On the Low Power Design of DCT and IDCT for Low Bit Rate Video Codecs

Nathaniel August

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. Dong S. Ha, Chairman
Dr. J. R. Armstrong
Dr. F. G. Gray

April 2001
Blacksburg, VA

Keywords: DCT, IDCT, H.263, Low power

On the Low Power Design of DCT and IDCT for Low Bit Rate Video Codecs

Nathaniel August
Dr. Dong S. Ha, Chairman
Bradley Department of Electrical and Computer Engineering

(ABSTRACT)

Wireless video systems have applications in cellular videophones, surveillance systems, and mobile patrols. The design of a wireless video system must consider two important constraints: low bit rate and low power dissipation. The ITU-T H.263 video codec standard is suitable for low bit rate wireless video systems, however it is computationally intensive. Some of the most computationally intensive operations in H.263 are the Discrete Cosine Transform (DCT) and the Inverse Discrete Cosine Transform (IDCT), which perform spatial compression and decompression of the data. In an ASIC implementation of H.263, the high computational complexity of the DCT and IDCT leads to high power dissipation of the blocks. Low power design of the DCT and IDCT is essential in a portable wireless video system.

This paper examines low power design techniques for DCT and IDCT circuits applicable for low bit rate wireless video systems. Five low power techniques are applied to baseline reference DCT and IDCT circuits. The techniques include skipping low energy DCT input, skipping all-zero IDCT input, low precision constant multipliers, clock gating, and a low transition data path. Gate-level simulations characterize the effectiveness of each technique. The combination of all techniques reduces average power dissipation by 95% over the baseline reference DCT and IDCT blocks.

Acknowledgements

I first want to thank Dr. Dong S. Ha for initiating an interesting research project and for providing excellent support and guidance throughout the research, design, testing, and writing. He has made my graduate experience worthwhile and enjoyable both in and out of the lab. I would also like to extend my appreciation to Dr. James R. Armstrong and to Dr. F. G. Gray for serving on my advisory committee.

Next, I would like to thank my fellow researchers in the Virginia Tech VLSI for Telecommunications (VTVT) laboratory. It has been a great pleasure working with them for the past two years. I would like to thank Suk Won Kim, Jos Sulistyo, Hyung Jun Kim, Jina Kim, Hyung Jin Lee, and Steve Richmond for all their support and help. I wish them all great success in their future academic and professional lives.

I am also beholden to the Bradley Department of Engineering for providing support in the form of an assistantship. I also greatly appreciate the support of NeoReach Inc. and the extraordinary opportunity to work for a start-up corporation.

Finally, thanks to everyone who has supported me outside of my research endeavors. I give special thanks to my parents, Mary Kay and John August. Their love and support play a crucial role in everything I do. In fact, I would like to dedicate this to my mom. I would also like to thank my friends and roommates in Blacksburg for making my free time enjoyable. Cheers to Mici, Aaron, Pat, Alec, Colin, and Kevin.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

For the increasing number of portable wireless devices, a key design constraint is power dissipation. Limited battery life constrains portable devices to low power dissipation; advances in battery life do not grow as fast as the density and the operating frequency of ASICs [1]. The ever-growing circuit densities and operating frequencies of ASICs only result in greater power dissipation. Moreover, low power designs are more environmentally friendly. Furthermore, since technology sizes continue to shrink, area and speed constraints become less important.

One such portable device that must meet low power design constraints is a wireless video system. For wireless video systems, the relatively small capacity of the wireless channel constrains the system to a low bit rate. The low bit rate makes speed constraints less important, and shrinking technology sizes make area constraints less important; so we focus on low power design of a wireless video system in this thesis. Low bit rate, wireless video systems have applications in cellular videophones, surveillance systems, and mobile patrols.

With a suitable codec, we can provide enough compression to achieve satisfactory video quality at low bit rates. For our video system, the codec must also define important parameters such as frame size, frame rate, input format, and output format. The ITU-T H.263 [2] video codec standard is suitable for low bit rate, wireless video systems. It provides acceptable picture quality and frame rate for video streams at N x 64 kbps and less. Because the codec must heavily compress the video, an H.263 system is computationally intensive. The computational complexity of H.263 is increasing, since recent revisions include new features that enhance the capabilities of H.263. The enhanced features attempt to provide higher quality video at the expense of higher computational complexity. Since higher computational complexity translates to increased power dissipation, an implementation of H.263 for portable, wireless systems must focus on low power design.

An essential component of the H.263 codec is the compression of video frames in space. This compression is performed with the discrete cosine transform (DCT). The inverse discrete cosine transform (IDCT) is essential in the decompression process. The

DCT and the IDCT are some of the most computationally intensive blocks in H.263. The combined computational complexity of the DCT and IDCT in the coder surpasses that of any other unit, consuming 21% of the total computations [3]. The IDCT in the decoder also incurs the largest computational cost in the decoder. In an H.263 system running at resolution of 144 x 176 pixels at 10 frames per second, DCT and IDCT related calculations require over 45 million multiplication and addition operations per second. The high computational complexity of the DCT and the IDCT leads to high power dissipation in hardware, so low power design of DCT and IDCT units are essential for portable wireless video systems based on H.263. In fact, most leading DCT and IDCT designs now stress battery life in addition to performance [4].

The remainder of this thesis explores the development of low power hardware designs for the DCT and the IDCT. The organization is as follows. In Chapter 2, we provide the background information necessary to understand the H.263 codec; we emphasize the background information relative to the DCT and IDCT operations. We also review previous architectures and low power design techniques for DCT and IDCT units. Next, in Chapter 3, we describe the baseline architecture for our DCT and IDCT. The baseline architecture provides us with a platform to test our low power techniques. Chapter 4 details each of the low power design techniques that we apply to the baseline architecture. The resulting improvement in power dissipation for each of these low power techniques appears in Chapter 5. We also give the total improvement in power dissipation for the combined effect of all the low power techniques. Finally, in Chapter 6, we draw conclusions about our work on low power DCT and IDCT circuits.

Chapter 2: Preliminaries

In this chapter, we first explain the concepts and terms necessary to understand the H.263 codec. We review the major computational blocks in an H.263 system, as well as the format of the transmitted data. Additionally, we give more detail about the DCT and IDCT transforms and give an example of a DCT operation in the context of H.263. Finally, we review previous work done in algorithms, architectures, and low power techniques for the DCT and IDCT.

2.1 H.263

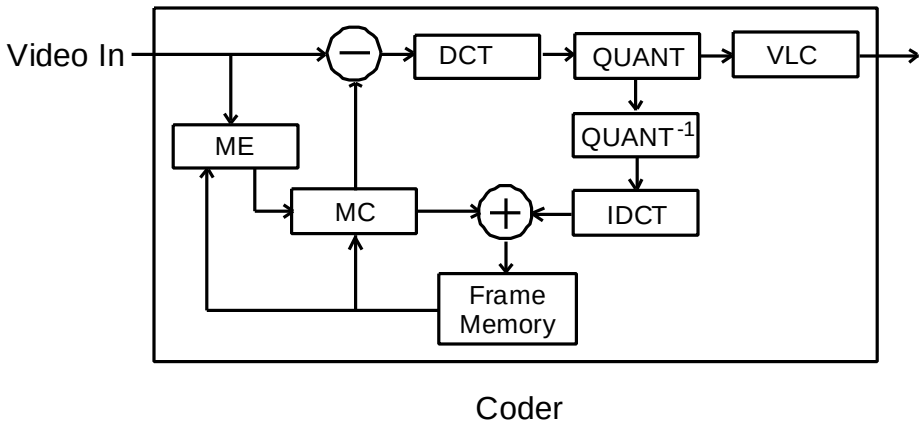


Figure 2.1: H.263 Coder Block Diagram

Figure 2.1 shows a block diagram of the major operations performed in an H.263 coder. The arrows represent the flow of data. Starting at the encoder, the motion estimation (ME) and motion compensation (MC) blocks perform temporal compression by computing the difference between the current frame and the previous frame. The temporally compressed data is sent to the DCT, which is a key component in the spatial compression of the data. The DCT block transforms the data into spatial frequency coefficients. The quantization (Quant) block divides each DCT coefficient by a quantization parameter, setting insignificant DCT coefficients to zero. The variable

length coder (VLC) performs run length coding on the quantized coefficients, compressing long runs of DCT coefficients.

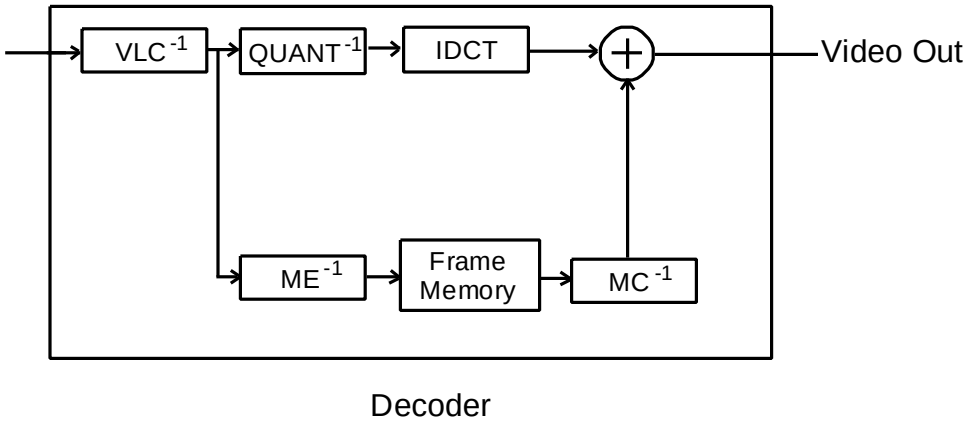


Figure 2.2: H.263 Decoder Block Diagram

The compressed data is sent to the decoder, where the inverse process is applied to reconstruct a frame. Figure 2.2 shows the operation of an H.263 decoder. The variable length decoder (VLC^{-1}) reconstructs the quantized data, and the inverse quantization ($Quant^{-1}$) block multiplies by the quantization parameter to reconstruct the DCT coefficients. Then, the IDCT block transforms the DCT coefficients back into the spatial domain. Finally, the inverse motion estimation (ME^{-1}) and compensation (MC^{-1}) blocks reconstruct the original frame by adding the transmitted difference to the previous frame. These reconstruction steps also occur in the encoder, so both the encoder and the decoder have identical reference copies of the previous frame.

The following sections give more detail about the data structure of H.263 and the above H.263 operations.

2.1.1 Data Structure

The H.263 video format is described in terms of the size of each frame and the transmission rate of the frames. The size is measured in terms of the number of pixels per line and the number of lines per frame. In this paper, we use a fixed size frame of QCIF size, which consists of 176 pixels per line and 144 lines. Since H.263 allows custom frame rates, we must choose a suitable frame rate for the 20 kbps bandwidth that is allocated for video data in our target 60 kbps wireless channel. We choose a constant frame rate of 10 fps, which provides a good compromise between frame rate and frame quality. The remaining 40 kbps of our wireless channel is reserved for turbo coding overhead.

Figure 2.3 shows the data partitions within an H.263 QCIF picture frame. Sixteen lines make up a **group of blocks** (GOB). The nine GOBs in QCIF are further divided into **macroblocks,** which are basic units of data in H.263. Quantization and motion estimation, and bit rate control are performed at the macroblock level. With QCIF resolution, one video frame requires 9 rows and 11 columns of macroblocks for a total of 99 macroblocks. Each macroblock contains six 8x8 **blocks**; the six blocks represent a 16x16 pixel area of a video frame. A block contains an 8x8 matrix of data samples for a total of 64 samples. The DCT, IDCT, and VLC operations are performed on these 8x8 blocks of data.

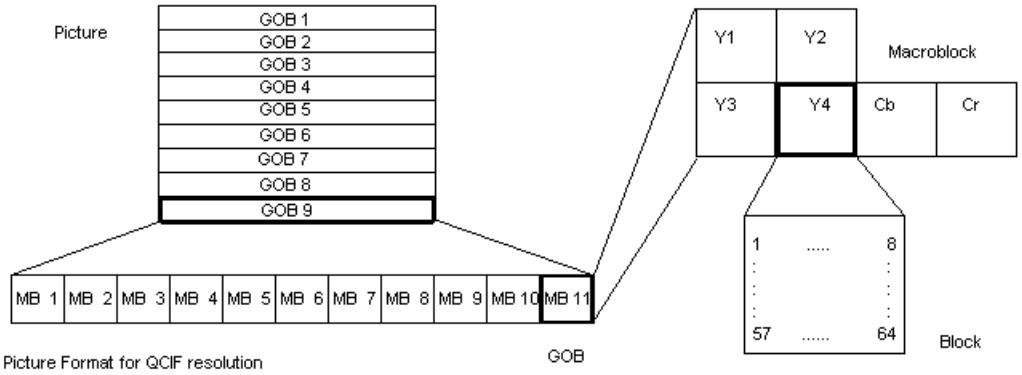


Figure 2.3: H.263 Data Structures

The six blocks in a macroblock conveniently represent data in YCbCr format, which contains a luminance component (Y), a blue chrominance component (Cb), and a red chrominance component (Cr). Within a macroblock, four blocks contain luminance values, one block contains blue chrominance values, and one block contains red chrominance values. Luminance blocks describe the intensity or brightness of pixels, while chrominance blocks contain information about the coloration of pixels. Since the human eye is less sensitive to color than to intensity, each chrominance block is downsampled by a factor of two in both the *x* and *y* directions. Figure 2.4 shows where the luminance and chrominance samples are taken. Each luminance value corresponds to one pixel, while each chrominance value is shared by four pixels.

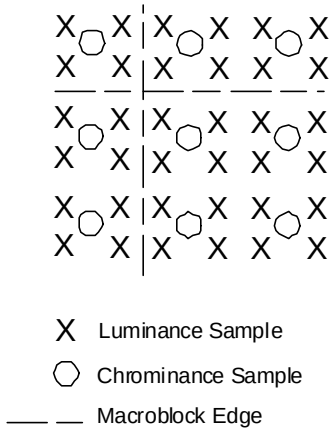


Figure 2.4: Sampling of Luminance and Chrominance in H.263

2.1.2 Temporal Compression

Temporal compression is the compression of video in time. In a temporally compressed frame – known as an INTER frame – only the difference between the current frame and the next frame is transmitted. A frame that is not temporally compressed is known as an INTRA frame. In our H.263 coder, the first frame is an

INTRA frame and all subsequent frames are INTER frames. Temporal compression occurs in two steps: motion estimation and motion compensation.

The goal of motion estimation is to find the macroblock in the previous frame that is the best match for a macroblock in the current frame. To compare the macroblocks, a copy of the previous frame is kept in memory. The displacement between the current macroblock and the best match is then encoded as motion vector. A common metric for finding the best matching block in the previous frame is the **sum of absolute differences** (SAD). The SAD is a measure of how well a macroblock in one frame matches a nearby macroblock in the next frame. Sequences with large amounts of motion tend to produce larger magnitude SAD values, while sequences with less motion tend to produces smaller magnitude SAD values. SAD values are obtained by the following equation:

$$SAD = \sum_{k=1}^{16} \sum_{l=1}^{16} \left| Y_{i,j}(k,l) - Y_{i-u,i-v}(k,l) \right| \tag{2.1}$$

In Eq. (2.1), the magnitudes of luminance pixels, $Y_{i,-u,j-v}(k,l)$, in macroblocks that are offset by (u,v) in the previous frame are subtracted from the corresponding magnitudes of each luminance pixel, $Y_{i,j}(k,l)$, of the macroblock that is at position (i,j) in the current frame. In motion estimation, the candidate macroblock with the lowest SAD is usually chosen as the most likely match for the current macroblock. The final result of the motion estimation process is a motion vector that describes the offset of the chosen macroblock from the current macroblock.

After motion estimation, the motion compensation unit computes the difference between the chosen macroblock and the current macroblock. In video with small amounts of motion, this difference usually has small magnitude or zero magnitude. This aids the compression process, since a large amount of highly correlated data is sent to the DCT unit, which has the ability to represent correlated data with a small number of uncorrelated coefficients.

The decoder applies the reverse process to reconstruct the picture. The decoder also keeps a copy of the previous frame in memory. The best match from the previous

frame is located from the information provided in the motion vector. The difference computed by the motion compensation unit is added to this best matching macroblock. In this process, the new picture is reconstructed from the motion vectors, the motion compensated data, and the previous frame.

2.1.3 Spatial Compression

After undergoing the motion estimation and motion prediction processes, H.263 data undergoes a three-step spatial compression process. The first step is the DCT, which transforms spatial data into spatial frequency coefficients. Next, the quantization unit attempts to force visually insignificant DCT coefficients to zero, thus producing long runs of '0' in the output matrix. Finally, the VLC unit compresses these long runs of '0'.

The DCT operates on 8x8 blocks of data. A detailed description of the DCT operation is given in section 2.2. The DCT transforms the spatial input data into an 8x8 matrix of spatial frequency coefficients. Typical picture data results in an output matrix that has larger magnitude low frequency coefficients and smaller magnitude high frequency coefficients. The human eye is more sensitive to the low frequency coefficients than to the high frequency coefficients.

Because of the characteristics of the DCT coefficients, the quantization block divides the DCT coefficients by a parameter called QUANT. The quantization block aids the compression process by forcing the small magnitude, high frequency coefficients to zero. The QUANT parameter varies depending upon the available bandwidth. QUANT will increase if less bandwidth is available, thereby forcing more coefficients to zero. Likewise, QUANT will decrease if more bandwidth is available, thereby forcing fewer coefficients to zero. Higher values of QUANT also introduce more distortion into the decoded video, while lower values of QUANT produce higher quality pictures. For our low bit rate system, the quantization process introduces a large amount of error. Since the quantization operation forces high frequency coefficients to '0', the output matrix is re-ordered into a vector with the low frequency coefficients at the beginning and the high frequency coefficients at the end. Since the high frequency

coefficients are clustered at the end of the input vector, large runs of ‘0’ should occur at the end of the vector.

The large runs of ‘0’ are further compressed in the VLC block. The output data of the quantizer block is treated as a series of events. An event is a combination of three parameters: LAST, RUN, and LEVEL. The LAST bit indicates whether there are any remaining non-zero coefficients in the block. When LAST is set to ‘1’, the rest of block will contain all zeroes, which do not have to be transmitted. RUN indicates how many successive zeroes preceded the current non-zero coefficient. Finally, LEVEL is the value of the current non-zero coefficient. In the VLC coder the most commonly occurring events are assigned short, variable length codes from a table. The variable length codes require three to thirteen bits. Other events are represented with a seven bit escape code followed by an explicit declaration of LAST with one bit, RUN with seven bits, and LEVEL with eight bits.

The decoder applies these processes in reverse order to reconstruct the blocks. First, the VLC decoder recovers the vector of IDCT coefficients. Next, the inverse quantizer multiplies these coefficients by the QUANT parameter to approximately reconstruct the DCT coefficients. Finally, the IDCT transforms the coefficients back into the spatial domain.

The inverse quantization, IDCT, and inverse motion compensation processes also occur in the coder. This is so that the decoder and the encoder have similar reference copies of the previous frame to use in motion estimation and compensation. Since the compression process changes the original image, the reference frame in the encoder must undergo the same operations as the reference frame in the decoder. If the encoder and the decoder use exactly the same process for performing the IDCT, then the copies of the reference frame will be identical and the decoded video will remain stable. With differing IDCT algorithms that do not have sufficient accuracy, the copies of the reference frame can become unsynchronized between the coder and the decoder. In this case, small differences in the IDCT calculations propagate to future frames and can grow large enough to make the system unstable.

2.1.4 Bit Rate Control

Because of the limited capacity of our wireless channel, we must keep the data transfer rate under 20 kbps. No rate control algorithm is specified in the H.263 recommendation. However, the ITU test model describes the rate control algorithm that is used in our prototype [5][6]. The data rate is controlled at the picture level and at the macroblock level.

At the picture level, rate control occurs by skipping entire pictures when the target frame rate is less than the frame rate of the original video sequence. Higher frame rates increase the bit rate and improve the temporal quality of a sequence, while lower frame rates decrease the bit rate and degrade the temporal quality of a sequence.

At the macroblock level, rate control is achieved by modifying the quantizer value, depending on the amount of data previously sent. The quantizer value can be incremented or decremented for each macroblock, depending on the rate the output buffer is filled. Pictures with high motion and high detail will have larger quantizer values than pictures with little motion and little detail. Higher quantizer values reduce the number of bits required to encode a picture and degrade the quality of the picture. Lower quantizer values increase the number of bits required to encode a picture and improve the quality of the picture. To optimize the tradeoff between bit rate and picture quality, the rate control attempts to use the highest quantizer values that do not cause the target bit rate to be exceeded.

2.1.5 Bit Stream

The bit stream is divided into layers that follow the hierarchical division of H.263 data: the picture layer, the GOB layer, the macroblock layer, and the block layer.

The picture layer includes information relative to an entire frame of data. The picture layer begins with a start code that is byte aligned and uniquely identifies the start of a frame. A field tracks the total number of frames in the current video sequence, including skipped frames. Also included is information about the type of screen, the picture size, the frame rate and the type of picture: INTER or INTRA. The frame layer

also specifies which, if any, of several optional H.263 extensions are present in the current frame. Another critical piece of information included in the picture layer is the initial quantizer value. This value is important since it can only be incremented and decremented at the macroblock layer. Optional stuffing can be inserted to obtain a byte-aligned bit stream. The picture layer is also responsible for sending an end-of-sequence code that identifies the end of a video.

The GOB layer contains information pertinent to each row of macroblocks. An advantage to having the GOB layer is that it provides synchronization between the coder and the decoder for each row of macroblocks. When the decoder synchronizes with the beginning of each GOB layer, errors will not propagate from the previous row of macroblocks. The GOB layer begins with a start code unique to the GOB layer. The GOB layer includes the frame number containing the current GOB and the ID number of the current GOB within that frame. Finally, the GOB layer provides a final opportunity to change the QUANT parameter to any value, since the QUANT parameter can only change by increments at the macroblock level.

The macroblock layer contains information related to macroblocks. The motion vector data is represented by VLC code words for both the horizontal and vertical components. Since predicted macroblocks can potentially remain exactly the same as in a previous frame, they may contain no new information. In this case, a single bit, the **coded macroblock indication** (COD) bit, signals whether to treat the entire macroblock as all zero data with zero motion. An INTER macroblock will produce all zero input to the IDCT if its COD field is zero. The macroblock layer also includes information about its luminance and chrominance blocks. The **coded block pattern** (CBP) describes the values of DCT coefficients for each luminance and chrominance block. The CBP of a block is set to '1' if the block contains at least one non-zero DCT coefficient that is not the DC coefficient at position (0,0). Otherwise, the CBP of a block is set to 0. An INTER block produces all zero input to the IDCT if its DC coefficient is zero and its CBP is zero. Additionally, the macroblock layer provides an opportunity to adjust the QUANT parameter by +/- 1 or +/- 2.

The block layer is at the lowest level of the hierarchy and contains information relevant to each 8x8 luminance and chrominance block. The information transmitted at

the block level includes the VLC codes that describe the events in a spatially compressed block. The only other information included at the block level is the DC coefficient for INTRA blocks. The DC coefficient is important for video quality in INTRA blocks, so it is treated differently from other coefficients; the quality of an INTRA block affects all subsequent INTER blocks.

2.1.6 Peak Signal to Noise Ratio

The average **peak signal to noise ratio** (PSNR) of a frame in a video sequence is measured as

$$PSNR = 10 \log \frac{1}{x \cdot y} \sum_{rows} \sum_{cols} \frac{255^2}{Y_1 - Y_2}, \quad (2.2)$$

where x is the number of rows, y is the number of columns, and Y_1 and Y_2 are the luminance values in the original and the reconstructed pictures. We use the average PSNR over all frames as a quantitative measure of the quality of a video sequence.

2.2 DCT/IDCT

2.2.1 Background

In signal processing applications, the DCT is the most widely used transform after the discrete Fourier transform [7]. The DCT and IDCT are important components in many picture compression and decompression standards, including H.263, MPEG, HDTV, and JPEG. The applications for these standards range from still pictures on the Internet to low quality videophones to high definition television. The DCT and IDCT also have applications in such wide-ranging areas as filtering, transmultiplexers, speech coding, and pattern recognition [7].

