7-12-2014

Dynamically Reconfigurable Architectures and Systems for Time-varying Image Constraints (DRASTIC) for Image and Video Compression

yuebing jiang

Yuebing Jiang

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Marios S. Pattichis , Chairperson

Dr. Christos G. Christodoulou

Dr. James R. Lyke

Dr. Constantinos Pattichis

Dynamically Reconfigurable Architectures and Systems for Time-varying Image Constraints (DRASTIC) for Image and Video Compression

by

Yuebing Jiang

B.S., Xi'an Jiaotong University, 2008

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Engineering

The University of New Mexico

Albuquerque, New Mexico

May, 2014

Submitted by: Yuebing Jiang

Supervisor: Dr. Marios S. Pattichs
Department of Electrical and Computer Engineering
University of New Mexico

Committee Members:

Dr. Christos Christodoulou
Department of Electrical and Computer Engineering
University of New Mexico

Dr. James Lyke
Air Force Research Labs

Dr. Constantinos Pattichis
Department of Computer Engineering, University of Cyprus
University of Cypress

Dedication

To my parents, Zuochen Jiang and Xizhi Jiang,
their pure, bright and beautiful love,
supported my stumble in the course of
seeking love and truth.

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Marios S. Pattichis, for his support, advice and encouragement during my years of graduate school. The patience and time spent on me are truly invaluable. Both the research topic and ideas owns large contributions from him. Thanks for supporting me when I needed help.

I would like to thank my dissertation committee for their time and support in reviewing and checking this work.

Thanks to staff and friends in the Department of Electrical and Computer Engineering of UNM, DGNSS Solutions LLC, UNM Hospitals, K&A Wireless LLC, Dolby Laboratory and Real Communications, Inc. A lot of experience was gained from you and I really enjoy our friendship.

Dynamically Reconfigurable Architectures and Systems for Time-varying Image Constraints (DRASTIC) for Image and Video Compression

by

Yuebing Jiang

B.S., Xi'an Jiaotong University, 2008

Ph.D., Engineering, University of New Mexico, 2014

Abstract

In the current information booming era, image and video consumption is ubiquitous. The associated image and video coding operations require significant computing resources for both small-scale computing systems as well as over larger network systems. For different scenarios, power, bitrate and image quality can impose significant time-varying constraints. For example, mobile devices (e.g., phones, tablets, laptops, UAVs) come with significant constraints on energy and power. Similarly, computer networks provide time-varying bandwidth that can depend on signal strength (e.g., wireless networks) or network traffic conditions. Alternatively, the users can impose different constraints on image quality based on their interests.

Traditional image and video coding systems have focused on rate-distortion optimization. More recently, distortion measures (e.g., PSNR) are being replaced by more sophisticated image quality metrics. However, these systems are based on fixed hardware configurations

that provide limited options over power consumption. The use of dynamic partial reconfiguration with Field Programmable Gate Arrays (FPGAs) provides an opportunity to effectively control dynamic power consumption by jointly considering software-hardware configurations.

This dissertation extends traditional rate-distortion optimization to rate-quality-power/energy optimization and demonstrates a wide variety of applications in both image and video compression. In each application, a family of Pareto-optimal configurations are developed that allow fine control in the rate-quality-power/energy optimization space. The term Dynamically Reconfiguration Architecture Systems for Time-varying Image Constraints (DRASTIC) is used to describe the derived systems. DRASTIC covers both software-only as well as software-hardware configurations to achieve fine optimization over a set of general modes that include: (i) maximum image quality, (ii) minimum dynamic power/energy, (iii) minimum bitrate, and (iv) typical mode over a set of opposing constraints to guarantee satisfactory performance. In joint software-hardware configurations, DRASTIC provides an effective approach for dynamic power optimization. For software configurations, DRASTIC provides an effective method for energy consumption optimization by controlling processing times.

The dissertation provides several applications. First, stochastic methods are given for computing quantization tables that are optimal in the rate-quality space and demonstrated on standard JPEG compression. Second, a DRASTIC implementation of the DCT is used to demonstrate the effectiveness of the approach on motion JPEG. Third, a reconfigurable deblocking filter system is investigated for use in the current H.264/AVC systems. Fourth, the dissertation develops DRASTIC for all 35 intra-prediction modes as well as intra-encoding for the emerging High Efficiency Video Coding standard (HEVC).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital image and video coding are computing intensive operations. Given the strong and still growing demand for processing and communications of digital images and videos, there is strong interest in the precise control of video communications, especially under certain constraints. The development of a platform that can provide fine control on power, bitrate, and image quality, we can lead to significant improvements in image and video processing and communications applications (e.g., video conferencing, remote sensing, home surveillance, web browsing).

Real-time constraints can come from many different sources. For example, video communication systems need to meet real-time constraints on available network bandwidth. This requires that the video be adaptively compressed at different bitrates so as to allow for communication within the available bandwidth. Energy constraints can be especially tight on mobile devices. For example, a low power mode is needed when no recharge station is available. Furthermore, a user's interest in video content determines minimum levels of desired video quality.

The use of Dynamic Partial Reconfiguartion (DPR) allows FPGA devices to change hardware configuration of different parts of the device without the need for a restart [4]. On

an FPGA, we can use bitstreams to reconfigure the configurable logic blocks(CLBs), input-output blocks(IOBs), block RAMs, clock resources, and also re-route signals. For Xilinx FPGAs, the configuration memory is organized into one-bit wide frames, where each frame can be written to and read from independendly. Thus, changing a small portion of the device can be performed quickly using a small size bitstream that can be reconfigured in a short time. For video processing applications, given the large amounts of data involved, small configuration bitstreams do not impose significant overhead.

1.1 Motivation

Traditional RD [5] optimization in video coding can significantly benefit from a redesign that incorporates power [6–8] or complexity [9–11] considerations, so as to extend battery life for mobile devices. For software implementations, power consumption can be modeled as a function of supply power and video coding mode (e.g., block search mode, frame-encoding type, etc), or a memory access power model [8]. Also for software implementations, complexity is usually measured as executing time or using a weighted sum of different complexity measures, (e.g., based on time, memory access complexity, parallelization, etc [11]). For hardware-software co-design, especially when using DPR [12], we can directly estimate power based on specific hardware configuration. Furthermore, in this case, effective control can be based on the use of Rate-Distortion-Power (RDP) optimization and constraints.

1.2 Thesis statement

This Ph.D. dissertation research will develop a DPR-based video processing architecture for video communications applications. The architecture system will be evaluated in terms of its ability to dynamically balance trade-offs among dynamic power/energy, bitrate, and image reconstruction quality. In particular, the research will be focused on the development of a

DCT implementation for JPEG/MJPEG compression, a Deblocking filter for H.264/AVC, and intra-encoding modes for the emerging HEVC standard. The thesis provides a significant advancement over the standard use of rate-distortion (RD) optimization by considering rate-distortion/quality-power/energy (RDP) space optimization and control.

1.3 Contributions

The dissertation will demonstrate the use of real-time optimization of joint software-hardware configurations for both still image and video compression. We use the term Dynamically Reconfigurable Architectures for Time-varying Image Constraints (DRASTIC) to summarize the developed architectures. A list of the main contributions of this dissertation includes:

- **JPEG Image Compression Using Quantization Table Optimization Based on Perceptual Image Quality Assessment:**
  A multi-objective stochastic optimization framework is presented for optimizing the quantization table for rate-SSIM curves (extending RD curves). The results show improved performance across a range of rates.

- **DRASTIC DCT for Motion-JPEG:**
  The dissertation introduces a real-time hardware reconfiguration framework that can be used to select and implement jointly-optimal (in the multi-objective sense) hardware realizations and associated software parameters for meeting real-time constraints for communications systems. Real-time constraints are grouped into four fundamental modes based on (i) minimum dynamic power, (ii) minimum bitrate mode, (iii) maximum image quality mode, and (iv) a typical mode that attempts to balance among multiple constraints. Real-time switching among the fundamental modes is demonstrated on motion-JPEG. Overall, the proposed approach yields significant savings over the use of comparable static architectures.

- **Reconfigurable Deblocking Filter for H.264/AVC**

  The dissertation introduces a scalable design approach that allows dynamic hardware reconfiguration of different modes based on power (or hardware complexity), bitrate, and image reconstruction quality. The modes are arranged hierarchically. A complex mode includes all of the deblocking filtering options of simpler modes. This provides scalable performance where the use of additional hardware resources (or power) result in better rate- distortion performance. In the optimal mode, the mode with the best reconstruction quality is selected for each video frame. The optimal mode provides an upper bound (in the rate-distortion sense) to what can be achieved with the current modes.

- **DRASTIC for HEVC intra-prediction mode implementation:**

  The dissertation presents a unified hardware architecture for implementing all 35 intra-prediction modes that include the planar mode, the DC mode, and all angular modes for all prediction unit (PU) sizes ranging from 4×4 to 64×64 pixels. This includes the use of a unified reference sample indexing scheme that avoids the need for sample re-arrangement suggested in the HEVC reference design.

- **DRASTIC for intra-encoding HEVC at the frame level:**

  A software-only implementation of DRASTIC for HEVC intra-encoding is demonstrated for optimizing complexity-rate-distortion for different operating modes.
  DRASTIC optimization involves dynamically reconfiguring parameters (e.g., the quantization parameter, encoding configuration modes) in software for achieving fine optimization control. The fine control achieved with the use of the DRASTIC modes is shown to perform significantly better than the standard use of fixed profiles.

1.4 Dissertation Overview

The dissertation is organized into 8 chapters based on the contributions of the research:

- **Chapter 1: Introduction**

  This chapter provided the motivation, thesis statement, and a summary of the contributions of the dissertation.

- **Chapter 2: JPEG Image Compression Using Quantization Table Optimization Based on Perceptual Image Quality Assessment: [13]**

  This chapter uses perceptual image quality assessment (SSIM) for quantization table (QT) optimization for JPEG compression. This leads to the study of rate-SSIM curves that replace the traditional use of rate-distortion curves based on the PSNR. The chapter introduces the use of a multi-objective optimization framework for estimating the best rate-SSIM curves. To estimate globally optimal quantization tables, a stochastic-optimization algorithm based on Simulated Annealing is proposed and its variations are studied.

- **Chapter 3: DRASTIC DCT for MJPEG: [14], [15]**

  The basic DRASTIC optimization modes are introduced in this chapter and are demonstrated on the use of DCT compression.

- **Chapter 4: A Dynamically Reconfigurable Deblocking Filter for H.264/AVC Codec**

  A scalable deblocking filter is introduced that is jointly optimal in terms of bitrate, dynamic power, and image reconstruction quality.

- **Chapter 5: High Efficiency Video Coding (HEVC):**

  The chapter provides a basic introduction to HEVC.

- **Chapter 6: DRASTIC for HEVC intra-prediction mode implementation:**

  The chapter provides a description of how to implement the 35 intra-prediction modes associated with HEVC.

- **Chapter 7: DRASTIC for HEVC intra-encoding at the Frame Level:**

  This chapter focuses on the use of DRASTIC modes for HEVC intra encoding. The

approach focuses on the development of scalable prediction modes.

- **Chapter 8: Conclusion and Future Work:**
  The final chapter provides a summary of the dissertation, provides concluding remarks, and a summary for future work in this area.

A list of the publications that have resulted from the Ph.D. dissertation are given in the Appendix.

Chapter 2

JPEG Compression Using Quantization Table Optimization Based on Perceptual Quality

2.1 Abstract

We consider the use of perceptual image quality assessment for quantization table (QT) optimization for JPEG compres- sion. For evaluating performance, we consider the use of the Structural Similarity Index (SSIM) for evaluating distortion in the compressed images. This leads to the study of rate-SSIM curves that replace the traditional use of rate-distortion curves based on the PSNR.

We introduce a multi-objective optimization framework for estimating the best rate-SSIM curves. To estimate glob- ally optimal quantization tables, A stochastic-optimization algorithm based on Simulated Annealing is proposed and its variations are studied.

We report results on all methods on the Lena image and results from selected methods on the LIVE image quality as- sessment database. For the LIVE database, compared to the use of

the standard JPEG quantization table at quality factor QF=95, QTs based on the training set give average bitrate re- ductions of 11.68%, 7.7% and an increase of 2.4%, while the SSIM quality changes from -0.11%,+0.05% and 0.12% respectively. In all cases, the results indicate that all considered methods improved over the use of standard JPEG tables.

2.2 Introduction

There is strong interest in the optimization of image compression methods based on objective methods for evaluating perceptual image quality. This renewed interest comes from the realization that the PSNR is a very poor measure of perceptual image quality [16] [17]. Thus, traditional rate-distortion curves based on PSNR cannot accurately reflect perceptually-optimal image compression. Instead, we consider the replacement of the PSNR by the Structural Similarity Index (SSIM) and the use of rate-SSIM curves for the evaluation of the performance of image compression methods (see [18] for SSIM).

For development of a perception-motivated optimization method for still image compression, Discrete Cosine Transform (DCT) coefficient quantization tables has significant role on the compression performance. In terms of applications, our current focus is on the use of the optimal quantization tables with JPEG [19].Here, we note that despite the introduction of the JPEG 2000 standard, JPEG remains the most common still image compression standard. However, it is important to note that our approach is very general and its extension to application with JPEG 2000 or other image and video compression methods should be straight-forward.

Some related work appears in [17]. In [17], the authors derived SSIM bounds as a function of the quantization rate for fixed-rate uniform quantization of image DCT coefficients, for high data rates. Here, we will consider rate-SSIM optimization for high SSIM index values. The SSIM is used for the design of optimal linear equalizers in [16] and optimal linear restoration in [20].

The rest of the chapter is organized as follows. We present a multi-objective optimization formulation in section 2.3. This is followed by the methodology in section 2.4. The results are presented in section 2.5, and concluding remarks are given in section 2.6.

2.3 A Multi-Objective Optimization Formulation

We begin this section by a brief overview of the multi-objective optimization approach. We then discuss a stochastic optimization approach for solving the generated equations. We consider the multi-objective optimization problem defined by

$$\min_{Q} \text{Rate}\,(Q,\, I), \quad \max_{Q} \text{SSIM}\,(Q,\, I) \tag{2.1}$$

where I denotes the input image, $Q = (Q_1, \ldots, Q_{64})$ defines the Discrete Cosine Transform (DCT) quantization table parameters, Rate denotes the achieved bitrate in average bits per sample, SSIM(.) refers to the SSIM index. Here, the quantization table elements are integers with values from 1 to 255.

To solve (2.1), we need to find Pareto optimal points. Here, a solution is defined to be Pareto optimal if no other feasible point can be found that has both a lower bitrate and a higher SSIM index. The set of solutions generated by this approach generates the rate-SSIM graph.

To estimate the Pareto optimal points, we consider a scalarization of (2.1), given by:

$$\max_{Q} \text{SSIM}\,(Q,\, I) - C_1 \text{Rate}(Q,\, I), \quad C_1 > 0 \tag{2.2}$$

Where C_1 denotes a scalar constant that provides for the relative weight between the conflicting objectives.

For initializing the optimization procedure, we will consider the standard JPEG quantization table defined in terms of the Quality Factor (QF) parameter. Thus, to generate the

rate-SSIM graph, we will consider initializing the search at multiple values of the QF. Note that for each value of the QF, we also have corresponding initial values for the SSIM and Rate index. This approach also provides for a method for estimating the scalar quantization parameter C_1. We estimate C_1 of (2.2) using central differencing:

$$C_1 \approx \frac{S_{QF_{i+1}} - S_{QF_{i-1}}}{R_{QF_{i+1}} - R_{QF_{i-1}}} \tag{2.3}$$

where $(S_{QF_{i+1}}, R_{QF_{i+1}})$ and $(S_{QF_{i-1}}, R_{QF_{i-1}})$ denote the initial SSIM and rate values for $QF = QF_{i+1}, QF_{i-1}$ respectively. It is interesting to note that (2.3) represents a slightly biased estimator since we expect the slope to increase as compared to standard JPEG.

Since the Quantization table is made up of integer values, we cannot consider Newton or other continuous-variable optimization method. Furthermore, due to the complexity of the search space, an integer programming approach will be intractable. Instead, we consider a global optimization approach based on Simulated Annealing(SA) [21].

2.4 Methods

Simulated Annealing estimates the optimal value by forming a probability density function in terms of the function to be optimized. A Markov Chain of possible quantization table (QT) values is generated based on a transition probability defined between different QT values. The key to speed up convergence is to define an appropriate transition probability that will move the QT values closer to the optimal value in fewer iterations. Here, we will consider four different methods and then derive the transition probabilities for the most promising method.

In summary, we consider the following methods for generating the Markov Chain:

- Method 1: Uniformly choose the QT coefficient index i to modify. Then modify Q_i by $+1$ or -1 with equal probability.

- Method 2: Use a low-frequency exponential rule to select the QT coefficient index i to modify (low-frequencies selected more frequently). Then modify Q_i by $+1$ or -1 with equal probability.
- Method 3: Uniformly choose the QT coefficient index i to modify. Then modify Q_i by drawing a sample from a discrete Gaussian distribution with $\sigma = 1$.
- Method 4: Use a low-frequency exponential rule to select the QT coefficient index i to modify. Then modify Q_i by drawing a sample from a discrete Gaussian distribution with $\sigma = 1$.
- Method 5:Use a high-frequency exponential rule to select the QT coefficient index i to modify (high frequencies selected more frequently). Then modify Q_i by $+1$ or -1 with equal probability.

