Low Power Data-Dependent Transform Video and Still Image Coding

by

Thucydides Xanthopoulos

Bachelor of Science in Electrical Engineering
Massachusetts Institute of Technology, June 1992

Master of Science in Electrical Engineering and Computer Science
Massachusetts Institute of Technology, February 1995

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 1999

Signature of Author _____

Department of Electrical Engineering and Computer Science
February 1, 1999

Certified by _____

Anantha Chandrakasan, Ph.D.
Associate Professor of Electrical Engineering
Thesis Supervisor

Accepted by _____

Arthur Clarke Smith, Ph.D.
Professor of Electrical Engineering
Graduate Officer

Low Power Data-Dependent Transform Video and Still Image Coding

by

Thucydides Xanthopoulos

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 1999 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

This work introduces the idea of data dependent video coding for low power. Algorithms for the Discrete Cosine Transform (DCT) and its inverse are introduced which exploit statistical properties of the input data in both the space and spatial frequency domains in order to minimize the total number of arithmetic operations. Two VLSI chips have been built as a proof-of-concept of data dependent processing implementing the DCT and its inverse (IDCT). The IDCT core processor exploits the presence of a large number of zero-valued spectral coefficients in the input stream when stimulated with MPEG-compressed video sequences. A data-driven IDCT computation algorithm along with clock gating techniques are used to reduce the number of arithmetic operations for video inputs. The second chip is a DCT core processor that exhibits two innovative techniques for arithmetic operation reduction in the DCT computation context along with standard voltage scaling techniques such as pipelining and parallelism. The first method reduces the bitwidth of arithmetic operations in the presence of data spatial correlation. The second method trades off power dissipation and image compression quality (arithmetic precision.) Both chips are fully functional and exhibit the lowest switched capacitance per sample among past DCT/IDCT chips reported in the literature. Their power dissipation profile shows a strong dependence with certain statistical properties of the data that they operate on, according to the design goal.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor of Electrical Engineering

Acknowledgments

I am very grateful to a number of people for help and contributions during my years as a student.

I would like to thank my thesis advisor and mentor Anantha Chandrakasan for all his help and support. Anantha has been a constant source of stimulation and new research ideas and his impact on this work is immeasurable. I thank him for treating me as a peer and not as a subordinate, for providing me with the best possible (and expensive!) equipment for my work, for sending me to numerous conferences (as far as Hawaii and Switzerland) and for having a lot of faith in my abilities. Above all I wish to thank him for the unprecedented amount of time he has devoted to me and my work.

I wish to thank my thesis readers Tayo Akinwande and Ichiro Masaki for their time and for the very useful comments and suggestions. This work has been greatly improved because of their time and effort.

I am very grateful to my fellow graduate students in 38-107 for a number of reasons. Raj Amirtharajah has been a constant source of wit and interesting ideas (in addition to complaints!) and has been very helpful in clarifying various ideas regarding this work. He has probably had dinner with me more often in the little round table in 38-107 than he did with his girlfriend! Tom Simon, the brilliant designer, has been behind all the interesting circuits of the DCT chip such as the pads and the flip flops. His cerebral sarcasm and our long discussions about France, Mahler and crazy dictators have been excellent and fulfilling chip layout breaks! Jim Goodman, in addition to being an incessant source of coarse Canadian humor, has been instrumental in setting up the technology files and the libraries for the ADI process. His help during the DCT chip tapeout has been invaluable. Scott Meninger, fellow Patriot fan, taught me how to throw the football and introduced me to Mississippi Mud in addition to being an excellent officemate. He is probably the only one in 38-107 that can match my beer drinking abilities! Wendi Rabiner has been a great source of comments and suggestions both for my thesis and my area exam. Josh Bretz introduced me to the humor of the “Onion”, supplied me with great CDs and taught me never to bet against him especially when money is involved. SeongHwan Cho has been a great help with my area exam and has graciously added my name to all the VLD papers. Vadim Gutnik has been a source of interesting discussions and has been very helpful with Unix problems. Gangs Konduri has graciously agreed to help me with a very useful Perl script. I also thank him for the late night study breaks. I wish to thank Rex Min for doing a great job on the DCT/ IDCT demonstration board. I thank Dan McMahill, Don Hitko and

Nate Shnidman for stimulating discussions over the past years. Dan has also helped me a great deal with the Tango PCB software. I did not have much interaction with the newest members of our group Amit Sinha and Alice Wang due to my grueling schedule during the past nine months. Yet, I am well aware of their outstanding abilities and I wish them best of luck in their studies. I wish to thank Elizabeth Chung and Margaret Flaherty who have been extremely helpful with administrative matters. Tom Lohman, Myron Freeman and Tom Wingard have been very helpful with the computing infrastructure.

I thank National Semiconductor Corporation for providing me with a fellowship during my last two years. I thank Michael Sampogna, Chuck Traylor and Hemant Bheda for all their help during my summer at National.

I thank SHARP Corporation for their partial financial support for this research. I thank Yoshifumi Yaoi of SHARP for his help with the DAC paper and with the MPEG compression standard in general.

I wish to thank Rich Lethin for entrusting me with interesting projects and providing me with great opportunities.

I thank Larry Dennison for his valuable advice in time of need and for his excellent SKILL tools that I used extensively throughout the development of the DCT chip.

I would like to thank my Master thesis advisor Bill Dally for his help and support during my years at the MIT AI Laboratory. I also thank him for pointing me to the FMIDCT paper.

I wish to thank my uncle John Xanthopoulos and aunt Virginia Xanthopoulos for all their help and support during my years in the States. Their house has always been open for me and their love has been unconditional.

I thank my grandmothers Georgia Xanthopoulos and Themelina Kapellas for their love and support from the day I was born.

I thank my parents Nikos and Maria and my sister Georgia for their love and support. I know how much you have missed me during the past 10 years because I have missed you exactly the same if not more. You have made me what I am and for this I am eternally grateful. My trips to Greece and our telephone conversations have been an incessant source of strength and courage for me to go on. I know that you want me to come back soon. I promise to you that I will come back sooner than you think.

Finally, I wish to thank my companion Sylvi Dogramatzian for her unconditional love and support from the day we met. Your love has made me a better person. I thank you for your patience and all your sacrifices. I promise that I will make up for all the lost ground. This work is as much yours as it is mine.

I wish to dedicate this work to the memory of both my grandfathers, Thucydides Xanthopoulos and Manolis Kapellas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Transform coding has been a dominant method of video and still image compression. Transform coding works by compacting the energy of spatially correlated picture elements (pels) into a small number of spatial frequencies. A smaller number of spatial frequencies is required to describe a still picture or video frame than space domain pels. The reason for this compaction is the fact that images on average exhibit spatial correlation which results in high-magnitude low spatial frequency coefficients and low-magnitude high frequency coefficients in the transform domain. Moreover, low-magnitude higher frequency coefficients may be discarded during the compression process at the expense of virtually imperceptible visual artifacts.

Since its conception in 1974, the Discrete Cosine Transform (DCT) [ANR74] is the dominant transform of choice for video and still image compression applications. The DCT uses cosines as basis functions as opposed to complex exponentials used in the Discrete Fourier Transform (DFT). As a result, it exhibits less computational complexity by almost a factor of 2 because it performs real as opposed to complex arithmetic. Low computational complexity is only one advantage of the DCT vs. other transforms. Another most important advantage is that due to its symmetry the DCT does not have the ringing artifacts known otherwise as the Gibbs phenomenon when discontinuous signals undergo transformation and consecutive inverse transformation back to the spatial domain. This results in much less conspicuous "blocking artifacts" when applied to disjoint $n \times n$ image pixel regions than the DFT for the same compression factor. The final advantage of the DCT is that it produces virtually optimally decorrelated frequency domain coefficients. This property is only slightly surpassed by the theoretically optimal and computationally intensive Karhunen-Loeve Transform. Due to its most desirable properties, the DCT has been part of virtually every video and still image compression published standard (i.e. JPEG, MPEG-1, MPEG-2 etc.)

Until recently, dedicated signal processing hardware was necessary for DCT/IDCT computation in real time video applications. MPEG2 compression/ decompression for a 720×480 frame size at 30 frames per second (fps) 4:2:0 requires approximately 132 MOPS (Mega Operations Per Second) worst case assuming that a fast computation algorithm is used. Furthermore, 8×8 2D DCT/IDCT computation algorithms usually exhibit 8-way

fine grain parallelism due to the lack of data dependencies at each computation stage in the signal flowgraph. Current 300 MHz microprocessors with instruction sets enhanced with multimedia extensions (i.e. Intel MMX) are capable of performing multiple instructions per cycle by partitioning a single 64-bit integer datapath to 2 32-bit, 4 16-bit or 8 8-bit independent integer datapaths operating in parallel. Such instruction sets are ideal for the implementation of signal processing tasks that involve multiple independent low precision integer operations. A 300 MHz Intel Pentium II is capable of 1.2 GOPS on 16-bit media data types. Given the above numbers, we observe that the DCT/IDCT approximately occupies 11% of the processing resources available in commodity microprocessors today in a real time video application. The remaining 89 % of the machine resources may be allocated to other video processing tasks such as motion compensation and entropy decoding or even graphics computations for updating the display.

The analysis above only involves arithmetic operations as opposed to total machine instructions and does not include machine cycles consumed by control instructions (such as conditional and unconditional branches) and data reorganization instructions (register moves, pack and unpack media instructions as well as memory loads and stores.) As a result, the resources required by a 2D DCT/IDCT computation are reasonably underestimated. A more accurate analysis regarding the computational requirements of the DCT on a general purpose machine can be done using the following published facts: Intel has recently published an application note showcasing the benefits of the MMX technology that exhibits an MMX implementation of the AAN [AAN88] IDCT algorithm. The machine code requires 240 cycles per 8×8 coefficient block. On a 300 MHz Pentium II, an 8×8 block requires 0.8 μsec for processing. To sustain 720×480 real time 4:2:0 video at 30 fps, the worst case block processing time (including all other compression / decompression steps) must not exceed 4.1 μsec. Therefore, the DCT real-time computation requirements consume about 20% of the machine resources.

1.1 A Case for Low Power Hardware Compression/Decompression Macrocells

Although the real-time computation requirements of fast DCT/IDCT algorithms do not pose a challenge to contemporary general purpose microprocessors, the power dissipation of such a software implementation can be substantial. According to Intel published specifications [Int98], a Tillamook-class mobile Pentium MMX microprocessor (1.8V, 233 MHz) exhibits about 7 nF of switched capacitance per machine cycle which corresponds to 5.3 Watts of power dissipation at 1.8V, 233 MHz. The DCT/IDCT software computation required for a single 720×480 video frame corresponds to 9 mF of switched capacitance (1.3M machine cycles at 7 nF per cycle). A software application supporting real-time video at 30 fps will need 0.88 Watts of power just for the transform computation. This power level is unacceptable for portable battery-powered devices such as mobile video-playback and videoconferencing terminals, or information appliances with built-in web browsers.

On the other hand, carefully optimized hardware macrocells can exhibit as little as 0.1 mF of switched capacitance per video frame, which amounts to < 5 mWatts of power dissi-

pation for the real time transform computation. Power savings over a software implementation (after process feature size normalization) exceed a factor of 200. At current process feature sizes and supply voltage levels, fixed-function processing units are required for real-time portable media appliances.

Low power multimedia processing macrocells have non-mobile applications as well. Current trends in general purpose microprocessor design fueled by the availability of more and more transistors dictate the integration of various and diverse application-specific units on the same chip as the CPU. The main motivation is cost and ease of use. An example of such a design approach is the new Cyrix/ National Semiconductor MediaGX microprocessor which includes a graphics accelerator, an Extended Data Out (EDO) DRAM controller, a display controller with SVGA output and a PCI controller on the same chip as the x86 compatible CPU. The next generation MediaGX scheduled to be released in the second half of 1998 will also integrate MMX support, an Accelerated Graphics Port (AGP) controller, Digital Versatile Disc (DVD) playback and a synchronous DRAM controller. The MediaGX CPU has pioneered the sub $700 PC which has achieved great popularity in the consumer market.

Such levels of integration present acute power management problems to system designers and integrators. Excessive power dissipation can prevent function integration due to heat removal problems or require the use of expensive packaging and cooling mechanisms thus raising total system cost. The use of low power macrocells implementing common and frequently used tasks will help lower the system average and peak power dissipation and will enable even greater levels of integration and lower implementation costs.

It is the focus of this resarch to propose ultra low power hardware DCT/IDCT implementations suitable for use in portable media applications and for integration in future highly-integrated systems-on-a-chip. This thesis will explore novel methods for power reduction that span both the algorithm and architectural design space. The final product of this work will include a DCT/IDCT chipset exhibiting the lowest reported switched capacitance requirements per sample, when process parameters are normalized.

1.2 Low Power Design Methodoloy

Our methodology for power reduction involves three main steps:

1.2.1 Algorithm Optimizations

Since the DCT/IDCT macrocells are targetted for video and still image coding, the computation algorithm is optimized for the statistical characteristics of video and image data both in the spatial and frequency domains.

The main characteristic of spatial image data is spatial correlation. This property can be used to reduce the number of bits required to describe such a dataset by appropriate level

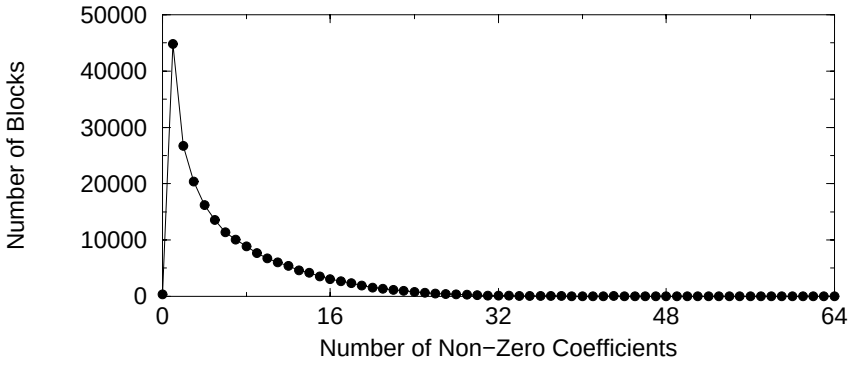


Figure 1.1: Histogram of Non-Zero DCT Coefficients in a Sample MPEG Stream

shifting or other related linear operations. Moreover, adaptive bitwidth computation units can be designed that exhibit reduced activity in the presence of low-magnitude inputs. The overall effect will be reduced switching activity within the arithmetic units without affecting computation precision.

An orthogonal method of reducing activity in the presence of correlated inputs can be devised on an apriori knowledge of the output data distribution and the corresponding output quantization matrix. Spatially correlated video inputs result in low-magnitude high spatial frequencies which most often are quantized to zero. Adaptive thresholding mechanisms can be implemented that power down computation units and feed forward zeroes at the outputs upon examination and classification of intermediate results, thus resulting in further activity reduction.

Finally, there exists output coefficient precision information embedded within the output quantization matrix. Special hardware control mechanisms can use this apriori information and dynamically lower the computation precision of certain output frequencies resulting in minimal and practically imperceptible PSNR degradation vs. a baseline coder. Such a technique will also result in reduced switching activity within the arithmetic units.

On the other hand, the inputs to the IDCT unit exhibit rather different statistical properties. The input sequence is decorrelated but exhinbits a distribution heavily skewed towards zero. An example of a typical MPEG sequence DCT coefficient histogram is shown in Figure 1.1

An IDCT algorithm has been used that exploits the presence of zero coefficients by preventing them from entering the computation pipeline. This method has resulted in activity reduction proportional to the percentage of zero-valued coefficients within the video block.

The collection of algorithmic optimizations used have reduced the total switching of the macrocells and is the main source of power savings.

1.2.2 Architectural Optimizations

Previously proposed architectural-driven voltage scaling techniques [CSB92] [CBB94] have been adapted and used for power minimization. Deep pipelining has been used to achieve voltage scaling at real-time data rates. Moreover, extensive clock gating has been used to capitalize on the adaptive utilization of the arithmetic units.

1.2.3 Circuit Optimizations

Special circuit techniques have been used to implement arithmetic circuits that exhibit good performance at low voltage given the assymetrical speed degradation of PMOS vs. NMOS transistors. Moreover, efforts have been made to design common circuits (i.e. flops and arithmetic circuits) that minimize switched capacitance and thus power dissipation.

1.3 Thesis Contributions and Overview

This dissertation introduces the idea of data-dependent video processing for low power and proposes two data-driven algorithms for the computation of the DCT and IDCT that minimize the average number of arithmetic operations required when operating on image or video data. In addition, it presents a proof-of-concept VLSI implementation of a DCT/IDCT chipset (chapters 3 and 5). The power dissipation profile of both chips depends on some statistical property of the input data according to the design goal.

Throughout the course of this work, a number of custom CAD tools have been developed and used during the design of the DCT and IDCT chips. Appendix A describes these tools and documents the design flow.

Chapter 2

Background

This chapter provides disjoint condensed treatments of various subjects directly related to this work to render this document self-contained. First, the mathematical definition of the discrete cosine transform and its inverse is presented along with a set of fast algorithms for its computation.

A comprehensive presentation of distributed arithmetic follows. Distributed arithmetic is a bit serial computation mechanization that is extensively used in the DCT core processor. We also present a concise definition of quantization along with a a method for designing optimum quantizers (Lloyd-Max). Quantization is a post-DCT operation that reduces the precision of less visually significant spectral coefficients for compression purposes. We take advantage of this operation in the design of the DCT chip by folding the precision reduction inside the DCT computation and saving arithmetic operations.

The chapter concludes with the definition of the peak signal-to-noise ratio (PSNR) as an image quality measure which is extensively used in the DCT chapter (5) and a brief description of the MPEG compression standard which is an important application for both our chips.

2.1 The Discrete Cosine Transform and its Inverse

The Forward Discrete Cosine Transform (FDCT) is the process of decomposing a set of image samples into a scaled set of discrete cosine basis functions. On the other hand, the process of reconstructing a set of image samples from a set of scaled discrete cosine basis functions is called the Inverse Discrete Cosine Transform.

The 8-point 1-Dimensional Forward Discrete Cosine Transform and the 8-point 1- Dimensional Inverse Discrete Cosine Transform are defined as follows:

$$X[u] = \frac{c[u]}{2}\sum_{i=0}^{7} x[i]\cos\left(\frac{(2i+1)u\pi}{16}\right) \tag{2.1}$$

$$x[i] = \sum_{u=0}^{7} \frac{c[u]}{2} X[u]\cos\left(\frac{(2i+1)u\pi}{16}\right) \tag{2.2}$$

where $c[u] = 1/\sqrt{2}$ if $u = 0$ and 1 otherwise.

The 64-point 2-D FDCT and 2-D IDCT are defined as follows:

$$X[u,v] = \frac{c[u]c[v]}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} x[i,j] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{2.3}$$

$$x[i,j] = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c[u]c[v]X[u,v] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{2.4}$$

The FDCT/IDCT can also be defined in matrix form [RY90]. Let us define a matrix C such that:

$$[C]_{ij} = \frac{1}{2}\left[c[j] \cos\left(\frac{(2i+1)j\pi}{16}\right)\right] \quad i,j = 0,1,\ldots,7 \tag{2.5}$$

Matrix C is referred to in the literature as the DCT-II matrix [RY90]. The 2-Dimensional FDCT of an 8×8 block of image samples $[x]_{ij}$ can also be defined in block form as:

$$[X]_{ij} = [C]_{ij}[x]_{ij}[C]_{ij}^T \tag{2.6}$$

The 2-Dimensional IDCT of an 8×8 block of DCT coefficients is defined as in eq. 2.6 by replacing $[C]_{ij}$ with $[C]_{ij}^T$ and $[X]_{ij}$ with $[x]_{ij}$.

Equation 2.6 indicates an important property of any *separable* transform such as the DCT, DFT, WHT and HT: *A 2D $N \times N$ point DCT can be computed by means of $2N$ N-point DCTs as follows: First, the 1D N-point DCT of each block row is computed; then the intermediate result is transposed, and the 1D N-point DCT of each column is computed in an identical fashion.* This property has been widely used in VLSI implementations of such transforms because of the reduced arithmetic complexity achieved. $2N$ N-point DCTs require substantially fewer arithmetic operations than one N^2-point DCT ($N = 8$).

2.1.1 Fast DCT/IDCT Algorithms

Numerous fast algorithms for both FDCT and IDCT have appeared in the literature that attempt to minimize the number of arithmetic operations (multiplications and additions) required. Blind application of equation 2.6 results in 1024 multiplications and 896 additions per block. Table 2.1 lists some popular algorithms along with the number of operations that they require.

Fast algorithms usually rely on symmetry properties of the cosine basis functions, on similarities with the Fast Discrete Fourier Transform computation and on certain factorizations of matrix C in eq. 2.6.

Chen's algorithm [CSF77] has been extensively used in VLSI chips [SCG89] [UIT+92] [MHS+94] due to its symmetry and direct implementation using Distributed Arithmetic (section 2.2). Feig's algorithm is not popular among VLSI implementations due to its awkward data shuffling requirements. The AAN algorithm produces scaled results. The computation of the actual DCT coefficient values can be merged with the subsequent quantization step that involves scaling anyway.

ALGORITHM	MULTIPLICATIONS	ADDITIONS
Ligtenberg & Vetterli [LV86]	208	464
Chen [CSF77]	256	416
Feig [FW92]	54	462
Lee [CL92]	112	472
Arai Agui Nakajima (AAN) [AAN88]	80	464

Table 2.1
Fast FDCT / IDCT Algorithms

The algorithms of Table 2.1 have been formulated in a way such that the computation takes a minimal yet constant number of operations, independent of the input data. McMillan and Westover [MW92] [MW93] [MW94] have proposed a direct realization of the 2D IDCT that has a number of operations proportional to the number of non-zero DCT coefficients [XCSD96]. The worst case performance of this algorithm varies significantly from the average performance. The McMillan-Westover Forward-Mapped IDCT (FMIDCT) is formulated as follows:

$$\begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ \vdots \\ x_{8,8} \end{bmatrix} = X_{0,0} \begin{bmatrix} c_0^{0,0} \\ c_1^{0,0} \\ c_2^{0,0} \\ \vdots \\ c_{64}^{0,0} \end{bmatrix} + X_{0,1} \begin{bmatrix} c_0^{0,1} \\ c_1^{0,1} \\ c_2^{0,1} \\ \vdots \\ c_{64}^{0,1} \end{bmatrix} + \cdots + X_{8,8} \begin{bmatrix} c_0^{8,8} \\ c_1^{8,8} \\ c_2^{8,8} \\ \vdots \\ c_{64}^{8,8} \end{bmatrix} \tag{2.7}$$

where $x_{i,j}$ are the reconstructed pels, $X_{i,j}$ are the input DCT coefficients (mostly zeroes) and the column vectors $[c_k^{i,j}]$ are constant reconstruction *kernels* easily derivable from the DCT-II matrix C (eq. 2.5). It is rather obvious that for compressed image and video streams exhibiting the frequency distribution of Figure 1.1, eq. 2.7 results in a small number of operations since multiplication and accumulation with a zero constitutes a NOP.

The IDCT chip presented in chapter 3 utilizes a similar algorithm redesigned for a row-column approach that results in minimal average activity for MPEG-compressed input data. On the other hand, the DCT chip discussed in chapter 5 cannot use a zero-dependent algorithm simply because pel data does not exhibit a skewed-toward-zero distribution. A version of Chen's algorithm will be used enhanced with adaptation techniques that minimize the total number of operations in the presence of a highly correlated input sequence.

2.2 Distributed Arithmetic

Distributed Arithmetic (DA) [PL74] [Whi89] is a bit-serial operation that computes the inner product of two vectors (one of which is a constant) in parallel. Its main advantage is the efficiency of mechanization and the fact that no multiply operations are necessary. DA has an inherent bit-serial nature, but this disadvantage can be completely hidden if

the number of bits in each variable vector coefficient is equal or similar to the number of elements in each vector.

As an example of DA mechanization let us consider the computation of the following inner (dot) product of M-dimensional vectors **a** and **x**, where **a** is a constant vector:

$$y = \sum_{k=0}^{M-1} a_k x_k \tag{2.8}$$

Let us further assume that each vector element x_k is an N-bit two's complement binary number and can be represented as

$$x_k = -b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} b_{kn} 2^n \tag{2.9}$$

where $b_{ki} \in \{0,1\}$ is the ith bit of vector element x_k. Please note that b_{k0} is the least significant bit (LSB) of x_k and $b_{k(N-1)}$ is the sign bit.

Substituting eq. 2.9 in eq. 2.8 yields:

$$y = \sum_{k=0}^{M-1} a_k [-b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} b_{kn} 2^n] \tag{2.10}$$

$$y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{k=0}^{M-1} a_k \sum_{n=0}^{N-2} b_{kn} 2^n \tag{2.11}$$

$$y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} [\sum_{k=0}^{M-1} a_k b_{kn}] 2^n \tag{2.12}$$

The interchange of the order of summations in eq. 2.12 is the crucial step which yields a distributed arithmetic computation. Let us consider the term in brackets:

$$q_n = \sum_{k=0}^{M-1} a_k b_{kn} \tag{2.13}$$

Because $b_{kn} \in \{0,1\}$, q_n has only 2^M possible values. Such values can be precomputed and stored in a ROM of size 2^M. The bit serial input data ($\{b_{0i}, b_{1i}, b_{2i}, \ldots, b_{ki}\}$ for $i = 0, 1, \ldots, N-1$) is used to form the ROM address, and the ROM contents can be placed in an accumulator structure to form the outer sum of eq. 2.12. Successive scalings with powers of 2 can be achieved with an arithmetic shifter in the accumulator feedback path. The first term of eq. 2.12 ($\sum_{k=0}^{M-1} a_k b_{k(N-1)}$) is also stored in the ROM at address $\{b_{0(N-1)}, b_{1(N-1)}, b_{2(N-1)}, \ldots, b_{k(N-1)}\}$. Some extra control circuitry is necessary to ensure that the accumulator subtracts (as opposed to adding) the partial sum to the total result at sign bit time. After N cycles (as a reminder N is the bitwidth of the x_k vector elements) the final result y has converged to its final value within the accumulator.

Figure 2.1 shows a detailed example of a Distributed Arithmetic computation. The structure shown computes the dot product of a 4-element vector X and a constant vector

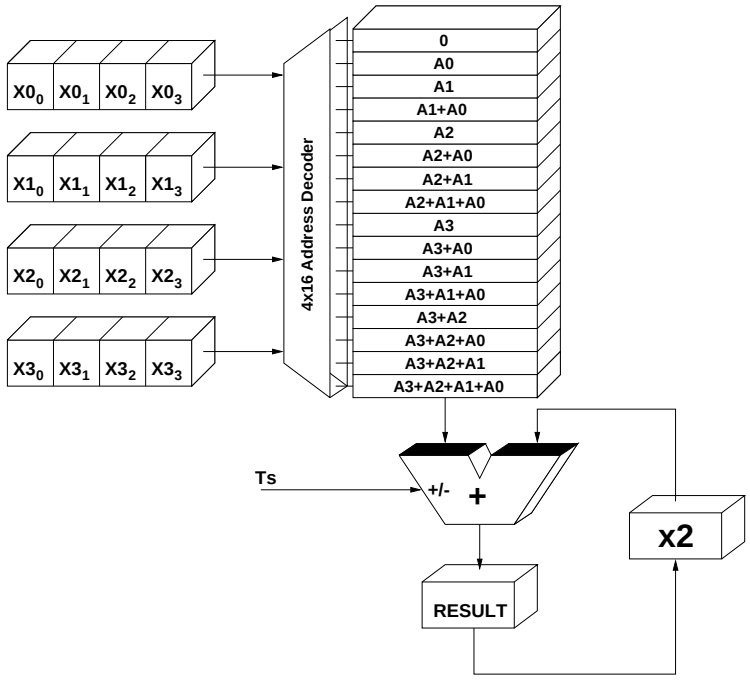


Figure 2.1: Distributed Arithmetic ROM and Accumulator (RAC) Structure

A. All 16 possible linear combinations of the constant vector elements (A_i) are stored in a ROM. The variable vector X is repackaged to form the ROM address most significant bit first. We have assumed that the X_i elements are 4-bits 2's complement (bit 3 is the sign bit.) Every clock cycle twice the value of the RESULT register (which is initially reset to zero) is being added to the current ROM contents. The sum is placed back into the RESULT register. Moreover, each cycle the 4 registers that hold the four elements of the X vector are shifted to the right. The sign timing pulse T_s is activated when the ROM is addressed by bit 3 of the vector elements (sign). In this case the adder subtracts the current ROM contents from the accumulator state. After four cycles (bitwidth of the X_i elements) the dot product has been produced within the RESULT register.

2.2.1 On the Successive Approximation Property of Distributed Arithmetic

The DCT core processor of chapter 5 exploits a significant property of Distributed Arithmetic for power minimization. This property is developed in the present section.

When the Distributed Arithmetic operation is performed MSB first, it exhibits stochastically monotonic successive approximation properties. In other words, each successive intermediate value is closer to the final value in a stochastic sense. An analytical derivation follows:

The ith intermediate result of an MSB-first DA computation ($i > 0$) is:

$$y_i = -q_{(N-1)} + \sum_{n=N-1-i}^{N-2} q_n 2^n \tag{2.14}$$

where

$$q_n = \sum_{k=0}^{M-1} a_k b_{kn} \tag{2.15}$$

Please note that when $i = N-1$, eq. 2.14 yields eq. 2.12.

Let us define an error term e_i, $i = 0, 1, \ldots, N-1$ as the difference between each intermediate value y_i and the final value y:

$$e_i = y - y_i \tag{2.16}$$

$$e_i = \sum_{n=0}^{N-2-i} q_n 2^n \tag{2.17}$$

We model q_n as experimental values of a discrete random variable $\mathbf{q}$. The underlying stochastic experiment is random accesses of the DA coefficient ROM in the presence of random inputs. The experimental values of $\mathbf{q}$ are the DA ROM contents. The first and second order statistics of the error term e_i are:

$$E[e_i] = E[\mathbf{q}] \sum_{n=0}^{N-2-i} 2^n \tag{2.18}$$

$$= E[b_{kn}] \sum_{k=0}^{M-1} a_k \sum_{n=0}^{N-2-i} 2^n \tag{2.19}$$

$$= \frac{2^{N-1-i}-1}{2} \sum_{k=0}^{M-1} a_k \tag{2.20}$$

$$\sigma_{e_i}^2 = \sigma_{\mathbf{q}}^2 (1 + 4 + \cdots + 2^{2(N-2-i)}) \tag{2.21}$$

$$= \frac{2^{2(N-1-i)}-1}{3} \sigma_{\mathbf{q}}^2 \tag{2.22}$$

$$= \frac{2^{2(N-1-i)}-1}{3} Var\left[\sum_{k=0}^{M-1} a_k b_{kn}\right] \tag{2.23}$$

$$= \frac{2^{2(N-1-i)}-1}{3} \sum_{k=0}^{M-1} a_k^2 \sigma_{b_{kn}}^2 \tag{2.24}$$

$$= \frac{2^{2(N-1-i)}-1}{12} \sum_{k=0}^{M-1} a_k^2 \tag{2.25}$$

where equations 2.20 and 2.25 have been computed under the assumption that the least significant bits b_{kn} (i large) are independent identically distributed random variables uniformly distributed between 0 and 1 ($E[b_{kn}] = 1/2,\ \sigma_{b_{kn}}^2 = 1/4$). This is a valid assumption for input DSP data [LR95]. The fact that equations 2.20 and 2.25 are monotonically decreasing functions of i (RAC cycles) shows the successive approximation property (in probabilistic terms) of the Distributed Arithmetic mechanization.

2.3 Quantization

The process of scaling the DCT coefficients and truncating them to integer values is called quantization. The process of rescaling to restore approximately the original coefficient magnitude is called inverse quantization. In the image and motion picture coding context, quantization is invariably uniform: The decision and the reconstruction intervals are identical and equal to the quantizer step size Δ. Such quantization and subsequent inverse quantization can be described with the following operation:

$$\hat{x} = \lfloor \frac{x + 0.5\Delta}{\Delta} \rfloor \Delta \tag{2.26}$$

where x is the input value, $\hat{x}$ is the quantized output value and Δ is the quantizer step size. Uniform quantization is not necessarily optimum in terms of minimizing some measure of the quantization error. The following section describes an important method of designing minimum distortion quantizers given knowledge of the statistical properties of the input data. This method is referenced widely in chapter 4 and we present it here for completeness.

2.3.1 The Lloyd-Max Optimum Quantizer

Max [Max60] and Lloyd [Llo82] have independently derived methodologies of constructing minimum distortion quantizers given a fixed number of quantization intervals and a probability distribution for the input signal.

Max has derived a framework for optimum quantization given any distortion criterion (where distortion D is defined as the expected value of any monotonically increasing function of the difference between the quantizer input and output) whereas Lloyd has limited his treatment to minimizing the mean square error (MSE). Both arrive at the same results when D= MSE.

The problem of optimum quantization in the mean square error sense can be stated as follows:

A quantization scheme consists of a class of sets $\{Q_1, Q_2, \cdots, Q_N\}$ and a set of quanta $\{q_1, q_2, \cdots, q_N\}$. The ranges $\{Q_a\}$ are disjoint and cover a continuous axis in its entirety. The quanta $\{q_a\}$ are any N finite discrete values. Let x (the input to the quantizer) be a random variable (sample of a stationary stochastic process) with a probability density function $f_x(x_0)$. The output of the quantizer must be a quantum q_a such that the quantization noise N given by the following equation is minimized:

$$N = \sum_{a=1}^{N} \int_{Q_a} (q_a - x)^2 f_x(x) dx \tag{2.27}$$

The problem of optimum quantization given a fixed number of ranges and quanta N and a fixed PDF $f_x(x_0)$ is reduced to finding $2N-1$ numbers

$$q_1 < x_1 < q_2 < x_2 < \cdots < q_{N-1} < x_{N-1} < q_N, \tag{2.28}$$

where the $\{x_a\}$ are the endpoints of the $\{Q_a\}$ intervals and $\{q_a\}$ are the corresponding quanta such that the following expression (mean square quantization error) is minimized:

$$\int_{-\infty}^{x_1}(q_1-x)^2 f_x(x)dx+\int_{x_1}^{x_2}(q_2-x)^2 f_x(x)dx+\cdots+\int_{x_{N-1}}^{\infty}(q_N-x)^2 f_x(x)dx \tag{2.29}$$

Both Max and Lloyd prove that two necessary but not sufficient conditions for minimizing expression 2.29 are:

$$q_a = \frac{\int_{Q_a} x f_x(x)dx}{\int_{Q_a} f_x(x)dx}, \quad a=1,2,\cdots,N \tag{2.30}$$

$$x_a = \frac{q_a+q_{a+1}}{2}, \quad a=1,2,\cdots,N-1 \tag{2.31}$$

Equations 2.30 and 2.31 are known as the Lloyd-Max equations and they exhibit intuitive sense: Eq. 2.30 implies that each quantum q_a must be the "center of mass" of each range Q_a. This is a classical result attributed to the minimization of The "moment of inertia" in elementary mechanics. Eq. 2.31 says that the interval endpoints must bisect the distances between quanta. This is also an intuitive result.

Equations 2.30 and 2.31 suggest a trial and error iterative method for finding local noise minima. We choose a value for q_1 ($q_1 < \int_{-\infty}^{\infty} x f_x(x)dx$ according to Lloyd [Llo82]) and then solve the following equation for x_1 under the condition that q_1 is the center of mass of Q_1:

$$q_1 = \frac{\int_{-\infty}^{x_1} x f_x(x)dx}{\int_{-\infty}^{x_1} f_x(x)dx} \tag{2.32}$$

Eq. 2.31 then suggests that q_2 can be determined as

$$q_2 = 2x_1 - q_1 \tag{2.33}$$

If q_2 lies to the right of the center of mass of the interval (x_1,∞) then the process stops and starts over again with a different trial value q_1. Otherwise x_2 is computed and the process continues:

$$q_2 = \frac{\int_{x_1}^{x_2} x f_x(x)dx}{\int_{x_1}^{x_2} f_x(x)dx} \tag{2.34}$$

$$q_3 = 2x_2 - q_2 \tag{2.35}$$

$$\vdots$$

$$q_N = 2x_{N-1} - q_{N-1} \tag{2.36}$$

If at the and of this process q_N ends up being the center of mass of interval (x_{N-1},∞)

$$q_N = \frac{\int_{x_{N-1}}^{\infty} x f_x(x)dx}{\int_{x_{N-1}}^{\infty} f_x(x)dx} \tag{2.37}$$

then y_1 has been chosen appropriately and points q_a, x_a constitute a local noise minimum. However, in general eq. 2.37 won't be satisfied and the process must be restarted with a different starting value until the discrepancy is reduced to zero. Solutions of the Lloyd-Max equations are carried exclusively on fast electronic computers. Improved methods for fast convergence of this algorithm have been suggested [NL86].

2.4 Image Quality Measure

A very common measure of processed (i.e. compressed) image quality vs. the original image is the peak SNR (PSNR) defined below:

$$PSNR = -10log_{10}\left[\frac{\sum_{i=1}^{i=N}(q[i]-q'[i])^2}{(2^n-1)^2 \times N}\right] \tag{2.38}$$

where N is the total number of picture elements in the image (pels), $q[i]$ is the ith pel of the original image, $q'[i]$ is the ith pel of the processed image, and n is the pel bitwidth.

Although the PSNR is not perfectly correlated with visual image quality, it is the most widely accepted arithmetic measure and will be used extensively in this work. The best measure of image quality is human subject evaluation under well defined viewing conditions (distance, angle and lighting.)

2.5 The MPEG Video Compression Standard

The MPEG [ISO94][LeG91][MPFL97] video compression standard combines two of the methods described in this chapter (DCT and quantization) along with motion estimation and entropy (Huffman [Huf52]) coding. The DCT removes the spatial redundancy present in motion pictures and motion compensation removes the temporal redundancy. Quantization introduces extra compression due to precision and minimal quality reduction. Finally, entropy coding introduces a final step of additional lossless compression.

A video sequence is divided into one or more groups of pictures (GOPs) and each GOP is composed of one or more pictures of three different types (I, P, and B). An I picture (intra-coded) is coded independently without any reference to any other adjacent pictures. A P picture (predictive-coded) is compressed by coding the differences between the present picture and the I or P picture which temporally precedes it. A B-picture (bidirectionally predictive-coded) is compressed by coding the differences between the present picture and the nearest preceding and / or upcoming I or P picture in the sequence. Disjoint regions of B pictures can use different predictors from previous and / or future pictures. Since the compression algorithm may sometimes use prediction information from future pictures, the coding order is not the same as the display order. An example group of pictures along with prediction dependencies is illustrated in Figure 2.2.

The basic unit of an MPEG picture is the macroblock. It is the unit where motion estimation / compensation is performed. It consists of a 16×16 array (4 8×8 blocks) of luminance samples (Y) plus 2 8×8 blocks of chrominance samples (Cr, Cb). The entire macroblock represents a 16×16 array of pels within a single picture. The 8×8 block (either Y, Cr or Cb) is the unit where the DCT / quantization is performed. It is this basic unit that the chips of chapters 3 and 5 operate on.

Motion-compensated prediction exploits the temporal redundancy of video sequences. The assumption is that the current picture can be *locally* modelled as a translation of a

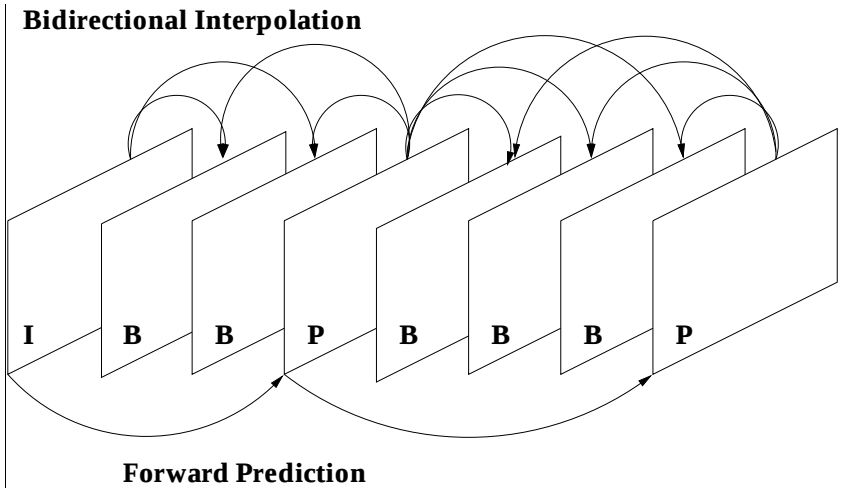


Figure 2.2: MPEG Group of Pictures. Figure reproduced from [ISO94]

previous or future picture. Each macroblock in the MPEG bitstream carries a *motion vector* which is a horizontal and vertical index of a particular 16×16 pel array (not necessarily on a macroblock boundary) in a past or future I or P image (P-coded pictures may only reference past I, P pictures.) The macroblock data is differentially coded and added to the predictor as the final frame reconstruction step.

Chapter 3

IDCT Core Processor

In this chapter, we present the design of an IDCT macrocell suited for mobile applications [XC98] [XC99]. We present the inverse transform computation first, because the implementation of the IDCT chip predated the implementation of the DCT processor. Section 3.1 reiterates some general background information and presents the IDCT algorithm used. Section 3.2 summarizes the chip architecture. Section 3.3 presents some interesting aspects of the IDCT chip circuit design. Section 3.4 discusses power estimation methodology issues and results. Sections 3.5 and 3.6 discuss chip usage and packaging. Section 3.7 presents laboratory test results. Finally, section 3.8 summarizes and concludes the chapter.

Our strategy for reducing the chip power was two-fold: First, we selected an IDCT algorithm that minimizes activity by exploiting the relative occurrence of zero-valued DCT coefficients in compressed video. Past IDCT chips (e.g. [MHS+94], [BH95], [KFN+96]) have relied on conventional fast IDCT algorithms that perform a constant number of operations per block independent of the data distribution. Our selected algorithm described in section 3.1 performs a variable number of operations that depends on the statistical properties of the input data. We implemented this algorithm in hardware through the use of extensive clock gating that minimized the average switched capacitance per sample.

Second, previously proposed architectural-driven voltage scaling techniques [CSB92] [CBB94] have been adapted and used for power minimization. Deep pipelining and appropriate circuit techniques have been employed so that the chip could produce 14 Msamples / sec (640x480, 30 fps, 4:2:0) at 1.3V (3.3V process) and meet the requirement for MPEG2 MP@ML.

3.1 Background and Algorithmic Design

For convenience, we briefly repeat the IDCT definition that has appeared in chapter 2 The 8-point 1-D Inverse DCT [ANR74] of a coefficient vector X is given by the following equation:

$$x[n] = \sum_{k=0}^{7} \frac{c[k]}{2} X[k] \cos(\frac{(2n+1)k\pi}{16}) \tag{3.1}$$

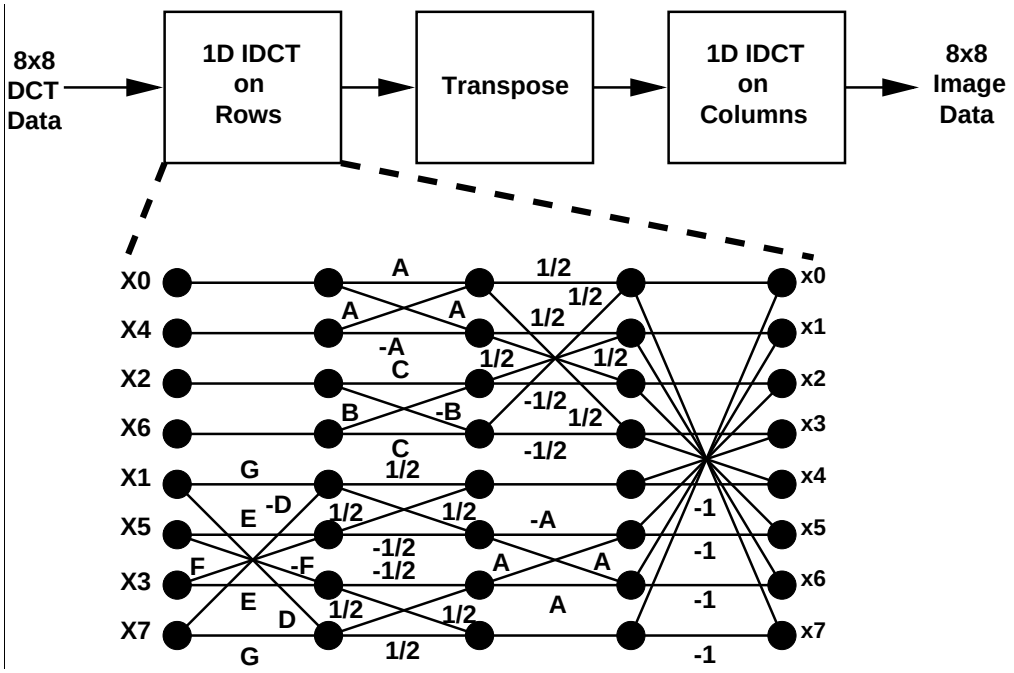


Figure 3.1: 2D Chen Separable IDCT Algorithm

where $c[k] = 1/\sqrt{2}$ if $k = 0$ and 1 otherwise. Since the DCT is a separable transform, the 2-D IDCT of an 8×8 coefficient block can be computed by applying equation 3.1 on each block row, transposing the intermediate result and reapplying equation 3.1 on each column.

There have been numerous fast IDCT algorithms that minimize the number of multiplications and additions implied by equation 3.1 [CSF77] [FW92]. A popular algorithm which has been embedded in various VLSI implementations [SCG89] [UIT+92] is the Chen algorithm [CSF77] depicted in Figure 3.1. The Chen algorithm is based on a sparse matrix factorization of the matrix form of equation 3.1. Letters $A - G$ represent constant values and each dot in the flowgraph represents a dual multiply-and-sum operation. The Chen algorithm requires a constant number of operations per block: 256 multiplications and 416 additions.

The statistical distribution of the input DCT coefficients possesses unique properties that may affect IDCT algorithm design. Previous work [RG83] [SR96] [NH95] has established that AC (non-zero spatial frequency) DCT coefficients of compressed images and video exhibit zero-mean Laplacian distribution:

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x|} \qquad \lambda = \frac{\sqrt{2}}{\sigma_x} \tag{3.2}$$

Muller [Mul93] has proposed a slightly different statistical model based on Kolmogorov-Smirnov (KS) and χ^2 fitness tests [KS67]:

$$f(x) = \frac{\nu\alpha(\nu)}{2\sigma\Gamma(\frac{1}{\nu})} e^{-[\alpha(\nu)|\frac{x}{\sigma}|]^\nu} \tag{3.3}$$

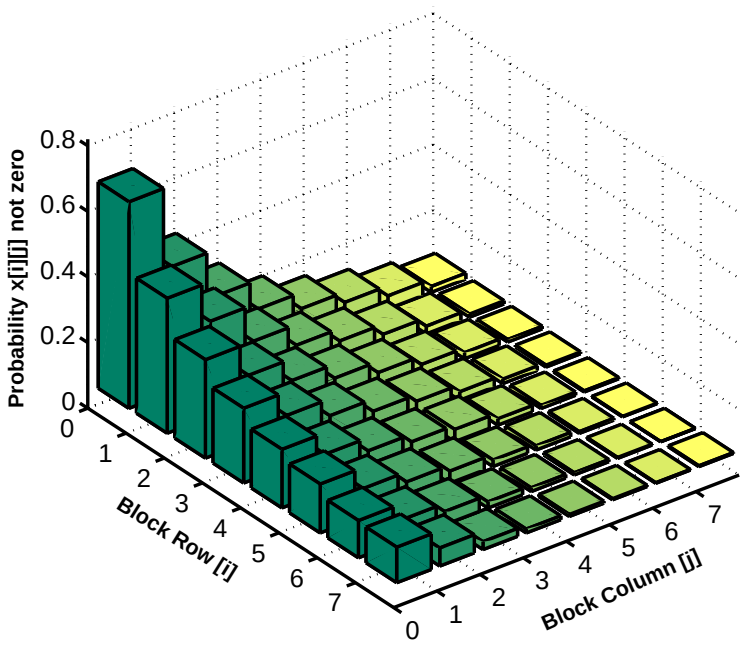


Figure 3.2: Probabilities of Non-Zero Occurrence for 8×8 DCT Coefficient Blocks

$$\alpha(\nu) \quad = \quad \sqrt{\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}} \tag{3.4}$$

where $\Gamma(.)$ denotes the gamma function and ν and σ are positive constants. Equation 3.3 denotes a generalized Gaussian distribution. Choice of parameter ν transforms equation 3.3 into a Laplacian or a Gaussian ($\nu = 1$ or $\nu = 2$ respectively). Lee et. al [LKRR93] have also concluded that the generalized Gaussian distribution (3.3) is valid for a class of medical images. Appendix B contains a complete set of histograms for all 63 AC DCT coefficients of the natural image PEPPERS plotted along with the corresponding zero-mean dual-sided Laplacian distribution of equal variance.

Sharp zero-centered dual-sided Laplacian distributions for 63 out of 64 spectral coefficients implies a large number of zero-valued coefficients per 8×8 block for image and video data. Typically, DCT blocks of MPEG-compressed video sequences have only 5-6 non-zero coefficients, mainly located in the low spatial frequency positions. The three-dimensional histogram of Figure 3.2 plots the probability of occurrence of a non-zero spatial frequency for each block position for a typical video sequence. Higher probabilities are concentrated in the low spatial frequency range and this is consistent with the spatial low pass characteristics of video sequences. The histogram of Figure 3.3 shows the frequency of block occurrence plotted vs. the number of non-zero coefficient content for a typical MPEG sequence. The mode of such distributions is invariably blocks with a single non-zero spectral coefficient (typically the DC.)

Based on the information above on the statistical distribution of DCT coefficients, we

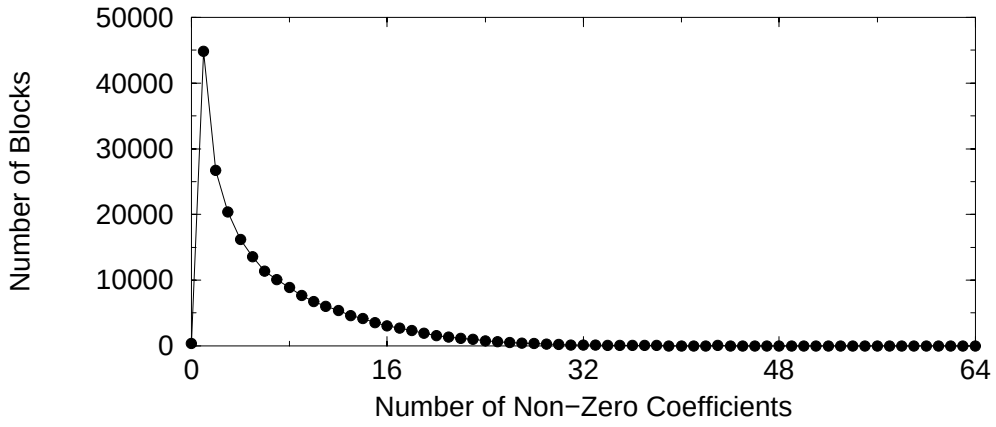


Figure 3.3: Histogram of Non-Zero DCT Coefficients in Sample MPEG Stream

decided to depart radically from conventional IDCT algorithms that perform a fixed number of operations per block. Given such input data statistics, we observe that direct application of equation 3.1 will result in a small average number of operations since multiplication and accumulation with a zero-valued $X[k]$ coefficient may constitute a NOP. Appropriate implementation of equation 3.1 can result in an algorithm with a variable number of operations per block as opposed to standard fast algorithms that cannot exploit data properties. This idea of a coefficient-by-coefficient IDCT algorithm has originated with McMillan and Westover [MW92] in the context of a software implementation. The McMillan and Westover forward-mapped IDCT (FMIDCT) algorithm is described in section 2.1.1. The IDCT chip presented here consists of two 1-D IDCT units implementing equation 3.1 and a transposition memory structure in between. On the other hand, conventional architectures perform a constant number of operations per sample and do not exploit data properties. Butterfly-style operations absorb the zero-valued coefficients early in the signal path without affecting average switched capacitance. Figure 3.4 shows the instantaneous and average number of multiplications and additions required by the data-driven algorithm of equation 3.1 for 1000 blocks extracted from an MPEG video sequence. We observe that the data-driven algorithm requires a smaller number of operations per block compared to the conventional Chen algorithm. The potential for a low power implementation of this algorithm is evident due to the small arithmetic operation requirements.

