 2004 with focus on high definition videos. [7] It added the High profile, which included intra prediction with a block size of

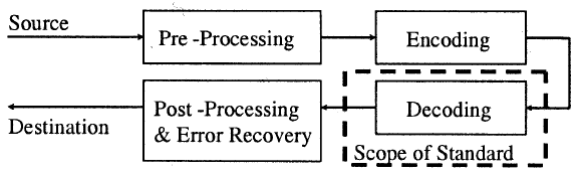

Figure 2.8: Scope of video coding standardization. [2]

8×8 pixels which is the focus in this thesis, in addition to an 8×8 integer DCT[1] transform used in the spatial model of the codec.

2.5 H.264 Intra Prediction

In the H.264/AVC standard, an image frame is divided into macroblocks (MB), each consisting of 16×16 pixels. The macroblocks can be further divided into sub-blocks depending on the block size used (4×4, 8×8 or 16×16) for luminance intra prediction. An appropriate prediction block size is chosen by the encoder for each macroblock, and is dependent on the resulting number of bits in the prediction and the residual. Smaller block sizes give a more accurate prediction which yields a smaller coded residual. If a 4×4 block size is chosen, more bits are needed to code the prediction choices since it will be 16 4×4 sub-blocks in the macroblock. A larger block size on the other hand, will often result in a less accurate prediction, but with fewer bits required to store the prediction choice. Highly textured regions tend to use low blocks size, while more smooth regions use a higher block size.

The predicted sub-blocks are based on neighboring pixels from earlier encoded blocks, typically from the left edge, upper-left edge, upper-edge and upper-right edge as seen in Figure 2.5. There are also a number of different prediction modes available to each intra prediction type that varies with block size and if the block is a luma or a chroma block. Table 2.1 gives an overview over the different intra prediction types possible, and lists how many prediction modes there are for each type. The prediction modes for 16×16 luma and 8×8 chroma are similar, and the 4×4 luma and 8×8 luma use the same prediction modes. It should be noted that for chroma intra prediction, a macroblock consists of one 8×8 block for each chroma component (C_B for blue-difference and C_R for red-difference), where the same prediction mode is always used by both blocks.

[1]Integer DCT is an approximation of the transform shown in Section 2.3.2

Table 2.1: Intra prediction types.

Intra prediction block size	**Number of prediction blocks in a MB**	**Number of possible prediction modes**
16x16 luma	1	4
8x8 luma	4	9
4x4 luma	16	9
8x8 chroma	1	4

Table 2.2: Intra 8×8 prediction modes.

ID	**Name**	**Angle**
0	Vertical (V)	
1	Horizontal (H)	
2	Mean (DC)	
3	Diagonal down-left (DDL)	45°
4	Diagonal down-right (DDR)	45°
5	Vertical-right (VR)	26.6° right of V
6	Horizontal-down (HD)	26.6° below H
7	Vertical-left (VL)	26.6° left of V
8	Horizontal-up (HU)	26.6° up from H

2.5.1 Intra 8×8 Prediction Modes

There are nine prediction modes that are used to predict an 8×8 block from neighboring pixels as shown in Figure 2.9. The modes used in 8×8 intra prediction are the same as for 4×4 intra prediction, and are listed in Table 2.2. The angles of the prediction modes are illustrated in Figure 2.10 where the IDs correspond to the modes in Table 2.2. The vertical and horizontal modes copy the neighboring pixels from their left side and upper side respectively. In the DC mode, all predicted pixels are the average of the neighboring pixels from both the left and upper side. For the diagonal modes (DDL, DDR, VR, HD, VL and HU) are the individual predicted pixels a result of interpolation of two or three neighboring pixels.

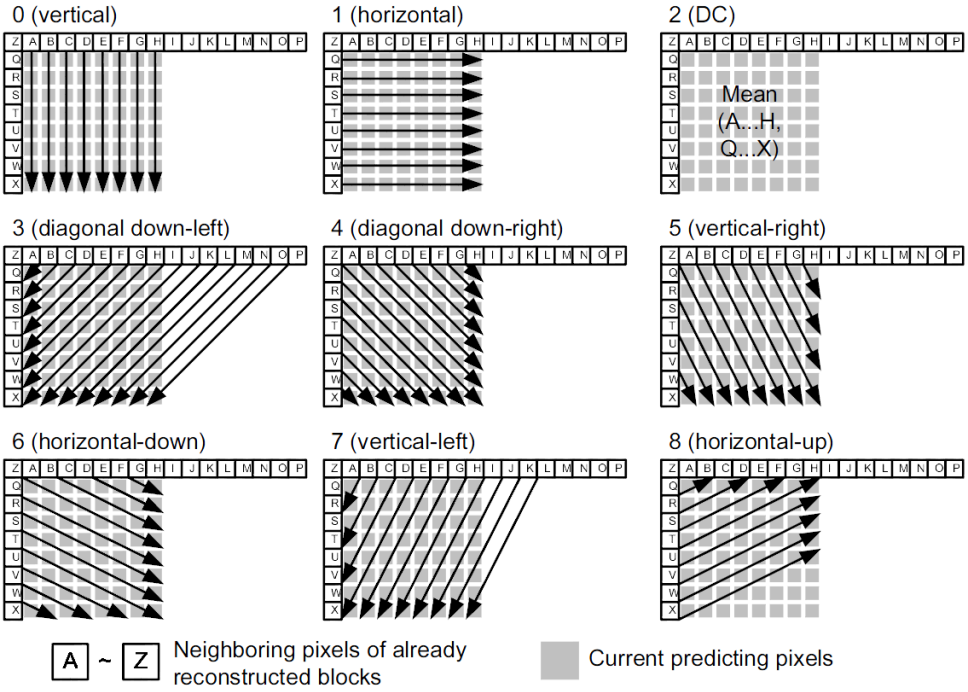

Figure 2.9: Illustration of nine 8x8 prediction modes. [3]

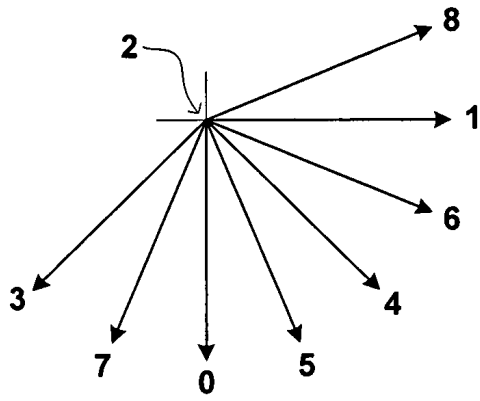

Figure 2.10: Angles of the prediction modes.

Chapter 3
Hardware Architecture

This chapter describes the hardware architecture of an 8×8 luminance intra prediction module. It shows the functionality necessary for the system to work from a system level perspective. The first section will show an overview of the chosen architecture, with the following sections going into more detail of how the system is divided into one inner and one outer module.

3.1 System Overview

The 8×8 intra prediction module is a part of an H.264/AVC encoder as shown in Figure 3.1. Data from the current frame are sent into the intra prediction module,

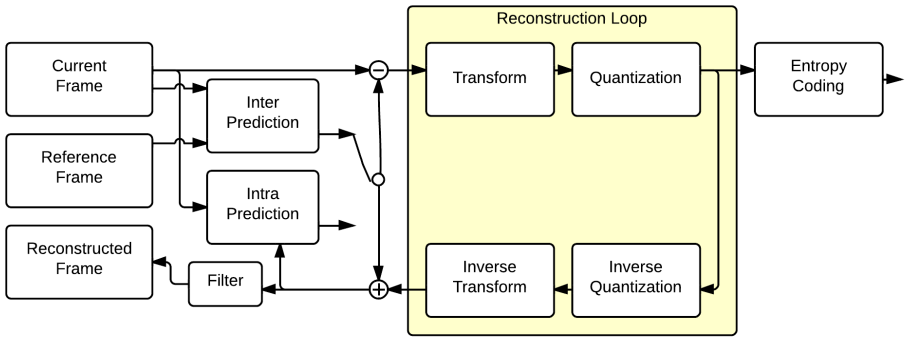

Figure 3.1: H.264/AVC encoder block diagram

and residuals are output before they are subtracted with the current frame data. After being transformed and quantized in a reconstruction loop, the data is fed back into the intra prediction module to be used as a reference frame for the next frame

to be processed. This data is reconstructed residuals. Because the residual data must be reconstructed to be used in the next frame, the intra prediction module is hard to implement while achieving high throughput. The system has to wait for the completion of the reconstruction loop before it can process a new block.

The proposed 8×8 intra prediction module computes predicted pixels for 8×8 blocks, but the input size is 16 pixels and not 64 which would be the required amount of input pixels needed to calculate all pixels in one clock cycle. Figure 3.2 shows the ordering of input blocks within a macroblock, where the system is processing macroblock by macroblock in a raster order[1]. The system is based on an existing design of a 4×4 luma intra prediction module where 16 pixels can be processed in parallel. The same requirement applies in this design, because of how the rest of the H.264/AVC encoder handles input values to the different intra prediction modules.

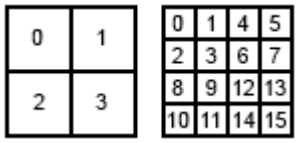

Figure 3.2: Block ordering within one MB, as 8×8 blocks (left) and 4×4 blocks (right).

As an abstraction of functionality, the hardware architecture of the intra prediction system can be seen as two modules. In the rest of this paper, the two modules will be referenced to as an inner and an outer module. The inner module does the intra prediction, and the outer module keeps track of timing and the signals used to compute the predicted pixels and residuals.

3.2 Outer Module Overview

The outer module of the intra prediction design can be seen as an outer layer to the calculation of the predicted pixels. This module gets pixels from the current frame, sends this into an inner module that computes the predicted pixels, and stores information from previous frames to keep track of neighbouring pixels which are used in future frames. Figure 3.3 illustrates how the outer module is connected with the inner prediction module. For consistency, the signal names used in the figure are the same as in the design this system is based on.

There is one clock signal, `clk`, in the design, and one reset signal, `rst`, which is active high. There are 16 input pixel signals for the current frame, `ycbcr[0..15]`, where each pixel is an 8 bit vector. The module utilizes a reference frame consisting

[1]Read row by row from top to bottom, where each row is read from left to right

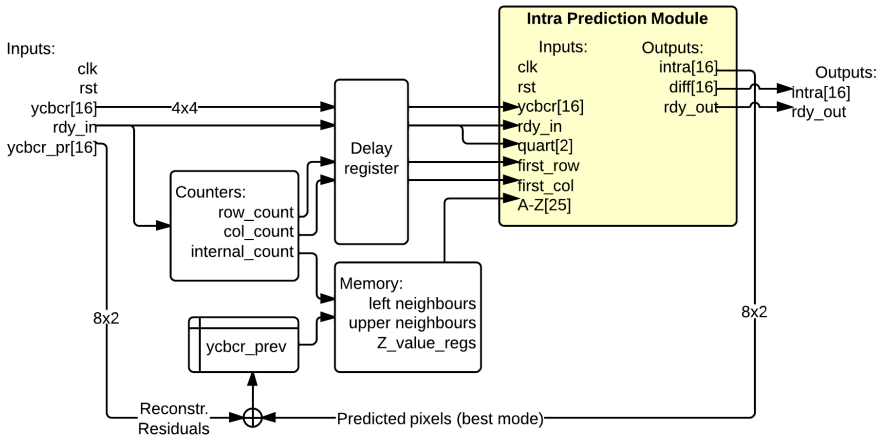

Figure 3.3: Block diagram of the 8×8 intra prediction module.

of 16 previously encoded pixels, `ycbcr_pr[0..15]`. These are reconstructed residuals of 9 bit vectors (8 bits plus a sign bit) which are the difference between input pixels and best mode predicted pixels. They are used to compute neighbouring pixels that are used to make prediction of the current block based on earlier encoded blocks. The output signals from the design are residuals, `intra[0..15]`, that come directly from the inner module and are then sent through a reconstruction loop with an 8×8 integer DCT , quantization (Q), inverse quantization (IQ) and inverse discrete cosine transform (IDCT).

3.2.1 Modelling Reconstruction Loop Behavior

Since the reconstruction loop is not implemented in the system, the correct behavior is modelled by having reconstructed residuals as inputs to the outer module. The reconstructed residuals and the best mode predicted pixels are added together and stored as reconstructed original pixels (labeled `ycbcr_prev`) which are the basis for computation of neighbor pixels. The amount of clock cycles needed for residuals to go through reconstruction is important to know for the outer module. This is because the predicted pixels output from the inner module must be stored in a register, and are read at the same time the reconstructed residuals are ready.