Method 1 is fairly straight-forward. Every possible neighbor is considered with equal probability, and the change between neighbors is with the smallest step. Methods 2, 4 and 5 use an exponential rule for deciding which QT coefficient should be changed next. To explain the exponential choice rule, let us consider the 8×8 DCT coefficient grid. We define the exponential choice rule based on:

$$f(i,\, j) = D \cdot \exp\left[-c(i+j)/15\right], i, j \in [1, \ldots, 8], \tag{2.4}$$

Where D is an appropriate normalization constant so that the discretized probabilities add up to 1. By abuse of notation, we thus select the $(i,\, j)$-th QT coefficient to modify based on $f(i,\, j)$. To return to the 1-D indices, simply map $(i,\, j)$ to the 1-D index given by $8(i-1)+j$. Furthermore, we map the 1-D index k to $(i,\, j)$ using $j = k \mod 8$ and $i = (k-j)/8 + 1$. We can use the exponential rule in (2.4) to encourage the selection of low-frequencies using c positive (methods 2 and 4 in section 7.5 use $c = 0.5$), or for encouraging the selection of high-frequencies using c negative,(method 5 in section 7.5 uses $c = -0.5$). In what follows, we let $f_E(.)$ denote the one-dimensional exponential distribution.

Method 3 allows for both small and larger changes in the QT coefficients based on a discretized Gaussian distribution. Method 4 combines the exponential rule of method 2 with the Gaussian changes of method 3. Method 5 is the same as method 2 but uses an exponential rule that verifies the selection of high-frequencies.

It can be shown that the use of the exponential rule in method 4 leads to the transition probability given by

$$f(Q^+, Q) = \sum_{i=1}^{i=64} \mathcal{N}_{Q_i^+}(\mu = Q_i, \sigma = 1) \cdot \delta(Q_i^+ = Q_i) \cdot f_E(i)$$

where Q denotes the current QT state, Q^+ denotes next state in the Markov Chain, $\mathcal{N}_{Q_i^+}$ denotes the one-dimensional discrete Gaussian distribution for the i-th coefficient of Q^+. Also, $\delta(Q_i^+ = Q_i)$ means that Q^+ keeps all of the coefficients of Q except for the i-th coefficient.

After generating the next state Q^+ from Q, the Simulated Annealing algorithm will jump to the next state with probability [21]:

$$min\{1, exp(\lambda \cdot (O^+ - O))\} \tag{2.5}$$

where λ denotes a temperature parameter, O^+ represents the optimization function given by (2.2) evaluated at Q^+, and O represents (2.2) evaluated at Q.

The entire algorithm is summarized in Fig. 1. To generate the entire rate-SSIM graph, we need to run this algorithm for different values of the quality factor QF. For each value of QF, we may generate several Pareto-optimal points that outperform the standard JPEG QT.

2.5 Results

We present results for the Lena image and the LIVE image quality database [22,23]. We will first use the Lena image to discuss the proposed methods. Then, we provide a summary of

Figure 2.1: Simulated Annealing (SA) Optimization for computing a Rate-SSIM optimal point.

Input: Image I, Quality Factor (QF), SA parameter C_0.
Output: Optimal quantization table Q.

1: Initialize Q to the standard JPEG quantization table for the specified quality factor QF. Compute the standard rate R_0 based on the optimal Huffman tables. Set $i = 1$.
2: Compute multi-objective parameter C_1 using Eq. (2.3).
3: Use method 1, 2, 3, 4, or 5 to select the next state Q^+ from the current state Q.
4: Compute the optimal Huffman table based on Q^+.
5: Compute the objective function O^+ based on Eq. (2.2).
6: Set $\lambda = C_0 * ln(1 + i)$, and decide whether to transition to Q^+ based on the transition probability defined by Eq. (2.5).
7: Set $i = i + 1$ for the next iteration.
8: Repeat 3 to 7 until i reaches the maximum number of iterations $MaxIterations$

the performance on the LIVE image quality database on methods 1, 2 and 5 in Tables 2.6, 2.7 and 2.8.

As described in the Methods section, we consider five different methods. We have results in Tables 1-5 from Methods 1-5 respectively. In all cases, we consider JPEG compression for the Lena image for Quality factor values: $QF = 85, 90, 95$ to initialize the search. Furthermore, we set the SA parameter at $C_0 = 5000$ for a maximum of 600 iterations. The final results are with the best object function evaluations during each iteration.

Table 2.1: Method 1 results for the Lena (512×512) image.

QF	Final rate	Final SSIM	Rate change	SSIM change
95	2.3717	0.9814	**-2.00%**	**0.36%**
90	1.5164	0.9621	-8.07%	-0.02%
85	1.2133	0.9523	-6.76%	-0.07%

In Figures 2.2, 2.3, 2.4, we present the Pareto-optimal fronts for all 5 methods. In Fig. 2.2, we have the Pareto-optimal results for QF=95. Note that method 1 moves orthogonal to the original Rate-SSIM graph as intended. The rest of the methods either move above

Table 2.2: Method 2 results for the Lena (512 × 512) image based on the low-frequency exponential rule.

QF	Final rate	Final SSIM	Rate change	SSIM change
95	2.3036	0.9804	**-4.81%**	**0.25%**
90	1.5138	0.9617	-8.23%	-0.05%
85	1.1985	0.9524	-7.89%	-0.06%

Table 2.3: Method 3 results for the Lena (512 × 512) image based on discretized Gaussian model.

QF	Final rate	Final SSIM	Rate change	SSIM change
95	2.1843	0.9766	-9.74%	-0.13%
90	1.3341	0.9568	-19.12%	-0.56%
85	1.0812	0.9478	-16.90%	-0.55%

Table 2.4: Method 4 results for the Lena (512 × 512) image based on the low-frequency exponential rule and discretized Gaussian model.

QF	Final Rate	Final SSIM	Rate change	SSIM change
95	2.0840	0.9759	-13.89%	-0.20%
90	1.3491	0.9575	-18.21%	-0.49%
85	1.0975	0.9486	-15.66%	-0.46%

Table 2.5: Method 5 results for the Lena (512 × 512) image based on the high-frequency exponential rule.

QF	Final Rate	Final SSIM	Rate change	SSIM change
95	2.6117	0.9824	7.92%	0.47%
90	1.6592	0.9626	0.59%	0.04%
85	1.3015	0.9531	0.02%	0.02%

or below method 1. This trend seems to also occur in Figures 2.3 and 2.4. In Fig. 2.4, it is interesting to note that an entire, nearly-continuous Pareto front was generated from the use of the 5 methods for $QF = 85$. Overall, we select Methods 1, 2 and 5 to run with the LIVE database. Our choice is based on generating a diverse set of results as shown in Fig.2.

In Fig. 2, it is clear that methods 2 and 5 seem to go into different optimization directions. It is also interesting to note that method 2 favors low-frequency changes while method 5 favors high-frequency changes, while method 1 becomes the same as methods 2 and 5 for $c = 0$. It is easy to see that Method 1 is much better than the orginal standard QT since it produces a decrease in bitrate and an increase in the SSIM index. Method 2 can cause significant bitrate reduction, but this comes at a slight decrease in the SSIM index. Method 5 can lead to an increase in the SSIM index at a slightly increases bitrate, The methods using Gaussian models don't show significant difference from the methods using the exponential rule. Generally, by adjusting c in the exponential rule and C_1 in (2.2), the methods behave similarly. However, in all cases, the proposed methods yield improvements over the use of standard JPEG QTs (e.g., see Figs. 2-4).

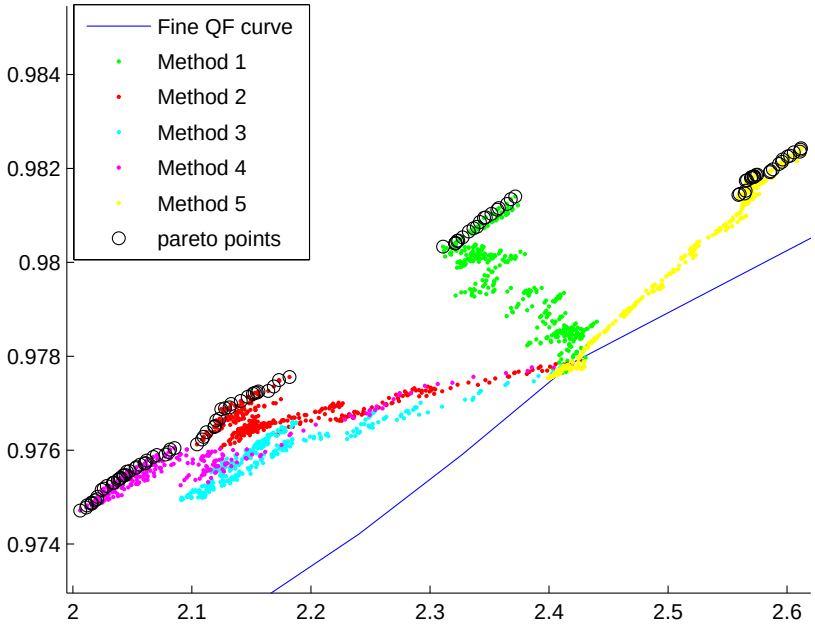


Figure 2.2: Rate-SSIM comparisons for all methods for the Lena image at QF=95. The circled points represent Pareto-optimal results.

We use the leave one out method to test methods 1, 2 and 5 on the LIVE database. Here,

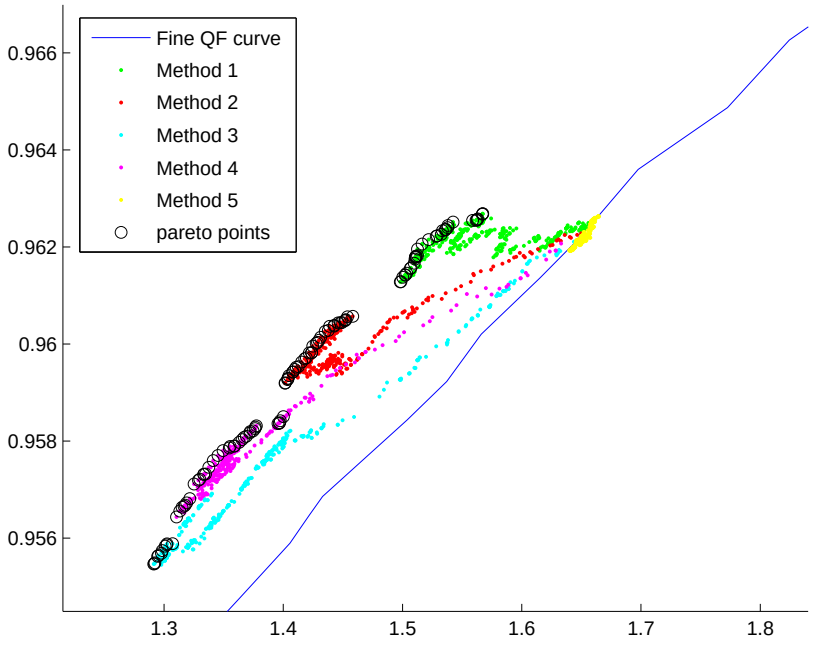


Figure 2.3: Rate-SSIM comparisons for all methods for the Lena image for QF=90. The circled points represent Pareto-optimal results.

we use 28 images for training and the remaining one for testing. In training, we compute the optimal quantization table for each one image and then take the median of all of them. During testing, the median quantization table is used on the remaining image. The results for methods 1, 2 and 5 are given in Tables 6 and 7 8 respectively.

From Table 6, it is clear that method 1 produced better results. The median quantization tables gave a rate decrease as well as an SSIM improvement. On average, we have a rate decrease of 7.7% and an SSIM increase of 0.06%. For method 2, we had an average rate decrease of 11.68% at a slight SSIM decrease of 0.11% (see Table 7). Method 5 emphasizes on SSIM improvement, with an average 0.12% increase of the SSIM, but also, the average bitrate increase 2.4%.

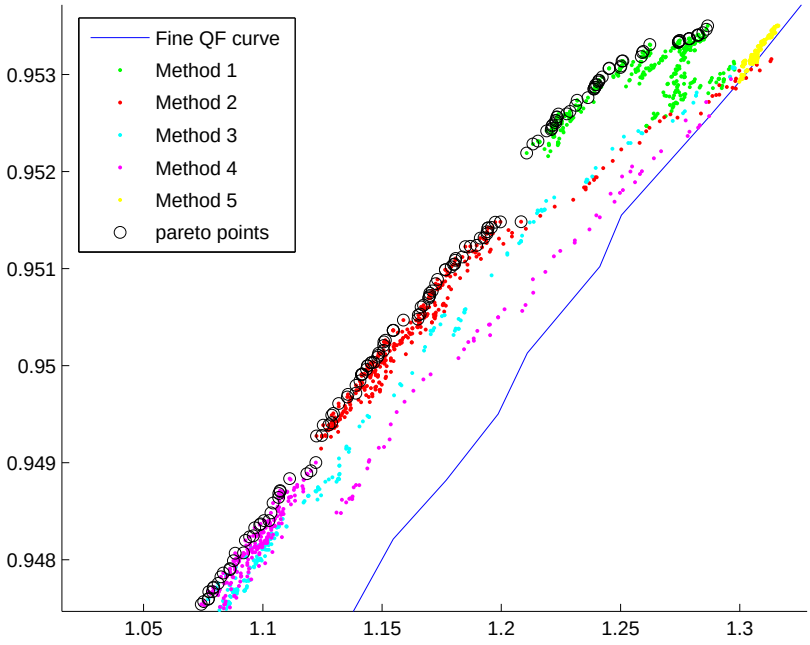


Figure 2.4: Rate-SSIM comparisons for all methods for the Lena image at QF=85. The circled points represent Pareto-optimal results.

Table 2.6: Method 1 results for the LIVE database using leave one out validation on the median QT (QF=95). Here, we report the average, and the percentile results (min, 25th, 50th, 75th, max) on rate change.

File Name	Final Rate	Final SSIM	Rate change	SSIM change
Average			-7.70%	0.06%
'parrots'	1.5036	0.9817	-13.14% (min)	-0.09%
'sailing3'	2.1073	0.9802	-8.72% (25%)	0.06%
'sailing1'	2.7234	0.9885	-7.22% (50%)	0.07%
'cemetry'	3.3588	0.9910	-6.41% (75%)	0.11%
'building2'	4.2494	0.9953	-5.43% (max)	0.08%

2.6 Conclusion

In this paper, we present a multi-objective optimization framework for JPEG image compression based on perceptual image quality assessment. In particular, we consider the use

Table 2.7: Method 2 results for the LIVE database using leave one out validation on the median QT (QF=95). Here , we report the average, and the percentile results (min, 25th, 50th, 75th, max) on rate change.

File Name	Final Rate	Final SSIM	Rate change	SSIM change
Average			-11.68%	-0.11%
'parrots'	1.4204	0.9802	-17.95% (min)	-0.24%
'womanhat'	2.1189	0.9799	-13.54% (25%)	-0.17%
'lighthouse2'	2.4562	0.9805	-11.28% (50%)	-0.14%
'woman'	3.0394	0.9829	-9.70% (75%)	-0.07%
'building2'	4.1291	0.9942	-8.10% (max)	-0.02%

Table 2.8: Method 5 results for the LIVE database using leave one out validation on the median QT (QF=95). Here, we report the average, and the percentile results (min, 25th, 50th, 75th, max) on rate change.

File Name	Final Rate	Final SSIM	Rate change	SSIM change
Average			2.40%	0.12%
'flowersonih35'	3.8631	0.9963	1.64%(min)	0.05 %
'bikes'	3.6631	0.9930	2.11%(25%)	0.10%
'rapids'	3.3482	0.9886	2.40%(50%)	0.17%
'lighthouse'	2.8465	0.984	2.62%(75%)	0.14%
'sailing2'	2.2470	0.978	3.39%(max)	0.17%

of rate-SSIM graphs for replacing the traditional use of rate-distortion graphs based on PSNR. As a particular example of this framework, we consider the optimization of the DCT coefficient quantization table.

For high quality factors, as compared to the use of the standard JPEG quantization tables, the optimized quantization tables form new rate SSIM curves that represent significant improvements. Our approach can also be extended to cover other image and video compression standards.

Chapter 3

DRASTIC DCT for MJPEG

3.1 Abstract

Video communications dominate bandwidth requirements in real-time communications systems. Yet, constraints on real-time communications can vary based on available bandwidth, power, or the required level of image quality. This chapter introduces a real-time hardware reconfiguration framework that can be used to select and implement jointly-optimal (in the multi-objective sense) hardware realizations and associated software parameters for meeting real-time constraints for communications systems. Beyond standard rate-distortion optimization, the approach is demonstrated in jointly optimizing dynamic power consumption, the quality of the reconstructed image, and the required bitrate. Real-time constraints are grouped into four fundamental modes based on (i) minimum dynamic power, (ii) minimum bitrate mode, (iii) maximum image quality mode, and (iv) a typical mode that attempts to balance among multiple constraints.

Real-time switching among the fundamental modes is demonstrated on motion-JPEG. To meet the real-time constraints, the system uses dynamic partial reconfiguration over a set of 2D DCT modules. The scalable 2D DCT modules compute lower magnitude DCT frequen-

cies using zonal control at different bitwidths (2-9 bits). In addition to hardware control, the system adjusts the quality factor (a software parameter) to generate a total of 1280 configurations that include 841 that are Pareto optimal. Beyond the fact that the proposed proposed approach is scalable in dynamic-power, image quality, and bit rate, it also provides full 2D DCT calculation that is at least as good or significantly better than any previously published approach. A scalable, real-time controller is used for selecting an appropriate configuration so as to meet time-varying constraints. The real-time controller is shown to satisfy the constraints of the fundamental communications modes as well as to be able to effectively switch configurations to follow mode changes. Overall, the proposed approach yields significant savings over the use of comparable static architectures.

3.2 Introduction

The performance of video communication systems depends on balancing requirements associated with the network, the user experience, and the video display device. For example, the network imposes bandwidth constraints. On the other hand, users require sufficient levels of video quality. For display on mobile devices, it is also important to conserve power. Often, the constraints can lead to opposing requirements. For example, delivering higher video quality requires higher levels of power and bandwidth. This paper describes a dynamically reconfigurable system that allows the users to meet real-time constraints on image quality, dynamic power consumption, and available bandwidth. More generally, the term *Dynamically Reconfigurable Architecture System for Time varying Image Constraints* (DRASTIC) is used to describe a video processing system that can meet real-time constraints through dynamic reconfiguration.