The DCT transforms data from the spatial domain to the spatial frequency domain. The DCT attempts to decorrelate image data, which is typically highly

correlated for small areas of an image. Heavily correlated data samples provide much redundant information, whereas just a few pieces of uncorrelated information can represent the same data much more efficiently. For example, just five pieces of uncorrelated information completely describe a sinusoid: the phase, the frequency, the starting time, the amplitude, and the fact that it is a sinusoid. It is far less efficient to transmit samples for the duration of the sinusoid. Through the DCT transform, the energy of typical image data is packed efficiently into a few uncorrelated spatial frequency coefficients. The spatial frequency coefficients represent a small area of image much more efficiently than the sampled data.

2.2.1.1 Formulation

In most video compression applications, DCT and IDCT operations operate on 8x8 blocks of data. For an 8x8 block of samples *x(m,n)*, the two dimensional (2-D) DCT and IDCT are formulated in (2.3) for $0 \le k,l \le 7$ and in (2.4) for $0 \le m,n \le 7$

$$Y(k,l) = \frac{1}{4}a(k)a(l)\sum_{n=0}^{7}\sum_{m=0}^{7} x(m,n)\cos\left(\frac{(2m+1)pk}{16}\right)\cos\left(\frac{(2n+1)pl}{16}\right) \quad (2.3)$$

$$x(m,n) = \frac{1}{4}a(k)a(l)\sum_{k=0}^{7}\sum_{l=0}^{7} Y(k,l)\cos\left(\frac{(2m+1)pk}{16}\right)\cos\left(\frac{(2n+1)pl}{16}\right) \quad (2.4)$$

where $\alpha(0) = 1/\sqrt{2}$ and $\alpha(j) = 1$ for $j \neq 0$. A measure of the efficiency of a DCT/IDCT algorithm is the number of multiplications. The straightforward implementations of the DCT and IDCT require N^4 multiplications for an NxN block, so they are rarely implemented. Most implementations use fast algorithms that reduce the number of multiplications to $O(N^3)$. Fast algorithms not only improve the speed of the DCT but also reduce the number of computations.

The 2-D DCT transforms an 8x8 block of spatial data samples into an 8x8 block of spatial frequency components. The IDCT performs the inverse of DCT, transforming spatial frequency components back into the spatial domain. Figure 2.5 shows the

frequency components represented by each coefficient in the output matrix. The low frequency coefficients occur in the top left side of the output matrix, while higher frequency coefficients occur in the bottom right side. The DC coefficient at position (0,0) gives an idea of the average intensity (for luminance blocks) or hue (for chrominance blocks) of an entire block. Moving horizontally from position (0,1) to position (0,7), the coefficients give the contributions of increasing vertical frequency components to the overall 8x8 block. The coefficients from position (1,0) to position (7,0) have similar meaning for horizontal frequency components. Moving diagonally through the matrix gives the combined contribution of horizontal and vertical frequency components. The original block is rebuilt by the IDCT with these discrete frequency components.

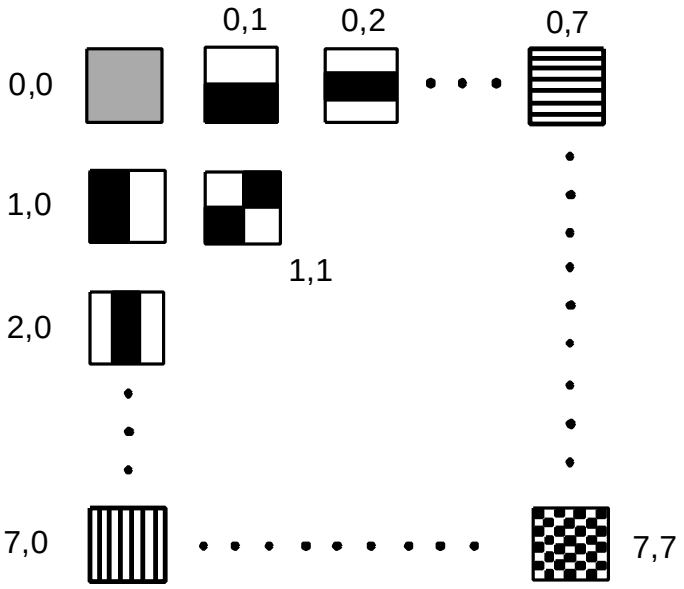


Figure 2.5: DCT Coefficients

High frequency coefficients have small magnitude for typical video data, which usually does not change dramatically between neighboring pixels. Additionally, the human eye is not as sensitive to high frequencies as to low frequencies. It is difficult for the human eye to discern changes in intensity or color that occur between successive

pixels. The human eye tends to blur these rapid changes into an average hue and intensity. However, gradual changes over the 8 pixels in a block are much more discernable than rapid changes. When the DCT is used for compression purposes, the quantizer unit attempts to force the insignificant high frequency coefficients to zero while retaining the important low frequency coefficients.

2.2.1.2 H.263 Example DCT and IDCT used for Spatial Compression

This section gives an example of spatial compression on an 8x8 block of video data taken from a luminance block in a video sequence. We show the process of extracting the data, performing the DCT, performing quantization, and performing VLC encoding. The example shows the efficient compression enabled by the DCT transform.

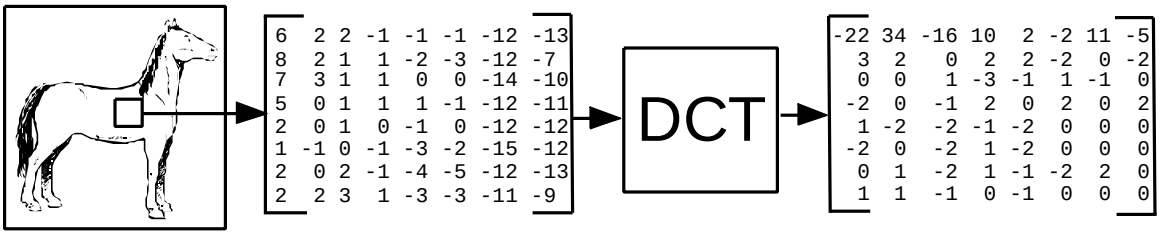


Figure 2.6: Example DCT

Figure 2.6 shows the process of taking an input block from a picture and applying a DCT transform. An 8x8 block of samples (B) is taken from the original picture (A) and sent to the DCT block (C). The output (D) of the DCT is representative of typical video data. The larger magnitude coefficients are concentrated in the low frequency area of the output matrix, while smaller magnitude coefficients occupy the rest of the output matrix. These coefficients are then sent to the quantizer block.

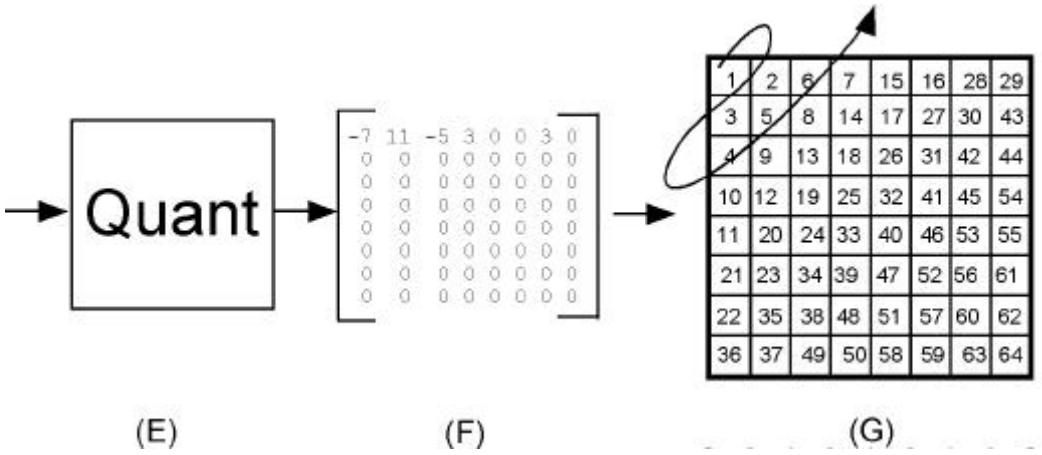


Figure 2.7: Example Quantization Process and Zigzag Scan

Figure 2.7 shows the quantization process followed by a zigzag scan of the results. After the quantization operation (E) divides each coefficient by three, the quantized matrix (F) contains a few non-zero coefficients concentrated in its upper left corner. Note that many of the insignificant high frequency coefficients are now set to zero. Because of this distribution of data, the quantized DCT coefficients are then scanned into a vector according to the scanning scheme in (G). The arrow shows the progression of the scan for the first six values in the matrix. This scanning scheme attempts to force the coefficients into a vector that contains long runs of zeroes.

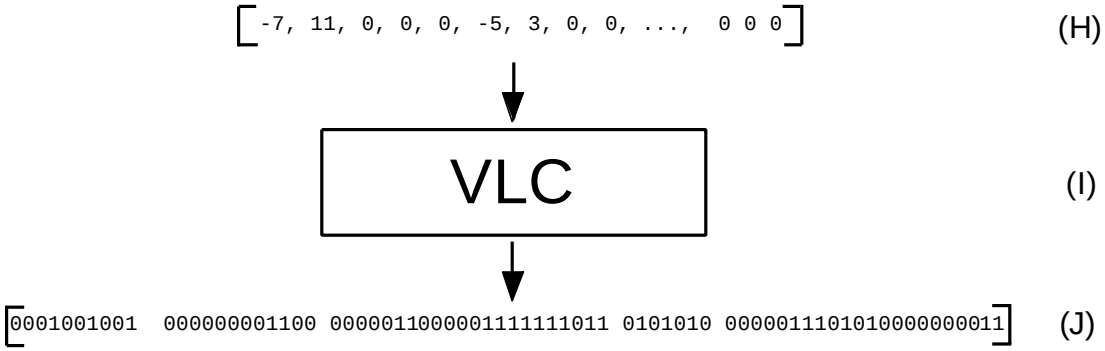


Figure 2.8: Example VLC Encoding

Figure 2.8 shows the operation of the VLC encoder. The vector with long runs of zeroes (H) is sent through the VLC block (I), where the long runs are compressed into efficient variable length codes (J). Note the length of the entire code needed to represent a block of 64 samples is only 73 bits long. To represent the block as an 8x8 matrix containing 64 discrete samples of spatial data would take 64x8 = 512 bits. The compression enabled by the DCT saves over 85% of the bits that would have been required by sending each sample.

In the decompression scheme, the opposite process is performed to reconstruct the original block. This section gives an example of spatial decompression on the 8x8 block of video data. The decoding process involves VLC decoding, inverse quantization, and an IDCT operation. The resulting data approximates the original data, because the DCT and the quantization processes introduce error into the transmitted data.

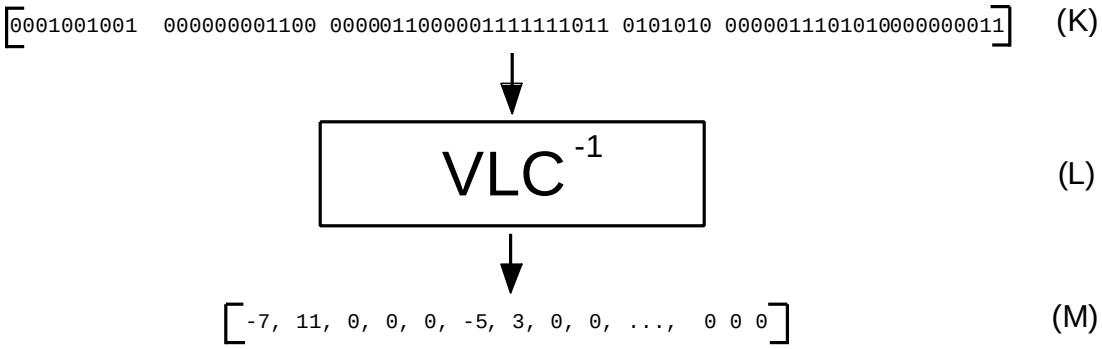


Figure 2.9: Example VLC Decoding

Figure 2.9 shows the operation of the VLC decoder. The decoder (L) reconstructs the variable length codes (K) into a vector of quantized coefficients (M).

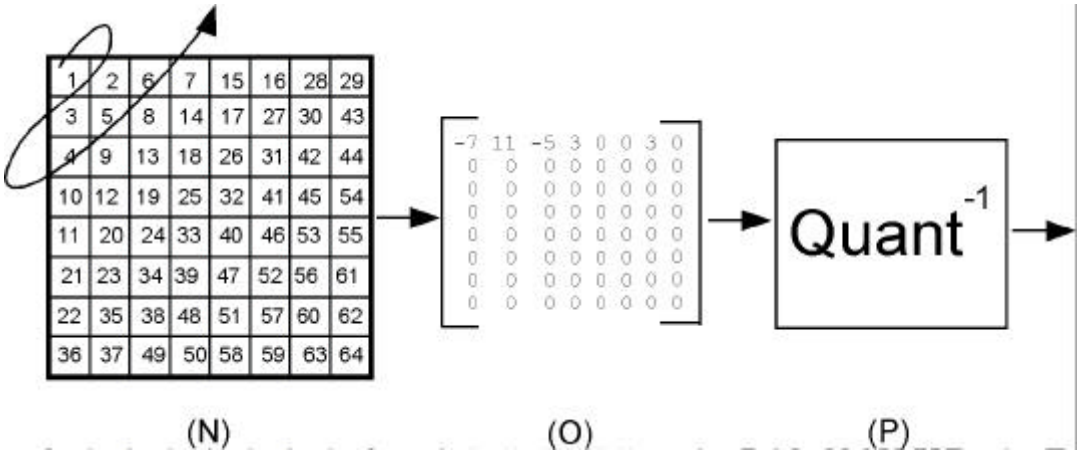


Figure 2.10: Example Inverse Quantization

Figure 2.10 shows the de-scanning process (N), which places the coefficients back into an 8x8 matrix (O). The matrix is then de-quantized in the inverse quantization unit (P), which multiplies each coefficient by three to approximate the original DCT coefficients. Note that the small values of the insignificant high frequency coefficients are not recovered.

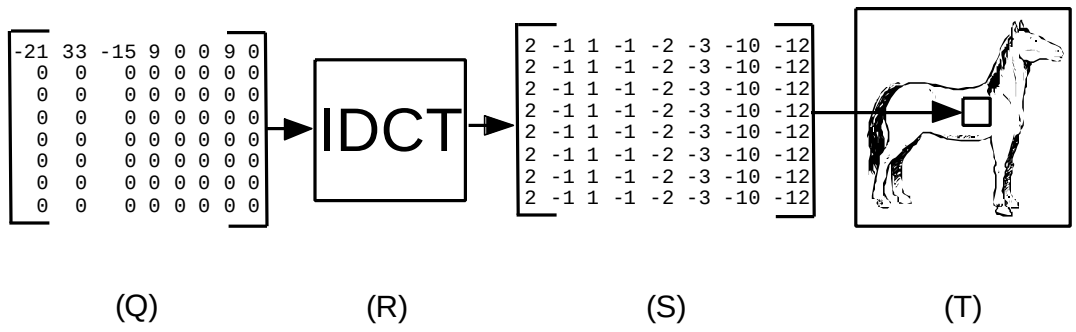


Figure 2.11: Example IDCT

Figure 2.11 shows the results of the IDCT process. The IDCT unit (R) transforms the reconstructed coefficients (Q) back into the spatial domain data (S). The resulting block of data is then added back into the transmitted picture (T). Note that the reconstructed samples (S) are not equivalent to the input samples (B) from the original

picture. The quantization process and the finite precision arithmetic of the DCT and the IDCT contribute to the error in the transmitted block. However, the transmitted block approximates the original pattern; samples with small positive values occur at the left hand side of the block, and samples with larger negative values occur at the right hand side of the block.

2.2.2 DCT/IDCT Architectures

Since the straightforward implementations of equations (2.3) and (2.4) are computationally expensive, most implementations employ fast algorithms that reduce the computational cost. Fast algorithms can be broken down into two broad categories: row/column approaches and direct, fast 2-D approaches. Both use 1-D DCTs as their basic building blocks. For the row/column approach, the 1-D DCT/IDCT of each row of the 8x8 input data is taken, and these intermediate values are transposed. Then, the 1-D DCT/IDCT of each row of the transposed values results in the 2-D DCT/IDCT. The row/column approach results in simple and regular implementations, but it is less computationally efficient than the direct fast 2-D approach. The direct approach is faster and requires about half the multiplications of the row/column approach, but it has a more complicated controller and data path.

2.2.2.1 Row/Column Approaches

Since the 2-D DCT is an orthogonal transform, decomposing the operation into two sets of 1-D DCTs reduces the number of multiplications to $O(N^3)$ for an NxN block of data. First, the 1-D DCT of each row of the NxN input data is taken, and the intermediate values are stored in a transposition memory. Then, the 1-D DCT of each column of the intermediate values results in the 2-D DCT. Row/column approaches are the most popular approach for computing the 2-D DCT/IDCT. They result in simple, regular implementations that are less computationally efficient than direct 2-D implementations. However, row/column approaches also suffer from increased latency because of the transposition operation. The straightforward equations for computing the

8-point 1-D DCT and 1-D IDCT required for the row/column approach are written in (2.5) for $0 \le k \le 7$ and in (2.6) for $0 \le n \le 7$

$$Y(k) = a(k) \sum_{n=0}^{7} x(n) \cdot \cos\left(\frac{(2n+1)pk}{16} \right) \tag{2.5}$$

$$x(n) = a(k) \sum_{k=0}^{7} Y(k) \cdot \cos\left(\frac{(2n+1)pk}{16} \right) \tag{2.6}$$

where $a(0) = 1/\sqrt{2}$ and $a(j) = 1$ for $j \neq 0$. Many row/column implementations further reduce computation by using fast 1-D DCT/IDCT algorithms.

The Chen algorithm and similar algorithms [8], [9], and [10] improve on the direct implementation of the 1-D DCT/IDCT. The Chen algorithm requires only 16 multiplications for an eight point 1-D DCT/IDCT, so an 8x8 row/column implementation requires just 256 multiplications for a 2-D DCT/IDCT. Using the straightforward formulations of the 1-D DCT/IDCT, the row/column approach would require 1024 multiplications for a 2-D DCT/IDCT. Architectures based on the Chen algorithm are a popular choice for implementing the 2-D DCT/IDCT and are also employed for our DCT/IDCT unit [11]-[19].

2.2.2.2 Direct Approaches

The direct, fast 2-D DCT/IDCT approach requires about half of the calculations of the row/column approach at the expense of irregularity in the data paths and more complex control logic. Additionally, the elimination of the transposition memory reduces latency. However, the register space saved by the elimination of the transposition memory is usually offset by the additional registers necessary to re-order inputs and store intermediate values. Three different methods are often used to implement a direct, fast 2-D approach: the matrix method, the vector-radix method, and the time-recursive method.

Matrix approaches are among the most efficient of any approaches to the 2-D DCT [20][21]. Feig and Winograd's algorithm requires just 94 multiplications to

compute the 2-D DCT. The number of multiplications is further reduced to 54 by computing a scaled version of the DCT. These algorithms require complex matrix operations, which make most approaches difficult to implement in hardware.

Popular for hardware implementations, the vector-radix approach results in a fast 2-D DCT/IDCT algorithm that is more easily realized in hardware [22]-[27]. The vector-radix, 2-D DCT/IDCT operates similarly to a radix-2 fast Fourier transform. Based on the indices of the pixels, the 8x8 input matrix is divided into four blocks of data: odd/odd indices, odd/even indices, even/odd indices, and even/even indices. The decomposition results in a form that can be realized with just eight 1-D DCT/IDCT operations and some post processing. This is half the number of 1-D DCT/IDCTs required for the row/column approach. Additionally, Lee, Chen, Chen, Chen, and Ku find that the direct 2-D method leads to lower internal bit-width requirements than the row/column method [23]. This is because the data path from any input to the output travels through just one fixed-point multiplier. In the row/column approach, the data must travel through a fixed-point multiplier in both the row operation and in the column operation. However, the vector-radix approach suffers from irregularity and a large amount of preprocessing and postprocessing.

For time recursive DCT/IDCT algorithms, the IIR filter-like architecture results in simple I/O behavior, parallel implementation, modular design, regular layout, ease of pipelining the input data, and highly localized connections [28][29]. Implementations of the time recursive algorithm require just eight 1-D DCT operations. Architectures have been proposed that trade off between area, speed, and power [28]-[31], (23), (24). The time recursive algorithm for the 2-D DCT computes both the DCT and discrete sine transform (DST) coefficients; DST coefficients can be used to perform motion estimation in the DCT domain [30][31]. The advantage of motion estimation in the DCT domain is a reduction in chip area, since the motion estimation unit and the 2-D DCT unit are now combined. Motion estimation in the DCT domain also eliminates the need for an IDCT unit in the encoder. One disadvantage of time recursive algorithms is the heavy use of the arithmetic units, which are active on nearly every clock cycle.

2.2.2.3 Approximate Algorithms

Approximate algorithms reduce the complexity of the DCT/IDCT at the expense of introducing error in addition to the error introduced by quantization and finite precision arithmetic. Approximate algorithms reduce the number of multiplications, simplify the data path, and perform the DCT/IDCT operations much more quickly than in the other algorithms described above.

The variable complexity approximations of [32] reduce the DCT complexity. The authors use the SAD and QUANT parameters to determine how the DCT should be performed. For blocks with small SAD values, the 1-D DCT is computed with an approximation that requires just 24 adds and 2 shifts. For sequences with high QUANT parameters, the authors find that the approximations have little effect on the distortion and decrease computations by 22%-27%. Girod and Stuhlmuller calculate the 2-D DCT with just the DC coefficient and the first two non-DC coefficients [33]. They approximate these coefficients with the following expressions.

$$\begin{aligned} X_{0,0} &= k_{DC}\left(c_0 + c_7 + r_0 + r_7\right) \\ X_{0,1} &= k_{AC}\left(r_0 - r_7\right) \\ X_{1,0} &= k_{AC}\left(c_0 - c_7\right) \end{aligned} \tag{2.7}$$

X - DCT coefficient
k – Scaling Factor
c_x – Sum of Column x
r_x – Sum of Row x

This method requires just 33 additions and 3 multiplications to compute the approximate 2-D DCT of an 8x8 block. The approximation is performed on blocks that have a SAD rating of less than 20 x QUANT. For this threshold, the authors report that the approximation results in overall computational savings of about 50% with no degradation in quality.

2.2.2.4 Multipliers

Because of the large number of multiplication operations, a major concern in DCT/IDCT architectures is the method of implementing the multipliers. Most approaches use fixed-point arithmetic, since floating-point units are needlessly too complex. The two most popular forms of implementing the fixed-point multipliers are a distributed arithmetic architecture (bit-serial) and a straightforward architecture (bit-parallel).

Operating on one to five bits concurrently, the distributed arithmetic (DA) architecture processes data in a bit-wise order. The DA architecture uses a ROM look-up table in conjunction with a shift-and-add approach to iteratively compute results, which gives it the advantage of taking up minimal area with a high-speed capability. Another advantage of DA architectures is the ease of implementing variable precision arithmetic. If the DA starts processing the most significant bits first, each iteration comes closer to the final result. By controlling the number of iterations, a fine grain, variable precision arithmetic can be implemented. Implementations of the DA architecture are area efficient and have a regular VLSI realization [14]. A disadvantage of the DA architecture is the frequent ROM accesses, which result in larger power dissipation. An analysis finds that the DA architecture provides higher speed than a hardwired multiplier but also dissipates more power [34]. The advantages of the DA make it a popular choice for high-speed, low area implementations [12][14][16][21][26][28][34]-[36].