In [8], DCT, Q, IQ and IDCT each takes one clock cycle to complete. This is with 16 pixels per cycle, which is the same as in this design. However, the DCT algorithm is different: this design may use an 8×8 sized transformation matrix,

while the 4×4 intra prediction module uses a 4×4 sized matrix for transformation. A hardware implementation of a 2-D integer 8×8 DCT is seen in [9], where 8 pixels can be transformed in one clock cycle. By assuming that two such modules can be run in parallel, with Q and IQ modules based on [8] and a IDCT module based on [9], 4 clock cycles to complete the reconstruction loop is possible in a best case scenario. The outer module should be implemented with a best guess delay to synchronize the residuals and the best mode prediction, which also should be easy to reconfigure.

3.3 Inner Module Overview

The intra prediction module of the design is tasked with calculating the best prediction mode for an 8×8 block of pixels based on neighbouring pixels, and returning the residual and the predicted pixels of the 8×8 block. A block diagram of the module is illustrated in Figure 3.4. The following sections explain how each of the blocks in the module works.

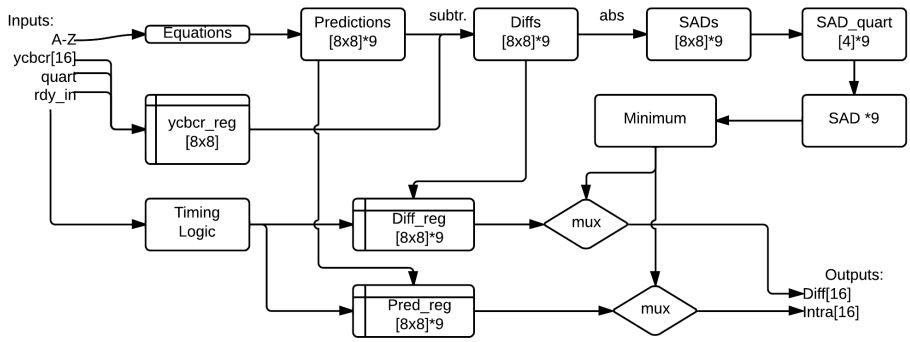

Figure 3.4: Inner prediction module block diagram.

3.3.1 Inputs and Outputs

The inner intra prediction module receives the 25 neighboring pixels of the 8×8 block as inputs from the outer module. They are labelled `A` to `Z`, and their placement in relation to the 8×8 block is illustrated in Figure 3.5.

Other inputs are 16 reference pixels, `ycbcr`, used to calculate the sum of absolute differencess (SADs) that are used to find the best prediction mode. The prediction mode with the lowest SAD is the best mode. Since the number of pixels input to the module is 16 for each clock cycle, the module needs to buffer up and store these until

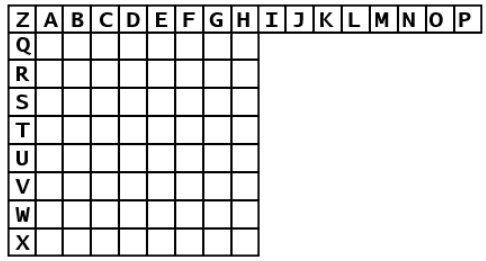

Figure 3.5: Neighboring pixels for an 8×8 block.

it has all 64 pixels required to calculate the SADs, which in turn return the residual and predicted pixels out. The output size is limited to 16 residual and predicted pixels each cycle as well.

The input signal `rdy_in` should be high when a new set of 4×4 blocks are coming in. To fill an 8×8 block with data, four 4×4 blocks are needed. The input signal `quart` is used to describe which 4×4 block that is input to the module (zero to three).

3.3.2 Predictions

For each set of neighboring pixels input to the module, nine 8×8 blocks of predicted pixels will be computed (one for each of the nine different prediction modes). The following equation is used:

$$\text{Pred}_M = (E + R) \ll S, \tag{3.1}$$

where E is the equation based on neighboring pixels, R and S are constants representing the rounding of E and the number of left-shifts respectively. M represents the prediction mode used. There are two kinds of equations used for E: A two-tap filter or a three-tap filter equation. An example of the former is $E = eAB = A + B$ where A and B are the neighboring pixels used to calculate the predicted pixel and `eAB` is the signal name for that particular equation. In this case, $R_{eAB} = 1$ to round the result up to the nearest integer and $S_{eAB} = 1$ is used to divide by two (left shift by one position). This makes the predicted pixel equal to the mean of pixels A and B.

For a three-tap filter equation an example can be $E = eABC = A + 2B + C$. Here, two 2-tap filter equations can be combined. To make eABC, eAB and eBC can be added together. $eAB + eBC = A + 2B + C$ which gives

$$\text{Pred}_M = eAB + eBC + R_{eAB} + R_{eBC} \ll (S_{eAB} + S_{eBC}) = A + 2B + C + 2 \ll 2 \tag{3.2}$$

where both $R_{eAB} = R_{eBC} = 1$ and $S_{eABC} = 1 + 1 = 2$ (divide by four).

A table showing all equations used for each pixel for each prediction mode can be seen in Appendix B.

Corner Cases

For 8×8 blocks that are in the first row or first column in a frame, there is a limited amount of neighboring pixels available. The very first block (first row and first column) has no neighboring pixels, so no predictions can be made. For the rest of the blocks in the first row, only pixels Q~X are available. As a result, only the following prediction modes can be used: Horizontal, horizontal-up, and DC(Q, R, S, T, U, V, W, X). For the first column blocks, only prediction modes vertical, vertical-left, diagonal down-left, and DC(A, B, C, D, E, F, G, H) can be used. This is shown in Table 3.1. The modified DC mode values are calculated based on the eight available pixels.

Table 3.1: Prediction modes in corner cases.

First row blocks	**First column blocks**
DC(Q, R, S, T, U, V, W, X)	DC(A, B, C, D, E, F, G, H)
H	V
HU	DDL
	VL

3.3.3 Differences

The differences are the same as residuals in this case, and are defined by the following equation:

$$\text{Diff}(x, y) = \text{Orig}(x, y) - \text{Pred}_M(x, y) \tag{3.3}$$

The differences will be calculated for all nine prediction modes and for all 64 pixels in each 8×8 block. The calcuated differences should be stored in registers, where the set of differences corresponding to the best prediction mode will be output.

3.3.4 Sum of Absolute Differences

SADs are calculated to find the prediction mode with a minimal error, and are defined as:

$$SAD(\text{Orig}, \text{Pred}_M) = \sum_{x=1}^{8} \sum_{y=1}^{8} |\text{Orig}(x, y) - \text{Pred}_M(x, y)| \tag{3.4}$$

Or equivalently, based on already calculated differences:

$$SAD(\text{Orig}, \text{Pred}_M) = \sum_{x=1}^{8} \sum_{y=1}^{8} |\text{Diff}(x, y)| \tag{3.5}$$

When the nine SADs are computed, the best prediction mode is defined as the one belonging to the minimal SAD. After the minimal SAD is found, the inner prediction module can output the predicted 8×8 block, and the residual corresponding to the chosen prediction mode. This must be done over four clock cycles with the 16 pixel input and output size.

Chapter 4
Implementation

The design is split into two modules implemented in VHDL. The main reason for this is to make testing of the whole system easier, by keeping the intra prediction calculations separated from the part of the intra prediction module that keeps track of neighboring pixels and the timing logic needed to wait for data to come from the reconstruction loop. A focus has been on keeping the maximum frequency of the synthesized design high, in addition to limiting the use of excessive logic elements.

The source code for both implemented modules can be found in a zip-file distributed with this paper. For the inner module, the file *pred.vhd* consists of 5362 lines of code, and the outer module, *intra.vhd*, are 967 lines.

4.1 Inner Module

The inner prediction module were realized in VHDL based on the overview from section 3.3. To maximize the throughput of the module, a pipeline is desired. The ideal solution can manage a new set of 8×8 input blocks every fourth clock cycle. This would create the need of storing intermediate results in registers so that overlapping ongoing calculations are supported. A timing diagram was created, and is shown in Figure 4.1. In this diagram, q1, q2, q3, and q4 (short for quarter) corresponds to the four 4×4 input blocks that represent a full 8×8 input set of original pixels, while s1, s2, and s3 (short for set) corresponds to data connected to three different 8×8 input sets shown in the figure. It is estimated that seven clock cycles are needed from the first 16 input pixels are read to the first set of outputs are ready.

The registers that store predictions and residuals needs to update at the right time. This is implemented by having eleven 1-bit signals (`rdy_in_X` to indicate which clock cycle each set of calculations are on, where `X` is a number between 1 and 11). `rdy_in_1` is set to the input signal `rdy_in` each clock cycle, `rdy_in_2` is set to be equal to `rdy_in_1` the next cycle and so on. This way, all of the registers can be

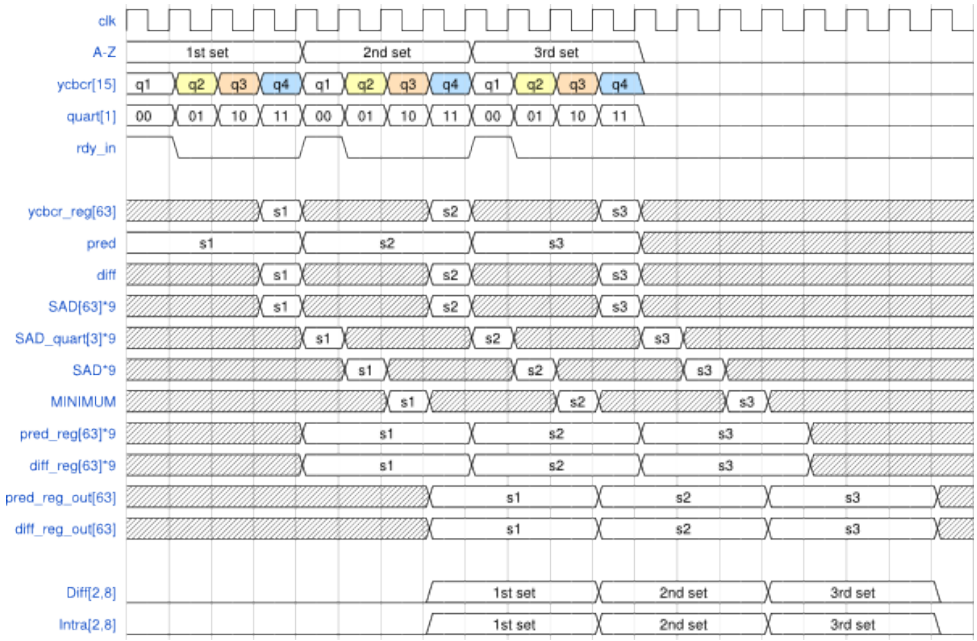

Figure 4.1: Timing diagram for the implemented inner module.

forced to update only at a specific clock cycle count after the first 4×4 block is input to the module.

Making a pipeline work in the inner module proved to be the biggest challenge. Other problems was knowing how much work that could be done each clock cycle without limiting the synthesized max frequency considerably. A preliminary limit was set to 18 8-bit additions in one clock cycle. This was close to the nested levels of logic required by the DC prediction mode: $\mathrm{DC} = A + B + C + D + E + F + G + H + Q + R + S + T + U + V + W + X + 8 \ll 4$. 17 additions and one shift operation. The other prediction mode equations would require fewer levels of logic. 3-tap equations are implemented as two 2-tap equations added together which gives 5 additions in total.

By storing computed values in registers that updates each clock cycle, a work chain was made where the next operation in the chain uses the value stored in the corresponding register. This is an easy way of deciding how much work that should be done each clock cycle. This increases the logic element count, where each one-bit register are synthesized as a D-flip-flop.

The difference (`diff`) signals consist of one 9-bit subtraction per signal, and there

are $9 \cdot 64 = 576$ of them in the implemented design. The absolute value of each of them is modelled with one adder in the worst case, using the two's complement method, where the bits are inverted and added with one if the first bit is 1. The differences and their absolute values are computed in one clock cycle.

Computation of the SADs in one clock cycle might be a problem, since there are 64 8-bit numbers that should be added. To avoid getting a slow synthesized system, SADs were instead calculated over two clock cycles where in the first cycle, four intermediate sums was computed in parallel (named `SAD_quarts`), being 16 8-bit additions. On the next clock cycle these four sums were added to give the final SAD for each prediction mode. The calculation of the nine SADs can be done in parallel.

4.1.1 Finding the Minimal Prediction Mode

To choose the best prediction mode, the lowest of the nine SADs must be found. The lowest one decides which of the 8×8 matrices containing predictions and differences (residuals) are set to the output. This was implemented with four levels of comparators as seen in Figure 4.2, using the same method as in [5]. The nine input values in the figure represents the SADs for each prediction mode. Since the nine SADs are 14 bit numbers, this could be the bottleneck of the inner module.

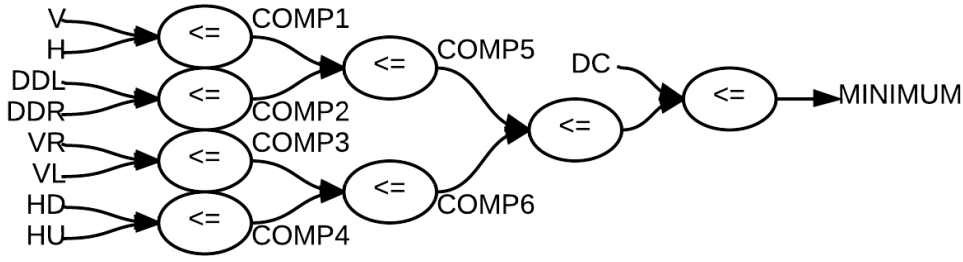

Figure 4.2: Implementation of minimal SAD computation.