Four fundamental communications modes can be used to summarize the requirements for optimal performance subject to real-time constraints:

- **Minimum dynamic power mode:** The goal is to minimize dynamic power subject to available bandwidth and a minimum level of acceptable image quality. In this mode, a mobile device can reduce it's power requirements without sacrificing the user experience. Furthermore, in this mode, a mobile device can conserve energy and maximize its operating time.

- **Minimum bitrate mode:** The goal is to minimize bitrate requirements subject to a maximum level of dynamic power and a minimum level of acceptable image quality. Thus, the user can enjoy the compressed video without sacrificing video quality. Furthermore, since the bitrate is minimized, the network can accommodate a large number of users without sacrificing the service.

- **Maximum image quality mode:** The goal is to maximize image quality without exceeding the maximum available bandwidth or the maximum available dynamic power. In this mode, the user will be able to examine the video at the maximum possible video quality that can be delivered by all available bandwidth and computing power.

- **Typical mode:** In this mode, the goal is to optimize a weighted average of the required dynamic power, bitrate, and image quality within constraints on all of them. Here, we have a balanced approach that supports trade-offs between dynamic power, quality, and bitrate.

Clearly, by selecting appropriate weights in the typical mode, we can achieve the performance of the other three modes. Yet, we still focus on the different modes in order to emphasize the user requirements for minimizing power, bitrate, or maximizing quality.

The video encoding system is demonstrated on motion JPEG (MJPEG). The focus on motion JPEG is motivated from its relatively low complexity (e.g., see [24]) that make it popular in low profile webcams, surveillance systems [25, 26] and emerging applications in virtual network computing (VNC) [27]. Clearly though, the concepts that are introduced

here can be applied to any video coding standard. In fact, the use of the Discrete Cosine Transform (DCT) and quantization tables is shared by all video coding standards.

The basic system is shown in Fig. 4.1. The approach is demonstrated in the compression of the Y-component of color video. Here, a joint software-hardware optimization system uses a Dynamic Reconfiguration (DR) controller to select DCT hardware cores and quality factor (QF) values to meet constraints in bitrate, image quality, and power. To solve the optimization problem, the system relies on the use of feedback from the current bitrate, image quality measured using the structural similarity index metric (SSIM), and pre-computed dynamic-power consumed by an adaptive DCT IP core. A dynamic reconfiguration (DR) controller compares the current bitrate, image quality, and power with the required levels to determine if constraints are met. Depending on each optimization mode, a suitable DCT IP core and quality-factor value is selected for the next video frame. Alternatively, the dynamic reconfiguration overhead can be reduced by fixing the hardware configuration over a number of video frames or until a maximum number of reconfigurations has been met.

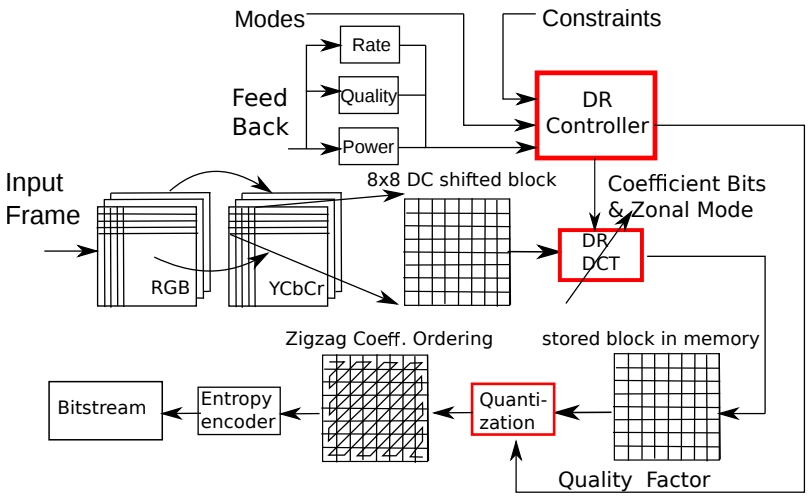


Figure 3.1: Dynamically Reconfigurable Architecture System for Time varying Image Constraints (DRASTIC) for motion JPEG.

The basic contributions of the paper are summarized as follows:

- *DRASTIC optimization modes for video communications:* The paper introduces new real-time optimization approaches that can be used to minimize dynamic power, maximize image quality, reduce bitrate, or provide balanced solutions for meeting real-time video constraints. This approach extends traditional rate-distortion optimization approaches that do not consider dynamic energy or power constraints or the use of image quality metrics (e.g., SSIM [18]).

- *Scalable, Pareto-optimal DCT cores with quantization control:* A scalable architecture is used to generate a family of hardware cores that can be used to compute lower-frequency subsets of the DCT frequencies using different bit-widths. The approach is motivated by the observation that significant compression can be achieved through the effective quantization of high-frequency components. Thus, in addition to the scalable DCT cores, we investigate the use of different quality factors that control the DCT quantization tables. This results in a joint software-hardware optimization approach. Yet, not all generated configurations will necessarily be useful. We use a training set to determine the hardware configurations that are Pareto optimal.

- *Scalable dynamic reconfiguration controller based on feedback:* A dynamic reconfiguration controller is used for meeting real-time constraints through a joint optimization of the software-hardware configuration. In real-time, the controller selects the active DCT core and quality factor from a set of pre-computed, Pareto-optimal configurations. After a selection is made, real-time feedback is used for adjusting the DCT core and the quality factor. Dynamic reconfiguration overhead is controlled in a scalable fashion by adjusting the number of video frames between configurations or the total number of reconfigurations per 100 frames that can be used. In the case of unrealistic constraints that cannot be met, the controller selects the best solution based on reformulation of the problem using unconstrained optimization.

The rest of the paper is organized as follows. Related work background is given in Section II. The proposed architecture is described in Section III. An implementation of the

architecture is given in Section IV. Hardware testing of proposed architecture is given in Section V. Concluding remarks and future work are given in Section VI.

3.3 Background and related work

3.3.1 Dynamic Reconfiguration Based on Multi-objective Optimization

Some of the basic concepts behind the use of dynamic partial reconfiguration (DPR) for meeting real-time constraints have been recently presented in [28]. In [28], the focus was on the development of a video pixel processor that can be adapted to meet real-time constraints in Power/Energy-Performance-Accuracy. Here, the focus is on the development of a dynamically reconfigurable system that can meet communication constraints through the use of joint software-hardware optimization that involves the Discrete Cosine Transform and the quality factor. The paper also introduces a scalable controller for controlling the reconfiguration overhead. Nevertheless, the current paper shares some of the theory with the work presented in [28]. For completeness, a summary of this related background research will be provided.

To satisfy multi-objective optimization constraints in hardware, there is a need to generate a family of hardware cores that sample different points in the multi-objective space. The Pareto front is computed from the family of the generated hardware realizations. The Pareto front represents the set of optimal configurations. To meet real-time constraints, a dynamic reconfiguration controller selects a Pareto-optimal realization and implements it in hardware using dynamic partial reconfiguration. In this paper, this basic dynamic reconfiguration controller is extended through the use of feedback and scalability so as to control the reconfiguration overhead.

To generate the Pareto-front for the current application, prior research focused on the computation of DCTs that was reported in [29–31]. The basic idea here was to avoid the computation of higher-frequency components by only computing the $N \times N$, $N = 1, 2, \ldots, 8$ lower frequency components. In [29–31], the authors demonstrated the use of this adaptive DCT in an MPEG2 system. Furthermore, some preliminary work related to the current paper was reported in [14,15]. In this prior work, the examples were from applications using still images without considering time-varying constraints in digital video. Beyond digital video, the current paper provides a significant extension beyond this prior research that includes the use of a scalable dynamic reconfiguration controller that uses feedback to meet the time-varying constraints.

3.3.2 Rate-Distortion-Complexity Control

This section provides a summary of prior research focused on controlling computational complexity for real-time video encoding. The section then describes the extensions provided by the proposed research.

A relatively recent attempt to manage real-time computational complexity in video encoding has been described in [32]. In [32], in order to limit computational complexity, the authors recommended dropping video frames while attempting to manage image quality. Overall, this direct approach attempts to manage video quality losses by smoothing frame-rates. As verified by subjective video quality measurements, the managed approach will perform better than a reference encoder.

An approach related to the research presented in this paper has been recently introduced in [6] and further developed in [7]. In [6], the authors were interested in Power-Rate-Distortion (P-R-D) optimization for wireless video communication under energy constraints. Here, the authors use dynamic voltage scaling (DVS) to control power consumption and then investigate the rate-distortion performance under power control. The paper demon-

strates how the system adjusts its complexity control to match the available energy supply while maximizing the picture quality. In [7] the authors show that they can achieve up to a 50 percent reduction in power consumption by adjusting the hardware configuration to follow the non-stationary characteristics of the video. Some of the issues associated with the attempt to use complexity-control with motion estimation have been the focus of more recent research in medium-granularity complexity control (MGCC) reported in [11]. In [11], the authors introduced a rate-complexity-distortion mode for a group of pictures (GOP) to allocate complexity at the frame level.

As noted earlier, for low-energy devices that use MJPEG, motion estimation is avoided. Furthermore, a scalable and parametrizable system based on the DCT and the quality factor, such as the one developed here, can also be applied for motion-compensation of motion-based video coding. Clearly-though, our optimization modes provide a general framework for extending this prior research of minimizing energy to new modes that support maximizing quality and minimizing bitrate while also allowing the user to switch among different modes. The scalable DR controller also allows the user to control the reconfiguration overhead while estimating performance at the frame level.

3.3.3 DCT hardware implementations

This section provides a review of 2D DCT implementations. The review focuses on the complexity of each approach that suggests the need for a separable implementation that allows scalability in the number of accuracy bits and the number of DCT frequencies to be computed.

Let $X_{8\times8}$ represent an input image block after DC shift. Here, note that the DC shift is implemented by subtracting 128 from the unsigned 8-bit input image. The DCT output image is then represented as a 16-bit signed integer given by:

$$Z_{u,v} = C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} X_{i,j} \cos\left(\frac{u(2i+1)\pi}{16}\right) \cos\left(\frac{v(2j+1)\pi}{16}\right) \quad (3.1)$$

where:

$$C_u = \begin{cases} \frac{1}{2\sqrt{2}} & \text{for } u = 0 \\ \frac{1}{2} & \text{for } u > 0 \end{cases}$$

DCT implementations can be classified into the following categories:

- **Direct approaches: [33–39]:** The 2D DCT is implemented using matrix-vector products. Direct methods based on Chen's algorithm [40] are very effective and represent a very popular choice.
- **Distributed arithmetic (DA)-based designs [41–44]:** The 2D DCT result is computed bit by bit by considering the products of the DCT basis functions with the input image block. DA-based designs are inherently bit-serial in nature and this issue cannot be addressed effectively except for the special cases (see [41]). Given the complexity and focus on bit-by-bit computation, DA-based approaches cannot be easily adapted for of computing a limited number of DCT frequencies, as required for DRASTIC.
- **Systolic array (SA)-based designs [30, 45–47]:** The DCT is computed using a relatively-large array of processing elements (PEs) arranged in a systolic array pattern. Unfortunately, SA implementations require significant resources and sophisticated I/O control.
- **CORDIC-based designs [48–50]:** A CORDIC processor is used for computing the cosine coefficients in the DCT. Similar to SA implementations, CORDIC implementations require significant resources.
- **Algebraic Integer(AI)-based designs [51]**: By mapping possibly irrational numbers to an array of integers, these methods can achieve high precision. However, good precision requires significant resources.

Compared to separable approaches, non-separable approaches require more resources since the number of required FIR taps grows as N^2 as opposed to N for the separable case. Furthermore, for dynamic reconfiguration, it is clear that non-separable approaches require considerable overhead since we will have store the architecture descriptions in memory. Thus, in what follows, the paper will focus on a separable approach based on Chen's algorithm [40].

3.3.4 A separable implementation of the 2D DCT based on Chen's algorithm

From (3.1), a separable implementation of the 2D DCT is given by $Z = (MX)M^T$, where $M_{i,j} = C_i \cos(\frac{i(2j+1)\pi}{16})$. Here, $(MX)M^T$ is implemented by first transposing (MX) and then applying M. Thus, a separable implementation is based on: $Z = (M(MX)^T)^T$.

To efficiently implement multiplication by M, let $D_i = \cos(i\pi/16)/2$, and define $a = D_4$, $b = D_1$, $c = D_2$, $d = D_3$, $e = D_5$, $f = D_6$, $g = D_7$, which gives the following expression for M:

$$M = \begin{bmatrix} a & a & a & a & a & a & a & a \\ b & d & e & g & -g & -e & -d & -b \\ c & f & -f & -c & -c & -f & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & a \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & c & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix}. \tag{3.2}$$

The output $Y = MX$ is computed using

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} X(0)+X(7) \\ X(1)+X(6) \\ X(2)+X(5) \\ X(3)+X(4) \end{bmatrix} \tag{3.3}$$

where the matrix multiplication is efficiently implemented using:

$$\begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} = \begin{bmatrix} a & a & 0 & 0 \\ 0 & 0 & c & f \\ a & -a & 0 & 0 \\ 0 & 0 & f & -c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}. \tag{3.4}$$

Similarly, an efficient matrix decomposition can be used to implement the odd-indexed output expressed as:

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(0)-X(7) \\ X(1)-X(6) \\ X(2)-X(5) \\ X(3)-X(4) \end{bmatrix}. \tag{3.5}$$

To produce a frequency-scalable representation, begin with the lower-indexed DCT coefficients given by $Y(0), Y(1), \ldots, Y(n)$, where $n \leq 7$. Then, in the implementation of the DCT, the corresponding rows in (3.3) and (3.5) need to be implemented so as to compute the required DCT coefficients. In the separable approach described here, the 2D DCT coefficients are given by $X_{u,v} \quad 0 \leq u, v \leq n$ (see (3.1)).

3.3.5 Video image quality assessment using SSIM

Video image quality will be assessed using the Structural Similarity Index (SSIM) [18]. Here, note that video quality assessment is still an open problem (e.g., see [52–55]). However, SSIM provides a simple and effective method for assessing video image quality of individual frames.

Assuming that x, y represent the original and reconstructed images, SSIM is given by:

$$SSIM(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \tag{3.6}$$

which is expressed as the product of the luminance ($l(x,y)$), the contrast ($c(x,y)$), and structure components ($s(x,y)$), and $\alpha, \beta, \gamma > 0$ are set to the default value of 1.

3.3.6 Quantization table specification using a quality factor

The DCT quantization level will be controlled using the quality factor (QF). QF is given as integer value that is constrained between 1 and 100. The DCT quantization table is then given by:

$$Q_{ij} = \text{Clip}_{1,255}\left[\frac{Q^*_{ij} \cdot \texttt{scale} + 50}{100}\right]$$

clipped to stay within 1 and 255, Q^*_{ij} refers to the standard JPEG quantization table for $\texttt{scale} = 1$ and the `scale` is given by:

$$\texttt{scale} = \begin{cases} 5000/QF, & \text{for } 1 \leq QF < 50; \\ 200 - 2 \cdot QF, & \text{for } 50 \leq QF \leq 99; \\ 1, & \text{for } QF = 100. \end{cases}$$

3.4 A Dynamically Reconfigurable Architecture System for Time-Varying Image Constraints (DRASTIC)

3.4.1 Constrained optimization formulation

This section introduces a constrained optimization framework for defining DRASTIC modes. The implementation of the framework uses joint software-hardware implementation.

The optimization objectives are defined in terms of: (i) DP which denotes the dynamic power consumed by the FPGA (see [28]), (ii) BPS which denotes the number of bits per sample, and (iii) Q which denotes the image quality metric. In terms of the free parameters that we can modify to achieve our objectives, let HW denote the different hardware configurations, and let QF denote the quality factor used for controlling the quantization table (software controlled). To formulate constraints on the objectives, let Q_{min} denote the minimum acceptable image quality level, P_{max} denote the maximum dynamic power available, and Q_{min} denote the minimum level of acceptable image quality.

The DRASTIC modes are then defined as constrained optimization problems using:

- **minimum power mode** (`mode=0`):

$$\min_{HW,QF} DP \text{ subj. to: } (\text{SSIM} \geq Q_{min}) \,\&\, (\text{BPS} \leq B_{max}).$$

- **minimum bitrate mode** (`mode=1`):

$$\min_{HW,QF} \text{BPS subj. to: } (\text{SSIM} \geq Q_{min}) \,\&\, (\text{DP} \leq P_{max}).$$

- **maximum image quality mode** (`mode=2`):

$$\min_{HW,QF} -\text{SSIM subj. to: } (\text{BPS} \leq B_{max}) \& (\text{DP} \leq P_{max}).$$

- **typical mode** (`mode=3`):

$$\min_{HW,QF} -\alpha \cdot \text{SSIM} + \beta \cdot \text{BPS} + \gamma \cdot \text{DP}$$

subj. to:

$$(\text{BPS} \leq B_{max}) \,\&\, (\text{SSIM} \geq Q_{min}) \,\&\, (\text{DP} \leq P_{max}).$$

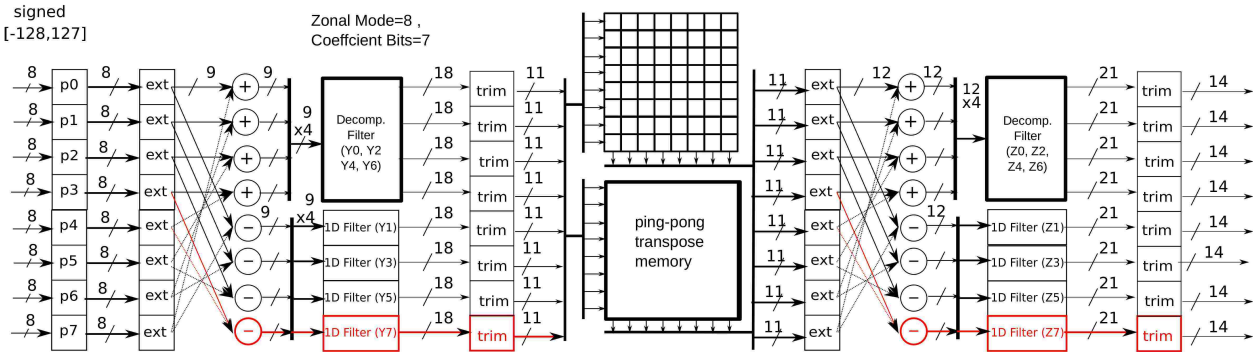


Figure 3.2: Scalable data path for the 2D-DCT using ping-pong transpose memory. The number of bits used at each stage are given in the figure. The input image block is assumed to be of size 8 with signed, 8-bit integer values. The removal of the highest frequency components is highlighted in red.