3.2 Chip Architecture

A block diagram of the IDCT chip is shown in Figure 3.5. Incoming coefficients are scaled by a maximum of four constants at a time for a maximum of seven different constant scalings (C0-C6) due to the symmetry of the cosine basis functions. The scaled values are being added or subtracted to the state of 8 accumulators attached to the scaling multipliers through a full crossbar switch. The scaling constants and the crossbar setting are determined by the index k of each coefficient $X[k]$ within the 8-point vector X. Every 8 cycles, the

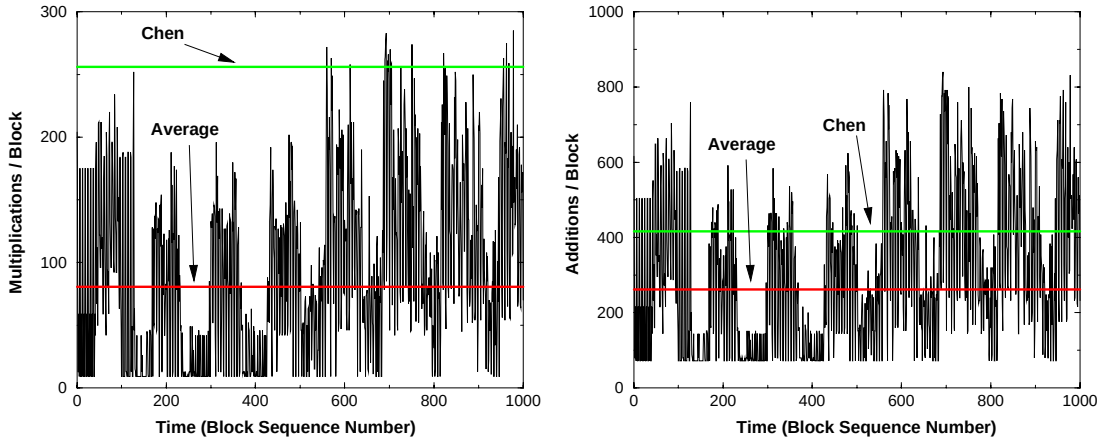


Figure 3.4: Number of Operations Comparison Between Chen and Data-Driven IDCT

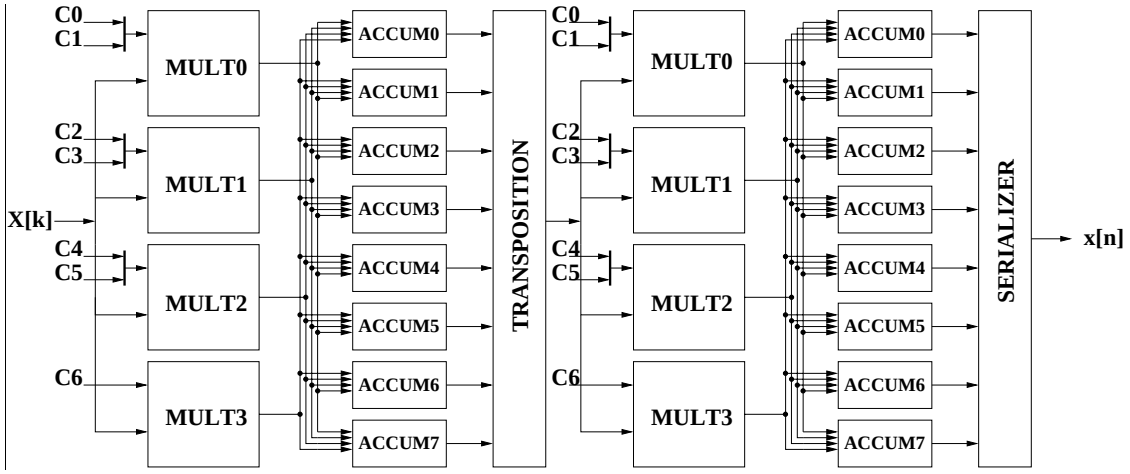


Figure 3.5: IDCT Chip Block Diagram

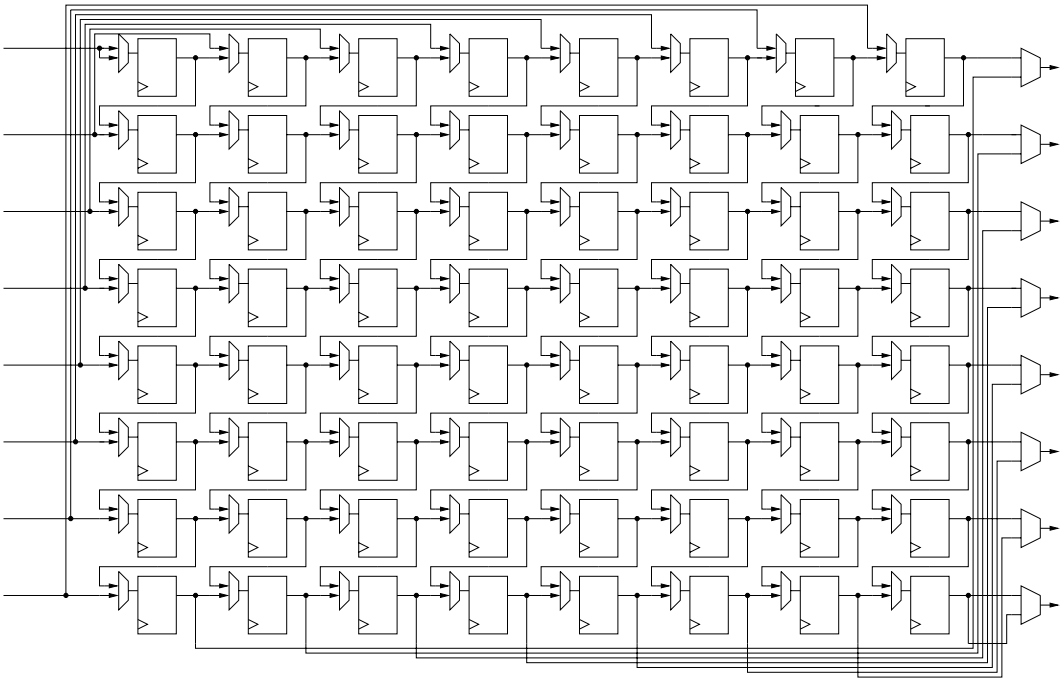


Figure 3.6: Transposition Structure (TRAM). Data is Transposed On-The-Fly by Changing the Shifting Direction (T $\rightarrow$ B or L $\rightarrow$ R.) This scheme eliminates the need for ping-pong buffering.

1-D IDCT of a block row has converged within the 8 accumulators. The intermediate result is fed into the transposition structure (TRAM) of Figure 3.6. This structure is a 2-D array of shift registers that can shift data from left to right and top to bottom. The TRAM performs transposition on-the-fly and *eliminates* the need for double buffering which would have been necessary had a static dual-addressed RAM been used. The second 1-D stage is identical to the first except for a change in the precision of the scaling constants.

The chip input spectral coefficients are 12-bit 2's complement integers. Before the multiply-accumulates of the first stage, the data is converted to 12-bit sign magnitude representation. The scaling constants are 14-bit sign-magnitude. After the completion of the first stage computation, the intermediate data is 16-bit 2's complement. Before entering the second state the data is shifted and rounded to 14 bits and converted to sign magnitude. The scaling constants of the second stage are 13-bit sign magnitude. The final result is rounded to 10-bit 2's complement representation before being shipped off-chip.

In order to meet the bandwidth requirement of 14 Msamples/sec at V_{DD}=1.3V using a high VT process, the multiply-accumulate operations must be pipelined by a factor of 4. Figure 3.7 shows the multiply-accumulate (MAC) structure used. For simplicity, the diagram does not show the crossbar switch between the multiplier and the adder structures. The Figure depicts a 5×4 unit as opposed to the full 14×14 structure used. The level of pipelining is identical. The multiplier is a standard carry-save array multiplier.

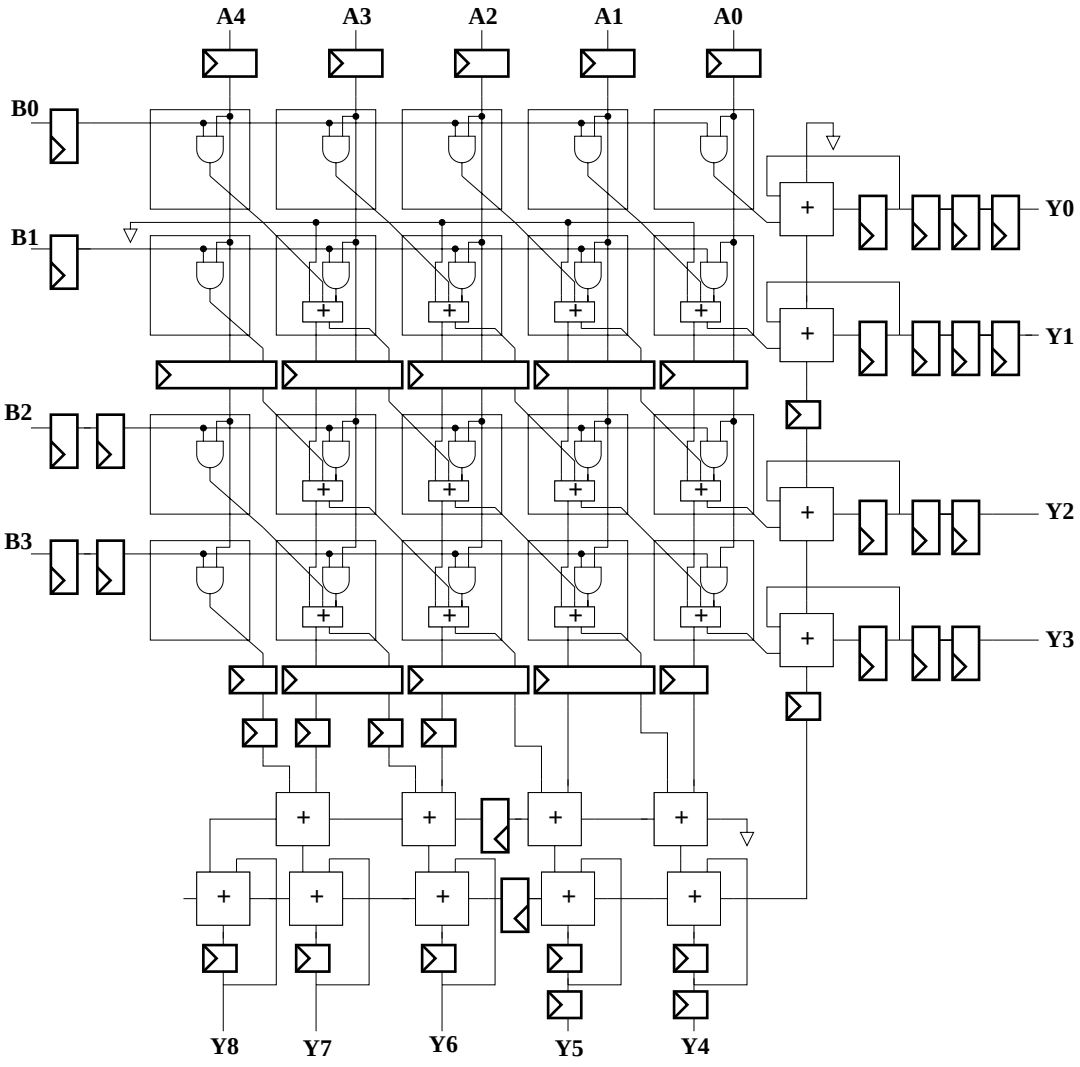


Figure 3.7: Pipelined Multiply-Accumulate (MAC) Structure

Dataset	PMSE <0.16	OMSE <0.02	PME <0.015	OME <0.0015
$[-256,255]$	0.015800	0.013597	0.002500	0.000153
$[-5,5]$	0.011000	0.009136	-0.002400	-0.000005
$[-300,300]$	0.014100	0.011886	0.002700	0.000180
$-[-256,255]$	0.016000	0.013578	-0.002500	-0.000122
$-[-5,5]$	0.011000	0.009156	0.002500	0.000016
$-[-300,300]$	0.014000	0.011878	-0.002900	-0.000122

Table 3.1
IEEE Standard 1180-1990 Compliance

The accumulator structure is a simple ripple carry adder. The accumulator adds a single full adder delay to the critical path due to the coincident carry propagation with the availability of product bits of the multiplier. Although pipelining increases the total switched capacitance of the computation unit, it has the side benefit of reducing the propagation of spurious transitions within the MAC structure.

The datapath width of the arithmetic units have been optimized to meet IEEE Standard 1180-1990 for IDCT precision by a comfortable margin but at the same time discard unnecessary precision bits that would increase the power dissipation. Table 3.1 summarizes our fixed point precision deviations. All datasets pass the pixel peak error requirement. The zero-input requirement is also passed.

3.2.1 Clock Gating

The presence of many zero-valued coefficients must be exploited in order to reduce the switching activity and reap the low power benefits of this particular algorithm. If the MAC structure of Figure 3.7 was not pipelined, conditional gating of the accumulator clock based on whether the incoming coefficient is zero or not would substantially reduce the switching activity and still produce the correct result. Pipelining though makes clock gating more complicated. If all pipeline stages are clocked with a single clock net, clock gating becomes impossible since different pipeline stages are processing different and possibly non-zero DCT coefficients. Powering down all the stages will produce erroneous results. Clock gating can be implemented if each pipeline stage uses a separate clock net gated by an appropriate qualifying pulse. The qualifying pulse propagates from stage to stage along with the non-zero coefficient that requires processing. If a zero-valued coefficient enters the pipeline, only the stage that corresponds to the zero is powered down. The other upstream and downstream stages remain unaffected. Our clock gating scheme is graphically depicted in Figure 3.8. The IDCT chip features ten separate clock nets in addition to the master clock for fully qualifying all steps of the entire pipeline formed by both 1-D stages.

A clock-gated pipeline presents certain physical design challenges in order to avoid common race conditions. Race conditions may arise in the following situation which is depicted in Figure 3.9(a): Let us assume that $CLK0$ arrives at the clocked element of stage 0

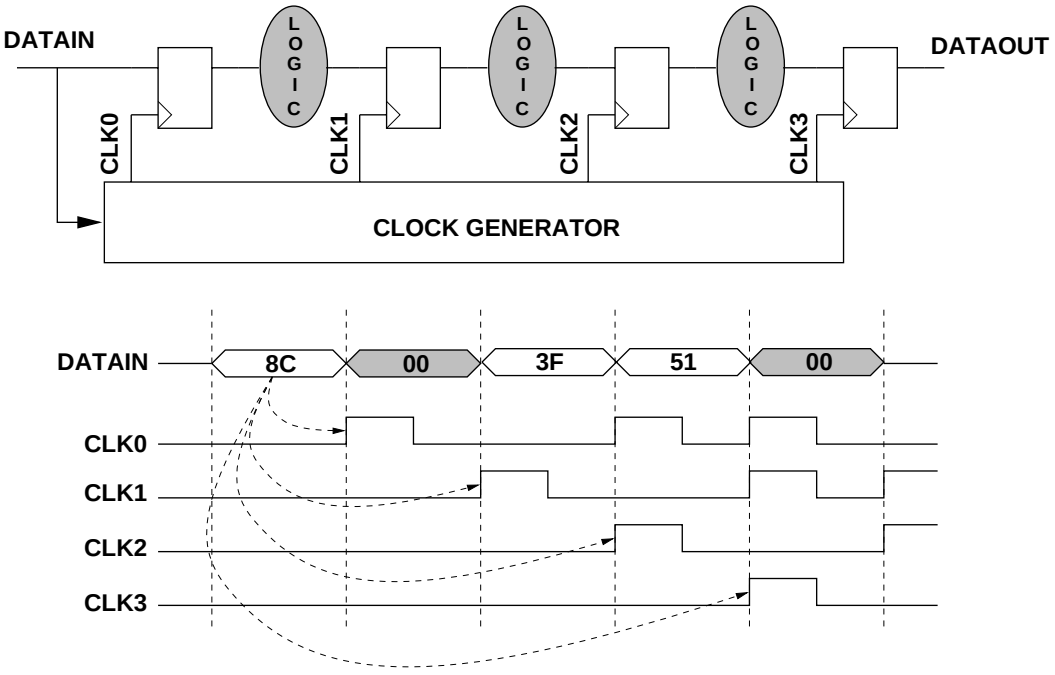


Figure 3.8: Clock Gating Approach in a Pipeline

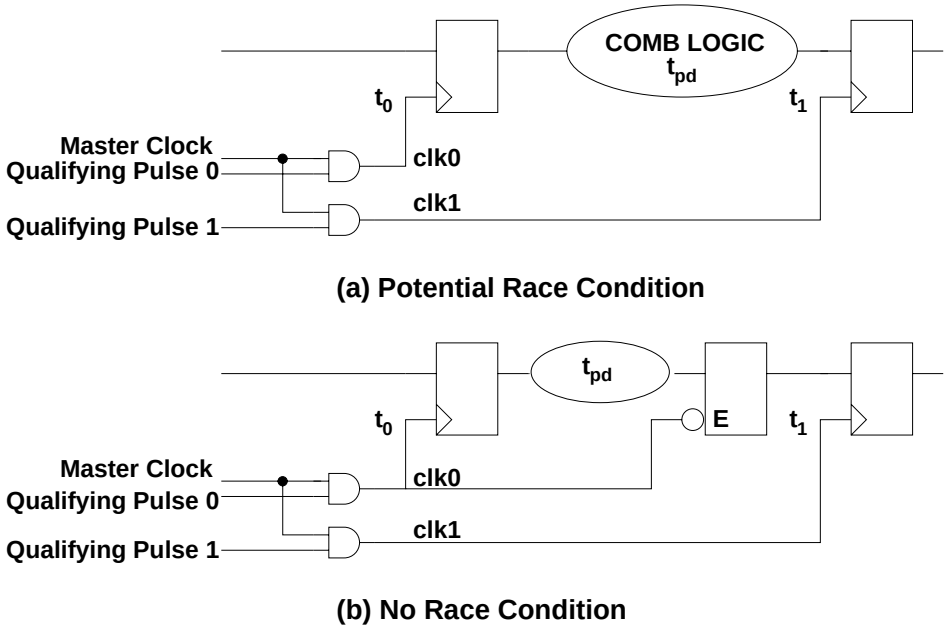


Figure 3.9: Potential Race Conditions in Clock-Gated Pipelines

at time t_0 and $CLK1$ arrives at time t_1. If $t_1 - t_0 > t_{CLK \to Q} + t_{pd} - t_{hold}$, the wrong data will be sampled at stage 1. The expected value of $\Delta t = t_1 - t_0$ can be greater than typical clock skew values because the clock nets are physically separate and are affected by more mismatch phenomena than distributed RC delays on a nominally equipotential surface. Moreover, in the IDCT chip all gated clock nets are global spanning all the pipelines formed by the accumulators. The gate load seen by each clock net is quite different due to data skewing (Figure 3.7) in addition to different physical wire loading. This problem becomes very important when the intermediate combinational logic is minimal or non-existent (i.e. back-to-back shift registers) and $t_{pd} \to 0$.

We solved this problem by careful buffering and physical design of the clock nets and multiple simulation iterations with accurate extracted parasitics. In cases of small timing margins (i.e. shift registers and least significant bits of adder stages), a negative level-sensitive latch was inserted clocked by the upstream pipeline clock as shown in Figure 3.9(b). This ensured functional correctness with a minimal penalty ($<$2%) in terms of power dissipation and no effect on the system critical path.

3.3 Circuit Design

This section presents interesting aspects of the IDCT chip circuit design.

3.3.1 Arithmetic Circuits

The greatest circuit challenge presented was the design of the pipelined multiply-accumulate unit shown in Figure 3.7 given $V_{GS} - V_T$ =0.4V for PMOS devices and 0.6V for NMOS devices. The critical path for this operation is 7 full adder delays given a 14×14 pipelined multiplier. We imposed the following requirements on the full adder design used in the multipliers (Figure 3.10): First, since the sum output (S) propagation delay is also critical (in addition to the $\overline{COUT}$) we wanted S to be computed using a single gate, as a function of A, B and CIN. A second gate delay would increase the critical path substantially and would force us to raise the power supply above 1.3V. Second, the S gate should not employ more than 2 series PMOS pullup devices in the signal path. PMOS transistors see at maximum 0.4V above threshold across their gate and source terminals and their mobility is three times smaller than that of the NMOS devices. Given these facts, we observed that more than two series PMOS devices in the sum circuit would be considerably slow in pulling up the sum output unless substantially oversized, which would considerably increase the total cell switched capacitance. Finally, we wanted the sum gate to be fully static. Although a dynamic implementation would remove weak PMOS devices from the signal path, the non-monotonicity of the S gate would introduce extra design complications and circuit overhead.

Given all the constraints above, we decided to use the hybrid full adder circuit of Figure 3.10. In this circuit, $\overline{COUT}$ is derived in a static CMOS fashion but $\overline{S}$ is derived using Cascade Voltage Switch Logic (CVSL). The advantage of using this logic family for the sum

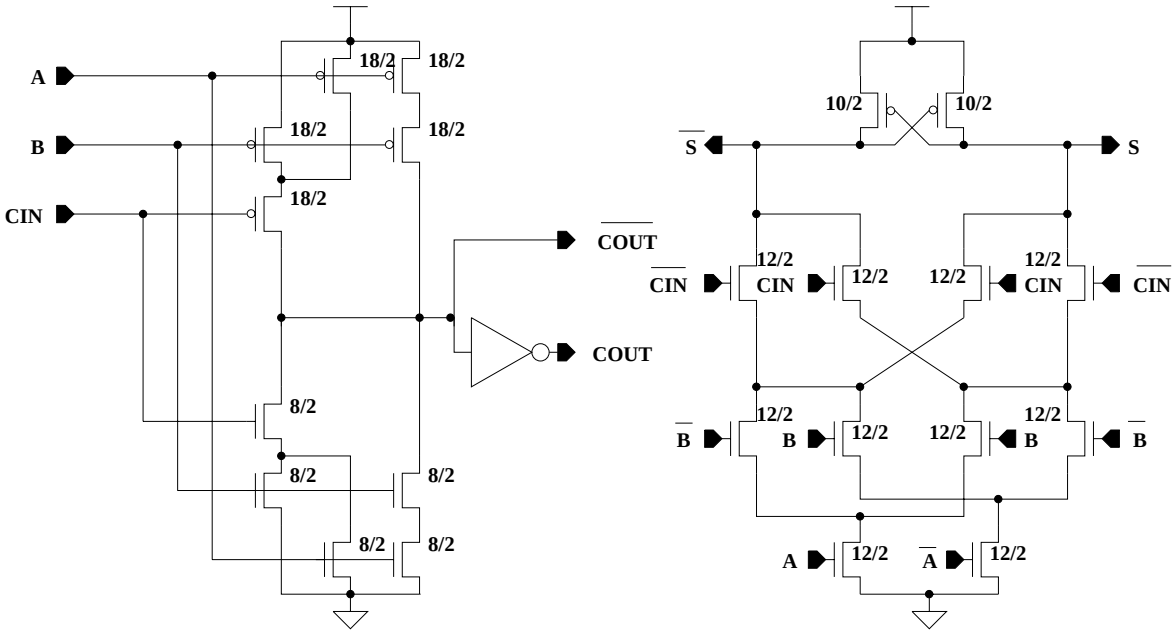


Figure 3.10: Hybrid Adder Schematics (sizes in $\lambda = 0.35\mu$)

output is that S is pulled high indirectly through a stack of NMOS transistors overpowering a weak PMOS cross-coupled pair as opposed to a stack of PMOS devices pulling high. The main disadvantages are that proper ratioing is needed for different supplies, the fact that the complementary NMOS switch structures require complementary inputs and the presence of small crossover currents. Since these full adders are used within an array multiplier, one of the complementary inputs is the S output of the previous row adder which is already in complementary form. Only $\overline{COUT}$ needs to be complemented in order to be used as an input in the next row of adders. It must be noted that the total device count has not been increased from the popular 24-transistor inverting full static CMOS adder implementation ("mirror adder" [WE93]). Although CVSL can result in inefficient implementations of native sum-of-products expressions, it is ideally suited for multiple-input XOR functions such as a sum computation.

Figure3.11 shows an Hspice simulation ($\overline{S}$ sum output) of both the mirror and the hybrid adder for typical load conditions at V_{DD}=1.3V. The mirror adder employs a stack of 3 PMOS devices pulling $\overline{S}$ high when all three inputs are low. The circuit simulated is shown in Figure 3.12. Table 3.2 summarizes the simulated rise and fall delays. The small discrepancy between the rise and fall delay of the hybrid adder output is attributed to slight mismatches in the arrivals of the A and $\overline{A}$ inputs in our simulation setup. The hybrid adder exhibits a slower fall delay because the NMOS pulldown stack needs to overpower the PMOS pullup. On the other hand, the rise time is faster as expected for a total worst case speedup of 37%. At the same time the transistor sizes are much smaller in the hybrid adder for additional area and power savings. The use of the hybrid adder of Figure 3.10 helped us achieve real-time performance at 1.3V.

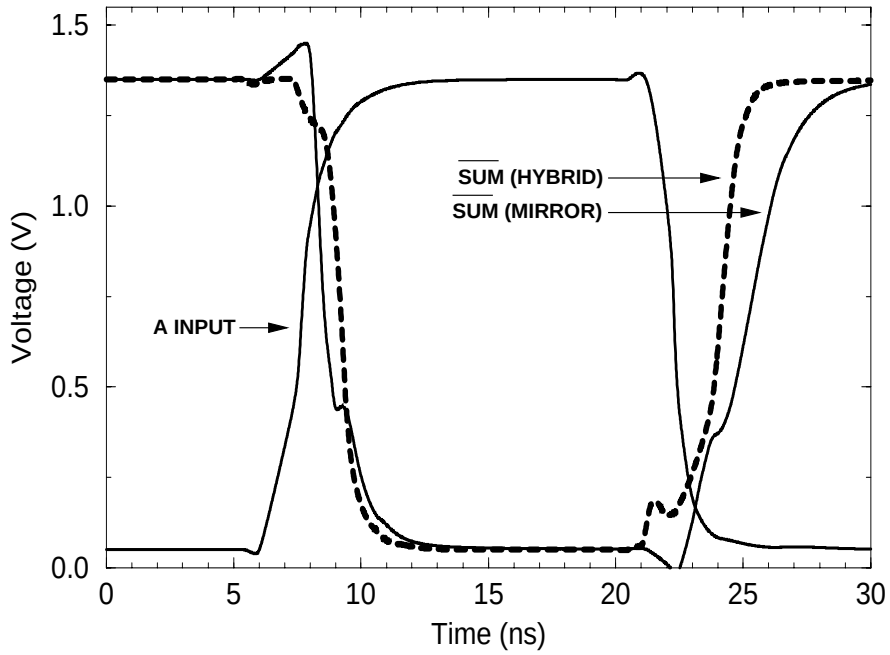


Figure 3.11: Hspice Simulation of CMOS vs. Hybrid Adder

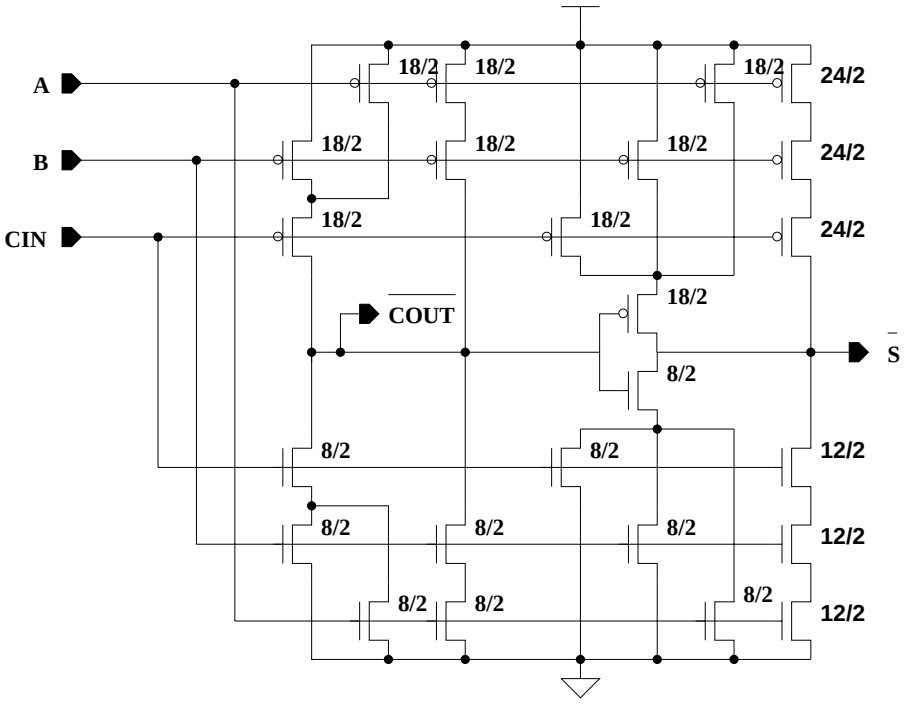


Figure 3.12: Mirror Adder Schematics (sizes in $\lambda = 0.35\mu$)

Circuit	Fall Delay (ns)	Rise Delay (ns)
Static CMOS	0.9	2.93
Hybrid	1.53	1.85

Table 3.2
Rise and Fall Delays for Hybrid and CMOS Adders

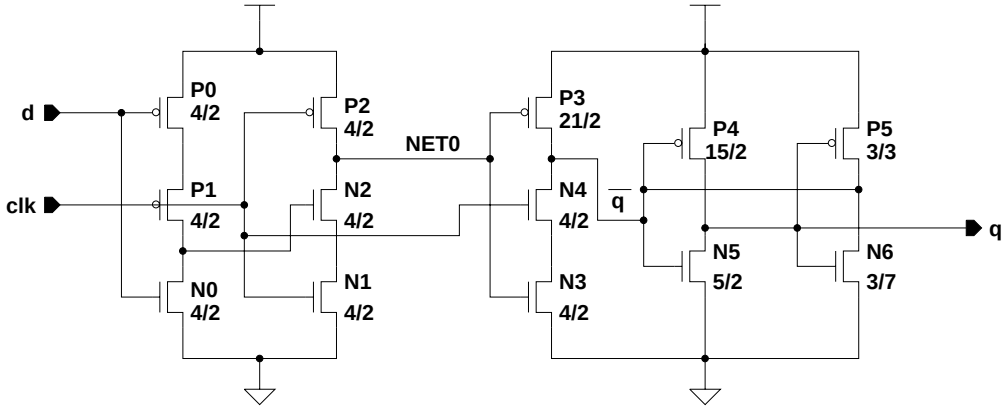


Figure 3.13: Basic TSPC Flop Used in the IDCT Chip (sizes in $\lambda = 0.35\mu$).

3.3.2 Flip-Flops

The basic static edge-triggered flip-flop used in the IDCT chip is a version of the true single phase clock (TSPC) flops presented in [YS89]. It is shown in Figure 3.13. Its basic advantage for low power is the fact that a clock complement is not necessary.

When clock is low, transistors N0, P0, P1 form a transparent inverting latch and node NET0 is precharged. When clock goes high, if data had been high, then node NET0 stays precharged and the logic one propagates to the output q. If data had been low, node NET0 is discharged through N2, N1 and a logic 0 propagates to the output. Weak transistors P5 and N6 staticise the output node. Transistor P3 is made large enough to overpower the weak staticizer even at low supply voltages.

This flop has an internal race condition and care should be exercised before it is ported to different processes. When d is low and clock goes from low to high, the following race condition may occur: Transistors N4 and N3 are both active at the same time and start pulling node $\overline{q}$ low. Then, transistors N2 and N1 (which are both active too) start discharging node NET0 which brings node $\overline{q}$ back to logic one. The overall effect is a 0-1-0 glitch at the flip flop output on the rising edge of clock when d is low. To avoid the glitch, the N2, N1 pulldown path must be made faster than the N4, N3 pulldown path by increasing the sizes of the N2, N1 transistors. In Figure 3.13 we avoid the glitch partly due to the staticizer which slows down considerably the N4, N3 discharge path.

Figure 3.14 shows a TSPC with asynchronous clear used in the chip. Transistors P6 and

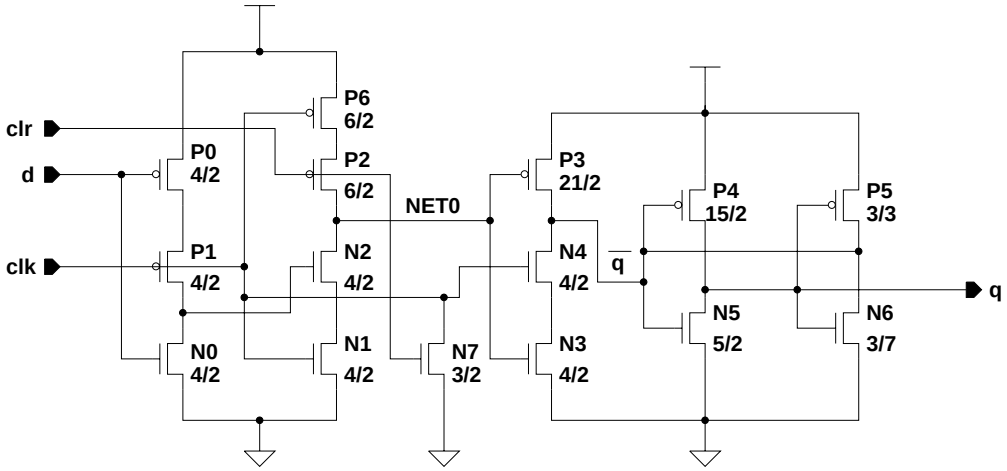


Figure 3.14: TSPC Flop with Asynchronous Clear (sizes in $\lambda = 0.35\mu$).

N7 add the capability to set the staticised output to logic zero on assertion of the clr signal (active high).

3.3.3 I/O Pads

Figure 3.15 shows the level-converting output pad used in the IDCT chip. This design (both schematic and layout) has been adapted from [Gea96]. Two differential-amplifier-style level converters are used [CBB94] to raise the voltage from VDD (1.1-2.2V) to VHH (5V). A split-output exponential buffer drives the output bonding pad.

Figure 3.16 shows the level-converting input pad. Primary ESD protection is provided by bipolar diodes D1 (N-diffusion to P-substrate), D2 (P-diffusion to N-well) and N-well resistor R. Diodes D1, D2 protect the input of the sensing inverter from straying more than 0.6V (either direction) from its specified range (0-VHH).

3.4 Power Estimation Methodology

The IDCT chip is a reasonably large system (160K transistors) and needs to be simulated for a large number of cycles so that conclusions regarding correlation of power dissipation vs. processing load can be drawn. Circuit simulators (i.e. Hspice, Powermill) would be slow to run and provide insufficient information. For this purpose, the authors have developed Pythia [XYC97], a fast and accurate power estimation tool that works with a structural Verilog description. Pythia uses a collection of Verilog-PLI access and utility routines to traverse circuit hierarchy, extract user-provided parameters and monitor signal transitions. Each net in the design is annotated with four different capacitance values:

1. A gate capacitance value indicating the total gate capacitance attached to the net.

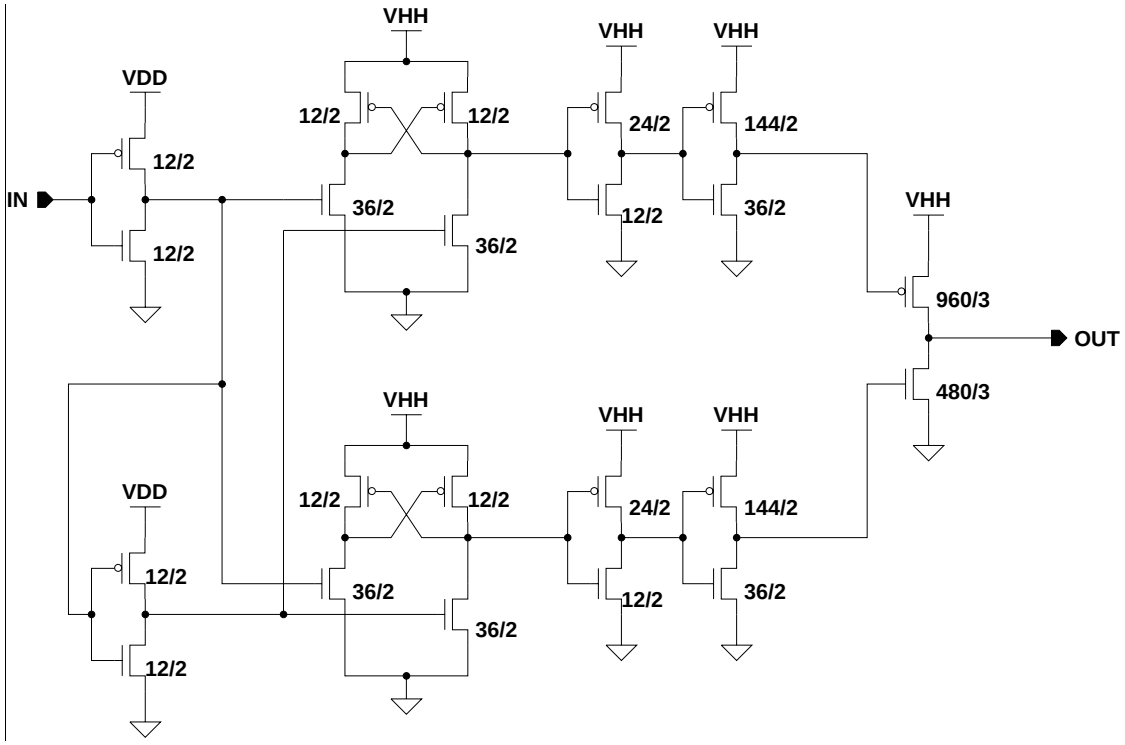


Figure 3.15: IDCT Chip Output Pad (sizes in $\lambda = 0.35\mu$).

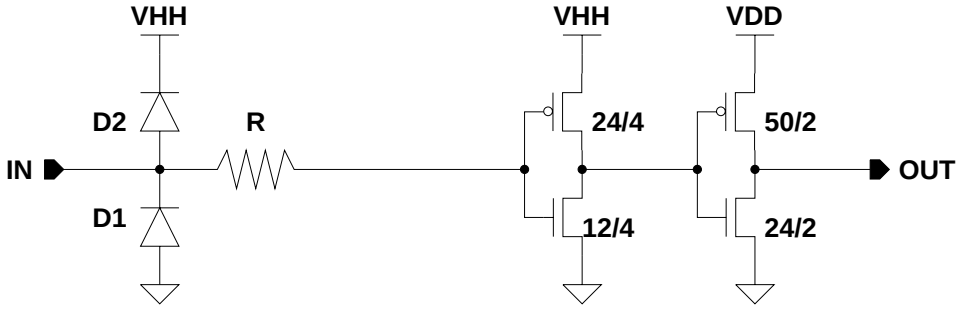


Figure 3.16: IDCT Chip Input Pad (sizes in $\lambda = 0.35\mu$).

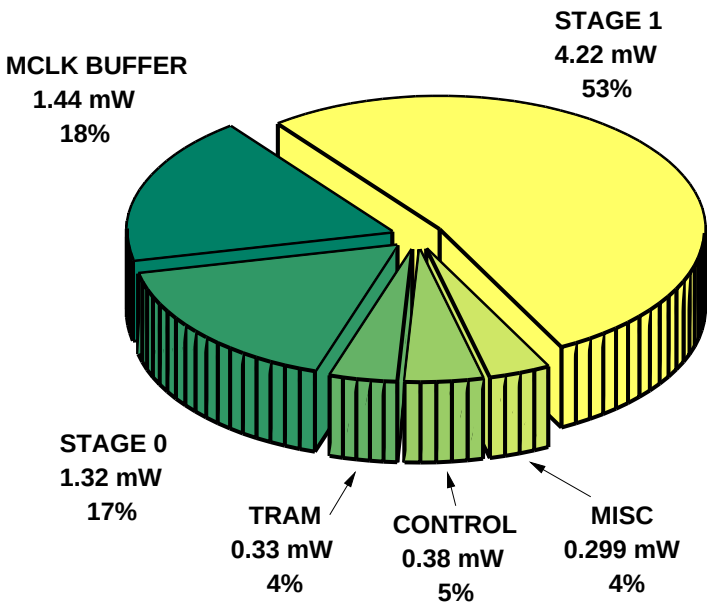


Figure 3.17: IDCT Chip Pythia Simulation Results

2. An NMOS drain capacitance value indicating the total n-drain to p-substrate or p-well junction capacitance attached to the net.

3. A PMOS drain capacitance value indicating the total p-drain to n-substrate or n-well junction capacitance attached to the net.

4. A routing capacitance indicating the total metal wiring capacitance of the net.

During each net transition, there is a different energy contribution from each particular capacitive component. Pythia performs numerical integration in order to account for the fact that gate capacitances are a function of voltage. Moreover, it uses the depletion approximation and a first order analytical solution for the CV integral in the case of junction capacitances. Process parameter information is provided in the form of spice model files. Finally, Pythia expands cell internal capacitive nodes that are collapsed in a Verilog gate-level simulation so that every circuit capacitive node is accounted for. Pythia is very fast (5-7 times slower than Verilog) and reasonably accurate (typically within 5% from Hspice and Powermill). Feedback is provided in the form of a report with energy and power information for each hierarchical block in the design.

Power library generation has been automated within the Cadence Design Framework. Each library cell requires two phases of annotation. During the first phase, cell terminal capacitances are added to the cell Verilog functional view. In the second phase, an internal model is generated which identifies the internal nodes (non-terminals) of each cell and describes the switching state of such nodes for each input state. In this way, all capacitive nodes are included in the power estimation.

Figure 3.17 shows the IDCT chip power estimation results partitioned among the toplevel modules of the design. The data was obtained by feeding the chip 1000 random blocks from the MPEG sequence Cheerleader (a more complete description of the test sequences used appears in section 3.7.) The simulated clock frequency and power supply are 14 MHz and 1.3V respectively. The software tool includes an elementary interconnect capacitance estimation mechanism based on first order process parameters and net fanout. Approximately 30% of the total chip power of 7.95 mW is attributed to interconnect capacitance.

Stage 0 and Stage 1 which are the main computation stages dissipate the bulk of the power (70%). The clock buffer of the master clock dissipates a substantial 18% of the total. This is the globally distributed clock that produces all derivative pipeline clocks. The final inverter of this exponential driver has a total gate width of 56 μm (W/L ratio of 93.3). Although the TRAM contains a large number of flip-flops, its duty cycle is low (clocked once every 8 cycles) and only requires 4% of the total power. Global control circuitry requires 5% of the power with the remaining 4% attributed to miscalleneous circuitry for block I/0 and glue logic between stages. We note that stage 1 requires a lot more power than stage 0. This happens because more non-zero coefficients are processed on average by stage 1 due to the nature of equation 3.1. Figure 3.18 plots the number of non-zero coefficients per block encountered at stage 1 of the IDCT chip vs. the number of non-zero coefficients encountered at stage 0.

Figure 3.19 shows the distribution of power within the stage 1 computational block. We observe that a substantial portion of the power (27%) is used for the clock generator (and buffer) for all pipeline stages. The rest is mainly computational power.

3.4.1 Glitch Power Investigation

Pythia can report the power due to spurious transitions (glitches). This is accomplished by keeping track of each signal pulse width and reporting separately pulses that have a width less that half the master clock period. Figure 3.20 shows the percentage of power due to glitches for a number of blocks in the design. The accumulators in the design are the most glitch-prone blocks because all three inputs (A, B, CIN) of each full adder become valid at different times. The multipliers exhibit a smaller percentage of spurious transitions because our gate model assumes equal delays between the sum and carry outputs of our full adders. As a result, two out of three inputs of each full adder in the Braun array (Figure 3.7) become valid simultaneously and glitching is reduced. We tried to equalize these two delays in the actual full adder circuit design (Figure 3.10) for this purpose. The total simulated chip power due to glitching is 1.16 mW (14.59%.)

3.4.2 Pipeline Power Savings Investigation

The IDCT chip employs a substantial level of pipelining for the sole purpose of lowering the power supply at real-time video sample rates; it is not required for algorithm functionality. We wanted to identify the total power savings attributed to pipelining by assessing

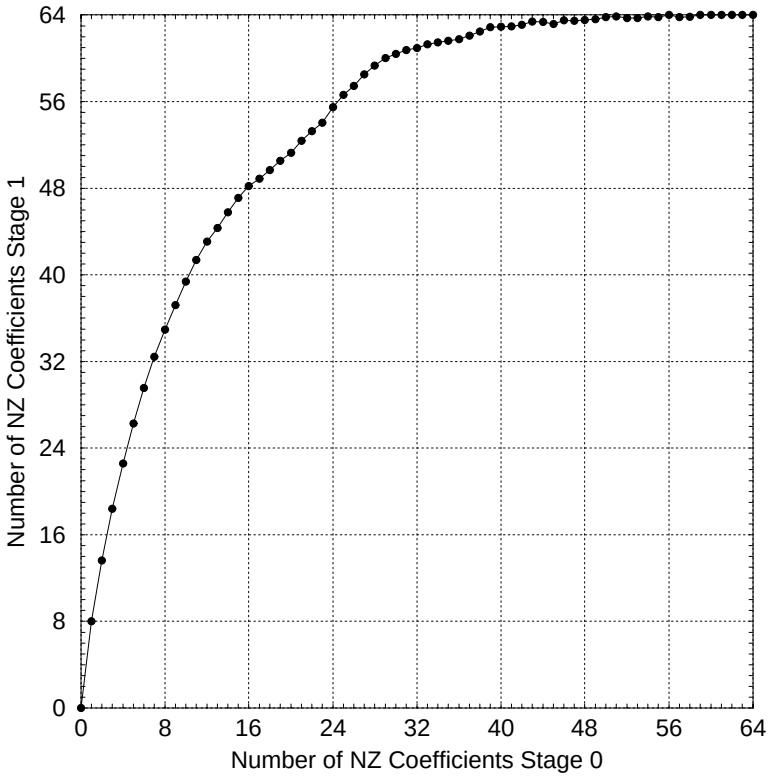


Figure 3.18: Stage 1 NZ Coefficients per Block vs. Corresponding Stage 0 NZ Coefficients for a Typical MPEG Sequence

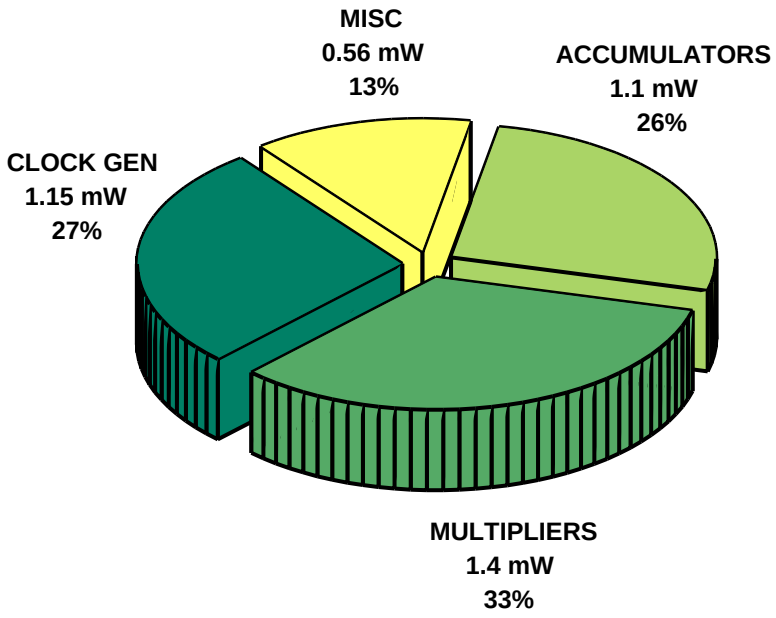


Figure 3.19: Stage 1 Pythia Simulation Results

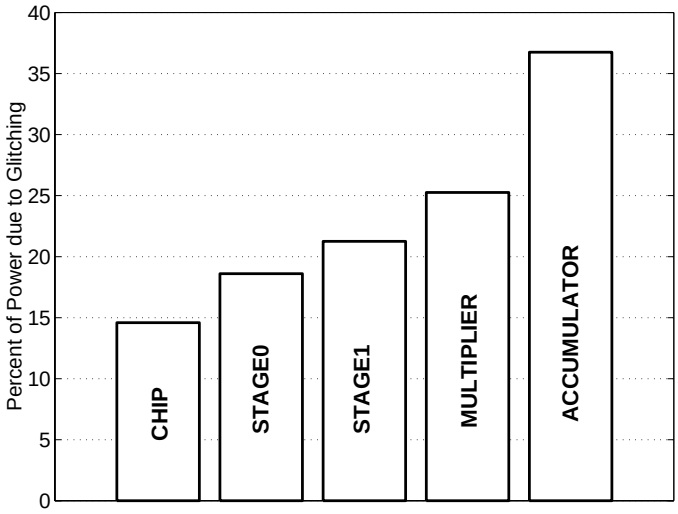


Figure 3.20: Glitch Power Estimation

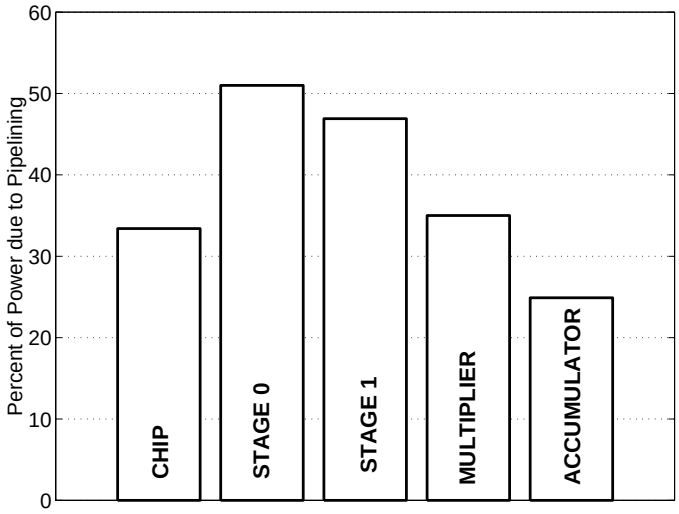


Figure 3.21: Pipeline Power Estimation

the overhead required and adjusting for a higher supply voltage to maintain the sample rate.

Figure 3.21 shows the percentage of power for major chip blocks which is attributed to pipelining. This power includes the two main pipeline clock generators (one for each stage) in addition to all the pipeline flip-flops in the signal path. The 33% percent as a total power cost of pipelining is exaggerated because a significant percentage of the pipeline flip-flops serve as the state registers of the accumulators which would still be present (along with their appropriate clock buffer) in the absence of pipelining. If we perform this adjustment, we estimate the actual cost of pipelining to about 20% of the total power.

In the absence of pipelining, the chip critical path grows by a factor of four. Testing the chip at various clock frequencies (section 3.7) indicated that the supply voltage should be raised to 2.2V in order to maintain a sample rate of 14 MS/sec. Subtracting the overhead and accounting for the increased supply, we estimate that the chip total average power would have been about 10 mW. Pipeline therefore accounts for more than 50% of power savings.

3.5 Chip I/O Specification and Usage

The user provides the chip with DCT spectral coefficients values on the rising edge of clock. The beginning of a new coefficient block is indicated with the assertion of the StartBlockIn pulse when coefficient (0,0) is present on the input bus. After a delay of 90 cycles, the pel values start to come out of the chip output pins. The start of a block is indicated by the assertion of the StartBlockOut pulse when output pel block element (0,0) is present on the output bus. Figure 3.22 presents a timing diagram that summarizes the chip operation. We

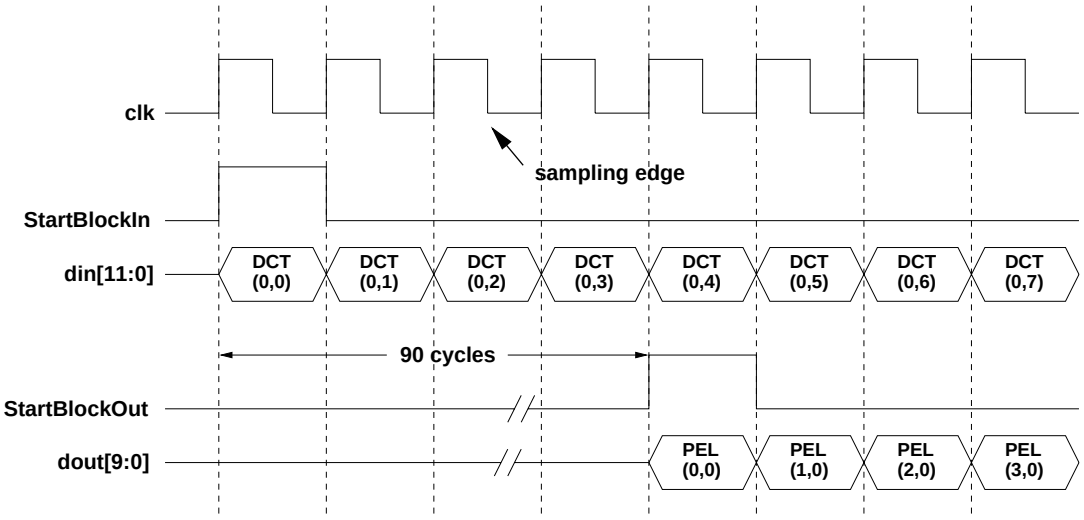


Figure 3.22: IDCT Chip Timing Diagram

note that the output block comes out in transposed form. Figure 5.41 in the corresponding section of the next chapter shows the sequence that the DCT coefficients must be presented to the chip and the sequence that the video pels are being produced by the chip. The negative clock edge is used for sampling the input data and the StartBlockIn strobe on the chip so that the edges are sufficiently spread apart and there is sufficient setup and hold time if the data edge occurs close to the positive clock edge (in either direction.)

3.6 Chip Physical Design and Packaging

The chip has been laid out using the Cadence Design Framework. Details about the design flow and methodology are presented in appendix A. It has been fabricated in the Hewlett-Packard AMOS14TB Triple-Metal Single-Poly 0.5 μm 3.3V process available through the MOSIS Design Service. The MOSIS scalable design rules have been used (SCMOS with $\lambda = 0.35\mu$m) and as a result the minimum drawn device length is 0.7 μm. The annotated chip microphotograph is shown in Figure 3.23. The core area of the DCT chip (not including the I/O pad frame) measures 20.7 mm^2 and includes approximately 160K transistors. The chip specifications are summarized in Table 3.3.

The chip package is a ceramic PGA (11x11 grid, 84-pin) available through MOSIS (PGA84M, 350mil cavity). The chip pinout appears in Table C.1 (Appendix C). The VHH supply is a TTL-compatible 5V supply for the pads and VDD is the core supply at 1.1-2.2V. VLL is the pad ground and may be shorted to GND on the board. The package PCB footprint (top view) is shown in Figure C.1 (Appendix C).