4.1.2 Pipeline Implementation

To allow new 4×4 blocks as input every clock cycle, a pipeline was implemented for the inner module. This was done by storing predictions and differences in registers that only updates its value at specific times. The registers are marked as `pred_reg`, `diff_reg`, `pred_reg_out` and `diff_reg_out` in Figure 4.1, where it is shown that the registers updates before a new set of inputs arrive. The update times are controlled by one of the eleven `rdy_in_X` signals that keeps track of how many clock cycles have passed since the last set of inputs arrived. The sizes of `pred_reg` and `diff_reg` are $9 \cdot 64 \cdot 8 = 4608$ bits and $9 \cdot 64 \cdot 9 = 5184$ bits respectively (the `diff` signals have

a sign bit). The two output registers are smaller, since at the time they are updated, the minimal SAD is found. Therefore, only the best mode prediction and difference values are stored. The values in `pred_reg_out` and `diff_reg_out` are then output over four clock cycles.

Output Optimizations

For the output residuals and predictions from the inner prediction module, a choice was made to make them 2×8 matrices rather than 4×4 blocks like the input blocks are. With this approach the first two rows of the complete 8×8 block are output at the first instance, the third and fourth row at the second instance, and so on. There are two main reasons for this: It could decrease the time needed in the reconstruction loop since the integer DCT could begin processing of two complete rows immediately, instead of waiting one extra clock cycle for four rows to be ready. Since the intra prediction module might need to wait while residuals are being processed in the reconstruction loop, reducing the number of clock cycles needed there is important for the throughput of the system. The second main reason lies in how the outer module reads and stores edge pixels used for processing the next data block. Only the predicted pixels on the right side and pixels on the last row will ever be used by the outer module. This corresponds to pixels 7-15 in Figure 4.3. This simplifies the logic in the outer module and allows us to remove all signals that computes prediction pixels 0-6, which in turn yields a lower logic element count in the synthesized module.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

Figure 4.3: Output block matrix.

4.2 Outer Module

The outer module was implemented according to Figure 3.3. This module is not as complex as the inner module, but the implementation was not without challenges. One problem was how to synchronize the signals `rdy_in`, `quart` and the input pixels `ycbcr[0..15]` sent to the inner module. Another challenge was how to implement correct usage of the reconstructed residuals without having a reconstruction loop implemented yet.

To support different video resolutions, the module was implemented with reconfigurable constants, `frame_width` and `frame_height`. This was necessary for the module to be able to detect when the frame boundaries are reached.

4.2.1 Delay Registers

To ensure correctness and robustness of the inner prediction module when it is connected to the outer module, some of the input values (`ycbcr[0..15]`, `quart` and `rdy_in`) cannot come at the same time as the clock signal goes high. If that is the case, the intra prediction module will try to store the difference values, `diff`, into a register before they are computed. Because of this, an extra clock cycle will be needed as a delay. The outer module will calculate and send the input values to the inner prediction module when the clock signal goes high, so this is a necessary fix. A solution is to have a register that delays some of the input values before they are sent to the intra prediction module.

All input signals except clock and reset signals will have to pass through a register that stores the values when the clock signal is high, and outputs the values when the clock signal goes low, a half clock period later. By doing this, all signals that enter the inner prediction module are read at the same time, and the system will behave as expected.

Figure 4.4 illustrates how this kind of register can be synthesized by using two flip-flops for each signal that shall be delayed. The leftmost register reacts on a positive clock edge and the rightmost updates values on negative clock edge. An alternative here was to add this delay into a register inside the intra prediction module, but this could require more work than to implement it into the outer module.

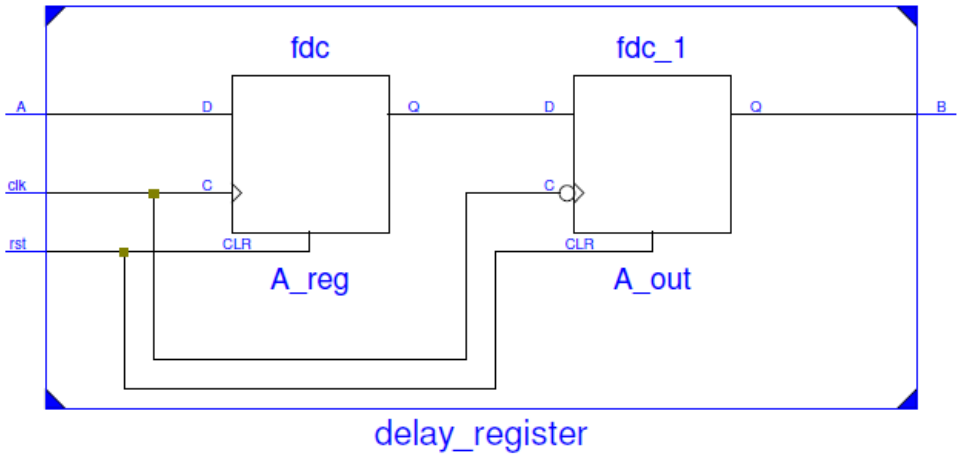

Figure 4.4: Synthesized delay register for a one-bit signal.

4.2.2 Counters

To keep track of which 8×8 block is currently being processed, counters are needed. The implemented design includes three counters: `row_count`, `col_count` and `internal_count`, where the first two stores current row and column respectively (measured in 8×8 blocks), and the latter keeps track of which 8×8 block in a macroblock are being processed. This behavior corresponds to the numbering in Figure 3.2. The counters for row and column enables an easy way to signal `first_row` and `first_col` input signals to the inner prediction module, which controls the behavior in corner cases.

The `internal_count` signal is used to set `next_row` and `next_col` signals that updates the row and column counter as seen in Table 4.1. There is one exception to this rule. That is when the current column is the last one in the video frame as dictated by a constant called `frame_width` and `internal_count` = 3. In this case the `next_col` is set to 0, and `next_row` is incremented.

Table 4.1: Changes to row and column counters based on current internal count.

`internal_count`	`next_row`	`next_col`
0	+0	+1
1	+1	−1
2	+0	+1
3	−1	+1

4.2.3 Storage of Neighbor Pixels

The implementation of neighbor pixel memory is based on the scheme used in [5] and [4] for 4×4 intra prediction. The idea is to store the all previously coded upper and left-side neighbors in memory such that they are available to the next macroblock at any time. An illustration can be seen in Figure 4.5. The storage of neighbors happens in three different processes. One process for the left-side neighbors, upper-side neighbors and upper-left-side neighbor (the `Z` value). All neighbor values (`A-Z`) are updated before a new set of inputs arrives, and are based on the `internal_count`, `next_row` and `next_col` signals. They can be updated before a new set of inputs arrive because the inner module does not use these before the `rdy_in` signal goes high.

Upper Neighbor Values

An array of registers was made to enable the storage of previous-frame pixels for the entire row of the frame. Whenever a set of reconstructed pixels arrive, the bottom

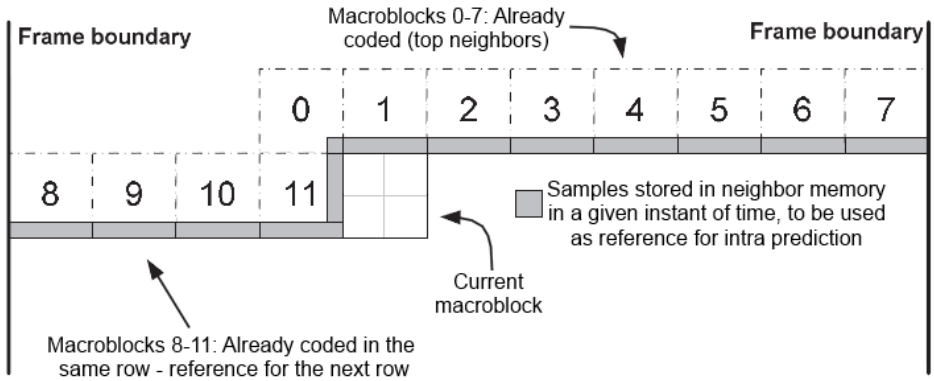

Figure 4.5: Neighbor pixels storage scheme. [4]

row (`ycbcr_prev[8..15]` from the fourth 4×4 block) is stored at the position of the array that corresponds to the current `col_count`.

A process updates neighbor values `A~P`, where the register array is read from positions `next_col` and `next_col`+1. There is one exception to this, which is when the next 8×8 block is the last one in the macroblock (`internal_count`=3). The problem is illustrated in Figure 4.6, where the values `I~P` are not coded yet. These values are then set to be equal to `H`, the right-most available neighbor value. The same problem is evident at the right side frame boundary, and the same solution is chosen.

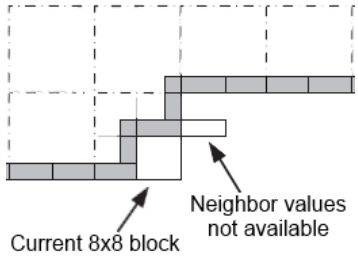

Figure 4.6: Upper neighbors for the last 8×8 block in a MB.

Left Neighbor Values

For the left side neighbor values, a register with room for 16 pixels was created. When a set of reconstructed pixels arrive, the rightmost values (`ycbcr_prev[7]` and `ycbcr_prev[15]`) are stored in either the upper or lower half of this register. It depends on if the *current* 8×8 block is in in the upper or lower half of the macroblock

respectively. The process that updates neighbor values `Q~X`, reads from the upper or the lower half of the register, depending on if the *next* 8×8 block is in the upper or lower half of the MB respectively. This behavior is illustrated in Figure 4.7.

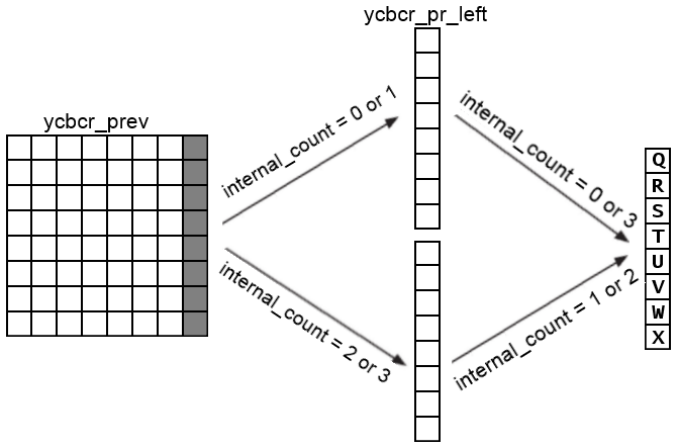

Figure 4.7: Left neighbor storage scheme.

The Z Value

Implementation of the upper-left-side neighbor (`Z`) was done with four 8-bit registers `z0`, `z1`, `z2` and `z3`. They store the values seen in Figure 4.8, with the following criteria:

```
z0 = H when internal_count = 1
z1 = H when internal_count = 0
z2 = X when internal_count = 0
z3 = ycbcr_prev[15] when internal_count = 0 and quart_pr = 3
```

Only for the `z3` register, are current reconstructed pixels used for future `Z`-values. The rest of the registers copy the information from already coded neighbor pixels used in an earlier frame.

The `Z` signal is set to be `z1` when current `internal_count` is 0, `z2` when current `internal_count` is 1, and so on. This way, the correct value is stored for the next 8×8 block. An exception is when the current 8×8 block is the last column and the last 8×8 block in a MB. Then the `Z` value is set to 0.

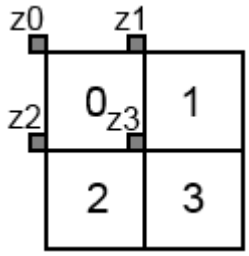

Figure 4.8: Z value register source samples seen from a macroblock.

Chapter 5
Results

This chapter presents key results from the verification stage and synthesis of the design.

5.1 Verification

In order to verify the behavior of the implemented design, a MATLAB model of the 8×8 luma intra prediction module was made. The script takes an image frame as input and computes the best mode predictions and residuals. It also provides a way of checking all internal variables used during the calculation. This was used to verify testbench simulation results of the implemented design. The MATLAB code can be found in the attachment distributed with this paper (*intra8x8.m*). All simulation was done in *Xilinx ISE Simulator (ISim) P.20131013* and the same picture was used as a source of all input values (*lena64.png*).

5.1.1 Simulation Results for the Inner Module

By creating a testbench, the inner prediction module written in VHDL was tested for correctness. To test the basic functionality, a modified MATLAB script was used to generate samples for the testbench (see *inner_intra8x8_test.m* in the attachment). This script creates testbench stimulus for one 8×8 block and ignores if said block is the first row or column. A first test was done for one 8×8 block. All output results was compared to those generated in the MATLAB script. Internal signals were also compared, most notably the SADs for each prediction mode. Some errors in the prediction equations were found and corrected this way.