3.4.2 Hardware design

A scalable and separable implementation of the DCT is shown in Fig. 3.2. A ping-pong memory [38] implementation is used for efficient implementation of the transpose operation. The Decompose filter shown in Fig. 3.2 is used to implement (3.4), (3.3) as shown in Fig. 3.3. The 1D filter used in implementing (3.5) is shown in Fig. 3.4.

The implementation in Fig. 3.2 represents a parallelized and pipilined implementation. In terms of parallelism, we note that the column DCTs can be implemented in parallel, followed by transposition in ping-pong memory, and then the row DCTs. Row operations in (3.5) are carried out in parallel using 1D filters. The trim operations implement floor operations by truncating the results towards zero as shown in Fig. 3.5. For each 1D DCT, we have a 4-staged pipeline. The Ping-Pong transpose memory consists of two 8×8 transpose memory arrays. In pipelined operation, a row DCT is computed in each cycle. Furthermore, it takes 8 cycles to complete a 2D DCT.

The scalability required for effective multi-objective optimization is implemented using zonal control [14, 29–31, 56], and output bit width control. As stated earlier, the basic idea is to achieve perceptual scalability by keeping the lower frequency components while elimi-

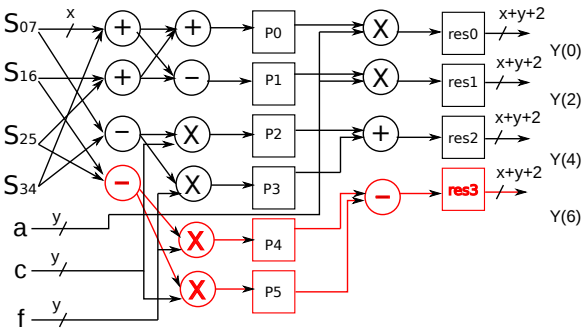


Figure 3.3: Scalable Decompose filter implementation of the matrix-vector product given in (3.4). Refer to Fig. 3.2 for how the decompose filter fits the DCT core. The inputs S_{ij} refer to the $X(i)+X(j)$ sum of equation (3.3). The outputs correspond to $Y(0)$, $Y(2)$, $Y(4)$, $Y(6)$ of (3.3). The datapath associated with the highest frequency component is highlighted in red. Note how tracing backwards from each output, we can generate a scalable datapath that removes the circuitry associated with each frequency component.

nating the computation of higher frequency components. We implement 8-levels associated with keeping the DCT lower-frequency subsets of the complete frequency set. We use `Z0` to `Z7` to denote the different zones (levels) associate with the DCT computation. Similarly, for bitwidth control, we keep the most significant bits $[15, 57]$. This is implemented adjusting the word-length implementation of the DCT coefficients a to g in (3.2) using $WL \in [2, 9]$.

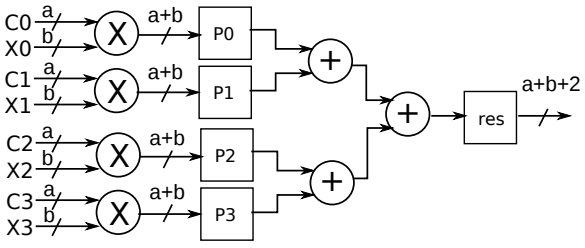


Figure 3.4: Implementation of the 1D filters shown in Fig. 3.2. Here, `C0-C3` refer to the DCT Kernel coefficients and `X0-X3` refer to sums and differences computed on the input data (see Fig. 3.2).

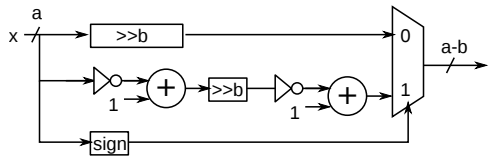


Figure 3.5: Signed integer trimming of `a`-bit input `x` to an `(a-b)`-bit output by truncating the output towards zero (`floor` operation). This component is used to control the bit-width in the optimization process.

3.4.3 Pareto Front and Constraint Satisfaction

The range of possible DCT hardware cores needs to be jointly considered with control of the quantization table. For example, gains due to increasing the bit-width in the DCT cores may be offset by a decrease in the Quality Factor (see section 3.3.6). The goal here is to eliminate configurations that are not Pareto-optimal [28]. In other words, we eliminate configurations for which we can find another configuration that delivers performance that is at-least as good in all of the objectives (quality, power, bitrate), and performs better in at-least one of the objectives [28]. The remaining configurations represent the Pareto-front that will be used in further optimization.

In practice, the Pareto-front can be computed offline using a training set. For each configuration, the average performance for each set of objectives will be used for determining the Pareto-front.

We consider a direct implementation of DRASTIC modes by searching through the Pareto-front. In this direct approach, we select the configuration that minimizes the optimization objective while satisfying the constraints. For example, in the maximum quality mode, we would select the configuration that provides for the best (max) image quality while not exceeding constrains on power and bitrate. As we shall describe next, we can reduce the overhead by controlling the number and frequency of dynamic reconfigurations.

3.4.4 Scalable Control of Reconfiguration Overhead

Unlike H.264 and H.265 video encoding, for our target MJPEG application, feedback can be considered to be an expensive operation. Thus, we avoid the use of feedback and dynamic partial configuration (DPR) while processing a single video frame. Instead, we propose the use of periodic updates and a bound on the maximum number of reconfigurations. We provide details below.

In what follows, let `RecP` denote the reconfiguration period that describes how often the system reconfigures in terms of the number of frames. Also, let `RecC` denote the maximum number of allowed reconfigurations for a specified duration of `RecDur` (from *rec*configuration *dur*ation) video frames. Here, `RecC` is used to control the total overhead associated with dynamic reconfiguration for adapting to the DRASTIC mode. Alternatively, `RecC` could have been set as a function of the reconfiguration period `RecP`. In this paper, we just use `RecDur` frames for each mode, and assign a default value of 100 video frames. Thus, unless otherwise specified, we assume `RecDur` $= 100$ so that `RecC` can be interpreted as the percentage of the number of fames for which the system is allowed to reconfigure. Since dynamic reconfiguration is not allowed within a single video frame, all of the objectives are also updated as a function of the number of processed video frames. The goal here is to adjust the constraints set for each video frame so that the constraints are met on-average over the processed frames.

To illustrate the basic idea for the bitrate constraint, let n denote the current video frame, and recall that B_{max} denotes the maximum number of bits per pixel that appear in the DRASTIC mode. After processing the video frame, let $\texttt{BPS}_\texttt{n}$ denote the measured number of bits per sample that were used in encoding the n-th frame. We then have that the remaining bits that can be allocated (or deallocated) to future frames are given by

$$\Delta_{\texttt{BPS,n}} = B_{max} - \texttt{BPS}_\texttt{n}. \tag{3.7}$$

Assuming that we periodically reconfigure after `RecP` frames, we would then adjust the

maximum bitrate allocated for the n-th frame $\mathtt{B}_{\mathtt{max,n}}$ using:

$$\mathtt{B}_{\mathtt{max,n}} = \begin{cases} B_{max} + \Delta_{\mathtt{BPS,n}}/\mathtt{RecP}, & \text{for } (n-1)\%\mathtt{RecP} = 0 \\ \mathtt{B}_{\mathtt{max,n-1}}, & \text{otherwise,} \end{cases} \tag{3.8}$$

which allocates (or deallocates) the remaining bits over RecP frames. Thus, by adjusting the number of bits per sample for each video frame, we expect that the constraints will be met on average. Similarly, the approach can be applied for updating constraints on image quality $\mathtt{Q}_{\mathtt{min,n}}$ and dynamic power $\mathtt{P}_{\mathtt{max,n}}$. Furthermore, the rule in (3.8) can be easily extended so that it will only apply when the number of dynamic reconfiguration does not exceed a maximum bound. Clearly, it is possible that the constraints cannot be met on the Pareto-front. In this case, we reformulate the problem using unconstrained optimization so that the controller will select a configuration that will be as close as possible to the desired constraints. For example, for the *typical mode*, when all constraints are active, we select the configuration that solves:

$$\begin{aligned} \min_{HW,QF} \quad & a(\mathtt{B}_{\mathtt{max,n}} - \mathtt{BPS}_{\mathtt{n}})^2 \\ & +b(\mathtt{P}_{\mathtt{max,n}} - \mathtt{DP}_{\mathtt{n}})^2 \\ & +c(\mathtt{Q}_{\mathtt{min,n}} - \mathtt{SSIM}_{\mathtt{n}})^2, \end{aligned} \tag{3.9}$$

where the weights a, b, c can be set equal or adjusted to give emphasis to different constraints. When a constraint is not active, its corresponding weight is set to zero. For example, for the maximum quality mode, we will set $c = 0$. We select the weights so as to scale each constraint violation by the user-specified range of bounds. For example, if Q_1, Q_2 represent the minimum and maximum bounds on image quality, we set c to $c = 1/(Q_2 - Q_1)^2$. We use a similar approach for a and b.

3.4.5 Scalable DRASTIC Controller

The general framework for implementing a DRASTIC mode is given in Fig. 3.6. Switching between DRASTIC modes requires that the code gets adapted to the new mode. On the

other hand, we note that the Pareto front applies to all of the modes and only needs to be computed once over the training data. The controller controls the overhead based on maximum number of reconfigurations `RecC` per `RecDur` frames and the reconfiguration period `RecP`.

Initially, a single video frame is processed in order to estimate the objectives based on the initial configuration. The relevant constraint budgets are then updated and used to search the Pareto front for the optimal configuration (see (3.8)). Failure to find a configuration that satisfies the constraints will force a reformulation of the problem as an unconstrained optimization problem. Once the optimal configuration has been found, it is used for processing the remaining $\texttt{RecP} - \texttt{1}$ frames of the current period. The procedure is then repeated for the next set of `RecP` video frames.

3.5 Results

3.5.1 Pareto-front estimation & comparisons to full 2D DCT implementations

To generate an estimate of the Pareto front, we use the LIVE image database as a training set [23]. For each configuration, we generate the hardware core and estimate the required bitrate, image quality, and dynamic power that is required for compressing each image. For the dynamic power, we use Xilinx's XPower tool to estimate power consumption on a Virtex-5 device (Xilinx XC5VLX110T). Then, the Pareto-front is estimated based on the median value of each configuration.

We generate hardware configurations by varying: (i) the software-based quality factor $QF = 5, 10, 15, \ldots, 100$ (20 settings), (ii) the DCT hardware to compute $Z_{u,v}$ for $0 \leq u, v < \mathcal{Z} = 1$ to $0 \leq u, v < \mathcal{Z} = 8$ (8 settings), and (iii) the DCT coefficients implemented in

hardware using word-length $WL = 2, 3, 4, \ldots, 9$ (8 settings). Based on the different settings, we have $20 \times 8 \times 8 = 1280$ possible configurations from which only 841 were found to be Pareto-optimal. The Pareto front is shown in Fig. 3.8. The Pareto front surface sets fundamental limits on the implementation of DRASTIC mode constraints as will be discussed in Section 3.5.2.

The resulting hardware configurations are summarized in Table 3.1 and the corresponding estimated dynamic power is shown in Fig. 3.7. In order to visualize the scalability of the proposed approach, we index the hardware configurations using $\texttt{Config} = (\mathcal{Z}-1)\cdot 8+WL-1$. Then, we plot the the required slices and dynamic power against `Config` in Fig. 3.7. From Fig. 3.7(b), it is clear that the dynamic power is densely sampled in the configuration space. Returning to the Pareto front results of Fig. 3.8, it is important to note the relatively dense sampling achieved over the Pareto front for image quality levels associated with $\texttt{SSIM} > 0.7$. This observation is important since reducing image quality below this level will produce images of unacceptable quality (e.g., see Fig. 3.12(d)).

A comparison of the full DCT implementation against other FPGA implementations is given in Table 3.4. As a result of the parallel and pipelined implementation, the proposed DCT architecture achieves the highest throughput by only requiring 8 cycles to compute a 2D DCT. Yet, the implementation requires lower numbers of FPGA slices and consumes low levels of dynamic power. In terms of dynamic power, we note the lower results due to Huang et al. [29] were achieved using a much simpler architecture at a much lower frequency (41.79 MHz versus 100 MHz of the proposed approach), that requires a significantly higher number of throughput cycles. In any case, the greatest advantage of the proposed approach is the fact that it is scalable in dynamic-power, image quality, and bit-rate while providing full DCT calculation that is at-least as good or better than any previously published approach.

3.5.2 DRASTIC mode implementation & comparison to optimized static approaches

This section provides specific definitions of the DRASTIC modes and explains how they can be derived from the Pareto front. An effective setup for controlling the dynamic reconfiguration overhead that requires reconfiguration for only 5% of the video frames is discussed afterwards. The section summarizes how the proposed approach can lead to significant savings over optimized static approaches. While training was performed on the UT LIVE image database, the system was validated on an independent testing video database of 9 standard videos: city, crew, football, foreman,hall monitor, harbor, mobile, mother and daughter and soccer (see [58]).

In order to define realistic DRASTIC constraint profiles, we need to select profiles that are compatible with the Pareto front (see Fig. 3.8). Given the wide applications of the UT LIVE image quality databases, we expect that the values derived from them will be widely applicable. In general, we can see that we can achieve higher values of image quality by allocating higher bitrate and larger values of dynamic power. To understand this trend, note that higher image quality results from the need to compute higher frequency components and longer word-lengths that result in higher dynamic power. Furthermore, storing the higher frequency components requires additional bits that raise the number of bits per sample. Realistically, image quality bounds need to have SSIM values about 0.7 to maintain a minimum level of acceptable image quality. This discussion leads to the low, medium, and high profiles given in Table 3.2.

The efficient implementation of the DRASTIC modes requires that we determine optimal parameters for `RecC` and `RecP` so as to minimize the reconfiguration overhead while still providing acceptable performance. We investigate the trade-off between `RecC` and `RecP` by considering all DRASTIC modes for (i) periodic update control using `RecP`=1, 5, 10, while allowing the maximum number of reconfigurations per 100 frames using `RecC`=100, and (ii)

initial adaptation control using `RecC`=5, 10, 100, while allowing the maximum number of periodic updates by using `RecP`=1. The results are summarized in Figs. 3.11 and 3.12.

For the typical mode plots of Fig. 3.12, it is clear that the constraints are met for all profiles. In other words, for most configurations, image quality remains above the minimum levels, while dynamic power and required bit rates remain below the bounds. For all of the other modes, only two of the three constraints are active, while the remaining constraint becomes an objective to be optimized. For example, for the maximum image quality mode demonstrated in Figs. 3.11(e), (f), it is clear that we have substantially higher image quality that the typical mode, which can push dynamic power consumption slightly above the constraints. On average though, it is clear that most constraints are met for the non-typical modes. Since the validation is independent of the training set, we can infer that the Pareto front from the UT LIVE image database captured the constraints in more general settings.

The use of larger reconfiguration periods (larger `RecP`) tends to spread out the distributions of the objectives. With larger spreads, we also get an increase in constraint violations. On the other hand, when reconfiguring after each frame (`RecP`=1), the number of reconfigurations (`RecC`) does not seem to provide significant improvements for larger values (1 to 5 to 100). Thus, by allowing an early adaptation to the input video using `RecP` = 1, and then limiting the number of dynamic reconfigurations (`RecC` = 5), we have an effective control of the overhead while still producing distributions that are centered on the desired constraints. At this setting, we only reconfigure at 5% of the input frames, providing a significant reduction in the overhead.

To demonstrate the overall advantages of the proposed DRASTIC modes, we provide a comparison against the use of static configurations in Table 3.3. For the static configuration, we select the one with the maximum performance metric. For example, for the maximum image quality mode, we select the configuration that gave the maximum image quality among all video frames. Compared to the optimized static configuration, at only 5% reconfiguration rate, we still get significant savings in dynamic power (25% to 37%), bit-rate (47% to 55%),

while reducing image quality from the maximum mode by very low percentages (3% to 6%). Full reconfiguration can provide additional savings for the non-typical modes.

3.5.3 DRASTIC Mode Transition Example

In this section, we provide an example of transitioning from one mode to another. The example demonstrates the ability of the proposed approach to adapt the input video content.

We consider the following simple mode transition example over a video consisting of 100 frames:

- **Max im. qual. mode with high-profile ($n = 1, \ldots, 25$):** Initially, the users will want to review video contents to see if there is something interesting. So, we use a maximum image quality for this initial mode.
- **Typical mode with medium-profile ($n = 26, \ldots, 50$):** After adapting to the video, the users will want to operate in a typical mode.
- **Min rate with medium-profile ($n = 51, \ldots, 75$):** A transition to low-coverage areas can stimulate a switch to a low bit-rate mode.
- **Min power with low-profile ($n = 76, \ldots, 100$):** Reduced battery life can induce a transition to the minimum power mode.