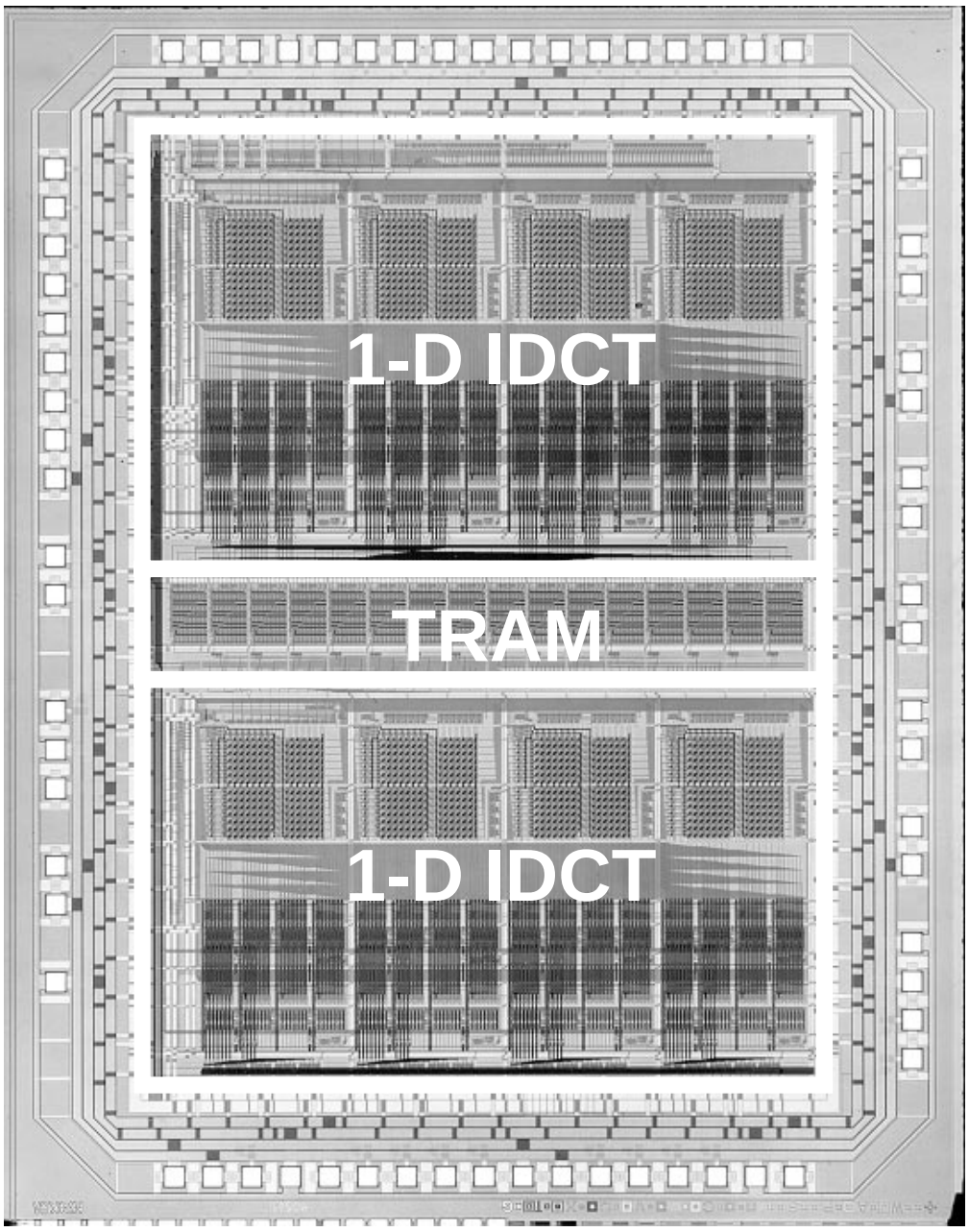


Figure 3.23: IDCT Chip Microphotograph

Process	0.5μm CMOS (0.7μm drawn), 3ML 3.3V
VTN	0.66V
VTP	-0.92V
TOX	9.6 nm
Supply	1.1-1.9 V
Frequency	5-43 MHz
Power	4.65 mWatts @ 1.32 V, 14 MHz
Area	20.7 mm^2
Transistors	160K

Table 3.3
Process and IDCT Chip Specifications

3.7 Testing

3.7.1 Test Setup

A printed circuit board has been built for testing and measurement. A dedicated PC interface through a standard ISA National Instruments 32-bit digital I/O card has been used for providing stimuli and reading results from the board under software control. The board contains SRAM buffers and control finite state machines to permit high-speed testing in spite of the slow PC ISA interface. Moreover, the IDCT chip core uses separate supply lines to permit accurate current measurements.

The board block diagram is shown in Figure 3.24. A photograph of the board is shown in Figure 3.25. The output of the DIO32F connector has been terminated with 470-Ohm resistors to ground and has been passed through two inverting CMOS Schmitt triggers (74HC14) for additional conditioning. The board operates as follows: The user uploads 256 64-element DCT blocks into the input dual port memory (Cypress CY7C006) and the input FSM (implemented in a Lattice GAL 22v10) is triggered. The input FSM produces appropriate control signals to read the blocks from the SRAM and stimulate the IDCT chip. The 256 blocks are cycled through the chip continuously so that power measurements can be obtained while the chip is being continuously stimulated. On the other hand, the output FSM captures the 256 output blocks in the output SRAM only once: The SRAM is activated only when the first batch of 256 pel blocks are being output from the chip. It is deactivated for the remainder of the chip operation. The user can upload the results on the PC from the second SRAM and can check the chip output for correctness.

In addition to the test structures described above, the board contains two more sections: One section contains an IDCT chip directly attached to the DIO32F connector so that the user has direct access to all chip terminals through the PC. The user has access to a mechanical multiplexer (set of 32 jumpers) to direct the single DIO32F connector to the fast or slow test section of the board. This section has been used for preliminary slow-speed functionality-only testing. The final section simply contains an IDCT chip with all relevant terminals brought out to vertical headers for last resort testing using pattern generators and logic analyzers. The IDCT test board has a single rather annoying bug: The

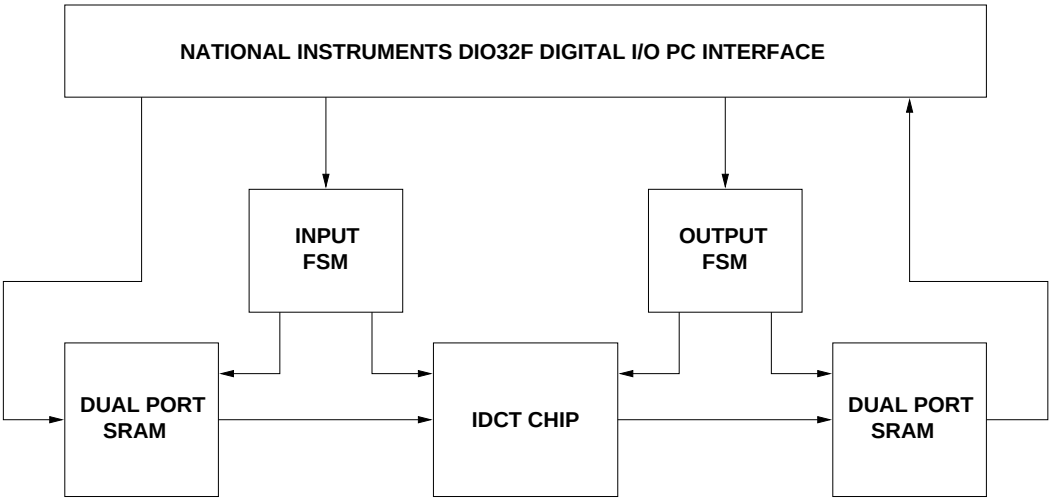


Figure 3.24: IDCT Chip Test Board Block Diagram

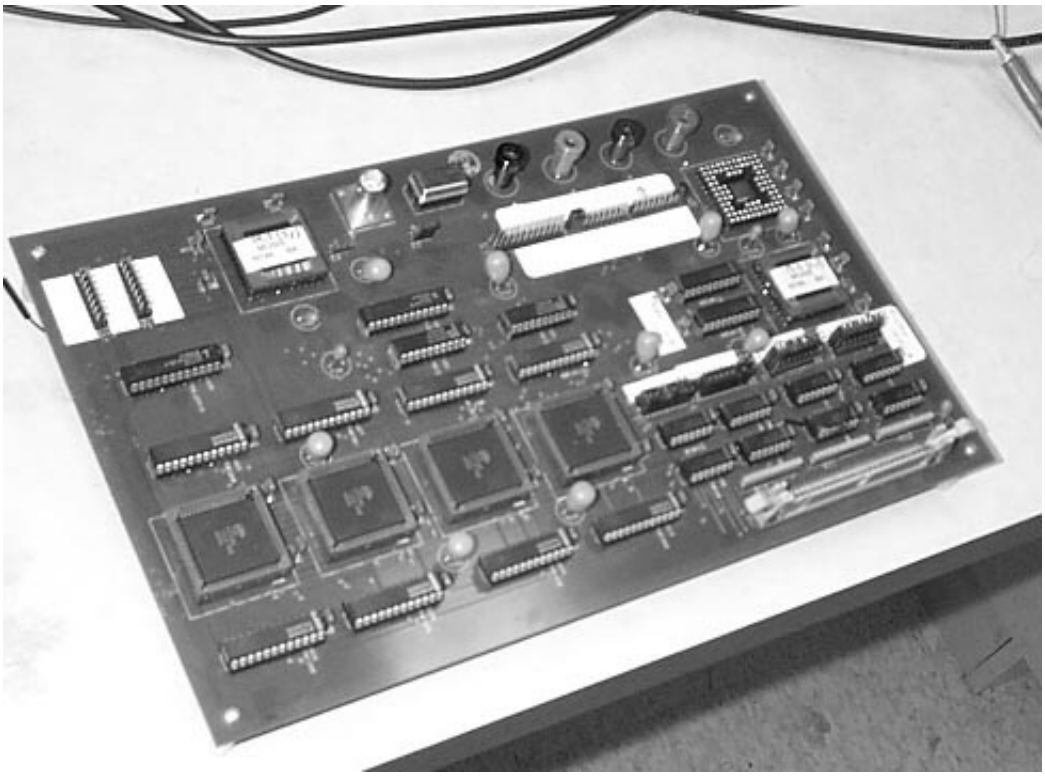


Figure 3.25: IDCT Chip Test Board Photograph

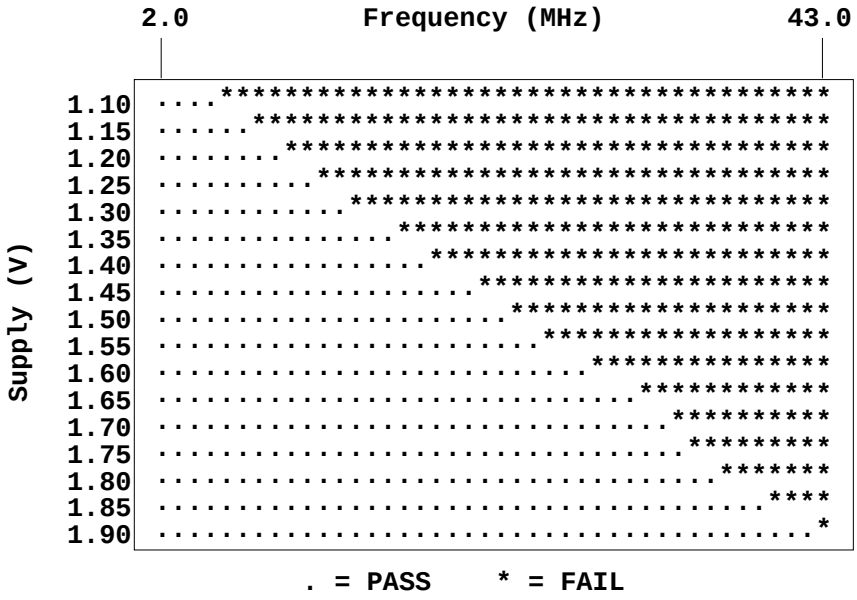


Figure 3.26: IDCT Chip Schmoo Plot

VLL signal was not shorted to GND as was the original intention but has accidentally been left floating. The problem has been fixed with the addition of several wires on the board solder side.

3.7.2 Test Results

The chip is functional over a wide range of frequencies and power supplies as indicated by the Schmoo plot of Figure 3.26. Operation at frequencies above 50 MHz at $<$2.5V has been observed using high-speed digital pattern generators.

Over 2.8 million separate power measurements have been taken during several weeks while the chip was stimulated from 6 different MPEG compressed sequences ranging from 1 MB/sec-5 MB/sec and having rather different content (Table 3.4). The entire chip pipeline was filled with data from a single 8×8 coefficient block repeated multiple times during each one of the measurements. The results of these measurements are plotted in Figure 3.27 vs. the number of non-zero coefficients within each DCT block. On the same graph the block non-zero content histogram is also shown. The average, as well as the 95% confidence interval is plotted for each data point. Figure 3.28 shows the average power dissipation for each one of the sequences of Table 3.4 along with the corresponding average number of non-zero coefficients per block. As expected, sequence FLOWER dissipates the most power because it contains global movement and a lot of high-frequency details. Figure 3.29 shows the detailed power dissipation profile for all 6 test sequences.

Figure 3.30 shows the average block power dissipation per MPEG macroblock type [MPFL97] and predictive picture type (I, P, B) for sequence SUSI. A brief description of I,

Sequence	Resolution	Bitrate	Content
BALLET	704x480	5M	Foreground movement panning, scene change
BICYCLE	704x480	5M	Background movement, zooming
CHEERLEADER	704x480	5M	Foreground movement
FLOWER	352x240	1.5M	Panning
SUSI	352x288	1.5M	Head and shoulders, little foreground movement
TIGER	352x240	1M	Foreground/Background movement, panning, scene change

Table 3.4
MPEG Sequences Used for Testing the IDCT Chip

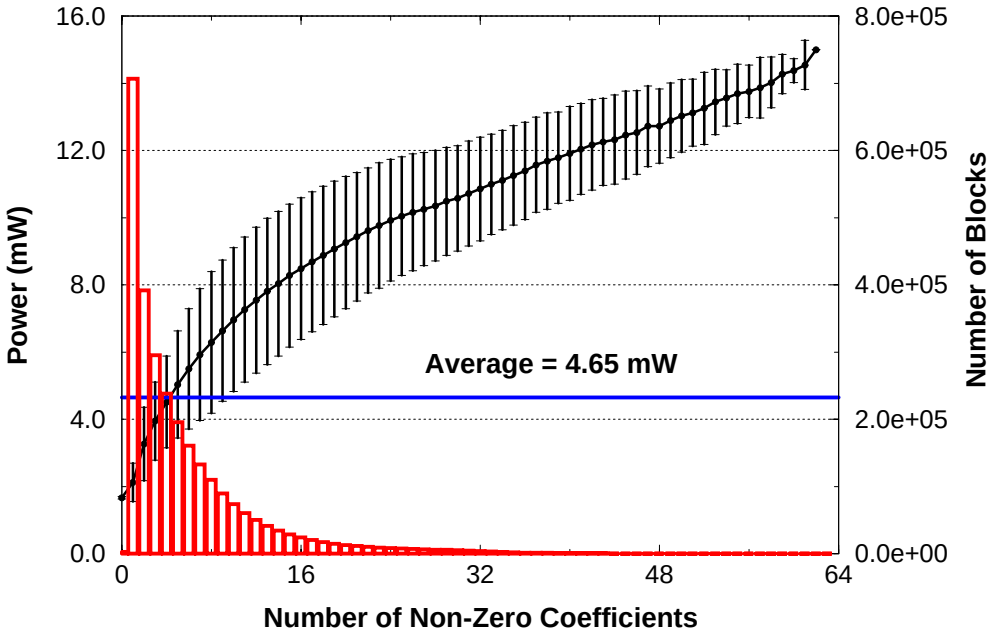


Figure 3.27: IDCT Chip Measured Power Results at 1.32V, 14 MHz. This plot represents over 2.8 million separate power measurements.

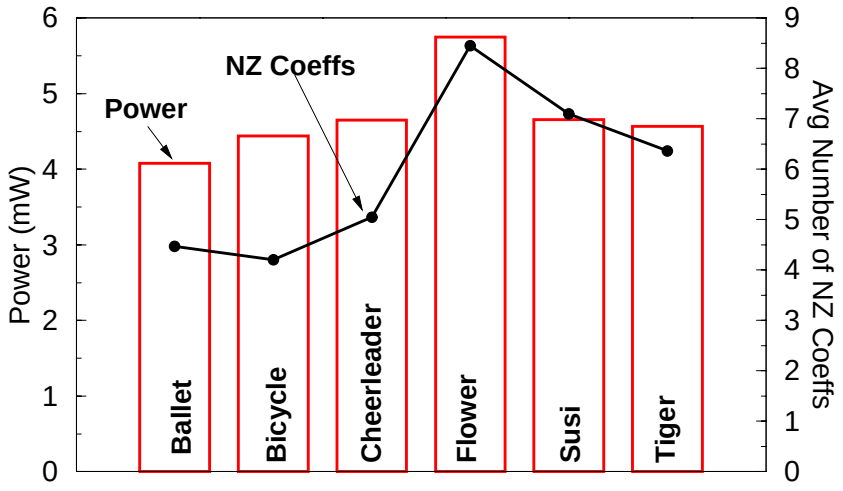


Figure 3.28: IDCT Chip Measured Power Results at 1.32V, 14 MHz. Average Power Dissipation per Sequence.

P, and B pictures in an MPEG sequence has been presented in section 2.5. We observe that intra-coded macroblocks dissipate the most power. This is expected since intra blocks have on average more non-zero coefficients due to lack of differential coding. We note that intra macroblocks may appear in all three types of pictures with the constraint that I pictures may only contain intra macroblocks. On the other hand, bidirectionally coded macroblocks dissipate the least power: Such macroblocks have the most freedom in predictor picture usage (both past and present) so that they are typically coded most efficiently and exhibit a large number of zero-valued coefficients per block.

Our power measurements have indicated that blocks with the same number of non-zero coefficients can exhibit a range of power dissipation. We have investigated this variation in power measurements by exercising the chip with blocks that contained a single non-zero coefficient exhaustively spanning the entire position-value space (2^{18} measurements.) Figure 3.31 shows the average power dissipation among all possible 12-bit values for each position within the 8×8 spectral block. We observe a maximum of 60% variation from position to position. This is mainly due to a different number of multiplications required for each spectral position within the block. The surface plot of Figure 3.32 shows power variation with respect to both block position and coefficient value. The two position axes of Figure 3.31 have been collapsed to a single 64-position axis. We observe a slight reduction in power with decreasing magnitude. This is due to the fact that all multipliers in the chip implement sign-magnitude arithmetic and exhibit reduced activity for small magnitude numbers. Along the zero-value axis there is an abrupt reduction in power because the arithmetic units are not active at all and power is only due to master clock distribution, transposition memory operation and global control.

Turning our focus back to Figure 3.27 we observe that power dissipation shows strong correlation vs. the non-zero coefficients because of the data-dependent processing algo-

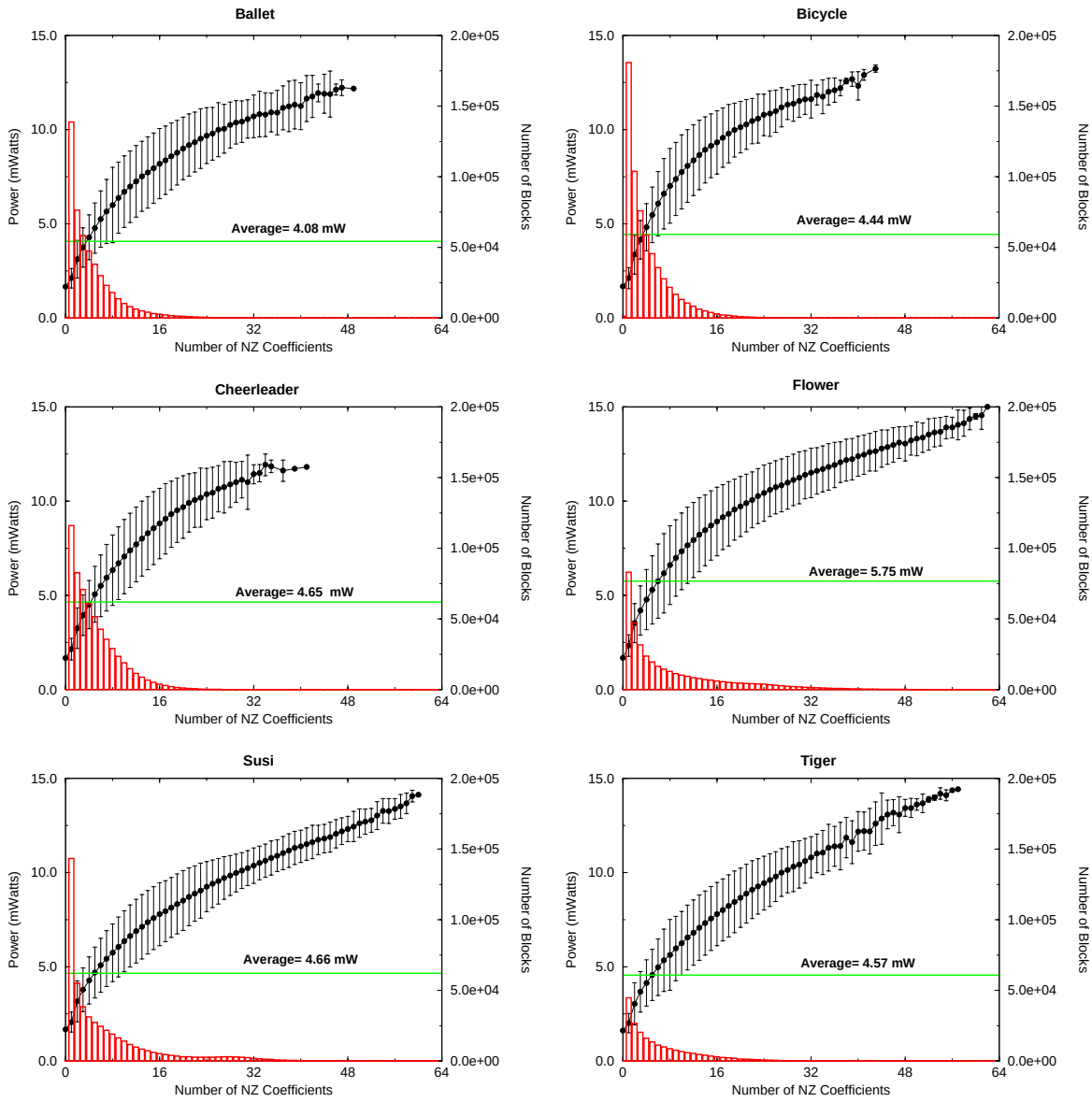


Figure 3.29: Power Dissipation Profile for 6 MPEG Sequences. IDCT Chip Measured Power Results at 1.32V, 14 MHz.

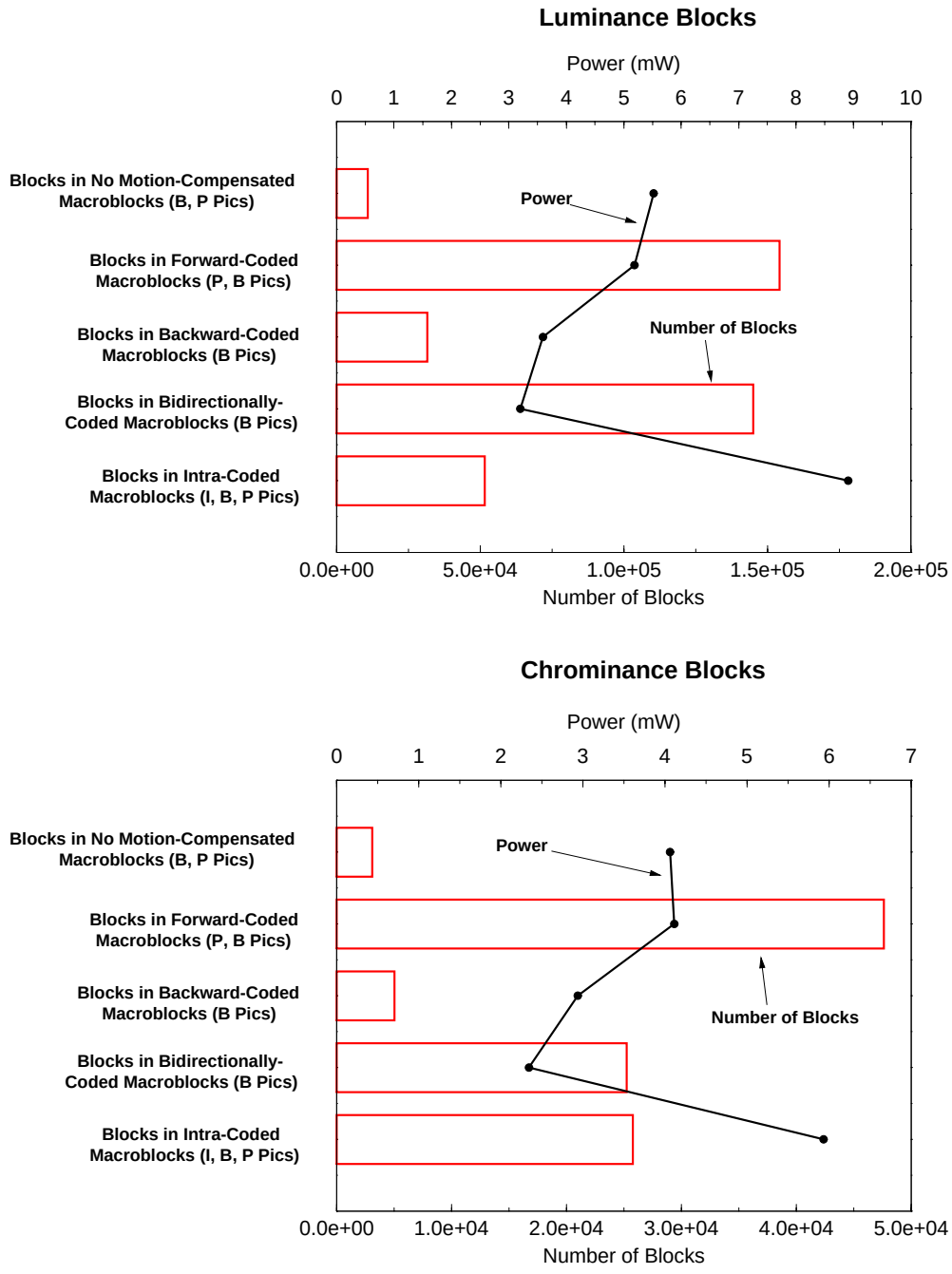


Figure 3.30: Average Block Power Dissipation per MPEG Macroblock Type (Sequence SUSI)

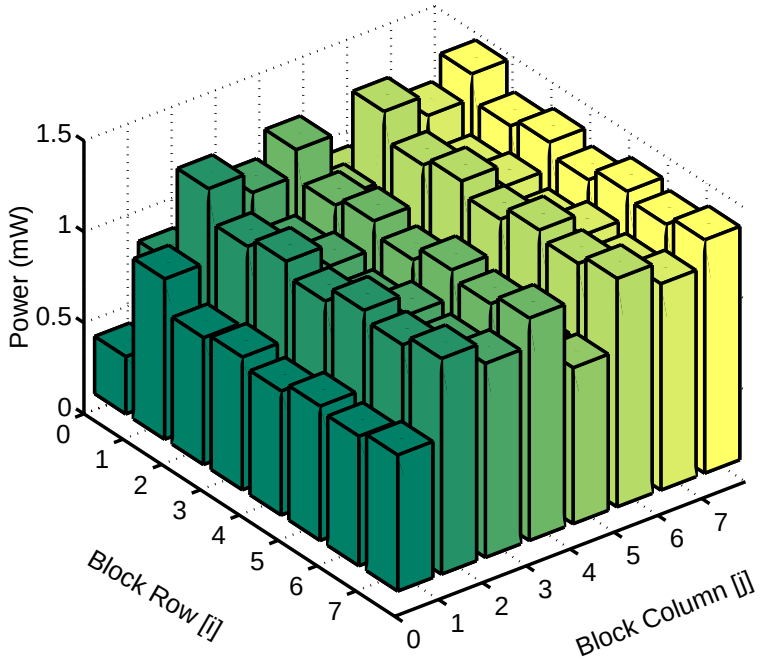


Figure 3.31: IDCT Chip Measured Power Results at 1.32V, 14 MHz: Average Power Dissipation per Block Position Across all Possible 12-bit DCT coefficient Values (1 NZ Coefficient per Block)

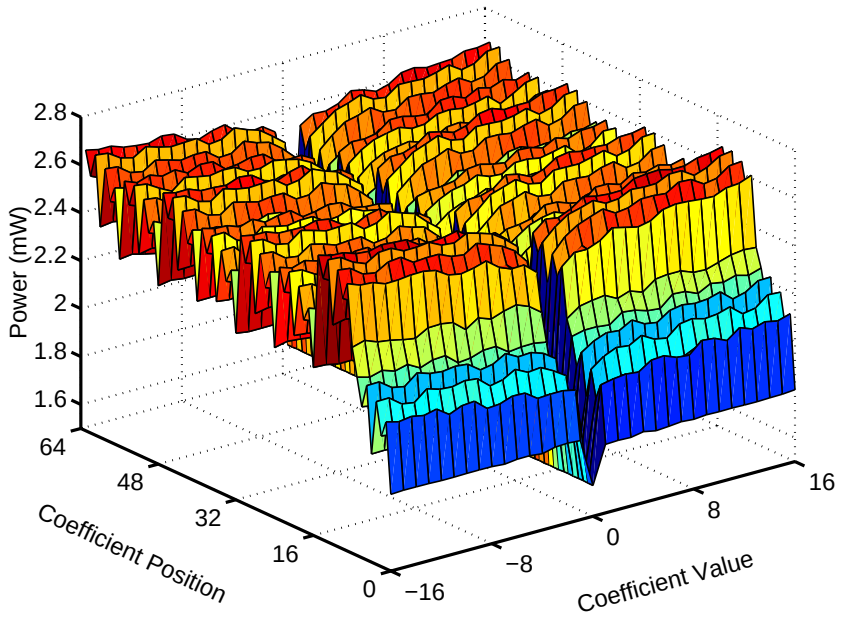


Figure 3.32: IDCT Chip Measured Power Results at 1.32V, 14 MHz: Blocks Containing 1 NZ Coefficient of Magnitude ≤ 16

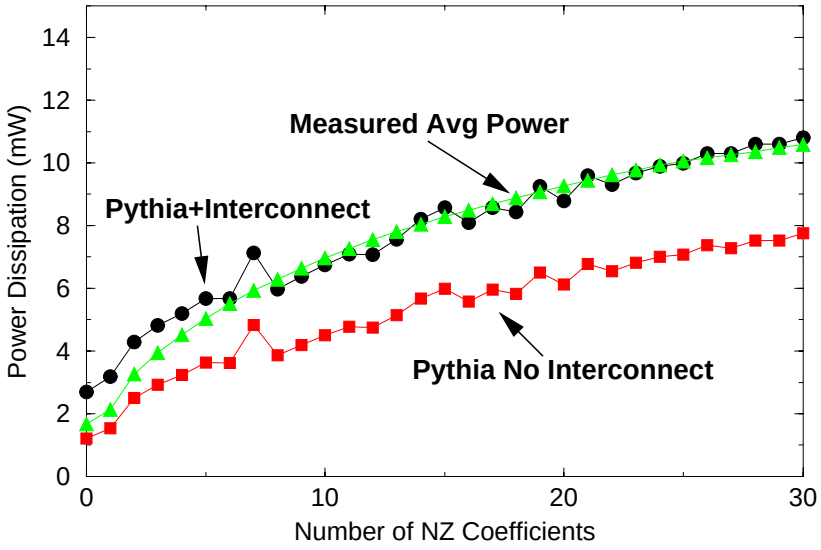


Figure 3.33: Comparison of Simulated and Experimental Power Results

rithm. The average power dissipation for this data set is 4.65 mWatts at 1.32 V, 14 MHz. The power savings exhibit diminishing returns as the number of non-zero coefficients increases. This happens because the first 1-D IDCT stage produces a lot more non-zeroes at its output as mentioned in section 3.4. As we move along the x-axis in Figure 3.27 most of the gains come from the stage 0 only; the gains from the second stage become less significant. At the average workload value though of 5.68 non-zero coefficients for this dataset, power savings from both stages are significant.

Figure 3.33 compares the results obtained from our Pythia simulator for a set of 300 random blocks from the Cheerleader sequence with the corresponding measured averages of Figure 3.27. Two sets of simulation results are shown: One set includes interconnect capacitance estimation and the second set was obtained with interconnect estimation off.

3.7.3 Comparison with Past DCT/ IDCT VLSI Implementations

There have been numerous VLSI DCT / IDCT implementations in the literature. Tables 3.5-3.7 summarize the specifications of recent high performance DCT / IDCT chips that use distributed arithmetic and have no significant architectural differences.

The 1996 Toshiba macrocell [KFN+96] (Table 3.7) has very impressive power dissipation characteristics (10 mW @ 0.9V, 150 MHz). There are two reasons for the low power dissipation: First, it is implemented in a much smaller feature size process (0.3 μm) than the other two chips. Second, it is implemented in a triple-well low VT process which permits aggressive voltage scaling to 0.9V. Moreover, the triple-well permits control over the threshold voltage value (through the source-to-bulk potential). Such control is used to raise the VT in idle periods and reduce standby leakage current. The authors maintain that

Area	21 mm^2, 0.8 μm 2ML process
Supply Voltage	5V
Transistors	102K
Clock Rate / Sample Rate	100 MHz / 100 Msamples / sec
Power Dissipation	N / A

Table 3.5
Mitsubishi Electric DCT / IDCT Processor [UIT+92]

Area	13.3 mm^2, 0.6 μm 2ML process
Supply Voltage	2V / 3.3V
Transistors	120K
Clock Rate / Sample Rate	200 MHz @ 3.3V, 100 MHz @ 2V
Power Dissipation	0.35W @ 3.3V, 200 MHz / 0.15W @ 2V, 100 MHz

Table 3.6
Toshiba 1994 DCT / IDCT Macrocell [MHS+94]

Area	4 mm^2, 0.3 μm 2ML process, triple well
Supply Voltage	0.9V (VT= 0.15V $\pm$0.1V)
Transistors	120K
Clock Rate / Sample Rate	150 MHz / 150 Msamples / sec
Power Dissipation	10 mW @ 0.9V, 150 MHz

Table 3.7
Toshiba 1996 DCT / IDCT Macrocell [KFN+96]

Area	7 mm^2, 0.5 μm process
Supply Voltage	3V
Transistors	69K
Clock Rate / Sample Rate	58 MHz / 58 Msamples/sec
Power Dissipation	250 mW @ 3V, 58 MHz

Table 3.8
AT&T Bell Labs 1993 IDCT Processor [BH95]

Chip	Sw-Cap/sample
Matsui et al. [MHS+94]	375 pF
Bhattacharya et al. [BH95]	479 pF
Kuroda et al. [KFN+96] (scaled to same feature size)	417 pF
Present Work	190 pF

Table 3.9
Energy Efficiency Comparison Among DCT/IDCT Chips

if the process VT was 0.5V, the chip supply would increase to 1.7V and the power dissipation to 40 mW in order to maintain performance at 150 MHz. This macrocell is far more interesting in terms of circuit design and implementation rather than architecture. In fact, examination of the microphotographs of chips [MHS+94] and [KFN+96] reveals that they are remarkably similar. Both of these chips will demonstrate similar switched-capacitance per sample metrics after process normalization.

The final reference that we will examine is an AT&T Bell Labs IDCT processor [BH95] that has a very different implementation from the previously described chips. Its specifications are listed in Table 3.8. It uses seven hardwired multipliers and exploits regularity in the IDCT matrix after some row/column reordering. This design has been used in an AT&T MPEG video decoder chip.

Table 3.9 shows the energy and the switched capacitance per sample for past IDCT chips summarized above and compares the results with the current work. The specification for [KFN+96] has been scaled up to account for a much smaller process feature size ($0.3\mu m$) and threshold voltage ($V_T = 0.1 \pm 0.15V$). The present work exhibits lower switched capacitance requirements per sample than previous implementations by a factor of 2. The switched-capacitance per sample metric takes into account only algorithmic and high level architectural design decisions that reduce the total activity; it factors out parameters such as clock frequency and supply voltage. It must be noted that the present chip employs a higher degree of pipelining than the other ones for the sole purpose of reducing the power supply and the switched capacitance metric penalizes such a design choice. Our algorithmic choice has therefore been justified given that all past chips employ standard algorithms performing a fixed number of operations per block.

Our final observation is that Figure 3.27 indicates that the power dissipation of the IDCT chip improves significantly at lower bitrates (coarser quantization). This makes our

approach ideal for emerging quality-on-demand compression protocols. At low rates, the present chip will exhibit a lot more efficiency vs. other implementations than Table 3.9 indicates.

3.8 Chapter Summary and Conclusion

We have demonstrated an ultra low power IDCT chip designed in a widely available $0.7\mu m$ high V_T process. The chip demonstrates the lowest switched capacitance per sample among past IDCT chips presented in the literature. Low power operation has been achieved through selection of a data-dependent algorithm, aggressive voltage scaling, deep pipelining, extensive clock-gating and appropriate transistor-level circuit techniques. The chip is fully functional and has undergone extensive testing. It dissipates 4.65 mW at 1.32V, 14 MHz.

Chapter 4

Adaptive Transform Coding

The input data of the DCT chip has very different statistical properties from the spectral coefficients that the IDCT chip operates on. In order to reduce the duty cycle of the DCT core processor we have resorted to original adaptation techniques based on ideas suggested by previous investigators.

Before proceeding with the description of the DCT core processor (chapter 5) we examine previous work in adaptive transform image and motion picture coding and survey various proposed image classifiers and adaptation techniques. A number of important ideas from this chapter have been embedded in the DCT processor design (i.e. classification and adaptive precision) for power minimization.

4.1 Early Adaptive Schemes [TW71] [Gim75]

As early as 1971, Tasto and Wintz [TW71] had proposed an image compression strategy based on an adaptive block quantizer. They classify each block into one of L categories and they apply a different Karhunen-Loeve Transform with different precision and quantization rules for each category. Their classification scheme defines three subimage categories:

1. Subpictures with much detail.
2. Subpictures with little detail and darker than average.
3. Subpictures with little detail and lighter than average.

This scheme is quite complicated because of the different transformations for each category. A much simpler scheme based on activity classes was proposed by Gimlett [Gim75] targetted to Fourier, Hadamard, Haar, Slant and Karhunen-Loeve transforms. Gimlett suggests the following activity index to serve as a measure of the “busyness” of a subpicture of N pels:

$$A = \sum_{i=1}^{N} a_i \mid F_i \mid \tag{4.1}$$

or

$$A = \sum_{i=1}^{N} a_i F_i^2 \tag{4.2}$$

The F_i are the transform coefficients and the a_i are weighting factors (i.e. unity, or inversely proportional to the variances of the F_i.) Gimlett refrains from suggesting a specific coding scheme, but uses as an example a classification on the 25, 50 and 75 percent ordinates of the cumulative distribution of A and subsequent use of adaptive quantization and zonal sampling.

4.2 Chen and Smith Adaptive Coder [CS77]

Chen and Smith [CS77] have reinvented Gimlett's results [Gim75] and presented a complete methodology for designing an adaptive image coder based on a simple statistical model of cosine transform samples. Although this work uses exactly the same classifier as [Gim75] (eq. 4.2 with $a_i = 1$), it enjoys seminal status in the adaptive image coding literature and has been widely referenced.

Chen and Smith assume that each image pixel $f(j,k)$ is a sample of a random process with mean m. Thus, the mean and variance of the transform coefficients $F(u,v)$ can be computed as follows:

$$E[F(u,v)] = \frac{4mc(u)c(v)}{N^2} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} cos\frac{(2j+1)u\pi}{2N} cos\frac{(2k+1)v\pi}{2N} \tag{4.3}$$

$$c(u) = 1/\sqrt{2} \text{ if } u = 0 \text{ and 1 otherwise} \tag{4.4}$$

$$E[F(0,0)] = 2m \tag{4.5}$$

$$E[F(u,v)] = 0, \quad (u,v) \neq (0,0) \tag{4.6}$$

$$\sigma^2_{F(0,0)} = E[F^2(0,0)] - 4m^2 \tag{4.7}$$

$$\sigma^2_{F(u,v)} = E[F^2(u,v)], \quad (u,v) \neq (0,0) \tag{4.8}$$

The authors justify an invocation of the central limit theorem because each DCT coefficient is a weighted sum of all the pixels in the original image. They conclude that the probability density functions for the DC and AC coefficients separately can be modelled by Gaussian PDFs:

$$p_{F(0,0)}(x) = \frac{1}{\sqrt{2\pi}\sigma_{F(0,0)}} exp[-\frac{(x-2m)^2}{2\sigma^2_{F(0,0)}}] \tag{4.9}$$

$$p_{F(u,v)}(x) = \frac{1}{\sqrt{2\pi}\sigma_{F(u,v)}} exp[-\frac{x^2}{2\sigma^2_{F(u,v)}}], \quad (u,v) \neq (0,0) \tag{4.10}$$

Given such a statistical model for the DCT coefficients, adaptivity lies in the assignment of more bits to the higher variance subblocks and fewer bits to the lower variance

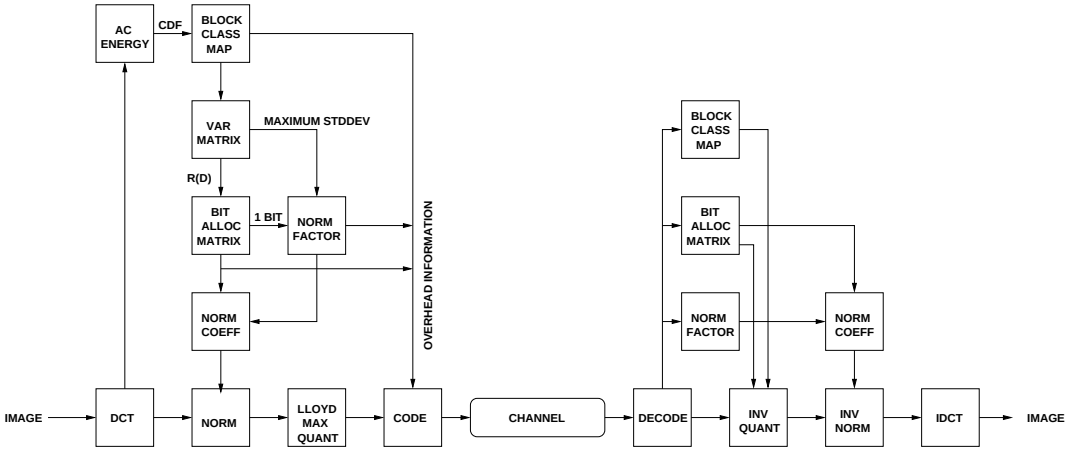


Figure 4.1: Chen-Smith Transform Adaptive Coding System [CS77]

subblocks. This amounts to adaptive quantization (or normalization according to the authors' terminology.) Eq. 4.8 links the AC coefficient variance to a sum-of-squares of AC DCT coefficients. Thus the classifier is defined as the AC energy in the (m,l)th subblock (assuming 16×16 subblocks):

$$E_{m,l} = \sum_{u=0}^{15} \sum_{v=0}^{15} F_{m,l}^2(u,v) - F_{m,l}^2(0,0) \tag{4.11}$$

Note that eq. 4.11 is equivalent to the Gimlett classifier (eq. 4.2) with $a_i = 1$.

The real-time adaptive image coder works in two passes: During the first pass the AC energy (eq. 4.11) of each subblock is computed and a cumulative distribution of the classifier is constructed. Next, each transform block is classified in one of four equally populated groups. The collective variance of the AC samples in each class is computed ($\sigma^2_{k,F(u,v)}, k \in \{0,1,2,3\}, (u,v) \neq (0,0)$) and the bit assignment $N_{B_k}(u,v)$ is determined using a well known relation from Rate Distortion Theory [Dav72] [CT91] [NH95]:

$$N_{B_k}(u,v) = \frac{1}{2} log_2 \frac{\sigma^2_{k,F(u,v)}}{D}, \quad k \in \{0,1,2,3\}, \quad (u,v) \neq (0,0) \tag{4.12}$$

Equation 4.12 is the rate-distortion function $R(D)$ for a Gaussian source given a fixed distortion D which in this particular case is the mean-square error between source data and binary representation. This function computes the minimum bit assignment $N_{B_k}(u,v)$ for a fixed MSE distortion criterion and is used to determine bit allocation matrices for each subblock class. Distortion D is initialized and the sum of all bit assignments $N_{B_k}(u,v)$ is iteratively computed until the desired bitrate for the image is achieved. DC coefficients are assigned to 8 bits in each one of the four classes. The final step of the first pass is to compute a common normalization factor to be applied before quantization. The authors use the maximum standard deviation of those elements that were assigned to one bit using eq. 4.12.

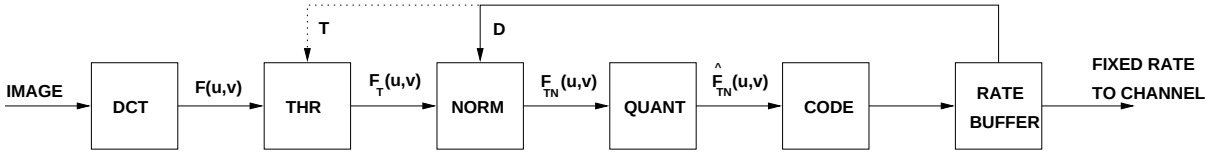


Figure 4.2: Chen-Pratt Scene Adaptive Coder [CP84]

During the second pass, the coefficient samples are multiplied by the normalization factor and are optimally quantized using Max's scheme (section 2.3.1). Then the data is entropy-coded and overhead information is added to the bitstream. A block diagram of the Chen-Smith coder is shown in Figure 4.1. The authors report very good results at 1 bit / pixel compression ration for two test images (PSNRs of 33 and 36 dB).

4.3 Scene Adaptive Coder [CP84]

In 1984, Chen and Pratt [CP84] attempted to remove the complexity of the previous two-pass scheme and propose a very simple single-pass coder that still maintains a level of adaptivity. The Scene Adaptive Coder is shown in Figure 4.2.

First, the input image undergoes a 16×16 two-dimensional DCT. The transform coefficients $F(u,v)$ undergo a thresholding process except for the DC $F(0,0)$ as follows:

$$F_T(u,v) = \begin{cases} F(u,v) - T & \text{if } F(u,v) > T \\ 0 & \text{if } F(u,v) \leq T \end{cases} \tag{4.13}$$

The authors demonstrate two test pictures with 90% of their DCT coefficient magnitudes below $T = 3$. The thresholding process (eq. 4.13) sets a major portion of the coefficients to zero and limits the portion of the coefficients to be quantized and transmitted. The threshold parameter T may vary with the global desired bitrate or it can be varied locally on a subblock basis.

The processed coefficients $F_T(u,v)$ are then scaled by a feedback normalization factor D that reflects the state of the rate buffer:

$$F_{TN}(u,v) = \frac{F_T(u,v)}{D} \tag{4.14}$$

Quantization is simply a floating-point to integer conversion. Essentially the entire normalization / quantization process can be thought of as uniform quantization with a variable stepsize D. No decision and reconstruction levels are required:

$$\hat{F}_{TN}(u,v) = \lfloor F_{TN}(u,v) + 0.5 \rfloor \tag{4.15}$$

The quantization process (4.15) will transform all the coefficients of fractional value to zero and leave only a limited number of significant coefficients to be Huffman-coded and

transmitted. The authors report PSNRs of 34.45 dB and 32.85 dB for *lena* and *plane* at 0.4 bits per pel.

The Scene Adaptive Coder can be thought of as similar to [CS77] with slightly different normalization / quantization techniques and a time-shifted and averaged classifier: The state of the rate buffer can be thought of as a measure of the AC coefficient energy in the previously coded blocks. As a result the Scene Adaptive Coder must exhibit lower PSNRs at the same bitrate given that the classifier does not depend on the currently coded block. No such comparison is performed by the authors. The main advantage of this scheme is that only one pass through the image is necessary.

4.4 Subjectively Adapted Image Coding for Progressive Transmission [Loh84]

Lohscheller [Loh84] has suggested an adaptive image communication system using a classification algorithm adapted to the visual threshold performance of the human eye. This communication scheme is not applied to image compression but rather to progressive image transmission. The target of this research is to reduce average image transmission rates by incorporating human interaction for selection of specific areas for buildup or rejection of additional image sharpening.

Under this scheme, the image receiver is provided with a recognizable image using as few bits as possible. The image buildup from stage to stage is realized without transmission of redundant information and the final image quality has no visual artifacts compared to the original. Such goals are achieved through class-oriented determination of the number and sequence of the spectral coefficients to be transmitted.

Lohscheller defines the visibility threshold γ_{uv} of a spectral coefficient $F(u,v)$ as the coefficient amplitude that results (after inverse transformation) in a just-visible stimulus for the human eye. Comprehensive results of extensive psychovisual experiments are cited to justify a certain set of minimum values for γ_{uv}. Then the block classifier is based on the frequency distribution for the threshold exceedings of spectral coefficients. Lohscheller's classification scheme has no influence on the quantization of the DCT coefficients (no adaptive quantization is performed as in [CS77] [CP84]). It only affects DCT coefficient transmission control, i.e. the number and transmission sequence of the coefficients. More specifically, the number and identity of spectral coefficients transmitted is determined using visibility threshold exceeding frequency distributions. On the other hand, transmission sequence is determined using coefficient variance information. An example 8-way classification scheme computed for a set of 17 images is shown in Table 4.1.

To quantify the superiority of this algorithm, the author plots MSE vs. image transmission time for both visually adapted progressive DCT and "standard" DCT. As expected, the MSE rolls off much more quickly in the progressive DCT. This comparison is of little value since there are no means for rate control of the "standard" DCT. Lohscheller essentially compares a progressively transmitted image with an incomplete non-progressive

Class	Coeff. Number	Transmission Sequence	Transmission Bits
K1	1	1,1	10
K2	2	1,1 / 1,2	17
K3	3	1,1 / 1,2 / 2,1	23
K4	4	1,1 / 1,2 / 2,1 / 1,3	28
K5	6	1,1 / 1,2 / 2,1 / 2,2 1,3 / 3,1	36
K6	8	1,1 / 2,1 / 1,2 / 2,2 1,3 / 2,3 / 3,2 / 3,1	43
K7	12	1,1 / 2,1 / 2,2 / 3,1 3,2 / 1,2 / 1,3 / 4,1 2,3 / 3,3 / 1,4 / 1,5	56
K8	16	1,1 / 1,2 / 2,1 / 1,3 2,2 / 3,1 / 1,4 / 2,3 3,2 / 4,1 / 1,5 / 2,4 3,3 / 4,2 / 3,4 / 4,3	67

Table 4.1
Example Classification Using Lohscheller's Visual Adaptive Scheme [Loh84]

image. A comparison of the visually progressive vs. some other progressive DCT (i.e. AC coefficient magnitude prioritization [HDG92]) would have been much more effective.

4.5 Adaptive DCT in the Perceptual Domain [NLS89]

Ngan, Leong and Singh [NLS89]have proposed an adaptive cosine transform coding scheme which incorporates human visual system (HVS) properties and employs adaptive quantization and block distortion equalization.

The authors exploit two aspects of the HVS in the design of the image coder. The first, is the spatial bandpass characteristic of human vision. A generalized model of the HVS can be represented by a function

$$H(\omega) = (a + b\omega)e^{-c\omega} \tag{4.16}$$

where ω is the radial spatial frequency in cycles per degree of angle subtended by the viewer, and a, b and c are modelling parameters, determining the shape of the HVS radial frequency response.

Nill [Nil85] has posed the problem of the frequency domain physical significance of the DCT and the intuitive meaning of its convolution-multiplication property. The use of the DCT in image coding assumes an even extension if the original scene (section 2.1). This causes loss of physical significance of the image representation in the DCT domain because such altered scene is not viewable to the human observer. To overcome this problem, Nill introduces a correction function $A(\omega)$ [Nil85]. The modified HVS function now becomes:

$$\hat{H}(\omega) = \mid A(\omega) \mid H(\omega) \tag{4.17}$$

A two-dimensional circularly symmetric version the HVS can be defined as:

$$\begin{aligned} \tilde{H}(u,v) &= \hat{H}(\omega) \\ \omega &= \sqrt{u^2+v^2} \end{aligned} \tag{4.18}$$

It is $\tilde{H}(u,v)$ that is used to weigh the DCT coefficients in the perceptual domain:

$$\hat{F}(u,v) = \tilde{H}(u,v)F(u,v) \tag{4.19}$$

The second aspect of the HVS exploited is visual spatial masking. This effect causes noise to be less perceptible in regions of high activity than in regions of low activity. The authors estimate the activity $C(m,n)$ of the (m,n)th block as the sum of the AC coefficients:

$$C(m,n) = \frac{1}{255}\left[\left(\sum_{u=0}^{15}\sum_{v=0}^{15}\hat{F}_{m,n}(u,v)^2\right) - \hat{F}_{m,n}(0,0)^2\right]^{1/2} \tag{4.20}$$

where $\hat{F}_{m,n}(0,0)$ is the DC coefficient and $\hat{F}_{m,n}(u,v)$ are the AC coefficients of the (m,n)th block respectively. The activity $C(m,n)$ is used to control an adaptive quantizer. Although $C(m,n)$ is very similar to previous classifiers [Gim75] [CS77], the authors use it in the opposite fashion than previous investigators. Blocks that exhibit a high activity index are more coarsely quantized than blocks that exhibit a low activity. The authors attribute this implementation decision to the effect of visual spatial masking.

Results are presented for LENA and PEPPERS. These images exhibit a 33.5 and 32.5 dB at 0.4 bpp. No details are given as to how the SNR is measured (before or after weighting in the perceptual domain.) Moreover, it is expected that the classification method employed by the authors (coarse quantization in blocks of high activity as opposed to blocks of low activity) will actually decrease the SNR value as opposed to a traditional scheme. No treatment of this issue is offered. The authors' conclusion is that given a certain compression ratio, their visually weighted scheme produces subjectively superior results.

4.6 Adaptive DCT Based on Coefficient Power Distribution Classification [KMO87]

Kato, Mukawa and Okubo [KMO87] have proposed an adaptive DCT encoding method based on coefficient power distribution (CPD) borrowing ideas form Vector Quantization image compression [NH95]. Their particular classifier attempts to capture two major characteristics of the DCT-coded subblocks:

- Positions of significant coefficients in the subblock.
- Power of significant coefficients.

Previous investigators up to this point have only been concerned with a single attribute (power of AC coefficients.) The authors claim that a multi-dimensional classification can produce increased coding efficiency.

First, the power distribution for the input DCT subblock is computed:

$$S(u,v) = F^2(u,v) \tag{4.21}$$

where $F(u,v)$ denotes the (u,v) element of the input DCT subblock. $S(u,v)$ represents the coefficient power distribution (CPD). A certain number of predetermined CPDs is at hand (S_i) and each subblock is assigned a category index i based on the closest match between S and S_i for all i. The matching criterion J_i is defined as follows:

$$J_i = \sum_u \sum_v [g\{S(u,v)\} - g\{S_i(u,v)\}]^2 \tag{4.22}$$

$$g(z) = \begin{cases} \frac{1}{2}\log_2 z + a & z > 1 \\ 0 & z \leq 1 \end{cases} \tag{4.23}$$

where $S_i(u,v)$ represents the ith predetermined CPD. Function $g(z)$ has been determined experimentally and is introduced on the grounds that the classification must be based on the entropy of signals with variance z. The predetermined CPDs are designed from a test sequence in a method similar to the LBG vector quantization codebook design algorithm [LBG80]. The adaptivity is provided by adaptive assignment of variable length codes (VLCs) based on the classification index i. The authors refrain from using adaptive quantization and bit allocation because a larger-value quantization error is introduced when the statistical characteristics of DCT coefficients are different from the coefficients used when the bit allocation matrices were initially calculated.