To test the implemented pipeline, a test for one macroblock was conducted. The modified MATLAB script was used to generate stimulus of four 8×8 blocks from the same picture. Figure 5.1 and 5.2 shows the results from the simulation. Key values presented are the nine SADs (internal signals), output predicted pixels and

output residuals. Output from the MATLAB script for this test set can be seen in Appendix C.1.

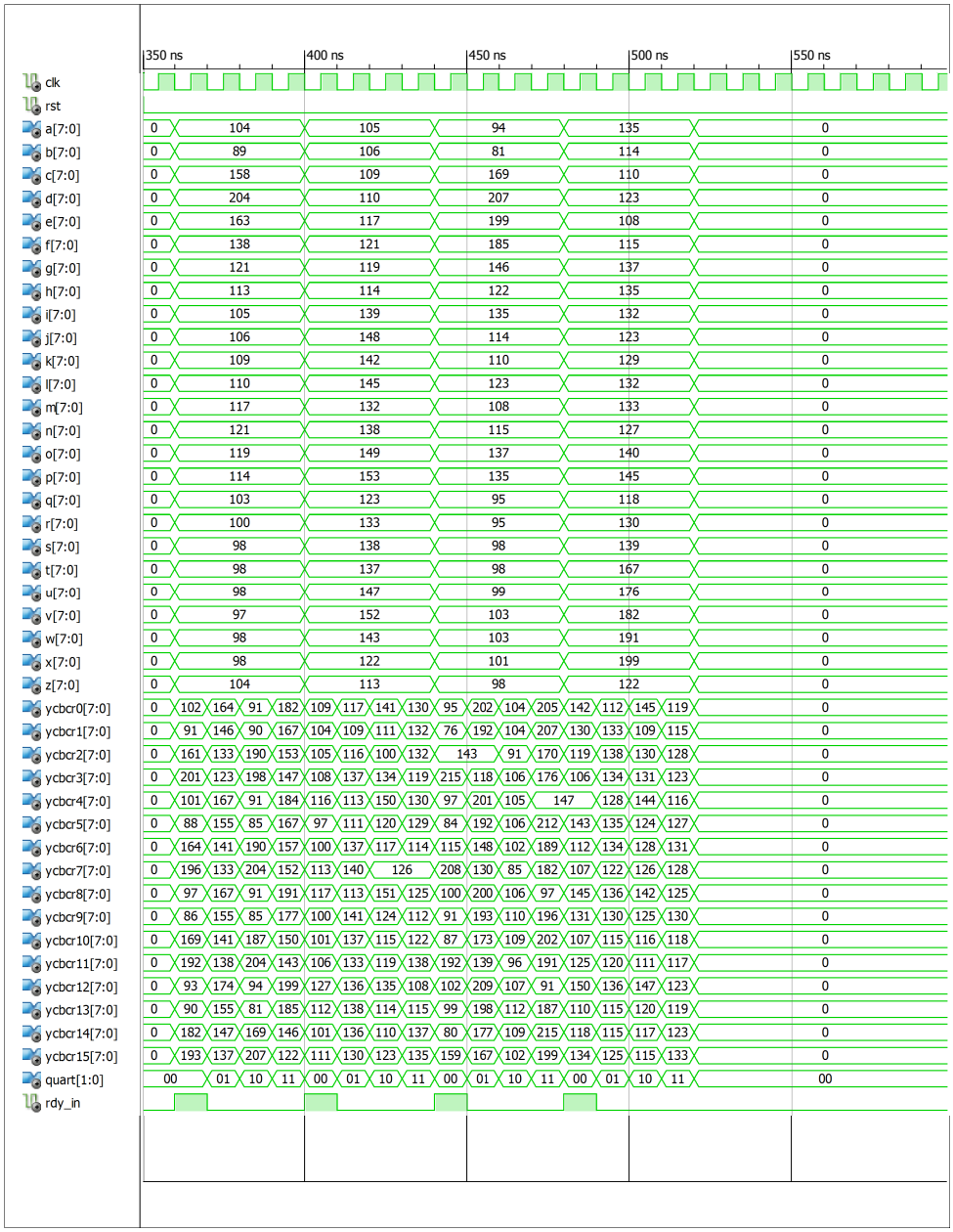

Figure 5.1: Inner module simulation results, input signals.

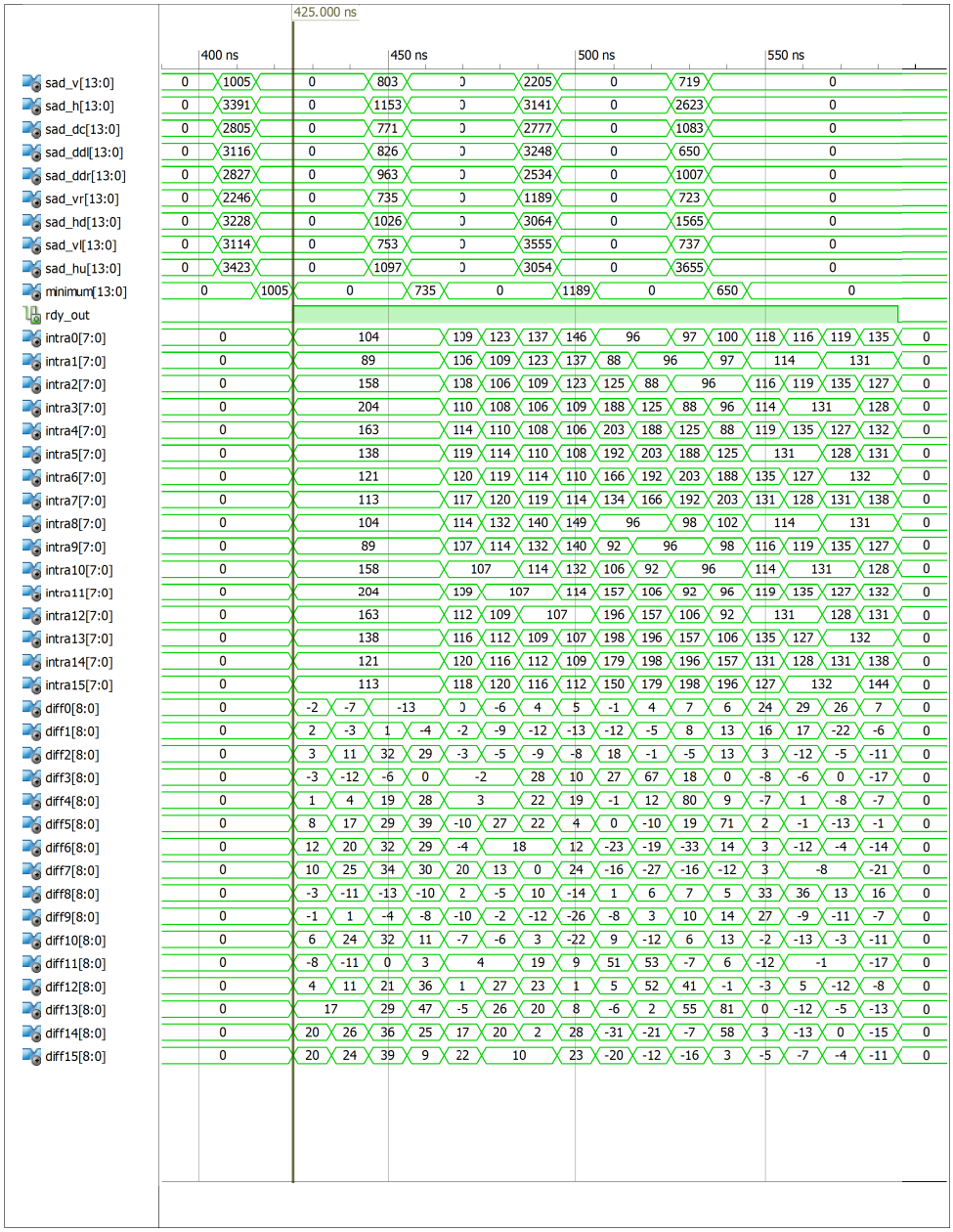

Figure 5.2: Inner module simulation results, internal and output signals.

5.1.2 Simulation Results for the Whole System

Simulation of both the inner and outer module together was done in the same manner as with the inner module alone. For this test, *one* macroblock was extracted from the attached source picture using posistions 18 to 33 for the x-axis, and 29 to 44 for the y-axis. Unlike in the simulation of the inner module, this test verifies correct behavior regarding `first_row` and `first_col`. With four 8×8 blocks forming a macroblock, the first 8×8 block will be first row and first column, the second and third block will be first row and first column respectively, and the last block will be neither. This test shows how the outer module handles storage and sending of the neighbouring pixels `A` to `Z`, in addition to the output residuals which should match the values from the MATLAB script. Key results from the simulation can be seen in Figure 5.3 and 5.4. A graphical presentation of the results generated by the MATLAB script is shown in Figure 5.5. Output from the MATLAB script can be seen in appendix C.2.

From the simulation results of the whole system, it is found that the number of clock cycles needed to process one 8×8 block is 18 cycles. This can be split up into 7 cycles for the inner module to complete processing, 10 cycles before results come back from the modelled reconstruction loop, and finally 1 cycle where the outer module computes neighboring pixels. The number of cycles needed to complete one macroblock is then $18 \cdot 4 = 72$ clock cycles.

5.2 Synthesis

Both the inner module and the outer module are synthesized using *Xilinx ISE Project Navigator (P.20131013)* with the target FPGA being *Kintex 704 XC7K325T FFG900*. Key results from the synthesis are presented in Table 5.1. More details on the synthesis results can be seen in Appendix D, where excerpts of the simulation reports for both modules are shown.

The synthesis of the whole system generated 0 errors, 0 warnings, and 931 info messages where all of them are from the optimization stage. All of these are messages about multiple registers in the inner prediction module being equivalent, and therefore replaces equivalent registers with one register and multiplexers to save device area usage. These registers are mainly from the inner prediction module where internal registers are used for each predicted pixel in a 8×8 matrix for each prediction mode. An example is for the diagonal-down right (DDR) mode where all predicted pixels in a diagonal line along the matrix are equivalent.

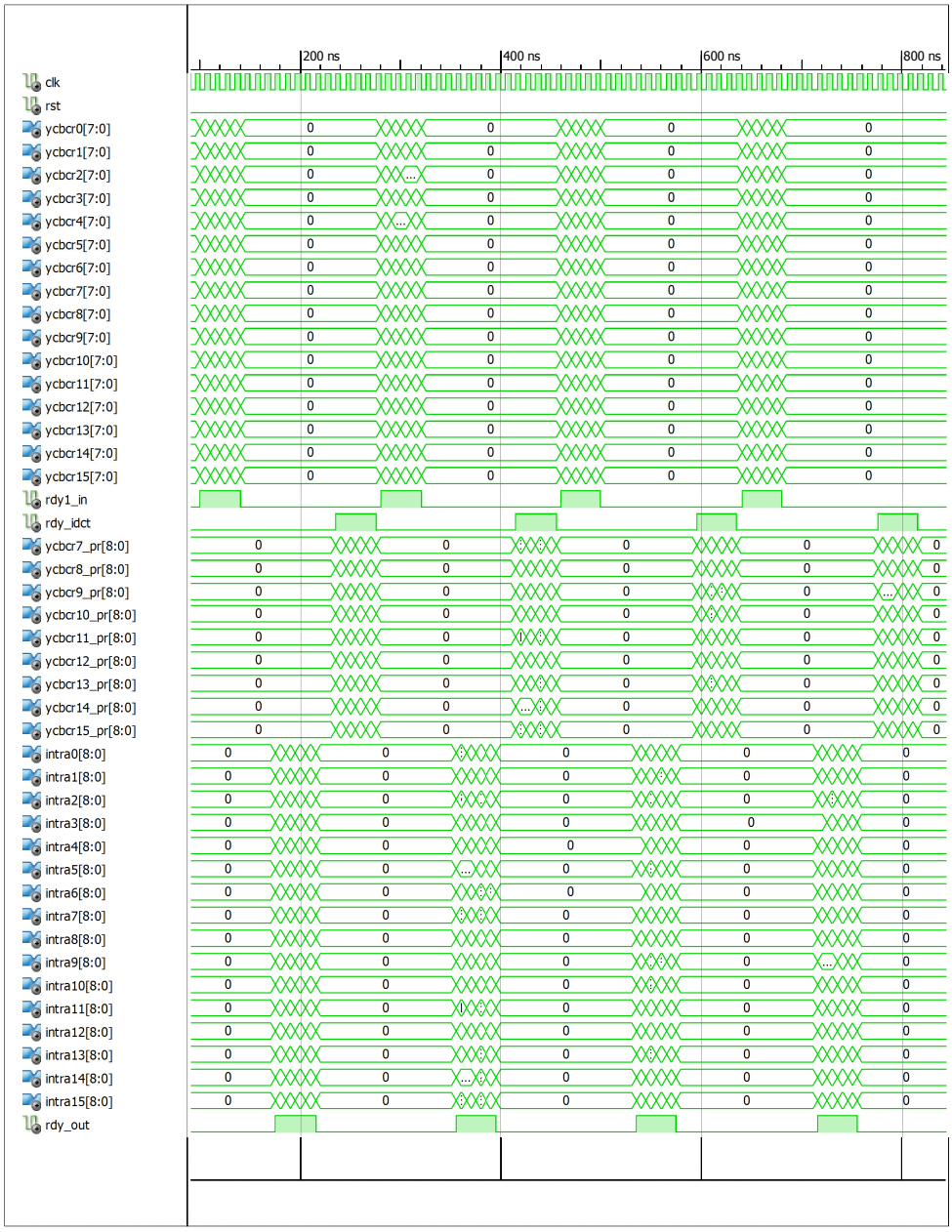

Figure 5.3: Whole system simulation results, outer module signals.