From the DRASTIC mode transitions of Fig. 3.9, it is clear that that the dynamic reconfiguration works well. The basic idea of adjusting to the input video at the beginning of the mode does allow the system to meet the constraints. Also, from the video images of Fig. 3.12, we can see exceptional image quality for the maximum image quality mode (3.12(a)), acceptable quality for the typical mode (3.12(b)), reduced quality for the minimum rate mode (3.12(c)), that reduces to barely acceptable quality for the minimum power mode (3.12(d)). In terms of dynamic power consumption, it is interesting to note that the typical

mode with a medium-profile requires only slightly mode power than the minimum power mode with a low-profile.

In real-life videos, we would expect the transitions to occur after several seconds. This implies that the dynamic reconfiguration overhead can be further reduced from the 5% rate demonstrated here to much lower rates. For example, this could be accomplished by using 5 reconfigurations to adjust to the input video and then maintaining the same profile over several hundreds of frames.

3.6 Conclusion and future work

The paper has introduced DRASTIC modes to allow for fine optimization control for maximizing image quality, minimizing bitrate requirements, reducing dynamic power consumption, or providing a typical mode that balances constraint requirements. An efficient and scalable architecture based on the 2D DCT was used for implementing the DRASTIC modes. From the results, it is clear that the use of DRASTIC can lead to significant power and bitrate savings over optimized static approaches. Furthermore, the dynamic reconfiguration overhead can be minimized by reducing the reconfiguration rate (5% or less).

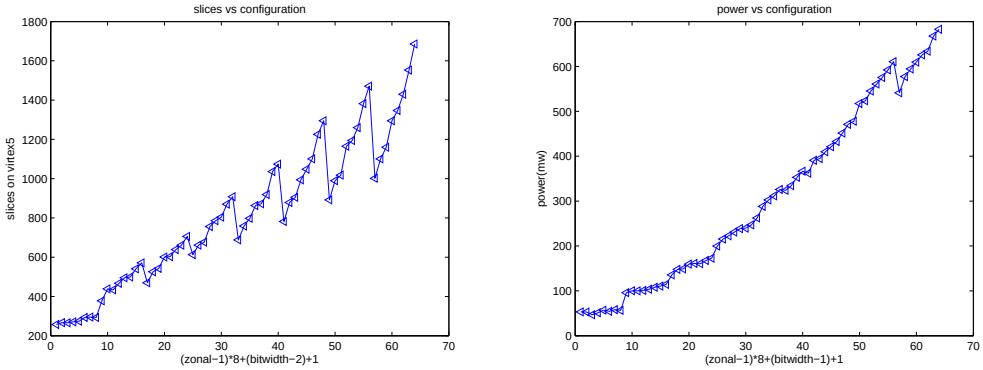
Future work will be focused on extending the DRASTIC approach to other video processing and communications applications. Furthermore, future work will look at methods to generate constraints dynamically based on video content.

Figure 3.6: General framework for DRASTIC mode implementation.

Input: input video, DRASTIC mode with associated constraints, offline trained Pareto front, reconfiguration period `RecP`, maximum allowed reconfigurations `RecC` per `RecDur` frames.

Output: generated compressed video stream that implements the given DRASTIC mode.

Initialize counter for dyn. reconf.: $\texttt{RecCtr} = 0$.
Initialize counter for mode change: $\texttt{DurCtr} = 0$.
Initialize constraint budgets: $\Delta_{\texttt{BPS},0} = \Delta_{\texttt{DP},0} = \Delta_{\texttt{Q},0} = 0$.
Initialize frame index: $n = 0$.
while video communication holds **do**
  if ($\texttt{RecCtr} < \texttt{RecC}$) **then**
    Search Pareto front for opt. HW, QF
    if no configuration satisfies constraints **then**
      Solve unconstr. opt. prob. for opt. HW, QF as given by eq. (3.9).
    end if
    Apply DPR with HW, QF to compress current frame.
    Update constr. budgets as given by eqns (3.7) and (3.8).
  end if
  Compress the $(n+1)$-th to $(n + \texttt{RecP} - 1)$-th frames using the current configuration.
  Update count $\texttt{RecCtr} = \texttt{RecCtr} + 1$.
  Update $n = n + \texttt{RecP}$.
  Update $\texttt{DurCtr} = \texttt{DurCtr} + \texttt{RecP}$.
  if $\texttt{DurCtr} \geq \texttt{RecDur}$ **then**
    Reset the mode frame counter : $\texttt{DurCtr} = 0$.
    Reset the reconf. counter : $\texttt{RecCtr} = 0$.
  end if
end while



(a) Slice res. (b) Dynamic power

Figure 3.7: Resource allocation and estimated dynamic power consumption for 64 hardware configurations based on bitwidth values: $2 \leq WL \leq 9$ and zonal values: $1 \leq \mathcal{Z} \leq 8$. (a) Slice resources as a function of the zonal configuration and bit width. (b) Dynamic power consumption as a function of zonal configuration and bit width. From the dynamic power results, it is clear that a scalable set of DCT architectures has been achieved.

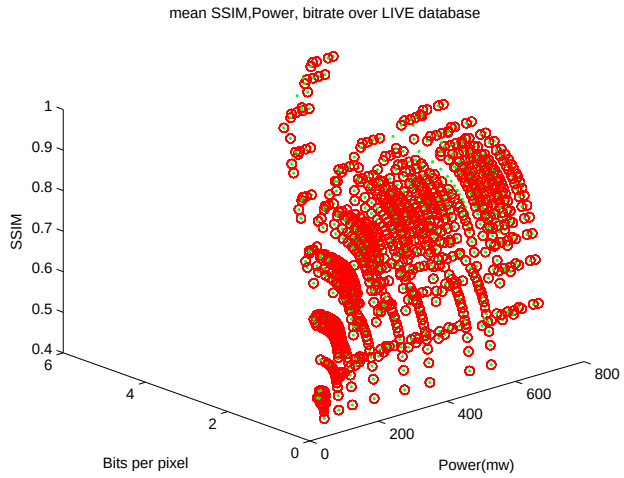


Figure 3.8: Pareto front estimation for the joint space of SSIM, bit rate, and dynamic power consumption. The Pareto front is estimated using the UT LIVE image database. Pareto optimal configurations are highlighted using red circles.

Table 3.1: Synthesized results for DCT Cores on XC5VLX110T−1FF1136.

WL	$\mathcal{Z}$	LUT	Registers	Slices	max freq (Mhz)	Power (mW)
2	1	622(1%)	534(1%)	257(1%)	250.376	53.03
3	1	684(1%)	540(1%)	267(1%)	249.066	52.98
4	1	659(1%)	540(1%)	266(1%)	249.066	46.67
5	1	713(1%)	545(1%)	271(1%)	245.881	51.21
6	1	756(1%)	547(1%)	274(1%)	204.834	56.87
7	1	768(1%)	548(1%)	293(1%)	204.834	54.37
8	1	799(1%)	552(1%)	295(1%)	203.832	58.22
9	1	806(1%)	553(1%)	293(1%)	203.832	56.58
2	2	847(1%)	886(1%)	378(2%)	250.689	95.85
3	2	1074(1%)	954(1%)	439(2%)	249.066	100.28
4	2	1099(1%)	958(1%)	434(2%)	247.463	100.06
5	2	1314(1%)	988(1%)	467(2%)	209.293	101.09
6	2	1361(1%)	999(1%)	495(2%)	204.834	103.57
7	2	1403(2%)	1006(2%)	499(2%)	204.750	108.03
8	2	1556(2%)	1024(2%)	541(3%)	203.832	110.65
9	2	1662(2%)	1038(2%)	571(3%)	203.293	114.56
2	3	951(1%)	1137(1%)	470(2%)	250.689	135.65
3	3	1238(1%)	1211(1%)	526(3%)	249.066	147.82
4	3	1238(1%)	1215(1%)	543(3%)	247.463	148.92
5	3	1507(2%)	1250(2%)	601(3%)	209.293	159.57
6	3	1597(2%)	1263(2%)	603(3%)	204.834	161.17
7	3	1651(2%)	1271(2%)	639(3%)	204.750	160.50
8	3	1835(2%)	1293(2%)	661(3%)	203.832	167.79
9	3	1948(2%)	1308(2%)	706(4%)	203.293	172.91
2	4	1242(1%)	1454(1%)	613(3%)	250.689	199.83
3	4	1540(2%)	1548(2%)	661(3%)	249.066	215.33
4	4	1573(2%)	1555(2%)	677(3%)	247.463	222.61
5	4	1936(2%)	1599(2%)	756(4%)	207.684	230.89
6	4	2070(2%)	1619(2%)	786(4%)	203.998	238.55
7	4	2139(3%)	1632(3%)	804(4%)	203.832	239.61
8	4	2416(3%)	1672(3%)	870(5%)	203.832	247.22
9	4	2624(3%)	1699(3%)	908(5%)	202.143	262.42
2	5	1367(1%)	1701(1%)	688(3%)	250.689	288.24
3	5	1727(2%)	1832(2%)	759(4%)	205.170	302.56
4	5	1844(2%)	1849(2%)	798(4%)	202.593	311.60
5	5	2214(3%)	1899(3%)	863(4%)	160.668	325.87
6	5	2341(3%)	1917(3%)	872(5%)	160.668	324.05
7	5	2464(3%)	1941(3%)	919(5%)	158.078	334.35
8	5	2771(4%)	1989(4%)	1036(5%)	157.953	352.93
9	5	3040(4%)	2019(4%)	1074(6%)	156.519	366.55
2	6	1560(2%)	1983(2%)	782(4%)	250.689	362.18
3	6	1977(2%)	2146(2%)	879(5%)	205.170	390.94
4	6	2125(3%)	2166(3%)	905(5%)	202.593	394.71
5	6	2613(3%)	2248(3%)	994(5%)	160.668	409.80
6	6	2742(3%)	2277(3%)	1048(6%)	160.668	421.03
7	6	2880(4%)	2306(4%)	1101(6%)	158.078	432.90
8	6	3315(4%)	2363(4%)	1226(7%)	157.953	451.66
9	6	3659(5%)	2412(5%)	1295(7%)	156.519	470.98
2	7	1685(2%)	2230(2%)	892(5%)	250.689	478.08
3	7	2164(3%)	2430(3%)	990(5%)	205.170	517.46
4	7	2396(3%)	2460(3%)	1018(5%)	202.593	523.60
5	7	2891(4%)	2548(4%)	1165(6%)	160.668	545.27
6	7	3013(4%)	2575(4%)	1194(6%)	160.668	560.92
7	7	3205(4%)	2615(4%)	1260(7%)	158.078	575.57
8	7	3670(5%)	2680(5%)	1382(7%)	157.953	592.27
9	7	4075(5%)	2732(5%)	1471(8%)	156.519	610.75
2	8	1895(2%)	2510(2%)	1002(5%)	250.689	541.17
3	8	2431(3%)	2742(3%)	1101(6%)	205.170	577.46
4	8	2694(3%)	2775(3%)	1160(6%)	202.593	594.46
5	8	3338(4%)	2898(4%)	1295(7%)	160.668	609.64
6	8	3481(5%)	2934(5%)	1347(7%)	160.668	625.68
7	8	3686(5%)	2979(5%)	1430(8%)	158.078	634.09
8	8	4212(6%)	3046(6%)	1553(8%)	157.953	667.75
9	8	4716(6%)	3122(6%)	1686(9%)	156.519	682.88

Table 3.2: DRASTIC constraint profiles. The constraints represent the bounds for (i) image quality (Q_{min}), (ii) the bitrate (B_{max}), (iii) and dynamic power (P_{max}) as described in section 4.4.

DRASTIC Constraint	**Constraints Profile**		
	Low	**Medium**	**High**
Image Quality (SSIM)	0.7	0.8	0.9
Bitrate (bits per sample)	0.5	1.0	1.5
Power (mW)	200	300	400

Table 3.3: DRASTIC mode savings over the use of the optimized maximum setting for each mode for the 9 testing videos. Here, the savings are computed as a percentage of the average performance metric. For example, for dynamic power, the percentage savings computed using $(P_{max} - P_{avg})/P_{avg} * 100$ where P_{avg}, P_{max} are computed from the selected DRASTIC architectures. For dynamic power and bitrate constraints, higher percentages indicate higher savings. For image quality, lower percentages are preferred since they indicate that the resulting videos will be of higher quality. The proposed reconfiguration (Prop. Rec.) refers to `RecC=5, RecP=1` while full reconfiguration refers to `RecC=100, RecP=1`. The proposed reconfiguration requires 5% of the overhead of the full reconfiguration. Also, note that the savings are conservative since they assume an optimal pre-selection of the static architecture.

DRASTIC Mode	**Constraints Profile**		
	Low	**Medium**	**High**
Min Dyn. Pwr (Prop. Rec.)	37.3%	36.9%	25.3%
Min Dyn. Pwr (Full Rec.)	57.7%	38.3%	31.7%
Min Bitrate (Prop. Rec.)	51.2%	46.7%	55.1%
Min Bitrate (Full Rec.)	57.9%	63.4%	58.1%
Max Im. Qual. (Prop. Rec.)	**6.1%**	5.4%	3.0%
Max Im. Qual. (Full Rec.)	6.7%	4.5%	2.8%
Typical Mode (Dyn. Pwr, Prop. Rec.)	**43.5%**	**37.9%**	24.9%
Typical Mode (Dyn. Pwr, Full Rec.)	35.6%	35.4%	27.3%
Typical Mode (Bitr., Prop. Rec.)	**34.7%**	**33.9%**	62.2%
Typical Mode (Bitr., Full Rec.)	32.0%	32.7%	65.0%
Typical Mode (SSIM, Prop. Rec.)	**8.3%**	**6.1%**	**2.4%**
Typical Mode (SSIM, Full Rec.)	10.2%	6.5%	2.9%

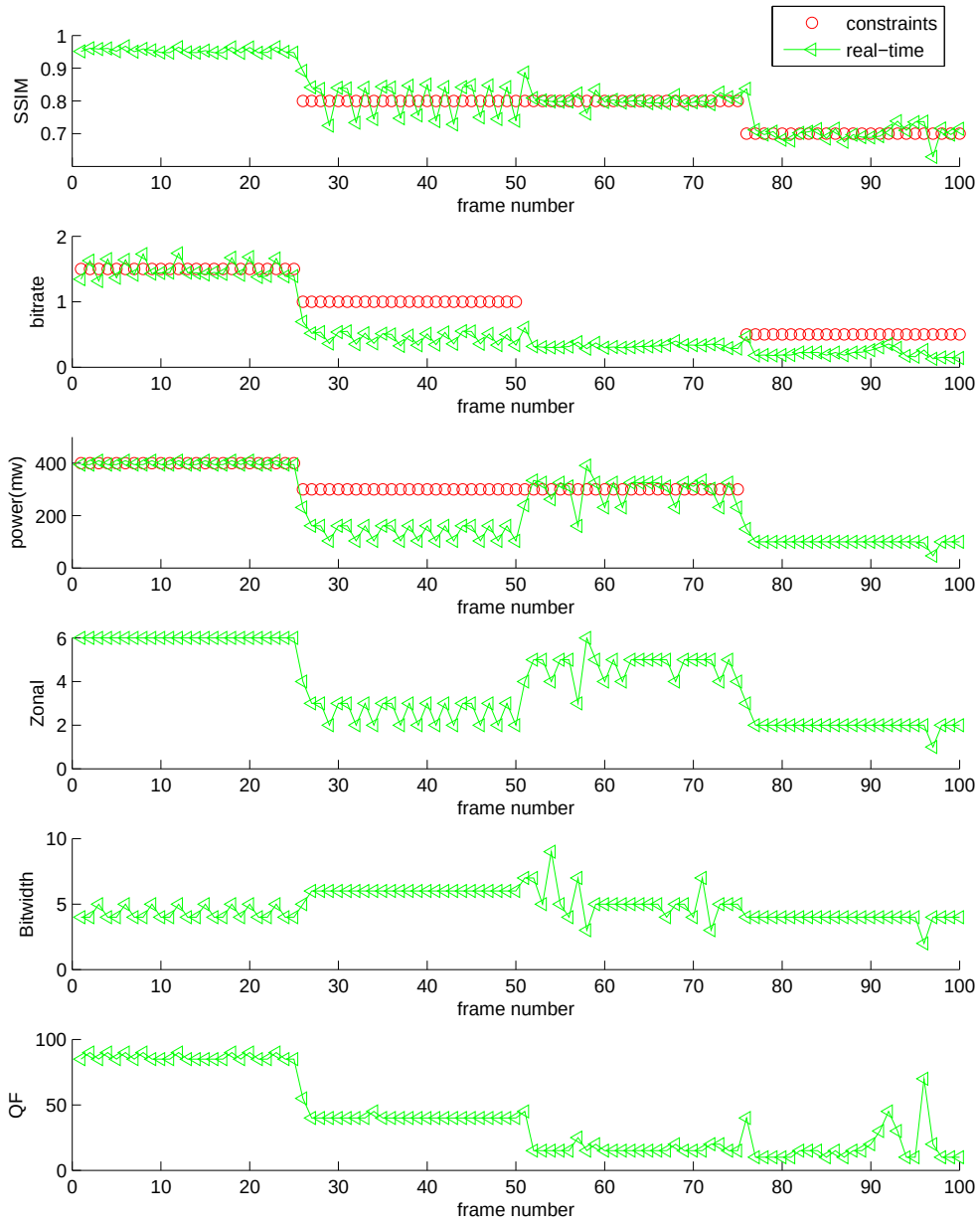
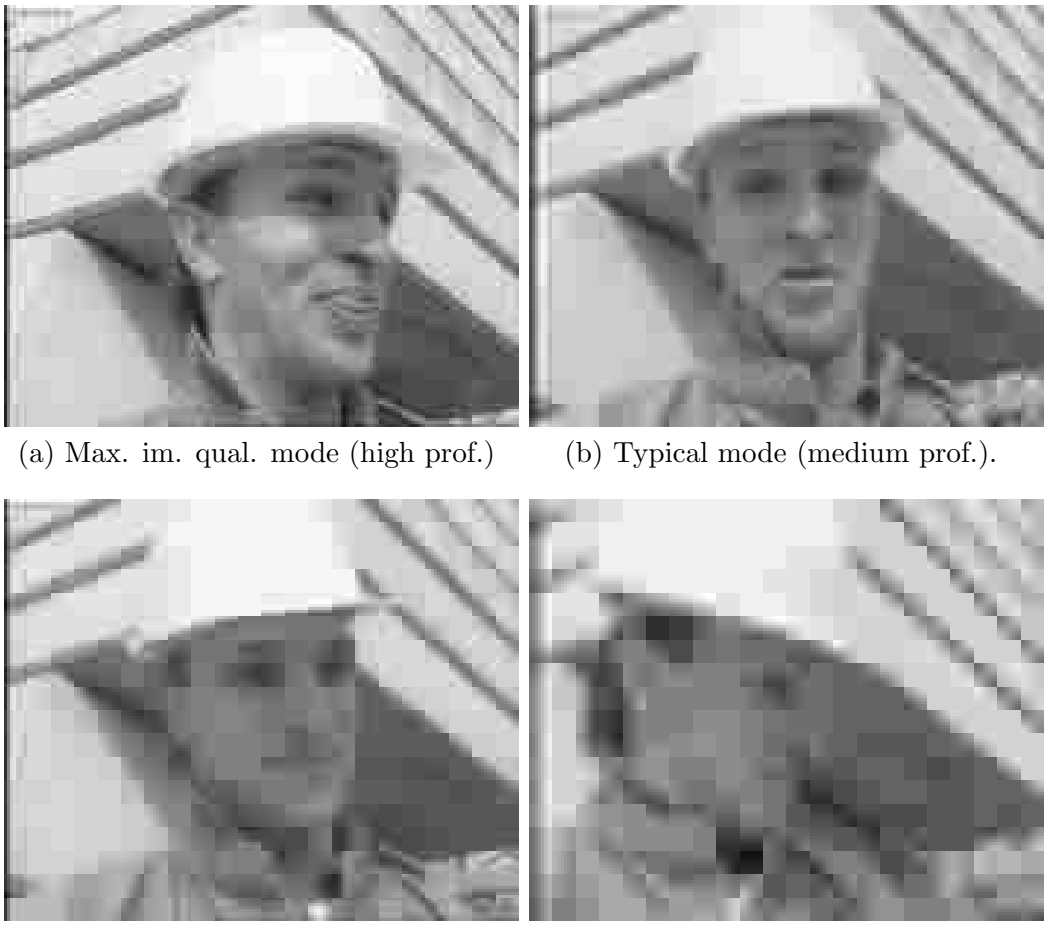