The authors present simulation results comparing a 65-class motion picture encoding system with a system based on the Scene Adaptive Coder [CP84]. Their results indicate a 0.5-1.0 dB higher SNR than the Scene Adaptive Coder for the same bitrate. No information is provided about the training set used for calculation of the 65 predetermined CPDs.

The main disadvantage of this adaptive scheme is that the determination of the classification index is a very computationally intensive operation and that the quality of the compressed stream is highly dependent on the training set and the predetermined CPDs.

4.7 Spectral Entropy-Activity Classification [MF92]

Mester and Franke [MF92] introduce a two-dimensional feature space for classification of image transform subblocks. The first feature is the block activity as suggested by Gimlett:

$$A = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \mid F(u,v) \mid, \quad (u,v) \neq (0,0) \tag{4.24}$$

Activity A is a measure of contrast and detail within a block. Yet, the authors maintain that adaptation of an image transform coding system on A alone is not sufficient. Although

in most transform block spectra only a small number of AC coefficients contain most of the total AC energy and have relatively large magnitudes compared to the rest, this effect is only guaranteed *on average*. There may be blocks where the AC energy is partitioned among a large number of coefficients. Adaptive coding schemes based on activity alone will produce significant distortions for such blocks that exhibit non-average behavior. The distribution of the AC energy must also be considered by measuring the actual degree of spectral energy compaction. For this purpose, the authors introduce the *spectral entropy*, a quantity equivalent to the entropy of a discrete random variable [CT91] which measures the degree to which its probability mass function is equally distributed among the experimental values:

$$A = -\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} a(u,v)\ln(a(u,v)), \quad (u,v) \neq (0,0) \tag{4.25}$$

$$a(u,v) = \frac{|F(u,v)|}{A} \tag{4.26}$$

where A has been defined in eq. 4.24. Mester and Franke maintain that the spectral entropy E is an excellent measure of energy compaction achieved by the transform. Low entropy values occur if the energy is distributed among only a few coefficients and vice versa.

The authors have designed a 100-rectangular region 2D feature classification space for 10 separate values of A and E. The adaptive coder variable is threshold control equivalent to [CP84]. Through subjective measurements, a threshold function $t(A,E)$ has been determined. Threshold values grow with increasing activity A (visual masking in the presence of high frequencies [NLS89]) and declining spectral entropy E.

In order to study the performance of this scheme, the authors coded (with no quantization) a set of 17 images using adaptive thresholding. Resulting images were reported to be of very high quality when observed under standard viewing conditions. In this set of images, the number of supra-threshold DCT coefficients varied between 2 and 7.5 with an average of four. No quantitative image quality results (in terms of PSNR or MSE) are reported.

4.8 Other Adaptive Methods

Similar treatment of high-magnitude DCT coefficients appears in [HDG92]. The authors have proposed a prioritized DCT for compression and progressive transmission of images (PDCT). Images are DCT coded and the resulting coefficients are partially prioritized in a decreasing magnitude order. The number of passes as well as the magnitude partition ranges are parametrised. Coded position information is also embedded with minimal overhead. The authors show that this scheme results in steep PSNR vs. bitrate curves which confirms the large perceptual significance of high-magnitude DCT coefficients.

WORK	CLASSIFIER	ADAPTATION TECHNIQUE
[Gim75]	AC Energy	Adaptive Quantization
[CS77]	AC Energy	Adaptive Quantization
[CP84]	Rate Buffer Status [AC Energy of Previous Blocks]	Adaptive Threshold Control Adaptive Quantization
[Loh84]	Frequency Distribution of Spectral Threshold Exceedings	Coefficient Transmission Control
[NLS89]	Visually Weighted Sum of AC Coefficients & Rate Buffer Status	Adaptive Quantization
[KMO87]	Coefficient Power Distribution	VLC Assignment
[MF92]	AC Energy & Spectral Entropy	Adaptive Threshold Control

Table 4.2
DCT-based Adaptive Coders

4.9 Chapter Summary and Conclusion

Table 4.2 summarizes the adaptive schemes presented so far in terms of the classifier and the adaptation technique used. References [Gim75][CS77][NLS89] and [MF92] agree that the AC Energy captures an important characteristic of the DCT subblock and use it as a classifier to modify compression parameters. Interestingly, previous investigators use the AC Energy in a very different fashion. [Gim75][CS77] allocate more bits to high-energy blocks and less bits to low-energy blocks in order to minimize mean square error. On the other hand, [NLS89] and [MF92] do exactly the opposite. They are not interested in minimizing MSE because it is not always a good indicator of compressed image quality. They take into account human visual system properties, one of which is visual masking. Visual masking results in much higher error tolerance in regions of high activity. For this reason, less bits are allocated for high-energy subblocks. The other feature used to characterize image subblocks is information about the position of the high-magnitude coefficients within the block. References [Loh84] [KMO87] and [MF92] have proposed different classifiers to capture the energy distribution within a DCT subblock.

The most popular adaption technique is adaptive quantization employed by [Gim75] [CS77] [CP84] and [NLS89]. On the other hand, references [KMO87] and [MF92] use adaptive VLC assignment and adaptive threshold control respectively. One disadvantage of adaptive quantization is that it can produce large quantization errors for subblocks that have substantially different statistical characteristics from the training set used to derive the bit allocation matrices. Advantages include simplicity, low overhead and probably better compression ratio.

We observe that all classifiers examined use spectral domain information. Moreover, all adaptation techniques operate on post-DCT computation data. For a power-conscious image coding system it would be highly desirable to use classification information in order to save on energy-consuming arithmetic operations during the transformation procedure itself. One of the problems addressed in this thesis is the selection of an appropriate

pel-domain classifier and adaptation technique to reduce arithmetic operations during the actual DCT computation. This is the subject of the next chapter.

Chapter 5

DCT Core Processor

This chapter describes the design, implementation and testing of a low power DCT core processor targetted to low power video (MPEG2 MP@ML) and still image (JPEG [Wal91]) applications. The chip exhibits two innovative techniques for arithmetic operation reduction in the DCT computation context along with standard voltage scaling techniques such as pipelining and parallelism. The first method exploits the fact that image pixels are locally well correlated and exhibit a certain number of common most significant bits. These bits constitute a common mode DC offset that only affects the computation of the DC DCT coefficient and is irrelevant for the computation of the higher spectral coefficients. This observation follows directly from the linearity of the transform. The DCT chip has adaptive-bitwidth distributed-arithmetic computation units that reject common most significant bits for all AC coefficient computations, resulting in arithmetic operations with reduced bitwidth operands thus reducing switching activity. We call this method MSB rejection (MSBR).

The second method exploits the fact that in image and video compression applications, DCT is always followed by a quantization step which essentially reduces the precision of the visually insignificant higher frequencies. The DCT chip allows the user to program the desired precision of each spectral coefficient on a row-by-row basis so that no unnecessary computation is performed if the precision will be lost anyway due to quantization. Additional adaptation in computation precision is provided by a row-column peak-to-peak detector that classifies each block row and column into one of four classes of computation precision for maximizing image peak SNR (PSNR) and minimizing the number of arithmetic operations. We call this method row-column classification (RCC). The chip user can define precisely all four classes in addition to the desired precision per coefficient per class.

5.1 DCT Algorithm and Chip Architecture

Our DCT unit implements a row-column Distributed Arithmetic version of the Chen [CSF77] fast DCT algorithm enhanced with the activity-reduction methods outlined above namely MSB rejection and row classification.

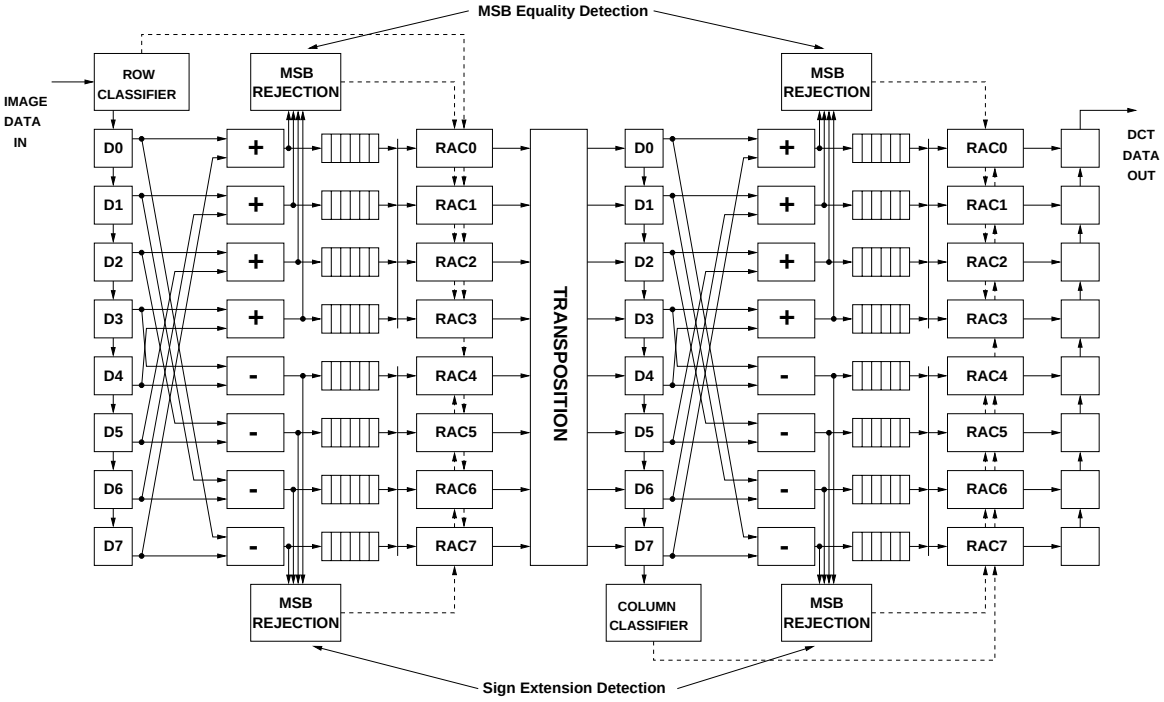


Figure 5.1: DCT Core Processor Block Diagram

The first step of the Chen algorithm is essentially a factorization of the DCT-II [RY90] matrix such that the subsequent computation of the even coefficients is fully separated from the computation of the odd coefficients. The 8-point 1-Dimensional DCT can be expressed as follows:

$$\begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix} \cdot \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \cdot \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \tag{5.2}$$

where

$$A = \cos\frac{\pi}{4},\ B = \cos\frac{\pi}{8},\ C = \sin\frac{\pi}{8},\ D = \cos\frac{\pi}{16},\ E = \cos\frac{3\pi}{16},\ F = \sin\frac{3\pi}{16},\ G = \sin\frac{\pi}{16} \tag{5.3}$$

Figure 5.1 shows a block diagram of the system architecture. It consists of two 8-point 1D DCT stages with a transposition memory structure in between. The transposition memory used is identical to the one used in the IDCT chip (Figure 3.6).

We will first concentrate on the basic functionality of this subsystem, ignoring for the moment the *row and column classifiers* that reduce the computation precision and the control blocks named *MSB rejection* that implement a sliding computation window that rejects common most significant bits and sign extension bits for the core computations.

Input pels are shifted into registers D0 through D7 of the first stage at a rate of one sample per clock. The subsequent column of adders and subtractors perform the first "butterfly" step contained in the last column vectors of equations 5.1 and 5.2. Data is then loaded into 8 shift registers that repackage the data into two 4-bit addresses that serially feed the ROM and ACcumulators (RAC0-RAC7) MSB first. Each RAC is a distributed arithmetic structure (Figure 2.1) that implements the dot product of the 4-element even (odd) processed pel vector and a row of the 4×4 coefficient matrix in eq. 5.1 (5.2). The ROM within each RAC contains linear combinations of the coefficient matrices row vector elements as eq. 2.13 suggests. Once every 8 cycles (given that pel sums and differences are rounded to 8 bits) the 8 dot products have assumed their final values within each RAC. The ROM within each RAC is a 16 by 10-bit wide structure, whereas the adder is 20-bit wide to accommodate continuously increasing operands due to the left logical shift in the accumulator feedback path. We should mention that there is no standard that defines minimum arithmetic precision requirements for the DCT (as opposed to the IDCT) and the implementation details are entirely left to the designer. Our choice of internal arithmetic bitwidths yields transformed images of very high quality with PSNRs typically in excess of 45 dBs. A detailed diagram of the DCT RACs in Figure 5.1 is shown in Figure 5.2. The ROM contents are labelled as follows: If the row vector $[A0\,A1\,A2\,A3]$ is replaced with a row of either constant matrix in equations 5.1 and 5.2, the resulting RAC computes the vector element in the same row position as the matrix row on the left-hand side of equations 5.1 and 5.2. We defer the discussion of the "Qualifying Pulse" until section 5.1.1.

Subsequent processing involves loading the results of the 8 distributed arithmetic computations in a transposition structure (1st stage) or loading in an 8-wide parallel-to-serial register that will output the coefficient stream at a rate of one sample per clock (2nd stage.)

Incoming pels are 8-bit wide unsigned integers. The result of the butterfly step is 9-bit 2's complement and is rounded to 8 bits. The results of the 1st stage 8 distributed arithmetic computations are 20-bit 2's complement and are rounded to 11-bit 2's complement before being passed to the second stage. The result of the second stage butterfly stage is 12-bit 2's complement rounded to 11-bit 2's complement. Second stage RACs 1-7 use only 8 significant bits of those 11 (using a sliding computation window as explained in section 5.1.1). RAC0 though uses the full 11 bits due to its high visual significance. The results of all RACs are 20-bit 2's complement rounded to 12-bit 2's complement before being shifted out of the chip.

5.1.1 MSB Rejection (MSBR)

We now focus on the mechanics of MSB rejection. Let us focus on the top half of the block diagram (both stages) in Figure 5.1. Due to spatial correlation in natural images, pel sums $x_0 + x_7, x_1 + x_6, x_2 + x_5, x_3 + x_4$ (equation 5.1) are likely to have a number of equal

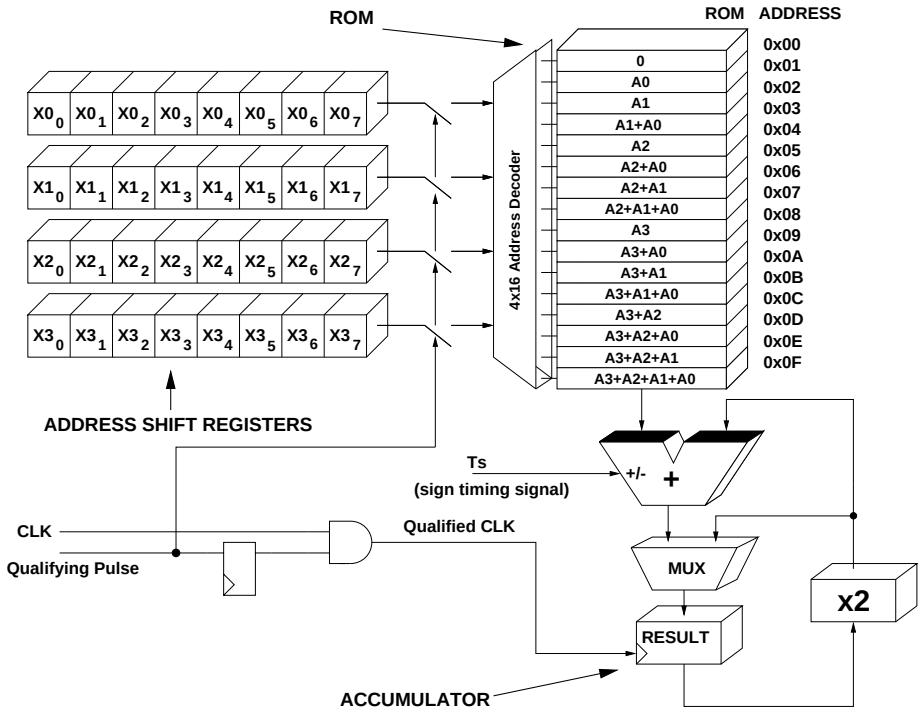


Figure 5.2: DCT ROM and Accumulator (RAC)

most significant bits. This statement will be quantified in section 5.2 where we evaluate the performance of this scheme. By definition (eq. 2.13) all Distributed Arithmetic ROMs store zero at address zero. Moreover, the ROMs within RAC1, RAC2, and RAC3 also store a zero at location 15 (0xF). Let us recall that location 0xF in a Distributed Arithmetic ROM stores the sum of all elements of the constant vector (eq. 2.13 with $b_{kn} = 1$). We observe from eq. 5.1 that all the rows of the 4×4 coefficient matrix except for the top row have elements that sum up to zero. Therefore, locations 0xF within those corresponding 3 ROMs store a zero. To summarize our observation, we have just shown that locations 0x0 and 0xF within RAC1, RAC2 and RAC3 contain the value zero. *This observation implies that equal most significant bits among pel sums have no effect on the total outcome of RAC1-RAC3 and can therefore be discarded from the computation.* The boxes labelled *MSB Equality Detection* detects the first different bit among pel sums starting from the most significant bit position. This information is used to produce the Qualifying Pulse of Figure 5.2 and only enable RACs 1-3 when the address at the corresponding ROM inputs refers to a non-zero-valued ROM location. An example of MSB rejection is shown in Figure 5.3(a).

RACs 1-3 therefore need less than 8 cycles on average to compute their corresponding dot products. This reduced duty cycle is exploited with the mechanism outlined in Figure 5.2 that disables the RACs in question in the presence of irrelevant inputs. We note that RAC0 has nonzero content at location 0xF (4A according to eq. 5.1) and therefore does not participate in this scheme. It is always clocked.

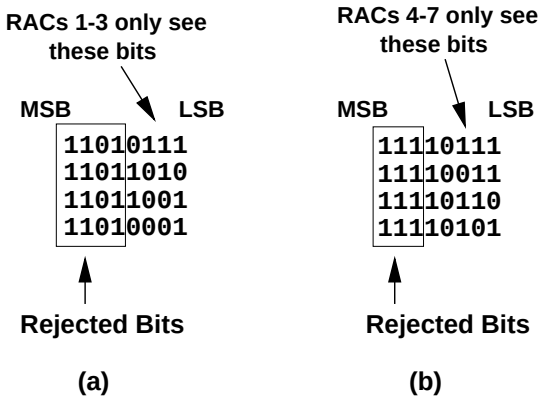


Figure 5.3: MSB Equality Detection and Sign Extension Detection Example

A similar method (but not 100% equivalent) has been applied to the pel differences $x_0 - x_7, x_1 - x_6, x_2 - x_5, x_3 - x_4$ (equation 5.2) at the bottom half of Figure 5.1 (both stages). Equation 5.2 implies that RACs 4-7 do not store zeroes at locations 0xF because the rows of the corresponding coefficient matrix do not sum to zero. Yet, another important observation implies that MSB rejection can be applied. Differences of highly correlated samples are very likely to result in small positive or negative values. Obviously, sign extension bits (either 0 for positive or 1 for negative values) do not affect the Distributed Arithmetic final results. The boxes labelled *Sign Extension Detection* detect the first different non-sign extension bit among the four pel differences. This information is used to reduce the duty cycle of RACs 4-7 in a fashion equivalent to the method applied to the pel sums. An example of sign extension detection is shown in Figure 5.3(b).

MSB rejection can be applied to all RACs on a row and column basis except for RAC0 (in both stages) which computes the DC coefficient among parts of other coefficients as well. Simulation experiments indicate that this method can reduce the duty cycles of the RAC units by a factor of 2 with no loss in arithmetic precision. This statement will be quantified in section 5.2.

5.1.2 Row-Column Classification (RCC)

As opposed to MSBR, Row-Column classification (RCC) reduces the overall signal activity by introducing a small error in the arithmetic computation. RCC sets an upper bound on the number of clocks that each RAC in Figure 5.1 will use to compute its corresponding dot product. As described in section 2.2.1, the DA operation (MSB first) can be thought of as a successive approximation operation where each cycle an extra bit from each element of the data vector is used to refine the final result. The RAC cycle upper bounds are implemented by user-programmable state registers within the chip and coupling to the preexistent (for MSBR) accumulator clock gating mechanism. The state registers store a clock mask which is ANDed with the clock mask generated by the MSB rejection mechanism. We should note that the cycle upper bounds refer to the maximum number of clock cycles a RAC is being

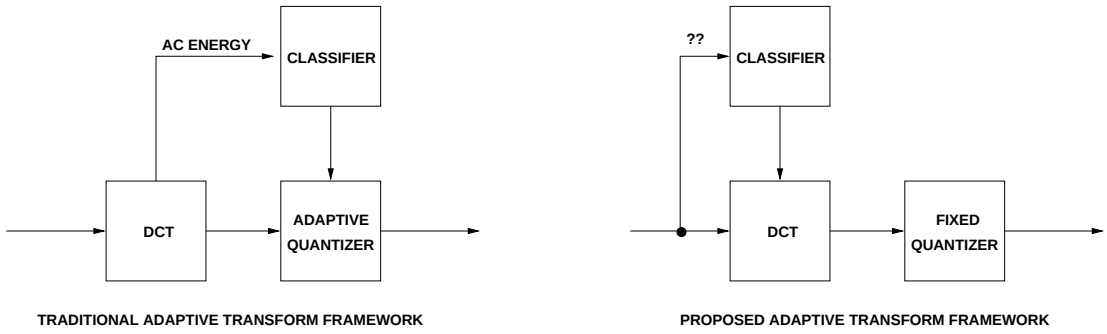


Figure 5.4: Traditional vs. Proposed Adaptive Transform Framework

clocked *after* MSB rejection has eliminated common most significant bits from the serial address registers. The user has serial access to the state registers through an IEEE Standard 1149.1-1990 Test Access Port (TAP) [IEE90b]. When the maximum number of cycles for a particular RAC has been reached and there are still data bits that await processing, the RAC simply powers down and uses its current dot product as an approximation of the final result. Please note that due to our MSB-first implementation of the Distributed Arithmetic Unit (Figure 5.2) stopping a RAC before it reaches its final value introduces scaling by a factor 2^{-n} where n is the number of remaining cycles required to complete the computation using full precision. An additional shift-only feedback path has been provided in each RAC (Figure 5.2) which does not engage neither the ROM nor the adder in order to undo this scaling.

We wish to provide additional adaptivity in the variable precision mechanism for further power and / or quality gains. In order to minimize the truncation noise introduced by the cycle upper bounds described above, we employ a row classifier for the first 1D stage and a column classifier in the second 1D stage. The idea is to have increased cycle upper bounds for rows (columns) that exhibit high "activity" (low correlation) and decreased bounds for rows (columns) exhibiting low activity (high correlation). We have verified that such a scheme will reduce the mean square error introduced for constant average RAC duty cycle. Quantitative simulation results will be presented in section 5.2.

Block classifiers proposed in the literature ([Gim75] [CS77] [MF92] etc.) cannot be applied in the present case because they all assume the availability of AC DCT coefficients. Figure 5.4 illustrates the difference between previously proposed classification schemes and the current requirements. The difference stems form the fact that we wish to use classification as a means to reduce computation while keeping quantization parameters constant as opposed to using classification for adapting the quantization matrix. We have considered using a block classifier based on the spectral coefficients of the previous block. This option is not attractive because the previous block may have rather different content than the present one. An additional problem is that such a scheme would cause bubbles in the pipelined structure of the system, since the classifier would not be available in time for the next pipelined block.

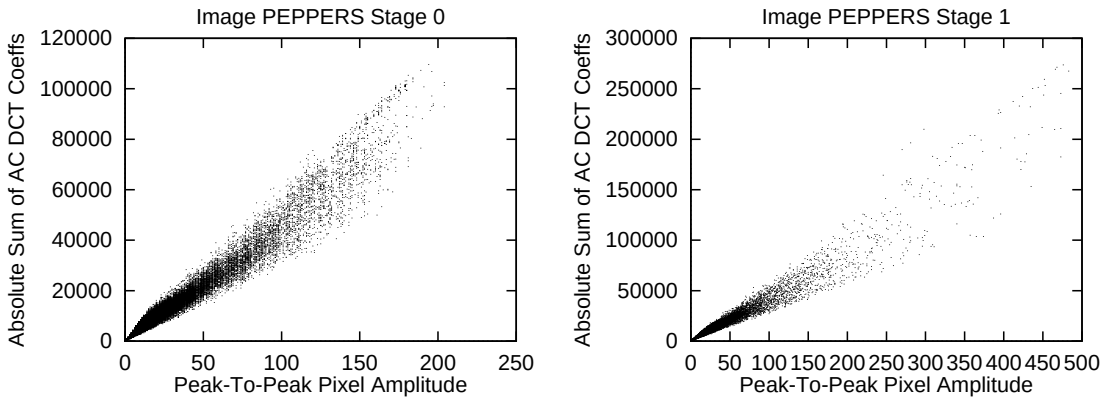


Figure 5.5: Correlation Between Proposed and Standard Classifier

Our requirements for an appropriate classifier geared to DCT computation reduction are:

- The classifier must be a function of the space domain pels and not a function of the transform domain coefficients.
- The classifier must be substantially correlated to prior proven classifiers such as the absolute sum of AC coefficients.
- The classifier must be a simple and easily computed function of the space domain pels. It should not add substantially to the total computational load otherwise energy savings won't be achieved.

We found that the peak-to-peak pixel amplitude per each row (PPA) has the potential to meet all three requirements. It is a function of the space domain pels (max(x_i)-min(x_i)) and is easy to compute. Furthermore, it exhibits substantial correlation with a widely used classifier:

$$\sum_{i=0}^{N} | X_i | \tag{5.4}$$

Figure 5.5 shows a scatter plot for the natural image PEPPERS that indicates substantial correlation between the proposed and standard classifier for both DCT stages. Table 5.1 lists the sample correlation between the two classifiers for a set of 11 test images. Figure 5.6 shows the cumulative distribution function for the PPA classifier for both DCT stages. The data has been collected from the set of 11 images listed in Table 5.1. We choose to divide all rows into four different classes defined by the 25, 50 and 75% ordinates of the distribution in agreement with [CS77]. Such classification results in equally populated classes. The PPA thresholds for each class are 6,15,37 for stage 0 and 5,12,29 for stage 1. The cycle bounds used for an initial design iteration are listed in Table 5.2. RAC0 in both stages has no cycle limit whatsoever and always carries its computation to full precision. Section 5.2.1 describes a systematic procedure for computing acceptable cycle bounds.

Image	Stage 0	Stage 1
CATHEDRAL	0.98763	0.98656
COUPLE	0.98429	0.98490
EUROPE	0.98423	0.98438
FLOWER	0.98123	0.97977
GIRL	0.97998	0.98307
LENA	0.98016	0.97932
MANDRILL	0.97808	0.98540
PEPPERS	0.97921	0.97519
PLANE	0.98647	0.98457
VAN GOGH	0.98624	0.98731
WATER	0.98431	0.98930

Table 5.1
Sample Correlation of PPA vs. AC DCT Coefficient Sum Classifier for 11 Test Images

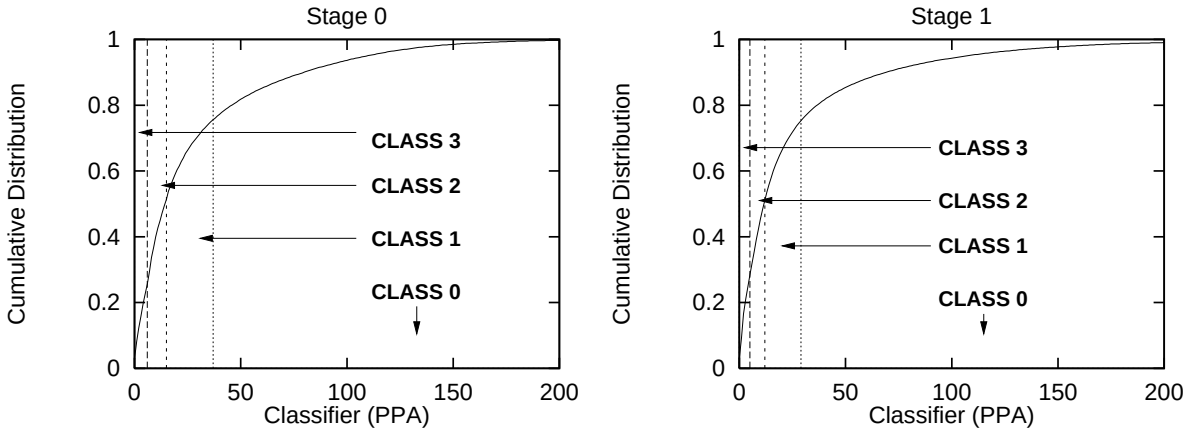


Figure 5.6: Image Row / Column Classification Threshold Determination

	RAC1	RAC2	RAC3	RAC4	RAC5	RAC6	RAC7
Class 0	8	6	6	4	4	3	2
Class 1	8	6	6	4	4	0	0
Class 2	6	4	4	0	0	0	0
Class 3	4	0	0	0	0	0	0

Table 5.2
RAC Cycle Upper Limits Per Class

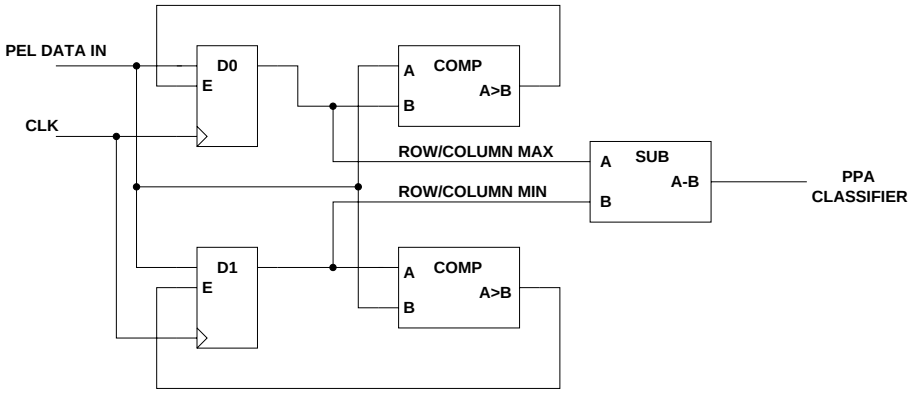


Figure 5.7: Row / Column Classifier Implementation

RCC is a simple and elegant way of trading off power dissipation and image quality with minimal circuit overhead (monitoring and clock gating mechanisms) and minimal image quality degradation in the mean square error sense. The PPA classifier is computed at each stage on a row (stage 0) or column (stage 1) basis using the circuit of Figure 5.7. This subsystem is contained within the row / column classifier blocks of Figure 5.1. The two comparators ensure that register D0 holds the row (column) maximum element and D1 holds the row (column) minimum element. The subtractor computes the maximum absolute difference (PPA classifier). The classifier is compared downstream with three user-supplied thresholds and the row (column) class identifier (0-3) is computed. The class identifier is used to select the appropriate set of RAC cycle upper limits for the distributed arithmetic computations.

5.2 Algorithm and Architectural Evaluation

A C model has been written that simulates the exact bitlevel behavior of the DCT processor before the design was committed to schematics. This high-level model has been used as a testbed in order to quantify the effect of the original methods in this work. This section summarizes architectural level simulation results. The 11 test images of Figure 5.8 have been used as an input to the simulator.

The first priority was to verify that there is indeed spatial correlation in typical images at the block row / column level and that the MSB rejection method would be capable of reducing the average number of operations (RAC accumulations) by a significant amount. Figure 5.9 displays histograms of RAC dot product computations vs. arithmetic bitwidth separately for each DCT stage. An arithmetic bitwidth of n indicates that a RAC unit has been clocked n cycles and thus has performed n ROM lookups and accumulations. This is equivalent to saying that the RAC in question has performed a dot product where the bitwidth of each element of the first (variable) vector was n bits.

In the absence of MSB rejection we would not have observed variable-bitwidth dot

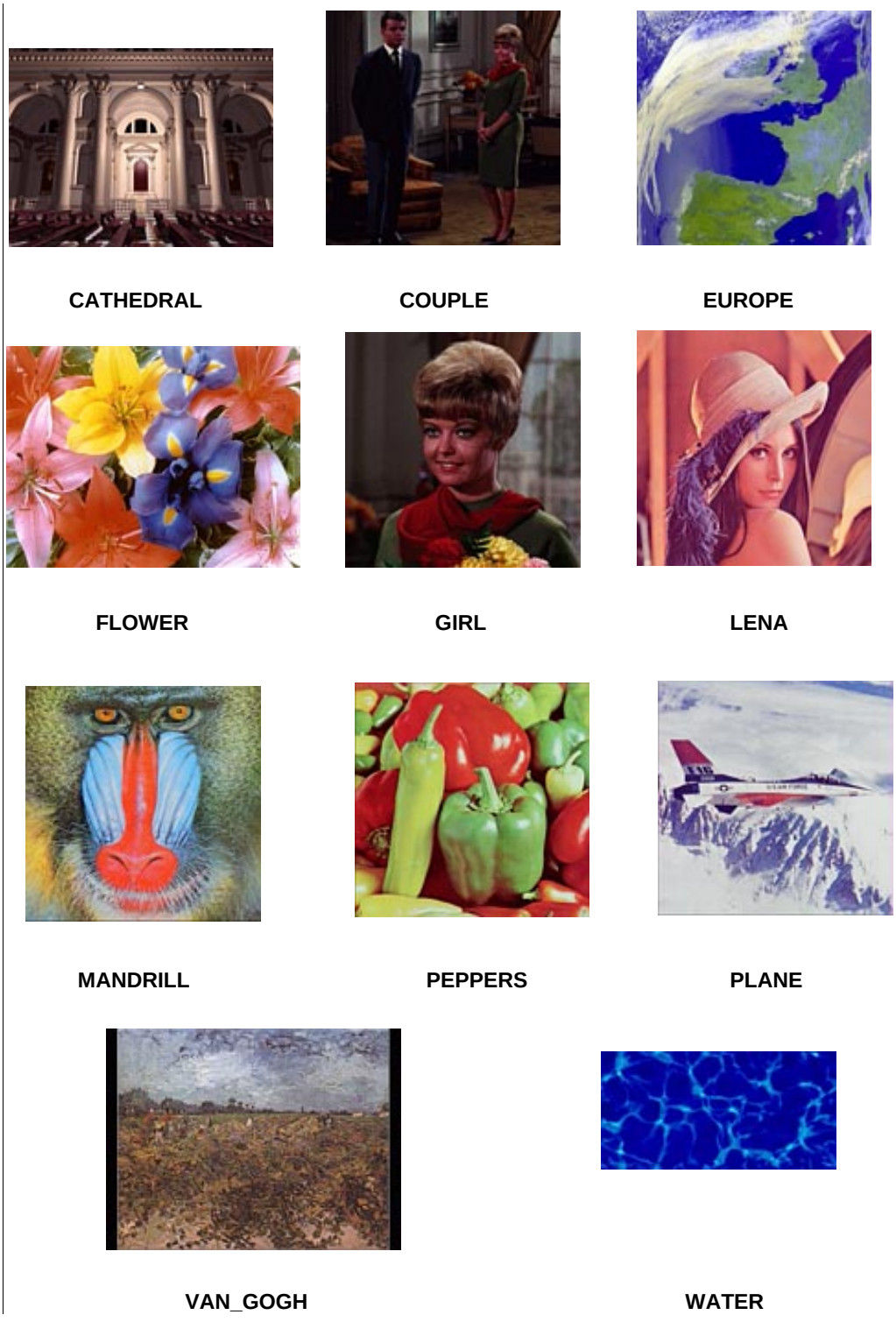


Figure 5.8: Images Used for Architectural Experiments

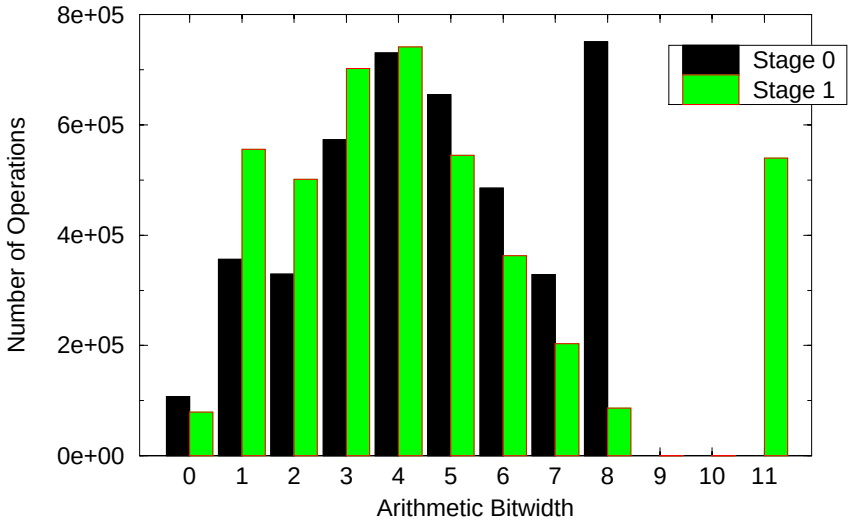


Figure 5.9: Histogram of Dot Product Operations vs. Bitwidth of Variable Vector in the Presence of MSB Rejection

product operations as the histograms indicate. The average dot product bitwidth would be 8 for stage 0 and 8.375 for stage 1. At this point, let us recall that RAC0 of stage 1 is always clocked 11 cycles (as opposed to 8) to achieve higher image quality. For this reason, the average for stage 1 is higher than 8 cycles. On the other hand, MSB rejection reduces the averages to 4.668 and 4.536 for stages 0 and 1 respectively. We conclude that MSB rejection can yield 40% savings in the number of operations required (RAC accumulations) and has the potential of being an interesting technique for power savings. The expected overhead is low: The duty cycle of the MSB detection mechanism is one out of 8 cycles, every time a new row (column) is loaded into stage 0 (stage 1) of the chip.

We now compare the total number of operations (additions) per block required by our DCT algorithm with that of the conventional Chen algorithm (distributed arithmetic implementation) that performs a constant number of operations per block. Figure 5.10 displays the number of additions per block required for the first 1000 blocks of image PEPPERS (Figure 5.8). The results plotted do not include computation savings due to RCC which has been disabled for this simulation. The graph also displays the number of additions required for the distributed arithmetic version of the Chen algorithm [CSF77] (Table 2.1). We observe that our MSBR-enhanced DCT algorithm requires about 40% fewer additions than the standard Chen algorithm implemented in a number of VLSI chips [SCG89][UIT+92][MHS+94]. We note that both computation algorithms displayed in Figure 5.10 include an equal number of ROM accesses in addition to the required arithmetic operations. MSBR also results in reduced ROM accesses by the same percentage (40%).

The quantitative results of Figures 5.9 and 5.10 indicate that the MSBR enhanced Chen DCT has substantial potential for low power and performs a smaller number of arithmetic operations when operating on image data than the most efficient DCT algorithms that

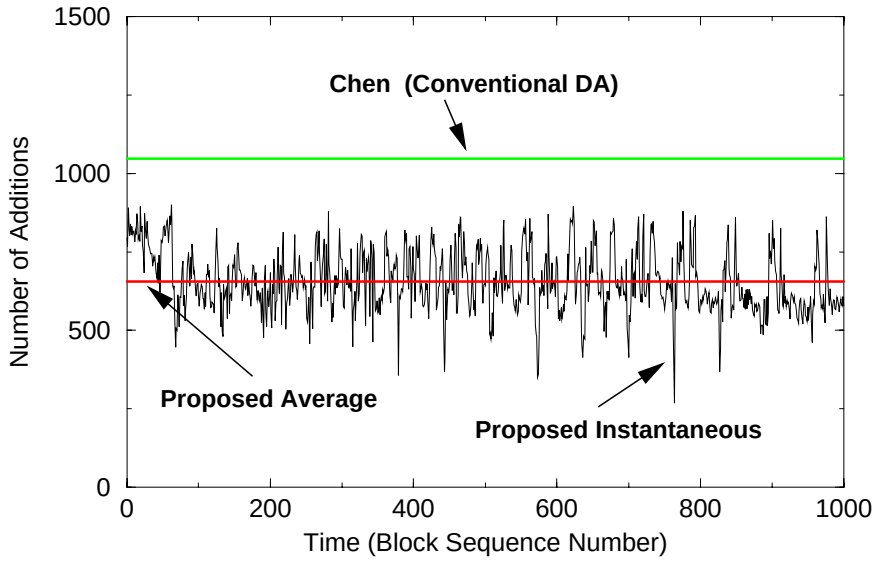


Figure 5.10: Comparison of DCT Additions (no RCC)

have been reported in the literature. The idea behind the algorithmic efficiency is identical to the one developed in chapter 3: We opt for algorithms that perform a variable number of operations per block depending on data statistical properties and result in an average lower than the most efficient fast algorithm that is not optimized for the the particular input data distribution.

Having established the potential of MSBR, we proceed to investigate the row / column classification method.

5.2.1 Row/ Column Classification Investigation

It is important to realize that MSB rejection does not reduce the arithmetic precision of the computation, but the row / column classification does. RCC essentially imposes a truncation on the input data in both 1D IDCT stages. We can model the loss in precision as two additive truncation noise sources, one per stage. These sources appear as $t_0[n]$ and $t_1[n]$ in Figure 5.11(a). In a typical DCT-based compression system, the DCT computation is followed by quantization. The quantization process applies large thresholds to the visually less significant higher spatial frequencies and produces a lot of zeroes at its output for purposes of compression. The process of quantization and inverse quantization at the image receiver can be modelled as an additive noise source $q[n]$ (Figure 5.11(b)) that results in image degradation (lossy compression). The main motivation for the RCC method is the fact that in almost all DCT applications, images incur quantization noise $q[n]$. If we could fold this process inside the computation and take advantage of the mandatory loss in precision, we could save computation and power while not affecting image quality. The quantization in a system such as the one in Figure 5.11(a) can be simply implemented as a

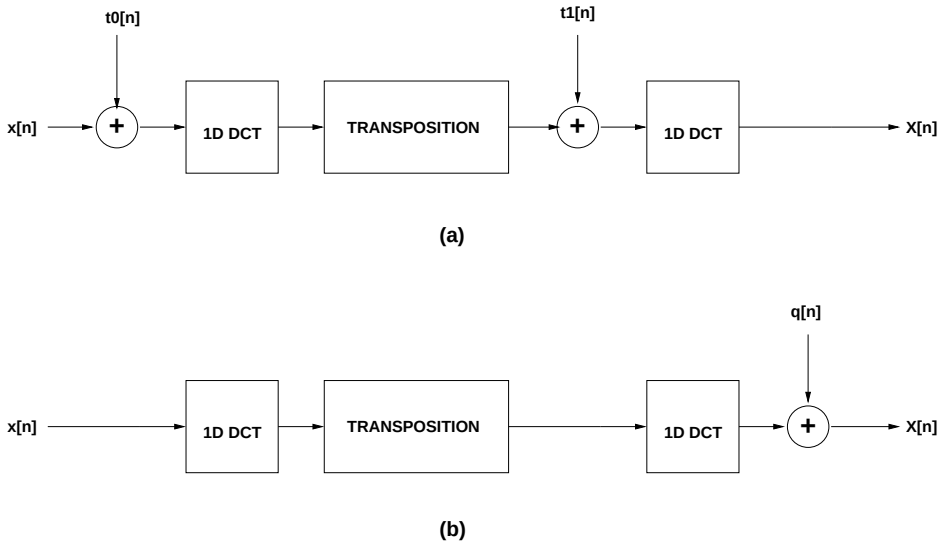


Figure 5.11: Additive Truncation / Quantization Noise Model

right shift of the data to get rid of the zeroes that have replaced the least significant bits of the computed coefficients due to premature computation inhibition.

The question we wish to answer at this point is what are the optimal cycle bounds for every row / column class so that the number of cycles that each RAC unit is clocked is minimized and the image PSNR (eq. 2.38) is maximized. The DCT chip has 56 individual degrees of freedom (2 stages $\times$ 4 classes $\times$ 7 cycle bounds per class). An additional question is what are the savings that we obtain (in terms of reduced cycles and increased PSNR) by having four separate classes instead of not classifying the rows and columns at all but simply imposing the same computation inhibition limits on all block rows and columns. We should note that we use the terms “computation inhibition limits”, “cycle bounds” and “cycle limits” interchangeably.

The starting point for our investigation was the computation inhibition limits of Table 5.2 which were determined by trial and error and visual inspection of the resulting compressed images. Table 5.3 list the average RAC cycles and PSNR that have been obtained from our simulator implementing MSBR and RCC with the cycle limits of Table 5.2 (same limits for both stages). As a reminder, RAC0 of both stages is clocked to completion and its computation is never inhibited.

The compressed images of Table 5.3 are quite acceptable visually. RCC has produced images of good quality (similar to ones produced by a full precision DCT followed by quantization) at an additional 35-40% savings in arithmetic operations. For comparison purposes, in Table 5.4 we list the PSNRs obtained for our 11 test images when compressed with a conventional high precision DCT algorithm and quantized with one-half and one times the step size of the matrices of Table 5.5 and 5.6 respectively. Quantization matrices in Tables 5.5 and 5.6 are listed as example luminance and chrominance matrices in the JPEG compression standard [PM92].

Image	Cycles	PSNR (dB)	Cycles (No RCC)	PSNR (dB) (No RCC)
CATHEDRAL	2.70	34.837	4.15	45.133
COUPLE	2.67	37.701	4.14	45.463
EUROPE	2.98	35.294	4.33	45.152
FLOWER	3.65	33.315	5.15	44.719
GIRL	2.86	36.326	4.38	44.976
LENA	2.93	35.853	4.49	44.807
MANDRILL	3.90	32.604	5.47	44.441
PEPPERS	3.07	34.476	4.63	44.857
PLANE	2.67	34.929	4.23	45.063
VAN GOGH	3.92	30.641	5.56	44.594
WATER	2.36	39.653	3.54	45.489

Table 5.3
Average RAC Cycles and Image PSNR for the Cycle Limits of Table 5.2

Image	PSNR (dB) (0.5 step)	PSNR (dB) (1 step)
CATHEDRAL	35.811	33.508
COUPLE	37.516	35.722
EUROPE	39.734	31.776
FLOWER	32.390	30.242
GIRL	36.269	34.762
LENA	36.809	35.290
MANDRILL	32.082	29.183
PEPPERS	35.263	33.927
PLANE	37.410	35.578
VAN GOGH	30.613	26.111
WATER	42.169	40.378

Table 5.4
PSNRs for Conventional DCT plus Quantization

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table 5.5
Luminance Quantization Step Matrix

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
26	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Table 5.6
Chrominance Quantization Step Matrix

Although the cycle limits (Table 5.2) that have been picked with trial and error perform very well, we should be able to do better with a systematic optimization procedure. An exhaustive (per image) search is computationally intractable since it requires 9^{56} iterations (56 variables taking values from 0 to 8) which we estimate to take 4.34^{46} years on a Sparc Ultra 60.

A fast algorithm has been devised to compute cycle limits that minimize PSNR degradation. Our algorithm is illustrated in Figure 5.12. It begins by assigning all 56 cycle limits the maximum value (8). Each iteration reduces one-by-one each one of the 56 cycle limits by one and recomputes the image PSNR. When all limit decrements have been tried and evaluated, the algorithm picks the decrement that reduced image PSNR by the least amount and commits it. The next iteration follows the same procedure, but from a different starting point. The algorithm finishes after 448 iterations when all 56 variable cycle limits are zero and returns the resulting average RAC cycles and PSNR per iteration. The chip user can use this table to figure out how the chip should be programmed given a minimum tolerable PSNR or given a maximum number of average RAC cycles. We observe that each one of the iterations is locally optimal, in the sense that the algorithm takes the best possible step at each point. We can make no claim however that at each point this particular algorithm maximizes PSNR and minimizes cycles. It is not always the case that a collection of consecutive optimal steps reaches a globally optimum point. Nevertheless, the present optimization algorithm has on average produced better results (higher PSNR given the same average computation cycles per RAC) than what we have achieved manually and provides good insight into the capabilities of the DCT chip.

Figure 5.13 shows the path that the algorithm traces while reducing the average number of RAC cycles for image PEPPERS. The fact that the PSNR is not a monotonic function of the number of iterations is attributed to finite precision in both the DCT and corresponding IDCT procedure necessary for the PSNR calculation between the original and processed image. The two bottom contour plots indicate the RAC cycle limits (for RACs 1-7) for all algorithmic iterations for both transform stages.

Figure 5.14 plots the achievable points that the optimization algorithm has yielded for all 11 test images. The optimization algorithm has produced better results than our intelligent guess for 8 out of 11 images. For images COUPLE, GIRL and LENA our intelligent

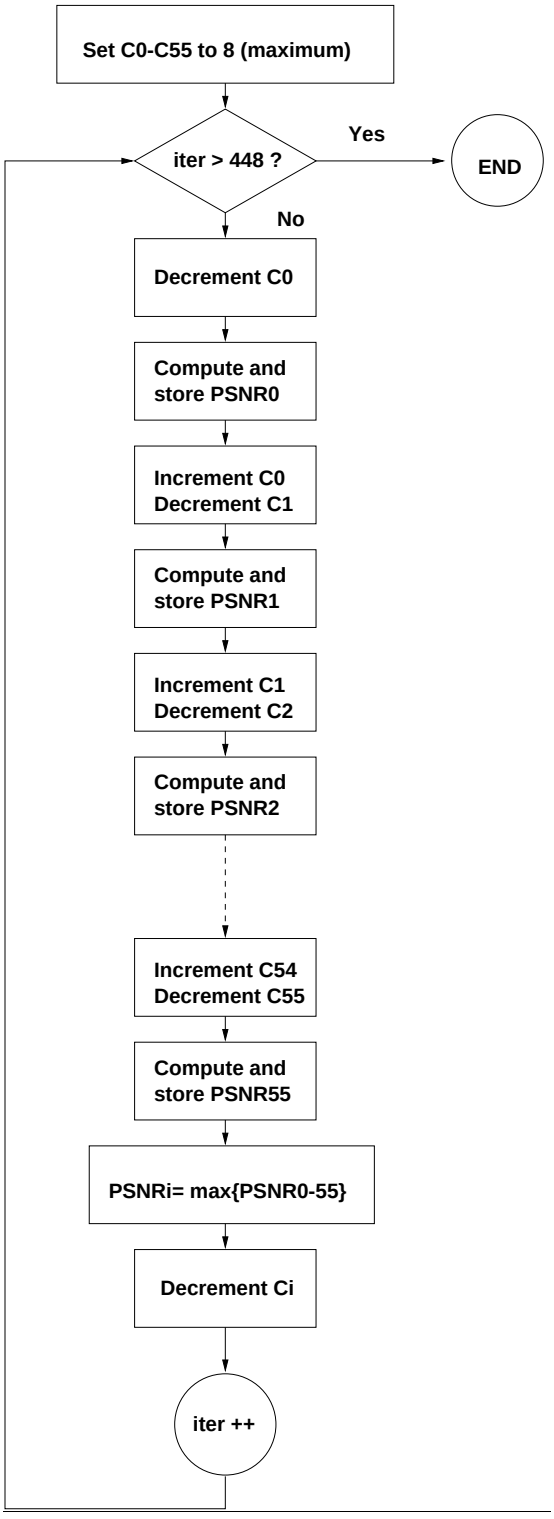


Figure 5.12: Optimization Algorithm for Computation Inhibition Cycle Limits

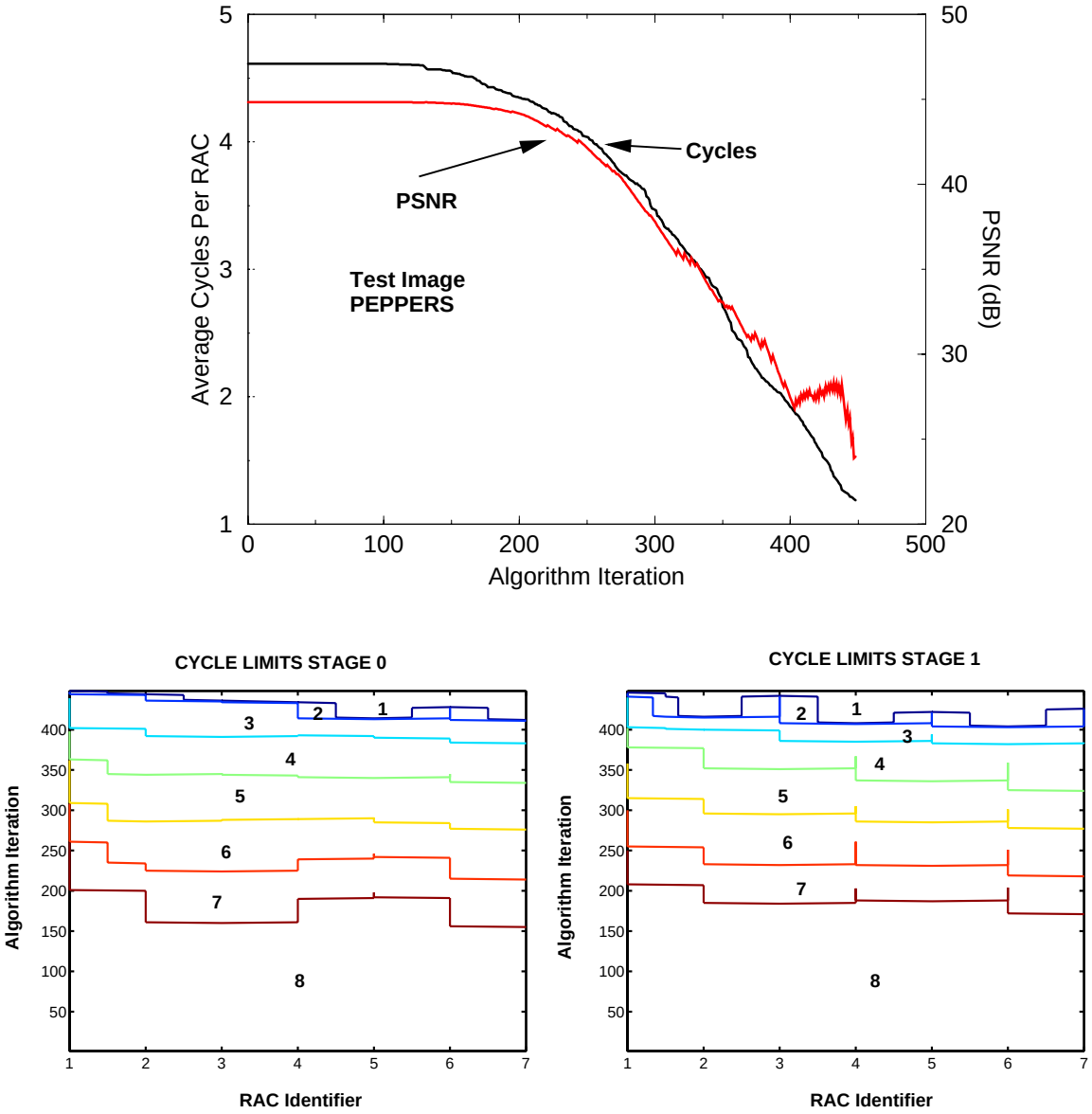


Figure 5.13: Optimization Algorithm Trace

guess has produced slightly better PSNRS than the optimization procedure, confirming our claim that this algorithm is not globally optimal. Figure 5.15 compares the PSNRs achieved by both procedures for the exact same number of cycles and specifically the cycle limits listed in Table 5.3.