Several designs of bit-parallel multipliers, which compromise speed, area, and power dissipation, have been proposed. An array multiplier is a straightforward implementation of the bit-parallel architecture, and it is easily implemented from design libraries such as Synopsys DesignWare. [19]. The direct implementation of an array multiplier requires far more area than the bit-serial approach of a DA implementation. Most architectures attempt to improve the speed and area of bit-parallel approaches with more efficient architectures. Some choices that improve speed include ROM-based multipliers [11] and PLA-based multipliers [34]. The fact that one multiplicand is known *a priori* in the DCT/IDCT can be exploited for optimization of constant

multipliers that use a shift-and-add approach [13][17][18][37]. A constant multiplier can multiply one input by a constant coefficient, so DCT/IDCT designs require a different constant multiplier for each coefficient. However, these multipliers are faster and more area efficient than array multipliers. Constant multipliers perform with accuracy close to that of a look-up table but with far smaller area. Additional area is saved with common subexpression sharing between the adder rows of a constant multiplier. One interesting multiplier design uses rotation-based arithmetic, which reduces shift-and-add operations by 28% [30][31]. Another alternative to an array multiplier is a look-up table. The look-up table provides better accuracy than other implementations at the cost of area. To save area, some low-speed implementations consider a version of the Booth algorithm [17][38].

Some bit-parallel implementations use constant multipliers to reduce the complexity of the multiplier circuit. Since architectures for constant multipliers can vary widely, we describe the operation of a constant multiplier that is suitable for use in a DCT/IDCT application.

The constant multiplier computes a product based on a shift-and-add approach. Consider the product of a number N and the nine-bit constant DCT coefficient a (181 = 010110101b)

$$N \times 010110101 = \left(2^7 \times N\right) + \left(2^5 \times N\right) + \left(2^4 \times N\right) + \left(2^2 \times N\right) + \left(2^0 \times N\right) \tag{2.8}$$

To compute this product, we need to multiply N by powers of two, which is accomplished by shifting N to the left. The shift operation is simple to implement in hardware; it requires re-routing the bits of the input and the intermediate results to the appropriate inputs of subsequent adders. To obtain the final product, we simply add together the five shifted versions of N.

Figure 2.12 shows the area efficient hardware configuration of the constant multiplier. An array multiplier would need eight adder rows for each significant bit in the constant a; the basic shift-add multiplier only needs an adder row for each '1' in the constant a. Additional area is saved by sharing common sums among the adders and shifting them appropriately. For example, in the constant a, the pattern "101" occurs

twice: in bit positions two through four and in bit positions five through seven (*a* = 0**10110**101). In Figure 2.12, the two occurrences of "101" are handled as follows. First, the sum $X = (2^0 \times N) + (2^2 \times N)$ is computed; then the sum $Y = (2^2 \times X) + (2^5 \times X)$ is computed. Computing *Y* requires just two add operations to handle the '1's that occur in bit positions two, four, five, and seven. Had we not shared the "101" subexpression, the sum $(2^2 \times N) + (2^4 \times N) + (2^5 \times N) + (2^7 \times N)$ would have required three add operations. To compute the final product, we need only add the sum from the '1' in bit position zero to *Y*. The overall multiplier requires only three adder rows to compute the product, while an array multiplier would have required eight adder rows.

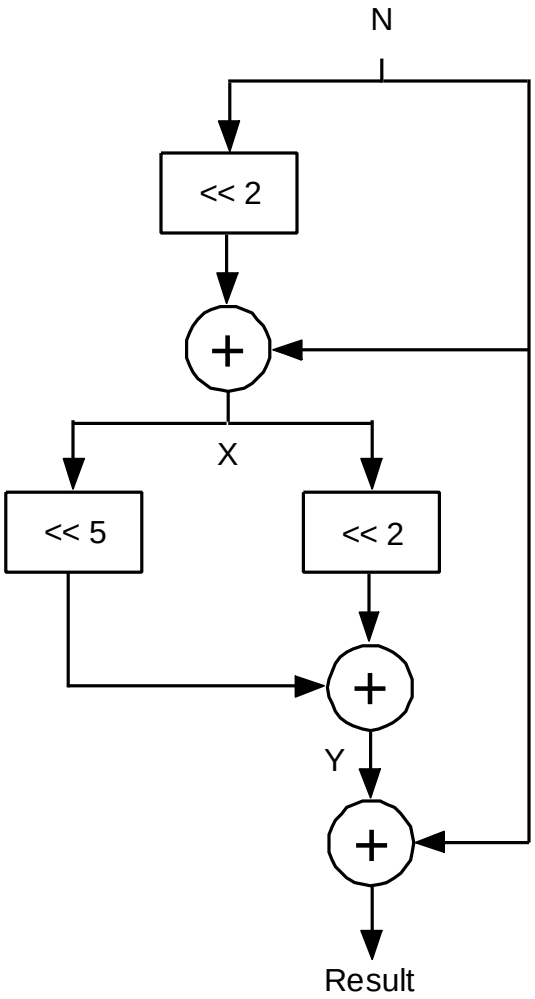


Figure 2.12: Shift-Add Multiplier

2.2.3 Review of Low Power DCT/IDCT Designs

Several previous proposals for DCT/IDCT architectures attempt to reduce power dissipation. The architectures use some common power reduction techniques and some power reduction techniques specific to DCT/IDCT architectures. Because of the wide-ranging applications of the DCT and IDCT, the input data pattern is widely varied for DCT/IDCT architectures in different applications. So various techniques have been proposed for data-dependant power reduction techniques. Some architectures reduce calculations for visually irrelevant DCT coefficients. Other architectures ignore redundant sign bits to save power in the arithmetic units. To save power, some designs disable arithmetic units for the many zero-valued operands in the IDCT. Many other implementations focus on power reduction in the multiplier units

2.2.3.1 Commonly Used Techniques

Some common power reduction techniques, which can be applied to most digital circuits, have been applied to low power implementations of the 2-D DCT. Some of these common low power techniques include clock gating, pipelining, and voltage scaling [39]. In one low power design, a parallel, distributed architecture permits a reduction in supply voltage [23]. Chen and Liu employ parallel processing units that enable power savings from a reduction in clock speed [30], [31]. Other architectures save power by disabling units that are not in use [11][21]. When the units are in a standby mode, they consume a minimum amount of power. Low power, standard cell libraries also reduce power in a few designs [18][23]. Some designs use custom low power units such as a low power SRAM, TSPC flip-flops, Manchester adders [23], or CORDIC processors [30][31]. Another common low power design technique is to reorder the input data so that a minimum number of transitions occur on the input data lines [18].

2.2.3.2 Reduced Calculations for Visually Irrelevant Coefficients

To save power, many architectures reduce calculations for the visually irrelevant high frequency DCT coefficients by adaptively changing the precision in the arithmetic units. The reduced precision saves power at expense of some degradation in picture quality. Xanthopoulos and Chandrakasan employ arithmetic units in which the precision changes adaptively depending on the visual significance of the data [16]. Another architecture allows fine (1-bit increments) resolution in a precision control scheme that successively approximates towards the final value [34]. The upper limit in the number of iterations is determined from the peak-to-peak pixel difference. The quantization parameter can also control the precision: the multiplier units will use fewer bits for quantization parameters with large magnitude and more bits for quantization parameters with small magnitude [21].

Another method for reducing computations is to skip the computation of visually insignificant DCT coefficients altogether [18][40]-[42]. This method, known as pruning, removes the circuit elements that compute high frequency coefficients and sets these coefficients to zero. Pruning techniques attempt to calculate the minimum number of DCT/IDCT coefficients without seriously degrading video quality. The amount of pruning can be based on the SAD of the incoming block. Blocks with a large SAD will have a higher pruning level than blocks with a small SAD [41]. One pruning design has been found to reduce the number of multiplications in a 2-D DCT/IDCT from 256 to 82 [40]. In another design, pruning saves about 33% of arithmetic operations [42]. An additional benefit of pruning is an increased compression ratio, since there is guaranteed to be a long run of trailing zeroes at the end of the DCT coefficient vector.

2.2.3.3 Ignoring Redundant Sign Bits

When they ignore redundant sign bits, arithmetic units save power in the DCT and IDCT circuits because of the large amount of small valued data. Small values occur frequently because only the small difference between the previous and current frame is

transmitted. Ignoring redundant sign bits has no effect on video quality. One architecture reduces power for small coefficients through the use of 4 partitions on its adders. The adder partitions are deactivated to avoid work on redundant sign bits [34]. Another architecture ignores the most significant bits of the inputs if they are common to both inputs, thereby reducing addition operations and ROM accesses [16].

2.2.3.4 Disabling Arithmetic Units

Since the majority of input data for the IDCT is comprised of zero-valued coefficients, significant power reduction can come from disabling adders and multipliers for zero-valued operands. Some architectures use a zero-detect signal that skips addition and multiplication operations for zero-valued data [16][21][34]. The arithmetic units benefit from a standby mode, where they consume a minimum amount of power while not in use. Power reduction can also come from including a blocking latch on adders and multipliers for zero-valued operands. The blocking latch prevents transitions in the arithmetic units and the output registers.

2.2.3.5 Multipliers

Another popular target for power reduction in DCT/IDCT blocks is the method of implementing multipliers. The two most popular implementations for multipliers are bit-serial (distributed architecture) and bit-parallel. A comparison between a bit-serial architecture and a bit-parallel architecture reveals that the bit-parallel architecture dissipates less power in a 2-D DCT circuit [43].

Bit-serial architectures dissipate power due to their high frequency, serialized operation, and high capacitance of the ROM address and bit lines. Because of its bit-serial approach, the DA architecture must operate at a higher internal frequency, resulting in higher power dissipation. Additionally, the ROMs in the DA have a path from VDD to GND during reset, resulting in extra power dissipation. However, the DA architecture can be modified for use in a low power design. Bit-serial architectures have the advantage of easily and finely partitioning data for use in architectures that rely on

variable precision arithmetic to save power [16][34][21]. Taking advantage of the DA's ability to perform variable precision arithmetic, these architectures exploit the high amount of correlation and the small dynamic range of typical video data. Power can also be saved for memory accesses by partitioning the ROM [21]. Another possible advantage of the DA architecture is the reduced area, since the arithmetic units need only be wide enough to operate on a few bits concurrently.

Many low power designs use a power efficient implementation of a bit-parallel format. For a bit-parallel implementation, power consumption is proportional to the number of multiply-and-add operations [34]. For the bit-parallel implementation, a reduction in the number of operations results in lower power dissipation. A common bit-parallel implementation in low power designs is the constant multiplier. A constant multiplier reduces power dissipation with no loss in accuracy over the DA architecture or the array multiplier. These multipliers are faster and more area efficient than array multipliers. Additional operations internal to a constant multiplier are saved with common subexpression sharing. Another approach, rotation-based arithmetic, further reduces shift-and-add operations by 28% [30], [31]. A similar multiplier design employs low-power, low-area CORDIC (coordinate rotation) processors to avoid the costly multiplication operations.

Chapter 3: Baseline DCT/IDCT Design

This chapter details the design process of our baseline DCT and IDCT units. We start with a software prototype system that implements the H.263 codec. Next, we describe the Synopsys environment for VHDL simulation and synthesis of the DCT and IDCT units. Next, we describe our DCT and IDCT algorithms and the baseline hardware architecture of these algorithms.

3.1 Design Flow

For our low power, low bit rate implementation of the DCT and the IDCT, we target our final design for an ASIC implementation. Before we implement the ASIC design, we start with a software prototype of the H.263 system that includes our DCT and IDCT blocks. Next, we implement the DCT and IDCT blocks at the RTL level in VHDL. Then, we synthesize the RTL code into gate-level descriptions of the circuits. The circuits are synthesized onto our VTVT 0.35µ standard cell library. In the future, we plan to implement the DCT and IDCT units in an FPGA and to fabricate the units as an ASIC. An ASIC implementation is necessary to achieve low power design. FPGAs, general-purpose processors, or DSP processors can handle the speed requirements of the DCT and the IDCT, but each dissipates too much power for portable devices. One study finds that an ASIC implementation of the DCT will result in at least $1/100^{th}$ the power dissipation of a general –purpose processor implementation of the DCT [26]. This section describes the baseline DCT and IDCT implemented in the software prototype, the baseline RTL VHDL models, and the baseline gate-level VHDL models.

Figure 3.1 shows the overall design flow of our implementations of the DCT and IDCT. We start with a software prototype of an entire H.263 system that uses the same DCT and IDCT algorithms we plan to use in hardware. At the software level, we start with the baseline designs and add low power techniques for the DCT and the IDCT. We can test the effect of our low power techniques on such parameters as the number of multiplications or the change in PSNR. Additionally, from the software

prototype, we extract data for VHDL simulation. Next, we code the DCT and the IDCT unit in VHDL at the RTL level. If we alter the RTL code, we mirror these alterations in the software prototype and obtain new simulation data and PSNR measurements. Using the test data extracted from the prototype, the RTL VHDL code is analyzed and simulated with Synopsys tools. We check the results for correctness. If the results are incorrect, we debug and fix the RTL model; otherwise we create a synthesized gate level circuit with Synopsys Design Analyzer. The gate-level model provides simulation results, power dissipation measurements, area measurements, and speed measurements with reasonable accuracy. If the results are unsatisfactory, we try other low power techniques in the RTL model and in the software prototype. Otherwise, we have a final gate-level model.

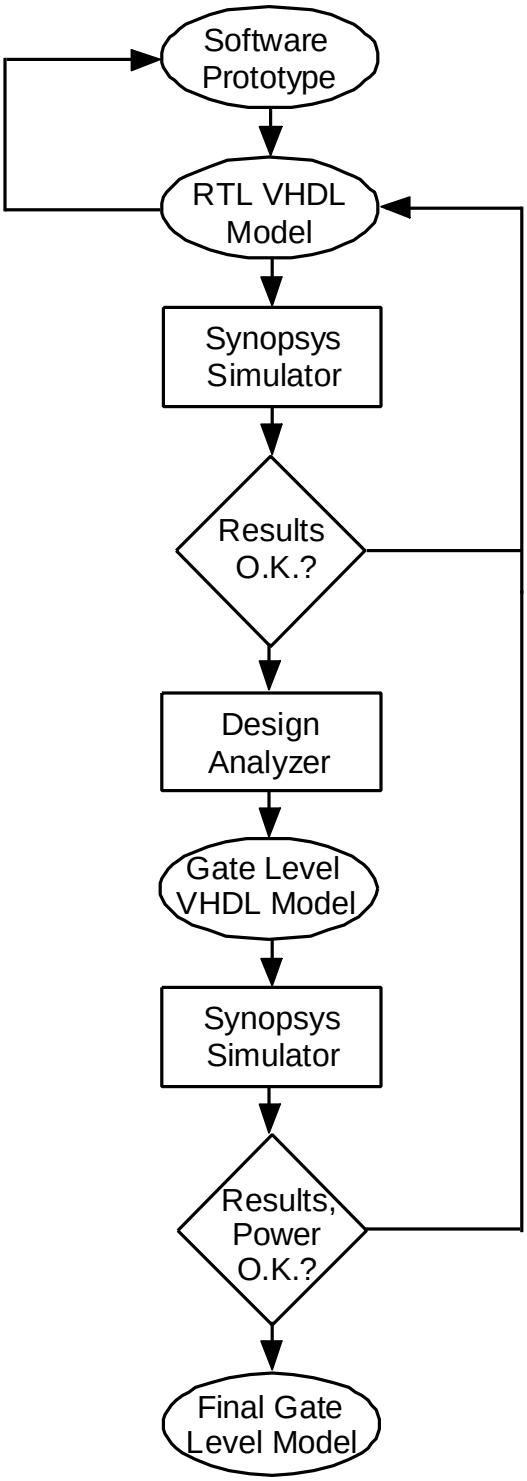


Figure 3.1: Design Flow

3.1.1 Software Prototype

Figure 3.2 shows a block diagram of the software prototype for our H.263 system. The main components of the system are a web camera, two PC's, a wireless transmitter, and a wireless receiver. The web camera takes video at 30 fps in the common red, blue, and green (RGB) format. The RGB format represents a pixel with one byte of data for each of the red, blue and green components. It is then sent through a universal serial bus (USB) interface to the coder program, which is running on PC 1. Optionally, the coder can take a pre-recorded video sequence in RGB format as input. The coder program displays the input video on its monitor. After encoding the input video in H.263 format, the coder sends an output bit stream through its serial port to a wireless transmitter. The wireless receiver collects the input bit stream and sends it to PC 2 through its serial port. The decoder program running on PC 2 decodes the H.263 bit stream and displays it to the monitor on PC 2 in YCbCr format. As an alternative to the wireless channel, the prototype is also capable of operating over an Ethernet channel using transmission control protocol (TCP) and Internet protocol (IP). The TCP/IP protocol ensures reliable transmission of the video to any computer with a valid IP address. The user can configure the coder for variable frame rates and bit rates.

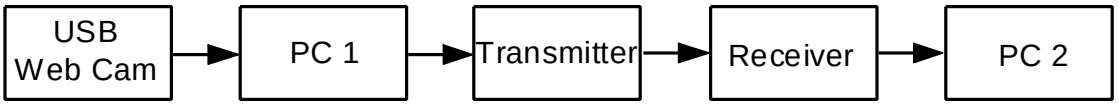


Figure 3.2: Data Flow in H.263 Software Prototype

The software prototype is implemented in the C and C++ languages and based upon an implementation by the multimedia compression (MC) team from Korea University. The core H.263 codec described in Chapter 2 is implemented in C, while the external interfaces to the camera, the video display, and the wireless channel are implemented in C++.

The prototype supplies us with ability to capture input and output data streams at any point in the coding or decoding process. For the DCT and the IDCT VHDL models, we use this data to verify the correct operation of the models. The prototype

also provides a convenient method of implementing new designs and seeing the effect in an actual video sequence. Additionally, we measure the effect of our low power techniques on the number of calculations and on the PSNR. To obtain consistent results, we use data from pre-recorded video sequences (as opposed to live video) for our simulations.

We use three test sequences to extract measurements from the prototype: "Claire", "Foreman", and "Miss America". All three video sequences are in the Quarter–Common Interchange Format (QCIF), which contains 176 x 144 pixels. "Claire" and "Miss America" are low motion video sequences that are typical for videophone applications. They use a stationary camera to record head-and-shoulders shots of a person talking, so the majority of movements are small movements occurring around the head area. The "Foreman" sequence is a high motion video sequence that would be typical for mobile patrol applications. There is a large amount of camera motion, including a large pan. Additionally, the subject exhibits a large amount of head motion as well as hand and arm motion.

3.1.2 VHDL models

3.1.2.1 Synopsys Simulation Environment

From the software prototype, we code an equivalent DCT unit in the VHDL hardware description language. To analyze, simulate, and synthesize the VHDL models of the DCT and IDCT, we use Synopsys version 1999.10. The simulation for both the RTL models and the synthesized models of the DCT and IDCT use input data taken from the software prototype system. We require enough data to draw a correlation between typical H.263 data and power reduction. For simulation, we choose one INTER frame of data selected at random from each of the three video sequences. An INTER frame is appropriate for test data, since only the first frame of a sequence is encoded in INTRA format; the remaining frames are encoded in INTER format. QCIF resolution requires the calculation of 594 DCTs and IDCTs for each frame in the video sequence. In our architecture, one frame requires over 230,000 clock cycles for DCT or IDCT computation. Because of the large number of calculations, it is inefficient to run

simulations at low levels in the design hierarchy such as the transistor level or the layout level. Even the gate level simulation can require up to three days of simulation time for the three sequences, depending on the load of the workstations. Previous simulation of top-level DCT architectures has also been performed at a high level [26].

3.1.2.2 RTL Model

The first step in the VHDL design of the DCT and IDCT circuits is an RTL model in VHDL. The VHDL is written in a style that can be synthesized with Synopsys Design Analyzer. The RTL version is used primarily for verification of the design with the software prototype. Simulation at the RTL level is much faster than simulation at the gate level, and modifications are much easier to make in the RTL level than at the gate level. Any modifications to the RTL description of the circuit are also implemented in the software prototype, and new simulation data and PSNR measurements are taken. The RTL code is analyzed and simulated with Synopsys tools. A test bench, written in VHDL, reads the input data from a file and applies it to the inputs of the DCT or IDCT circuit along with clock, reset, and enable signals. The test bench stores the results in a file, which is compared to a file containing results from the prototype. If the results are inconsistent, we correct the RTL model. Otherwise, we are ready to synthesize a gate level model of the circuit.

Figure 3.3 shows the VHDL file hierarchy of our RTL models. The changes in file names for the IDCT are shown in parenthesis, where different from the DCT file names. The top level file “tbdct2” is the test bench for the design that includes a clock generator and applies all input signals to the top level hardware design. The remaining files contain components of the DCT or IDCT system.

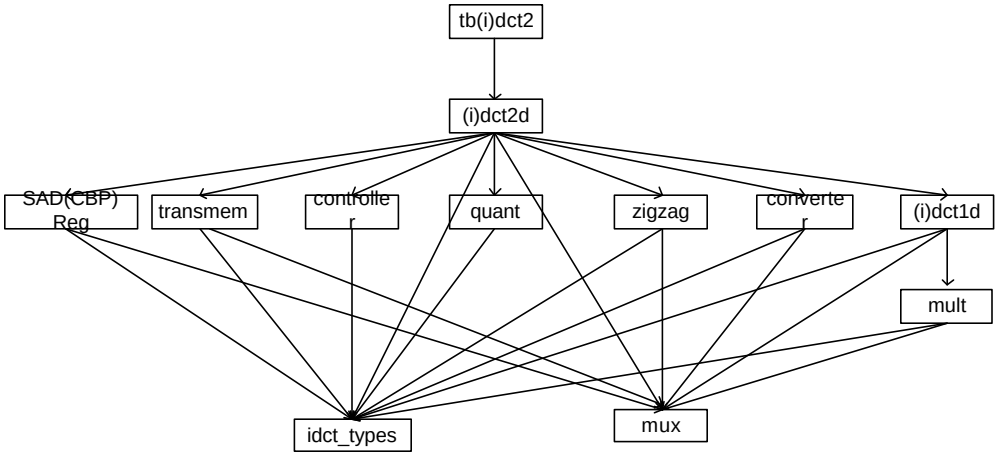


Figure 3.3: 2-D DCT/IDCT File Hierarchy

3.1.2.3 Synthesized Model

From the RTL description, we use Synopsys Design Analyzer to synthesize the design onto our CMOS standard cell library. The synthesized design is verified for correct output as well as power dissipation characteristics. Although less accurate than power estimation at the switch level, a gate level power estimation provides sufficient accuracy to experiment with several low power design techniques. The gate level simulation completes in a reasonable amount of time (3 days or less), and it provides a relative comparison of power dissipation between architectures. With relative comparisons, we identify low power design techniques that will save power over the baseline model.

Figure 3.4 shows the design flow for synthesizing and analyzing the gate level description of the DCT and IDCT circuits. Starting with the RTL description, we use design analyzer to produce a gate level design. Synopsys generates a single VHDL file

to describe all levels in the hierarchy of the gate level design. Simulation of the gate level circuit provides information on the toggle count of each node in the circuit. The toggle information must be converted to a script file so that Synopsys can produce a power report. Since Synopsys simulates the gate level VHDL model, we change the pin names in the synthesized circuit to comply with VHDL naming conventions. Synopsys design Power reads the script file and produces a power report based on the characteristics of our library cells, the gate-level VHDL model, and the script file with toggle information.