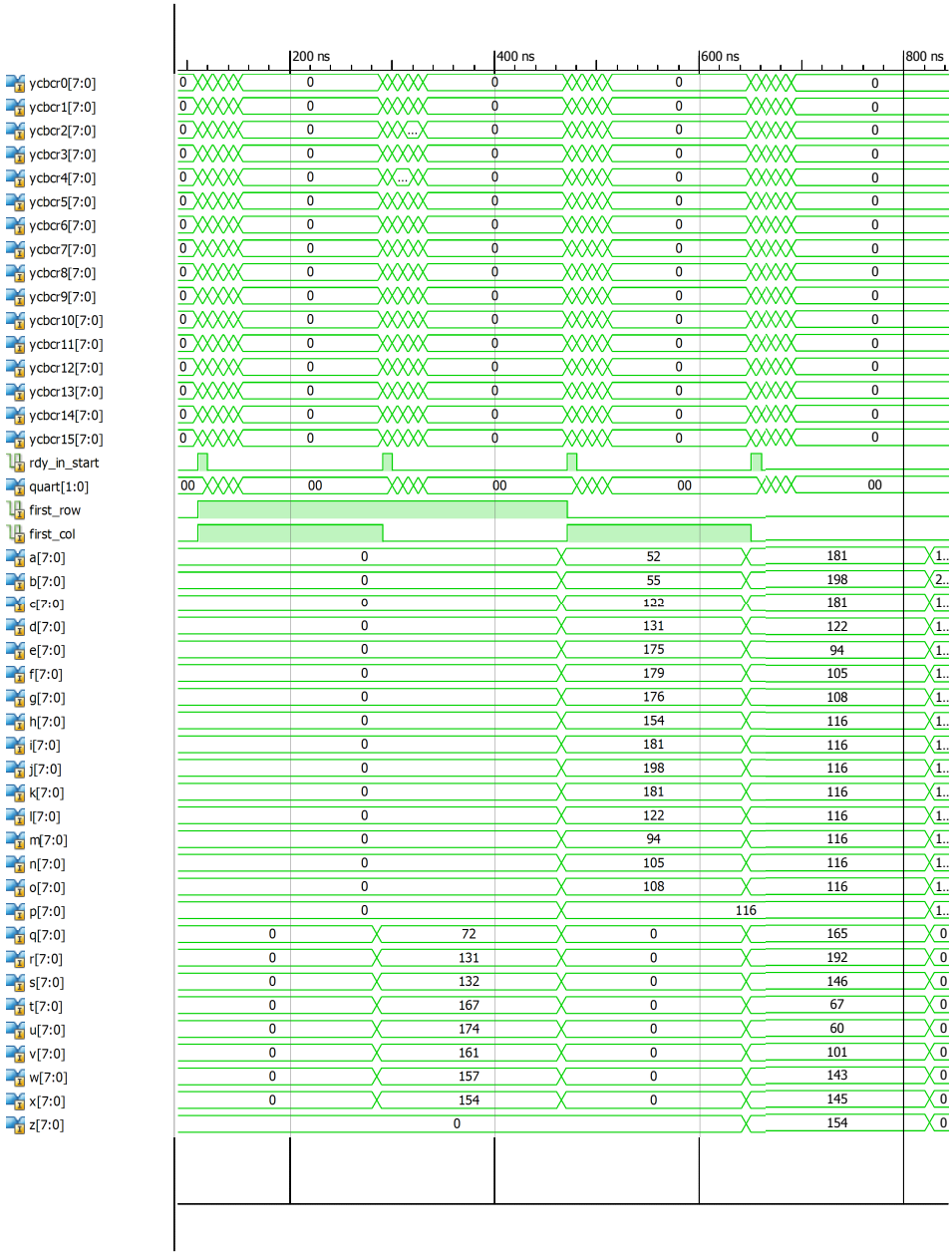

Figure 5.4: Whole system simulation results, inner module signals.

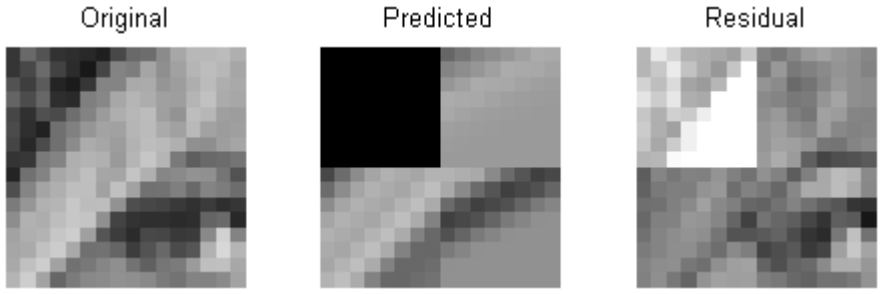

Figure 5.5: Graphical presentation of simulation results.

Table 5.1: Synthesis results from Xilinx ISE.

Module	Slice Registers	Slice LUTs	Max. freq. [MHz]
Inner module	11968	25505	179.62
Whole system	12958	26109	129.34

Chapter 6
Discussion

6.1 Implementation of the Proposed Hardware Architecture

An alternative to using registers to store values could have been done with internal built-in memory on the FPGA in some cases. Many registers used in the design are not updated at every clock cycle, but at predefined times. This applies to neighbor storage memory in particular, since these registers are not frequently read from or written to. By using built-in memory, the number of flip-flops used in the design are decreased, which affects the logic element usage of the entire H.264/AVC transcoder in a positive way.

6.1.1 Pipelining

The implemented outer module does not utilize the pipeline that is implemented in the inner module. Plans to support new input data sets while the outer module is otherwise idle were made, but not implemented due to lack of time. Another problem was the assumption of which order the input macroblocks comes in. With the macroblocks coming in a raster order, pipelining of the whole system is impossible since the next macroblock is dependent on data from a former macroblock to begin processing.

Pipelining is only possible if the order of macroblocks are changed. This is decided by a module in the H.264/AVC encoder outside the scope of this design. Whether this is applicable or not is therefore not known at the time of writing this. Some considerations to how pipelining can be done are shown in Section 7.1.

6.1.2 The Bottleneck of the System

More time could have been used to increase max clock frequency of the whole system. Since there is a relatively big difference between the inner module and the whole system when synthesized (179.62 MHz and 129.34 MHz respectively), there might

be a easy way to speed up the system. This notion is further increased by the fact that the inner module is the most complex module of the two, both in terms of size and functionality. By looking at the synthesis report for the whole system, the critical path can be found. It is found to be between signals `ycbcr_upper_64` and `Prediction/SAD63_DC_reg_7`, which is from the register that stores neighbor pixels `A` to `P` in the outer module and the stored absolute value of the difference between predicted pixels in the DC mode and the original input pixels. This is 24 levels of logic.

The bottleneck can be a result of not storing the neighboring pixel values in registers before they are sent into the inner module. In the inner module, the last 4×4 input block are not stored in a register before it is read directly to be used to compute predictions, differences and their absolute value. This is a long chain of operations, and by addressing this, the synthesized max frequency could be higher for the whole system. An attempt to fix this was done, where the neighbor pixels were stored in a delay register before they are output to the inner module. This is similar to what is done to the input original pixels. The estimated clock frequency with this approach was considerably lower, and an easy fix was not found.

Even though the synthesis estimate of the maximum clock frequency is above the initial target from the specifications (100 MHz), the bottleneck indicates a weakness in the implementation. A fix to this would require some changes to how the inner module reads its input values. Ideally, all input values are read and stored in registers before they are used in the module. By doing this for the inner module, the delay registers in the outer module would not be needed because the robustness the delay registers provide would be a part of the inner module already. This was however the chosen method of implementation because it took considerably less time to implement after the flaw was found.

6.2 Verification

In the verification process of the implemented 8×8 intra prediction module, only pixels from one picture was used to generate stimulus for testbenches. The tests completed in the verification stage were small scale tests to verify the functionality of the design. By not conducting more thorough testing, all cases leading to errors might not be covered. An example of this is the computed `Z`-values, which in some cases are stored and then used in a frame in the next row of macroblocks. The small scale test of the outer module does only feature one macroblock (four 8×8 blocks), and use only one of the four `Z`-values stored.

A larger scale test was not included because of the number of signals that would need to be verified in the design is very high. With the current MATLAB script

and testbench it is not feasible to compare values from all important signals for many input blocks of data. Given more time for testing, the testbench for the whole system and the MATLAB script could be expanded to be more automatic in verification. If the MATLAB script could create the stimulus in VHDL format, and the testbench could compare results with a MATLAB-generated solution text file, then the verification stage could include a much larger data set, preferably from a video sequence. In addition, formal verification techniques could have been used to further verify the functionality. This could reveal interactions between the inner and outer module that might be hard to find by simulation.

6.3 Performance

To compute the throughput of the system, the following equation can be used:

$$\text{Throughput} = \frac{\text{Maximum frequency} \cdot \text{Number of Samples}}{\text{Number of cycles}} \tag{6.1}$$

By using the synthesis estimate of the maximum frequency, 128.34 MHz, from Table 5.1, and the number of clock cycles per macroblock (72), it is found that the throughput of the proposed intra prediction module is 456.32 Mpixels/s. The maximum frequency can also be used to figure out how well the system can perform with various video resolutions and frame rates. The minimum frequency needed by the system manage real time processing of a video with a given resolution and frame rate is defined as:

$$\text{Minimum frequency} = \frac{\text{Resolution} \cdot \text{Frame rate} \cdot \text{Number of cycles}}{\text{Number of samples}} \tag{6.2}$$

Using a frame rate of 30 frames per second, Table 6.3 shows the minimum required system frequency needed for various video format standards.

Table 6.1: Minimum frequency requirements at 30 frames per second.

Standard	**Resolution**	**Min. freq. [MHz]**
720p (HD)	1280×720	7.8
1080p (Full HD)	1920×1080	17.5
2160p (4K UHD)	3840×2160	70.0
4320p (8K UHD)	7680×4320	279.9

From these numbers, it is seen that the implemented 8×8 luma intra prediction module can process 720p, 1080p and 2160p video in real-time. Note that these numbers are based on an estimated number of clock cycles needed to process one macroblock. In addition, the maximum frequency obtained from the synthesis is an estimate as well. As mentioned, the reconstruction loop is not implemented, so it is

currently not known if the reconstruction loop can be modelled with 6 clock cycles to complete.

6.4 Scalability

The device area usage is presented with absolute numbers in Section 5.2, showing how many LUTs and registers the implemented intra prediction module consumes on the target FPGA. It could be interesting to see how the design scales compared to the basis of this design, the 4×4 intra prediction module from [5]. By synthesizing the given source code, logic element usage is compared using the same metrics, and are presented in Table 6.2.

Table 6.2: Comparation of logic elements usage between 4×4 and 8×8 intra prediction modules.

Metric	4×4 [5]	8×8 (**proposed**)	**Difference** [%]
Slice registers	7364	14440	196
Slice LUTs	11575	21964	190

Compared with its basis design, the proposed 8×8 luma intra prediction module is close to twice the size in logic element usage. This is achieved even with registers in the inner module often being four times larger (4×4 stores 16 pixels and 8×8 stores 64 pixels). The comparison indicates sub-linear growth with regard to block size, but there are some differences between the two modules that makes such a direct comparison difficult. One such difference is the frame width and the frame height the designs are synthesized with, which makes differently sized outer modules. There are also differences in functionality between the two outer modules. The 4×4 module includes a pipeline that allows four 4×4 blocks to be processing at different stages at the same time. The 8×8 module does not include logic for more than one input 8×8 block.

Chapter 7
Conclusion

An 8×8 luminance intra prediction module is proposed as a part of an MPEG-2 to H.264/AVC transcoder. The block-based prediction module exploits similarities between neighboring image samples in the same video frame to reduce redundancy and compress the video. For a hardware implementation of an intra prediction module, the throughput is limited by a high data dependency between input data blocks.