The final question we wish to answer is what is the benefit of having four separate classes of computation inhibition cycle limits. Does it really yield higher PSNRs as was our initial guess? We have answered this question by running the optimization algorithm described above with the restriction that there are no four classes of cycle limits and there are only 14 free variables (2 stages $\times$ 7 limits per stage) and compared the results with the traces of Figure 5.14. The results for 6 test images are plotted in Figure 5.16 along with the corresponding 4-class traces. We observe that classification can increase the PSNR by as much as 100% and therefore is clearly justified. The reader should note that in certain images (i.e. PEPPERS and LENA) the two curves cross. This is another indication that our optimization algorithm for the 4 classes does not always yield globally optimal results. We note that more than four classes would introduce substantial complexity in the chip and would require substantial additional area. On the other hand, the adaptivity provided by less than four classes would reduce considerably the range of experiments we wished to perform.

5.3 Circuit Design

This section presents aspects of the DCT chip circuit design.

5.3.1 Flip-Flops

The core flip-flop design used in this chip is attributed to Thomas Simon [Sim99]. The author has made some small modifications to the original to add some extra functionality when needed. Figure 5.17 shows the basic flop.

Transistors P0, N0 form the master transparent latch. The slave latch is formed by transistors P1, P2, N1, N2. The output transistors P3, P4, P5, N3, N4, N5, invert the output and staticise the slave latch. This flop is static (slave only) and has no ratioed circuits. The staticizer feedback is broken when the slave latch is transparent. Since the slave latch is the only one staticised, it is only safe to gate the clock low. The clock inverter (P6, N6) can be amortized over multiple parallel flops although extreme care must be exercised by the designer: The master latch is non-inverting and therefore this flop is susceptible to skew between clock and its inverse: When both clock and its inverse are high for a brief period of time, a logic one in the data input d may propagate all the way to the output through transistors N0, N1 and N2. The same scenario can occur when both clock and its inverse are low and d is low. A local inverter (N6, P6) inside the cell minimizes such skew and guarantees correctness.

The present flop does not have any significant advantage for low power. On the contrary, it presents a considerable gate capacitance load to the clock (5 transistors.) We have

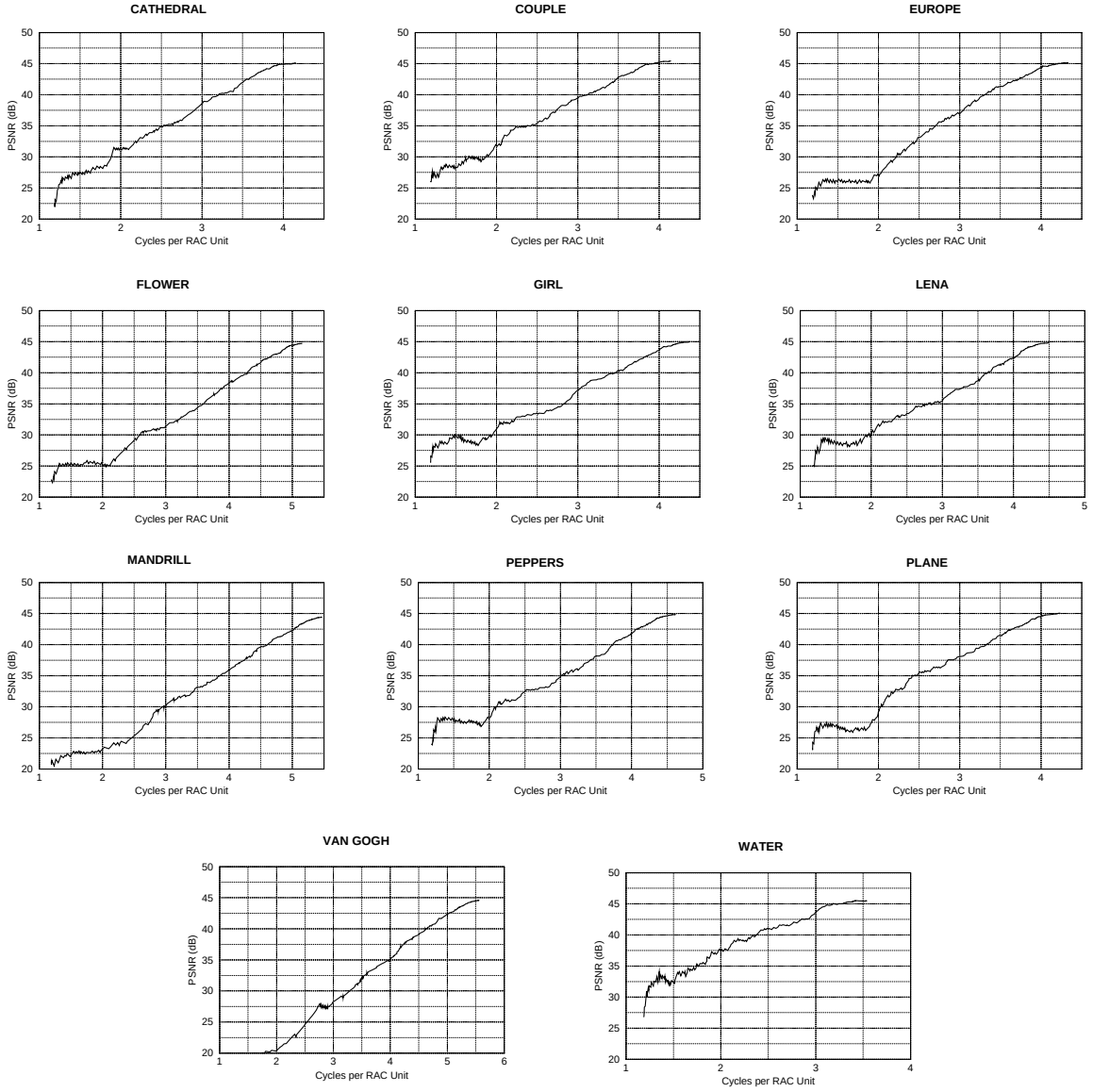


Figure 5.14: Optimization Algorithm Traces for all 11 Test Images

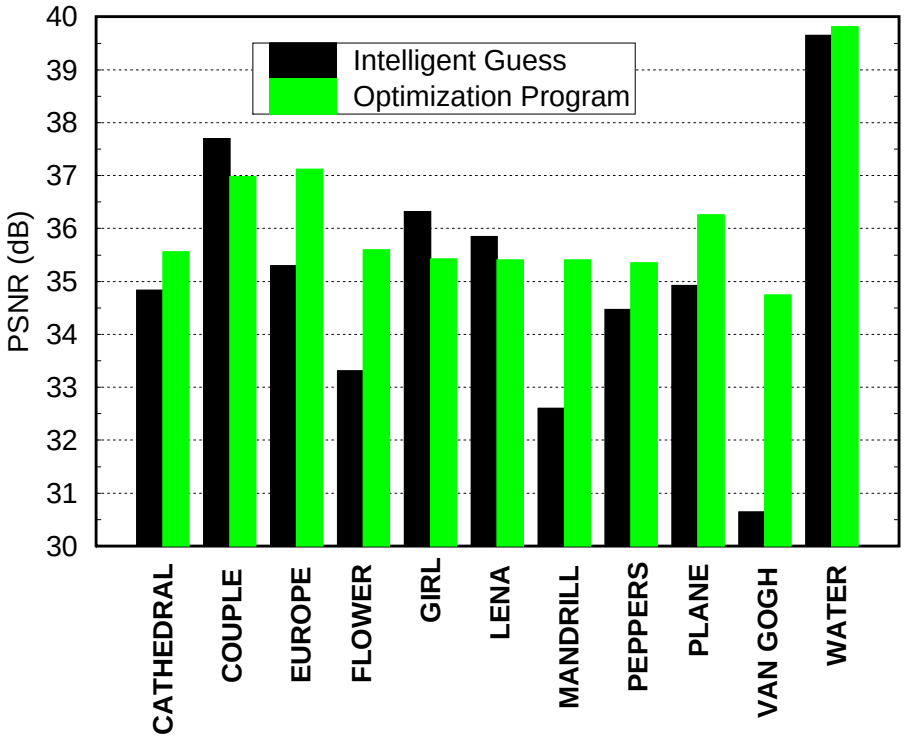


Figure 5.15: Comparison of Image PSNRs

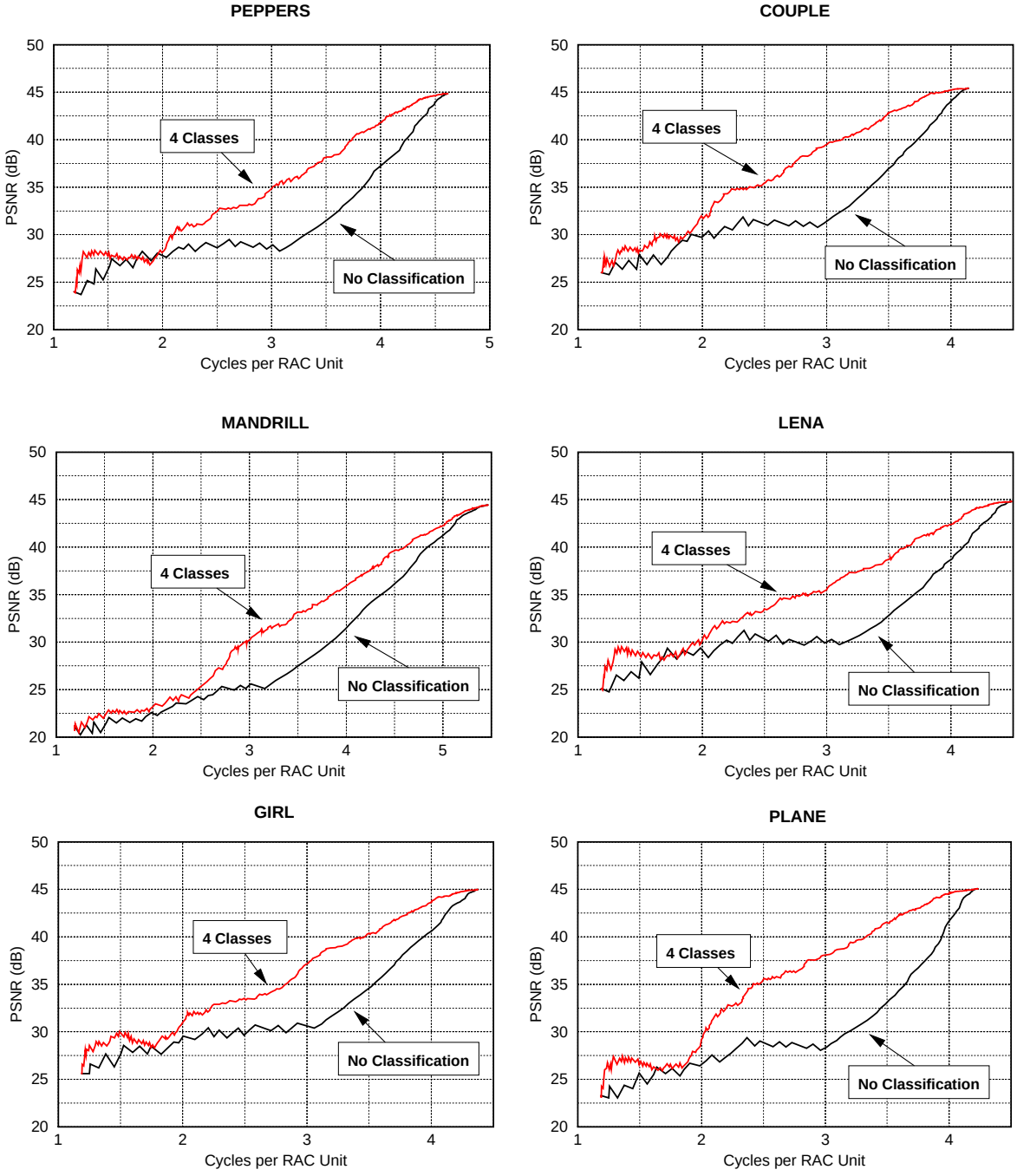


Figure 5.16: Optimization Algorithm Traces With and Without Classification

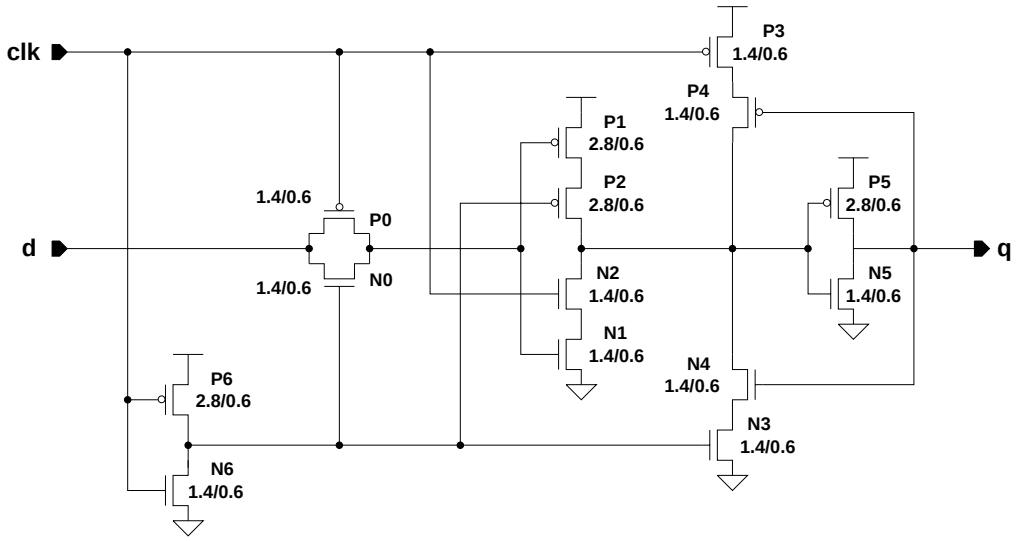


Figure 5.17: Edge-Triggered Static D Flip-Flop [Sim99]

used it because its functionality is guaranteed over process variations, supply voltages and small deviations in transistor sizing. There are no transistor fights and transistor sizing is not of critical importance.

Figures 5.18 and 5.19 show flops with clear and set control signals respectively. The clear and set signals can only be active though when clock is low. If clock is high while the clear or set strobe is active, both flops experience contention and their state may be corrupted: In Figure 5.18 if clock is high, clear is low and data had been high while clock was low, the input of the output inverter is simultaneously driven high and low by transistors P7, N1 and N2 respectively. The same is true in Figure 5.19. This time the fight is between N7 and the series combination of P1 and P2.

In the DCT chip where clock gating is abundant, such flops are widely used. In the rare occasion where a flip-flop must be preset and does not have a gated clock, the flops of Figures 5.20 and 5.21 are used. Transistors N8 (Figure 5.20) and P8 (Figure 5.21) ensure that contention never occurs when the preset pulse is active no matter what the value of the clock is. Although the last flops only contain a single extra transistor from the flops of Figures 5.18 and 5.19, this extra device makes a considerable difference in the flop layout and makes the clock-to-q propagation delay slower because it lies in the critical path.

The final type of flip-flop used in the DCT chip is a fully static one (both master and slave latch staticised) used in the JTAG instruction and data registers. This flop is shown in Figure 5.22. It has been constructed by cascading two static slave latches from the original flop of Figure 5.17. Both latches have been staticised to guarantee operation even at the slowest possible serial interface clocks (TCK).

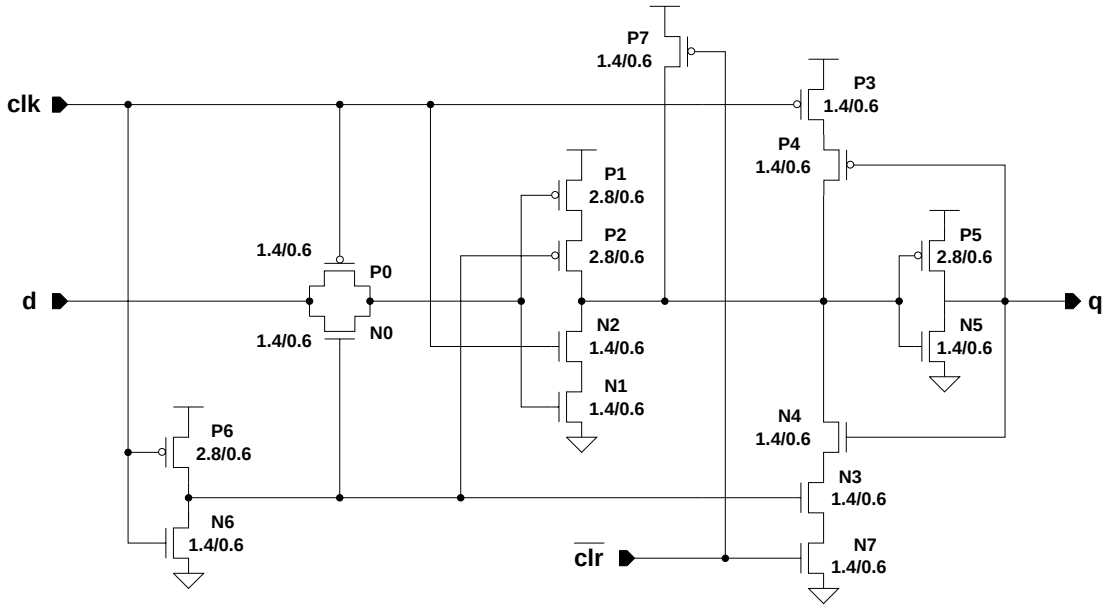


Figure 5.18: Edge-Triggered Static D Flip-Flop with Clear (Clock Low Only)

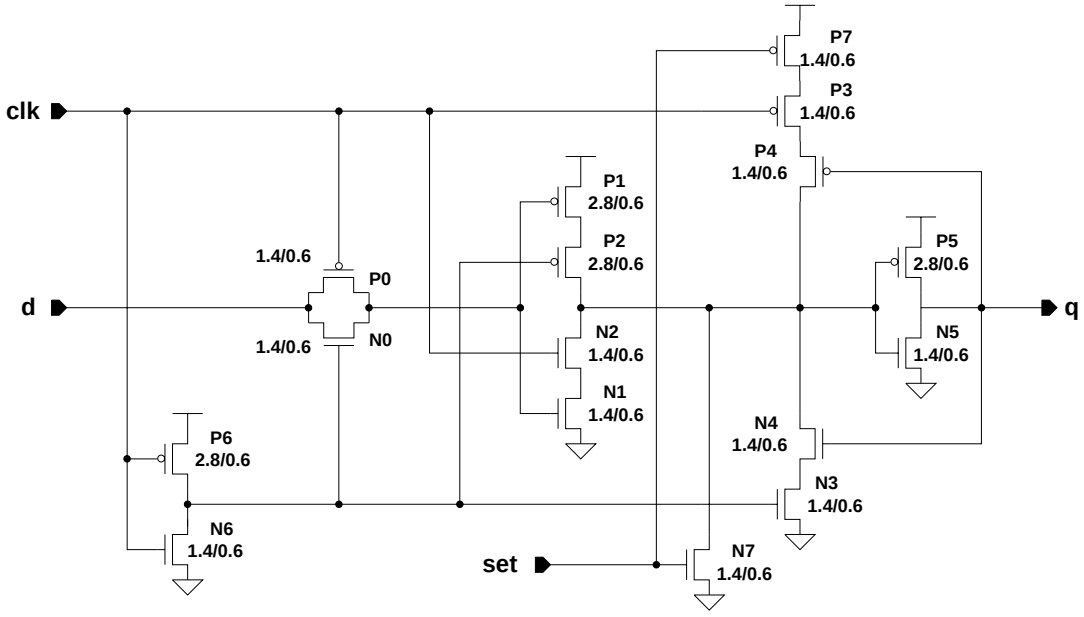


Figure 5.19: Edge-Triggered Static D Flip-Flop with Set (Clock Low Only)

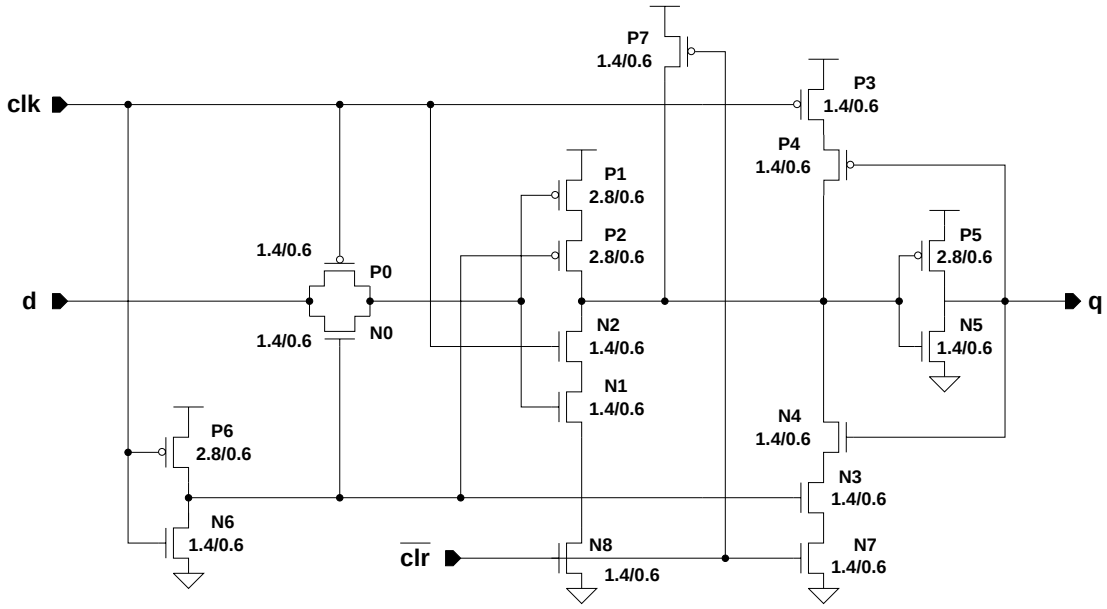


Figure 5.20: Edge-Triggered Static D Flip-Flop with Clear

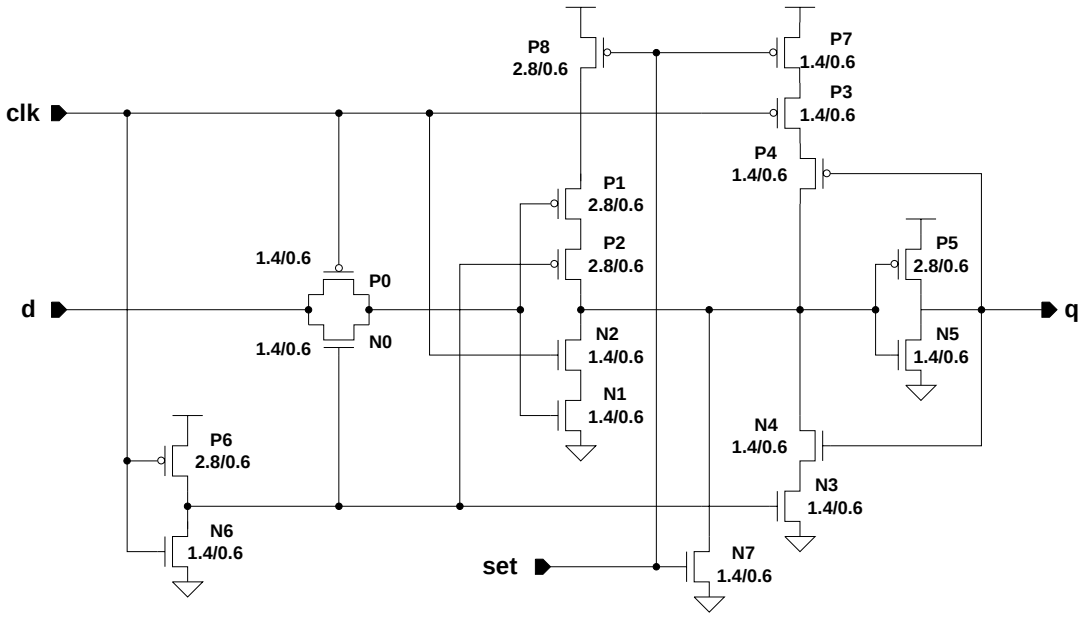


Figure 5.21: Edge-Triggered Static D Flip-Flop with Set

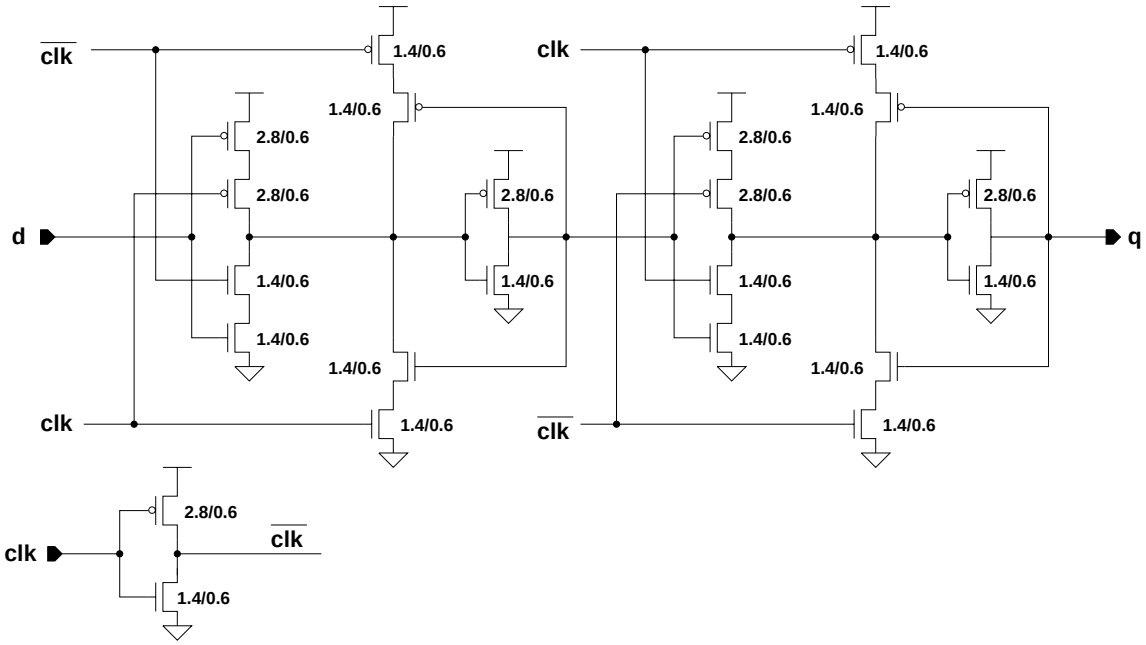


Figure 5.22: Fully Static Flop Used in JTAG Instruction and Data Registers

5.3.2 Read-Only Memory

A total of 17 separate 16x10 ROMs are used in the chip as part of the RAC units. The ROM circuit is shown in Figure 5.23. The 4x16 address decoder is implemented using 4-input NAND gates. As can be seen from the figure, the ROM is fully static (non-precharged). Bitlines are fully driven to the supply voltages through a 2.1/0.6 PMOS transistor and are driven to ground through a 1.4/0.6 NMOS. This fully static scheme is rather fast and performs very well at low supplies but has the additional overhead of requiring two wordlines (true and complement) for each row. Simulation has indicated that the average power dissipated on the wordlines is 7-10% of the total ROM power (depending on ROM contents) while the rest is dissipated by the bitlines. Therefore the additional overhead of dual wordlines is rather low at less than 5% of the total ROM power.

The main reason that a non-precharged ROM was selected was the fact that it is part of the RAC unit which has fully gated clocks. The generation of a precharge pulse would require considerable design overhead and would increase the overall switching activity of the unit.

Figure 5.24 shows a plot of ROM access time vs. supply voltage. The data has been obtained from Hspice simulations using extracted layout (wiring parasitic capacitances included.)

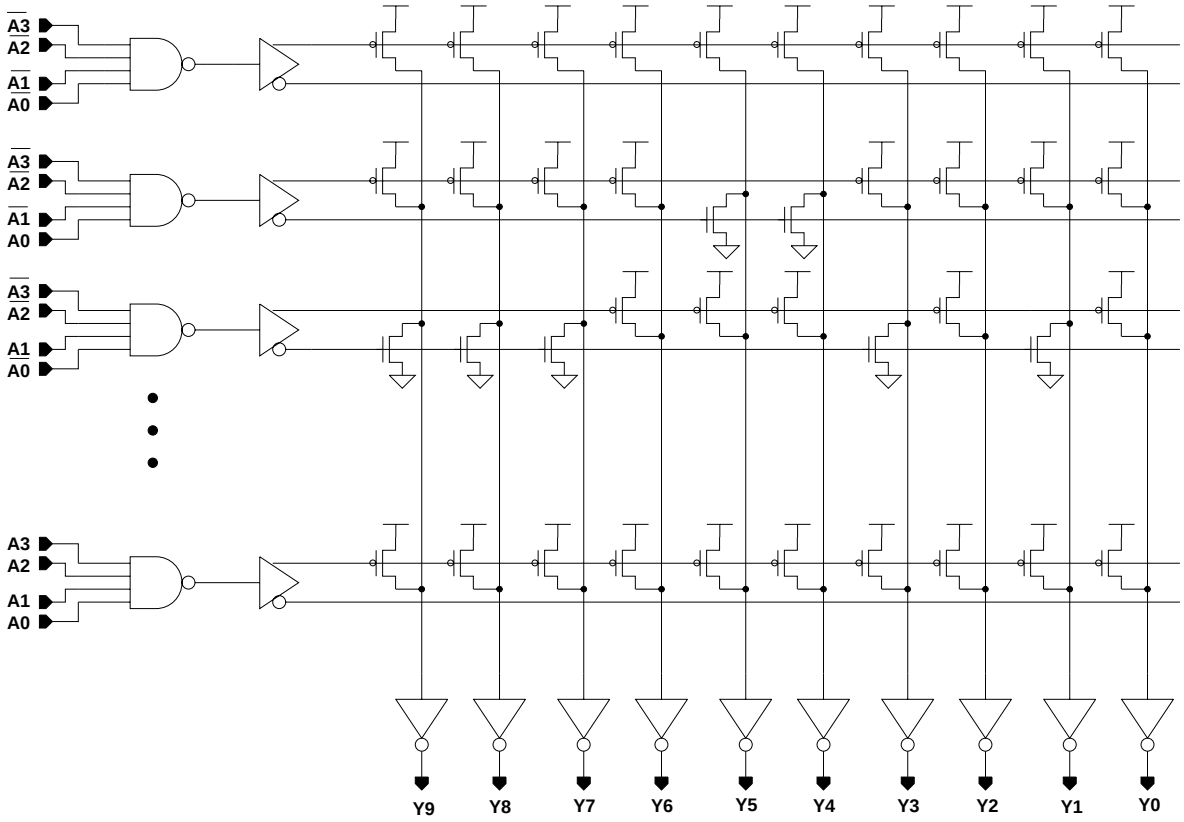


Figure 5.23: ROM Schematic

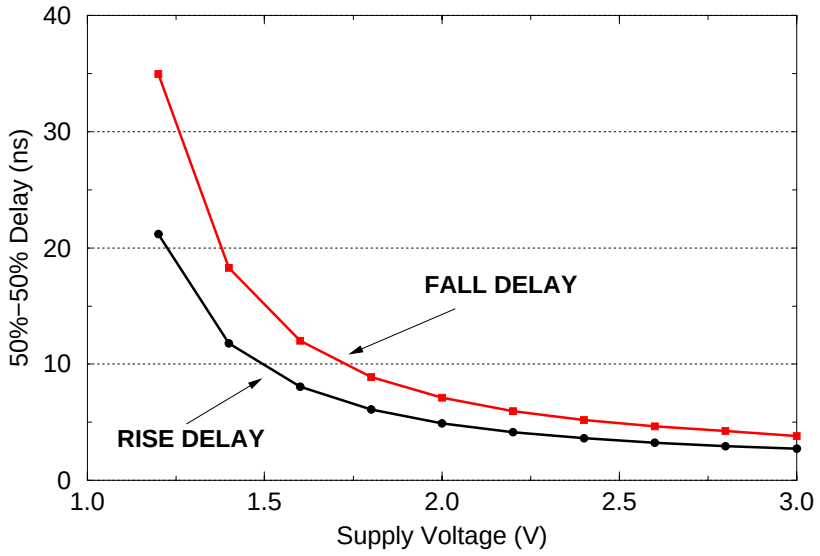


Figure 5.24: ROM Access Time vs. Supply Voltage

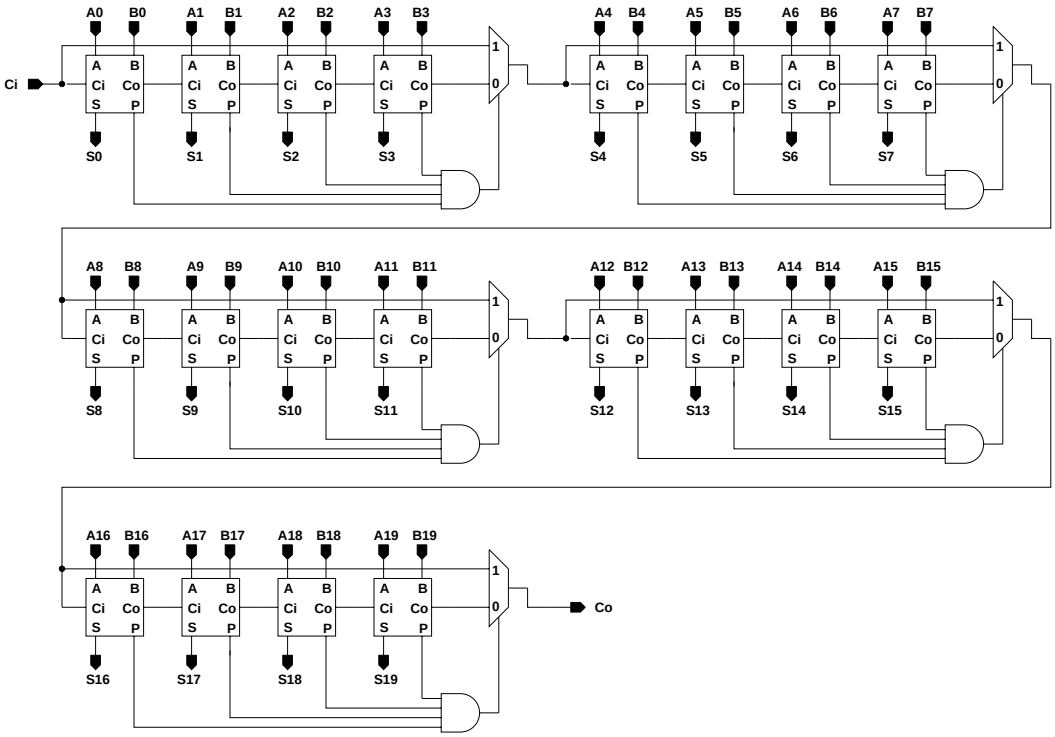


Figure 5.25: 20-bit Carry-Bypass Adder

5.3.3 Adder

The adders used throughout the DCT chip are the same as the carry-bypass adder shown in Figure 5.25. The figure shows a 20-bit adder which is the same one used in the RAC units (Figure 5.2). The bypass adder has been chosen because it has a short critical path at the expense of only a small increase in area. Its operation is very simple. The full adders are divided among cascaded groups of four. Although in a carry-bypass adder minimum delay is obtained when the adder groups have gradually increasing numbers of full adders, we decided not implement this idea mainly for layout considerations. When all the propagate signals within a certain full adder group are high, then the carry from the previous group is forwarded instead of the locally generated one without any change in I/O behavior. The critical path of the adder in Figure 5.25 is exercised when a carry is generated within the (A0, B0, S0) full adder and propagated all the way to the (A19, B19, S19) full adder. This critical path is approximately 8 full adder plus 4 multiplexer delays. The addition of 4 more extra bits of arithmetic bitwidth add a single multiplexer delay to the critical path.

The full adder used is shown in Figure 5.26. It has been selected because the propagate (P) signal necessary for the carry-bypass operation is computed explicitly, and due to its very small carry-in to carry-out delay. While cascading the full adders to form the 20-bit adder of Figure 5.25 we exploit the fact that a full adder is fully complementary (input

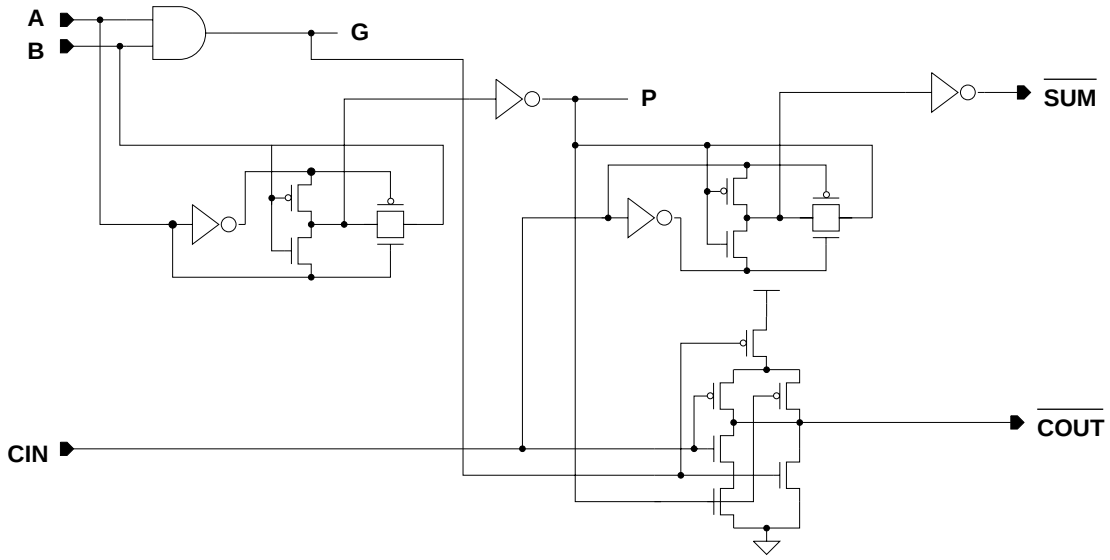


Figure 5.26: Full Adder Used in the DCT Chip

complements produce output complements) to avoid unnecessary inversions in the carry propagation path. Figure 5.27 plots the maximum propagation delay of the 20-bit adder in Figure 5.25 vs. supply voltage.

5.3.4 MSB Rejection

Figure 5.28 shows the circuit that detects common most significant bits and computes a clock mask that is used to disable RAC units 1, 2 and 3 of each stage when the ROM address inputs do not affect the computation. This circuit operates on the 4 sums (A7-0, B7-0, C7-0, D7-0) of the "butterfly" additions of each stage (Figure 5.1.) Transistors N1, N2, N3, N4 and N5, N6, N7, N8 detect the presence of all ones or all zeroes at the input respectively. The cascading of the outputs through the NAND gates and inverters activate all the remaining downstream mask bits as soon as the first bit position is found where 8-bit wide inputs A, B, C, D have different bits. The computed clock mask (CLKMASK) bits indicate when the RAC units 1 through 3 should be clocked (1 in the corresponding bit position) and when not (0 in the corresponding bit position.)

Figure 5.29 shows the sign extension detection circuit that operates on the 4 differences after the "butterfly" subtraction step of both stages (Figure 5.1). This circuit produces a clock mask (MASK) that controls the clock of RAC units 4-7. The daisy-chained XNOR gates of Figure 5.29 detect the first non-sign bit of each one of the four inputs. The cascaded AND gates activate all downstream partial mask bits from the position of the first non-sign bit. Then, the results of the four individual computations are merged with 4-input NAND gates to compute the first bit position that all four inputs *do not* contain sign extension bits. Both circuits of Figures 5.28 and 5.29 have a low duty cycle because the inputs change once

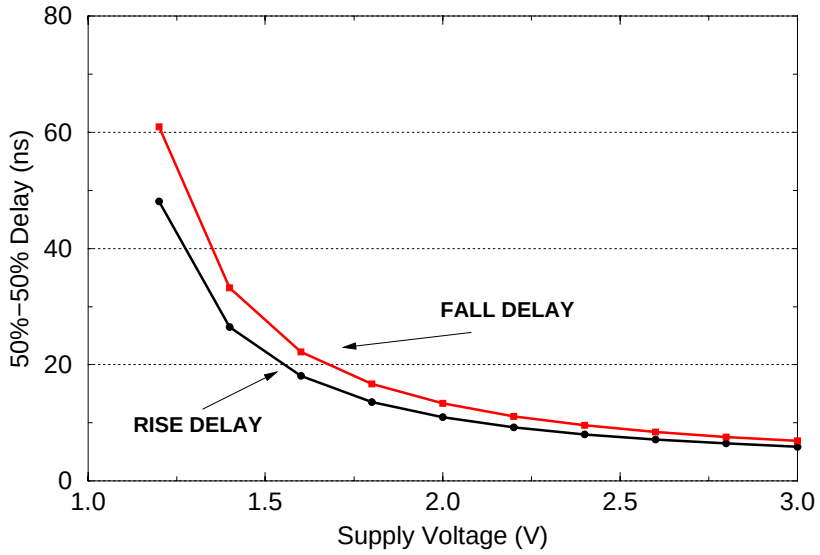


Figure 5.27: Carry-Bypass Adder Delay vs. Supply Voltage

every eight cycles (the time required to bring a new row on the chip). As a result they are responsible for a small percentage of the total power (section 5.4.)

5.3.5 I/O Pads

The entire pad design (schematic and layout) was done by Thomas Simon [Sim99]. The author's contribution was the determination of the pad driver transistor sizes and simulation and verification of the entire pad circuit. Both input and output pads are level converting (core voltage can be 1-3V while output voltage is TTL compatible at 5V).

Determining Driver Transistor Sizes for the Output Pad

The circuit used to determine driver transistor sizes is shown in Figure 5.30. The worst case capacitance and inductance values for the PGA package traces have been obtained from MOSIS package documentation available online at www.mosis.com. The rise time of the OUT signal must be fast enough to accommodate the speed which we wish to clock the chip but at the same time be slow enough to minimize $L(di/dt)$ ringing noise on the supply rails. The transistor sizes indicated in Figure 5.30 result in rise and fall times on the order of 2-3 ns which was deemed appropriate for our application. An Hspice simulation of the entire output pad will be presented in the following section.

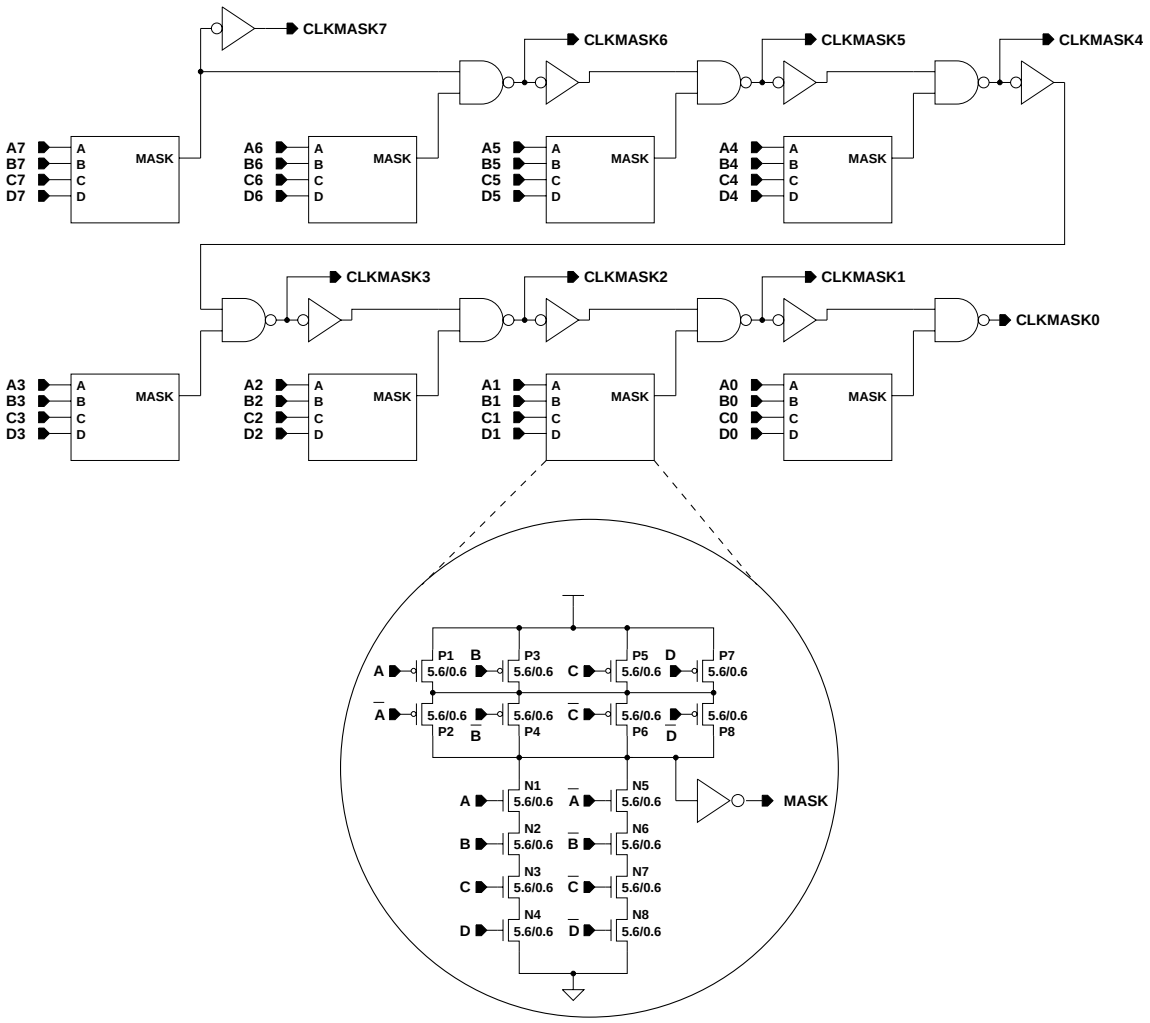


Figure 5.28: MSB Rejection Circuit

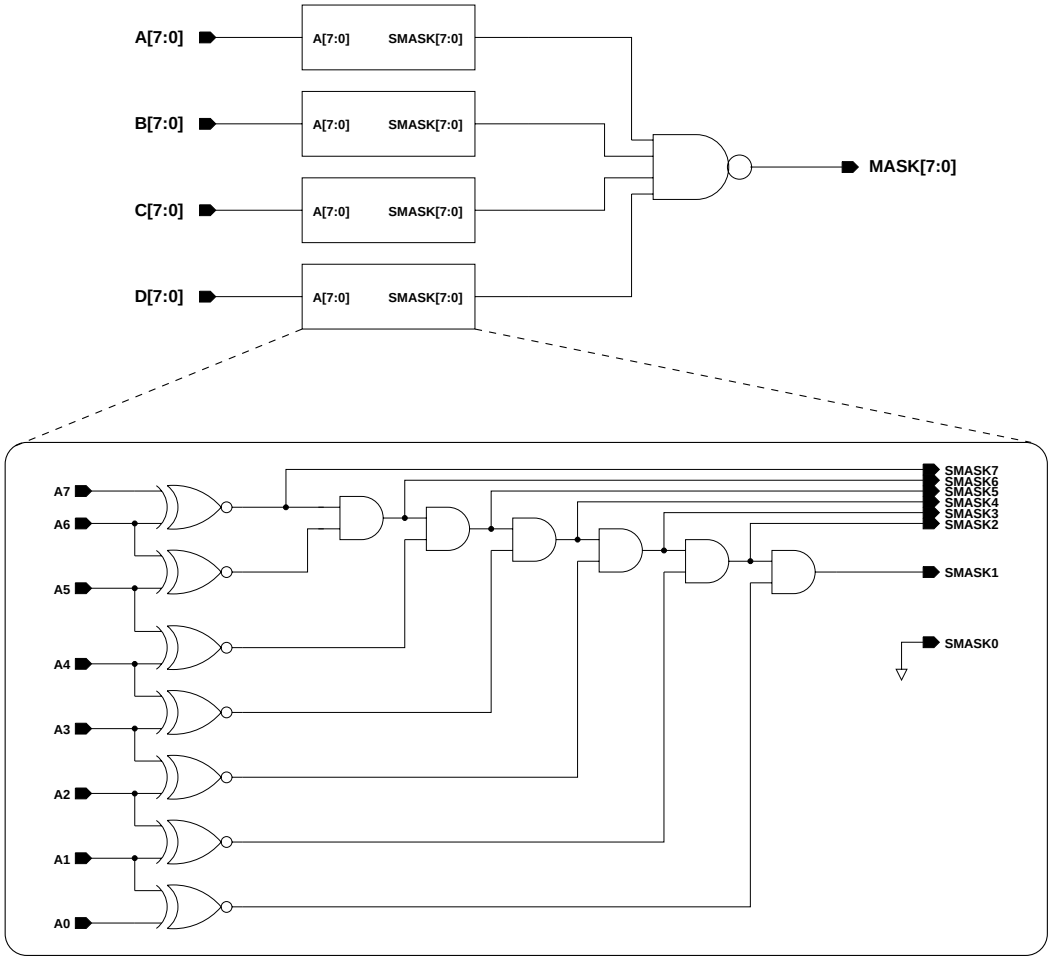


Figure 5.29: Sign Extension Detection Circuit

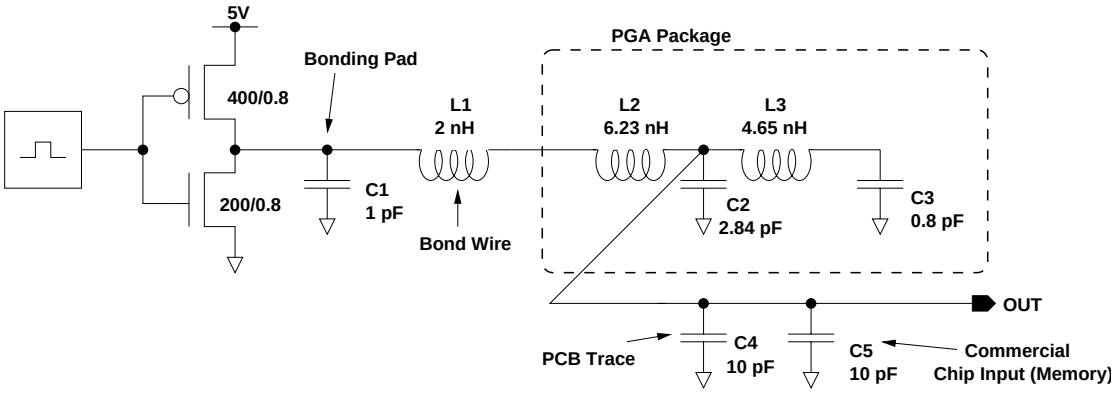


Figure 5.30: Circuit Used to Determine Driver Transistor Sizes

Output Pad

The entire schematic of the output level converting pad is shown in Figure 5.31. Devices P0 through P4, N0 through N4 and P9, N9 constitute a level converter from the chip core VDD (1.2V-3.0V) to the TTL-level output VHH (5V). The remaining transistors constitute an exponential driver that connects to the bonding pad. To understand the operation of the level converter, let us assume that the pad drives a zero, so that nodes IN, and NET2 are at zero, nodes NET1 and NET4 are at VHH and node NET3 is at VDD. Let us now assume that IN is driven to VDD. The series combination of N0 and N1 overpower the weak transistor P1 and start driving node NET4 to ground. P2 drives node NET2 to VHH which turns competing transistor P1 off. At this point, the one propagates all the way to the pad output. At the same time, NET1 becomes low and NET4 is driven to VHH preparing for the next input transition. When IN is driven back to ground, NET2 becomes zero and propagates to the output. The feedback path will guarantee that nodes NET1 and NET4 will become consistent with the new state and be prepared for the next input transition. From the preceding discussion, it is evident that special attention should be exercised in the relative sizings of transistors P1 and N0, N1 in addition to the sizing of transistors P4 and N2, N3. The NMOS pairs N0, N1 and N2, N3 must always be large enough to overpower P1 and P2 respectively given the VHH-VDD difference in their gate-to-source voltages. The author sees no particular advantage in the level converter of Figure 5.31 compared to the level converter of Figure 3.15.

Figure 5.32 shows an Hspice simulation of the output pad driving the load of Figure 5.30 at VDD= 1.3V and VHH= 5V. The rise and fall times are 1.73 ns and 1.42 ns respectively. The delay through the circuit is 7.9 ns.

Input Pad

The circuit diagram of the input pad is shown in Figure 5.33. Primary electrostatic discharge (ESD) protection is provided by bipolar diode D and resistor R. Bipolar diode D is an N-diffusion to P-substrate explicit diode. It protects the IN node from going lower than -0.6V or higher than its reverse breakdown voltage limit. Resistor R is approximately 3K Ohms and is formed by three squares of lightly doped N-well. Its purpose is to attenuate the off-chip signal before it reaches the gate of the sensing inverter (N1, P1). Transistor N0 is a large device providing secondary ESD protection as a second set of diodes that do not allow node IN to stray too much from its expected voltage range (0-5V). The diode to ground is formed by the source-to-P-substrate junction of transistor N0. The diode to VHH is the diode-connected MOS transistor itself. Voltage conversion is performed between inverters (N1, P1) and (N2, P2) which have separate supply voltages.

5.4 Power Estimation and Breakdown

We have used the Pythia [XYC97] power estimator to calculate the power dissipation of the DCT chip. A brief overview of the power estimation tool has been given in section 3.4.