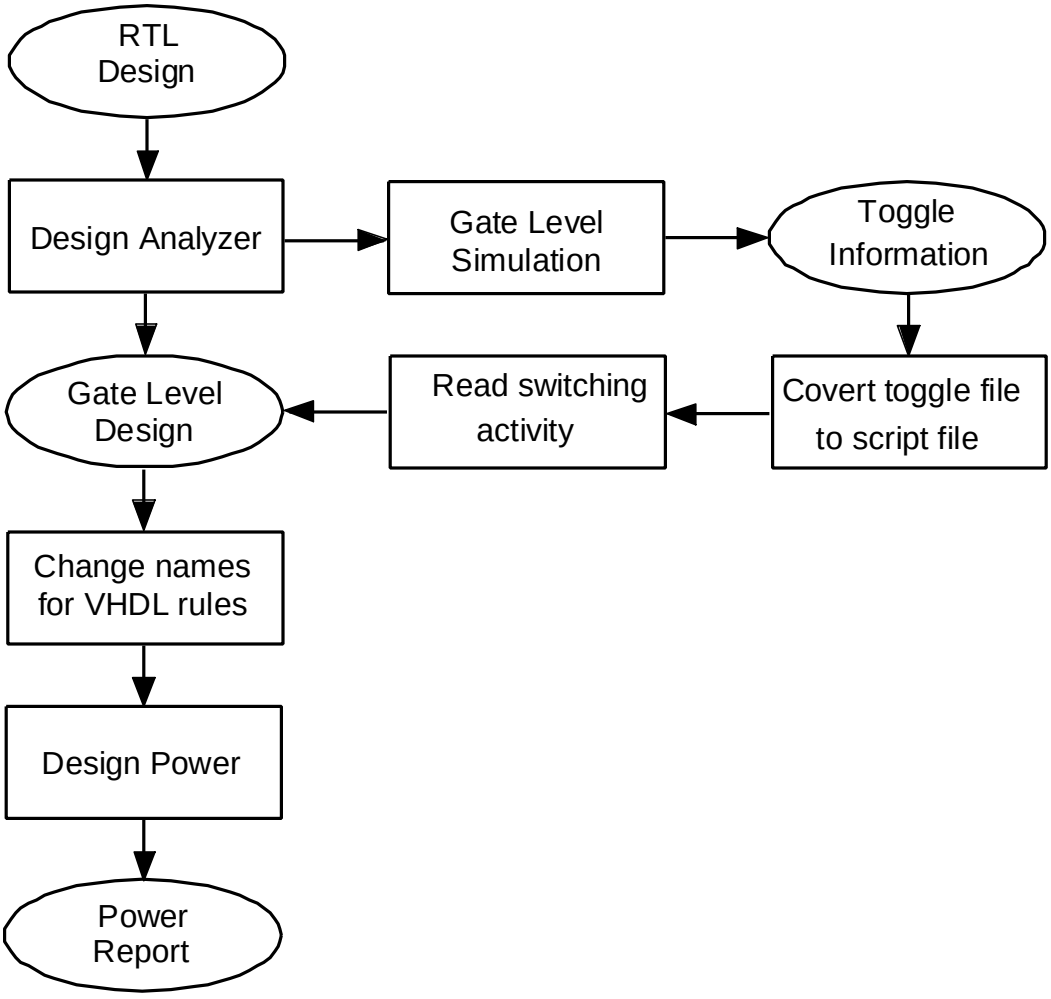


Figure 3.4: Methodology for Gate Level Power Estimation [Synopsys Manual]

Two separate components contribute to power dissipation in a CMOS circuit. The first is static power dissipation, which occurs when there is no switching activity. Static power dissipation results mainly from leakage current and subthreshold conduction current. Static power typically accounts for a small proportion of the overall power dissipation in a CMOS circuit. The second component, dynamic power dissipation, accounts for the majority of power dissipation in a circuit. Dynamic power dissipation results from switching activity in a circuit. For power estimation purposes, dynamic power includes two components: switching power and internal power. Switching power results from charging and discharging the external capacitive loads driven by a cell. Internal power results from short circuit conditions that occur during switching and from the charging and discharging of the internal capacitances within a cell [44].

Synopsys estimates dynamic power based on the following formula

$$P = aC_L V^2 f \tag{3.1}$$

where a is the switching activity, C_L is the capacitive load, V is the supply voltage, and f is the clock frequency. Synopsys Design Power computes the power at each node from the characteristics of the library cells, the toggle information, and knowledge of the supply voltage. The process of measuring power is as follows.

1. Run a simulation with the desired test data on the gate level model. For our simulation, the data is an entire frame in a video sequence.
2. Record the toggle count at each net.
3. Compute the average dynamic power dissipation of each node using Equation (3.1), the cell library characteristics, and the supply voltage.
4. Sum up the power dissipation at each node to obtain the overall power dissipation.

During synthesis, Synopsys maps our RTL code onto a standard cell library. We use a library developed by the Virginia Tech VLSI for Telecommunications (VTVT)

group that is targeted for 0.35µ technology from the chip foundry TSMC. The standard cell library contains the elements described in Table 3.1 The column labeled “Cell” gives the name of the cell, the column labeled “Strength” gives the available relative drive strengths of the cell, and the column labeled “I/O Options” gives the options available for the input and output pins.

Table 3.1 Standard Cell Library Elements

Cell	**Strength**	**I/O Options**
AND gate	1	2,3 inputs
Buffer	1,2,4	1 input
D Flip-Flop	2,4	Synchronous and asynchronous reset, Synchronous and asynchronous set, positive and negative outputs, rising edge
Inverter	1,2,4	1 input
Latch	2,4	Synchronous and asynchronous reset, Synchronous and asynchronous set, positive and negative outputs, positive enable
Multiplexer	2,4	2 inputs
NAND gate	1	2,3,4 inputs
NOR gate	1	2,3,4 inputs
OR gate	1	2,3,4 inputs
XOR gate	2	2 inputs

The cells are developed on three levels: the layout level, the netlist level, and the Synopsys library level. The layout of the cell provides the necessary characterization of a cell’s parasitic capacitances. From a netlist description of the cell, HSpice simulation extracts both timing and power characteristics. These characteristics are ported to a format suitable for Synopsys libraries. The Synopsys library contains information on the timing and power dissipation characteristics of each cell.

3.2 Chosen Algorithm

From the algorithms described in Chapter 2, we have a choice between a row/column approach and a direct 2-D approach. Each contains about the same number of registers and arithmetic units. The direct 2-D approach suffers from an irregular structure, a more complex control unit, and a more complex data path. Additionally, the lower latency of the direct 2-D approach is not beneficial for a target system with a low bit rate. Because of its regularity and simplicity, we choose a row/column approach in our baseline designs. The row/column approach requires three steps: eight 1-D DCTs along the rows of the input, a memory transposition of the results, and another eight 1-D DCTs along the rows of the transposed memory.

For fast operation with a minimum number of multiplications, we base the 1-D DCT and IDCT in our baseline designs on Chen's algorithm [8]. The matrix vector products in Equations 3.2 and 3.3 implement a fast version of the 1-D DCT and IDCT functions. The symbol Y(N) represents a DCT coefficient and the symbol x(n) represents a sample from the spatial domain.

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \cdot \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix}$$

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \cdot \begin{bmatrix} x(0)-x(7) \\ x(1)-x(6) \\ x(2)-(5) \\ x(3)-x(4) \end{bmatrix} \quad (3.2)$$

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \cdot \begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \cdot \begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix}$$

$$\begin{bmatrix} x(7) \\ x(6) \\ x(5) \\ x(4) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \cdot \begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} - \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \cdot \begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} \quad (3.3)$$

Equation (3.4) gives the values of the coefficients in Equations (3.2) and (3.3).

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} = \sqrt{\frac{2}{N}} \cdot \begin{bmatrix} \cos\frac{p}{4} \\ \cos\frac{p}{16} \\ \cos\frac{p}{8} \\ \cos\frac{3p}{16} \\ \cos\frac{5p}{16} \\ \cos\frac{3p}{8} \\ \cos\frac{7p\,0}{16} \end{bmatrix} = \begin{bmatrix} 0.35355 \\ 0.49039 \\ 0.46194 \\ 0.41573 \\ 0.27779 \\ 0.19134 \\ 0.09755 \end{bmatrix} \quad (3.4)$$

We implement the matrix vector products in Equations (3.2) and (3.3) in flowgraph form as shown in Fig 3.5 for the DCT and Fig 3.6 for the IDCT. The data paths are drawn so that there is no more than one multiplication along any path. This increases the accuracy of the computations, since each fixed-point multiplication potentially adds truncation error. The circles represent addition or subtraction

operations, while a coefficient appearing next to a path represents a multiplication by that coefficient. The symbol Y(N) represents a DCT coefficient and the symbol x(n) represents a sample from the spatial domain.

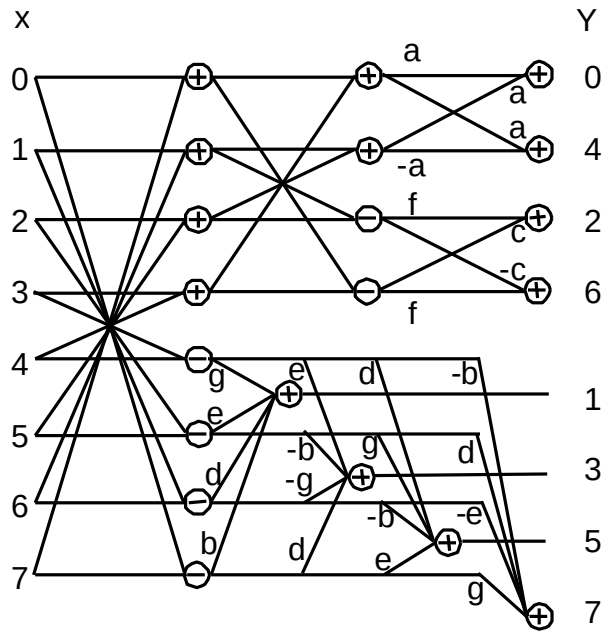


Figure 3.5: DCT flow graph.

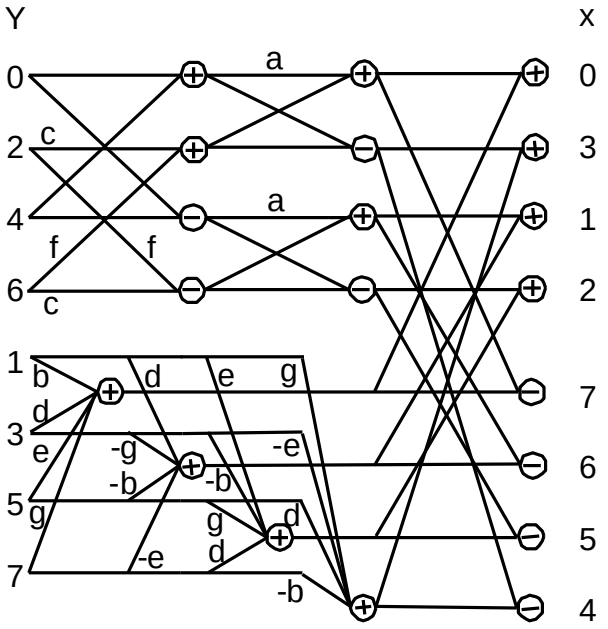


Figure 3.6: IDCT flow graph.

3.3 Architecture

A block diagram of our top-level architecture for the 2-D DCT/IDCT block is shown in Fig 3.7. The main components are a controller, a 1-D DCT/IDCT unit, a transposition memory, a parallel to serial converter (ser2par) unit, and a parallel to serial converter (par2ser) unit. These blocks are described in detail in the sections below.

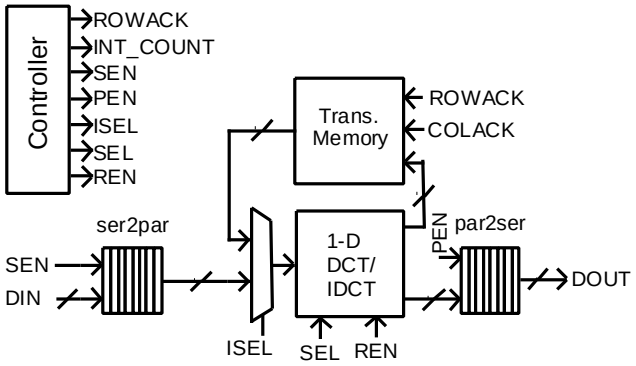


Figure 3.7: 2-D DCT/IDCT Architecture

Table 3.2 gives a description of each I/O pin in the top level architectures of the baseline DCT and IDCT units. Note that the I/O pins are the same for both the baseline DCT and IDCT units.

.

Table 3.2: I/O Signals in the DCT or IDCT unit

Signal Name	**I/O**	**Function**
CLK	I	Clock
RB	I	Active low reset
DIN	I	Serial input coefficients
DOUT	O	Serial output coefficients

3.3.1 Controller

3.3.1.1 Control Signals

Table 3.3 shows the name and function of each I/O signal in the controller. The controller starts the 2-D DCT/IDCT process by enabling input of the first row of data (DIN) through the ser2par unit under the SEN signal. It then activates the 1-D DCT/IDCT unit with the SEL and REN signals determining the data path. The results are stored in the first row of the transposition memory under ROWACK enabled. This process repeats for the remaining seven rows of the input block. Next, the ISEL and COLACK signals enable the 1-D DCT/IDCT unit to receive input data from the columns of the transposition memory. Eight 1-D DCT/IDCTs along the columns of the transposition memory result in a 2-D DCT/DCT operation. The results (DOUT) for the 2-D DCT/IDCT are available through the par2ser unit under PEN enabled.

Table 3.3: I/O Signals in the Controller Unit

Signal Name	**I/O**	**Function**
RST	I	Active low reset
CLK	I	Clock
ROWACK	I/O	Enables transposition memory for row input
INT_COUNT	I/O	Counts total number of 1-D DCTs or IDCTs performed
SEN	I/O	Enables serial input through ser2par
PEN	I/O	Enables serial output through pa2ser
ISEL	I/O	Selects between input from ser2par or from transposition memory
SEL	O	Selects inputs for multiplexers on registers and arithmetic unit inputs in the 1-D DCT or IDCT unit
REN	O	Enables registers in the 1-D DCT or IDCT unit

Internal counters generate the above control signals. One counter controls the flow of a 1-D DCT or IDCT operation. This counter counts from zero to twenty-four, since the 1-D DCT and IDCT each consume a total of twenty-five cycles. The 1-D DCT and IDCT row operations include the following stages:

- Shift input data through the ser2par unit. (8 cycles).
- Latch the input data into the internal registers. (1 cycle)
- Perform the 1-D DCT or IDCT operation. (15 cycles)
- Latch the 1-D DCT or IDCT results into the transposition memory. (1 cycle)

Similarly, the 1-D DCT and IDCT column operations include the following stages:

- Latch data from the transposition memory into internal registers. (1 cycle)
- Perform the 1-D DCT or IDCT operation. (15 cycles)
- Latch data into the ser2par unit. (1 cycle).
- Shift the 1-D DCT or IDCT results through the par2ser unit. (8 cycles)

The other counter tracks the number of 1-D DCTs or IDCTs performed and outputs the results on the INT_COUNT signal. This counter increments when the first internal counter rolls over from twenty-four to zero. The 2-D DCT/IDCT operation requires a total of sixteen 1-D DCT//IDCT operations.

3.3.1.2 Timing of 1-D Operations

From the flow graphs in Figure 3.5 and Figure 3.6, we schedule the operations necessary to perform the 1-D DCT/IDCT operations. The timing of row operations differs slightly from that of column operations, so we include a description of each.

3.3.1.2.1 Row Operation

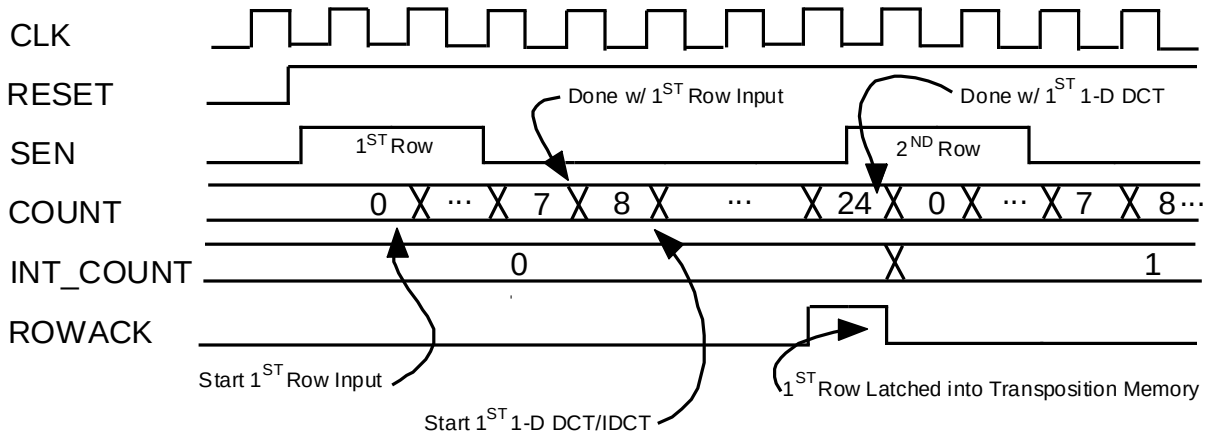


Figure 3.8: Timing of 1-D DCT/IDCT Row Operation

Figure 3.8 shows the timing for the first 1-D DCT/IDCT row operation and for the beginning of the second 1-D DCT/IDCT row operation. The overall process of a 1-D DCT/IDCT on an input row takes twenty-five cycles. With RESET activated (low), the controller deactivates its output signals. Additionally, the internal counters, COUNT and INT_COUNT, maintain a value of zero. COUNT tracks the cycle number within a 1-D DCT/IDCT operation, while INT_COUNT tracks the total number of 1-D DCT/IDCTs performed. The first 1-D DCT/IDCT starts when RESET is deactivated, allowing the internal counters COUNT and INT_COUNT to start counting up from zero. The first eight cycles (COUNT = 0-7) permit the input of data through the ser2par unit under SEN enabled. The input data is ready one cycle later (COUNT = 8), and the 1-D DCT/DCT unit computes the 1-D DCT/IDCT for the next 15 cycles (COUNT = 9-23). The DCT/IDCT coefficients are then clocked into the transposition memory with the ROWACK signal on the last cycle of COUNT (COUNT = 24). For the second 1-D DCT/IDCT, the INT_COUNT counter increments to indicate the occurrence of the second 1-D DCT/IDCT row operation, while COUNT rolls over to zero. The same 1-D DCT/IDCT operations are performed on the second row as the first row. This process continues for six more 1-D DCT/IDCTs. At this point, INT_COUNT reaches eight and signals the start of the column operations.

3.3.1.2.2 Column Operation

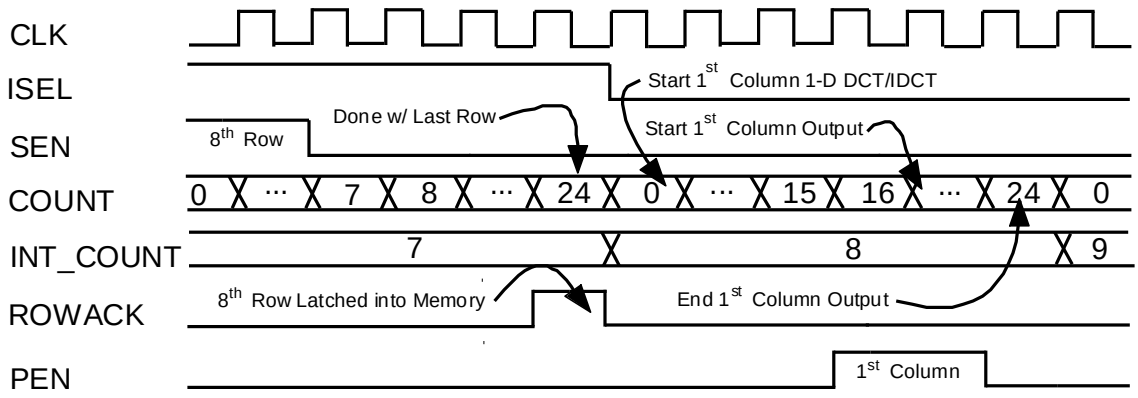


Figure 3.9: Timing of 1-D DCT/IDCT Column Operation

Figure 3.9 shows the timing of the last 1-D DCT/IDCT row operation and the first 1-D DCT/IDCT column operation. The column 1-D DCT/IDCT operation also takes 25 cycles to complete. At the end of the last 1-D DCT/IDCT on the input rows, the ISEL signal goes low, indicating the switch to column operations. When ISEL is low, the 1-D DCT/IDCT unit receives input from the transposition memory, instead of the ser2par unit. Additionally, the internal counter, COUNT, signals a different sequence of operations. First, the data from the transposition memory is latched into the internal registers on the first cycle (COUNT = 0). The next fifteen cycles (COUNT = 1-15) are dedicated to performing the column 1-D DCT/IDCT operation. The next cycle (COUNT = 16) latches the results of the column operations into the par2ser unit. The final eight cycles (COUNT = 17-24) clock the results through the par2ser unit under PEN enabled. The same process applies for the remaining seven columns in the transposition memory. At this point, INT_COUNT reaches fifteen and signals the end of the 2-D DCT/IDCT process.

3.3.1.3 Timing of 2-D Operation

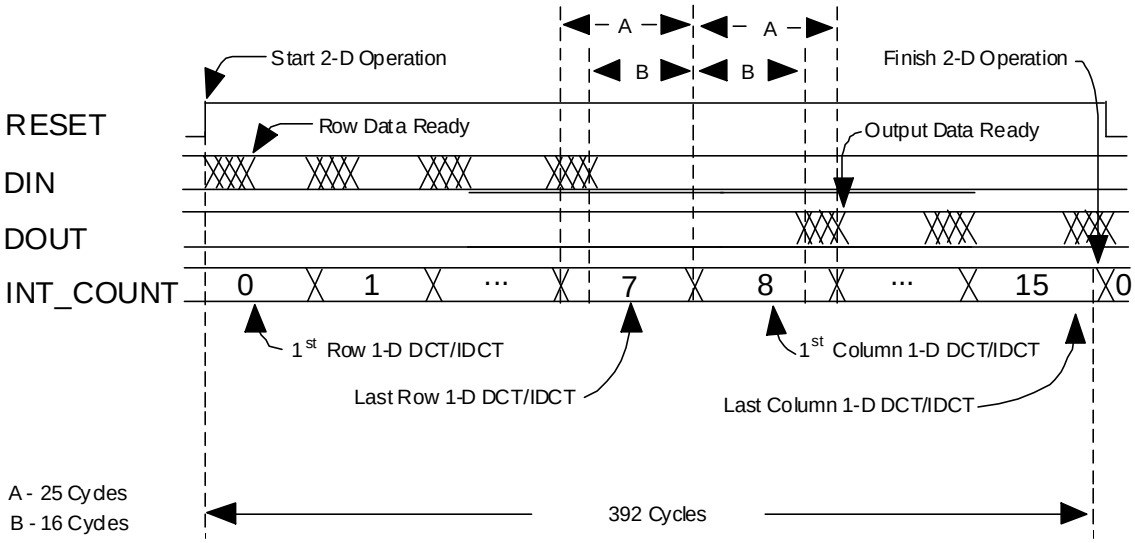


Figure 3.10: Timing of 2-D DCT/IDCT Operation

Figure 3.10 shows the timing for the overall 2-D DCT/IDCT operation. The 2-D DCT/IDCT starts when the reset signal is deactivated (high). The 2-D DCT/IDCT completes 392 cycles later. When reset is deactivated, the data for the first 1-D DCT/IDCT row operation is shifted in through the ser2par unit. Then, a 1-D DCT/IDCT is performed, and the results are stored in the transposition memory. The row operations continue for the remaining seven rows of input data. At this point, the eight column DCT/IDCTs operate on the columns of the transposition memory. Both the row and column operations require 25 cycles (A). Ten clocks are used for input and output operations, while fifteen clocks (B) are used for the computation of the 1-D DCT/IDCT. Table 3.4 shows the times at which each column of output data appears on the DOUT pin in Figure 3.7.

Table 3.4: Clock Cycles when 2-D DCT/IDCT Output is Available

Output Column	Clock Cycles
1	210-217
2	235-242
3	260-267
4	285-292
5	310-317
6	335-342
7	360-367
8	385-392

3.3.2 1-D DCT/IDCT Unit

Since video coding for low bit rates does not require a high-speed architecture, we save area by using a small number of arithmetic units in our baseline 1-D DCT and IDCT designs. This is a different approach than most of the DCT and IDCT implementations in Chapter 2, which are designed with the goal of high throughput to meet the demands of MPEG or HDTV video encoding. HDTV requires a throughput of 75 mega-samples per second, and MPEG requires a throughput of 14 mega-samples per second. In contrast, H.263 QCIF video encoding at 10 frames/sec requires a throughput of only 380 kilo-samples per second. Therefore, our implementation need not be concerned with a parallel architecture, a pipelined architecture, strict timing constraints, or high clock rates.