High throughout is achieved by allowing 16 image samples to be processed at the same time in 4×4 blocks. The design is implemented in VHDL, and its functionality is verified by simulation. The proposed hardware architecture is synthesized on a *Kintex-7* FPGA with a maximum clock frequency of 129.34 MHz, which translates into a throughput of 456.32 Mpixels/s. This makes the 8×8 luminance intra prediction module able to encode 2160p (4k UHD) video in real-time given a frame rate of 30 fps.

7.1 Future Work

The throughput of the implementation can theoretically be improved. From how the whole H.264/AVC transcoder is designed, the 8×8 luminance intra prediction module must process macroblocks in a raster order. There is a possibility of extending the implemented design to process multiple macroblocks at the same time, as much of the processing time in the intra prediction module is used to wait for results to come back from a reconstruction loop. Currently, only the prediction part of the implemented design supports such a pipeline. Given that the rest of the H.264/AVC encoder can reorder the macroblocks arriving to the 8×8 intra prediction module such that a new row of macroblocks can be processed before an earlier row is done, the throughput can increase considerably.

References

[1] I. E. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2 ed., August 2010.

[2] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 560–576, July 2003.

[3] C. Tzu-Der, C. Yi-Hau, T. Chen-Han, C. Yu-Jen, and C. Liang-Gee, "Algorithm and architecture design for intra prediction in H.264/AVC High profile," *SPE Journal*, vol. 9, no. 4, pp. 391–402, 2007.

[4] C. Diniz, A. Susin, and S. Bampi, "FPGA design of H.264/AVC intra-frame prediction architecture for high resolution video encoding," in *Programmable Logic (SPL), 2012 VIII Southern Conference on*, pp. 1–6, March 2012.

[5] M. Orlandic and K. Svarstad, "A low complexity H.264/AVC 4x4 intra prediction architecture with macroblock/block reordering," in *Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on*, pp. 1–6, Dec 2013.

[6] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC)." Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.

[7] G. J. Sullivan, P. N. Topiwala, and A. Luthra, "The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions," 2004.

[8] M. Orlandic and K. Svarstad, "An area efficient hardware architecture design for H.264/AVC intra prediction reconstruction path based on partial reconfiguration," in *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2013 IEEE 16th International Symposium on*, pp. 86–91, April 2013.

[9] T. L. da Silva, C. M. Diniz, J. A. Vortmann, L. V. Agostini, A. A. Susin, and S. Bampi, "A pipelined 8x8 2-D forward DCT hardware architecture for H.264/AVC High profile encoder," *PSIVT*, pp. 5–15, 2007.

Appendix A
Abbreviations

AVC Advanced Video Coding

chroma chrominance

codec coder/decoder pair

DC mean (prediction mode)

DCT discrete cosine transform

DDL diagonal-down left (prediction mode)

DDR diagonal-down right (prediction mode)

FPGA field-programmable gate array

H horizontal (prediction mode)

H.264 H.264/MPEG-4 Part 10 AVC (Advanced Video Coding)

HD horizontal down (prediction mode)

HU horizontal up (prediction mode)

HVS human visual system

IDCT inverse discrete cosine transform

IQ inverse quantization

ISE Integrated Software Environment

ISO International Standards Organisation

ITU International Telecommunication Union

JPEG Joint Photographic Experts Group

luma luminance

LUT look-up-table

MB macroblock

MPEG-2 Motion Picture Experts Group 2

Q quantization

RGB red/green/blue (color space)

SAD sum of absolute differences

V vertical (prediction mode)

VHDL VHSIC Hardware Description Language

VL vertical left (prediction mode)

VR vertical right (prediction mode)

YCrCb luminance/red chrominance/blue chrominance (color space)

Appendix B
Prediction Mode Equations

As stated in section 3.3.2, the equation for a predicted pixel is $Pred = (E + R) \ll S$. Table B.1 shows all intra prediction equations, E, for all 64 pixels in an 8x8 block, for the nine prediction modes. To get the complete equations, values for the rounding addition, R, and the shift operation, S must be added. For equations with three taps, $R = 2$ and $S = 2$, and for equation with 2 taps, $R = 1$ and $S = 1$. The lower case letters `a` to `p` represents the pixels in a 4×4 matrix as shown in figure B.1. Equations in **bold text** in the table means it is the first ocurrence of that particular equation, E.

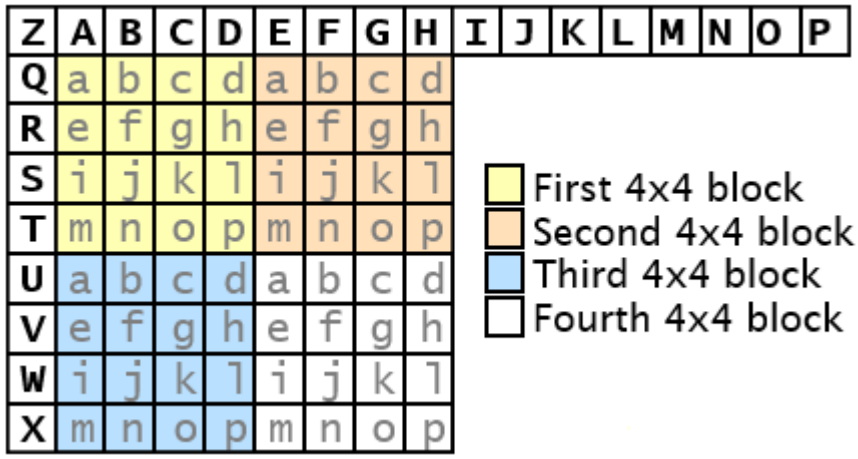

Figure B.1: Visual explanation of prediction mode equations.

The Ω symbol means $E = A + B + C + D + E + F + G + H + Q + R + S + T + U + V + W + X$ which is the case for all predicted pixels in the DC prediction mode. Here, $S = 4$ (divide by 16) and $R = 8$.

Table B.1: Intra prediction equations for the nine 8×8 prediction modes.

First 4×4 block									
	V	**H**	**DC**	**DDL**	**DDR**	**VR**	**VL**	**HD**	**HU**
a	A	Q	Ω	**A+2B+C**	**Q+2Z+A**	**Z+A**	A+B	**Q+Z**	R+Q
b	B	Q	Ω	**B+2C+D**	**Z+2A+B**	**A+B**	B+C	Q+2Z+A	S+2R+Q
c	C	Q	Ω	**C+2D+E**	A+2B+C	**B+C**	C+D	Z+2A+B	S+R
d	D	Q	Ω	**D+2E+F**	B+2C+D	**C+D**	**D+E**	A+2B+C	T+2S+R
e	A	R	Ω	B+2C+D	**R+2Q+Z**	Q+2Z+A	A+2B+C	**R+Q**	S+R
f	B	R	Ω	C+2D+E	Q+2Z+A	Z+2A+B	B+2C+D	R+2Q+Z	T+2S+R
g	C	R	Ω	D+2E+F	Z+2A+B	A+2B+C	C+2D+E	Q+Z	T+S
h	D	R	Ω	**E+2F+G**	A+2B+C	B+2C+D	D+2E+F	Q+2Z+A	**U+2T+S**
i	A	S	Ω	C+2D+E	**S+2R+Q**	R+2Q+Z	B+C	**S+R**	T+S
j	B	S	Ω	D+2E+F	R+2Q+Z	Z+A	C+D	S+2R+Q	U+2T+S
k	C	S	Ω	E+2F+G	Q+2Z+A	A+B	D+E	R+Q	**U+T**
l	D	S	Ω	**F+2G+H**	Z+2A+B	B+C	**E+F**	R+2Q+Z	**V+2U+T**
m	A	T	Ω	D+2E+F	**T+2S+R**	S+2R+Q	B+2C+D	**T+S**	U+T
n	B	T	Ω	E+2F+G	S+2R+Q	Q+2Z+A	C+2D+E	T+2S+R	V+2U+T
o	C	T	Ω	F+2G+H	R+2Q+Z	Z+2A+B	D+2E+F	S+R	**V+U**
p	D	T	Ω	**G+2H+I**	Q+2Z+A	A+2B+C	E+2F+G	S+2R+Q	**W+2V+U**

Second 4×4 block									
	V	**H**	**DC**	**DDL**	**DDR**	**VR**	**VL**	**HD**	**HU**
a	E	Q	Ω	E+2F+G	C+2D+E	D+E	E+F	B+2C+D	T+S
b	F	Q	Ω	F+2G+H	D+2E+F	E+F	F+G	C+2D+E	U+2T+S
c	G	Q	Ω	G+2H+I	E+2F+G	**F+G**	G+H	D+2E+F	U+T
d	H	Q	Ω	**H+2I+J**	F+2G+H	**G+H**	**H+I**	E+2F+G	V+2U+T
e	E	R	Ω	F+2G+H	B+2C+D	C+2D+E	E+2F+G	Z+2A+B	U+T
f	F	R	Ω	G+2H+I	C+2D+E	D+2E+F	F+2G+H	A+2B+C	V+2U+T
g	G	R	Ω	H+2I+J	D+2E+F	E+2F+G	G+2H+I	B+2C+D	V+U
h	H	R	Ω	**I+2J+K**	E+2F+G	F+2G+H	H+2I+J	C+2D+E	W+2V+U
i	E	S	Ω	G+2H+I	A+2B+C	C+D	F+G	Q+Z	V+U
j	F	S	Ω	H+2I+J	B+2C+D	D+E	G+H	Q+2Z+A	W+2V+U
k	G	S	Ω	I+2J+K	C+2D+E	E+F	H+I	Z+2A+B	**W+V**
l	H	S	Ω	**J+2K+L**	D+2E+F	F+G	**I+J**	A+2B+C	**X+2W+V**
m	E	T	Ω	H+2I+J	Z+2A+B	B+2C+D	F+2G+H	R+Q	W+V
n	F	T	Ω	I+2J+K	A+2B+C	C+2D+E	G+2H+I	R+2Q+Z	X+2W+V
o	G	T	Ω	J+2K+L	B+2C+D	D+2E+F	H+2I+J	Q+Z	**X+W**
p	H	T	Ω	**K+2L+M**	C+2D+E	E+2F+G	I+2J+K	Q+2Z+A	**3X+W**

Third 4 × 4 block									
	V	**H**	**DC**	**DDL**	**DDR**	**VR**	**VL**	**HD**	**HU**
a	A	U	Ω	E+2F+G	U+2T+S	T+2S+R	C+D	U+T	V+U
b	B	U	Ω	F+2G+H	T+2S+R	R+2Q+Z	D+E	U+2T+S	W+2V+U
c	C	U	Ω	G+2H+I	S+2R+Q	Z+A	E+F	T+S	W+V
d	D	U	Ω	H+2I+J	R+2Q+Z	A+B	F+G	T+2S+R	X+2W+V
e	A	V	Ω	F+2G+H	V+2U+T	U+2T+S	C+2D+E	V+U	W+V
f	B	V	Ω	G+2H+I	U+2T+S	S+2R+Q	D+2E+F	V+2U+T	X+2W+V
g	C	V	Ω	H+2I+J	T+2S+R	Q+2Z+A	E+2F+G	U+T	X+W
h	D	V	Ω	I+2J+K	S+2R+Q	Z+2A+B	F+2G+H	U+2T+S	3X+W
i	A	W	Ω	G+2H+I	W+2V+U	V+2U+T	D+E	W+V	X+W
j	B	W	Ω	H+2I+J	V+2U+T	T+2S+R	E+F	W+2V+U	3X+W
k	C	W	Ω	I+2J+K	U+2T+S	R+2Q+Z	F+G	V+U	X
l	D	W	Ω	J+2K+L	T+2S+R	Z+A	G+H	V+2U+T	X
m	A	X	Ω	H+2I+J	X+2W+V	W+2V+U	D+2E+F	X+W	X
n	B	X	Ω	I+2J+K	W+2V+U	U+2T+S	E+2F+G	X+2W+V	X
o	C	X	Ω	J+2K+L	V+2U+T	S+2R+Q	F+2G+H	W+V	X
p	D	X	Ω	K+2L+M	U+2T+S	Q+2Z+A	G+2H+I	W+2V+U	X