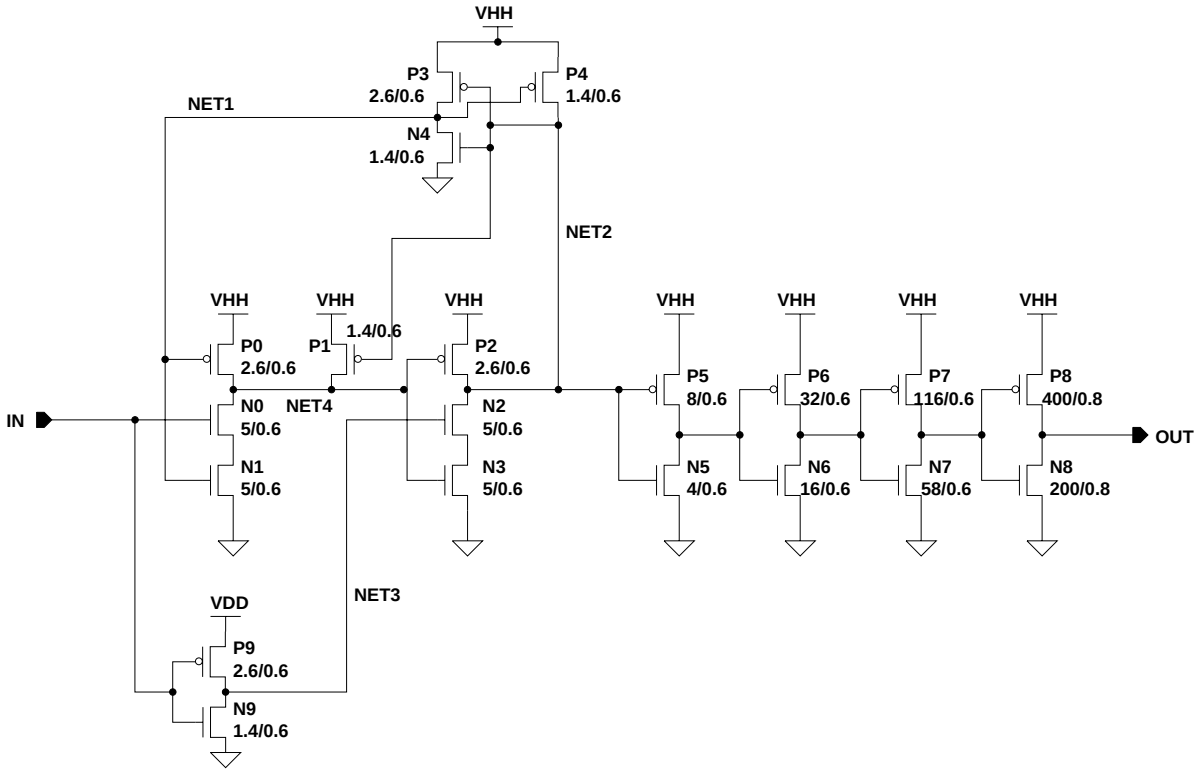


Figure 5.31: Output Pad Schematic

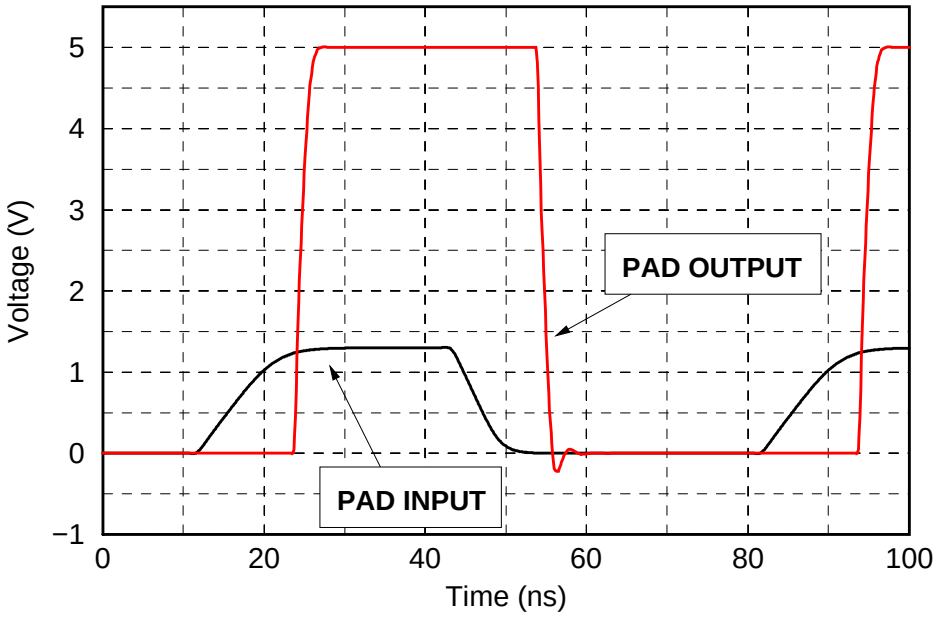


Figure 5.32: Output Pad Hspice Simulation

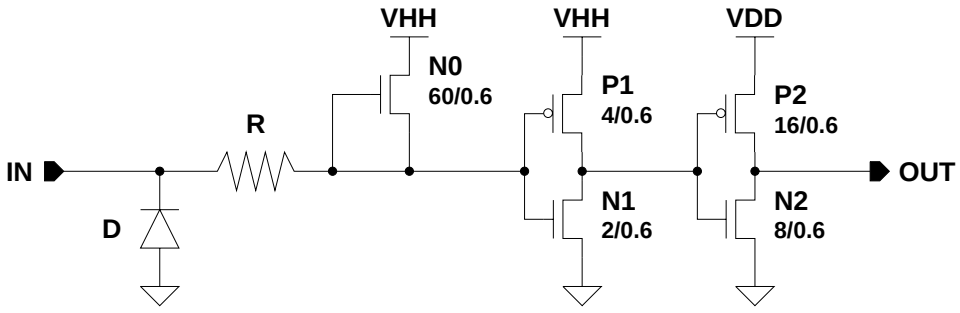


Figure 5.33: Input Pad Schematic

Figure 5.34 shows the power dissipated by the DCT chip partitioned among the toplevel design modules. The chip has been stimulated with the first 1000 blocks of test image PEPPERS using the cycle bounds of Table 5.2. The simulation includes estimated interconnect capacitance. The total power estimated for this particular data set is 7.4 mW at 1.56V, 14 MHz.

We observe that according to the power estimation results, the main clock buffer dissipates most of the chip power (almost 37%). Part of the reason why this power is so high is the fact that the Pythia interconnect estimation mechanism penalizes considerably nets with high fanout. Pythia assumes a square law dependence between net interconnect (wiring) capacitance and net fanout. As a result, the clock net which has the highest fanout is allocated a large value of total wiring capacitance. Almost 60% of the total clock buffer power is due to wiring parasitics. Figure 5.35 shows the same power breakdown data obtained by Pythia with power estimation turned off. We observe that the total clock buffer power has decreased to about 21% of the total. In addition to simulation artifacts, the clock buffer power is high due to the large inverters used to buffer long wires and the high duty cycle of the circuit. The last inverter of the driver chain has a 1.45 mm PMOS width and a 0.728 mm NMOS width.

The chip pure computation power (stages 0 and 1) are estimated at about 42% of the total (Figure 5.34). The charts of Figures 5.34 and 5.35 provide an additional very useful and relevant piece of information: They quantify the additional power cost of the power saving techniques employed (MSB rejection and computation inhibition plus classification.) We observe that the cost of the MSB rejection control logic is approximately 3-4% of the total (about half of the MSBR+Inhibition control sector) whereas the computation inhibition plus row / column classification is at 5-6% of the total (about half of the MSBR+Inhibition control sector plus the row / column classifiers.) We will use this result in section 5.7 where we will quantify the power savings obtained by these methods.

Finally, Figure 5.36 shows the power dissipation allocation within stage 0 of the DCT chip. The ROM and Accumulator Units are responsible for most of the power dissipation (78%) whereas the input shift registers, RAC address shift registers and “butterfly” adders and subtractors (Figure 5.1) dissipate the rest of the power.

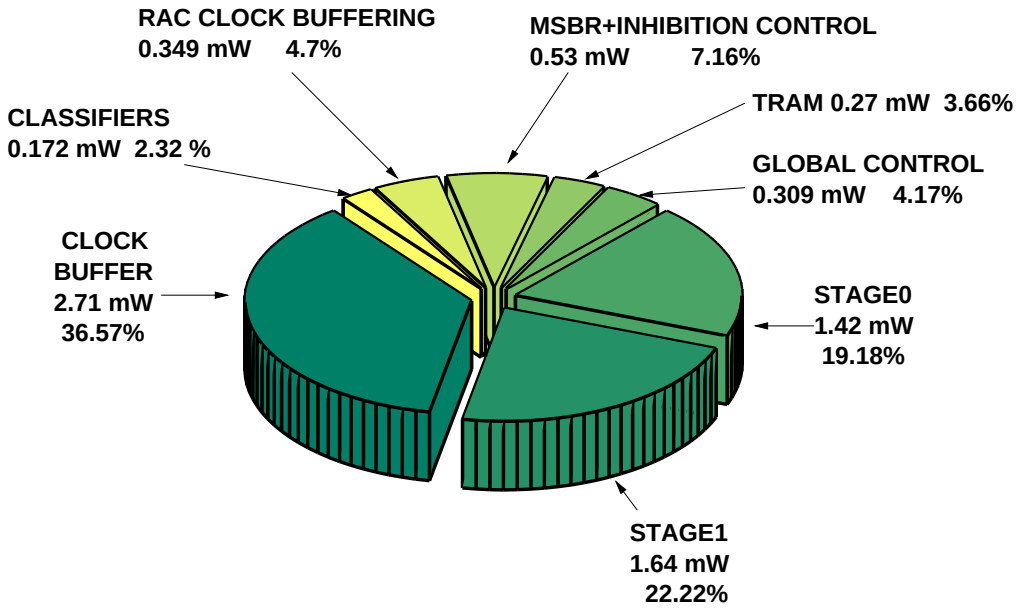


Figure 5.34: DCT Chip Power Estimation Results

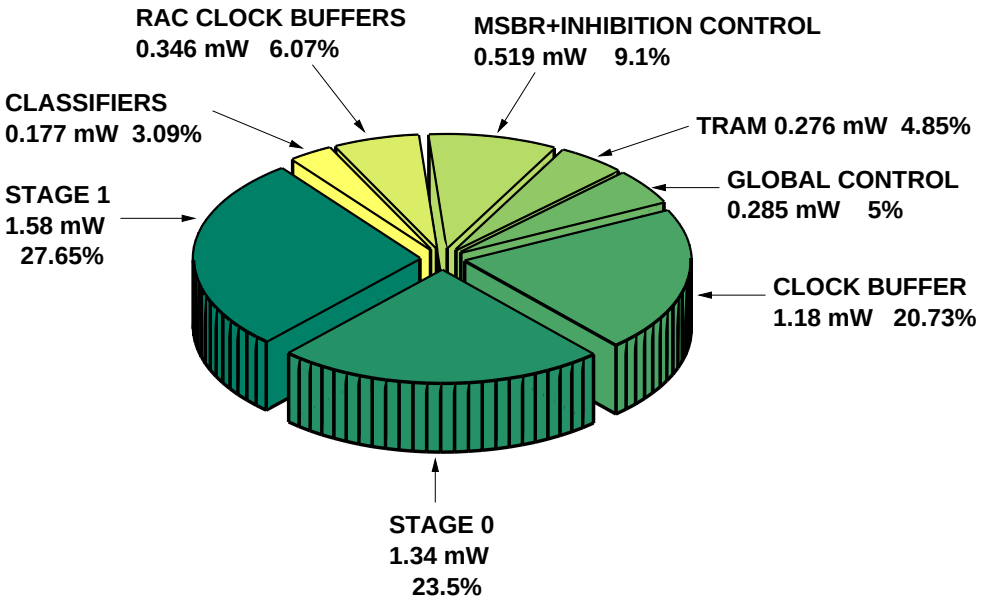


Figure 5.35: DCT Chip Power Estimation Results (No Interconnect)

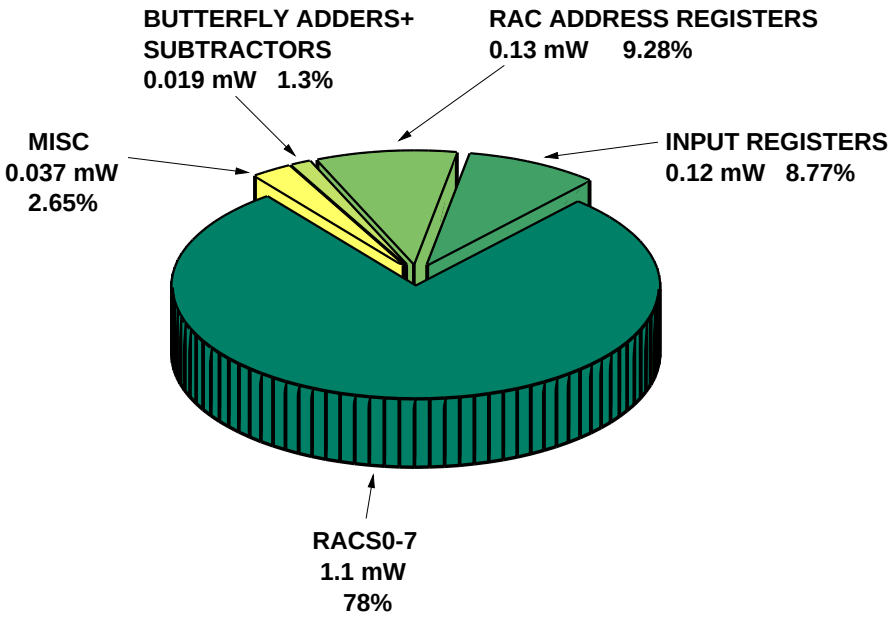


Figure 5.36: DCT Chip Stage 0 Power Estimation Results

5.5 Chip I/O Specification and Usage

The DCT chip is very easy to use in a PC board environment. It can be directly interfaced with 5V CMOS and TTL parts because its I/O pads are level converting. Three supply voltages are required: A ground connection (GND), a 5V pad supply connection (VHH) and a 1.2-3V core connection (VDD).

5.5.1 JTAG Programming

Before normal operation begins, the user needs to program a total of 6 internal chip JTAG registers. These registers are summarized in Table 5.7.

Access to the internal data registers is enabled after shifting in the address shown in the third column of Table 5.7 into the JTAG instruction register LSB-first. The instruction register (IR) consists of 10-bits and their significance is summarized in Table 5.8. The on-chip TAP controller enables the user to access the IR or the data register (Table 5.7) whose address is currently loaded in the IR through specified sequences of the TCK, TDI, and TMS pins. These sequences are standardized and documented in great detail in IEEE Standard 1149.1-1990 [IEE90b].

Figure 5.37 shows the bit field arrangement of both threshold data registers. Data is shifted serially LSB-first through the TDI pin. All of our experimental results have been obtained using the thresholds of Table 5.9. These values result in approximately equally populated classes as shown in Figure 5.6 for our 11 test images.

Name	Length	IR Address	Description
THRESHOLDS0	24	0x004	Stage 0 Classification Thresholds (Three 8-bit unsigned integers)
THRESHOLDS1	33	0x020	Stage 1 Classification Thresholds (Three 11-bit 2's complement integers)
CYCLESTOP0	128	0x008	Cycle Limits for Stage 0 RACS0-3 (16 8-bit clock masks)
CYCLESTOP1	128	0x040	Cycle Limits for Stage 1 RACS0-3 (16 8-bit clock masks)
CYCLESBOT0	128	0x010	Cycle Limits for Stage 0 RACS4-7 (16 8-bit clock masks)
CYCLESBOT1	128	0x080	Cycle Limits for Stage 1 RACS4-7 (16 8-bit clock masks)

Table 5.7
DCT Chip JTAG Data Registers

Bit	Description
0	Must always be at zero
1	Serial Compressed Output Enable (1 active, 0 inactive)
2	Data Register THRESHOLDS0
3	Data Register CYCLESTOP0
4	Data Register CYCLESBOT0
5	Data Register THRESHOLDS1
6	Data Register CYCLESTOP1
7	Data Register CYCLESBOT1
8	Reserved
9	Reserved

Table 5.8
DCT Chip JTAG Instruction Register Fields

Stage 0	T0	6
	T1	15
	T2	37
Stage 1	T0	5
	T1	12
	T2	29

Table 5.9
Class Thresholds Used

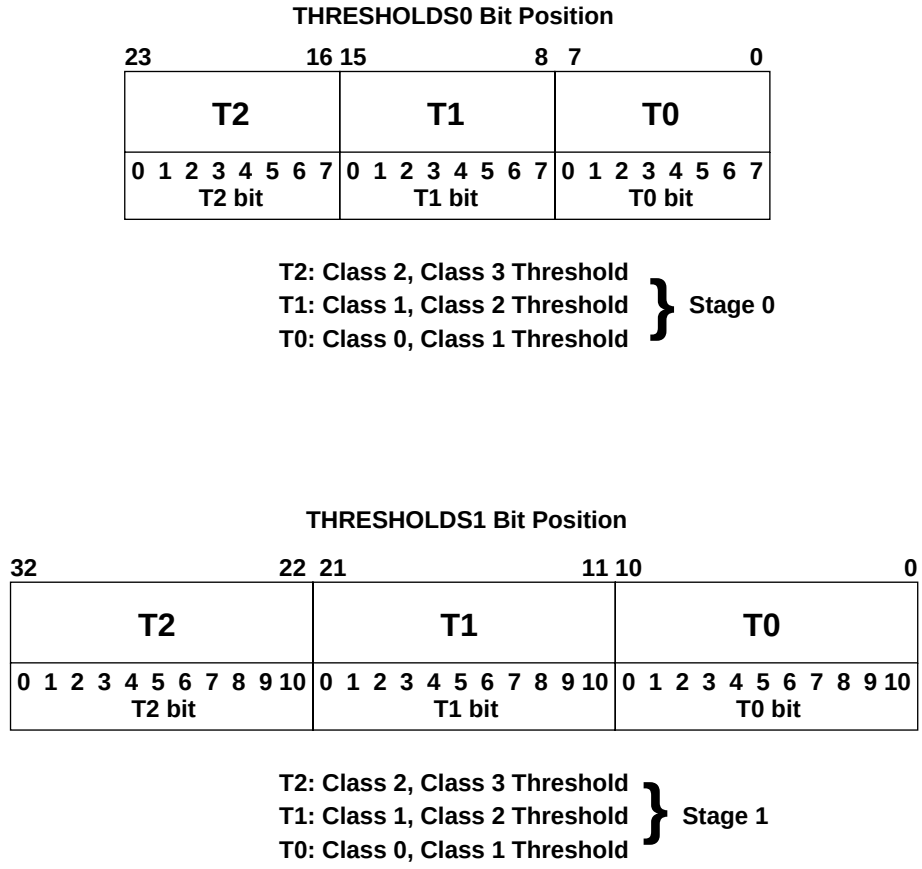


Figure 5.37: JTAG Threshold Registers

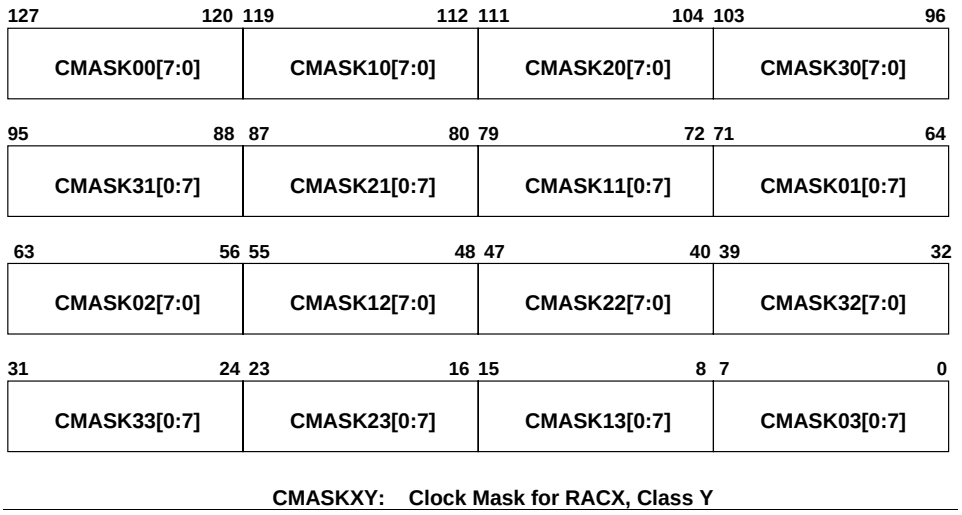


Figure 5.38: CYCLEREGTOP{0,1} Register Bit Fields

Figures 5.38 and 5.39 show the various fields of the CYCLESTOP and CYCLESBOT registers of both stages. Each CMASK field contains an 8-bit register clock mask that indicates the maximum number of cycles that the corresponding RAC will be clocked. The 9 allowable values that such an 8-bit mask may take are listed in Table 5.10. The behavior of the DCT chip is unspecified and undocumented for all other mask values.

5.5.2 Chip Regular Operation

After the 6 JTAG data registers have been programmed with appropriate class threshold and cycle limit values, chip normal operation can be initiated. The user provides the chip with image pel values on the rising edge of clock. The beginning of a new block is indicated

Mask Value	Chip Action
00000000	RAC is clocked a maximum of 0 cycles
10000000	RAC is clocked a maximum of 1 cycles
11000000	RAC is clocked a maximum of 2 cycles
11100000	RAC is clocked a maximum of 3 cycles
11110000	RAC is clocked a maximum of 4 cycles
11111000	RAC is clocked a maximum of 5 cycles
11111100	RAC is clocked a maximum of 6 cycles
11111110	RAC is clocked a maximum of 7 cycles
11111111	RAC is clocked a maximum of 8 cycles

Table 5.10
Clock Mask Values

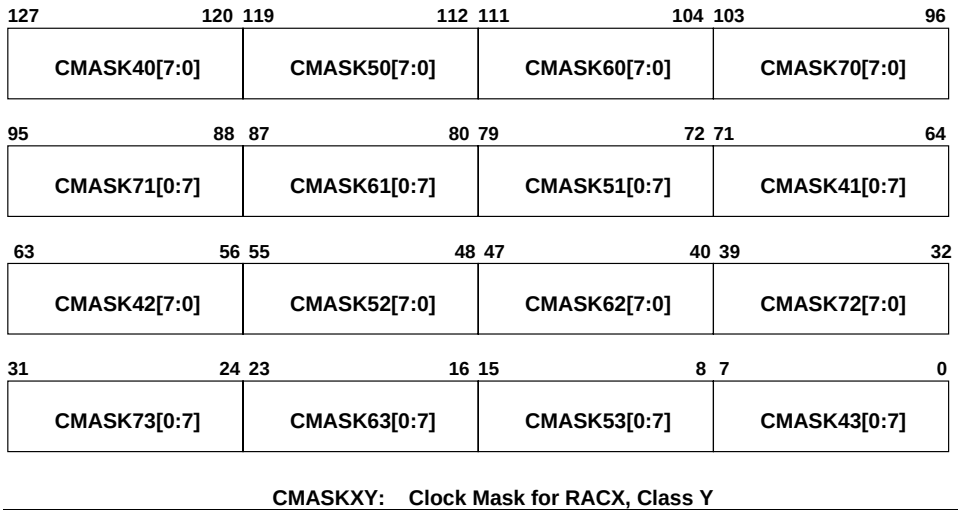


Figure 5.39: CYCLEREGBOT{0,1} Register Bit Fields

with the assertion of the StartBlockIn pulse when block element (0,0) is present on the input bus. After a delay of 97 cycles, the DCT coefficients start to come out of the chip output pins. The start of a block is indicated by the assertion of the StartBlockOut pulse when output block element (0,0) is present on the output bus. Figure 5.40 presents a timing diagram that summarizes the chip operation. We note that the output block comes out in transposed form. Figure 5.41 shows the sequence that block pels must be presented to the chip and the sequence that the DCT coefficients are being produced by the chip. The negative clock edge is used for sampling the input data and the StartBlockIn strobe on the chip so that the edges are sufficiently spread apart and there is sufficient setup and hold time if the data edge occurs close to the positive clock edge (in either direction.)

5.5.3 Serial Output

The DCT chip contains a serial output (LPEout) that produces compressed blocks at 1 bit per pixel (8:1 compression). The serial output is enabled when IR bit 1 is set to logic one. The timing of the LPEout output is identical to the dout[7:0] timing of Figure 5.40. Compression is performed by selecting 64 bits out of the 64-element compressed block according to bit allocation matrix of Table 5.11. Images of acceptable quality can be produced by reconstructing and dithering the DCT information provided by the serial output and computing the resulting IDCT using the available information.

5.6 Chip Physical Design and Packaging

The chip has been laid out using the Cadence Design Framework. Details about the design flow and methodology are presented in appendix A. It has been fabricated in the Analog

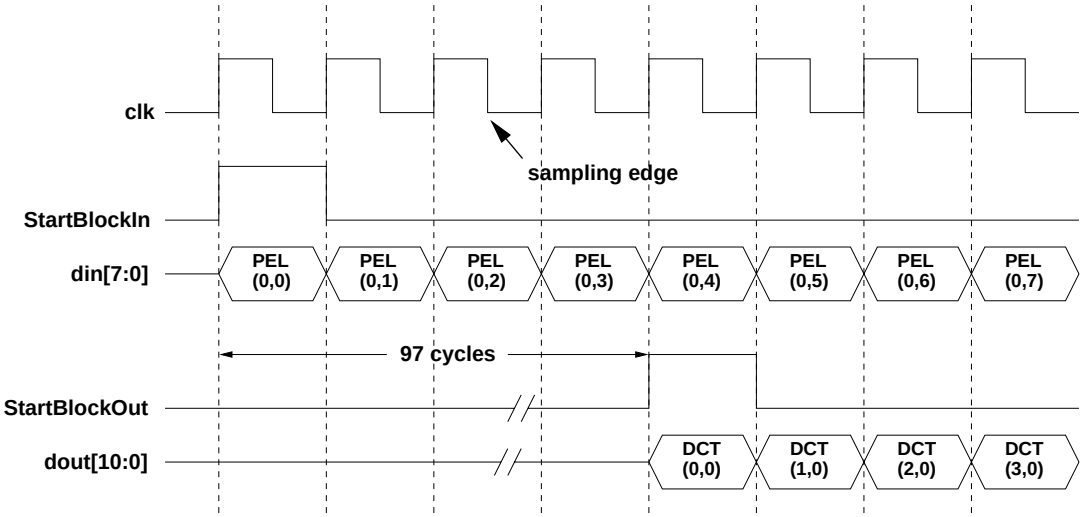


Figure 5.40: DCT Chip Timing Diagram

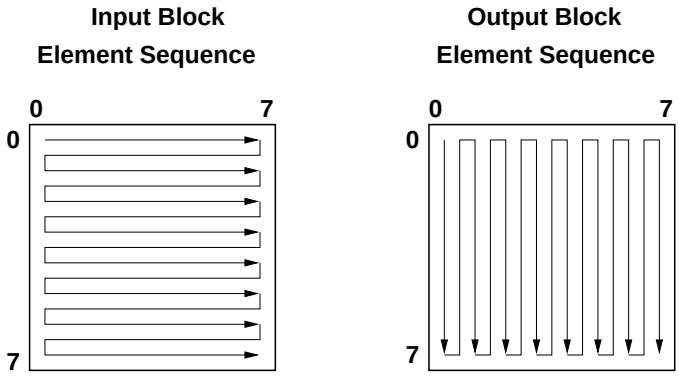


Figure 5.41: Input and Output Block Element Sequence

8	8	8	0	0	0	0	0
8	8	8	0	0	0	0	0
8	8	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 5.11
Bit Allocation per Coefficient Position for Producing the Serial Output

Process	0.6μm CMOS (0.6μm drawn), 3ML 5V
VTN	0.75V
VTP	-0.82V
TOX	14.8 nm
Supply	1.1-3 V
Frequency	2-43 MHz
Power	4.38 mWatts @ 1.56 V, 14 MHz
Area	14.5 mm^2
Transistors	120K

Table 5.12
Process and DCT Chip Specifications

Block	Area (mm^2)	Percentage
Stage 0	2	13.8
Stage 1	4.075	27.6
TRAM + wiring	1.62	11.2
MSBR Control	1	6.9
Classification	1.49	10.27
TAP Controller	0.4	2.75
VDD / GND routing, wiring, bypass capacitors etc.	3.9	27

Table 5.13
DCT Chip Area Breakdown

Devices (ADI) Dual-Poly Triple-Metal (DPTM) 0.6 μm 5V CMOS process. The annotated chip microphotograph is shown in Figure 5.42. The blocks labelled "MSB rejection" and "sign detection" in addition to the circuits of Figures 5.28 and 5.29 they include distributed control circuits and clock generators and buffers for the RAC units of both stages. Moreover, each contains a 128-bit JTAG register (visible as a dark rectangular box within each block) that stores cycle thresholds (four per class per block). These registers are all accessible through the IEEE 1149.1-1990 Test Access Port (TAP). The block labelled "JTAG" contains the IEEE 1149.1 Instruction Register and the finite state machine (TAP controller) which controls serial access to all JTAG-accessible registers. The core area of the DCT chip (not including the I/O pad frame) measures 14.5 mm^2 and includes approximately 120K transistors. The chip specifications are summarized in Table 5.12.

A chip area breakdown is shown in Table 5.13. The area penalty of the MSB rejection control is approximately 7% of the total chip area. This number includes buffering for all gated RAC clocks. The classification area penalty is shown to be higher at 10.27%. This high area percentage is entirely due to large JTAG registers required to store RAC cycle limits and class thresholds. Had a less convenient way been chosen to store this state (i.e. SRAM) the area penalty would have been much less.

The chip package is a Kyocera ceramic PGA (13x13 grid, 100-pin) available through Analog Devices. The chip pinout appears in Table C.2 (Appendix C). The VHH supply is a TTL-compatible 5V supply for the pads and VDD is the core supply at 1.2-3.0V. The

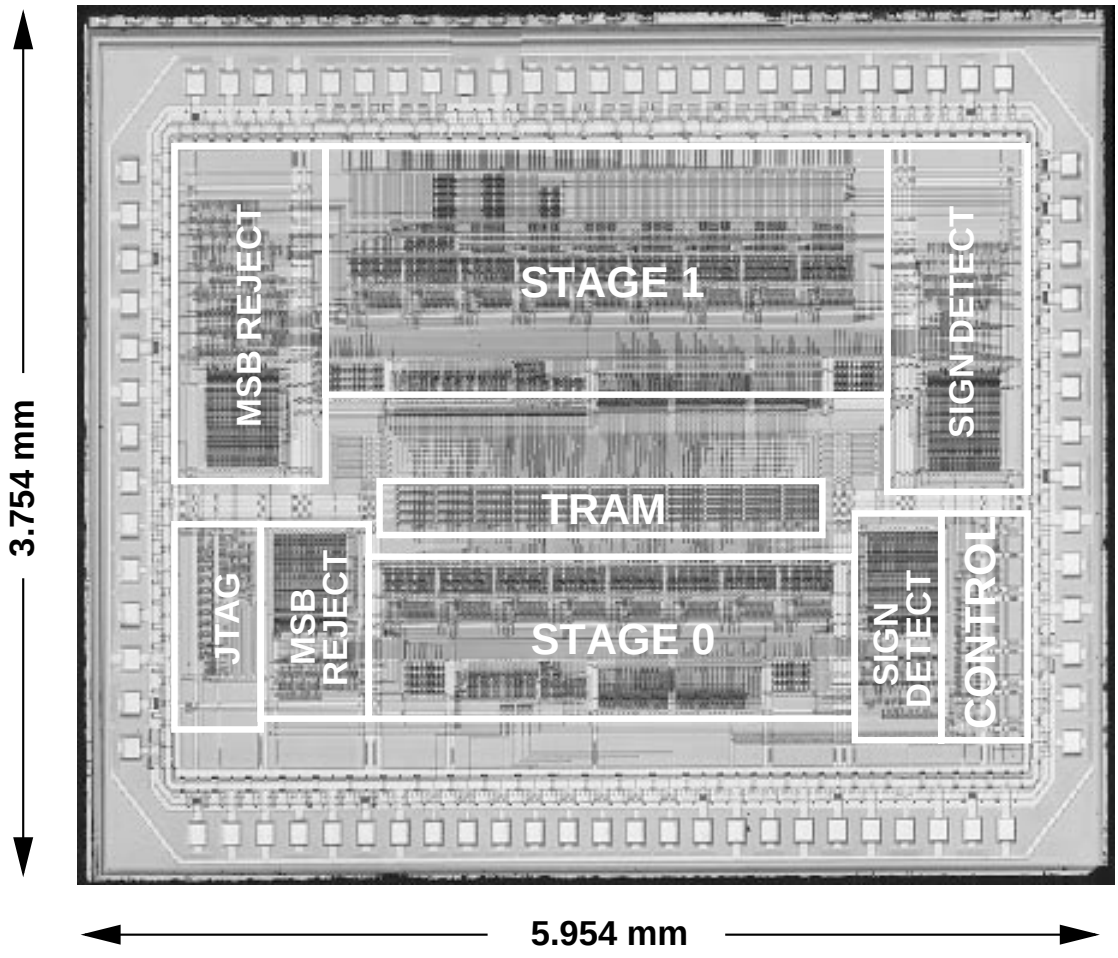


Figure 5.42: DCT Chip Microphotograph

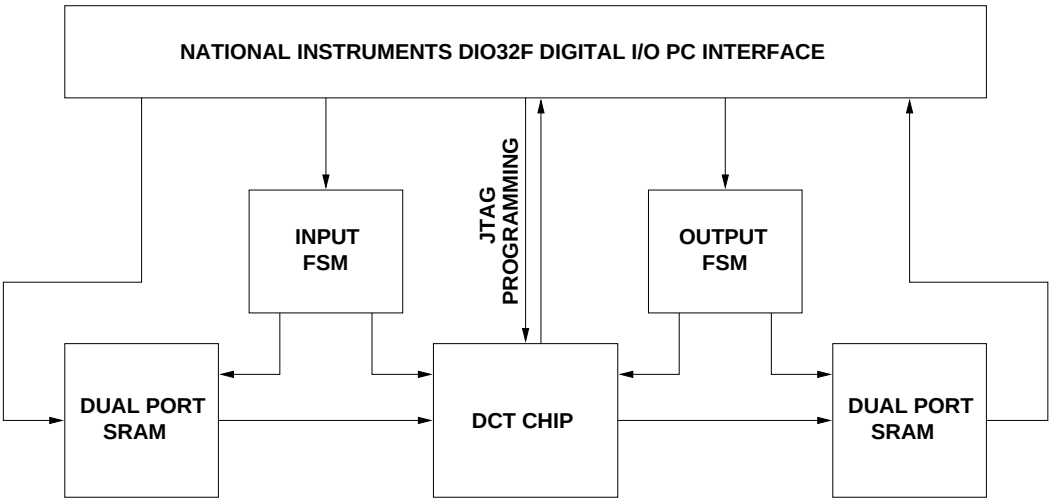


Figure 5.43: DCT Chip Test Board Block Diagram

package PCB footprint (top view) is shown in Figure C.2 (Appendix C).

5.7 Testing

5.7.1 Test Setup

A printed circuit board has been designed and built to test the DCT chip. The test structures are almost identical to the board built for the IDCT chip (section 3.7). The board block diagram is shown in Figure 5.43.

The board operates as follows: First, the user accesses the internal chip JTAG registers directly through the PC interface to shift in the class thresholds and the class cycle limits. Then, four 64-element blocks are uploaded into the input dual port memory (Cypress CY7C006) and the input FSM (implemented in a Lattice GAL 22v10) is triggered. The input FSM produces appropriate control signals to read the blocks from the SRAM and stimulate the DCT chip. The four blocks are cycled through the chip continuously so that power measurements can be obtained while the chip is being continuously stimulated. On the other hand, the output FSM captures the four output blocks in the output SRAM only once: The SRAM is activated only when the first four blocks are being output from the chip. It is deactivated for the remainder of the chip operation. The user can upload the results on the PC from the second SRAM and can check the chip output for correctness.

In addition to the test structures described above, the board contains two more sections: One section contains a DCT chip directly attached to a DIO32F connector so that the user has direct access to all chip terminals through the PC. This section has been used for preliminary slow-speed functionality-only testing. The final section simply contains a DCT chip with all relevant terminals brought out to vertical headers for last resort testing using

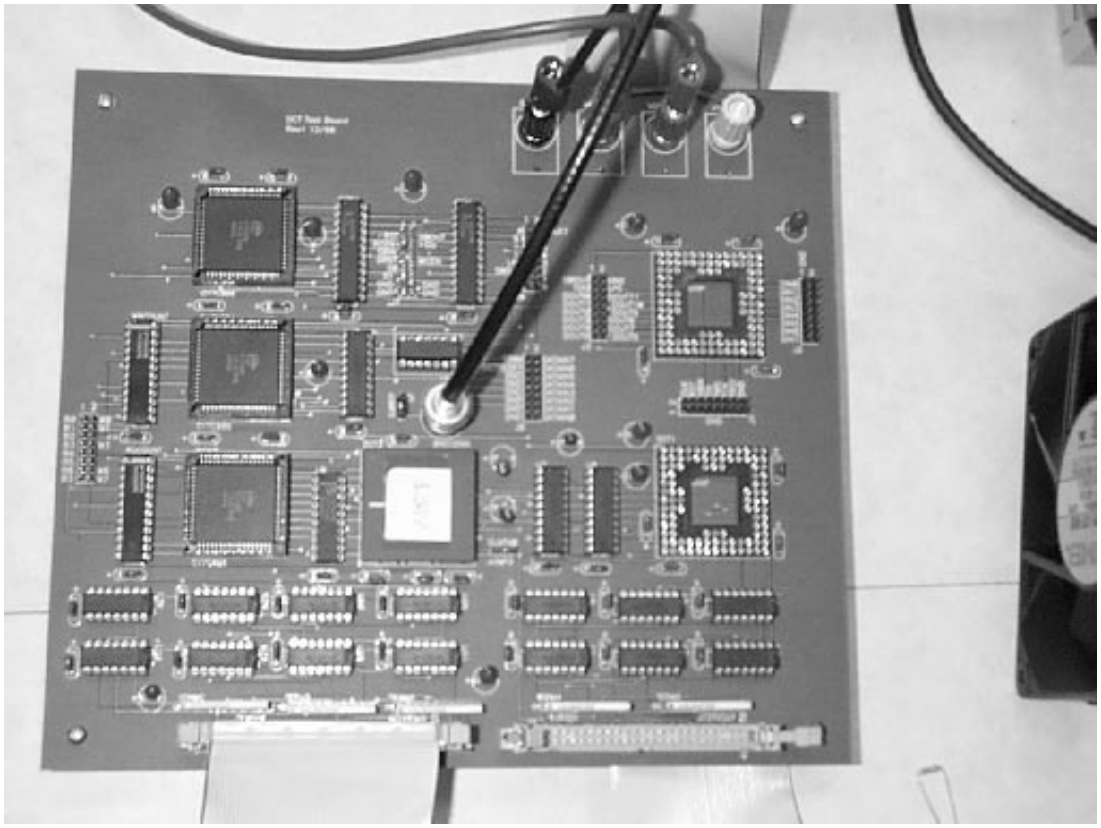


Figure 5.44: DCT Chip Test Board Photograph

pattern generators and logic analyzers. This section was never used. A photograph of the DCT test board is shown in Figure 5.44. The output of the DIO32F connectors has been terminated with 470-Ohm resistors to ground and has been passed through two inverting CMOS Schmitt triggers (74HC14) for additional conditioning.

The DCT test setup shown in Figure 5.45 consists of the DCT test board, a PC running the Linux operating system, a Keithley 2400 sourcemeter for providing the supply and measuring the current of the DCT chip core, a Tektronix HFS 9003 high-speed digital waveform generator for providing a variable clock to the board, and a Tektronix power supply for providing a TTL supply for the board. Both the Keithley 2400 and the Tektronix HFS 9003 have an RS-232 interface for remote operation. Both instruments are under software control through the PC so that supply and clock frequency can be changed through C program instructions and current measurements can be read from the instrument under program control.

Figure 5.45: DCT Chip Test Setup Photograph

5.7.2 Test Results

The DCT chip is functional over a wide range of frequencies and power supplies as can be seen from the Schmoo plot of Figure 5.46. Over 150,000 separate power measurements on a block-by-block basis have been taken using the test images of Figure 5.8 in addition to artificially generated data in order to fully characterize the chip power dissipation.

We begin by presenting the power dissipation results for each one of the 11 test images for three sets of cycle limit thresholds. The power results are plotted in Figure 5.47. The measurements have been taken at 1.56V, 14 MHz. The first set of bars are power measurements for maximum cycle limits (8) per RAC unit (no computation inhibition). This is the setting for maximum PSNR and power. The second and third set of bars are for the cycle limits of Table 5.2 and the cycle limits computed by our optimization program yielding the PSNRs of Figure 5.15 respectively. As expected, more power is dissipated when the cycle limits are at their maximum value since the chip executes a larger number of operations to compute the DCT using the highest possible precision. We also observe that our initial intelligent guess dissipates less power than the cycle limits computed by our optimization program although Figure 5.15 implies that both sets of cycle limits result in equal cycles per RAC. The explanation for this effect is a bit involved: The cycle limits of Table 5.2 have zeroes that are concentrated together in the high spatial frequency area. As a result, the first stage produces intermediate blocks that contain columns with a number of zero-valued elements. The second stage MSB rejection and sign extension mechanisms therefore will see correlated data and reduce the effective bitwidth of the number that the RAC units operate on. As a result, the reduced cycles per RAC are a result of increased

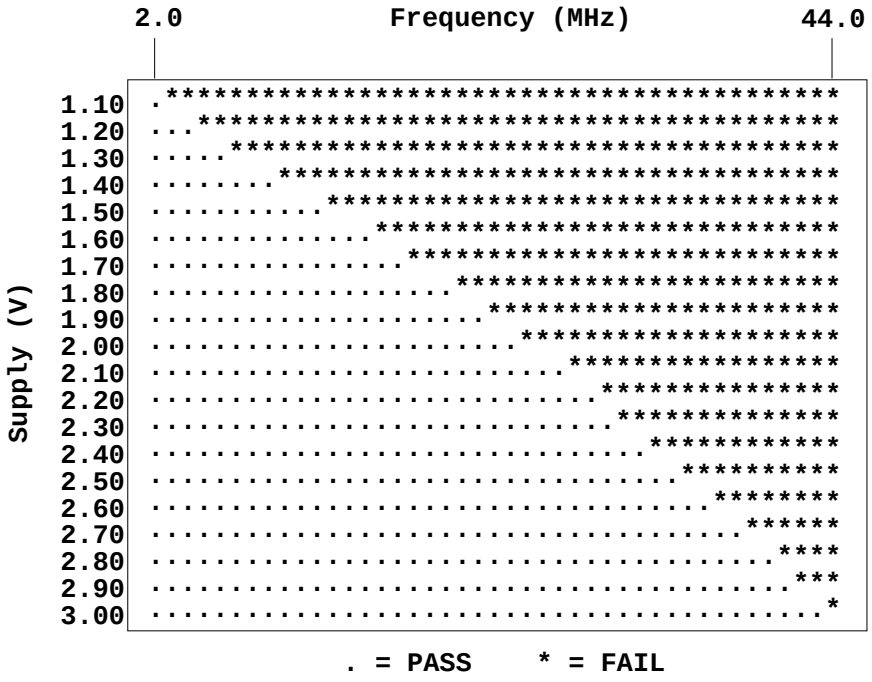


Figure 5.46: DCT Chip Schmoo Plot

MSB and sign extension detection in addition to computation inhibition. On the other hand, the cycle limits that have been computed by the optimization program do not contain so many zeroes close together, but the reduction in the cycle limit numbers is spread more evenly. As a result, the reduced cycles are due to computation inhibition rather than increased MSB / sign rejection. We ask the reader to recall from section 5.1.2 that stopping a RAC before it reaches its final value introduces scaling by a factor 2^{-n} where n is the number of remaining cycles required to complete the computation using full precision. In order to account for this scaling, the shift-only feedback path of Figure 5.2 must be activated for the n remaining cycles. This shift is not necessary in the MSB rejection case and it is this additional clocking that accounts for the increased power in the third set of bars in Figure 5.47.

The average power dissipation for all 11 images is shown in Table 5.14. We quote 4.38 mW for the chip power dissipation at 1.56V, 14 MHz because at this supply and clock frequency the chip can produce compressed images of high quality and at a rate high enough for MPEG2 640x480 4:2:0 video compression.

A final observation from Figure 5.47 is that the more "busy" images FLOWER, MANDRILL and VAN GOGH produce the highest power dissipation. This is expected because according to Table 5.3 these three images require the most cycles per RAC unit. This is attributed to reduced MSB and sign extension rejection due to reduced correlation among pel values. On the other hand, image WATER which consists of mostly blue pels of constant intensity demands the least power from the DCT processor.

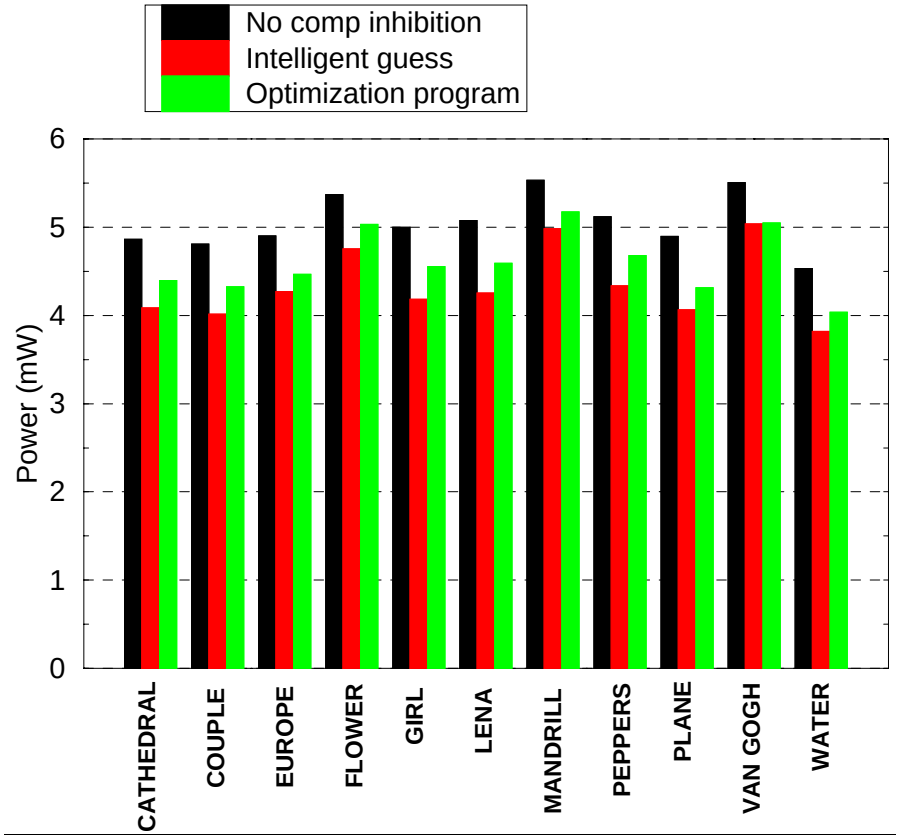


Figure 5.47: Power Dissipation of 11 Test Images at 1.56V, 14 MHz

	Power (mW)
No computation Inhibition	5.08
Intelligent Guess	4.38
Optimization Program	4.63

Table 5.14
Average Power Dissipation for All Test Images

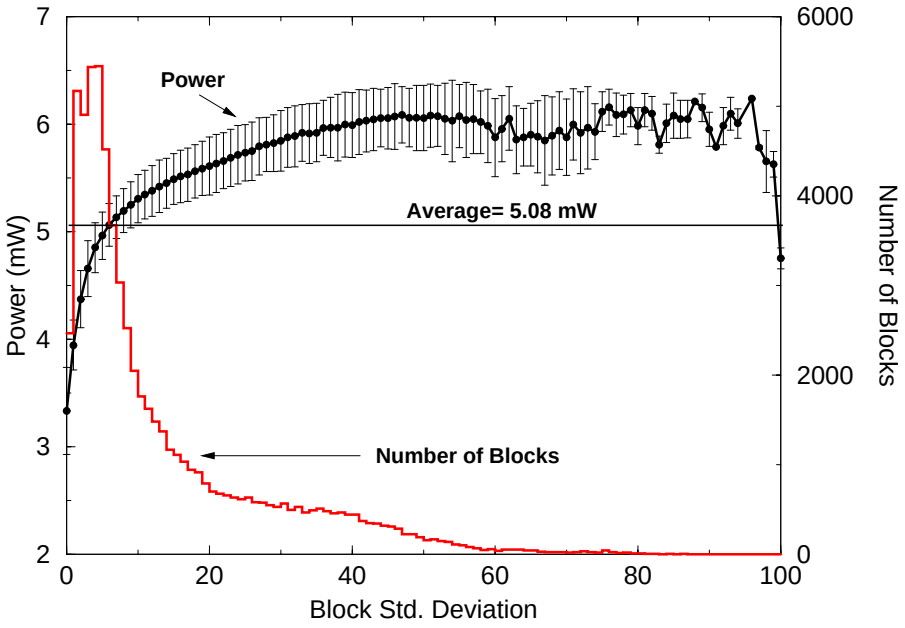


Figure 5.48: DCT Chip Power vs. Pel Standard Deviation. Chip Measured Power at 1.56V, 14 MHz. The cycle limits used are set at the maximum possible value (computation performed to full precision). This plot includes all blocks of all 11 test images.

We wish to investigate more the relationship between image pel correlation and power dissipation. The results of our power measurements (average $\pm$ 1 standard deviation) on a 64-element block basis are plotted in Figures 5.48 and 5.49 vs. the sample standard deviation of the 64 block elements. The image block frequency histogram vs. standard deviation is also shown on the same plot. We observe that according to the design goal, power dissipation shows strong dependence with data correlation (more MSB rejection and less arithmetic activity). To stress this dependence even more, we have measured the chip power while operating on blocks that have zero variance and consist of 64 occurrences of the same 8-bit value (0-255). The results of these measurements are plotted in Figure 5.50 vs. the block value. We observe that the average power dissipated is much lower (2.86 mW and 2.91 mW with and without computation inhibition respectively) than the average power we obtain when the chip is stimulated with image data (4.38 mW). Moreover, we observe that the power dissipation does not depend on the actual block value, since hardly any computation is performed and the power is mainly due to control, clock distribution and the TRAM.

Figure 5.51 compares the average power of the DCT chip for different types of stimuli ranging from fully correlated (constant) blocks to random blocks with maximum variance. The “IMAGE BLOCKS” bars are power averages when the chip is stimulated from all 11 test images. We observe that MSB rejection itself can be responsible for up to 55% power savings with 22% being more typical for image data. The savings figure will be increased for differential video data (i.e. MPEG) due to increased correlation (more zeroes and small

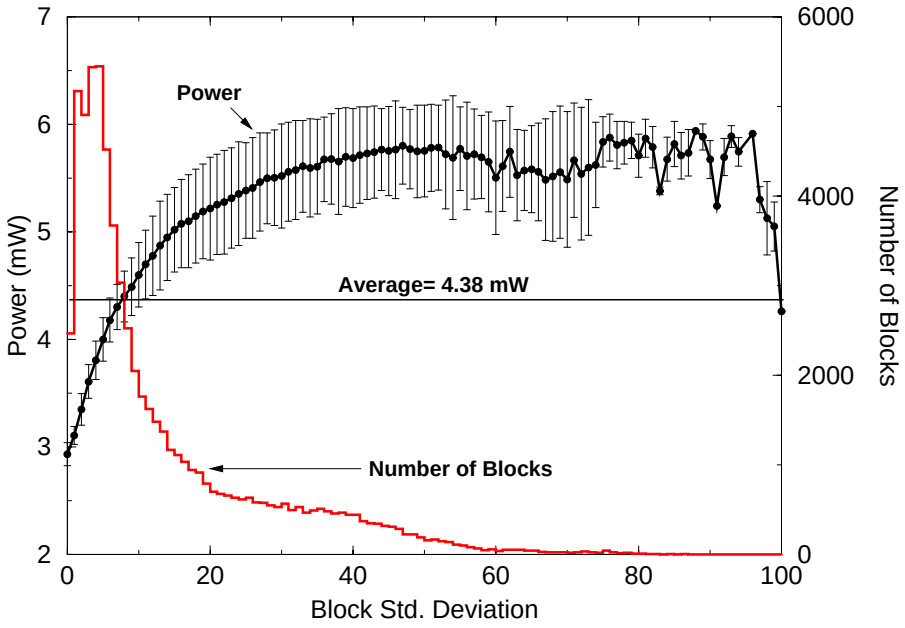


Figure 5.49: DCT Chip Power vs. Pel Standard Deviation. Chip Measured Power at 1.56V, 14 MHz. The cycle limits used are shown in Table 5.2. This plot includes all blocks of all 11 test images.

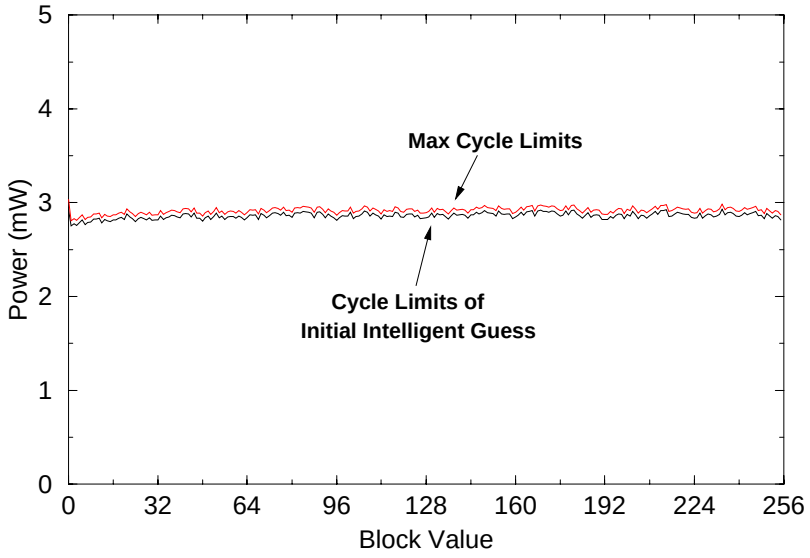


Figure 5.50: DCT Chip Power vs. Block Element Value. Chip Measured Power at 1.56V, 14 MHz. This plot contains measurements for blocks where all 64 elements have the same value.