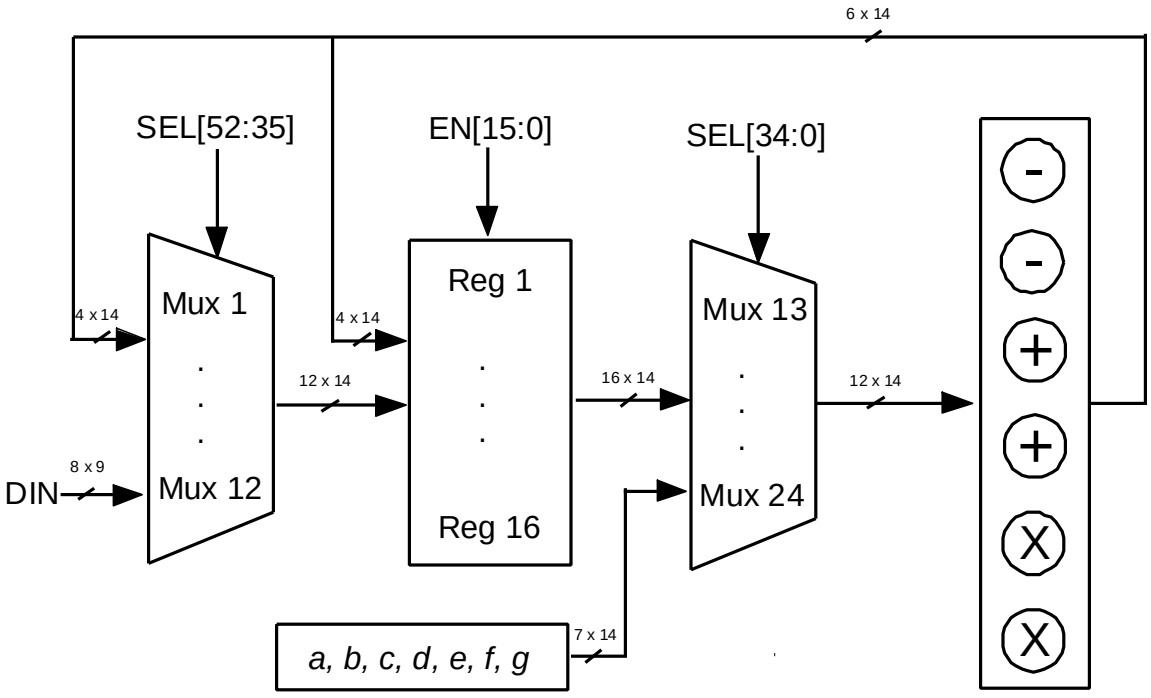


Figure 3.11: Architecture of 1-D DCT/IDCT Unit

Figure 3.11 shows a block diagram of the architecture of the 1-D DCT/IDCT unit. It is composed of six arithmetic units, sixteen registers (fifteen for the IDCT), and twenty-four multiplexers. The arithmetic units perform addition, subtraction, and multiplication operations. The registers store the input data, the intermediate values, and the final results. The multiplexers select the source data for the arithmetic units and the registers. Input data is first latched into the register bank through eight of the multiplexers in the first multiplexer bank. The second multiplexer bank selects the inputs for each of the six arithmetic units, if necessary. The second multiplexer bank can also select inputs from the constants in Equation 3.4. It is important to note that not every arithmetic unit nor every register receives input data on every cycle. Each arithmetic unit has two inputs and one output that are each the width of the internal data path (14 bits).

Table 3.5 shows each I/O pin in the top level architectures of the baseline 1-D DCT and IDCT units. Note that the I/O pins are the same for both the baseline 1-D DCT and IDCT units.

Table 3.5: I/O Signals 1-D DCT or IDCT unit

Signal Name	**I/O**	**Function**
CLK	I	Clock
RB	I	Active low reset
IN87-80	I	Eight parallel inputs for the 1-D DCT or IDCT operation
RE	I	Enables for internal registers
SELM	I	Selects inputs for multiplexers on internal registers
SELR	I	Selects inputs for multiplexers on the arithmetic units
OUT87-80	O	Eight parallel outputs of the 1-D DCT or IDCT operation

3.3.2.1 Arithmetic Units

We allocate two multipliers, two adders, and two subtractors for the 1-D DCT/IDCT unit. With these resources, a 1-D DCT/IDCT operation requires fifteen clock cycles. With the additional ten clock cycles required for I/O operations, the 1-D DCT/IDCT process takes twenty-five cycles, and the overall 2-D DCT/IDCT process takes 392 clock cycles. With 594 DCT/IDCT operations per frame, our targeted bandwidth of ten frames per second requires a clock frequency of 2.5 MHz. Our VTVT 0.35µ library easily reaches this frequency. For faster operation, experiments reveal that the frequency can increase by an order of magnitude or possibly more.

The adders and subtractors are implemented from library components in the Design Ware library. We make no effort to design low power adders or subtractors, since some preliminary experiments indicate that they account for an insignificant portion (less than 1%) of the total power in the baseline unit. The adders and

subtractors have a 14-bit width for the input operands and for the output results. The 14-bit internal width is necessary to avoid overflow in the DCT and IDCT operations.

We chose to implement the multipliers as fixed-point, array multipliers, which are fast and readily available from the Design Ware library. We make no special effort to implement low power multipliers, since preliminary experiments reveal that the array multipliers consume less than 10% of the total power dissipation in the baseline design.

A major concern for implementing the multipliers is accuracy. A significant factor in the accuracy of the multipliers is the precision of the coefficients. The DCT has no accuracy requirements imposed by H.263 because of the high distortion introduced by the quantization process. Previous DCT implementations have set the coefficient precision equal to the internal data width, which must be at least fourteen bits to avoid overflow [19]. For the IDCT to adhere to H.263 standards, it must follow the strict accuracy requirements set forth by the IEEE 1180-1990 standard. A brief description of the IEEE 1180-1990 standard for IDCT accuracy is given below.

1. Generate 10,000 8x8 blocks of random data for each range (-256, 256), (-5, 5), and (-300, 300).
2. For each of the above blocks, perform DCTs using the straightforward approach (Equation (2.2)) with 64-bit floating-point accuracy.
3. Round each of the resulting coefficients to the nearest integer and clip to the range (-2048, 2047) to get 12-bit input data for the IDCT.
4. Perform an IDCT on the input from step (3) using the straightforward approach (Equation (2.3)) with 64-bit floating-point accuracy. Round the results to the nearest integer and clip to the range (-256, 255). This is the reference result.
5. Perform an IDCT on the input from step (3) with the IDCT under test and clip the results to the range (-256, 255). This is the test result.
6. For each pixel in each set of 10,000 blocks, measure the peak, mean, and mean square error of the test results with respect to the reference results.
7. The error between the test and reference results should be less than the maximum errors in table 3.6.

Table 3.6 Maximum IDCT error to comply with IEEE 1180-1990

Measurement	**Maximum**
Peak error	1
Mean Square Error (one pixel)	0.06
Mean Square Error (overall)	0.02
Mean Error (one pixel)	0.015
Mean Error (overall)	0.0015

8. All zero input should produce all-zero output.
9. Repeat steps 1-7 with the sign of each pixel changed.

Most IDCT implementations [14][38][39] use enough coefficient precision to comply with the IEEE standard for IDCT accuracy. For low-power, low bit-rate video coding, the IEEE standard seems too strict a guideline to adhere to, since it is meant to ensure stability for systems with different IDCT architectures in the coder and the decoder. If we restrict our system to use matching IDCT architectures in both the coder and the decoder, the video will not suffer any noticeable degradation.

In the baseline DCT and IDCT units, we set the coefficient precision equal to the internal data width (14 bits), since the array multipliers from the library require both multiplicands to be the same width. Additionally, this precision has been previously chosen to balance accuracy and area [43] in a DCT architecture. In the prototype, we implemented our IDCT architecture in the coder and the decoder without regard to the IEEE accuracy standards. Since both the IDCT in the coder and the IDCT in decoder share the same architecture, the video remains stable. This results from the fact that both the coder and the decoder use exactly the same reference frame to reconstruct images.

A multiplier always multiplies one of the constants (*a, b, c, d, e, f,* and *g*) by one intermediate value from a register. The constants are implemented as the binary representations of the constants in Equation 3.4. These binary constants are shifted left

fourteen places to provide an integer input for the array multiplier. The resulting constants are represented in 14 bits. The other input comes from one of the 14-bit intermediate values stored in an internal register. Since the array multiplier operates on two 14-bit input multiplicands, the result will be a 28-bit number. The 28-bit result is then shifted right by fourteen places to compensate for the initial shift left by fourteen places in the constants. The final result is a 14-bit number that is the floor of pre-shifted result. Because of the right shift operation, each multiplication can potentially result in a truncation error of less than one. Since multiplications potentially result in truncation error, we attempt to perform as few multiplications as possible. Note that the flow graphs in Figures 3.4 and 3.5 have no more than one multiplication along any path, thus resulting in less accumulation of error along a path.

3.3.2.2 Data Path

The SEL and EN signals determine the data path. Each register receives input from the arithmetic units through the first bank of multiplexers in Figure 3.11. The multiplexers select the inputs to the registers thought the SELM signal from the controller. A register is updated only when its EN signal is active. Each register provides output to an arithmetic unit through the second bank of multiplexers in Figure 3.11. Each arithmetic unit receives input from two multiplexers in the second bank of multiplexers. The multiplexers select the inputs to the arithmetic units with the SELR signal from the controller. The DCT uses 16 internal registers to store intermediate values, while the IDCT requires 15 internal registers to store the inputs, the intermediate values, and the final results.

3.3.3 Transposition Memory

The transposition memory is an 8x8 array of registers that receive input in a row-wise fashion and provide outputs in a column-wise fashion, thus performing a matrix transposition. The results of the first eight 1-D DCT/IDCTs are stored in the

rows of the transposition memory. For final eight 1-D DCT/IDCTs, the columns of the transposition memory provide data to the 1-D DCT/IDCT unit. Table 3.7 shows the I/O signals for the transposition memory

Table 3.7: I/O Signals for the Transposition Memory

Signal Name	**I/O**	**Function**
CLK	I	Clock
RB	I	Active low reset
IN87-80	I	Eight parallel results from the 1-D DCT or IDCT operation
RACK	I	Enables row input
CACK	I	Enables column output
INT_COUNT	I	Determines which row to place the output of the 1-D DCT/IDCT operations
OUT87-80	O	Eight parallel input to the 1-D DCT or IDCT operation

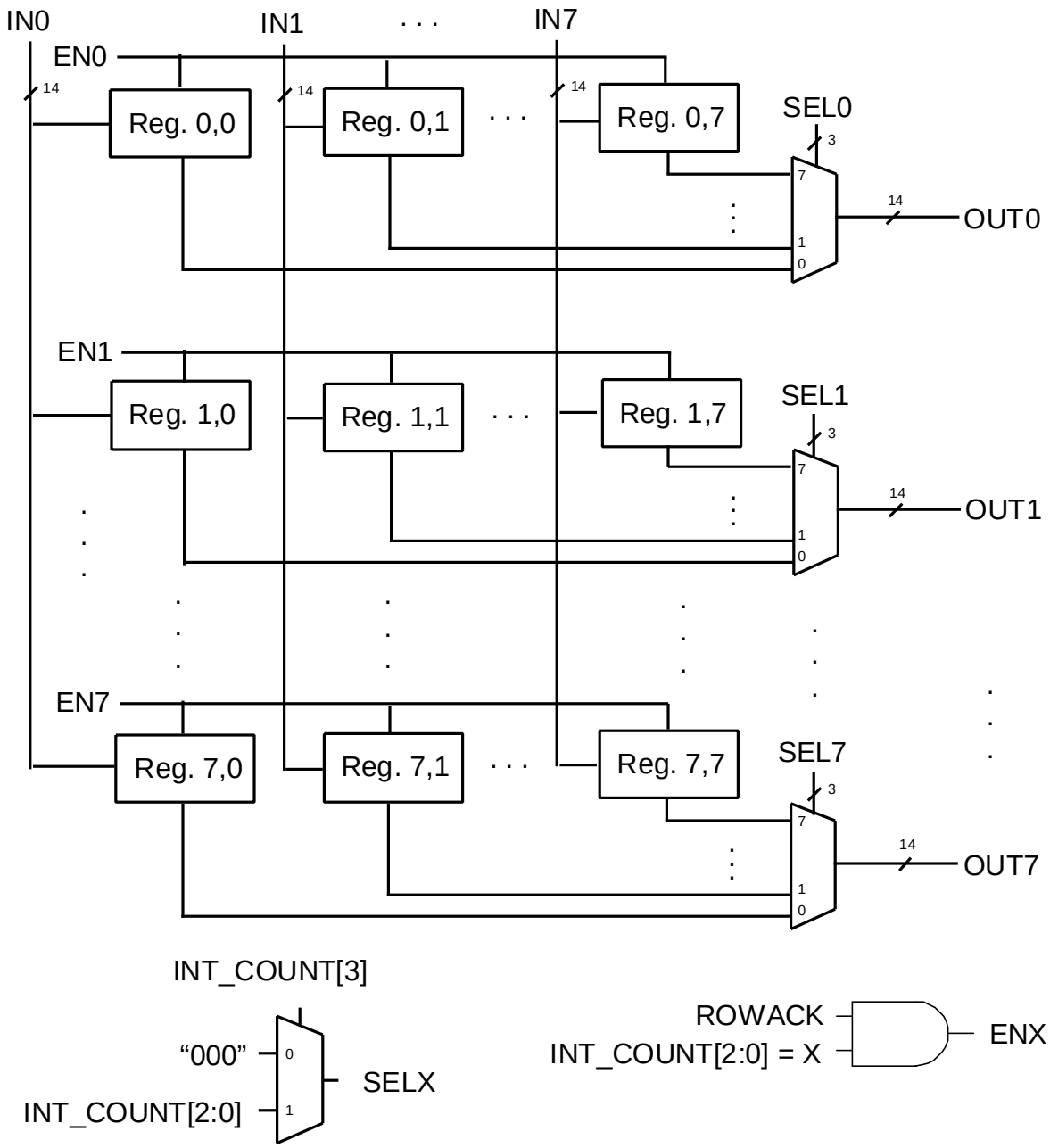


Figure 3.12: Architecture of Transposition Memory

Figure 3.12 shows the architecture of the transposition memory. Each row of eight registers receives input from the eight outputs of the 1-D DCT/IDCT unit. The first row receives input from the first 1-D DCT/IDCT, the second row receives input

from the second 1-D DCT/IDCT, and the pattern continues until the eighth row. At the completion of the 1-D DCT/DCT operation number X, row number X of the registers in the transposition memory is enabled for input by the ENX signal. X corresponds to the number of elapsed 1-D DCT/IDCT row operations. ENX is generated with the INT_COUNT signal and the ROWACK signal. The ROWACK signal is high during the last clock cycle of each of the first eight 1-D DCT/IDCTs. The registers receive no input for the last eight 1-D DCT/IDCT operations; they provide input to the last eight 1-D DCT/IDCT operations. The output multiplexers select one element from each row, depending on the value of the SEL signal. The SEL signal takes on the value of the three least significant bits of INT_COUNT when the most significant bit of INT_COUNT is '1'.

3.3.4 I/O Units

3.3.4.1 Serial to Parallel Converter

To reduce the number of input pins, we choose to implement the DCT/IDCT with serial input. Unfortunately, the 1-D DCT and IDCT units require eight parallel inputs. The ser2par unit in Figure 3.13 converts serial input data to parallel input data. Prior to the start of each of the first eight 1-D DCT/IDCTs, the ser2par shifts in eight values (SIN) through its eight serially connected registers. The data is shifted in under SEN enabled for eight clocks, after which the eight outputs (DIN[8:0]) are available for parallel input to the 1-D DCT/IDCT unit. To accommodate maximum input values, the ser2par registers must be 9 bits wide for the DCT and 12 bits wide for the IDCT.

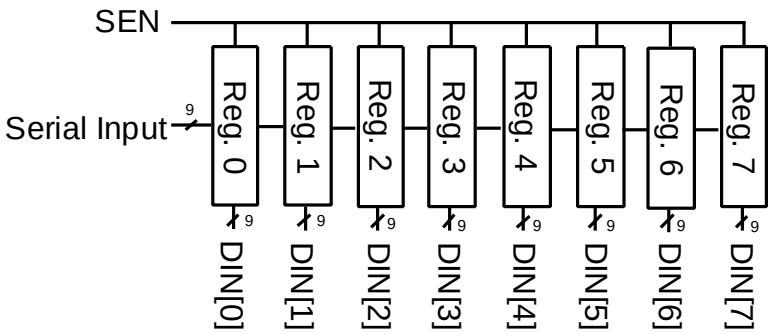


Figure 3.13: Architecture of Serial to Parallel Converter

Table 3.8 shows each I/O pin in the ser2par unit. The I/O pins are the same for both the baseline DCT and IDCT units.

Table 3.8: I/O Signals in the Ser2par Unit

Signal Name	**I/O**	**Function**
CLK	I	Clock
RB	I	Active low reset
EN	I	Enables serial input through the resisters
INPS	I	Serial input data
OUTP7-0	O	Eight parallel outputs for the 1-D DCT or IDCT operation

3.3.4.2 Parallel to Serial Converter

To reduce the number of output pins, we choose to implement the DCT/IDCT with serial output. However the 1-D DCT/IDCT produces its eight outputs in parallel. The par2ser unit in Figure 3.14 converts parallel output data to serial output data. The par2ser unit is active for the last nine cycles in each of the last eight 1-D DCT/IDCT operations. It latches in the results (DOUT[8:0]) of each 1-D DCT/IDCT to its eight registers under PEN disabled. For the next eight cycles, it shifts the results out serially under PEN enabled. With PEN enabled, each register's input will be the output of the

previous register. Thus, the data is shifted through the par2ser unit until each output appears and appears on SOUT. To accommodate maximum output values, the par2ser unit is 12 bits wide for the DCT and 9 bits wide for the IDCT.

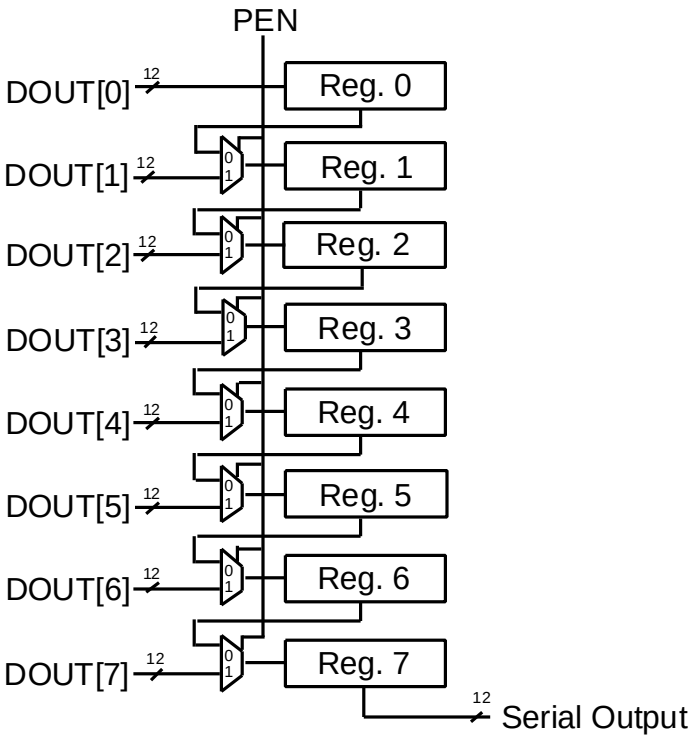


Figure 3.14: Architecture of Parallel to Serial Converter

Table 3.9 shows each I/O pin in the par2ser unit. The I/O pins are the same for both the baseline DCT and IDCT units.

Table 3.9: I/O Signals in the Par2ser Unit

Signal Name	**I/O**	**Function**
CLK	I	Clock
RB	I	Active low reset
EN	I	Enables input through the resisters
INP7-0	I	Eight parallel inputs from the 1-D DCT or IDC unit
OUTS	O	Serial output

Chapter 4: Proposed Low Power DCT/IDCT Design

In this chapter, we explain our proposed methods to save power for DCT and IDCT blocks in low bit rate H.263 video systems. We try five different methods. The first involves testing the input to the DCT to see of it has low enough energy to skip the DCT operation. The next method is to test for all-zero input blocks to the IDCT, which can also be skipped. Additionally, we explore the effects of clock gating in each unit that contains registers: the transposition memory, the 1-D DCT/IDCT unit, the ser2par unit, and the par2ser unit. The fourth method is to implement power efficient shift-and-add multipliers that use lower precision coefficients. Finally, we attempt to reduce transitions in the data paths of the 1-D DCT and IDCT units.

We also characterize the efficiency and the side effects, if any, of each proposed low power method. Since some methods (skipping the DC operation and lower precision coefficients) change the output of the coder, we measure the change of the PSNR under the employment of these methods. A method is employed only if the degradation of the picture quality is unnoticeable to human eyes, and the degradation of the PSNR is small. For the PSNR measurements and the visual examination, we use the software prototype H.263 system that was explained in Chapter 3. The prototype was also used to collect statistical data for the three common benchmarking video clips: Claire, Miss America, and Foreman. To review these clips, Claire and Miss America have little motion with a stable camera, while Foreman has frequent motion and camera movement. The remainder of this chapter details our investigation of low power design techniques for the DCT and IDCT blocks.

4.1 Skipping Input Macroblocks for the DCT

4.1.1 DCT Input Macroblock Characteristics

Typical video data changes very little from the previous frame, since the motion compensation unit sends only the difference from the previous frame. Many input

macroblocks to the DCT contain little new information. The result is small magnitude DCT coefficients that are likely to be quantized to zero. Experimental results show that the three test sequences have up to 80% of macroblocks that are quantized to zero. If we can find an efficient method of predicting these macroblocks, the DCT (and the quantization operation) can be skipped; all DCT coefficients are simply set to zero. The prediction technique need not be perfect, since there is a large amount of allowable distortion in the DCT of a low bit rate system. This results from the fact that low bit rates result in larger QUANT parameters, so the DCT coefficients will have a large quantization error in comparison with any DCT error.

Because of the characteristics of video data in low bit rate systems, we can approximate many DCT macroblocks as all zero. This approximation of the DCT will avoid many computations and improve power dissipation if we disable the DCT unit for these macroblocks. In fact, this method was suggested to speed up DCT operations [30][31], and we propose employment of this method to save power by disabling the DCT unit for these macroblocks. To disable the top level DCT unit, we gate the clock signal, deactivate the enable signal, and activate the reset signal. In the disabled state, the DCT produces the desired all-zero output. Note that this method is only applicable to the DCT circuit.

4.1.2 Methods of Predicting Skipped Macroblocks

For efficient power savings, the method of predicting all-zero macroblocks must be simple and fast. Additionally, the method must not consume significant hardware overhead, and it must dissipate significantly less power than a DCT computation. The method must also predict blocks accurately enough to avoid a noticeable degradation in video quality and PSNR. If too few macroblocks are predicted to be all-zero, then we will not save sufficient power. If too many macroblocks are erroneously predicted as all-zero, then noticeable degradation in video quality and PSNR may result.

Several methods, based on the energy of the incoming data, have been proposed to eliminate the DCT operation in a video system. An accurate method of predicting

all-zero macroblocks is to calculate the variance of the incoming data, which is very computationally intensive [45]. A better method in [46] bases its prediction on the value of the DC coefficient, which also requires some energy consuming calculations. Calculating the DC coefficient requires 64 additions and a shift operation. A more efficient method proposes to examine the SAD value of incoming macroblocks [45]. The SAD value provides a good measure of the energy of the incoming pixels and is readily available from the motion estimation unit. Recall, from Chapter 2, the SAD is a measure of how well the current macroblock matches a macroblock from the previous frame; a low SAD value indicates a good match. Macroblocks with low SAD values tend to produce little new information and are more likely to be quantized to zero.