Fourth 4 × 4 block									
	V	**H**	**DC**	**DDL**	**DDR**	**VR**	**VL**	**HD**	**HU**
a	E	U	Ω	I+2J+K	Q+2Z+A	B+C	G+H	S+R	X+W
b	F	U	Ω	J+2K+L	Z+2A+B	C+D	H+I	S+2R+Q	3X+W
c	G	U	Ω	K+2L+M	A+2B+C	D+E	I+J	R+Q	X
d	H	U	Ω	L+2M+N	B+2C+D	E+F	**J+K**	R+2Q+Z	X
e	E	V	Ω	J+2K+L	R+2Q+Z	A+2B+C	G+2H+I	T+S	X
f	F	V	Ω	K+2L+M	Q+2Z+A	B+2C+D	H+2I+J	T+2S+R	X
g	G	V	Ω	**L+2M+N**	Z+2A+B	C+2D+E	I+2J+K	S+R	X
h	H	V	Ω	**M+2N+O**	A+2B+C	D+2E+F	J+2K+L	S+2R+Q	X
i	E	W	Ω	K+2L+M	S+2R+Q	A+B	H+I	U+T	X
j	F	W	Ω	L+2M+N	R+2Q+Z	B+C	I+J	U+2T+S	X
k	G	W	Ω	M+2N+O	Q+2Z+A	C+D	J+K	T+S	X
l	H	W	Ω	**N+2O+P**	Z+2A+B	D+E	**K+L**	T+2S+R	X
m	E	X	Ω	L+2M+N	T+2S+R	Z+2A+B	H+2I+J	V+U	X
n	F	X	Ω	M+2N+O	S+2R+Q	A+2B+C	I+2J+K	V+2U+T	X
o	G	X	Ω	N+2O+P	R+2Q+Z	B+2C+D	J+2K+L	U+T	X
p	H	X	Ω	**O+3P**	Q+2Z+A	C+2D+E	K+2L+M	U+2T+S	X

Appendix C

Output from MATLAB Scripts

C.1 Stimulus for Inner Prediction Module

Generated from file *inner_intra8x8_test.m*: 16×16 pixels from *lena64.png*. Data should be compared to simulation results found in Figure 5.1 and 5.2.

```
Inputs and outputs for:
An 8x8 frame starting at position 2,2
A-Z:
  Columns 1 through 8

    104     89    158    204    163    138    121    113

  Columns 9 through 16

    105    106    109    110    117    121    119    114

  Columns 17 through 24

    103    100     98     98     98     97     98     98

  Column 25

    104

Input 8x8
    102     91    161    201    164    146    133    123
    101     88    164    196    167    155    141    133
     97     86    169    192    167    155    141    138
     93     90    182    193    174    155    147    137
```

```
     91     90    190    198    182    167    153    147
     91     85    190    204    184    167    157    152
     91     85    187    204    191    177    150    143
     94     81    169    207    199    185    146    122

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:3391  1005  2773  3116  2827  2246  3228  3114  3423
Predicted pixels:
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113
    104     89    158    204    163    138    121    113

Differences/Residuals:
     -2      2      3     -3      1      8     12     10
     -3     -1      6     -8      4     17     20     20
     -7     -3     11    -12      4     17     20     25
    -11      1     24    -11     11     17     26     24
    -13      1     32     -6     19     29     32     34
    -13     -4     32      0     21     29     36     39
    -13     -4     29      0     28     39     29     30
    -10     -8     11      3     36     47     25      9


Inputs and outputs for:
An 8x8 frame starting at position 2,10
A-Z:
  Columns 1 through 8

    105    106    109    110    117    121    119    114

  Columns 9 through 16

    139    148    142    145    132    138    149    153

  Columns 17 through 24
```

```
    123    133    138    137    147    152    143    122

  Column 25

    113

Input 8x8
    109    104    105    108    117    109    116    137
    116     97    100    113    113    111    137    140
    117    100    101    106    113    141    137    133
    127    112    101    111    136    138    136    130
    141    111    100    134    130    132    132    119
    150    120    117    126    130    129    114    126
    151    124    115    119    125    112    122    138
    135    114    110    123    108    115    137    135

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:1153   803   771   826   963   735  1026   753  1097
Predicted pixels:
    109    106    108    110    114    119    120    117
    114    107    107    109    112    116    120    118
    123    109    106    108    110    114    119    120
    132    114    107    107    109    112    116    120
    137    123    109    106    108    110    114    119
    140    132    114    107    107    109    112    116
    146    137    123    109    106    108    110    114
    149    140    132    114    107    107    109    112

Differences/Residuals:
      0     -2     -3     -2      3    -10     -4     20
      2    -10     -7      4      1     -5     17     22
     -6     -9     -5     -2      3     27     18     13
     -5     -2     -6      4     27     26     20     10
      4    -12     -9     28     22     22     18      0
     10    -12      3     19     23     20      2     10
      5    -13     -8     10     19      4     12     24
    -14    -26    -22      9      1      8     28     23


Inputs and outputs for:
An 8x8 frame starting at position 10,2
```

```
A-Z:
  Columns 1 through 8

     94     81    169    207    199    185    146    122

  Columns 9 through 16

    135    114    110    123    108    115    137    135

  Columns 17 through 24

     95     95     98     98     99    103    103    101

  Column 25

     98

Input 8x8
     95     76    143    215    202    192    143    118
     97     84    115    208    201    192    148    130
    100     91     87    192    200    193    173    139
    102     99     80    159    209    198    177    167
    104    104     91    106    205    207    170    176
    105    106    102     85    147    212    189    182
    106    110    109     96     97    196    202    191
    107    112    109    102     91    187    215    199

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:3141  2205  2773  3248  2534  1189  3064  3555  3054
Predicted pixels:
     96     88    125    188    203    192    166    134
     96     92    106    157    196    198    179    150
     96     96     88    125    188    203    192    166
     96     96     92    106    157    196    198    179
     97     96     96     88    125    188    203    192
     98     96     96     92    106    157    196    198
    100     97     96     96     88    125    188    203
    102     98     96     96     92    106    157    196

Differences/Residuals:
     -1    -12     18     27     -1      0    -23    -16
```

```
      1     -8      9     51      5     -6    -31    -20
      4     -5     -1     67     12    -10    -19    -27
      6      3    -12     53     52      2    -21    -12
      7      8     -5     18     80     19    -33    -16
      7     10      6     -7     41     55     -7    -16
      6     13     13      0      9     71     14    -12
      5     14     13      6     -1     81     58      3


An 8x8 frame starting at position 10,10
A-Z:
  Columns 1 through 18

    135    114    110    123    108    115    137    135

    132    123    129    132    133    127    140    145

    118    130

  Columns 19 through 25

    139    167    176    182    191    199    122

Input 8x8
    142    130    119    106    112    133    138    134
    147    143    112    107    128    135    134    122
    145    131    107    125    136    130    115    120
    150    110    118    134    136    115    115    125
    145    109    130    131    119    115    128    123
    144    124    128    126    116    127    131    128
    142    125    116    111    125    130    118    117
    147    120    117    115    123    119    123    133

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:2623   719  1083   650  1007   723  1565   737  3655
Predicted pixels:
    118    114    116    114    119    131    135    131
    114    116    114    119    131    135    131    127
    116    114    119    131    135    131    127    128
    114    119    131    135    131    127    128    132
    119    131    135    131    127    128    132    131
```

```
    131    135    131    127    128    132    131    132
    135    131    127    128    132    131    132    138
    131    127    128    132    131    132    138    144

Differences/Residuals:
     24     16      3     -8     -7      2      3      3
     33     27     -2    -12     -3      0      3     -5
     29     17    -12     -6      1     -1    -12     -8
     36     -9    -13     -1      5    -12    -13     -7
     26    -22     -5      0     -8    -13     -4     -8
     13    -11     -3     -1    -12     -5      0     -4
      7     -6    -11    -17     -7     -1    -14    -21
     16     -7    -11    -17     -8    -13    -15    -11
```

C.2 Stimulus for Whole System Test

Generated from file *intra8x8.m*: 16×16 pixels from *lena64.png*. Data should be compared to simulation results found in Figure 5.3 and 5.4.

```
Inputs and outputs for:
An 8x8 frame starting at position 1,1
A-Z:
  Columns 1 through 10

     0     0     0     0     0     0     0     0     0     0

  Columns 11 through 20

     0     0     0     0     0     0     0     0     0     0

  Columns 21 through 25

     0     0     0     0     0

Input 8x8
    54   105    83    50    46    40    31    72
    53    72   107    66    44    24    69   131
    82    62    91    36    27    54   132   132
   100    76   103    46    42   127   135   167
    70    48    80    74   110   131   155   174
    75    47    33   112   128   150   163   161
```

```
     69     40     75    112    139    172    166    157
     52     55    122    131    175    179    176    154

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:0  0  0  0  0  0  0  0  0
Predicted pixels:
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0

Differences/Residuals:
     54    105     83     50     46     40     31     72
     53     72    107     66     44     24     69    131
     82     62     91     36     27     54    132    132
    100     76    103     46     42    127    135    167
     70     48     80     74    110    131    155    174
     75     47     33    112    128    150    163    161
     69     40     75    112    139    172    166    157
     52     55    122    131    175    179    176    154

Inputs and outputs for:
An 8x8 frame starting at position 1,9
A-Z:
  Columns 1 through 10

      0      0      0      0      0      0      0      0      0      0

  Columns 11 through 20

      0      0      0      0      0      0     72    131    132    167

  Columns 21 through 25

    174    161    157    154      0

Input 8x8
```

```
    105    109    139    167    184    184    188    175
    125    165    143    161    181    189    183    166
    170    171    149    157    174    187    174    169
    181    173    160    158    180    193    171    167
    183    188    159    147    189    183    159    157
    181    187    170    161    185    162    156    159
    178    189    185    175    144    142    158    165
    181    198    181    122     94    105    108    116

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:1993  16384   7967  16384  16384  16384  16384  16384   1150
Predicted pixels:
    102    117    132    141    150    160    171    169
    132    141    150    160    171    169    168    163
    150    160    171    169    168    163    159    157
    171    169    168    163    159    157    156    155
    168    163    159    157    156    155    154    154
    159    157    156    155    154    154    154    154
    156    155    154    154    154    154    154    154
    154    154    154    154    154    154    154    154

Differences/Residuals:
      3     -8      7     26     34     24     17      6
     -7     24     -7      1     10     20     15      3
     20     11    -22    -12      6     24     15     12
     10      4     -8     -5     21     36     15     12
     15     25      0    -10     33     28      5      3
     22     30     14      6     31      8      2      5
     22     34     31     21    -10    -12      4     11
     27     44     27    -32    -60    -49    -46    -38

Inputs and outputs for:
An 8x8 frame starting at position 9,1
A-Z:
  Columns 1 through 10

     52     55    122    131    175    179    176    154    181    198

  Columns 11 through 20

    181    122     94    105    108    116      0      0      0      0
```

```
  Columns 21 through 25

      0      0      0      0      0

Input 8x8
     36    113    131    167    177    191    166    165
     78    133    164    175    190    190    161    192
    123    150    177    181    199    188    189    146
    141    177    173    189    199    196    144     67
    162    175    179    198    205    160     85     60
    166    170    193    204    169    112    114    101
    169    192    205    162    113    120    134    143
    180    205    163    102    123    136    139    145

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:16384   3408   6816   1001  16384  16384  16384   1874  16384
Predicted pixels:
     71    108    140    165    177    171    166    179
    108    140    165    177    171    166    179    190
    140    165    177    171    166    179    190    171
    165    177    171    166    179    190    171    130
    177    171    166    179    190    171    130    104
    171    166    179    190    171    130    104    103
    166    179    190    171    130    104    103    109
    179    190    171    130    104    103    109    114

Differences/Residuals:
    -35      5     -9      2      0     20      0    -14
    -30     -7     -1     -2     19     24    -18      2
    -17    -15      0     10     33      9     -1    -25
    -24      0      2     23     20      6    -27    -63
    -15      4     13     19     15    -11    -45    -44
     -5      4     14     14     -2    -18     10     -2
      3     13     15     -9    -17     16     31     34
      1     15     -8    -28     19     33     30     31

Inputs and outputs for:
An 8x8 frame starting at position 9,9
A-Z:
  Columns 1 through 10
```

```
    181    198    181    122     94    105    108    116    116    116

  Columns 11 through 20

    116    116    116    116    116    116    165    192    146     67

  Columns 21 through 25

     60    101    143    145    154

Input 8x8
    196    176    134    138    147    144    117     82
    162    113    114    127    105    132    124    109
     78     71     67     57     49     59     47     54
     51     47     47     43     46    100    106     40
     43     72     70     57     59    137    209    131
     61     83    104     75     85    177    216    181
    115     96    117    123    155    191    192    168
    143    137    122    115    131    130    138    163

     Hor,  Ver,  DC,   DDL,   DDR,  VR,  HD,   VL,   HU
SADs:2792  3776  2722  2301  3329  3879  3508  2606  2277
Predicted pixels:
    179    174    169    138    107     85     64     72
    169    138    107     85     64     72     81    101
    107     85     64     72     81    101    122    133
     64     72     81    101    122    133    144    145
     81    101    122    133    144    145    145    145
    122    133    144    145    145    145    145    145
    144    145    145    145    145    145    145    145
    145    145    145    145    145    145    145    145

Differences/Residuals:
     17      2    -35      0     40     59     53     10
     -7    -25      7     42     41     60     43      8
    -29    -14      3    -15    -32    -42    -75    -79
    -13    -25    -34    -58    -76    -33    -38   -105
    -38    -29    -52    -76    -85     -8     64    -14
    -61    -50    -40    -70    -60     32     71     36
    -29    -49    -28    -22     10     46     47     23
```

```
     -2     -8    -23    -30    -14    -15     -7     18

>>
```