Figure 3.9: DRASTIC reconfiguration results for switching between modes for the Foreman video. Here, the proposed reconfiguration settings are used (`RecC=5, RecP=1`).



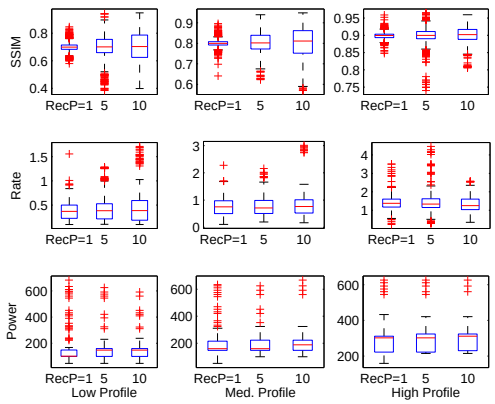
(a) Max. im. qual. mode (high prof.)

(b) Typical mode (medium prof.).

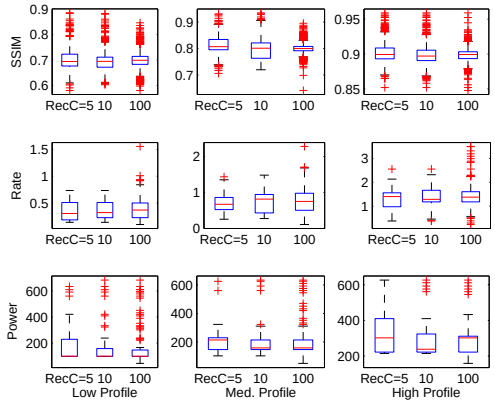
(c) Min. rate mode (medium prof.)

(d) Min. power mode (low prof.)

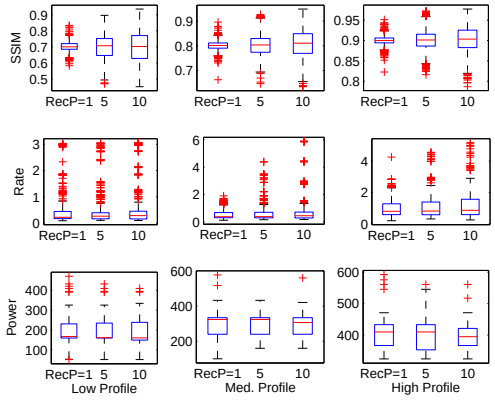
Figure 3.10: DRASTIC mode transition example results. (a) Max img qual. mode ($n = 5$): SSIM=0.95, Rate=1.36bps, DP=395mW which gives exceptional image quality while meeting the high-profile constraints. (b) Typical mode ($n = 35$): SSIM=0.84, Rate=0.51bps, DP=161mW which meets all of the medium-profile constraints at a much lower bitrate. (c) Min rate mode ($n = 60$): SSIM=0.79, Rate=0.31bps, DP=312mW which is right at the boundary of the image quality and dynamic power constraints (medium-profile) while using significantly less bitrate. (d) Min power mode ($n = 85$): SSIM=0.69, Rate=0.18bps, DP=100mW which is at the boundary of the image quality constraint for the low-profile, unable to further reduce power, but still operating at a very low bitrate.



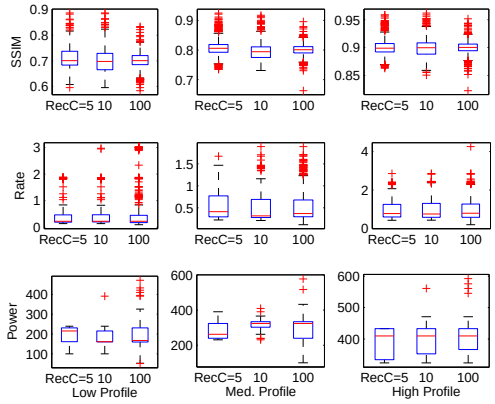
(a) Minimum Power Mode, `recC`=100.



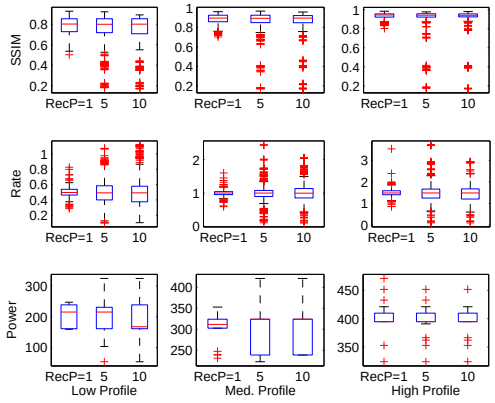
(b) Minimum Power Mode, `recP`=1.



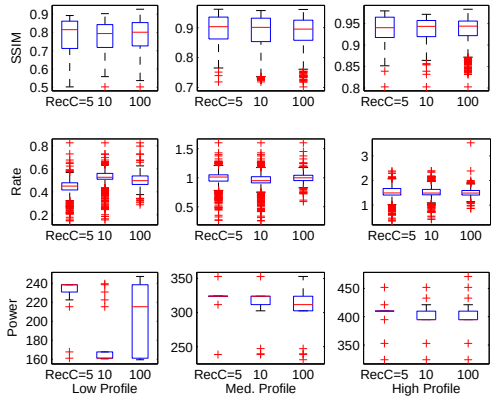
(c) Minimum Rate Mode, `recC`=100.



(d) Minimum Rate Mode, `recP`=1.

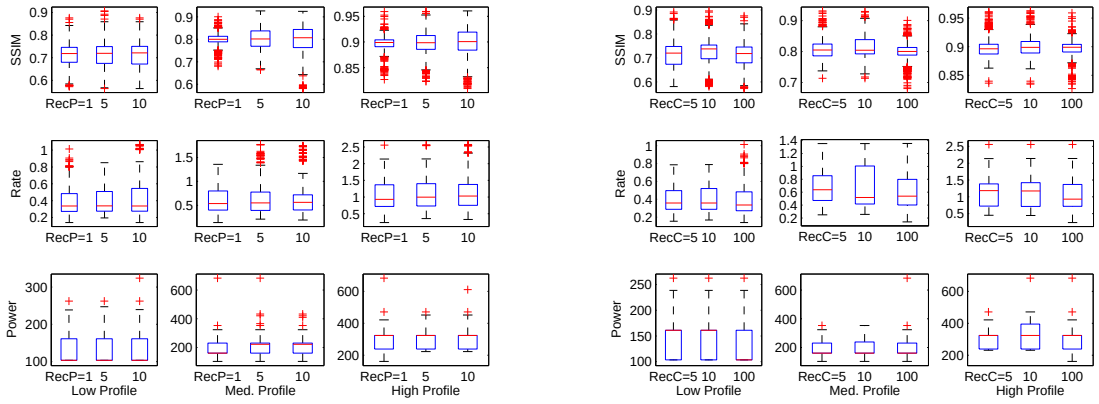


(e) Maximum Quality Mode, `recC`=100.



(f) Maximum Quality Mode, `recP`=1.

Figure 3.11: DRASTIC mode performance.



(a) Typical Mode, `recC`=100.

(b) Typical Mode, `recP`=1.

Figure 3.12: DRASTIC performance for the typical mode as a function of the reconfiguration period `RecP` and the number of reconfigurations `RecC`.

Table 3.4: A comparison of FPGA implementations of 2D DCTs. Dynamic power results are estimated for the operating frequency. Given the small number of cycles required by the proposed approach, it is clear that the proposed method yields the most energy efficient approach.

	Tumeo etc. [59]	Huang et al. [29]	Madanayake [51]	Sharma [60]	Yuebing [15]	Proposed
Arch. Structure	Single 1D DCT + ping pong TRAM	8 PE + TRAM	Two AI-DCT+ TBuffer	DA based	Double 1D DCT+ TRAM	Double 1D DCT+ ping pong TRAM
FPGA slices	2823	2944 (8 PEs)	2377-3618	1701	807-1657	257-1686
FPGA tech.	Xilinx Virtex II XC2VP30	Xilinx Virtex 4 XC4VSX35	Xilinx Virtex 6 XC6VLX240T	Xilinx Virtex II XC2VP30	Xilinx Virtex 5 XC5VLX110T	Xilinx Virtex 5 XC5VLX110T
Latency (cycles)	160	N/A	N/A	N/A	22	20
8×8 throughput (cycles)	64	25-102	N/A	N/A	16	8
Dyn. Power (mW)	N/A	24.03-26.27	897-1687	751	85.2-203.19	51-683
Max. Freq. (MHz)	107	N/A	123-308	45.17	200-275	157-251
Oper. Freq. (MHz)	N/A	41.79	123-308	45.17	100	100

Chapter 4

A Dynamically Reconfigurable Deblocking Filter for H.264/AVC Codec

4.1 Abstract

This paper introduces a dynamically reconfigurable implementation of deblocking filtering for H.264/AVC coding. The basic idea is to use a scalable approach that allows dynamic hardware reconfiguration of different modes based on power (or hardware complexity), bitrate, and image reconstruction quality.

The modes are arranged hierarchically. A complex mode includes all of the deblocking filtering options of simpler modes. This provides scalable performance where the use of additional hardware resources (or power) result in better rate-distiortion performance. Within each mode, the deblocking filter is selected based on boundary strength so as to balance computational complexity with performance improvement. An optimal mode approach is also introduced. In the optimal mode, the mode with the best reconstruction quality is selected

for each video frame. The optimal mode provides an upper bound (in the rate-distortion sense) to what can be achieved with the current modes.

The deblocking filters are implemented on a Virtex-5 FPGA. The results are validated over 11 commonly-used videos, where image quality is assessed using SSIM.

4.2 Introduction

The use of block transforms and quantization in modern video codecs can introduce significant blocking artifacts. The removal of such artifacts requires the use of deblocking filtering (DBF). Unfortunately, DBF comes with significant computational overhead due to the requirement for adaptivity to different types of block edges. Adaptivity requires execution of different code at different blocks and this can be a source of challenges for SIMD and pipelined implementations. Furthermore, the use of small blocksizes creates a need for a large number of memory accesses. Overall, execution time for DBF can account for a third of the whole computation complexity of an H.264/AVC decoder [61].

In what follows, we provide a brief summary of different methods used for addressing the computational complexity of DBF. In [62], the authors proposed implementing DBF as a co-processor which can interface with a general purpose or a DSP processor. In [63], the authors demonstrated an optimized SIMD implementation that delivered a speed up of 5.8 times. In [64], the authors implemented DBF in a dynamically reconfigurable embedded system. The authors studied the trade-offs of the use of different interconnection topologies and register file sizes used in a reconfigurable array of processors. In [65], the authors proposed the use of two DBF architectures. A high-performance, high-power DBF architecture was implemented at the 4×4 block level. A low-performance, low-power mode was implemented at the 16×16 macroblock level. In [66, 67], the authors developed parallel implementations at the block edge level. They reported on 3 configurations for parallel processing 1,2 or 4 block edges at the same time. They also suggested the use of dynamic reconfiguration among

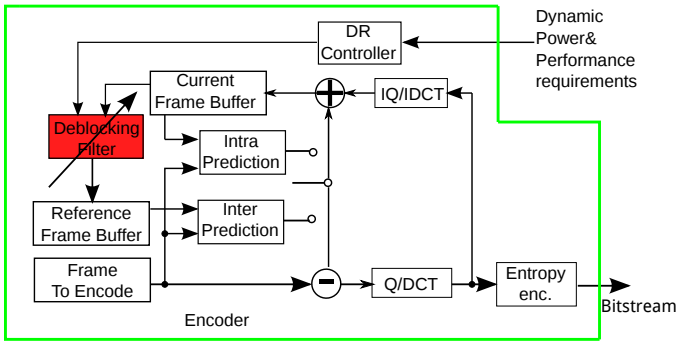


Figure 4.1: Dynamically reconfigurable DCT architecture for H.264 encoder.

the 3 configurations based on motion activity and the quantization parameter.

A summary of the contributions of the current paper includes: (i) a set of scalable DBF modes that balance trade-offs among image quality, bitrate, and dynamic power, (ii) a mode control algorithm that provides improved R-D performance over the use of any individual mode, and (iii) a rate-quality-power performance analysis of the DBF hardware modes. The system diagram of the H.264/AVC encoder that implements the proposed DBF system is shown in fig. 4.1. The basic blocks include intra/inter prediction, DCT transformation, quantization, and entropy encoding. In the decoding loop, results of reconstructed frames are input to the DBF module. We note that the Dynamically Reconfigurable (DR) controller selects the DBF mode based on Dynamic Power and performance requirements (bitrate and image quality).

The rest of the paper is organized as follows: The deblocking algorithm and related quality assessment is discussed in section 6.3. The proposed design methodology is presented in section 4.4. The results are discussed in section 7.5. Concluding remarks are given in section 4.6.

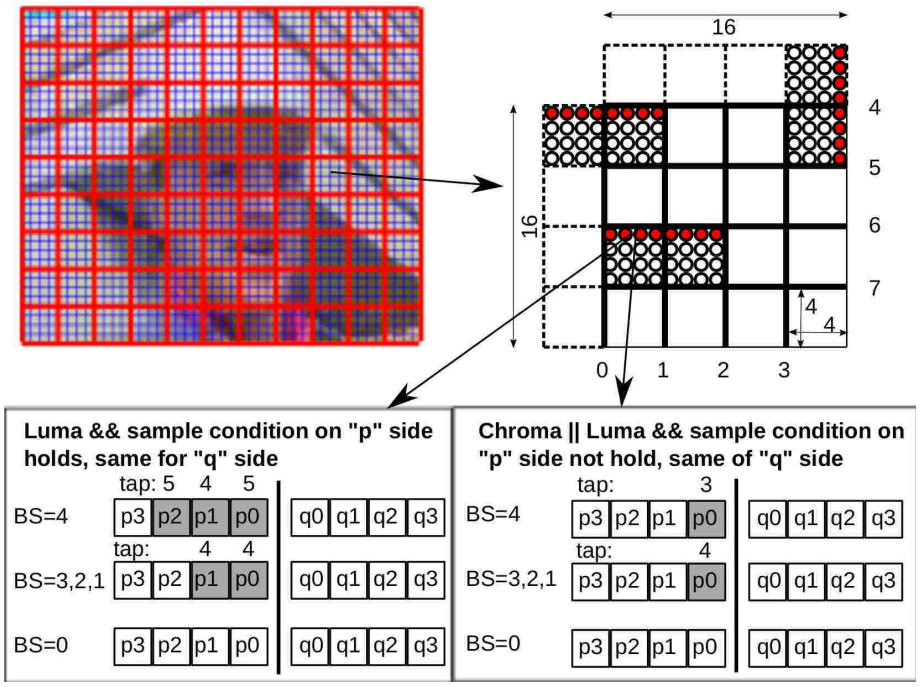


Figure 4.2: Deblocking filtering operation flow. Deblocking filtering is applied to the shaded pixels shown in the lower figures. For further details, we refer to [1].