Since a higher quantization parameter results in more coefficients being set to zero, a good prediction scheme should also consider the quantization parameter along with the SAD value. A macroblock is likely to be set to zero if the block has a low SAD value and a high QUANT parameter. Therefore, we propose skipping a macroblock if

$$SAD < THRESHOLD \times QUANT \qquad (4.1)$$

With this criterion for skipping macroblocks, a macroblock with a low SAD and a high QUANT is more likely to be skipped. Additionally, this method provides a simple, quick, and efficient technique of predicting an all-zero macroblock. The SAD and QUANT signals are readily available from other blocks in the system, and the calculation is straightforward and requires little hardware overhead. To simplify the computational complexity of our method, we restrict THRESHOLD to be a power of two. With this restriction, we need only shift the bits in the QUANT signal. Shifting is a trivial hardware operation, and it eliminates the need for a costly multiplier that consumes area and power. The only additional hardware we require is a comparator.

4.1.3 Effect on H.263 System

Using the software prototype system, we examine the effect of different values of THRESHOLD on the PSNR and the number of skipped macroblocks. Table 4.1 shows the experimental results from the prototype system. The percentage of skipped blocks is expressed as the number skipped by our prediction method as compared to the total number of macroblocks in the sequence. This is an estimate of the power savings, as the DCT unit will be inactive for these blocks.

Table 4.1: Percentage Of Skipped Macroblocks Versus PSNR Degradation

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	PSNR	**Skipped Blocks**	**PSNR**	**Skipped Blocks**	**PSNR**	**Skipped Blocks**
Baseline	40.53		29.64		40.52	
THRESHOLD=16	40.53	0.70%	29.64	0.50%	40.51	3.30%
THRESHOLD=32	40.51	8.30%	29.65	2.60%	40.44	23.8%
THRESHOLD=64	40.39	53.9%	29.56	15.5%	40.20	51.4%
THRESHOLD=128	39.41	79.8%	29.41	57.8%	40.08	76.4%
THRESHOLD=256	38.12	91.4%	29.40	85.7%	38.13	91.7%
THRESHOLD=512	36.72	96.1%	29.40	91.2%	37.77	95.9%

As expected, more blocks are skipped for higher thresholds at the expense of increased PSNR degradation. Fewer macroblocks are skipped for “Foreman,” since it contains more motion and, hence, higher SAD values. From Table 4.1, we see that a significant number of macroblocks can be skipped using our prediction method, depending on the value of THRESHOLD. Additionally, the PSNR does not decrease significantly (more than 3dB) for most values of THRESHOLD. These results indicate that this method is a good candidate for a low power design.

However, aside from the effect on PSNR, we would like to know the accuracy of our prediction method. Table 4.2 shows experimental results on the percentage of correctly predicted macroblocks for different values of THRESHOLD. The first column for each video clip shows the percentage of macroblocks that we correctly predicted to be zero as compared to total number of macroblocks that we predicted to be zero. This gives an idea of accuracy of the technique. Setting too many macroblocks to zero (that would not otherwise be zero) can seriously degrade picture quality. The second column shows the number of macroblocks that we predicted to be zero as compared to the number of macroblocks that should be zero. This is measure of the wasted computation in the DCT unit that results from a THRESHOLD parameter that is too low. In this case, the DCT unit will be active for a macroblock, where it could be disabled. It is desirable for both parameters to be close to 100%.

Table 4.2: Accuracy of All-Zero Macroblock Prediction

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	Correctly Predicted MB's	**Total MB's Predicted**	**Correctly Predicted MB's**	**Total MB's Predicted**	**Correctly Predicted MB's**	**Total MB's Predicted**
THRESHOLD=16	99.7%	1.31%	100%	0.91%	100%	5.8%
THRESHOLD=32	99.3%	9.72%	99.9%	4.36%	99.9%	41.2%
THRESHOLD=64	97.4%	74.6%	98.5%	23.6%	97.8%	69.9%
THRESHOLD=128	92.6%	95.9%	86.6%	81.8%	90.6%	93.5%
THRESHOLD=256	84.6%	99.8%	73.6%	99.9%	79.9%	99.9%
THRESHOLD=512	80.8%	100%	72.6%	100%	76.9%	100%

As expected, the number of correctly predicted macroblocks decreases with increasing THRESHOLD. However, increasing the THRESHOLD also means we predict a larger proportion of the macroblocks that should be zero. For both parameters to be as large as possible, the THRESHOLD value should be 128. From Table 4.1, note

that this THRESHOLD value also results in a small decrease in PSNR and a large percentage of skipped macroblocks. Additionally, with THRESHOLD = 128 we observe no noticeable degradation in video quality for all three video clips. Hence, we set THRESHOLD to 128 in our low power DCT design. Also note that the prediction method is not nearly as accurate for the high motion "Foreman" video sequence. However, we still skip a significant number of macroblocks with little effect on video quality.

4.1.4 Description of Additional Logic

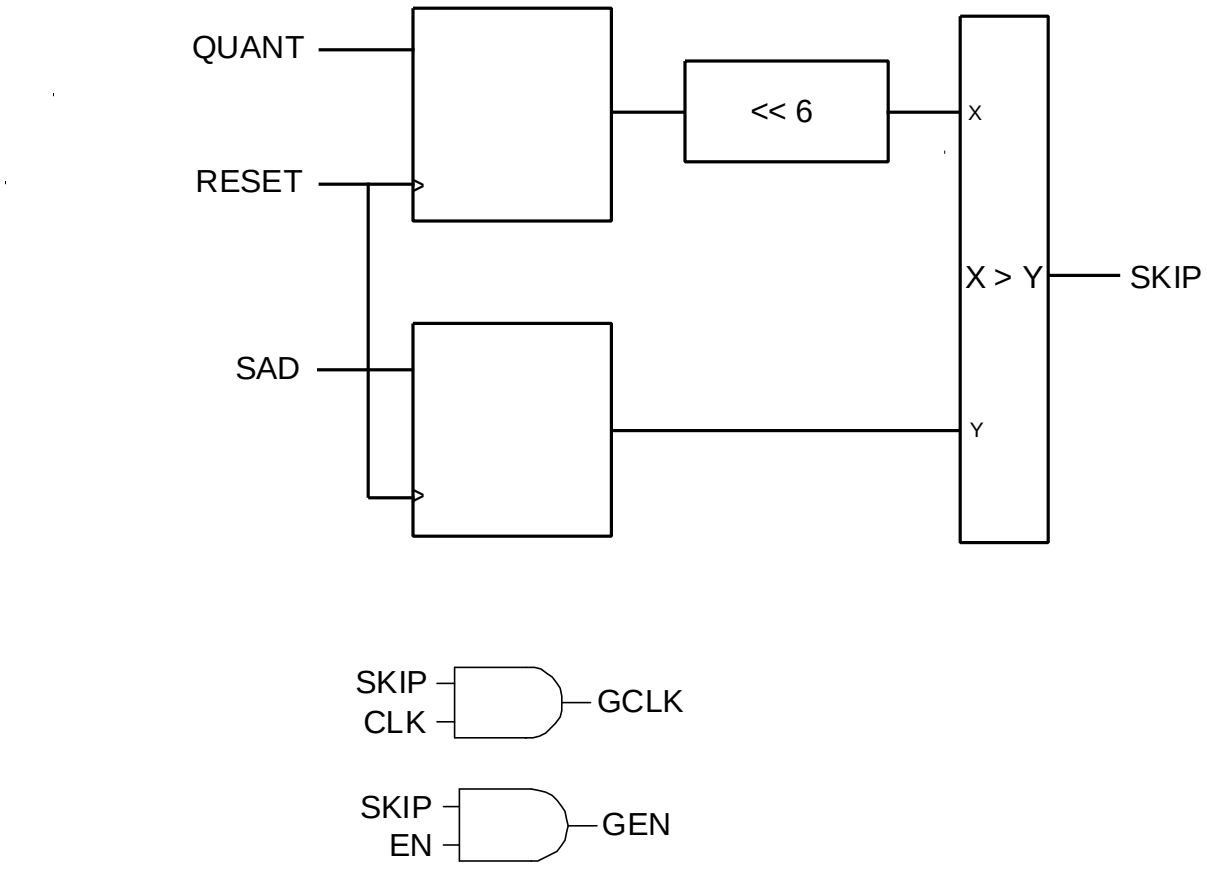


Figure 4.1: Additional Hardware for Skipping All-Zero Macroblocks in the DCT

To implement this prediction method in our baseline DCT architecture, we add the hardware block shown in Figure 4.1. The QUANT and SAD parameters are clocked into their respective registers on the rising edge of RESET, which occurs at the start of every 2-D DCT cycle, and, hence, at the beginning of every block. Since the SAD and QUANT parameters are fixed for an entire macroblock, we clock in redundant values for the last five blocks in a macroblock. To avoid these redundant inputs, we would need an additional counter to track the six blocks in each macroblock. To avoid this additional counter, we allow the input of the SAD and QUANT parameters on every block, and no harm is done. The SAD parameter is then shifted left by seven bits to perform the multiplication by 128. To perform the shift, the six least significant bits of the X input to the comparator are set to zero, and the five most significant bits receive the QUANT value. The comparator processes the scaled QUANT value and the SAD value. An output of '1' indicates that our test condition ($SAD < THRESHOLD \times QUANT$) is false. This suggests the input data contains a significant amount of energy, and it should undergo the DCT operation. However, if our test condition is true, SKIP is set to '0', and it gates the system CLK, RB, and EN signals. With SKIP = '0', no register receives a CLK input, and all registers produce zero as output, which is the desired output for a predicted block.

4.2 Skipping Input Blocks for the IDCT

4.2.1 IDCT Input Block Characteristics

In low bit rate video coding, the large quantization parameters result in a large number DCT output coefficients that are quantized to zero. After reconstruction, these coefficients remain zero, resulting in a large number of zero-valued inputs to the IDCT. In fact, many entire blocks of IDCT input data are zero. Since an IDCT results in all-zero output for such an input block, the IDCT unit can be disabled for all-zero input data blocks. To disable the IDCT unit, we gate the clock signal, deactivate the enable signal, and activate the reset signal. In the disabled state, the IDCT unit produces all-zero output. Note that this method is only applicable to the IDCT unit.

4.2.2 Method of Detecting Skipped Blocks

Detection of all-zero IDCT input blocks is performed in a simple and efficient manner; no prediction is necessary, and the method is 100% accurate. The decoder simply parses the H.263 bitstream to detect an all-zero input block. When COD = '1', the current macroblock contains only all-zero blocks. Additionally, an input block will be all-zero if both the CBP field and the DC coefficient of the block are zero. These parameters are easily extracted from the H.263 bit stream. For the IDCT in the encoder, the VLC unit can simply signal the occurrence an all-zero output block (the VLC already tests for this condition during normal operation). Since this method will not change any data in the encoder or the decoder, no degradation of PSNR or perceived quality results. Detection of all-zero blocks is applied at the block level; a non-zero coefficient in a block causes an IDCT operation on one block only, not the entire macroblock.

4.2.3 Number of Blocks Skipped

We use the software prototype system to measure the percentage of all-zero input blocks to the IDCT. The results in Table 4.3 show that a significant number of input blocks are all-zero. Note that we can skip a larger percentage of IDCT operations than DCT operations, since the decision is made at the block level for the IDCT (instead of the macroblock level for the DCT). Also, note that this method has no effect on PSNR or perceived picture quality. Because of the large number of skipped blocks and the simplicity of detecting these blocks, we expect this technique to result in significant power savings.

Table 4.3: Percentage of All-Zero IDCT Input Blocks

	Claire	Foreman	Miss Am.
Percentage All-Zero Input Bocks	97.0%	93.8%	96.3%

4.2.4 Description of Additional Logic

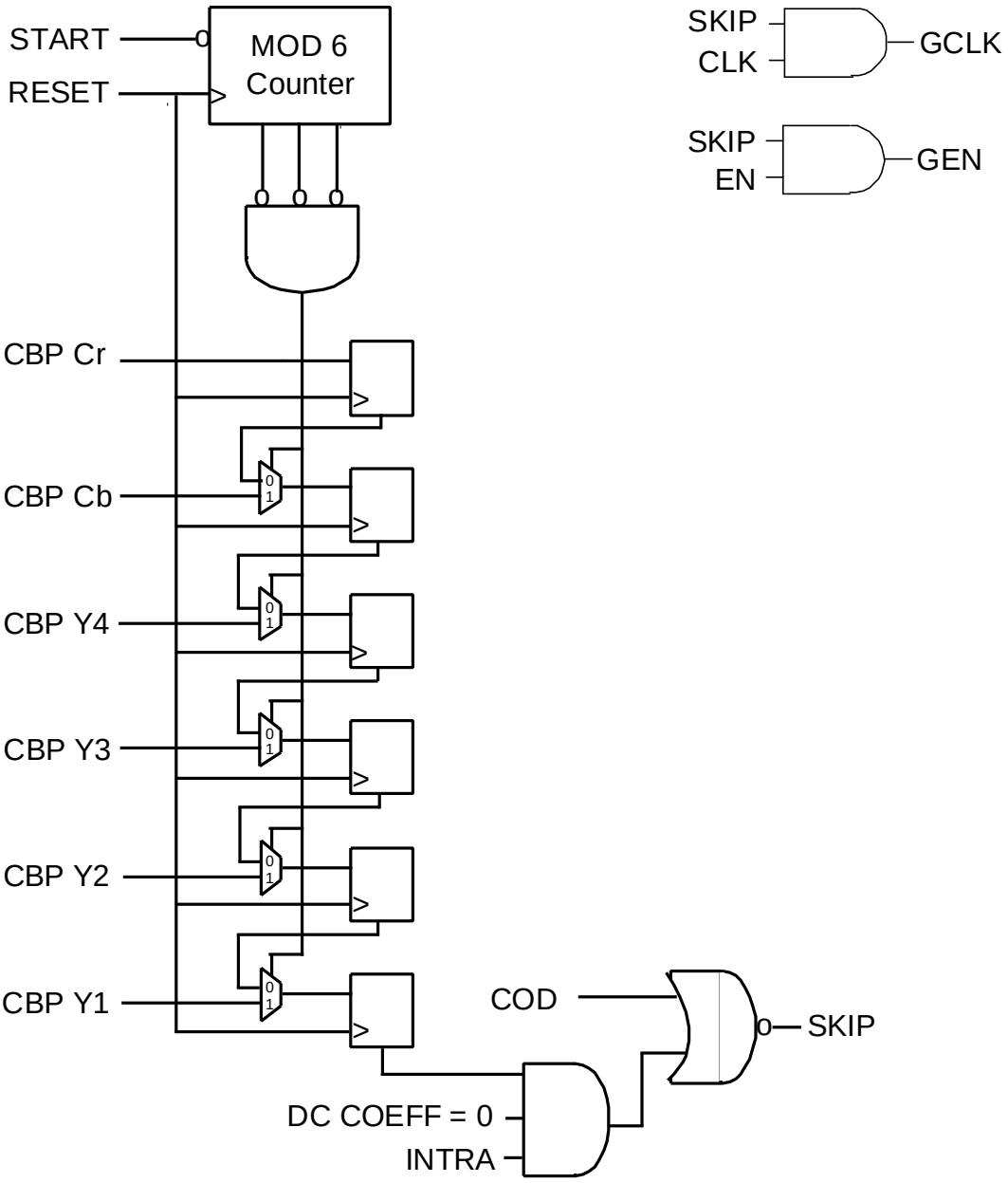


Figure 4.2: Additional Hardware for Skipping All Zero Blocks in the IDCT

Figure 4.2 shows the additional hardware necessary to skip all-zero input blocks for the IDCT in the decoder. The IDCT in the encoder simply receives a SKIP signal from the VLC unit. Prior to the start of each macroblock in the decoder, the mod 6 counter is reset to zero. At the start of each macroblock, the START signal latches the CBP for the macroblock into the six flip-flops. Since the mod 6 counter contains a value of zero, the flip-flops receive the CBP as input. As each successive block is available to the IDCT, the controller enables the IDCT unit by de-asserting the RESET signal, which also causes the mod 6 counter to increment. When it increments past its initial state, the mod 6 counter forces each flip-flop to receive input from the previous flip-flop. For the next five cycles, the CBP for each block ripples through the six flip-flops. In turn, each CBP serves an input to the NAND gate that determines the value of the SKIP signal. The current block is skipped if its CBP is zero, the DC coefficient is zero, and the current block is an INTER block. Alternatively, if the COD signal is '0', then the SKIP signal is active for the entire macroblock. As in the DCT, the SKIP signal gates the CLK, EN, and RB signals in the top level architecture, thus disabling the IDCT unit and forcing all outputs to zero.

4.3 Gated Registers

Four units (the two I/O units, the 1-D DCT/IDCT unit, and the transposition memory) of the baseline DCT/IDCT block in Figure 3.7 contain over 99% of the flip-flops of the block. Since these registers comprise the majority of power dissipation in the circuit, their clocks can be disabled to save power during periods of inactivity. To disable these registers, the clock signal is gated and the enable signal is deactivated. The values in these registers should be preserved, so the reset signal remains deactivated. Clock gating should save power in both the registers and in the clock distribution tree for the DCT and the IDCT units. Note that clock gating will not degrade video quality, as there is no change in the output.

4.3.1 Characteristics of Registers

Table 4.4 shows the percentage of the total power dissipation contributed by the registers in each unit, as measured by Synopsys Design Power. The results are similar for both the DCT and the IDCT. The transposition memory contributes the most power dissipation, since it contains the most registers. The I/O units and the 1-D DCT/IDCT unit both contain about the same number of registers. However, the registers in the 1-D DCT/IDCT unit undergo more transitions, and, hence, dissipate more power.

Table 4.4: Percentage Of Power Contributed by Registers in Different Units

Unit	**Claire**		**Foreman**		**Miss Am.**	
	DCT	**IDCT**	**DCT**	**IDCT**	**DCT**	**IDCT**
Transposition Memory	56.9%	57.2%	56.6%	56.9%	56.9%	57.0%
1-D Unit	20.1%	19.1%	19.5%	18.8%	20.1%	19.1%
I/O Registers	10.8%	11.0%	10.7%	10.9%	10.8%	10.9%

Experimental results indicate that the above units only need to receive clocks for a small proportion of the entire 2-D DCT/IDCT cycle. If we can gate the clock signal for the remaining time, then we expect to reduce both switching and internal power dissipation in the registers.

4.3.1.1 Transposition Memory

In the transposition memory, the registers need to be updated only at the end of each of the first eight 1-D DCT/IDCT operations. This means the transposition memory registers require a clock in less than 3% of the clock cycles. For the remaining time, these registers can be disabled, since they only need to store values. Figure 4.3 shows the logic we use to gate the transposition memory registers.

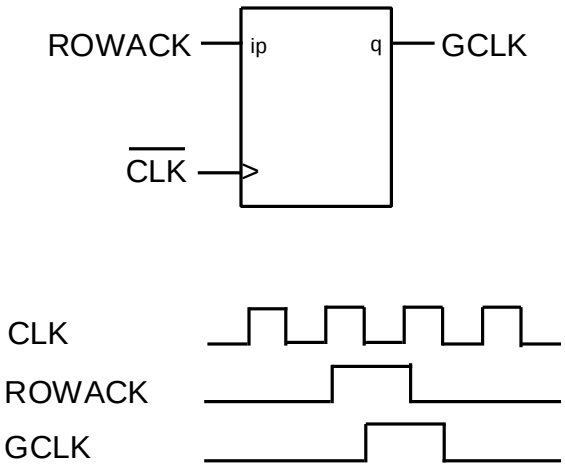


Figure 4.3: Clock Gating Hardware for the Transposition Memory

To provide a gated clock to the registers in the transposition memory, we use the existing signal ROWACK, which enables the transposition memory in the baseline unit. ROWACK is active for one cycle at the end of each of the first eight 1-D DCT/IDCTs. We use ROWACK as the input to a latch that is active for CLK = '0'. The gated clock, GCLK, is the output of the latch. We make the latch active for CLK = '0' to delay it by half a clock cycle. This delay ensures that there will be no race between the enable input to the transposition memory (ROWACK) and the GCLK signal. In the baseline circuit without clock gating, the input data is available just after the rising clock edge before ROWACK rises. Since CLK triggers ROWACK, the ROWACK signal will always arrive after CLK. Therefore, in the baseline unit, the input data has one clock period to settle before it is clocked in. However, the gating circuitry may delay GCLK until shortly after ROWACK enables the transposition memory; at this point, the input data is still unstable. To allow time for the input data to settle, we must ensure the gated clock GCLK arrives much later than the ROWACK signal. Enabling the latch with CLK '0' has the effect of delaying the gated clock by a half cycle, thus ensuring the input data is stable.

4.3.1.2 1-D DCT and IDCT Units

The registers in the baseline DCT and IDCT blocks are active for only about 35% of the entire 2-D DCT/IDCT operation, since many registers spend significant time storing values for future arithmetic operations. When they store values for longer than one clock cycle, the registers in the 1-D DCT and IDCT units can be disabled. Figure 4.4 illustrates the logic we use to gate the registers in the 1-D DCT and IDCT units.

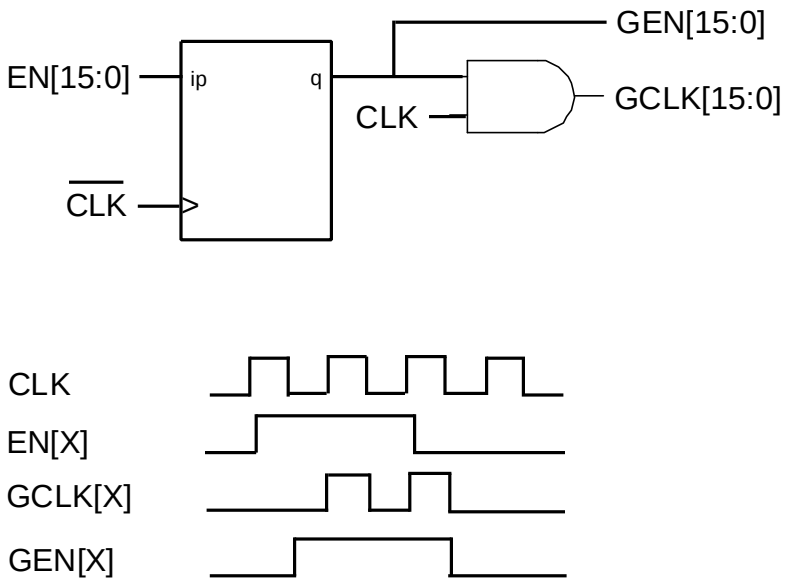


Figure 4.4: Clock Gating Hardware for the 1-D DCT and IDCT Units

For the 1-D DCT and IDCT units, the clock gating is done slightly differently than in the transposition memory, since the registers sometimes need to receive several clocks in succession. Additionally, each register requires a different gated clock signal, since each register requires input data at different times. As in the transposition memory, each enable signal is sent through a latch that is active for CLK = '0'. The output of the latch, GEN, follows the EN signal by half a cycle and enables the registers. GEN also gates the original clock signal, CLK. This ensures that the gated

clock, GCLK, arrives well after the enable signal, GEN, thus giving the input data sufficient time to settle.

4.3.1.3 I/O Units

Finally, the input registers in the ser2par unit need one clock for each of the 64 input values. In the par2ser unit, the output registers require one clock for each of the 64 output values and one clock at the end of each of the first eight 1-D DCT/IDCT operations. Overall, these I/O registers need clocks for less than 20% of the total 2-D DCT or IDCT operation. For the remaining time, the registers in the ser2par and par2ser units can be disabled, since they perform no useful function. Figure 4.5 shows the logic we use to gate the registers in the ser2par and par2ser units.