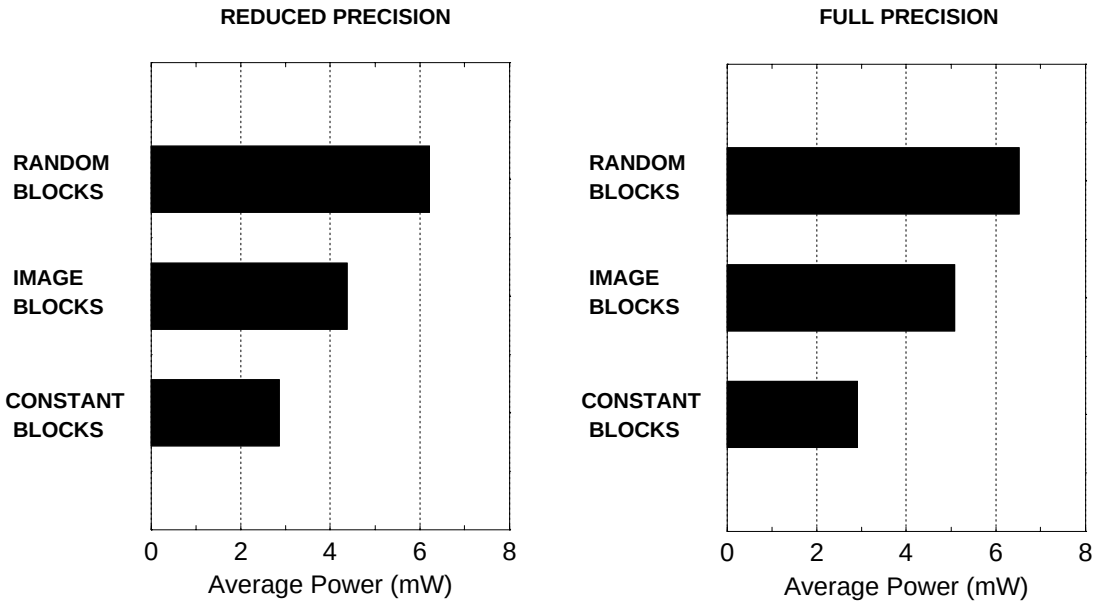


Figure 5.51: DCT Chip Average Power Comparison for Different Stimuli. Chip Measured Power at 1.56V, 14 MHz.

magnitude numbers present.) Our power estimation results (section 5.4) indicate that the power cost of the additional control logic required to implement MSB rejection is approximately 3-4% of the chip total. Therefore, MSB rejection produces net power gains on the order of 18-19% for typical image data and up to 52% for fully correlated data.

In addition to establishing the strong dependence between chip power and block sample correlation, we also wish to establish the obvious relationship between chip power dissipation and the average number each RAC unit must be clocked per block row/column for the DCT computation (computation bitwidth.) We remind the reader that each RAC is clocked a maximum of 8 times except for RAC0 of stage 0 which is always clocked 8 times and RAC0 of stage 1 which is always clocked 11 times. Figure 5.52 plots our measurement data (averages only) vs. the computation bitwidth along with the corresponding average block PSNR. Plots (a) and (c) show power and PSNR vs. bitwidth with row-column classification disabled (no premature computation inhibition is performed.) We observe that PSNR stays constant at well above 40 dB while power increases with bitwidth as expected. Plots (b) and (d) show power and PSNR in the presence of reduced precision selected according to row-column maximum absolute element difference. We observe that PSNR drops as the average bitwidth increases due to the approximation performed by the RAC units. This approximation is more coarse for higher bitwidth inputs and therefore PSNR drops with increased bitwidth.

Finally, we wish to establish the relationship between power and quality of image coding. We used the PEPPERS image PSNR trace (Figure 5.13) and we picked 11 sets of cycle limits than span the entire PSNR range for this image (23.99 dB to 44.84 dB). We determined

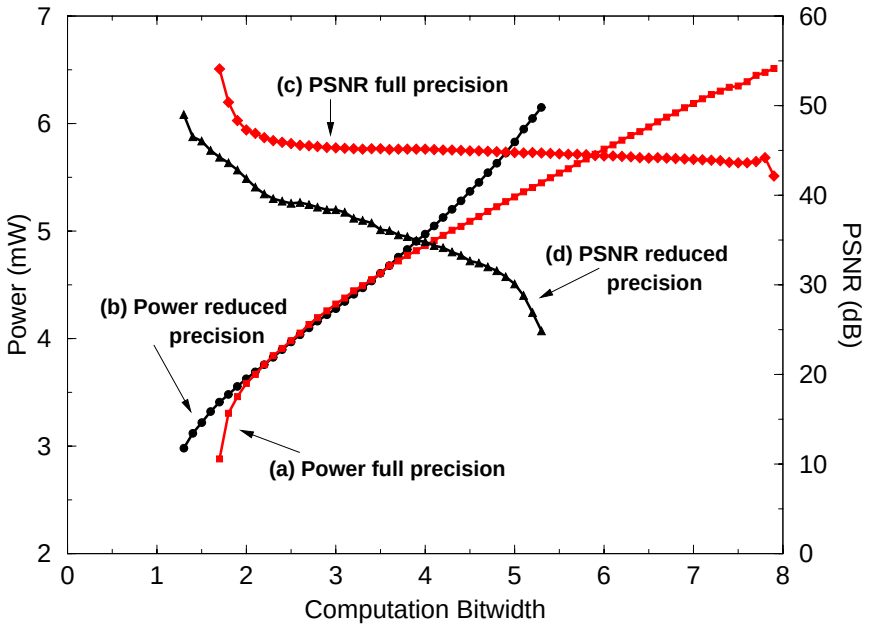


Figure 5.52: DCT Chip Power and Block PSNR vs. Computation Bitwidth. Chip Measured Power at 1.56V, 14 MHz.

the specific cycle limits from the contour plots of Figure 5.13. We measured the power that image PEPPERS dissipates for each one of the 11 limit sets and plotted the results vs. PSNR in Figure 5.53.

Figure 5.54 displays the actual compressed images for each (power, PSNR) datapoint of Figure 5.53. Figures 5.53 and 5.54 establish our claim that the present chip trades-off image quality and power dissipation.

Finally, we wish to compare the experimental power results with the ones obtained from Pythia to validate the estimation results of section 5.4. Figure 5.55 plots experimental measurements with Pythia estimated power for the first 12 blocks of test image CATHEDRAL. Pythia results with and without interconnect estimation are shown. Matching is remarkably good and we claim that the estimated results of section 5.4 are rather accurate.

We can summarize our conclusions from the experimental results as follows: We have observed that power savings due to MSBR can be as high as 55% (random blocks vs. fully correlated blocks) with 22% being more typical for still images. The additional power cost of MSBR is 3-4% of the total chip power. Differential video will result in higher savings due to the increased presence of zero-valued data. RCC adds an additional 15% of power savings for minimal PSNR degradation at the expense of 5-6% more power in additional control logic (section 5.4). Much higher power savings can be achieved if we are willing to tolerate more image degradation. Both MSBR and and RCC account for about 40% of power savings for still images at an additional combined overhead of 10% for a net power reduction of 30%. We expect higher power savings for differential video coding.

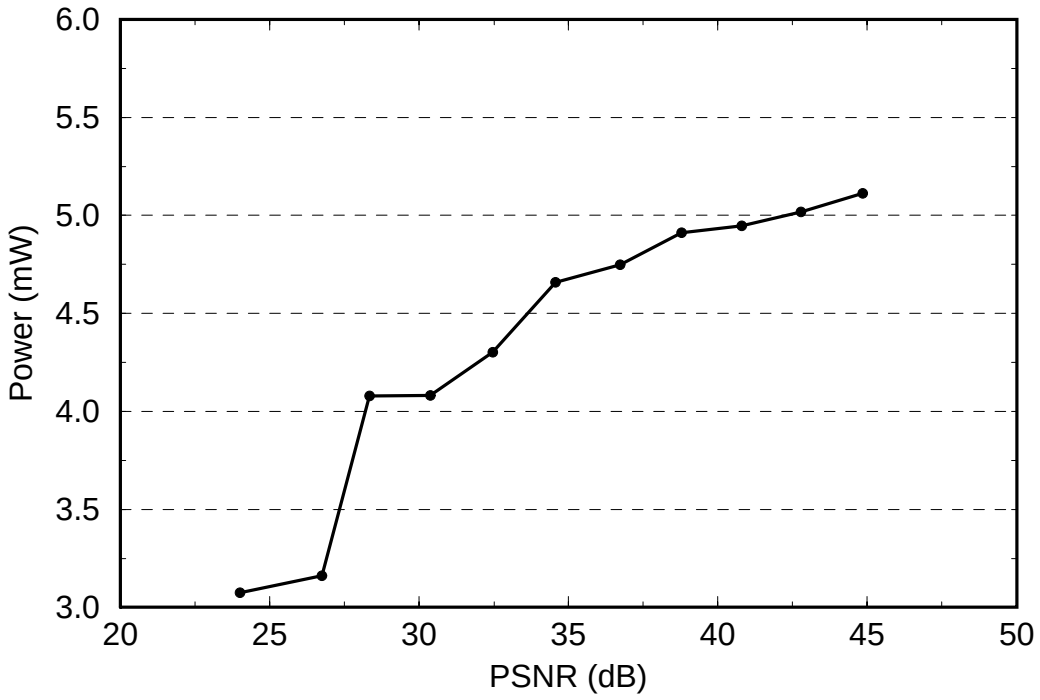


Figure 5.53: DCT Chip Average Power vs. Compressed Image Quality. Chip Measured Power at 1.56V, 14 MHz. The power measurements are for test image PEPPERS.

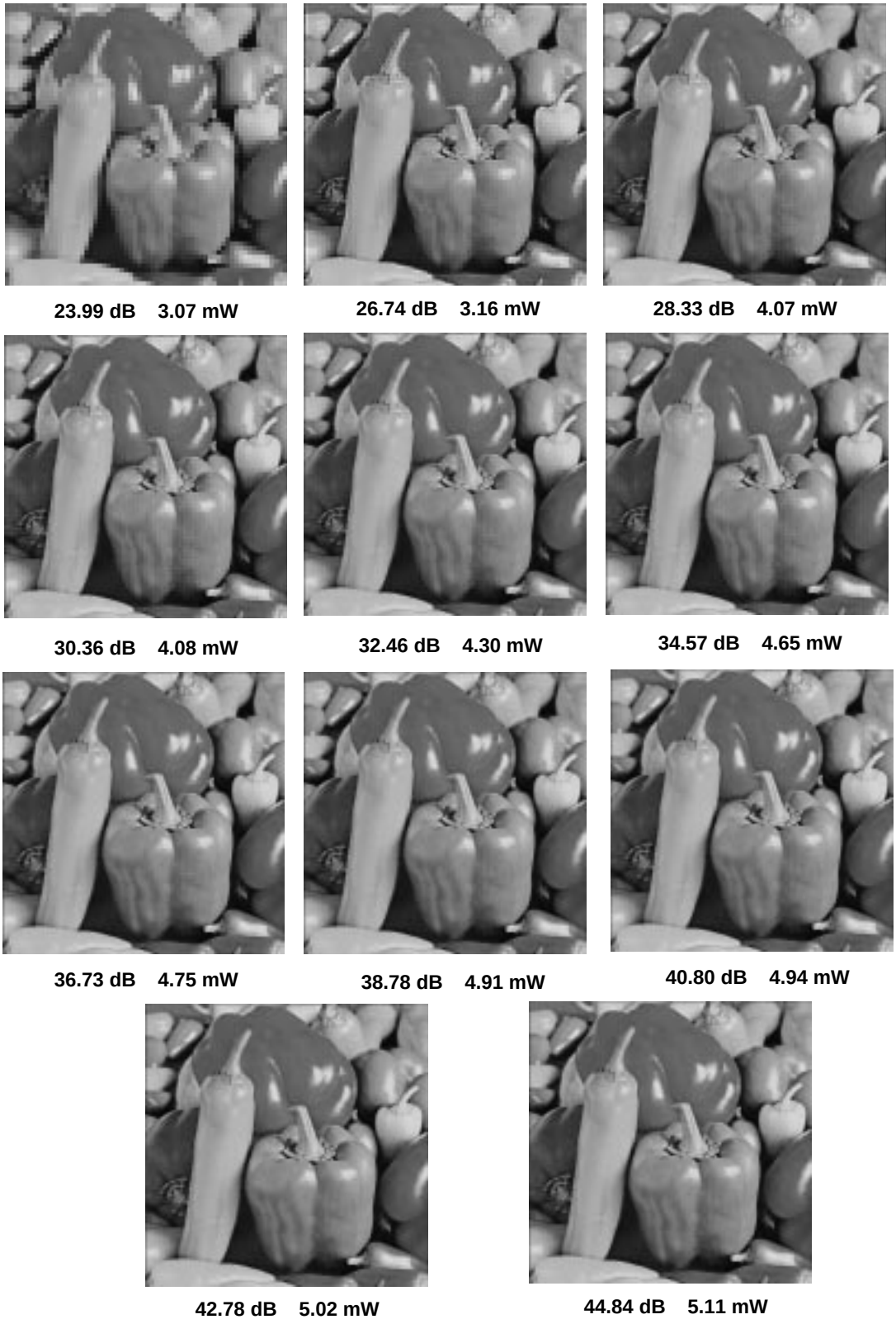


Figure 5.54: Compressed Image Quality and Power. The displayed images constitute the data points of Figure 5.53.

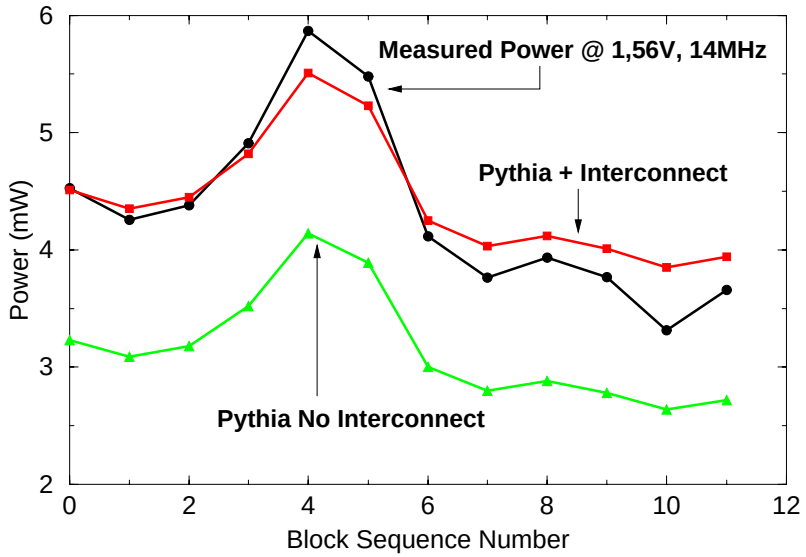


Figure 5.55: DCT Chip Measured vs. Estimated Power

Chip	Sw-Cap/sample
Matsui et al. [MHS+94]	375 pF
Bhattacharya et al. [BH95]	479 pF
Kuroda et al. [KFN+96] (scaled to same feature size)	417 pF
IDCT chip (chapter 3)	190 pF
Present DCT chip	128 pF

Table 5.15
Energy Efficiency Comparison Among DCT/IDCT Chips

The DCT chip exhibits less switched capacitance per operation than the IDCT chip of chapter 3 (128 pF vs. 190 pF) and is much more energy efficient than similar chips that have appeared in the literature from a switched capacitance perspective. In Table 5.15 we reproduce the switched capacitance data that has appeared in Table 3.9 along with an extra entry for the present DCT chip. We observe that the present chip exhibits less switched capacitance by a factor of three when compared to past DCT processors. The activity reduction methods account for a significant portion of this reduction. The remaining savings are attributed to optimised internal arithmetic bitwidths, additional clock gating and adder/subtractor input shielding in the "butterfly" stages. Additional energy savings are reaped from reduced on-chip latency. Chips [MHS+94] and [KFN+96] impose 112 cycles of latency from chip input to chip output. The latency of the present DCT chip is 97 cycles.

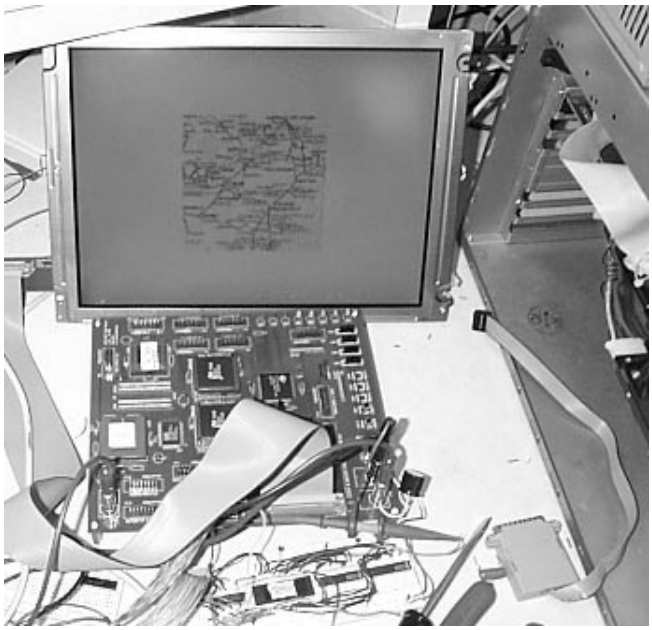


Figure 5.56: DCT / IDCT Demonstration Board

5.8 DCT/ IDCT Demonstration Board

Rex Min [Min99] has built a board that demonstrates the functionality of the DCT and IDCT chips working together in a still image coding / decoding environment. The board contains a DCT and IDCT chip directly connected to each other and supporting circuitry that stimulates the DCT chip with still images and displays the output of the IDCT chip on a grayscale LCD display. A photograph of the demonstration board is shown in Figure 5.56.

A new demonstration board is being designed which will be capable of digitizing NTSC motion video, code it through the DCT chip, decode it through the IDCT chip and display it on the LCD screen at 30 frames per second.

5.9 Chapter Summary and Conclusion

This chapter has presented the design, implementation and testing of the DCT core processor. The chip is targetted to low power video (MPEG2 MP@ML) and still image (JPEG) applications. It exhibits two innovative techniques for arithmetic operation reduction in the DCT computation context along with standard voltage scaling techniques such as pipelining and parallelism. The first method reduces the bitwidth of arithmetic operations in the presence of data spatial correlation. The second method trades off power dissipation and image compression quality (arithmetic precision.) The chip dissipates 4.38 mW at 14 MHz, 1.56V.

Chapter 6

Conclusion

The contributions of this work can be classified in three categories:

- Algorithmic Contributions
- Architecture / Circuit Contributions
- System-level Contributions

6.1 Algorithmic Contributions

This work has introduced the idea of designing ad hoc DSP algorithms whose number of arithmetic operations depends on some statistical property of the input data. Such algorithmic design can result in a small number of operations for the average case (but not for the worst case) and lead to very low power implementations.

A data-dependent algorithm for the computation of the IDCT has been introduced. This algorithm requires a variable number of additions and multiplications per block depending on the number of zeroes present in the input data stream. When operating on MPEG-compressed video data, the data-dependent algorithm requires a smaller average number of operations per block than previously published fast algorithms. A similar forward-mapped IDCT algorithm has been proposed by McMillan and Westover [MW92] and has been the basis for our own implementation. As opposed to the McMillan-Westover FMIDCT, the present algorithm is a row-column implementation and is more suitable for low-area VLSI implementations.

This work has also proposed an algorithm for the computation of the DCT. The Chen [CSF77] algorithm has been the basis for this implementation. We have made the simple observation that for the DCT computation (and other frequency-domain transforms such as the DFT), the DC offset of the input data is only relevant for the computation of the DC coefficient and does not alter the higher spectral coefficients. This can reduce substantially

the input bitwidth in the presence of correlated inputs and result in a considerable reduction in the number of arithmetic operations without loss in precision. An additional extension is an adaptive method for further operation reduction with minimal visual image quality degradation. There has been a substantial body of work in adaptive video coding using classification based on post DCT spectral information (chapter 4). Most previous investigators suggest variable quantization as the adaptation parameter. On the contrary, the present work demonstrates the use of a pre-DCT classifier (row / column peak-to-peak pel amplitude) to reduce arithmetic operations during the actual DCT computation.

6.2 Architectural/ Circuit Contributions

The VLSI implementation of the IDCT chip has demonstrated that clock gating can be taken to the limits. Clock gating *can* be implemented in a pipelined data path by having separate qualified clock nets for each pipeline stage. Skew problems can be dealt with at a small cost in terms of power and practically zero hit in performance.

The VLSI implementation of the DCT chip has demonstrated the suitability of distributed arithmetic structures for low power. Past researchers have attempted to design arithmetic structures that can adapt to the dynamic range of the input data: Nielsen and Sparso [NS96] [NS98] have proposed a sliced data path for a digital hearing aid filter bank that exploits small magnitude arithmetic inputs for low power. The arithmetic data path has been partitioned in an MSB and an LSB slice. The MSB slice is only enabled when the input bitwidth requires it. Activation of the slices is performed by using special data tags that indicate the presence of sign extension bits in the MSB input slice. Additional circuit overhead is required for the computation and update of tags. Moreover, dynamic bitwidth adaptation is very coarse and can only be performed on a slice basis. On the other hand, the distributed arithmetic structures used in the DCT chip can reject sign extension bits in a very elegant fashion using minimal control overhead and at a very fine granularity (chapter 5).

An additional desirable property of distributed arithmetic structures is based on the successive approximation property (section 2.2.1). Recently, a number of researchers have resorted to approximate processing as a method for reducing average system power: Ludwig et. al [LNC96] have demonstrated an approximate filtering technique which dynamically reduces the filter order based on the input data characteristics. More specifically, the number of taps of a frequency-selective FIR filter is dynamically varied based on the estimated stopband energy of the input signal. The resulting stopband energy of the output signal is always kept under a predefined threshold. This technique results in power savings of a factor of 6 for speech inputs. Larsson and Nikol [LN97] [NLAO97] have demonstrated an an adaptive scheme for dynamically reducing the input amplitude of a Booth-encoded multiplier to the lowest acceptable precision level in an adaptive digital equalizer. Their scheme simply involves an arithmetic shift (multiplication / division by a power of 2) of the multiplier input depending on the value of the error at the equalizer output. They report power savings of 20%.

The ROM and accumulator arithmetic units can be used in an approximate processing framework with minimum extra hardware control overhead. Simply stopping the clock of a RAC before it completes its dot product computation produces an approximation of its final result with almost linear power savings. The DCT chip has demonstrated this property by producing approximate results for non-visually significant spectral coefficients and for low activity input data.

6.3 System-level Contributions

One of the author's main motivations to embark on this research is the strong belief that the next step in computer architecture is billion-transistor chips which will substantially deviate from the general purpose microprocessors of our times. Such complex chips will contain special-purpose macrocells for common tasks (i.e. DCT/IDCT) and large complex macroinstructions in addition to standard general purpose ALUs and instruction sets. Migration of common computation tasks from software to specialized hardware will lower substantially the average chip power dissipation and will enable portability and higher levels of integration. This work has provided the future system-level integrator with two such low power macrocells for two very frequently used tasks.

Bibliography

[AAN88] Y. Arai, T. Agui, and M. Nakajima. A fast DCT-SQ scheme for images. *Transactions of the IEICE*, E71(11):1095–1097, November 1988.

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, January 1974.

[BH95] A. K. Bhattacharya and S. S. Haider. A VLSI implementation of the inverse discrete cosine transform. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(2):303–314, 1995.

[Bur94] T. Burd. Low-power CMOS library design methodology. Master's thesis, UC Berkeley, 1994.

[CBB94] A.P. Chandrakasan, A. Burstein, and R.W. Brodersen. A low-power chipset for a portable multimedia I/O terminal. *IEEE Journal of Solid State Circuits*, 29(12):1415–1428, December 1994.

[Cho96] P.L. Chou. Low power ROM generation. Master's thesis, Massachusetts Institute of Technology, 1996.

[CL92] N. Cho and S. Lee. A fast 4x4 DCT algorithm for the recursive 2-D DCT. *IEEE Transactions on Signal Processing*, 40(9):2166–2172, September 1992.

[CP84] W. H. Chen and W. Pratt. Scene adaptive coder. *IEEE Transactions on Communications*, COM-32(3):225–232, March 1984.

[CS77] W. H. Chen and H. Smith. Adaptive coding of monochrome and color images. *IEEE Transactions on Communications*, COM-25(11):1285–1292, November 1977.

[CSB92] A.P. Chandrakasan, C. Sheng, and R.W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid State Circuits*, 27(4):473–484, April 1992.

[CSF77] W. H. Chen, C. H. Smith, and S.C. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, COM-25(9), September 1977.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. J. Wiley, 1991.

[Dan96] A.P. Dancy. Power supplies for ultra low power applications. Master's thesis, Massachusetts Institute of Technology, 1996.

[Dav72] L.D. Davisson. Rate-distortion theory and application. *Proceedings of the IEEE*, 60(2):800–808, July 1972.

[Den95] L. Dennison. *The Reliable Router : An architecture for fault tolerant interconnect*. PhD thesis, Massachusetts Institute of Technology, 1995.

[FW92] E. Feig and S. Winograd. Fast algorithms for the discrete cosine transform. *IEEE Transactions on Signal Processing*, 40(9):2174–2193, September 1992.

[Gea96] J. Gealow. *An Integrated Computing Structure for Pixel-Parallel Image Processing*. PhD thesis, Massachusetts Institute of Technology, 1996.

[Gim75] J. Gimlett. Use of "activity" classes in adaptive transform image coding. *IEEE Transactions on Communications*, COM-23(7):785–786, July 1975.

[HDG92] Y. Huang, H. Dreizen, and N. Galatsanos. Prioritized DCT for compression and progressive transmission of images. *IEEE Transactions on Image Processing*, 1(4):477–487, October 1992.

[Huf52] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(9):1098–1102, September 1952.

[IEE90a] Standards Committee IEEE. IEEE standard specifications for the implementation of 8x8 inverse discrete cosine transform. IEEE Standard 1180-1990, 1990.

[IEE90b] Standards Committee IEEE. IEEE standard test access port and boundary scan architecture. IEEE Standard 1149.1-1990, 1990.

[Int98] Intel Corp. Mobile Pentium Processor with MMX[tm] Technology on .25 micron. http://developer.intel.com/design/mobile/datashts/243468.htm, January 1998.

[ISO94] International Organization for Standardisation ISO. Generic coding of moving pictures and associated audio, Reccomendation H.262. ISO/IEC 13818-2 Draft International Standard, 1994.

[KFN+96] T. Kuroda, T. Fujita, T. Nagamatu, S. Yoshioka, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai. A 0.9V, 150-MHz, 10-mW, 4mm^2, 2-D discrete cosine transform core processor with variable-threshold-voltage (VT) scheme. *IEEE Journal of Solid State Circuits*, 31(11):1770–1777, November 1996.

[KMO87] Y. Kato, N. Mukawa, and S. Okubo. A motion picture coding algorithm using adaptive DCT encoding based on coefficient power distribution classification. *IEEE Journal on Selected Areas in Communication*, SAC-5(7):1090–1099, August 1987.

[KS67] M.G. Kendall and A. Stuart. *The Advanced Theory of Statistics*. Charles Griffin, London, 1967.

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95, January 1980.

[LeG91] Didier LeGall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[LKRR93] H. Lee, Y. Kim, A.H. Rowberg, and E.A. Riskin. Statistical distributions of DCT coefficients and their application to an interframe compression algorithm for 3-d medical images. *IEEE Transactions on Medical Imaging*, 12(3):478–485, September 1993.

[Llo82] S.P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, IT-28(2):129–137, March 1982.

[LN97] P. Larsson and C.J. Nikol. Self-adjusting bit precision for low-power digital filters. In *Symposium on VLSI Circuits*, pages 123–124, June 1997.

[LNC96] J.T. Ludwig, S.H. Nawab, and A. Chandrakasan. Low-power digital filtering using approximate processing. *IEEE Journal of Solid State Circuits*, 31(3):395–400, March 1996.

[Loh84] H. Lohscheller. A subjectively adapted image communication system. *IEEE Transactions on Communications*, COM-32(12):1316–1322, December 1984.

[LR95] P.E. Landman and J.M Rabaey. Architectural power analysis: The dual bit type method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(2):173–187, June 1995.

[LV86] A. Ligtenberg and M. Vetterli. A discrete fourier cosine transform chip. *IEEE Journal on Selected Areas in Communication*, SAC-4(1):49–61, January 1986.

[Max60] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, pages 7–12, March 1960.

[MF92] R. Mester and U. Franke. Spectral entropy-activity classification in adaptive transform coding. *IEEE Journal on Selected Areas in Communication*, 10(5):913–917, June 1992.

[MHS+94] M. Matsui, H. Hara, K. Seta, Y. Uetani, L. Kim, T. Nagamatsu, T. Shimazawa, S. Mita, G. Otomo, T. Oto, Y. Watanabe, F. Sano, A. Chiba, K. Matsuda, and T. Sakurai. 200 MHz video compression macrocells using low-swing differential logic. In *IEEE International Solid-State Circuits Conference*, pages 76–77, February 1994.

[Min99] R. Min. Demonstration system for a low-power video coder and decoder. Master's thesis, Massachusetts Institute of Technology, In Preparation, 1999.

[MPFL97] J.L. Mitchell, W.B Pennebaker, C.E. Fogg, and D.J. LeGall. *MPEG Video Compression Standard*. Chapman & Hall, New York, 1997.

[Mul93] F. Muller. Distribution shape of two-dimensional DCT coefficients of natural images. *Electronic Letters*, 29(22):1935–1936, October 1993.

[MW92] L. McMillan and L. A. Westover. A forward-mapping realization of the inverse discrete cosine transform. In *Proceedings of the Data Compression Conference (DCC '92)*, pages 219–228. IEEE Computer Society Press, March 1992.

[MW93] L. McMillan and L. A. Westover. Method and apparatus for fast implementation of inverse discrete cosine transform in a digital image processing system using optimized lookup tables. United States Patent No. 5,224,062, June 1993.

[MW94] L. McMillan and L. A. Westover. Method and apparatus for fast implementation of inverse discrete cosine transform in a digital image processing system using low cost accumulators. United States Patent No. 5,301,136, April 1994.

[NH95] A.N. Netravali and B.G. Haskell. *Digital Pictures: Representation, Compression and Standards, Sec. Ed.* Plenum, 1995.

[Nil85] N.B. Nill. A visual model weighted cosine transform for image compression and quality assessment. *IEEE Transactions on Communications*, COM-33(6):551–557, June 1985.

[NL86] K.N. Ngan and K.S. Leong. A fast convergence method for Lloyd-Max quantizer design. *Electronic Letters*, 22:944–946, July 1986.

[NLAO97] C.J. Nikol, P. Larsson, K. Azadet, and N.H. O'Neill. A low-power 128-tap digital adaptive equalizer for broadband modems. *IEEE Journal of Solid State Circuits*, 32(11):1777–1789, November 1997.

[NLS89] K. N. Ngan, K. S. Leong, and H. Singh. Adaptive cosine transform coding of images in perceptual domain. *IEEE Transactions on Acourstics, Speech and Signal Processing*, 37(11):1743–1749, November 1989.

[NS96] L. Nielsen and J. Sparso. A low-power asynchronous data-path for an FIR filter bank. In *Second International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 197–207, March 1996.

[NS98] L. Nielsen and J. Sparso. An 85 μW asynchronous filter bank for a digital hearing aid. In *IEEE International Solid-State Circuits Conference*, pages 108–109, February 1998.

[OHM+84] J.K. Ousterhout, G.T. Hamachi, R.N. Mayo, W.S. Scott, and G.S. Taylor. Magic: A VLSI layout system. In *Proceedings of the 21st Design Automation Conference (DAC '84)*, pages 152–159, June 1984.

[PL74] A. Peled and B. Liu. A new hardware realization of digital filters. *IEEE Transactions on Acoustics Speech and Signal Processing*, ASSP-22, December 1974.

[PM92] W.B Pennebaker and J.L Mitchell. *JPEG: Still Image Data Compression Standard*. Van Nostrand Reinhold, 1992.

[RG83] R.C. Reininger and J.D. Gibson. Distributions of the two-dimensional DCT coefficients for images. *IEEE Transactions on Communications*, COM-31(6):835–839, June 1983.

[RY90] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.

[SCG89] M. T. Sun, T. C. Chen, and A. M. Gottlieb. VLSI implementation of a 16 $\times$ 16 discrete cosine transform. *IEEE Transactions on Circuits and Systems*, 36(4), April 1989.

[Sim99] T. Simon. *A Low Power Video Compression Chip for Portable Applications*. PhD thesis, Massachusetts Institute of Technology, In Preparation, 1999.

[SR96] S.R. Smoot and L.A. Rowe. Study of DCT coefficient distributions. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 2657, pages 403–411, January 1996.

[TW71] M. Tasto and P. Wintz. Image coding by adaptive block quantization. *IEEE Transactions Communication Technology*, COM-19(3):957–972, May 1971.

[UIT+92] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, Y. Yamashita, H. Terane, and M. Yoshimoto. A 100 MHz 2-D discrete cosine transform core processor. *IEEE Journal of Solid State Circuits*, 36(4), April 1992.

[Wal91] G.K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[WE93] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design, Sec. Ed.* Addison-Wesley, 1993.

[Whi89] Stanley A. White. Applications of distributed arithmetic to digital signal processing: A tutorial review. *IEEE ASSP Magazine*, July 1989.

[XC98] Thucydides Xanthopoulos and Anantha Chandrakasan. A low-power IDCT macrocell for MPEG2 MP@ML exploiting data distribution properties for minimal activity. In *1998 Symposium on VLSI Circuits Digest of Technical Papers*, pages 38–39, June 1998.

[XC99] Thucydides Xanthopoulos and Anantha Chandrakasan. A low-power IDCT macrocell for MPEG2 MP@ML exploiting data distribution properties for minimal activity. *IEEE Journal of Solid State Circuits*, 34(4), To Appear: April 1999.

[XCSD96] Thucydides Xanthopoulos, Anantha Chandrakasan, Charles G. Sodini, and William J. Dally. A data-driven IDCT architecture for low power video applications. In *European Solid State Circuits Conference (ESSCIRC)*, September 1996.

[XYC97] Thucydides Xanthopoulos, Yoshifumi Yaoi, and Anantha Chandrakasan. Architectural exploration using Verilog-based power estimation: A case study of the IDCT. In *Proceedings of the 34th Design Automation Conference (DAC '97)*, pages 415–420, June 1997.

[YS89] J. Yuan and C. Svensson. High-speed CMOS circuit technique. *IEEE Journal of Solid State Circuits*, 24(1):62–70, February 1989.

Appendix A

Design Tools and Methods

This appendix describes the tools and methodologies used for the physical design of the IDCT and DCT core processors of chapters 3 and 5 respectively. The design flow includes a number of custom tools developed by the author for library development, netlist processing, standard cell place and route etc. This appendix is meant to serve as a source of documentation for such tools so that the expertise developed over the course of the last four years can be passed on to new students.

A.1 Design Flow

Figure A.1 illustrated the flow followed during the design and implementation of the DCT and IDCT chips of the previous chapters. This design flow skips some standard steps (i.e. Verilog behavioral modelling) but has been followed by the author over the course of two reasonably large designs (120K and 160K transistors) and has been found to produce working chips within a reasonable amount of time (approximately 6-8 months per chip starting from a blank sheet of paper.)

At the top of the flow is the chip functional modelling using a high level systems language (i.e. C/C++). This step is essential for a proof-of-concept chip implementation in a short time and also as a "live" chip specification used to produce input/output vectors for subsequent verification steps. We proceed by entering gate-level schematics using standard cells from a previously developed library. Schematics are simulated using gate-level Verilog and verified against the C/C++ chip model. Before proceeding to chip physical design, schematics are simulated at the transistor level using circuit simulators such as Hspice and Timemill/Powermill. The next step is the system physical design (layout) in two dimensions. The final two steps are comparison of the schematic and extracted layout netlist (LVS) and circuit simulation of the extracted layout netlist which includes extracted wiring parasitic capacitances. This last step is essential for correct chip operation at the specified supply voltage and clock frequency. After the completion of this step, manufacturing output (ASCII CIF or gdsII binary stream) is produced for the fabrication house.

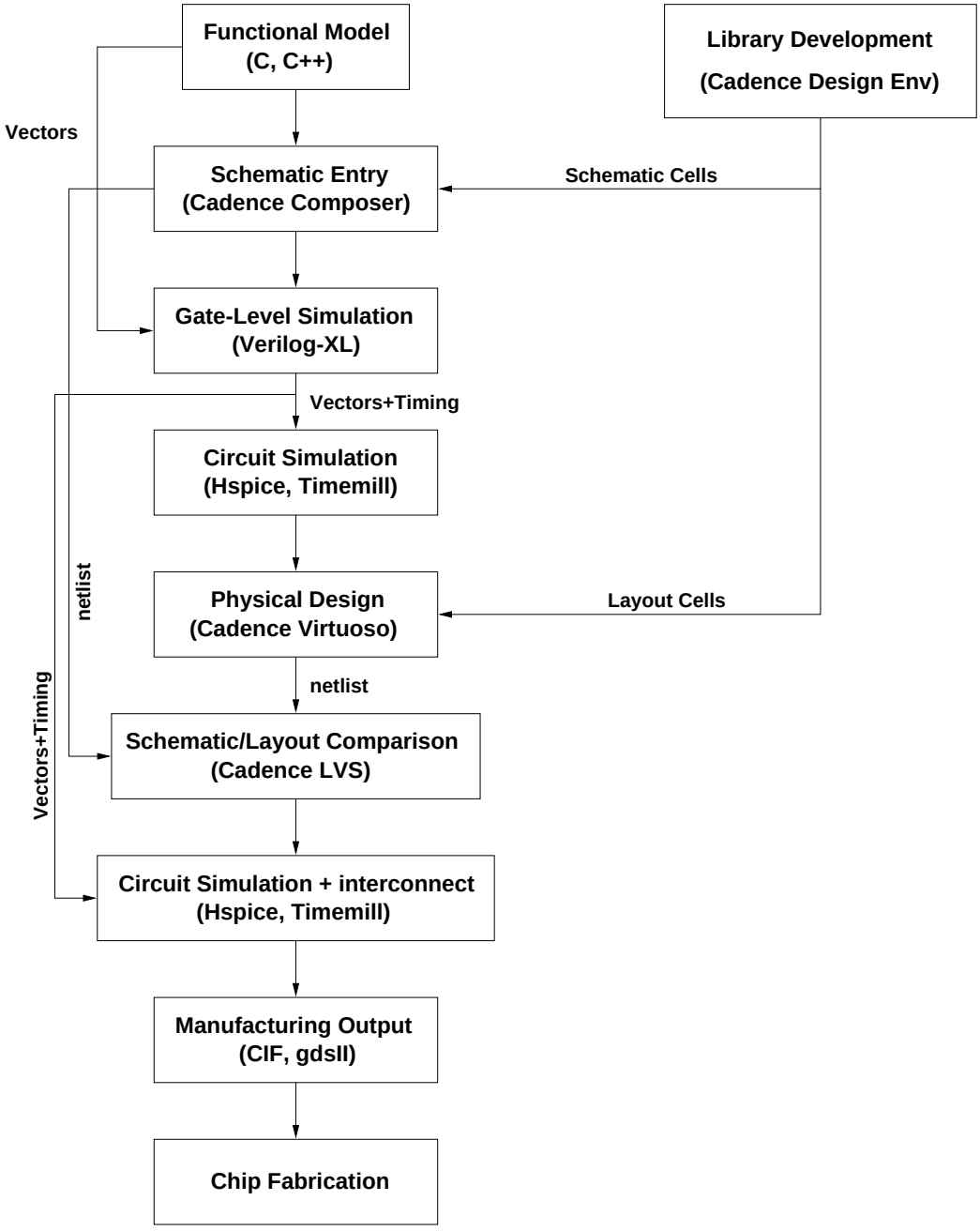


Figure A.1: Design Flow

A.2 Functional Chip Model

A chip functional model using a high level system language is essential for three main reasons: First, it provides a quick proof-of-concept implementation of the system under consideration and it is also an excellent form of documentation for the intended chip functionality. Second, it is a source of stimuli and verification vectors (digital inputs and outputs) that will be used in subsequent flow steps (Figure A.1). Third, due to the execution speed of a system language, the functional model can be used as a testbed for implementing and evaluating design alternatives (i.e. bitwidth of arithmetic units.)

The functional model has a notion of all the system functional blocks (usually implemented as separate function calls) and should produce bit-equivalent results at each functional block boundary. Exact timing information is not embedded in the code, but a high-level task sequence must be present and must be equivalent to the sequence of tasks that will be performed by the actual chip.

Two separate functional models in C have been developed for both the DCT and the IDCT processor. Both have been used to determine the optimum arithmetic unit bitwidth for minimum power and acceptable precision [IEE90a]. Both have been combined to produce a rudimentary still image compression software package (tcode) emulating the behavior of the DCT/IDCT core chipset. The functional models have been appropriately instrumented so that a number of intermediate outputs can be extracted for purposes of incremental debugging.

A.3 Library Development

A standard cell library complete with cell symbol, functional (Verilog), schematic (transistor-level) and layout views is essential for speedy design entry. The author has developed two such libraries during the course of the chip designs, one for the HP CMOS14 (0.5 μm) process available through MOSIS using SCMOS parametrised (λ-based, λ =0.35 μm) design rules, and one for the Analog Devices (ADI) 0.6 μm DPTM process using micron-based design rules. Raj Amirtharajah and Tom Simon have also made substantial contributions for the completion of the ADI library.

In the Cadence design framework a standard cell has 4 required views:

Symbol The symbol view is created manually by the designer or copied from a template library provided by the CAD vendor. The symbol view is the main cell placeholder within a schematic drawing.

Functional The functional view is a Verilog module (preferably primitive) which functionally describes the cell. This view is also manually created by the designer and is used for Verilog gate-level simulations.

Schematic This view contains the transistor-level schematic of the cell along with transistor size information. This schematic is entered manually by the designer.

Layout This view contains the layout polygons. This view can be generated in one of three ways within our CAD framework:

1. The layout can be generated manually by the designer (an activity otherwise known as "polygon pushing"!). This is the most common form of layout design entry and the only available method for complex cells such as flip-flops and XOR gates.
2. A number of standard cell libraries (some of them of dubious quality) are freely available among the academic community (i.e. Berkeley low-power standard cell library [Bur94]). Invariably, all these libraries are available in Magic [OHM+84] format. The author has developed a software package (mag2skill) which converts such cells in Cadence layout given simple configuration information.
3. A layout generation package for simple static CMOS cell generation from schematics is also available. This package (layoutGen) was originally developed by Larry Dennison [Den95] but has been substantially modified and parametrised by the author and Raj Amirtharajah to produce high quality layout with enforceable parametrised rules. This package reads a cell schematic along with design rule information and cell dimensional specifications (i.e. cell height and width of power and ground rails) and produces layout of quite acceptable quality complete with substrate and well contacts obeying all furnished design rules.

Layout generation techniques 2 and 3 are expanded upon in the following sections.

A.3.1 Magic to Cadence Translation

The present location of the mag2skill package is under /jake/mag2skill. It has been written entirely in the C programming language. The program operates on .mag cell files. A section from such a layout description is reproduced in Figure A.2. The double angle brackets specify a layer change, whereas the "rect" declarations specify a rectangle instance. The following 4 integers specify the x and y coordinates of the rectangle diagonal.

Before running the executable, a technology file is necessary. A sample technology file included in the mag2skill distribution is reproduced in Figure A.3. This file provides certain design rule information necessary to translate Magic contacts and contact arrays to Cadence raw contact cuts and contact cut arrays. In addition to design rule information, the file provides a correspondence between the layer names assumed by each separate layout editor.

The DERIVE_SELECT flag should be set to one if the translator is to derive the N-select and P-select layers. Such derivation is performed by expanding the N-diffusion and P-diffusion layers by one SELECT_OVERLAP unit (3 λ in this example).

Magic has a unique way of representing contacts (poly-metal, diffusion-metal and vias). All three physical layers (top contact layer, bottom contact layer and contact cut including contact surround) are represented with a single rectangle on the corresponding

```
.
.
.
<< nwell >>
rect -3 24 51 58
<< polysilicon >>
rect 11 54 29 56
rect 7 50 9 52
rect 11 50 13 54
.
.
.
```

Figure A.2: Magic Layout Cell Description

contact layer. Only during the extraction of manufacturing output are the underlying layers derived. As a result, the translation to a standard layout editor (Cadence Virtuoso) should perform this physical layer derivation. Figure A.4 demonstrates the translation of a Magic via array to a Cadence via array. The reader should note that the technology file in Figure A.3 specifies three separate cadence layers for each Magic contact layer (e.g. "polycontact" translates to "poly", "metal1" and "cont" in Cadence).

Except for all the contacts and the two transistor devices (poly overlapping diffusion) all the other rectangles undergo a one-to-one translation. The mag2skill executable produces a SKILL source code file (SKILL is an interpreted LISP-like dialect that implements the procedural interface of the Cadence CAD package) which when run in the Cadence Command Interpreter Window (CIW) produces a cell layout view. Figure A.5 contains the section of the generated SKILL code that produces the Magic rectangles of Figure A.2.

An Example

In this section, a sample magic cell from the Berkeley low power library (drif301.mag, 2-stage driver) is translated to Cadence format. The following steps are followed:

1. The technology file (/jake/mag2skill/tech/m2s.tech, also shown in Figure A.3) must be copied in the present directory at the same level as the cell to be translated (drif301.mag).

2. The following commands must be typed at the Unix prompt: "m2s drif301 stdcells". Note that the .mag extension is omitted. The second command line argument ("stdcells") is the name of the Cadence library where the user wishes to place the newly translated cell.

3. After the previous step, a file called "drif301.il" has been created in the present directory. This file contains SKILL code that when run generates all the shapes that

```
#  magic2skill Technology File
#  D. Xanthopoulos 1996

# this must be set to 1 if the select mask must be included
DERIVE_SELECT    1

# Necessary Design Rules
CONTACT_SIZE 2
CONTACT_SPACING 2
CONTACT_OVERLAP 1
VIA_SIZE 2
VIA_SPACING 2
VIA_OVERLAP 1
SELECT_OVERLAP 3

#Local Name     Magic layer             Cadence layer(s)
PW              pwell                   none
NW              nwell                   nwell
POLY            polysilicon             poly
NDIFF           ndiffusion              ndiff
PDIFF           pdiffusion              pdiff
M1              metal1                  metal1
M2              metal2                  metal2
M3              metal3                  metal3
NT              ntransistor             ndiff poly
PT              ptransistor             pdiff poly
PSUB            psubstratepdiff         psub
NSUB            nsubstratendiff         nsub
GLASS           glass                   overgla
# VERY IMPORTANT!!!!
# Contacts must be specified as layer1 layer2 contact_cut
#
PC              polycontact             poly metal1 cont
NDC             ndcontact               ndiff metal1 cont_aa
PDC             pdcontact               pdiff metal1 cont_aa
M2C             m2contact               metal1 metal2 via
PSC             psubstratepcontact      psub metal1 cont_aa
NSC             nsubstratencontact      nsub metal1 cont_aa
end
```

Figure A.3: mag2skill Technology File. All integers represent λ units (MOSIS SCMOS design rules).

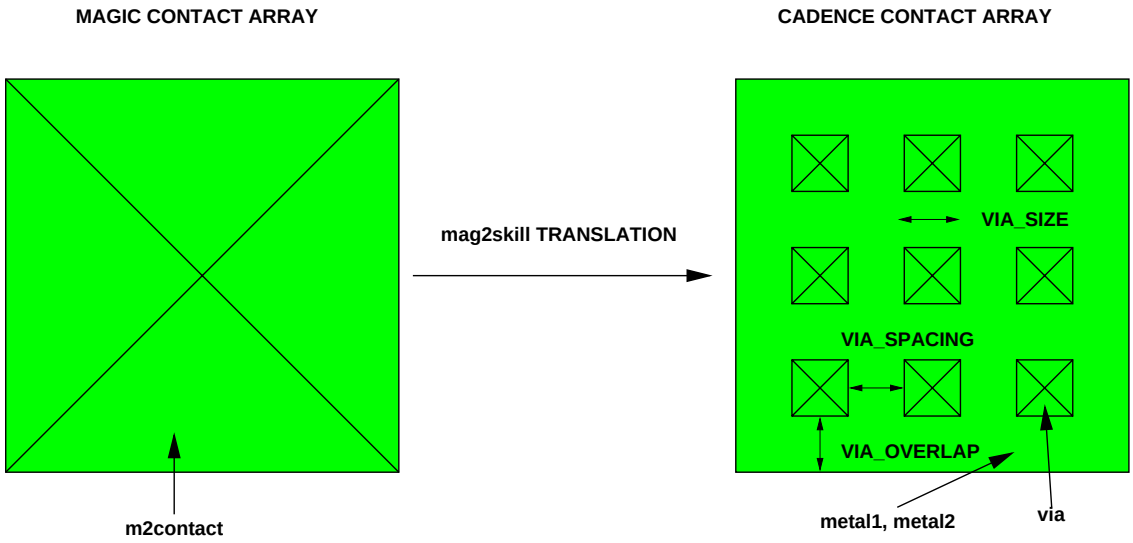


Figure A.4: Translation of a Magic Via Array to Cadence Layout

```
.
.
.
(dbCreateRect cv (list "nwell" "drawing")
        (list (list -3 24) (list 51 58)))
(dbCreateRect cv (list "poly" "drawing")
        (list (list 11 54) (list 29 56)))
(dbCreateRect cv (list "poly" "drawing")
        (list (list 7 50) (list 9 52)))
(dbCreateRect cv (list "poly" "drawing")
        (list (list 11 50) (list 13 54)))
.
.
.
```

Figure A.5: SKILL Code Generated by mag2skill Corresponding to the Rectangles of Figure A.2

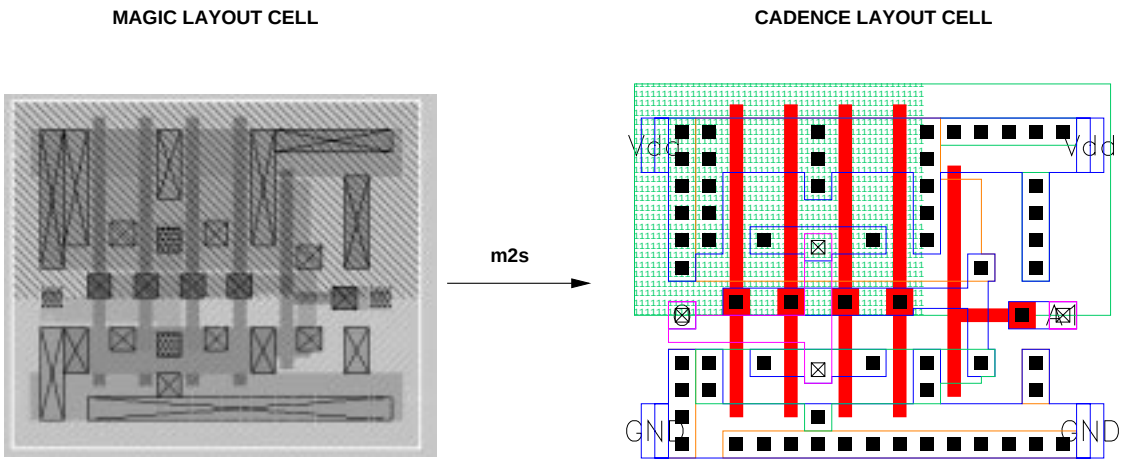


Figure A.6: Translation of a Magic 2-stage Driver to Cadence Format

comprise the buffer. The user should type "load "drif301.il"" in the Cadence CIW window.

4. The user should create a Cadence cell named "drif301" within the "stdcells" Cadence library.

5. Finally, the user should type "lg" (Layout Generate) in the CIW. Under cell drif301, a layout view has been created that contains the same shapes (with appropriately translated contacts and contact arrays) as the Magic layout. The Magic cell and the newly generated Cadence cell appear in Figure A.6. The reader's attention is directed to the translation of contacts in the figure.

The author has used certain Berkeley cells (TSPC flops) in the design of the IDCT processor which had been translated using m2s. Paul Chou [Cho96] used m2s extensively to develop a low power ROM generator in the Cadence environment using Berkeley Magic cell blocks. Abram Dancy [Dan96] extended m2s to enforce micron-based design rules. He used the translator to convert Berkeley Magic cells in the Cadence environment obeying ADI 0.6 μm micron-based rules.

A.3.2 CMOS Cell Library Generator (layoutGen)

A SKILL packaged originally developed by Larry Dennison [Den95] for MOSIS scalable CMOS (SCMOS) rules has been adapted by the author and Raj Amirtharajah for use with programmable design rules. The program (layoutGen) can produce very usable layout for basic static CMOS structures (inverters, and / nand / or / nor gates) by reading device sizes and connectivity from schematics.

Usage is very simple: After the designer has finished entering the schematics for a standard cell, he / she needs to load the layoutGen package in the Cadence CIW window:

```
CIW> load ‘‘layoutGen.il’’
```

Then, he/she proceeds to create the cell layout:

```
CIW> (layoutGen ‘‘libname’’ ‘‘cellname’’)
```

After the execution of the layoutGen command, a layout cellview for the corresponding cell will have been created. Specification of design rules along with other cell parameters (i.e. cell height, with of supply rails) appears in a separate file (“layoutGlobals.il”) which is read at runtime by layoutGen.

The layout generation program first places all transistors using an ad-hoc procedure to maximise contact sharing. Then the supply rails are drawn along with internal connectivity wires. Finally, the cell is scanned for possible places for well and substrate contacts, and the maximum possible contacts are placed. During the last step, the cell terminals are also placed.

When a complex cell is given to layoutGen, the generator attempts a best effort layout (device placement always completes) and exits after producing a cell with incomplete connectivity. The produced cell is usually an excellent starting point for hand layout. Figure A.7 shows example cell layouts produced by layoutGen for the ADI 0.6 μm process. All the displayed layouts have been produced by software only with no intervention from the designer. All layouts pass the ADI design rules.

Substantial thought has been put to a number of layout details. As an example, note how the generator does not center the N-diffusion and P-diffusion islands for the 2-input and the 4-input NAND gates but does so for the 5-input AND so that the “jogged” input gate polysilicon still fits within the specified cell height.