4.3 Deblocking Algorithm and Deblocking quality assessment

The basic deblocking filtering steps are shown in Fig. 4.2. Deblocking filtering is applied at the Macroblock (MB) level, with vertical edges being filtered first, followed by horizontal edges. At the MB level, Luma is filtered first, followed by chroma (Cb and Cr).

The DBF has 3 levels of adapativity:

- **Slice:** At the slice level, DBF depends on Offset_A and Offset_B to adjust the threshold values given by α and β . Here, we are using default values as given in [1].
- **Edge:** Block edge level adaptivity depends on boundary strength (BS). BS takes

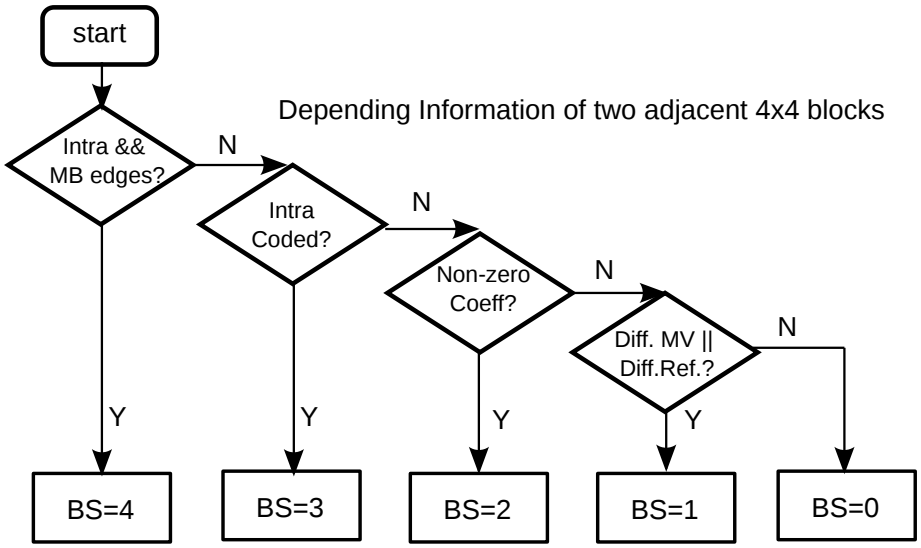


Figure 4.3: Assignment of boundary strength BS. Here, MV refers to motion vector and Diff. Ref. refers to a change in the reference frame.

values from 0 (no edge) to 4 (strong edge) that indicate the detected edge artifacts. The method for setting BS is shown in Fig. 4.3.

- **Sample:** Sample level adaptivity select filtering functions and parameters based on BS and sample values in the line of 8 pixels (horizontal or vertical) (see Fig. 4.2). To avoid over-filtering real-edges, DBF is only applied when the pixel intensities remain lower than the α, β parameters as given by:

$$|p_0 - q_0| < \alpha \qquad \text{and } |p_1 - p_0| < \beta \qquad \text{and } |q_1 - q_0| < \beta. \tag{4.1}$$

In terms of bitrate, [61] shows that the use of DBF can reduce bitrate requirements by 9% at equivalent PSNR levels. In terms of DBF image quality assessment, instead of PSNR, we also have metrics focused on edges. For example, block edges are assess by PSNR-B in [68]. Similarly, image quality focused on Canny-detected edges is assessed by Canny-Edge PSNR in [69]. Alternatively, perceptual quality assessment can be assessed by SSIM [18]. For the purposed of this paper, we will consider SSIM. We note that SSIM is described as a better

alternative to PSNR in [68].

4.4 Methodology

We are interested in developing a scalable approach that adjusts DBF filtering based on the boundary strength BS. We accomplish this by controlling how boundary strength determines the level of filtering. The basic idea is to apply DBF for stronger boundaries while possibly avoiding DBF for weaker edges. We provide 5 scalability levels represented by BSKill $= 0, 1, \ldots, 4$. We refer to Table 4.1 for the description of how BSKill controls the level of DBF filtering.

Table 4.1: Scalable DBF modes based on BSKill parameter setting.

BSKill	DBF filtering
0	edge is filtered if $BS > 0$ (e.g., $BS = 1, 2, 3, 4$), (full filter)
1	edge is filtered if $BS > 1$ (e.g., $BS = 2, 3, 4$)
2	edge is filtered if $BS > 2$ (e.g., $BS = 3, 4$)
3	edge is filtered if $BS > 3$ (e.g., $BS = 4$)
4	edge is filtered if $BS > 4$, (No filter operation)

To see how this works, note that increasing the value of BSKill will reduce the amount of filtering. Thus, small values of BSKill will provide more filtering that should result in better visual quality. The basic idea is demonstrated in Fig.4.4. In this example, decreasing BSKill does result in better visual quality. On the other hand, over-filtering can result in degradation of image quality. This problem is demonstrated in Fig. 4.5.

We also consider a greedy optimization method for selecting the BSKill value at each video frame based on the maximum image quality. The basic idea is to start with no-filtering (BSKill = 4) and keep decreasing BSKill to the full-filtering mode (BSKill = 0). In this scalable optimization approach, computational complexity is kept reasonable because

lower-levels reuse results from higher-levels and only need to apply DBF to the affected block edges (as opposed to the full image). In terms of computational-complexity, our approach is significantly less demanding than the method described in [69].

We provide a pipelined implementation for each DBF filtering level as shown in Figs. 4.6 and 4.7. In our implementation, α, β are read using table lookup at stage 1. The threshold conditions are implemented in the second stage (see definitions in the top of Fig. 4.6). The rest of the DBF filtering is implemented in two halves as described in the captions of Figs. 4.6 and 4.7. Similar to our implementation, [70, 71] provide a fixed-architecture implementation. Here, we note that our implementation is scalable (also see caption of Fig. 4.6) and dynamically reconfigurable.

4.5 Results

The DBF was tested on 11 videos (15 fps) of the first 64 QCIF (144x176) frames (BUS, CITY, CREW, FOOTBALL, FOREMAN, HARBOUR, MOBILE, SOCCER,MOTHER AND DAUGHTER, HALL MONITOR, SILENT). Over all videos, we measure bitrate in bits per pixel, image quality based on average SSIM, and dynamic DBF power. In addition to BSKill, we also vary the quantization parameter QP from 38 to 44 (for lower bitrates).

The average bitrate and image quality results are summarized in Tables 4.2 and 4.3. In terms of image quality, SSIM decreases with less filtering as shown in Table 4.3. The decrease is more significant at higher quantization levels (top to bottom). Bitrate behaves similarly. The rate-distortion performance is shown in Fig.4.8. From the R-D curve, we can see that the optimal mode can out-performs all other modes. The worst results are given by no-filtering (BSKill = 4). Also, full-filtering (BSKill = 0) is approximated by BSKill = 1.

The hardware synthesis results are summarized in Table 4.4. Dynamic power results are shown in Table 4.5 and Fig. 4.9. For dynamic power estimation, we used the FOREMAN

Table 4.2: Average bitrate results for all 11 videos. GOP encoding is "IPPPPPPP". See Table 4.3 for corresponding SSIM values. Opt refers to the optimal method that is described in Section 4.4.

	Opt.	BSKill =0	BSKill =1	BSKill =2	BSKill =3	BSKill =4
QP=38	0.0998	0.0996	0.0997	0.1004	0.1009	0.1014
QP=39	0.0900	0.0897	0.0898	0.0904	0.0911	0.0917
QP=40	0.0780	0.0778	0.0779	0.0785	0.0792	0.0797
QP=41	0.0686	0.0683	0.0682	0.0688	0.0695	0.0700
QP=42	0.0601	0.0597	0.0599	0.0603	0.0610	0.0615
QP=43	0.0552	0.0549	0.0551	0.0553	0.0560	0.0565
QP=44	0.0476	0.0473	0.0475	0.0478	0.0484	0.0490

Table 4.3: Average SSIM results for the setup described in Table 4.2.

	Opt.	BSKill =0	BSKill =1	BSKill =2	BSKill =3	BSKill =4
QP=38	0.8212	0.8193	0.8189	0.8180	0.8163	0.8135
QP=39	0.8037	0.8016	0.8014	0.8004	0.7980	0.7947
QP=40	0.7831	0.7803	0.7800	0.7792	0.7771	0.7733
QP=41	0.7581	0.7551	0.7549	0.7542	0.7513	0.7464
QP=42	0.7352	0.7314	0.7311	0.7307	0.7284	0.7221
QP=43	0.7170	0.7142	0.7135	0.7129	0.7098	0.7026
QP=44	0.6843	0.6811	0.6804	0.6803	0.6770	0.6687

video. Dynamic power is estimated by Xilinx's XPower tool based on the reconstructed FOREMAN frames (prior to DBF) together with the synthesized DBF netlist files. We note the scalability of the dynamic power measurements demonstrated in Fig. 4.9. From the results, it is clear that we can select a hardware implementation that can meet certain dynamic power constraints on the DBF.

Table 4.4: Deblocking filter synthesis results. DBF was implemented on XC5VLX110T(Virtex 5) device. Clock frequency was constrained to 100 MHz.

	BSKill =0	BSKill =1	BSKill =2	BSKill =3	BSKill =4
Slice Registers	449	442	436	372	131
Slice LUTs	571	575	565	341	128
Slices	311	300	289	184	39
max. freq.(Mhz)	126	126	126	169	529

Table 4.5: Power simulation results for Foreman video. Dynamic power is estimated using XPower.

	BSKill =0	BSKill =1	BSKill =2	BSKill =3	BSKill =4
QP=38	159.0100	159.1900	156.6700	143.2400	122.8400
QP=39	157.9200	158.0900	155.5800	142.5900	122.8300
QP=40	156.1200	156.2900	153.8300	140.8300	120.5800
QP=41	156.3600	156.5300	154.0700	141.1600	121.7000
QP=42	156.3900	156.5700	154.1000	141.1300	121.5200
QP=43	152.5000	152.6900	150.2400	137.5400	116.4100
QP=44	144.7600	145.0000	142.6400	131.7400	108.8200

4.6 Conclusion

In this chapter, we described a scalable set of pipelined DBF implementations for dynamic reconfiguration based on video frame performance metrics. Each DBF implementation was tested on 11 standard videos for their performance in terms of bitrate, reconstruction image quality, and dynamic power. We also introduced an optimal method that selects the DBF implementation that provides the DBF with the best image quality. Future work could be focused on the development of incorporating DBF into the emerging High-Efficiency Video Coding (HEVC) standard.

(a) Foreman: Frame2 Original (b) Foreman:QP=42,BSKill=0

(c) Foreman:QP=42,BSKill=1 (d) Foreman:QP=42,BSKill=2

(e) Foreman:QP=42,BSKill=3 (f) Foreman:QP=42,BSKill=4

Figure 4.4: Deblocking filter results for foreman video frame 2 where DBF works as expected. In this example, we are using H.264/AVC JM 18.2 with $QP = 42$. We demonstrate the results for: (a) original image, (b) $BSKill = 0$ gives SSIM=0.8182, (c) $BSKill = 1$ gives SSIM=0.8155, (d) $BSKill = 2$ gives SSIM=0.8145, (e) $BSKill = 3$ gives SSIM=0.8005, and (f) $BSKill = 4$ gives SSIM=0.7882.

(a) Mobile: Frame11 Orginal (b) Mobile: QP=42,BSKill=0

(c) Mobile:QP=42,BSKill=1 (d) Mobile:QP=42,BSKill=2

(e) Mobile:QP=42,BSKill=3 (f) Mobile:QP=42,BSKill=4

Figure 4.5: Deblocking filter results for mobile video frame 11 where DBF can perform worse than expected. In this example, we are using H.264/AVC JM 18.2 with $QP = 42$. We demonstrate the results for: (a) original image, (b) $BSKill = 0$ gives SSIM=0.7203, (c) $BSKill = 1$ gives SSIM=0.7199, (d) $BSKill = 2$ gives SSIM=0.7194, (e) $BSKill = 3$ gives SSIM=0.7358, and (f) $BSKill = 4$ gives SSIM=0.7365. To see the effects of over-filtering, compare the two ellipsoidal regions shown in (b) with the original image in (a), and the rest of the sub-figures.

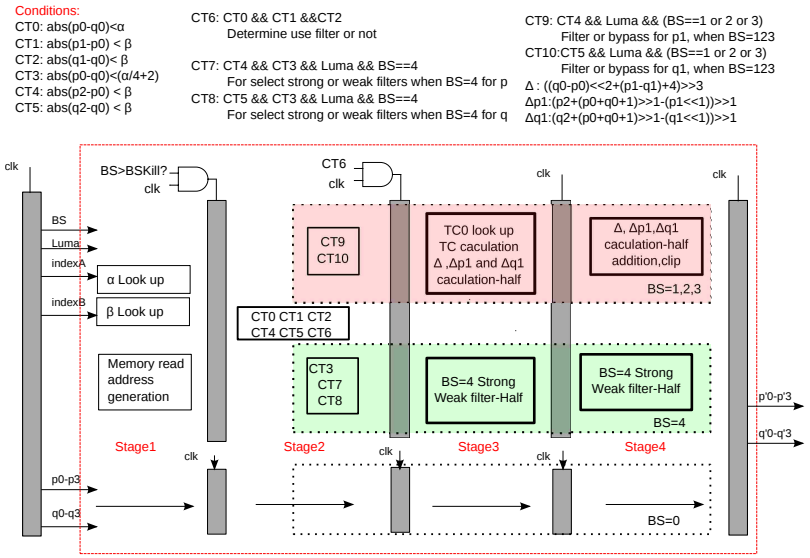


Figure 4.6: Deblocking filter implementation using a 4-stage pipeline. The diagram shows the full-filter implementation (BSKill = 0). The other 4 modes are implemented by simply removing logic from the full-filter. The pink regions are implemented for weaker-edges ($BS = 1, 2, 3$). The green regions are implemented for strong edges ($BS = 4$). In the bottom of the figure, we have a FIFO implementation that is used for the case of no filtering ($BS = 0$).

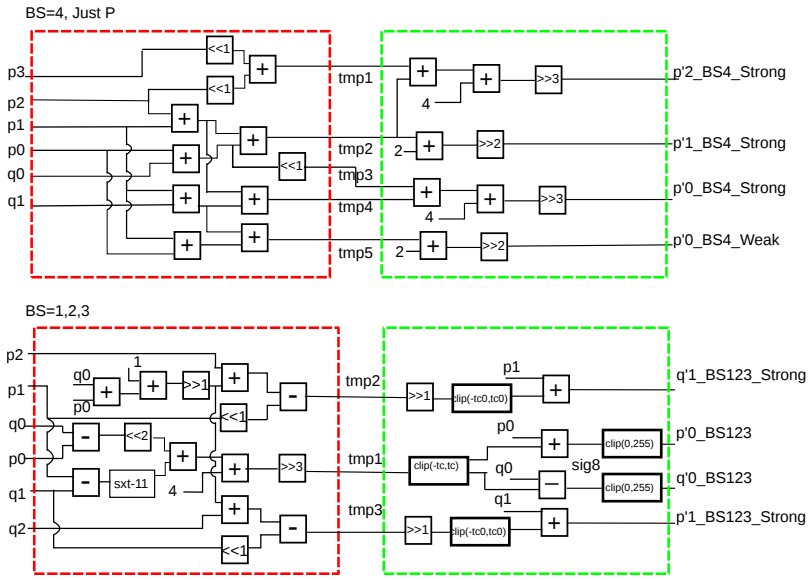


Figure 4.7: A two-stage pipeline implementation of the strong and weaker filter halves (see Fig. 4.6).

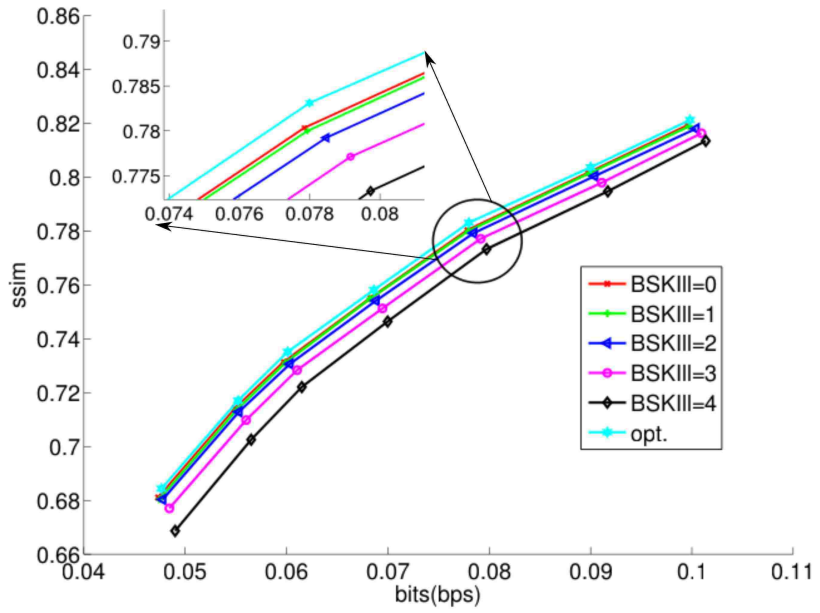


Figure 4.8: R-D performance with BSKill mode and opt. mode. Tested over 11 QCIF videos, each video has 64 frames, with "IPPPPPPP" GOP.

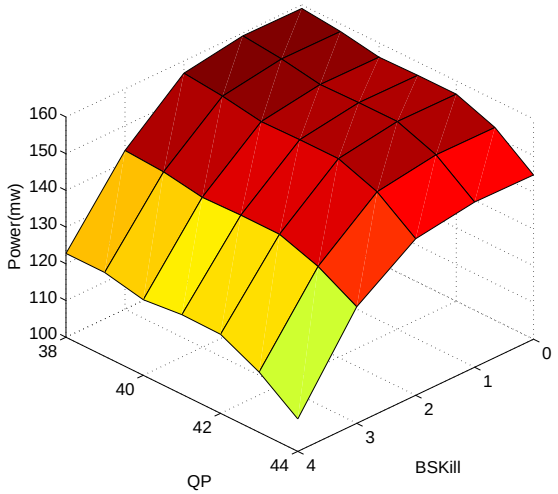


Figure 4.9: Dynamic power as a function of BSKill and QP. Refer to Table 4.5.