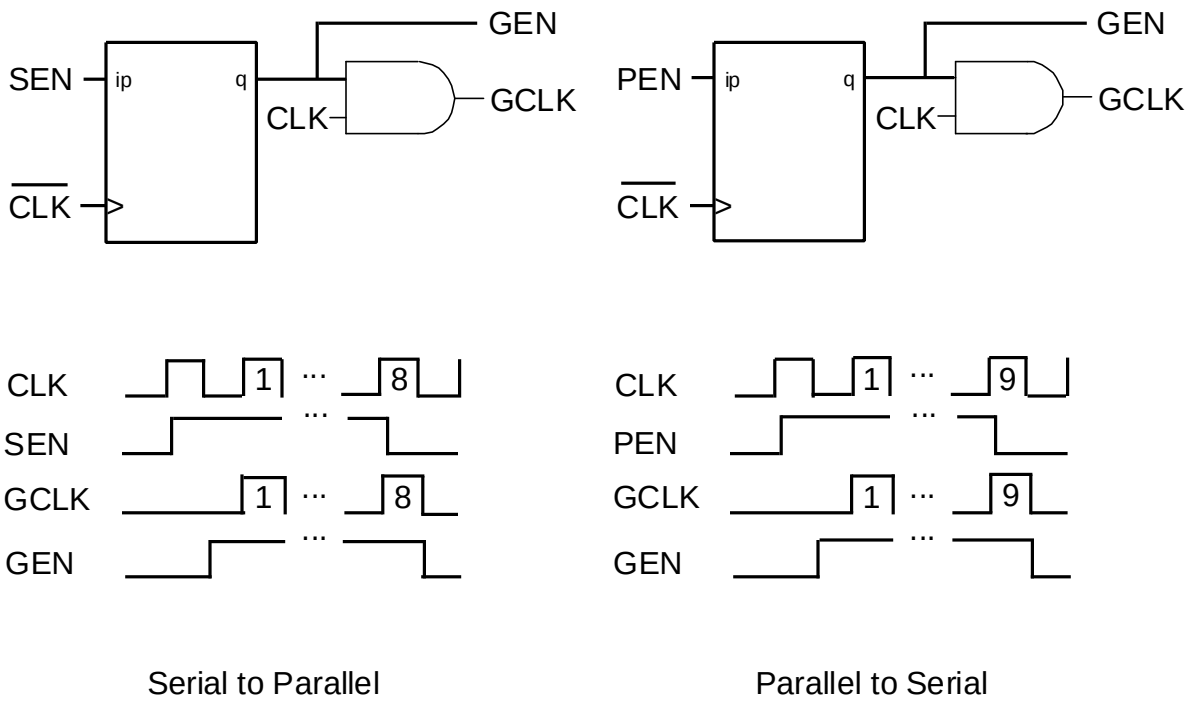


Figure 4.5: Clock Gating Hardware for the I/O Units

The clock gating scheme is similar to the 1-D DCT/IDCT units, except that only one gated clock signal, GCLK, is necessary in each unit. We use the existing enable signals, SEN and PEN, as inputs to a latch that is active for CLK = '0'. The

output of the latch, GEN, enables the registers in each unit and gates the original clock signal.

4.4 Constant Shift-and-Add Multipliers with Reduced Precision

Array multipliers are currently employed for the baseline unit, since they contribute less than 10% of total power dissipation. However, if a large power reduction is achieved through the above proposed methods, the proportion of power dissipated by multipliers may be significant for the circuit. Hence, it is desirable to employ a more power efficient multiplier design at the cost of higher design complexity. Instead of using an array multiplier from a library, we implement our own design of shift-and-add multipliers for each coefficient.

4.4.1 Characteristics of Shift-and-Add Multipliers

Several low power DCT and IDCT designs have reported that shift-and-add multipliers are more efficient in power dissipation than array multipliers [13][18][34]. The reduced number of adder rows over the array design saves at least half of the hardware overhead and operations as an array multiplier. In addition, the shift-and-add multipliers require the least modification to the baseline design. The power dissipation of the shift-and-add multipliers is further reduced with common subexpression sharing. In common subexpression sharing, common multiples of a number are shared among the various adders. The type of shift-and-add multiplier we implement is discussed in Chapter 2.

4.4.2 Accuracy of Multiplier

In the baseline DCT and IDCT units, we set the coefficient precision equal to the internal data width, since the array multipliers from the library require both multiplicands to be the same width. Our custom shift-and-add multiplier design lifts

this restriction of the library array multiplier. The coefficient precision no longer must match the internal data width, which is fourteen bits. We now desire to use the minimum coefficient width that does not significantly degrade video quality. Since our system uses matching IDCT architectures in both the coder and the decoder, the video will remain stable for reduced precision coefficients. This holds true for IDCT implementations such as ours that do not meet the IEEE accuracy standards.

In a system with an identical IDCT architecture in the encoder and the decoder, extra coefficient precision only contributes to extra power dissipation. Increased precision brings no noticeable improvement in video quality. This results from the fact that the quantization parameter - and hence the quantization noise - is usually large for low bit rates. The quantization noise dominates any error introduced by the DCT and IDCT units. Since the DCT and IDCT may require less than 14 bits of precision, we propose reduction of coefficient precision to reduce power dissipation. If we reduce the precision of our coefficients, then less complex shift-and-add multipliers will result.

The DCT and IDCT coefficients should be represented with the minimum amount of precision that does not noticeably degrade perceived video quality or PSNR. Table 4.6 shows the effects of applying reduced precision to the DCT and IDCT units in the software prototype.

Table 4.6: PSNR Degradation for Different Coefficient Precisions

Unit	Claire		Foreman		Miss Am.	
	DCT PSNR (dB)	IDCT PSNR (dB)	DCT PSNR (dB)	IDCT PSNR (dB)	DCT PSNR (dB)	IDCT PSNR (dB)
Baseline	40.53	40.53	29.64	29.64	40.52	40.52
12 bit	40.54	40.58	29.64	29.64	40.52	40.54
10 bit	40.52	40.51	29.61	29.62	40.52	40.49
8 bit	40.12	40.17	28.97	29.14	40.26	40.26
6 bit	29.72	30.57	27.53	27.62	31.72	32.29

Table 4.6 shows that coefficient precisions over eight bits improve the PSNR only fractionally with our DCT and IDCT architectures. Therefore, we decide to use 8 bit coefficients for both the DCT and IDCT units. Table 4.7 shows the values we use for each coefficient. Note that we shift the binary representations left eight places for fixed-point multiplication. This left shift is corrected by discarding the eight least significant bits of the multiplication result.

Table 4.7: Coefficient Values

Coefficient	**Decimal Value**	**Binary Value**
A	0.35355	0101 1010
B	0.49039	0111 1101
C	0.46194	0111 0110
D	0.41573	0110 1010
E	0.27779	0100 0111
F	0.19134	0011 0000
G	0.09755	0001 1000

4.5 Low Transition Data Paths

As described in Chapter 3, the registers and the arithmetic units in the 1-D DCT/IDCT unit select inputs from multiple sources through a bank of multiplexers. The controller generates the select inputs (SELR and SELM in Figure 3.7) for these multiplexers. Each multiplier receives input from two multiplexers, and each register receives input from one multiplexer. In the baseline design, the SELR and SELM signals receive a “don’t care” value when their register or arithmetic unit is inactive. When a register or arithmetic unit is inactive, it is desirable that the select inputs on its multiplexer(s) remain unchanged. To save power, we propose to hold the SELR and SELM signals at the last known value (‘0’ or ‘1’) for all “don’t care” values in the

baseline design. When the SELR or SELM signal remains unchanged during these "don't care" states, fewer transitions will occur on the inputs of the registers and arithmetic units. Additionally fewer transitions occur on the select inputs of the multiplexers. These new data paths should reduce power dissipation in the registers and arithmetic units. Note that this method produces no reduction in PSNR, and it increases the complexity of the control unit slightly.

As an example of the changes made to the control unit, we show the SEL outputs of the controller as coded in the baseline VHDL RTL model. The MSEL vector represents the outputs on the select signals (SELR and SELM) during three successive clock cycles. These are the desired outputs from the first three cycles of the baseline 1-D IDCT. The "don't care" conditions are represented by the "-".

```
MSEL <= "000000000--0--------------------------------";
MSEL <= "01-------00--000-----00000000000-----000000000";
MSEL <= "------------00-----01---001-----00---000001000";
```

To save power, we change the code at the expense of a more complex synthesized controller circuit by eliminating the "don't care" conditions. We replace each "don't care" condition with the last value that is not a "don't care" condition. If there is no previous value, the "don't care" condition is changed to the next value. The modified code is as follows.

```
MSEL <= "00000000000000000001000000000000000000000000";
MSEL <= "01000000000000000001000000000000000000000000";
MSEL <= "01000000000000000001000001000000000000001000";
```

Chapter 5: Experimental Results

In this chapter, we present the results of gate-level simulations on the low power techniques described in Chapter 4: skipping the DCT, skipping the IDCT, gating registers, using low-precision shift-and-add multipliers, and implementing a low-power data path. We start with a review of the test video sequences and the power estimation method. Then, we measure the power savings of applying each method to the baseline unit independently of the other methods. We use these results to find the most effective method. Next, we measure the effect of applying each method to the baseline unit in the order from most effective to least effective. We take these two different measurements to ensure that the best techniques are included in the final design. If we add a new method to a circuit that includes other low-power methods, the new method may not save additional power. If this new method saves significant power in the baseline unit, we want to include it before other, less efficient methods. Finally, we end with a comparison of our architecture to other implementations in terms of area, speed, and power.

5.1 Description of Test Video Sequences

Using Synopsys Design Power, we estimate power dissipation at the gate level for three test video clips: “Claire”, “Miss America”, and “Foreman”. The three video clips are in QCIF format, which contains 176x144 pixels per frame. For our test data, we use an entire INTER frame of each sequence, which requires 594 DCT or IDCT operations per frame. An INTER frame is a good choice for observing the effects of our low power techniques, since only the first frame in a video sequence is an INTRA frame. All remaining frames are encoded in INTER mode.

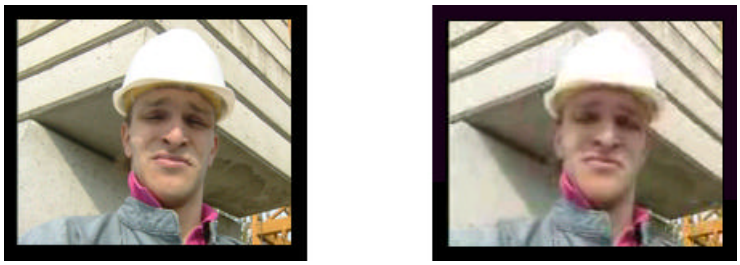
To review the video sequences, “Claire” and “Miss America” are both low motion sequences. These two clips contain a small amount of motion in the subject’s head and face area, and the camera remains stable for the duration of the video. “Claire” and “Miss America” represent typical data streams from videoconferencing

applications. The “Foreman” sequence provides a contrast to the other two sequences, as it contains a large amount of motion in the subject and an unstable camera. “Foreman” is more representative of data from a mobile patrol application.

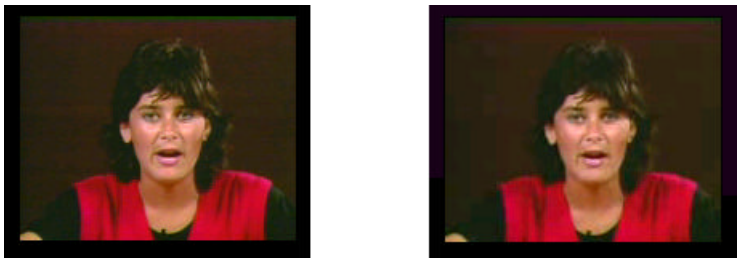
The relative amounts of motion are important for the low power techniques that depend on input data (skipping of DCT macroblocks and skipping of IDCT blocks). Sequences with a large amount of motion contain a large amount of non-zero data, which requires the DCT and IDCT operations. Less motion results in more zero-valued data, and, hence, we can skip more DCT and IDCT operations. For the data dependant low power techniques, we expect less power savings for the high motion “Foreman” sequence. For all other low power techniques, we expect approximately the same amount of power savings for all three video sequences.

Figure 5.1 shows an INTER frame of each video sequence before and after decoding by our software prototype. The software prototype includes the low power techniques that affect the data: skipping DCT macroblocks and lower precision multiplier coefficients. The screen capture on the left shows the original frame, while the screen capture on the right shows the decoded frame. These sequences are encoded at a frame rate of 8 fps and a bit rate of 20 kbps. The “Foreman” capture shows some degradation in quality, whereas the other two sequences show very little difference between the original and reconstructed frames. The degradation in “Foreman” results from the high motion of the sequence.

(A) Claire



(B) Foreman



(C) Miss America

Figure 5.1: Test Video Sequences

5.2 Power Estimation Method

We measure power dissipation from simulation of the test data from each of the above images. We verify the output data of the simulation against the output data produced by the software prototype. Synopsys calculates power with an estimation of the dynamic power based on the formula

$$P = aC_L V^2 f \tag{4.1}$$

where a is the switching activity, C_L is the capacitive load, V is the supply voltage, and f is the clock frequency. Synopsys Design Power computes the power at each node from the VTVT 0.35μ library cell characteristics and from toggle information at each node in the gate-level model. Summing the power dissipations at each node results in the total power for the circuit.

5.3 Individual Low Power Methods

We first apply each low power technique to the DCT/IDCT circuits independently. This shows the effectiveness of each technique without interference from the other techniques. After we know the effectiveness of each method individually, we can apply each technique to the baseline unit one at a time, starting with the most effective method. Tables 5.1 and 5.2 show the effects of each method when added to the baseline DCT and IDCT units for each video sequence. The column heading “Power” denotes the power dissipation of a method when applied to the baseline unit. The column heading “Percent Reduction” denotes the percentage of the power savings of the method over the baseline unit.

Table 5.1: Efficiency of Low Power DCT Methods When Applied Individually

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	Power (mW)	**Percent Reduction**	**Power (mW)**	**Percent Reduction**	**Power (mW)**	**Percent Reduction**
Baseline	9.25		9.39		9.32	
Skip Low SAD Blocks	2.24	75.8%	4.73	49.6%	2.66	71.5%
Gated Registers						
Transposition Memory	3.96	57.2%	4.13	56.0%	4.07	56.3%
1-D DCT	7.91	14.5%	8.18	12.9%	8.11	13.0%
I/O Registers	8.23	11.0%	8.48	9.69%	8.43	9.55%
Constant Multipliers	8.18	11.6%	8.46	9.90%	8.41	9.76%
Low Transition Path	8.38	9.4%	8.63	8.09%	8.57	8.05%

Table 5.2: Efficiency of Low Power IDCT Methods When Applied Individually

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	Power (mW)	**Percent Reduction**	**Power (mW)**	**Percent Reduction**	**Power (mW)**	**Percent Reduction**
Baseline	8.53		8.49		8.50	
Skip All-Zero Blocks	1.23	85.6%	1.41	83.4%	1.40	83.5%
Gated Registers						
Transposition Memory	3.55	58.4%	3.29	61.2%	3.53	58.5%
1-D DCT	7.41	13.1%	7.37	13.2%	7.38	13.2%
I/O Registers	7.55	11.5%	7.56	11.0%	7.54	11.3%
Constant Multipliers	7.64	10.4%	7.64	10.0%	7.59	10.7%
Low Transition Path	7.68	9.96%	7.71	9.19%	7.66	9.88%

From Table 5.1, the most efficient method for the DCT is the skipping of macroblocks with low SAD parameters, which reduces power dissipation by an average of 65.6%. This is attributable to the fact that 57.8% to 79.8% of macroblocks are skipped. The “Foreman” sequence shows less power reduction than the other sequences because its high amount of motion causes higher SAD values in relation to the quantization parameter. Hence, fewer blocks are skipped.

From Table 5.2, the most efficient method for the IDCT is the skipping of all-zero input blocks. The power savings is larger than the DCT, because the decision to skip is made at the block level, not at the macroblock level. The DCT may produce all-zero blocks that are not part of an all-zero macroblock.

Surprisingly, the power savings from skipping all-zero input blocks to the IDCT is about the same for the three video clips. It is explained as we set a constant target bit rate for each video clip. With equal target bit rates, high motion video sequences such as "Foreman" have higher quantization parameters than low motion video sequences; hence, just as many coefficients are forced to zero in high motion video sequences as in low motion video sequences. This also means that our prediction method for all-zero DCT macroblocks could be more efficient for high motion video. Future research could improve the efficiency of all-zero macroblock prediction in the DCT for high motion video sequences. Since these video sequences result in high SAD and QUANT parameters, one proposal would be to weight high QUANT and SAD values with a higher THRESHOLD value. This could possibly result in more accurate prediction of all-zero DCT macroblocks in high motion video sequences.

The second most efficient low power method for both is the gating of registers, which affects both the DCT and the IDCT unit almost equally. Registers account for a large portion of the power in the DCT and IDCT units. Since registers need only be enabled when they have meaningful input, gating registers saves a significant amount of power. The greatest power savings from results from gating the transposition memory. The transposition memory accounts for about 70% of the total number of flip-flops in the DCT or IDCT circuit, and it needs to be enabled for only a very small portion of the 2-D cycle. Gating the 1-D DCT/IDCT unit and the I/O unit also results in significant power savings, but not as much as in the transposition memory. The 1-D

DCT/IDCT and the I/O units both contain fewer registers than the transposition memory. These registers require clocks for a larger percentage of time than the transposition memory registers. Therefore, gating the I/O and 1-D DCT/IDCT registers results in less overall power savings than in the transposition memory.

The low power, low-precision multipliers make a smaller, yet significant, improvement in power dissipation. Note that the these multipliers can save more than 10% of the power dissipation over the baseline unit, when the array multipliers dissipated less than 10% of the power in the baseline unit. This is explained by analyzing the power savings in the individual units in the low power 1-D DCT and IDCT blocks. The constant multipliers now dissipate less than 1% of the total power. The additional power is saved in the data paths of the 1-D DCT and IDCT blocks. The constant multipliers have one fixed input that never transitions, whereas both inputs in the array multipliers made transitions. Additionally, the constant multipliers have fewer adder rows and undergo fewer glitches on the outputs. So the constant multipliers result in fewer transitions in both the input and output multiplexers and in the output registers. The total power savings in the multipliers, registers, and multiplexers can add to more than 10% of total power savings.

Optimizing the data path reduces power dissipation by a similar amount as the shift-and-add multipliers. The optimized data path reduces power by 15%-35% in the registers, arithmetic units, and multiplexers of the 1-D DCT and IDCT units. The overall power savings is around 10%.

5.4 Combined Low Power Methods

The five low power methods are then added to the baseline unit in order from the most efficient method to the least efficient method. Skipping blocks (in the IDCT unit) and macroblocks (in the DCT unit) is considered first. Next we add clock gating, following the order from the unit with the most power savings to the unit with the least power savings: the transposition memory, the 1-D DCT/IDCT unit, and the I/O

registers. Finally, we include the low power multipliers, followed by the low transition data path.

For the DCT and IDCT blocks, Table 5.3 and Table 5.4 show the experimental results under the cumulative employment of the methods. The column heading "Power" denotes the power of a method combined with all the methods above it. Similarly, the column heading "Percent Reduction" denotes the power savings of a method combined with all the methods above it.

Table 5.3: Efficiency of Low Power DCT Methods When Added Incrementally

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	Power (mW)	**Percent Reduction**	**Power (mW)**	**Percent Reduction**	**Power (mW)**	**Percent Reduction**
Baseline	9.25		9.39		9.32	
Skip All-Zero Blocks	2.24	75.8%	4.73	49.6%	2.66	71.5%
Gated Registers						
Transposition Memory	1.64	82.3%	2.89	69.2%	1.87	79.9%
1-D DCT	1.16	87.5%	1.87	80.1%	1.26	86.5%
I/O Registers	0.687	92.6%	1.23	86.9%	0.776	91.7%
Constant Multipliers	0.425	95.4%	0.639	93.2%	0.470	95.0%
Low Transition Data Path	0.314	96.6%	0.501	94.7%	0.354	96.2%
Total Power	**0.314**	**96.6%**	**0.501**	**94.7%**	**0.354**	**96.2%**

Table 5.4: Efficiency of Low Power IDCT Methods When Added Incrementally

Configuration	**Claire**		**Foreman**		**Miss Am.**	
	Power (mW)	**Percent Reduction**	**Power (mW)**	**Percent Reduction**	**Power (mW)**	**Percent Reduction**
Baseline	8.53		8.49		8.50	
Skip All-Zero Blocks	1.23	85.6%	1.41	83.4%	1.40	83.5%
Gated Registers						
Transposition Memory	0.776	90.9%	0.914	89.2%	0.910	89.3%
1-D DCT	0.377	95.6%	0.442	94.8%	0.440	94.8%
I/O Registers	0.240	97.2%	0.296	96.5%	0.297	96.5%
Constant Multipliers	0.161	98.1%	0.218	97.4%	0.226	97.3%
Low Transition Data Path	0.150	98.2%	0.189	97.8%	0.187	97.8%
Total Power	**0.150**	**98.2%**	**0.189**	**97.8%**	**0.187**	**97.8%**

We can observe from Tables 5.3 and 5.4 that each power savings technique has an impact, even after others are added. For both the DCT and the IDCT, the combination of power savings methods impact the overall power far more than any single design. The average power reduction of the proposed methods for DCT and IDCT is 95.8% and 97.9%, respectively.

5.5 Effect on PSNR

Since the skipping of DCT macroblocks and the lower precision multipliers degrade picture quality, we include the overall effect on PSNR when both methods are

added. We implement both methods in the software prototype, and Table 5.5 shows the effect of the methods on PSNR. Note that the combined methods degrade PSNR more than individual methods. However, the degradation is unnoticeable to human eyes.

Table 5.5: PSNR Degradation of Proposed Low Power Methods

Configuration	PSNR (dB) Claire	PSNR (dB) Foreman	PSNR (dB) Miss Am.
Baseline	40.53	29.64	40.52
Lower Precision Multipliers	39.19	29.33	39.46
Skip Low SAD Blocks	39.41	29.41	40.08
Combined	38.51	29.12	39.10

5.6 Comparison to Other Architectures

In this section, we summarize the characteristics of previous architectures to obtain some comparisons to our architecture. It is difficult to compare the power efficiency of different DCT/IDCT designs due to differences in supply voltages, operating frequencies, throughput rates and processes. Additionally, most of the previous DCT and IDCT designs target high throughput applications such as MPEG or HDTV, while our proposed design targets low bit rate video. We look at the power, supply voltage, technology size, area, clock speed, and throughput of other DCT/IDCT designs. Table 5.6 gives these parameters for several other architectures as well as our proposed designs for the DCT and the IDCT. For our power measurements, we use the average power dissipation of all three sequences

Table 5.6: Comparison of DCT and IDCT Architectures

Chip	**Power (mW)**	**VDD (V)**	**Tech. Size (mM)**	**Trans. Count**	**Area (mm^2)**	**Clock (MHz)**	**Throughput (Msmp/sec)**
[7]a	23.9	3.3	0.5	--	1.28	18.5625	74.25
[7]a	4.1	1.375	0.5	--	1.28	18.5625	74.25
[7]b	39	3.3	0.5	--	2	18.5625	74.25
[7]b	6.8	1.375	0.5	--	2	18.5625	74.24
[8]	4.65	1.32	0.5	160K	20.7	14	14
[9]	--	--	0.6	10.9K	7.07	40	3.63
[10]	--	3.3	0.6	130K	9.76	80	23.27
[11]	--	--	0.8	67K	10	100	100
[13]	--	--	0.5	10K *	--	33	33
[15]	--	--	0.6	36K	--	100	100
[17]a	22.46	--	0.7	--	--	--	--
[17]b	17.96	--	0.7	44K	6.41	--	--
[18]	4.28	1.56	0.6	120K	14.5	14	14
[19]	138	2.0	0.6	152K	50	100	100
[21]	--	--	1.5	98K	141	--	--
[22]	--	--	--	402K	--	1	16
[23]	--	--	1.2	320K	240	50	50
[26]	--	--	0.6	24K	--	54.9	23.6
[27]	--		0.8	--	.212	100	100
[28]	--	--	--	7.3K *	--	27	--
[29]	--	--	0.8	140K	49.21	50	50
[31]	17	1.5	1.2	--	--	20	--
Proposed DCT	**0.390**	**3.3**	**0.35**	**22K***		**2.5**	**0.38**
Proposed IDCT	**0.175**	**3.3**	**0.35**	**22K***		**2.5**	**0.38**

* count is in gates

From Table 5.6, the proposed designs compare favorably to others in terms of transistor (gate) count. The low power methods do not contribute significant area overhead to the baseline design. Also, note that most other designs target a throughput that can be several orders of magnitude faster than in our proposed designs. Hence, our design performs much better in an absolute comparison of power efficiency. This comparison is unfair because of the differences in throughput. To overcome the difficulties of comparing the power efficiency of different DCT/IDCT designs, the authors of [18] propose scaling formulae for the purpose of comparing power dissipation between different DCT designs. The scaling formulae are as follows:

$$
\begin{aligned}
S &= W/W', \\
U &= V/V', \\
P' &= P \times S/U^3 \\
f' &= f \times S^2/U \\
PP' &= P' \times f_{ref}/f'
\end{aligned}
\qquad (4.2)
$$

where,

W' - Reference Process
W - Process
V' - Reference Supply Voltage
V - Supply Voltage
P' - Power after Scaling
P - Power
f' - Frequency after Scaling
f - Frequency
PP' - Power for Fixed Throughput

For example, when a process is scaled by *1/S* and voltage by *1/U*, then the power scales by S/U^3 and frequency will scale by S^2/U. The power further scales by f_{ref}/f' if the frequency is scaled to the reference frequency by f_{ref}.