Two optional command line options are available for layoutGen. They are invoked as follows:

```
CIW> (layoutGen ‘‘libname’’ ‘‘cellname’’ ?splitTrans t ?inv t)
```

The command line option “splitTrans” directs layoutGen to split large transistors and draw them with multiple fingers. If this option is not set, large transistors will be drawn with a single gate finger and the cell height will possibly exceed the spacing between supply rails. The “inv” option should be used when a multiple-finger inverter is to be generated. It directs layoutGen to short all gate poly fingers from one end of the cell to the other.

A.4 Schematic Entry and Gate-Level Simulation

Schematic entry has been the basic form of design entry during the development of both the DCT and IDCT chips. The reader will note that HDL-level modelling (i.e. behavioral Verilog/VHDL) is notoriously absent from this design flow. There are two main reasons that we concluded that this step can be skipped:

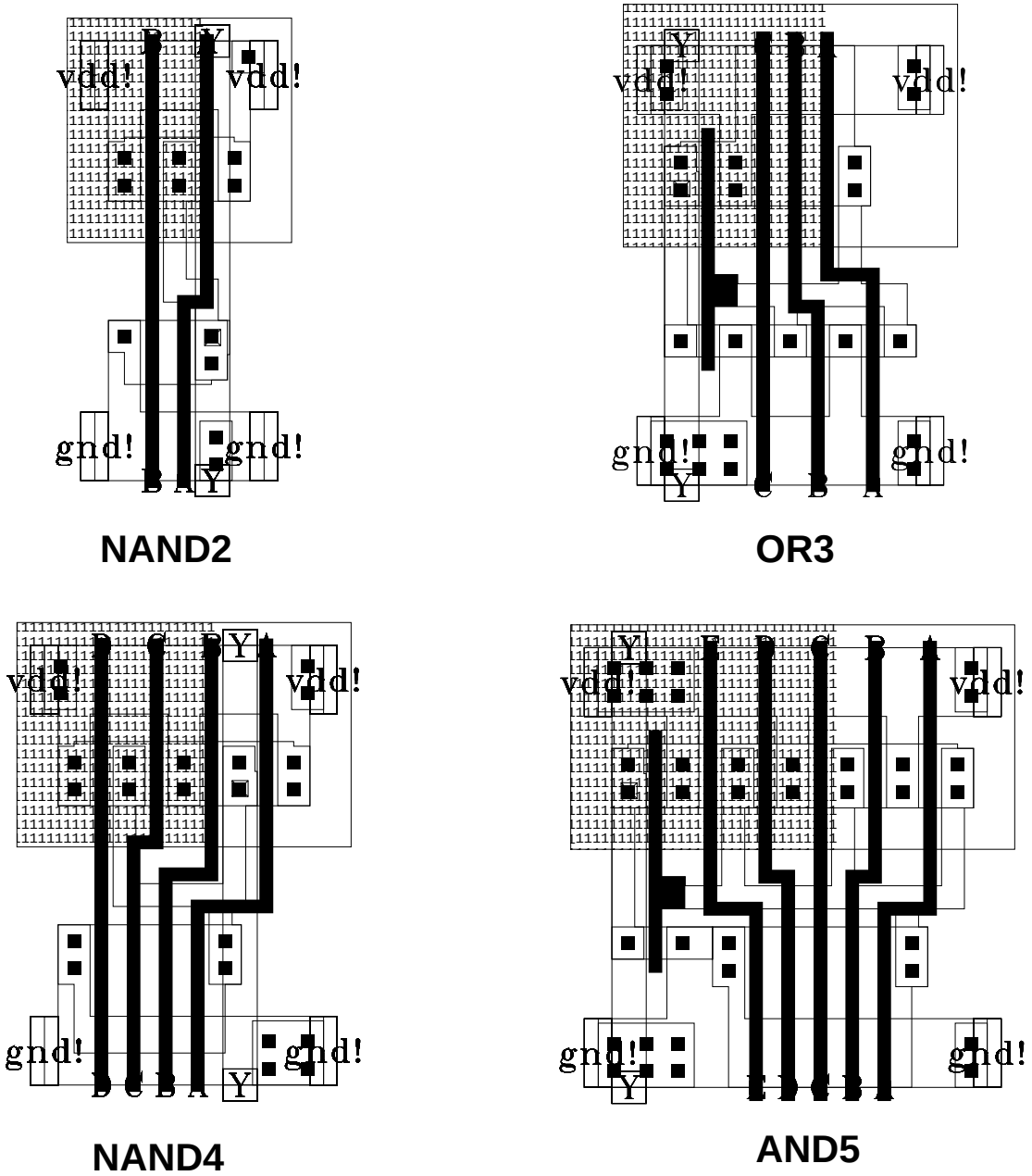


Figure A.7: Standard Cells Produced by layoutGen

1. The C-level functional model has already provided a proof-of-concept system implementation without detailed timing information. Yet, detailed circuit design may easily introduce new constraints which will affect the timing structure assumed during the behavioral modelling. In many cases, gate-level schematic entry can render behavioral modelling quite obsolete. On the other hand, the C-level model provides enough information for the designer to proceed directly to gate-level schematic entry. A time-consuming step can thus be saved.

2. Debugging at the HDL behavioral modelling can be quite time consuming for a number of reasons. Hardware description languages are essentially parallel and deviate substantially from the sequential execution model of most programming languages. Moreover, the event-triggered character of such languages can lead to erroneous event loops which are difficult to identify by looking at textual program listings. On the other hand, schematic printouts provide very effective visual information that can easily summarize dependencies and timing and render debugging far more effective and less time consuming.

Every library cell is backed by a Verilog textual view that encapsulates its functionality. A schematic can be netlisted into a hierarchical Verilog circuit description. The Verilog netlist is simulated using logic vectors generated from the C-level model. When I/O equivalence is achieved for a large number of vectors, the chip schematic is considered validated at the gate level.

A.5 Schematic Circuit Simulation

The Verilog simulation guarantees logical correctness but cannot catch circuit-level bugs such as underdriven nets, charge sharing problems, race conditions, setup/hold time violations etc. A circuit simulator such as Hspice and Timemill/Powermill is necessary for this step.

Hspice is a high quality circuit simulator but it cannot handle large circuits effectively. Hspice is ideal for things such as simulating individual cells, and identifying the critical path but it cannot handle an entire chip such as the DCT and IDCT chips. On the other hand, Timemill is capable of handling large systems at the expense of a lower-quality event-triggered simulation. The author has found that Timemill may produce erroneous results at low voltages ($<$1.5V) especially for non-static CMOS circuit styles. Nevertheless, it is currently the only tool in the market than can simulate large netlists at the circuit level. Powermill is essentially the same simulation engine with more precision and the ability to monitor supply currents for power measurements.

Both DCT and IDCT chips have undergone extensive Timemill simulations and have been validated at the circuit schematic level.

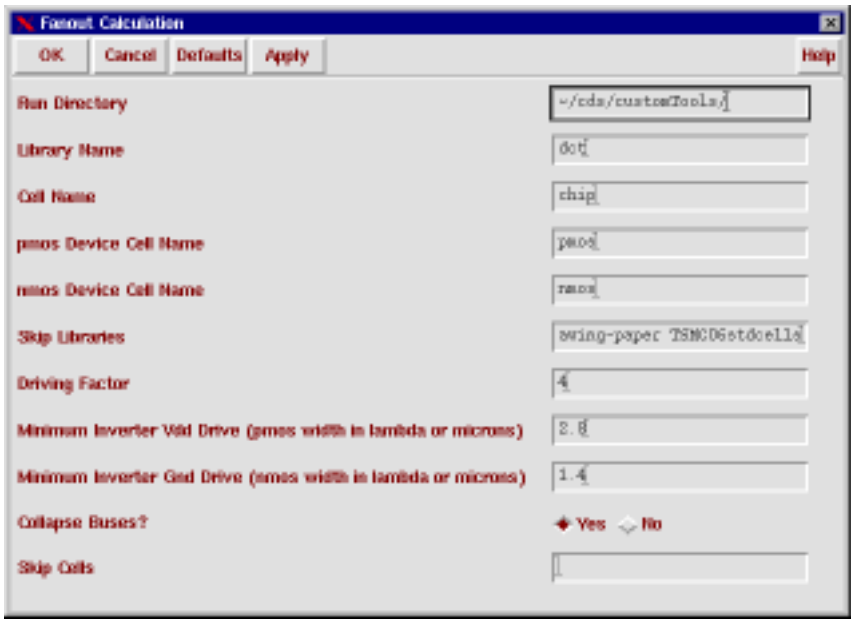


Figure A.8: Static Net Fanout Checker Entry Form

A.5.1 Static Net Fanout Checking

An additional step that should come before the circuit simulation that can save a lot of debugging time is the static net fanout checking. The author has developed a fanout checker than can descend into a hierarchical circuit, compute the total gate load on each circuit node, trace the driving transistors and report nets that are underdriven and will have large rise and fall times. The checker can identify such problematic nodes within a few seconds and save hours of erroneous circuit simulations.

The checker is very easy to use and has been fully integrated within the Cadence design framework. It has been added under a “Custom Tools” Menu in the Cadence CIW Window. Selection of the corresponding menu entry brings up the form of Figure A.8. A brief description of the form fields follows:

Run Directory Specifies the directory where the resulting file will be placed.

Library Name Library Name of the hierarchical cell on which we wish to run the checker.

Cell Name Cell Name of the cell in question.

pmos/nmos Device Cell Names Names of PMOS and NMOS transistor library elements in the schematic. These are the stopping cells in the hierarchical check.

Skip Libraries Cells that belong to these libraries are also treated as stopping cells so that the output and run time of the program can be pruned. As an example, we typically

do not wish for the checker to descend all the way into logic gates and check the fanout of local interconnect.

Driving Factor This number specifies what the desirable driving strength multiplicative factor is. Any net whose load-to-driving-strength ratio is less than this number is reported as being underdriven.

Minimum Inverter VDD/ GND Drive Size in microns of minimum inverter PMOS and NMOS width. This field is used to determine an appropriate driving inverter for the loaded net. This inverter is suggested to the user as the minimum acceptable driver for any net that is found to be underdriven.

Collapse Buses? This option reports vector nets (buses) together if they share the same load-to-driving-strength ratio instead of individually. This option reduces the textual output and produces fewer and more useful warning messages.

Skip Cells The checker does not descend into the cells that are listed in this field. This option can be useful if the schematic design is not in its final form and contains empty cells that are not backed by schematic views. Such cells should be listed in this field.

Execution of the fanout checker brings up a text window that contains a list of warnings about underdriven nodes. An example entry is shown below:

```
------------------------------------------------------
         Cell: msbrclass_ctrl_bot0   Library: dct
------------------------------------------------------
Nets that are 100% underdriven:

Net ldmaskb is underdriven to Vdd (load = 222.60 drive= 33.60)
    Consider inv13.2x
Net ldmaskb is underdriven to Gnd (load = 222.60 drive= 16.80)
    Consider inv13.2x
Net mask is underdriven to Vdd (load = 207.20 drive= 33.60)
    Consider inv12.3x
Net mask is underdriven to Gnd (load = 207.20 drive= 16.80)
    Consider inv12.3x
Net zero_mask is underdriven to Gnd (load = 28.00 drive= 1.40)
    Consider inv1.7x
```

The drive reported is the gate width of the transistor pulling the node towards the VDD or GND supply. The load reported is the total gate width of all transistors whose gate is directly attached to the net. The checker does not take into account source / drain junction capacitances.

A.6 Physical Design

The physical design (layout) is by far the most time-consuming part of the design process. Both the DCT and IDCT chips have a large number of full custom layout blocks. Yet, the DCT chip has large random-logic control blocks that have been automatically generated using an internal standard cell place and route tool. This tool is described in the next section.

A.6.1 Custom Standard Cell Place and Route Tool

The place and route tool has been originally developed by Larry Dennison [Den95]. The author has rewritten a number of sections in order to parametrise the design rules and render it process independent. Moreover, the router has been completely rewritten to allow more routing freedom and efficiency.

The tool has been written entirely in SKILL. While describing the tool, we will use the terms "function" and "procedure" interchangeably to describe SKILL language functions. The starting point is a gate-level schematic. In the beginning, the user creates a piece of SKILL source code that encapsulates the entire layout generation. We call this SKILL program, the "generator". Then, the user edits the generator, records his/hers preferred standard cell placement, and makes some adjustments to the wiring. Then the generator is run, and the layout is produced.

First, the user must load the relevant SKILL packages within the Cadence environment. The following must be typed in the Cadence CIW:

```
CIW> load ‘‘mgg3.il’’
CIW> load ‘‘std.il’’
```

Both skill files are located in the main SKILL repository on /yoda. The generator is created when the user types the following command in the Cadence CIW:

```
(mgg ‘‘libname’’ ‘‘cellname’’)
```

where "libname" and "cellname" are the library name and cell name of the schematic cell which we wish to generate layout for. The **mgg** program (the name stands for **m**odule **g**enerator **g**enerator) produces a SKILL source code file ([cellname].il) in the current working directory. This file contains the definition of the generator procedure.

The generator starts by defining a toplevel SKILL function (mg) that encapsulates the entire layout process. The structure of the generator function is shown in Figure A.9. First, the layout cellview is created and bound to the local variable cv. Then, we proceed by performing the leaf cell instantiation in the current cellview (call to the "instantiate" procedure). Then the "placement" procedure is called which places all the previously instantiated cells. After the placement is done, the "channelHorzFind" procedure is called which

```
(defun mg ( @key (report_length 300.0))
  (let (
        <local variables>
        )

    (setq cv (dbOpenCellViewByType <libname> <cellname>
                "layout" "maskLayout"  "w"))

    (instantiate)
    (placement)
    (setq channels (channelHorzFind cv
                                    ?westExtend 13.2
                                    ?eastExtend 13.2
                                    ?topChannelHeight 50.0
                                    ?botChannelHeight 50.0
                                    ))
    (wireVddGnd cv channels)

    (wire_clk)
    (wire_reset)
    (wire_net0)
    .
    .
    .
    .
    (wire_net1000)
  ))
```

Figure A.9: Module Generator SKILL Source Code

identifies all the rectangular routing channels given the placement that has just been performed. This procedure returns an array of rectangles (channels[]) that correspond to the channels available for routing tracks between cell rows (Figure A.10). The arguments to "channelHorzFind" are also illustrated in Figure A.10. The next function call ("wireVddGnd") draws the power and ground rails (horizontal metal1 and vertical metal2) that shorts the supply inputs of all the cells together and produces a power / ground grid (Figure A.10). All the procedure calls that follow draw individual wires among the cells. There is one procedure call for each wire in the block.

The "instantiate", "placement" and all the wiring procedures are also defined in the same file that the generator function (mg) is found. All other functions (i.e. "channelHorzFind") are part of the place and route package and they have been defined after the execution of the first two "load" statements.

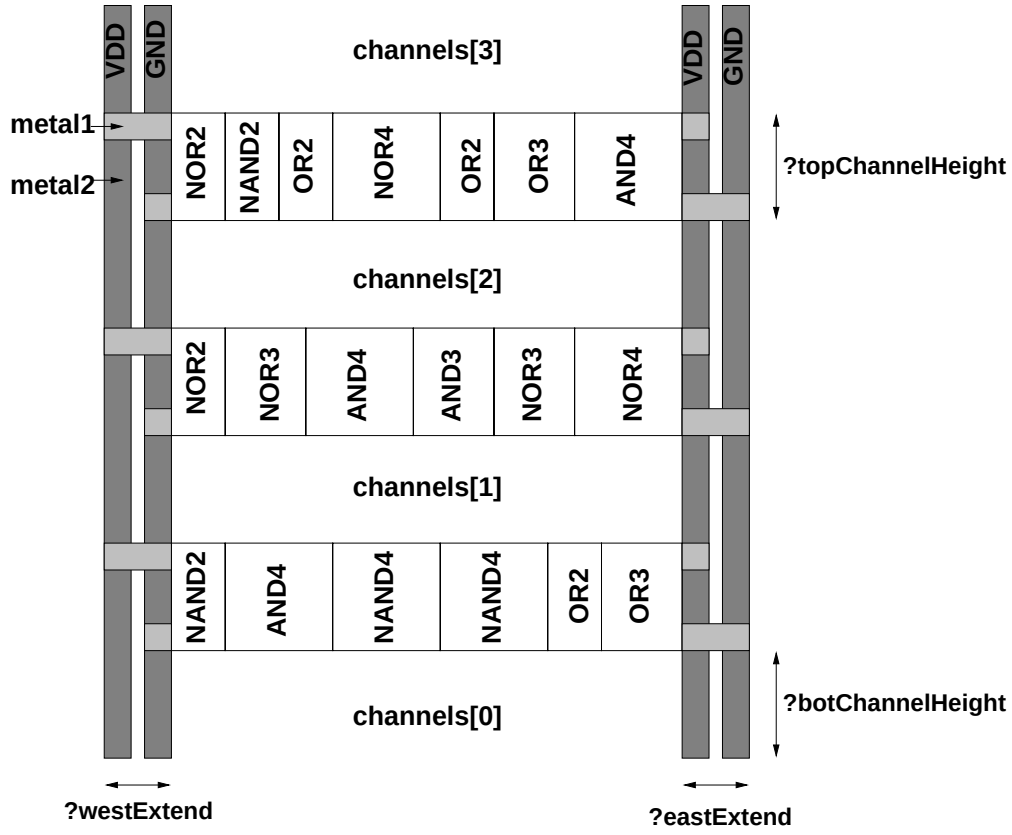


Figure A.10: Placement, Routing Channel Identification and Power / Ground Routing

```
(defun instantiate ()
  (let (
      nand2
      nor2
      inv2x
      inv
      and2
    )

    (setq nand2 (dbOpenCellViewByType "stdcells" "nand2"
                  "layout" nil "r"))
    (setq nor2 (dbOpenCellViewByType "stdcells" "nor2"
                  "layout" nil "r"))
    (setq inv2x (dbOpenCellViewByType "stdcells" "inv2x"
                  "layout" nil "r"))
    (setq inv (dbOpenCellViewByType "stdcells" "inv"
                  "layout" nil "r"))
    (setq and2 (dbOpenCellViewByType "stdcells" "and2"
                  "layout" nil "r"))

    I158 = (dbCreateInst cv and2 "I158" (list 0.0 0.0) "R0")
    I189 = (dbCreateInst cv nor2 "I189" (list 0.0 0.0) "R0")
    I64 = (dbCreateInst cv nand2 "I64" (list 0.0 0.0) "R0")
    I167 = (dbCreateInst cv inv2x "I167" (list 0.0 0.0) "R0")
    I193 = (dbCreateInst cv inv "I193" (list 0.0 0.0) "R0")
    ))
```

Figure A.11: Cell Instantiation Procedure

Before describing placement and routing, we note that the mg procedure may take a single optional argument (report_length [number]). This argument causes the wiring functions to report any routed net that is longer than [number] microns. This functionality is very useful in order to determine long wires where more buffering may be necessary.

Cell Instantiation

Figure A.11 shows the structure of an example “instantiate” procedure. The first 5 SKILL assignments open the master layout cells and bind them to local variables. The second set of 5 statements instantiates the master cells in the current cellview (cv) using the primitive SKILL *dbCreateInst* procedure. The “instantiate” function is always generated automatically by **mgg** and there is never any need for designer input. We should note that this procedure preserves instance names from schematic to layout.

```
(defun placement ()
  (let ()
    (instsPlaceRow (list
                     I149
                     I150
                     I151
                     I193
                     I192
                     I194
                    )
          0.0 0.0 )
  ))
```

Figure A.12: Cell Placement Procedure

Cell Placement

The "placement" procedure is where most of the user input is concentrated. Initially, **mgg** produces a blank placement procedure. The easiest way to specify cell placement, is by using the predefined "instsPlaceRow" procedure. This is illustrated in Figure A.12. This procedure needs three arguments: a list of instances, an x-coordinate and a y-coordinate. The procedure places all standard cell instances in a row at the specified coordinates. Multiple calls to "instsPlaceRow" produces multiple rows of standard cells as shown in Figure A.10.

Cell placement is the most important design decision as far as this phase is concerned and has a great impact on the subsequent wiring steps.

Routing

Figure A.13 shows an example net wiring function. The "wireMultiChannel" procedure draws the wiring trace when given the following arguments:

- Current layout cellview
- Routing channel array (computed by "channelHorzFind")
- Number of routing channels (size of channel array.)
- List of output terminals connected by the current wire (each list item is another two element list. The first element is the cell instance identifier and the second element is the cell terminal name.)
- List of input terminals connected by the current wire (same format as above)

```
(defun wire_net101 ()
  (let ()
     (wireMultiChannel cv channels num_channels
      (list
        (list I4 "q")
       )
      (list
        (list I24 "A")
        (list I5 "A0")
       )
            ?netName "net101"
            )
    ))
```

Figure A.13: Example Wiring Function

Optional arguments include the net name and flags indicating whether the particular net is a toplevel terminal. If this is true, "wireMultiChannel" places a pin on the net. If all the cell terminals are found in the same wiring channel (wire0 in Figure A.14), the "wireMultiChannel" procedure finds an empty horizontal metal1 track in the channel using a greedy search, draws it and also draws the relevant vertical metal2 or poly stubs to connect the cell terminals to the horizontal track. If on the other hand the cell terminals span multiple routing channels, the net is broken into multiple subnets that are connected together with vertical metal2 jumpers routed over the cells (wire1 in Figure A.14). The tool guarantees that the vertical metal2 jumpers never interfere with underlying metal2 inside the cells and all relevant design rules are judiciously enforced.

The routing algorithm implemented is rather simple. It is 100% greedy in the sense that when a wiring function is called, the best location for the trace is found and the trace is placed. The algorithm does not have a global view of all nets and does not try to produce a globally optimal trace allocation. Changing the order that the wiring functions are called can change the total number of traces required per routing channel. We have found that a few iterations of changing cell placement and wiring order can produce quite acceptable results.

Figure A.15 shows an example standard cell logic block generated using the process described. The block shown is part of the MSB rejection control logic of the DCT chip. The block of Figure A.15 was generated by a 3700-line SKILL program.

A.7 Extracted Layout Simulation

The final simulation phase before layout is a transistor-level simulation including wiring parasitic capacitances. This is necessary to ensure that all nets have appropriate driv-

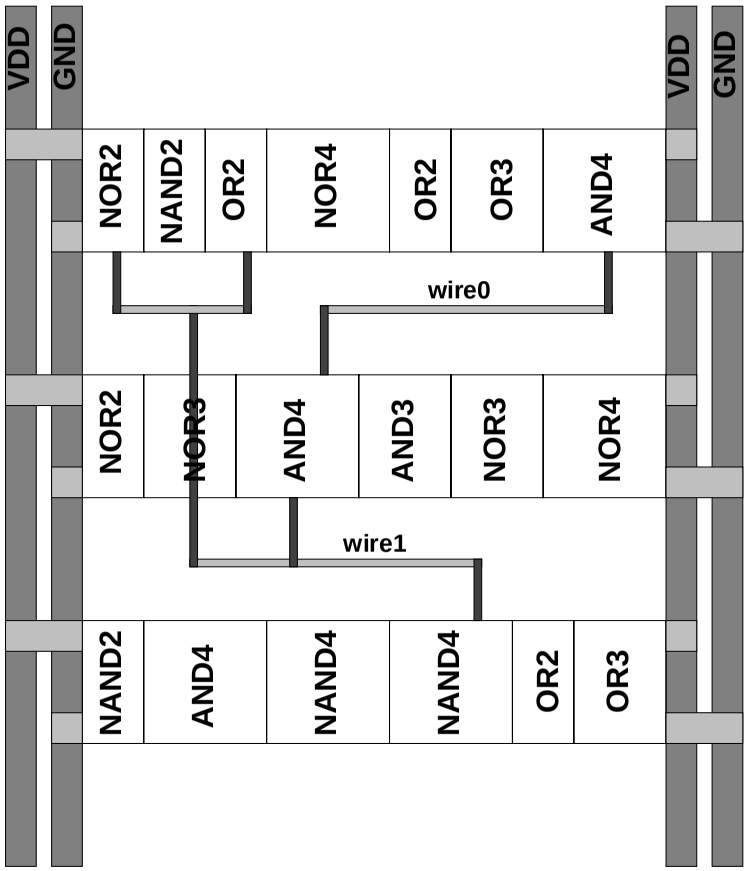


Figure A.14: Example Generated Wire Traces

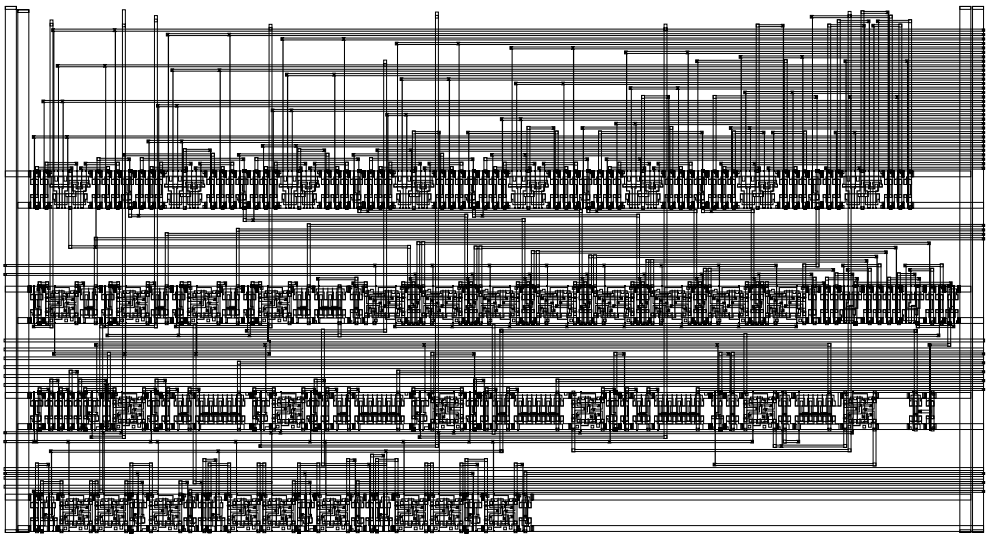


Figure A.15: Automatically Generated Random Logic Block

ing strength to meet the timing requirements when the layout wiring capacitances are accounted as well.

A.7.1 Capacitance Extraction

The Cadence design framework contains a sophisticated layout extraction tool that can calculate all possible parasitic capacitances among layout wires when given appropriate rules. The author has found that the best strategy for fastest simulation is to refer all wiring parasitic capacitances to ground. Figure A.16(a) shows the cross section of two parallel wires forming capacitors C1, C2, and C3. Figure A.16(b) shows the equivalent circuit used for simulation. Capacitor C2 is no longer a line-to-line capacitor but has been referred to ground with a Miller multiplicative factor of 2 to account for the worst case when the two lines move in opposite directions. The reason that we do not want line-to-line capacitances in circuit simulations is that capacitor loops (Figure A.16(a)) impose additional circuit constraints and they slow down the simulation, sometimes causing convergence problems. It must be noted that an equivalent circuit such as the one shown in Figure A.16(b) cannot catch errors that occur due to line-to-line coupling (i.e. glitching due to coupling on a net that must be glitch free.) The designer must identify sensitive wires (i.e. clocks, asynchronous resets etc.) and make sure that they are laid out appropriately, minimizing line-to-line coupling with adjacent nets.

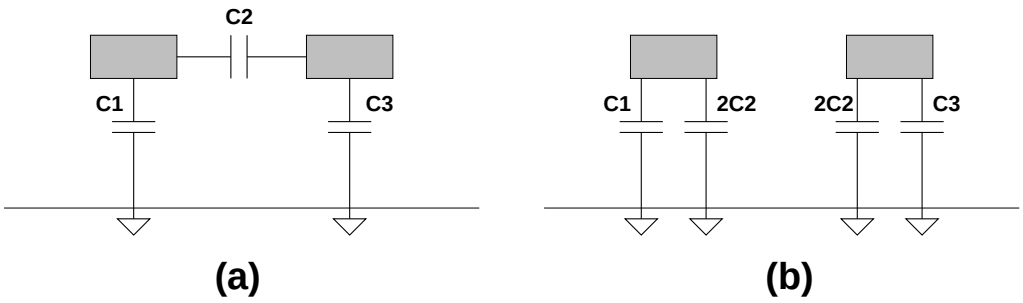


Figure A.16: Wiring Parasitic Capacitance Equivalence

A.7.2 Netlist Renaming (rnmNetlist)

Layout extraction produces a flat transistor-level netlist without preserving the net names of the corresponding schematic view. This net renaming is rather inconvenient when the designer simulates extracted layout because net monitoring and error tracking becomes a problem. We have solved this problem by writing a perl script that reinstalls the schematic net names in an extracted layout Hspice netlist. The script ("rnmNetlist") has been developed by Gangadhar Konduri with minor additions and modifications by the author.

The script works as follows: The Layout-Versus-Schematics (LVS) verification procedure produces a net cross-reference table after successfully matching the layout and schematic netlists net-by-net and device-by-device. The script utilizes this information to produce an Hspice netlist that has the connectivity of extracted layout and the original net names of the schematic view.

Usage is as follows:

```
unix> rnmNetlist schem_netlist lay_netlist xref.out hsp_netlist
```

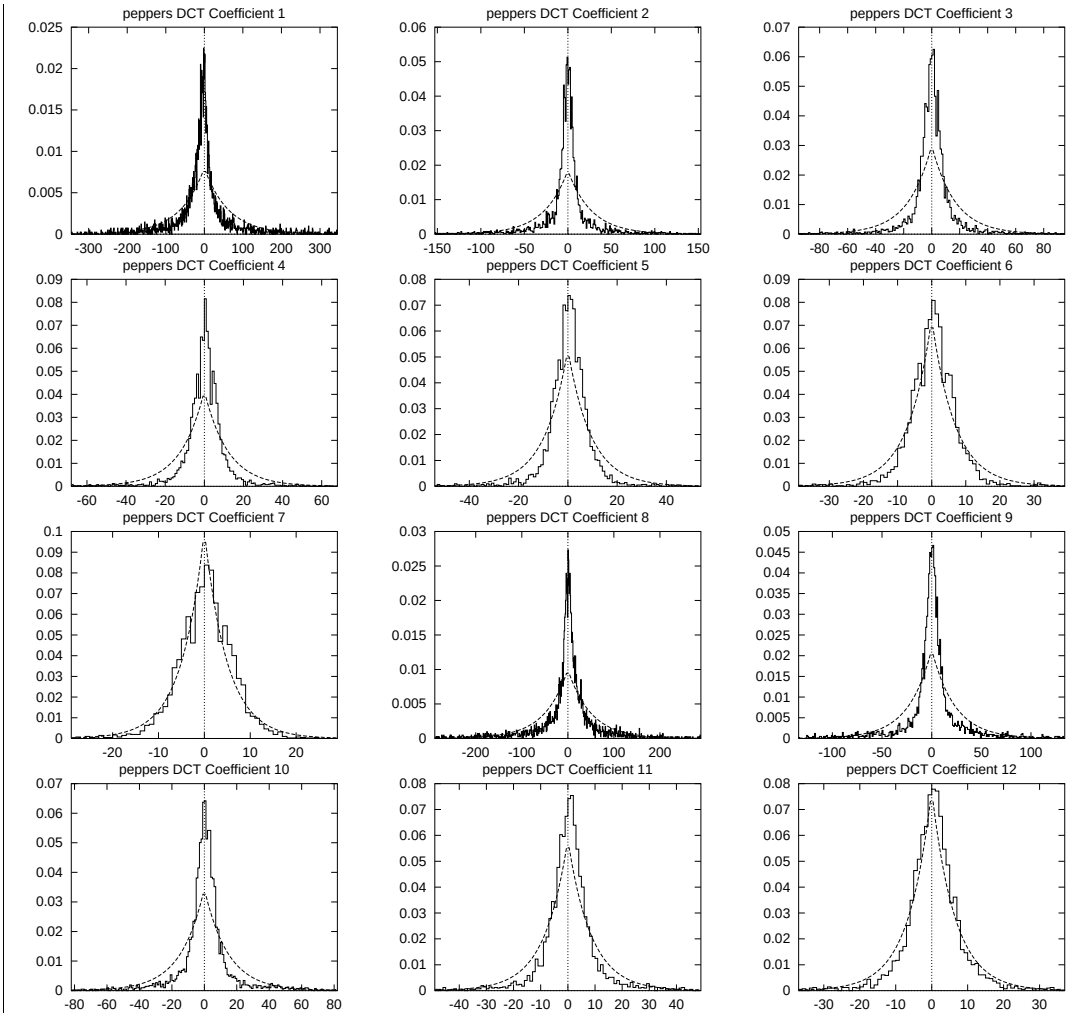
where schem_netlist and lay_netlist are the schematic and layout LVS netlists, xref.out is the cross-reference file produced by LVS and hsp_netlist is the Hspice netlist produced from extracted layout that contains the random net names we wish to replace. The script produces the renamed Hspice netlist in the standard output.

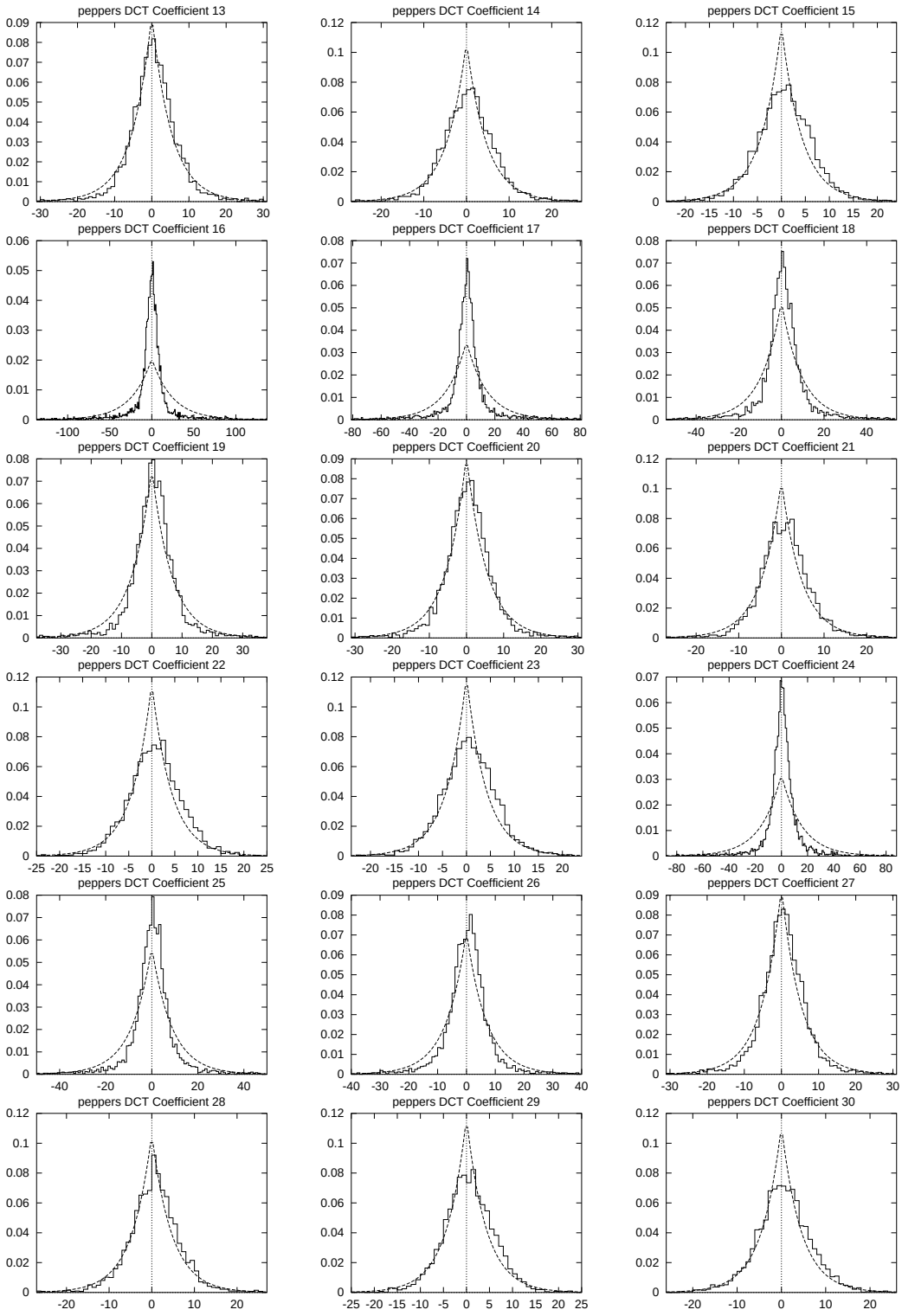
A.8 Summary and Conclusion

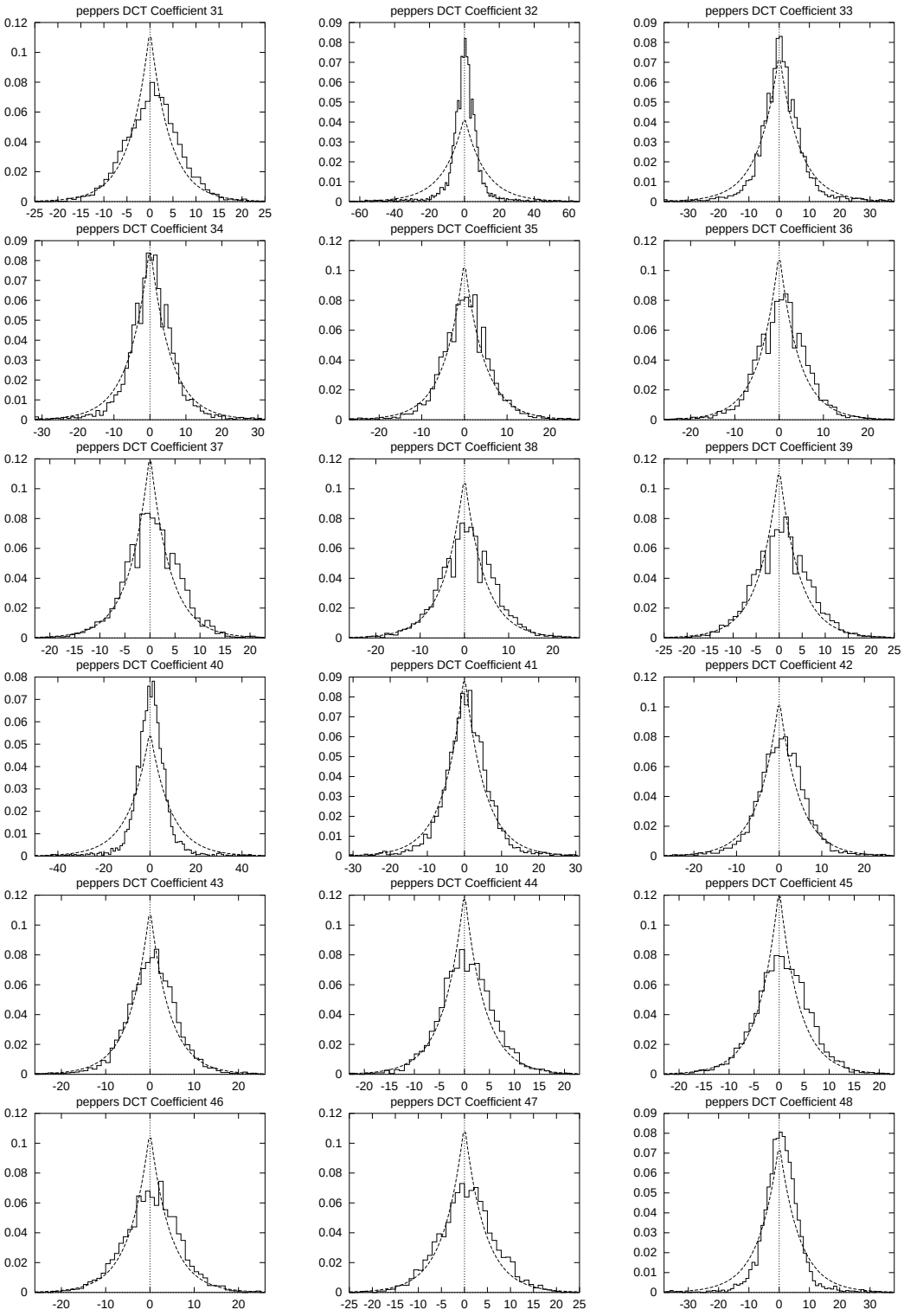
This appendix has described the design flow followed during the development of the DCT and IDCT chips. A number of custom tools have been developed to aid in the process. These tools include a Magic-to-Cadence cell translator, a standard cell layout generator, a net fanout checker, a custom standard cell place and route tool and a netlist processor. Short tutorials on the usage of these tools have been presented.

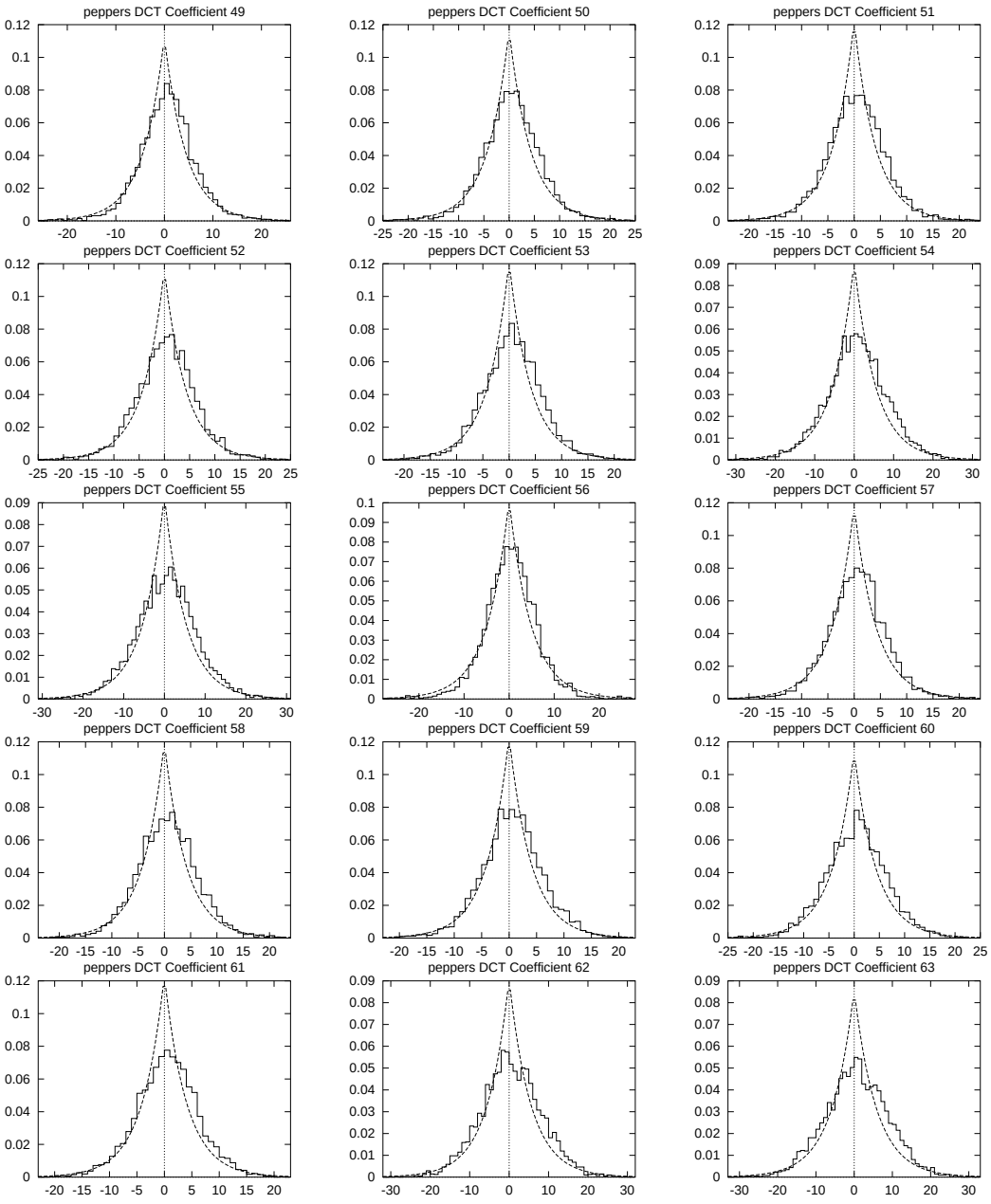
Appendix B

DCT AC Coefficient Distributions: Test Image PEPPERS




peppers DCT Coefficient 13
peppers DCT Coefficient 14
peppers DCT Coefficient 15
peppers DCT Coefficient 16
peppers DCT Coefficient 17
peppers DCT Coefficient 18
peppers DCT Coefficient 19
peppers DCT Coefficient 20
peppers DCT Coefficient 21
peppers DCT Coefficient 22
peppers DCT Coefficient 23
peppers DCT Coefficient 24
peppers DCT Coefficient 25
peppers DCT Coefficient 26
peppers DCT Coefficient 27
peppers DCT Coefficient 28
peppers DCT Coefficient 29
peppers DCT Coefficient 30


peppers DCT Coefficient 31
peppers DCT Coefficient 32
peppers DCT Coefficient 33
peppers DCT Coefficient 34
peppers DCT Coefficient 35
peppers DCT Coefficient 36
peppers DCT Coefficient 37
peppers DCT Coefficient 38
peppers DCT Coefficient 39
peppers DCT Coefficient 40
peppers DCT Coefficient 41
peppers DCT Coefficient 42
peppers DCT Coefficient 43
peppers DCT Coefficient 44
peppers DCT Coefficient 45
peppers DCT Coefficient 46
peppers DCT Coefficient 47
peppers DCT Coefficient 48


peppers DCT Coefficient 49
peppers DCT Coefficient 50
peppers DCT Coefficient 51
peppers DCT Coefficient 52
peppers DCT Coefficient 53
peppers DCT Coefficient 54
peppers DCT Coefficient 55
peppers DCT Coefficient 56
peppers DCT Coefficient 57
peppers DCT Coefficient 58
peppers DCT Coefficient 59
peppers DCT Coefficient 60
peppers DCT Coefficient 61
peppers DCT Coefficient 62
peppers DCT Coefficient 63

Appendix C

Chip Pinout Information

Pin	Type	Signal Name	Pin	Type	Signal Name	Pin	Type	Signal Name
1	-	NC	29	Supply	VLL	57	Supply	VDD
2	Supply	VLL	30	Supply	VDD	58	Supply	GND
3	-	NC	31	Supply	GND	59	Input	din7
4	Supply	VHH	32	Supply	VHH	60	Supply	VHH
5	Output	clkout	33	Supply	VLL	61	-	NC
6	Supply	GND	34	Supply	VDD	62	Supply	VLL
7	Supply	VDD	35	Supply	GND	63	-	NC
8	Output	dout9	36	Supply	VHH	64	-	NC
9	Output	dout8	37	Supply	VLL	65	Input	din8
10	Output	dout7	38	Supply	VDD	66	-	NC
11	Output	dout6	39	Supply	GND	67	Input	clkin
12	Supply	VLL	40	-	NC	68	Input	din9
13	Supply	VHH	41	Input	StartBlockIn	69	Input	din10
14	Output	dout5	42	-	NC	70	Supply	GND
15	Output	dout4	43	-	NC	71	Supply	VDD
16	Output	dout3	44	Supply	VDD	72	Input	din11
17	Output	dout2	45	-	NC	73	Supply	VLL
18	Supply	GND	46	Supply	GND	74	Supply	VHH
19	-	NC	47	Input	reset	75	Supply	GND
20	Supply	VDD	48	Input	din0	76	Supply	VDD
21	-	NC	49	Input	din1	77	Output	StartBlockOut
22	-	NC	50	Input	din2	78	Supply	GND
23	Output	dout1	51	Supply	VHH	79	Supply	VDD
24	-	NC	52	Supply	VLL	80	Supply	VLL
25	Output	dout0	53	Input	din3	81	Supply	VHH
26	Supply	VDD	54	Input	din4	82	-	NC
27	Supply	GND	55	Input	din5	83	Supply	GND
28	Supply	VHH	56	Input	din6	84	-	NC

Table C.1
IDCT Chip Pinout

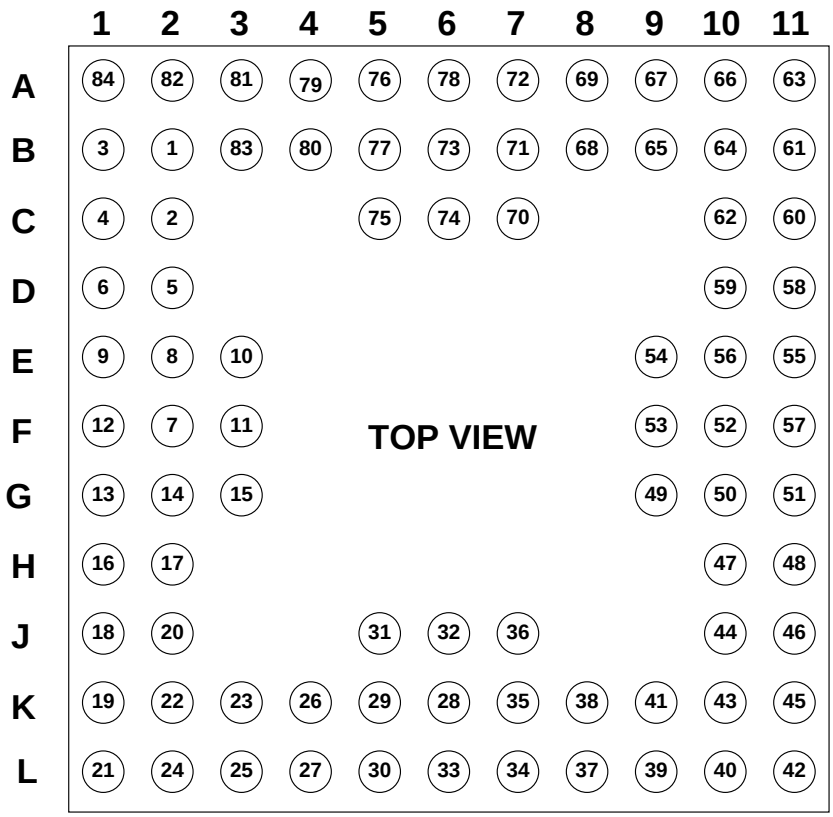


Figure C.1: IDCT Chip Package Footprint

Pin	Type	Signal Name	Pin	Type	Signal Name	Pin	Type	Signal Name
1	Supply	GND	35	Supply	GND	69	Supply	GND
2	Supply	VDD	36	Supply	VDD	70	Supply	VDD
3	Supply	GND	37	Supply	GND	71	Supply	GND
4	Supply	VHH	38	Input	TDI	72	Input	StartBlockIn
5	Supply	GND	39	Input	TCK	73	Supply	GND
6	Supply	VDD	40	Input	TMS	74	Supply	VDD
7	Output	LPEout	41	Input	$\overline{TRST}$	75	Supply	GND
8	Output	dout0	42	Supply	VHH	76	-	NC
9	Output	dout1	43	Supply	GND	77	-	NC
10	Output	dout2	44	Supply	VDD	78	-	NC
11	Output	dout3	45	Output	TDO	79	-	NC
12	Output	dout4	46	-	NC	80	-	NC
13	Output	dout5	47	-	NC	81	Input	clk
14	Output	dout6	48	-	NC	82	Input	reset
15	Output	dout7	49	-	NC	83	Output	StartBlockOut
16	Supply	GND	50	-	NC	84	Supply	GND
17	Supply	VHH	51	Supply	VDD	85	Supply	VHH
18	Output	dout8	52	Supply	GND	86	Supply	GND
19	Output	dout9	53	Supply	GND	87	Supply	VDD
20	Output	dout10	54	Supply	VHH	88	Supply	GND
21	Output	dout11	55	Supply	GND	89	Supply	VHH
22	Supply	GND	56	Supply	VDD	90	Supply	GND
23	Supply	VHH	57	Supply	GND	91	Supply	VDD
24	Supply	GND	58	Input	din0	92	Supply	GND
25	Supply	VDD	59	Input	din1	93	Supply	VDD
26	-	NC	60	Input	din2	94	Supply	VDD
27	-	NC	61	Input	din3	95	-	NC
28	-	NC	62	Input	din4	96	-	NC
29	-	NC	63	Input	din5	97	-	NC
30	-	NC	64	Input	din6	98	-	NC
31	-	NC	65	Input	din7	99	-	NC
32	Supply	VHH	66	Supply	VDD	100	-	NC
33	Supply	GND	67	Supply	GND			
34	Supply	VDD	68	Supply	VHH			

Table C.2
DCT Chip Pinout

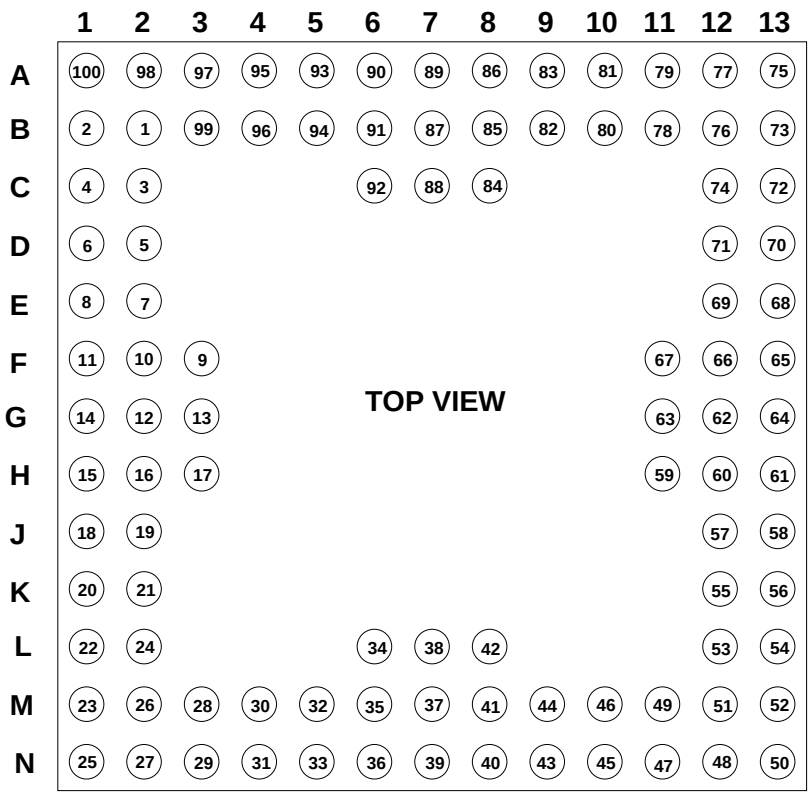


Figure C.2: DCT Chip Package Footprint