Appendix D
Synthesis Reports

D.1 Excerpt from Whole System Synthesis Report

```
Release 14.7 - xst P.20131013 (nt64)
================================================================================
*                      Synthesis Options Summary                               *
================================================================================
---- Source Parameters
Input File Name                    : "intrawrap.prj"
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "intrawrap"
Output Format                      : NGC
Target Device                      : xc7k325t-2-ffg900

---- Source Options
Top Module Name                    : intrawrap
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : LUT
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Shift Register Extraction          : YES
ROM Style                          : Auto
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Shift Register Minimum Size        : 2
```

```
Use DSP Block                       : Auto
Automatic Register Balancing        : No

---- Target Options
LUT Combining                       : Auto
Reduce Control Sets                 : Auto
Add IO Buffers                      : YES
Global Maximum Fanout               : 100000
Add Generic Clock Buffer(BUFG)      : 32
Register Duplication                : YES
Optimize Instantiated Primitives    : NO
Use Clock Enable                    : Auto
Use Synchronous Set                 : Auto
Use Synchronous Reset               : Auto
Pack IO Registers into IOBs         : Auto
Equivalent register Removal         : YES

---- General Options
Optimization Goal                   : Speed
Optimization Effort                 : 1
Power Reduction                     : NO
Keep Hierarchy                      : No
Netlist Hierarchy                   : As_Optimized
RTL Output                          : Yes
Global Optimization                 : AllClockNets
Read Cores                          : YES
Write Timing Constraints            : NO
Cross Clock Analysis                : NO
Hierarchy Separator                 : /
Bus Delimiter                       : <>
Case Specifier                      : Maintain
Slice Utilization Ratio             : 100
BRAM Utilization Ratio              : 100
DSP48 Utilization Ratio             : 100
Auto BRAM Packing                   : NO
Slice Utilization Ratio Delta       : 5

=========================================================================
HDL Synthesis Report

Macro Statistics
```

```
# Adders/Subtractors                                   : 1830
 1-bit adder                                           : 1
 10-bit adder                                          : 27
 12-bit adder                                          : 557
 14-bit adder                                          : 27
 2-bit adder                                           : 6
 2-bit subtractor                                      : 2
 8-bit adder                                           : 585
 9-bit adder                                           : 57
 9-bit subtractor                                      : 568
# Registers                                            : 1783
 1-bit register                                        : 22
 12-bit register                                       : 36
 128-bit register                                      : 1
 14-bit register                                       : 11
 2-bit register                                        : 7
 64-bit register                                       : 2
 8-bit register                                        : 1064
 9-bit register                                        : 640
# Comparators                                          : 19
 1-bit comparator greater                              : 2
 14-bit comparator equal                               : 9
 14-bit comparator lessequal                           : 8
# Multiplexers                                         : 2236
 1-bit 2-to-1 multiplexer                              : 5
 1-bit 4-to-1 multiplexer                              : 2
 12-bit 2-to-1 multiplexer                             : 38
 128-bit 2-to-1 multiplexer                            : 1
 128-bit 4-to-1 multiplexer                            : 1
 14-bit 2-to-1 multiplexer                             : 8
 2-bit 2-to-1 multiplexer                              : 4
 64-bit 2-to-1 multiplexer                             : 1
 64-bit 3-to-1 multiplexer                             : 3
 8-bit 2-to-1 multiplexer                              : 964
 8-bit 4-to-1 multiplexer                              : 1
 9-bit 2-to-1 multiplexer                              : 1208

=========================================================================

=========================================================================
*                      Advanced HDL Synthesis                           *
```

```
=========================================================================

INFO:Xst:2261 - The FF/Latch <V_reg_40_0> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_0> <V_reg_24_0> <V_reg_56_0>
INFO:Xst:2261 - The FF/Latch <V_reg_40_1> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_1> <V_reg_24_1> <V_reg_56_1>
INFO:Xst:2261 - The FF/Latch <V_reg_40_2> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_2> <V_reg_24_2> <V_reg_56_2>
INFO:Xst:2261 - The FF/Latch <V_reg_40_3> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_3> <V_reg_24_3> <V_reg_56_3>
INFO:Xst:2261 - The FF/Latch <V_reg_40_4> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_4> <V_reg_24_4> <V_reg_56_4>
INFO:Xst:2261 - The FF/Latch <V_reg_40_5> in Unit <Prediction> is
equivalent to the following 3 FFs/Latches, which will be removed :
<V_reg_8_5> <V_reg_24_5> <V_reg_56_5>
...


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 1220
 1-bit adder                                           : 1
 1-bit subtractor                                      : 1
 10-bit adder                                          : 23
 10-bit adder carry in                                 : 2
 12-bit adder                                          : 2
 2-bit adder                                           : 4
 2-bit subtractor                                      : 1
 8-bit adder                                           : 585
 9-bit adder                                           : 9
 9-bit adder carry in                                  : 24
 9-bit subtractor                                      : 568
# Adder Trees                                          : 46
 12-bit / 16-inputs adder tree                         : 37
```

```
 14-bit / 4-inputs adder tree                          : 9
# Counters                                             : 2
 2-bit up counter                                      : 2
# Registers                                            : 15146
 Flip-Flops                                            : 15146
# Comparators                                          : 19
 1-bit comparator greater                              : 2
 14-bit comparator equal                               : 9
 14-bit comparator lessequal                           : 8
# Multiplexers                                         : 2247
 1-bit 2-to-1 multiplexer                              : 19
 1-bit 4-to-1 multiplexer                              : 2
 12-bit 2-to-1 multiplexer                             : 38
 128-bit 2-to-1 multiplexer                            : 1
 128-bit 4-to-1 multiplexer                            : 1
 14-bit 2-to-1 multiplexer                             : 7
 2-bit 2-to-1 multiplexer                              : 2
 64-bit 2-to-1 multiplexer                             : 1
 64-bit 3-to-1 multiplexer                             : 3
 8-bit 2-to-1 multiplexer                              : 964
 8-bit 4-to-1 multiplexer                              : 1
 9-bit 2-to-1 multiplexer                              : 1208

=========================================================================
Final Register Report

Macro Statistics
# Registers                                            : 12810
 Flip-Flops                                            : 12810
# Shift Registers                                      : 72
 5-bit shift register                                  : 72

=========================================================================

=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : intrawrap.ngc

Primitive and Black Box Usage:
```

```
-----------------------------
# BELS                              : 44029
#      GND                          : 1
#      INV                          : 6
#      LUT1                         : 7
#      LUT2                         : 7259
#      LUT3                         : 7488
#      LUT4                         : 2430
#      LUT5                         : 2770
#      LUT6                         : 6077
#      MUXCY                        : 8657
#      MUXF7                        : 25
#      VCC                          : 1
#      XORCY                        : 9308
# FlipFlops/Latches                 : 12958
#      FDC                          : 5347
#      FDC_1                        : 3
#      FDCE                         : 7400
#      FDCE_1                       : 130
#      FDE                          : 72
#      FDPE                         : 6
# Shift Registers                   : 72
#      SRLC16E                      : 72
# Clock Buffers                     : 2
#      BUFG                         : 1
#      BUFGP                        : 1
# IO Buffers                        : 357
#      IBUF                         : 212
#      OBUF                         : 145

Device utilization summary:
---------------------------

Selected Device : 7k325tffg900-2


Slice Logic Utilization:
 Number of Slice Registers:           12958  out of  407600     3%
 Number of Slice LUTs:                26109  out of  203800    12%
    Number used as Logic:             26037  out of  203800    12%
    Number used as Memory:               72  out of  64000     0%
```

```
       Number used as SRL:               72

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  27420
   Number with an unused Flip Flop:   14462  out of  27420    52%
   Number with an unused LUT:          1311  out of  27420     4%
   Number of fully used LUT-FF pairs: 11647  out of  27420    42%
   Number of unique control sets:        28

IO Utilization:
 Number of IOs:                         358
 Number of bonded IOBs:                 358  out of    500    71%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                2  out of     32     6%

=========================================================================
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | IBUF+BUFG              | 13030 |
-----------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 7.731ns (Maximum Frequency: 129.341MHz)
   Minimum input arrival time before clock: 2.100ns
```

```
   Maximum output required time after clock: 1.844ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 7.731ns (frequency: 129.341MHz)
  Total number of paths / destination ports: 1115108936 / 20226
-------------------------------------------------------------------------
Delay:               7.731ns (Levels of Logic = 24)
  Source:            ycbcr_upper_64 (FF)
  Destination:       Prediction/SAD63_DC_reg_7 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: ycbcr_upper_64 to Prediction/SAD63_DC_reg_7
                              Gate     Net
    Cell:in->out fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDCE:C->Q       14   0.236   0.422  ycbcr_upper_64 (ycbcr_upper_64)
     LUT2:I1->O       1   0.043   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:S->O       1   0.238   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:CI->O      1   0.014   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:CI->O      1   0.014   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:CI->O      1   0.014   0.000  Prediction/ADDERTREE_INTERNAL...
     XORCY:CI->O      2   0.262   0.355  Prediction/ADDERTREE_INTERNAL...
     LUT3:I2->O       1   0.043   0.350  Prediction/ADDERTREE_INTERNAL...
     LUT4:I3->O       1   0.043   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:S->O       1   0.238   0.000  Prediction/ADDERTREE_INTERNAL...
     XORCY:CI->O      1   0.262   0.405  Prediction/ADDERTREE_INTERNAL...
     LUT2:I0->O       1   0.043   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:S->O       1   0.238   0.000  Prediction/ADDERTREE_INTERNAL...
     XORCY:CI->O      2   0.262   0.355  Prediction/ADDERTREE_INTERNAL...
     LUT3:I2->O       1   0.043   0.350  Prediction/ADDERTREE_INTERNAL...
     LUT4:I3->O       1   0.043   0.000  Prediction/ADDERTREE_INTERNAL...
     MUXCY:S->O       1   0.238   0.000  Prediction/ADDERTREE_INTERNAL...
     XORCY:CI->O      7   0.262   0.439  Prediction/ADDERTREE_INTERNAL...
     LUT4:I2->O       1   0.043   0.613  Prediction/Mmux_DC7_SW1 (N44)
```

```
    LUT6:I0->O      65   0.043   0.486  Prediction/Mmux_DC7 (Predicti...
    LUT2:I1->O       1   0.043   0.000  Prediction/Msub_GND_7_o_GND_7...
    MUXCY:S->O       1   0.238   0.000  Prediction/Msub_GND_7_o_GND_7...
    XORCY:CI->O      1   0.262   0.350  Prediction/Msub_GND_7_o_GND_7...
    LUT3:I2->O       2   0.043   0.355  Prediction/Mmux_Diff36_DC81...
    LUT6:I5->O       1   0.043   0.000  Prediction/Mmux_SAD36_DC81...
    FDC:D               -0.000          Prediction/SAD36_DC_reg_7
    ----------------------------------------
    Total                7.731ns (3.250ns logic, 4.482ns route)
                                 (42.0% logic, 58.0% route)
Cross Clock Domains Report:
--------------------------

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    7.731|    3.343|    1.135|         |
---------------+---------+---------+---------+---------+

=========================================================================

Total REAL time to Xst completion: 196.00 secs
Total CPU time to Xst completion: 196.23 secs

Total memory usage is 782368 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :  913 (   0 filtered)
```

D.2 Excerpt from Inner Prediction Module Synthesis Report

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : intrapred.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 43335
#      GND                         : 1
#      LUT1                        : 7
#      LUT2                        : 7118
#      LUT3                        : 7461
#      LUT4                        : 2384
#      LUT5                        : 2794
#      LUT6                        : 5741
#      MUXCY                       : 8585
#      MUXF7                       : 16
#      VCC                         : 1
#      XORCY                       : 9227
# FlipFlops/Latches                : 11968
#      FDC                         : 5140
#      FDCE                        : 6828
# Clock Buffers                    : 2
#      BUFGP                       : 2
# IO Buffers                       : 550
#      IBUF                        : 333
#      OBUF                        : 217

Device utilization summary:
---------------------------

Selected Device : 7k325tffg900-2


Slice Logic Utilization:
 Number of Slice Registers:           11968  out of  407600     2%
 Number of Slice LUTs:                25505  out of  203800    12%
```

```
    Number used as Logic:              25505  out of  203800    12%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  26802
   Number with an unused Flip Flop:   14834  out of  26802    55%
   Number with an unused LUT:          1297  out of  26802     4%
   Number of fully used LUT-FF pairs: 10671  out of  26802    39%
   Number of unique control sets:         8

IO Utilization:
 Number of IOs:                         552
 Number of bonded IOBs:                 552  out of    500   110% (*)

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                2  out of     32     6%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used

================================================================================
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | BUFGP                  | 11968 |
-----------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -2
```

```
   Minimum period: 5.567ns (Maximum Frequency: 179.618MHz)
   Minimum input arrival time before clock: 8.213ns
   Maximum output required time after clock: 1.844ns
   Maximum combinational path delay: No path found

-->

Total memory usage is 776224 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    1 (   0 filtered)
Number of infos    :  488 (   0 filtered)
```