Using the above formula, we compare power efficiency of various existing DCT and IDCT designs. The scaled power dissipation for the various methods is shown in

Table 5.7. Although the comparison may not be fair due to the scaling problem, it is a good indication on the standing of the proposed methods. Our methods perform far better than other methods, mostly by an order of magnitude.

Table 5.7: Scaled Comparison of DCT and IDCT Architectures

Chip	**Scaled Power (mW)**	**Normalized Power**
[11]	21.2	0.37
[13]	31.8	0.55
[18]	58.1	1.00
[26]	38.5	0.66
[28] (Flow Graph)	23.9	0.41
[28] (Distributed Arithmetic)	39	0.67
Proposed DCT	**3.27**	**0.056**
Proposed IDCT	**1.31**	**0.022**

Chapter 6: Conclusion

This paper presents five low power techniques for an 8x8 2-D DCT or IDCT block in a low bit rate, wireless video application. To review, the techniques are as follows.

- Skipping low energy DCT input macroblocks.
- Skipping all-zero IDCT input blocks.
- Clock gating for inactive registers.
- Low precision shift-and-add multipliers.
- Low transition data path.

Some techniques are much more effective than others. We observe more power savings from techniques that apply to higher levels in the design hierarchy. Two low power techniques apply to the top level circuit: skipping input macroblocks for the DCT and skipping input blocks for the IDCT. These two techniques produce far more power savings than the other techniques. The next most effective technique is clock gating, which affects sub-blocks (the transposition memory, the 1-D DCT/IDCT unit, and the I/O registers) within the circuit. Finally, the two least effective techniques are the low-precision shift-and-add multipliers and the low transition data path. These two techniques apply to basic circuit components such as multipliers, registers, and multiplexers.

For the DCT, the most efficient scheme is skipping macroblocks. Using this scheme can produce power savings from 50% to 75%, depending on the amount of motion in the video sequence. Low motion video sequences save more power than high motion sequences, because the decision to skip is based on the value of the SAD parameter with respect to the QUANT parameter. The SAD parameters from low motion sequences are lower with respect to the QUANT parameter. This technique is effective because the entire DCT circuit is placed in standby mode for a significant portion of time. In standby mode, the circuit dissipates very little power, because all registers are reset and the clock signal is gated. It is important to note that this

technique produces some small degradation in PSNR because of the inexactness of predicting all-zero macroblocks based on the SAD and QUANT parameters. However, the degradation is not noticeable to human eyes.

The most efficient scheme for the IDCT block is skipping all-zero input data blocks, which decreases overall power dissipation by 80% to 85%. The power savings are similar for all three sequences, because the constant bit rate forces sequences with more motion to use a higher quantization parameter. The result is that both high motion and low motion sequences have approximately the same number of zeroes in the output data. This technique is effective because it places the circuit in standby mode, where little power is dissipated for over 90% of the input blocks. An added benefit of this method is that there is no degradation of video quality.

The next most effective technique, clock gating, affects both the DCT and the IDCT similarly, regardless of the type of video sequence. The registers are gated for the same amount of time in both the DCT and the IDCT regardless of the input data. Clock gating is effective because registers account for the majority of power dissipation in the circuit. Since most registers simply store data for a majority of the time, we can gate the clock for a significant amount of time. This technique results in an overall power savings of around 55% - 60% from the transposition memory, 12 - 15% from the 1-D DCT/IDCT unit, and 9 - 11% from the I/O units. Note that this method produces no degradation in picture quality.

With a substantial reduction in power from the above techniques, multipliers now become a concern for low power design. The array multipliers dissipate an increased proportion of the total power in the circuit. Shift-and-add multipliers are an effective replacement, because they reduce the number of adder rows over an array multiplier. The shift-and-add multipliers save power in both the multiplier units and in the data path. The low precision coefficients also reduce power by reducing the complexity of the multipliers. Reducing precision has the adverse effect of degrading picture quality, but not by a noticeable amount. Reduced precision multipliers save between 9 -11% of overall power dissipation for all three sequences over the baseline circuit for both DCT and IDCT.

Finally, the data path in the 1-D DCT/IDCT unit can be optimized for power instead of hardware complexity. We do this by eliminating "don't care" conditions on the select inputs to the multiplexers in the data path. We replace the "don't care" conditions with the previous select input, so as to reduce transitions in the registers, multiplexers, and arithmetic units in the 1-D DCT/IDCT unit. This technique is effective in both the DCT and the IDCT, regardless of the input data. The optimized data path results in an overall power savings of 8% – 9% with no degradation in picture quality.

Each of these techniques produces enough power savings to make it worthwhile to include in a low power implementation. Even after we add the most effective low power techniques to the baseline circuit, the remaining power reduction techniques are important; they now reduce a greater proportion of power in the circuit. When combined, these techniques reduce average power dissipation by 96% for the DCT and by 98% for the IDCT.

Finally, it is important to note that our methods can be integrated with some other existing methods such as adaptive precision [16] and zero-valued coefficients [16][21][34] to further reduce power dissipation. Further power reduction can also result from more common techniques such as lowering the supply voltage, parallelization, or using a low power standard cell library.

We conclude the paper by summarizing the characteristics for our DCT and IDCT designs in Table 6.1.

Table 6.1: Characteristics of Our DCT and IDCT Architectures

	DCT	**IDCT**
Gate count	22,000	22,000
Frequency	2.5 MHz	2.5 MHz
Throughput	10 fps @ QCIF	10 fps @ QCIF
Processing Technology	.35µm	.35µm
Power	.390 mW	.175 mW

Appendix A: Port Connections for the DCT and IDCT

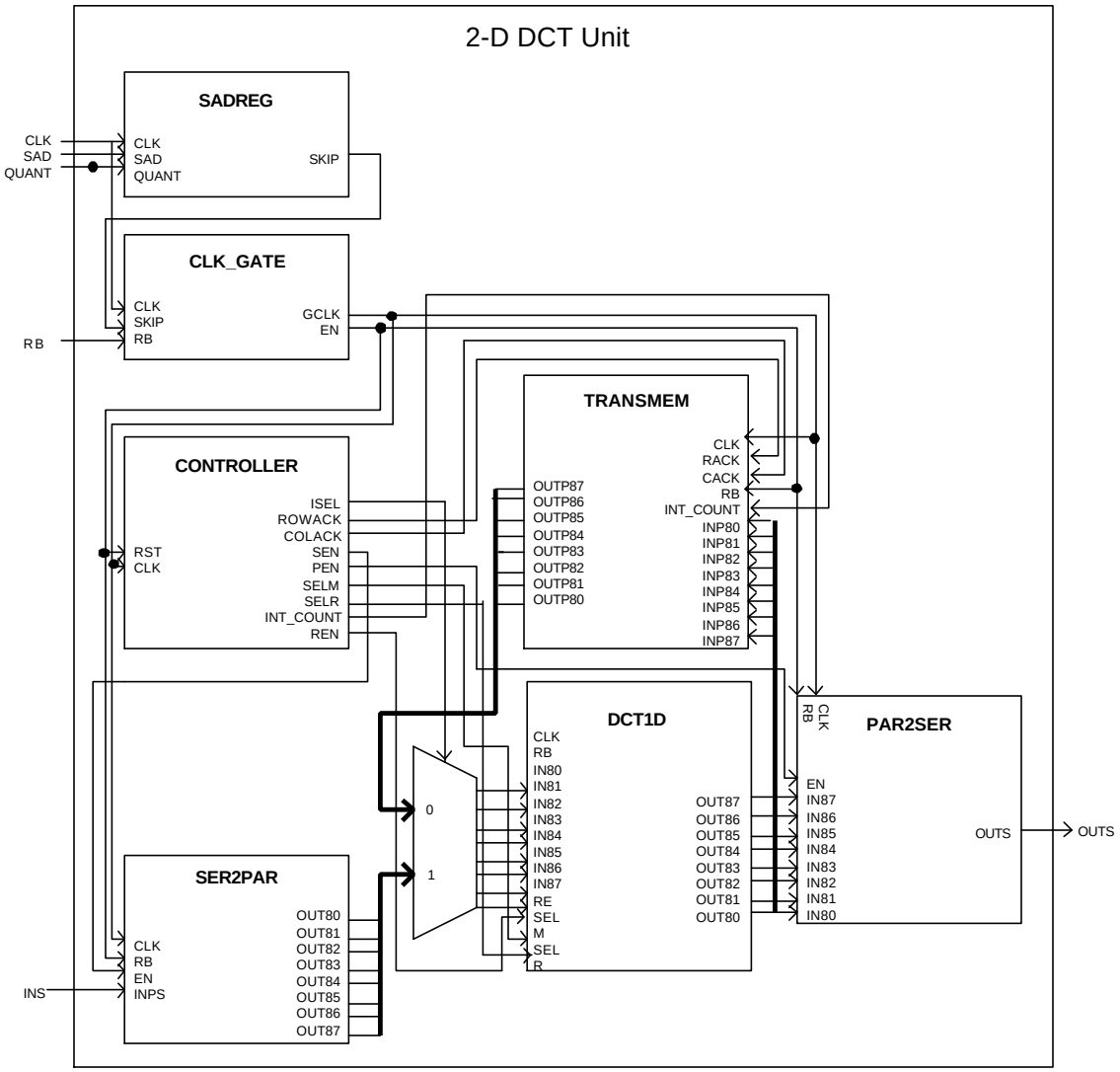


Figure A.1: Port Connections in the DCT Unit

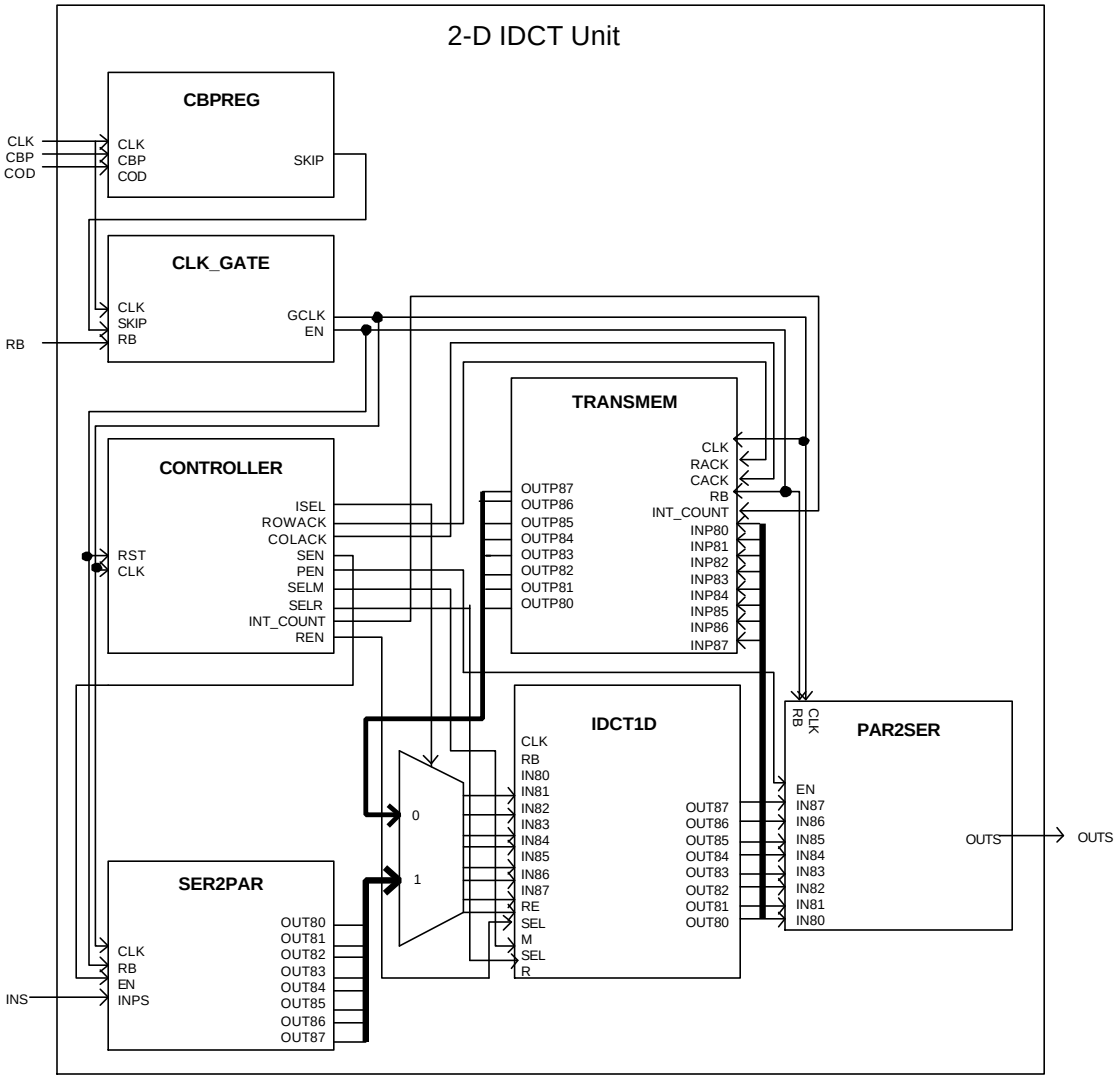


Figure A.2: Port Connections in the IDCT Unit

References

[1] G. Yeap, “Practical low power VLSI design,” Kluwer Academic Publishers, 1998.

[2] ITU-T Recommendation H.263 “Video coding for low-bitrate communication,” Draft International Standard, Feb. 1998.

[3] B. Erol, F. Kossentini, and H. Alnuweiri, “Implementation of a fast H.263+ encoder/decoder,” *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, 1998*, vol. 1, pp. 462-466, Sep. 1998.

[4] P. Master, “ACM augments DSPs for MPEG-4,” EE Times, Nov. 6, 2000, http://www.eet.com/story/OEG20001106S0037.

[5] ITU Telecom, “Video codec test model near term, Version 8 (TMN8),” H.263 Ad Hoc Group, June 1997.

[6] ITU Telecom, “Video codec test model near term, Version 11 (TMN11),” ITU-T/SG-16, Feb. 1999.

[7] Rao and Yip, “Discrete cosine transformation. algorithms, advantages, applications,” 1990.

[8] W. H. Chen, C. H. Smith, and S. C. Fralick, “A fast computational algorithm for the discrete cosine transform,” *IEEE Trans. Commun*., Vol. COM-25, pp. 1004-1009, Sept. 1977.

[9] B. G. Lee, “A new algorithm to compute the discrete cosine transform,” *IEEE Trans. Acoust., Speech, and Signal Processing*, Vol. ASSP-32, pp.1243-1245, Dec. 1984.

[10] N. I. Cho and S. U. Lee, “Fast algorithm and implementation of 2-D discrete cosine transform”, *IEEE Trans. Circuits and Systems*, Vol. 38, No. 3, pp. 297-305, Mar. 1991

[11] S. F. Hsiao, W. R. Shiue, and J.M. Tseng, "A cost-efficient and fully-pipelinable architecture for DCT/IDCT," *IEEE Transactions on Consumer Electronics*, vol. 45, pp. 515-525, Aug. 1999.

[12] K. Kim and J. S. Koh, "An area efficient DCT architecture for MPEG-2 video encoder," *IEEE Transactions on Consumer Electronics*, vol. 45, pp. 62-67, Feb. 1999.

[13] A. Madisetti, A. N. Willson, "A 100 MHz 2-D 8x8 DCT/IDCT processor for HDTV applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 158-165, Apr. 1995.

[14] I. K. Kim, J. J. Cha, and H. J. Cho, "A design of 2-D DCT/IDCT for real-time video applications," *6th International Conference on VLSI and CAD, 1999*, pp. 557-559, Oct. 1999.

[15] C. V. Schimpfle, P. Rieder, and J. A. Nossek, "A power efficient implementation of the discrete cosine transformation," *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers, 1999*, vol. 1, pp. 729-733, Nov. 1997.

[16] T. Xanthopoulos and A. P. Chandrakasan, "A low power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *IEEE Journal of Solid State Cricuits*, vol. 35, pp. 740-750, May 2000.

[17] Y. F. Jang, J. N. Kao, J. S. Yang, and P.C. Huang, "A 0.8u 100 MHz 2-D DCT core processor," *IEEE Transactions on Consumer Electronics*, vol. 40, pp. 703 – 710, Aug. 1994.

[18] J. Li and S. L. Lu, "Low power design of two-dimensional DCT," *Proceedings of the Ninth Annual IEEE International ASIC Conference and Exhibit, 1996*, pp. 309-312, Sep. 1996.

[19] H.B. Kim, "High-Level synthesis and implementation of built-in self-testable data path intensive circuit," PhD Dissertation, EE Dept., Virginia Tech, Dec. 1999.

[20] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2174-2193, Sep. 1992.

[21] E. Scopa, A. Leone, R. Guerrieri, and G. Baccarani, "A 2-D DCT low-power architecture for H.261 Coders," *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3271-3274, May 1995.

[22] Y. P. Lee, T. H. Chen, L. G. Chen, M. J. Chen, and C. W. Ku, "A cost-effective architecture for 8x8 two-dimensional DCT/IDCT using direct method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 459-467, Jun. 1997.

[23] L. G. Chen, J. Y. Jiu, H. C. Chang, Y. P. Lee, and C. W. Ku, "A low power 2-D DCT chip design using direct 2-D algorithm," *Proceedings of the ASP-DAC '98. Asia and South Pacific Design Automation Conference 1998*, pp. 145-150, Feb. 1998.

[24] K. W. Shin, H. W. Jeon, and Y. S. Kang, "An efficient VLSI implementation of vector-radix 2-D DCT using mesh-connected 2-D array," *1994 IEEE International Symposium on Circuits and Systems, 1994*, vol. 4, pp 47-50, Jun. 1994.

[25] Y. M. Huang, J. L. Wu, and C. T. Hsu, "A refined fast 2-D discrete cosine transform algorithm with regular butterfly structure," *IEEE Transactions on Consumer Electronics*, vol. 44, pp. 376-383, May 1998.

[26] T. S. Chang, C. S. Kung, and C. W. Jen, "A simple processor core design for DCT/IDCT," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 439-447, Apr. 2000.

[27] Liu, "Vector-radix DCT/IDCT implementation for MPEG DSP," *Proceedings of 3rd International Conference on Signal Processing 1996*, vol. 1, pp. 641-644, Oct. 1996.

[28] V. Srinivasan and K. J. R. Liu, "VLSI design of high-speed time-recursive 2-D DCT/IDCT processor for video applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 87-96, Feb. 1996.

[29] S. F. Hsiao and W. R. Shiue, "New hardware-efficient algorithm and architecture for the computation of 2-D DCT on a linear systolic array," *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999, Proceedings*, vol. 6, pp. 3517-3520 Mar. 1999.

[30] J. Chen and K. Liu, "Cost-Effective low-power architectures of video coding systems," *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 153-156, Jun. 1999.

[31] J. Chen and K. Liu, "Low-Power architectures for compressed domain video coding," *IEEE Transactions on Multimedia*, vol. 2, pp. 111-128, Jun. 2000.

[32] Lengwehasatit and Ortega, "Distortion/decoding time tradeoffs in software DCT-based image coding," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2725-2728, Apr. 1997.

[33] Girod and Stuhlmuller, "A content-dependant fast DCT for low bit-rate video coding," *Proceedings of the 1998 International Conference on Image Processing, 1998*, vol. 3, pp. 80-84, Oct. 1998.

[34] K. Kim and P. A. Beerel, "A high-performance low-power asynchronous matrix vector multiplier for discrete cosine transform," *The First IEEE Asia Pacific Conference on ASICs, 1999*, pp. 135-138, Aug. 1999.

[35] Chang, Guo, and Jen, "A compact IDCT processor for HDTV applications," *1999 IEEE Workshop on Signal Processing Systems*, pp. 151–158, Oct. 1999.

[36] Wolter and Laur, "Classification for 2D-DCT's and a new architecture with distributed arithmetic," *IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 2204-2207, Jun. 1991.

[37] Avinindra and Willson, "A 100 MHz 2-D 8x8 DCT/IDCT processor for HDTV applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 158 processor -165, Apr. 1995.

[38] Lu and Wen, "On the design of a selective coefficient DCT module," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 143-1146, Apr. 1998.

[39] T. Xanthopolous and A. P. Chandrakasan, "A low-power IDCT macrocell for MPEG-2 MP@ML exploiting data distribution properties for minimal activity," *IEEE Journal of Solid State Circuits,* vol. 34, pp. 693-703, May 1999.

[40] Walmsley, Skodras, and Curtis, "A fast picture compression technique," *IEEE Transactions on Consumer Electronics*, vol. 40, pp.11-18, Feb. 1994.

[41] Venkatesh and Srinivasan, "A pruning based fast rate control algorithm for MPEG coding," *Proceedings of the Third International Conference on Computational Intelligence and Multimedia Applications, 1999,* pp. 403-407, Sep. 1999.

[42] Wang, "Pruning the fast discrete cosine transform," *IEEE Transactions on Communications,* vol. 39, pp. 640-643, May 1991.

[43] M. Kuhlmann and K. K. Parhi, "Power comparison of flow-graph and distributed arithmetic based DCT architectures," *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, 1998*, vol. 2, pp. 1214-1219, Nov. 1998.

[44] J.B. Sulistyo, "On the characterization of library cells," M.S. Thesis, EE Dept., Virginia Tech, Aug. 2000.

[45] I. M. Pao and M. T. Sun, "Modeling DCT coefficients for fast video encoding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 9, pp. 608-616, Jun. 1999.

[46] A. Yu, R. Lee, and M. Flynn, "Performance enhancement of H.263 encoder based on zero coefficient prediction," *ACM Proceedings of the conference on Multimedia '97,* November 9 - 13, 1997, Seattle, WA USA, pp. 21-29, Nov. 1997.

Vita

Nathaniel John August was born in Cumberland, Maryland on May 5 1975. He graduated *magna cum laude* with a Bachelor of Science in Computer Engineering and a minor in Computer Science from Virginia Tech in Blacksburg, Virginia in May 1998. Before and after graduation, he obtained several years of internship experience as a Systems Design Verification Engineer at Intel Corporation in Folsom, California.

He re-entered the Bradley Department of Electrical and Computer Engineering at Virginia Tech in August 1999. He worked under Dr. Dong Ha at the Virginia Tech for VLSI Telecommunications (VTVT) Laboratory and studied low-power VLSI design, concentrating on video systems. After receiving his M.S.E.E. degree, he will work as a designer for the Network Components Division of Intel Corporation in Portland, Oregon.