Chapter 5

High Efficiency Video Coding (HEVC)

HEVC [2] is the emerging video compression standard that was finalized in 2013. HEVC seeks to provide higher coding efficiency to support the growing propularity of HD video and the higher video frame sizes of 2K ($4K \times 2K$), 4K ($8K \times 4K$). Subjective tests for WVGA and HD video sequences suggest that HEVC encoders can achieve the same subjective video quality as the H.264/MPEG-4 AVC (current standard) while reducing bitrate requirements by approximately 50% [72].

HEVC builds upon the current video standard based on H.264/AVC. H.264/AVC was initially developed during 1999-2003, and was further extended to support scalable video coding (SVC) and multi-view video coding (MVC) during 2003-2009. In 2004, the ITU-T Video Coding Experts Group (VCEG) began research on advances to be included in a new video compression standard or enhancements to the H.264/MPEG-4 AVC standard. In October 2004, various techniques for potential enhancement of the H.264/MPEG-4 AVC standard were investigated. In Oct 2005, at the Nice (FR) meeting of VCEG, VCEG began designating certain topics as "Key Technical Areas" (KTA) for further investigation. A

software codebase called the KTA codebase was established for evaluating such proposals [73].The KTA software was based on the Joint Model (JM) reference software that was developed by the MPEG & VCEG Joint Video Team for H.264/MPEG-4 AVC. Additional technologies were integrated into the KTA software and tested in experiment evaluations over the next four years [74]. A formal joint Call for Proposals(CfP) started by VCEG and MPEG was issued in January 2010. Companies that are working on HEVC and have also shown specific encoders and decoders include Qualcomm, Ericsson, Allegro DVT, Vanguard Software Solutions(VSS), Rovi Corporation and ATEME.

5.1 Background and Related work on HEVC

HEVC uses a *hybrid* approach by including modes from previous video coding standards that support both inter and intra prediction and transform coding to achieve high reconstruction image quality at low bitrates. The video encoder diagram is shown in Fig. 5.1.

At the picture level, each picture is split into block-shaped regions where the original block is termed the largest coding unit (LCU). The LCU can be subdivided into smaller coding units (CUs).

Pictures are organized into groups of pictures (GOPs). The first picture of a video sequence or a GOP is coded using intra-picture prediction. Here, intra-prediction implies implies that encoding is based on prediction within the picture itself, without referencing other pictures. The remaining pictures of GOP can use inter and intra prediction. Here, for inter-picture prediction, motion vectors (MV) are used to determine how blocks from a different picture should be used to predict the current block (motion compensation (MC)). After prediction, the residual data is transformed (integer transform), scaled, and quantized. The quantized results are entropy encoded. Based on the quantized data, the encoder will reconstruct the decoder so as to use the decoded picture(s) for further prediction. To reconstruct the decoded picture, the encode applies inverse quantization and inverse transform. Loop

filtering is applied to smooth out blocking artifacts induced by the block-wise processing. The reconstructed frames are saved in the decoded picture buffer for future prediction.

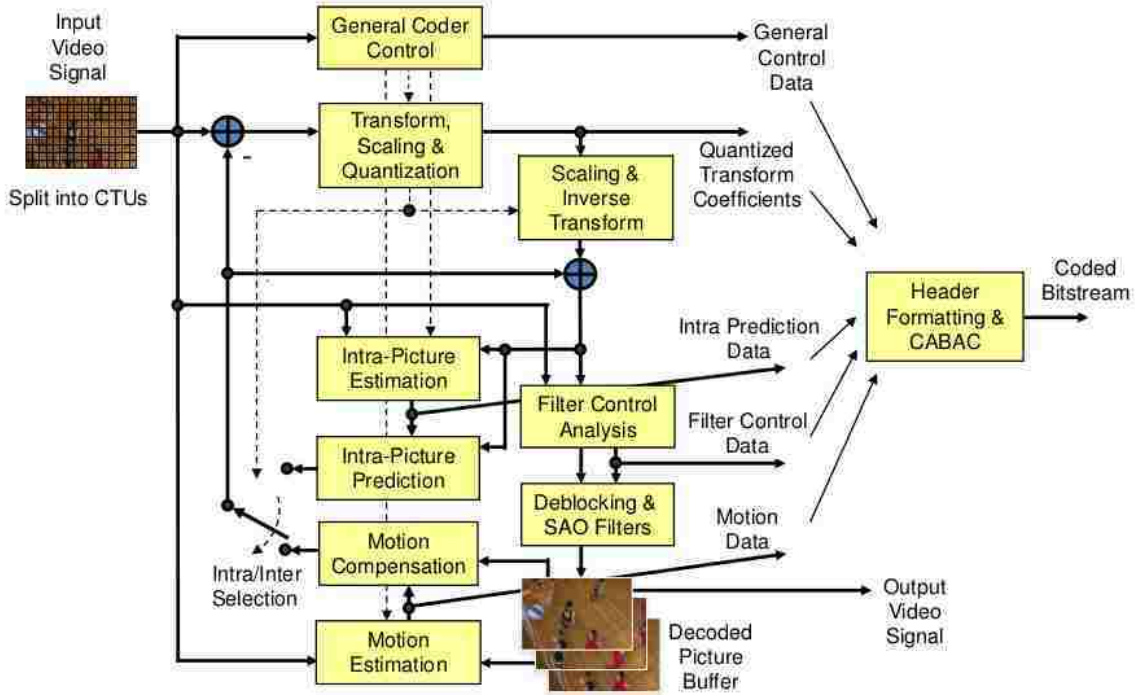


Figure 5.1: HEVC encoder diagram [2].

5.1.1 Coding Tools

Block Structure

HEVC replaces macroblocks used in previous standards with a new coding scheme that uses larger block structures of up to 64×64 pixels and sub-partition schemes that support variable-sized decompositions. The primary goal of this partitioning concept is to provide a high degree of adaptability for both temporal and spatial prediction as well as for the purpose of space-frequency representation of prediction residuals [3]. HEVC initially divides

the picture into coding tree units (CTUs) which contain one luma coding tree block (CTB) and two chroma CTBs. A CTB can be 64×64, 32×32, or 16×16, with larger block sizes usually resulting an increased coding efficiency. CTBs are then divided into coding units (CUs). The arrangement of CUs within a CTB is known as a quad-tree since a subdivision results in four smaller regions. CUs are then divided into prediction units (PUs) of either intra-picture or inter-picture prediction type which can vary in size from 64×64 to 4×4. Here, we note that prediction units coded using 2 reference blocks, known as bipredictive coding, are limited to sizes of 8×4 or 4×8 so as to save on memory bandwidth. The prediction residual is coded using transform units (TUs) which contain coefficients for spatial block transform and quantization. A TU can be 32×32, 16×16, 8×8, or 4×4. An example of quad-tree subdivision is shown in Fig. 5.2. Note that for inter prediction, the blocks use asymmetric partitions as shown in fig.5.3. However, asymmetric partitions are not allowed in intra-prediction.

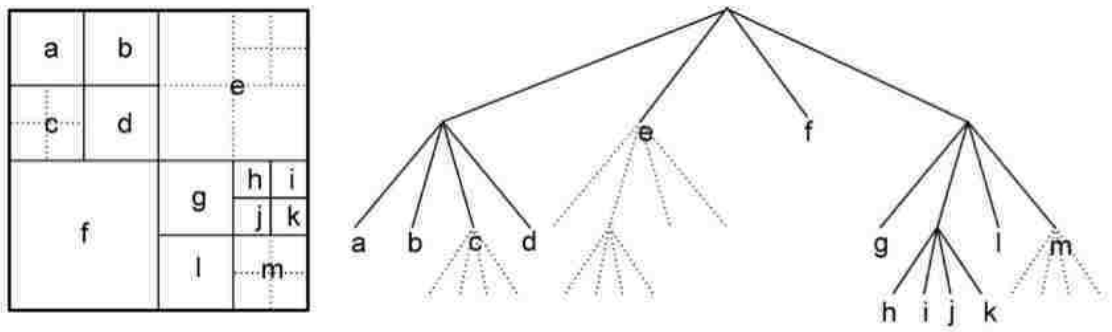


Figure 5.2: Example of a nested quad-tree structure (right part) for dividing a given coding tree block (left part, in black) into prediction blocks (solid gray lines) and transform blocks (dashed gray lines) of variable size. The order of parsing the prediction blocks follows their labeling in alphabetical order [3].

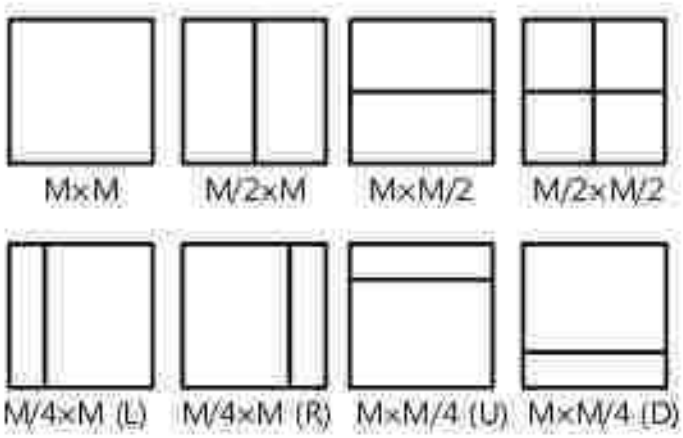


Figure 5.3: Modes for splitting a CB into PBs for inter prediction. For intra prediction, only quad-tree spliting is allowed [2].

Intra Picture Prediction

Direction prediction with 33 different directional orientations is defined for PU sizes from 4×4 up to 32×32. Besides the 33 modes for directional prediction, we also have planar prediction and DC prediction. The 35 prediction modes are shown in Fig. 5.4.

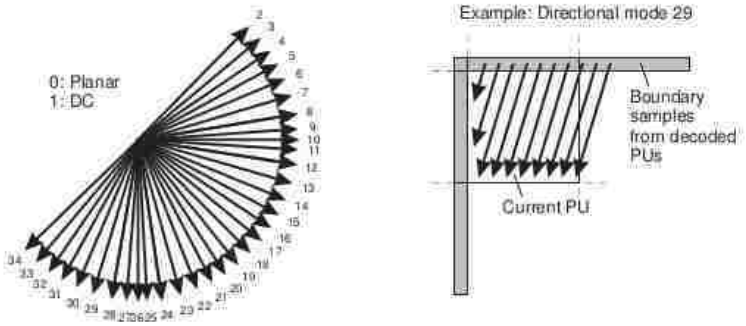


Figure 5.4: The 35 intra-prediction modes using 33 directions [2].

Each predicted sample $P_{x,y}$ is obtained by projecting its location to the reference row of

pixels and applying the selected prediction direction using interpolation to achieve 1/32-th pixel accuracy. Linear interpolation between the two closest reference samples is achieved using [75]:

$$P_{x,y} = ((32 - w_y) \cdot R_i + w_y \cdot R_{i+1} + 16) >> 5 \tag{5.1}$$

where R_i is the i^{th} reference sample on the reference row, R_{i+1} is the next reference sample, and w_y is a weighting factor between the two reference samples corresponding to the projected sub-pixel location in between R_i and R_{i+1}. The reference sample index i and the weighting parameter w_y are calculated based on the projection displacement d associated with the selected prediction direction. Here, d depends on the tangent of the prediction direction in units of 1/32-th of a sample and having a value from -32 to +32. The basic parameters are determined using:

$$\begin{aligned} c_y &= (y \cdot d) >> 5 \\ w_y &= (y * d)\&31 \\ i &= x + c_y. \end{aligned} \tag{5.2}$$

Transformation, scaling and quantization

The residual block in HEVC is partitioned into multiple square TBs. Possible TB sizes are 4×4, 8×8, 16×16 and 32×32. The standard defined the transform matrix for 32×32. Other transform matrix sizes can be derived from the 32×32 matrix (see Fig. 5.6). To maintain intermediate values within a 16-bit memory, a 7-bit right shift and a 16-bit clipping operation is inserted after the first 1D DCT. Unlike H.264/AVC, the DCT transform matrix in HEVC uses uniform scaling avoiding the use of frequency-specific scaling.

For TU size of 4×4, an alternative integer transform is used that is based on the discrete

Figure 5.5: Transform matrix for HEVC standard.

sine transform (DST) as given by:

$$H = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}. \tag{5.3}$$

The DST for 4×4 TU size provides approximately 1% bit rate reduction in intra predictive coding. Additional coding efficiency improvement for other sizes was not found to be marginal.

HEVC applies essentially the same quantization scheme as in H.264/AVC. The Quantization parameter (QP) ranges from 0 to 51. The Quantization step size doubles when QP increases by 6.

Entropy Coding

HEVC abandons the use of Context Adaptive Variable Length Coding (CAVLC) and only supports the use of Context Adaptive Binary Arithmetic Coding (CABAC). The CABAC is essentially the same as for H.264/AVC. A summary of some minor changes are presented next.

Coefficient scanning is performed in 4×4 sub-blocks for all TB sizes. Three coefficient scanning methods are selected implicitly for coding the intra-coded transform coefficients of 4×4 and 8×8 TB sizes. For inter prediction modes, 4×4 diagonal up-right scan is applied to all sub-blocks of transform coefficients.

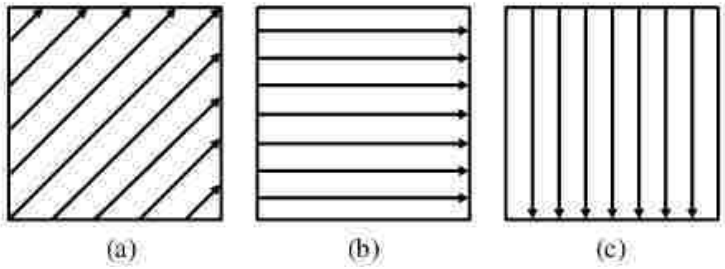


Figure 5.6: Coefficient scanning direction in HEVC. The intra mode implicitly selects the 3 modes. The inter mode only uses the diagonal up-right mode.

HEVC encodes the position of the last non-zero transform coefficient, a significance map, sign bits and levels for the transform coefficients. The position of the last non-zero transform coefficient in terms of the i-th and j-th coordinates are coded and compressed. The significance map is derived for the non-zero coefficients related to a 4×4 block.

In loop filters

In HEVC, two processing steps, the deblocking filter (DBF) followed by a sample adaptive offset (SAO) operation are applied to the reconstructed samples before writing them into the

decoded picture buffer in the decoder loop [2]. DBF is only applied to the samples located at block boundaries while the SAO is applied adaptively to all samples that satisfy certain conditions.

DBF is applied to all samples adjacent to PU or TU boundaries that do not fall on image boundaries. HEVC only applies DBF on an 8×8 sample grid for both luma and chroma samples.

SAO modifies the samples after the deblocking filter through a look-up table. It's a non-linear operation which allows additional minimization of the reconstruction error and enhance edge sharpness. SAO has 3 modes: (i) a disabled mode, (ii) an edge mode, and (iii) the band mode. In edge mode, each sample is offseted by values derived from the look-up table to either enhance edge sharpness or reduce artifact edges. In band mode, the samples in CTB are splitted into 32 bands by sample amplitude ranges. The sample values belonging to four consecutive bands are modified by adding the values denoted as band offsets.

Chapter 6

DRASTIC for HEVC intra-prediction mode implementation

6.1 Abstract

The High Efficiency Video Coding (HEVC) standard can achieve significant improvements in coding performance over H.264/AVC. To achieve significant coding improvements in intra-predictive coding, HEVC relies on the use of an extended set of intra-prediction modes and prediction block sizes. This chapter presents a unified hardware architecture for implementing all 35 intra-prediction modes that include the planar mode, the DC mode, and all angular modes for all prediction unit (PU) sizes ranging from 4×4 to $64 \times 64 pixels$. We propose the use of a unified reference sample indexing scheme that avoids the need for sample re-arrangement suggested in the HEVC reference design. The hardware architecture is implemented on a Xilinx Virtex 5 device (XC5VLX110T) for which we report power measurements, resource utilization, and the average number of required cycles per pixel.

6.2 Introduction

Among the most significant coding tools that improve over H.264/AVC, HEVC requires the implementation of extended prediction modes that need to work for different prediction unit (PU) sizes [2]. For intra-prediction, pre-decoded border pixels are used for predicting the entire PU. Depending on the prediction mode, predicted blocks can be reconstructed with: (i) the planar mode that uses bilinear interpolation. (ii) the DC mode that uses an average of the above and left reference pixels, (iii) the angular modes that use linear interpolation based on decoded pixels from the main-reference border or extrapolated reference pixels from the side-reference border. Under general testing conditions, PU sizes can vary from the smallest block of 4×4 to the largest block of 64×64. The goal of this paper is to provide a unifying framework that implements all of the 35 modes for block sizes from 4×4 to 64×64 while supporting parallelism in a pipelined architecture implementation.

To the best of our knowledge, current reported hardware implementations of the HEVC intra prediction either do not support all the prediction modes or don't support all the block sizes. In [10], the authors describe an intra-prediction architecture restricted to the angular modes on 4×4 blocks. The authors provide a *flexible reference sample selection* approach to minimize memory accesses. A more complete implementation was recently reported in [76]. In [76], the authors considered all except for the planar mode and gave implementations for all PU sizes. Instead of the unified approach proposed in this paper, the authors considered separate data paths for the horizontal and vertical data paths. To deliver higher performance, multipliers were implemented using custom shifters and adders that achieved a 500Mhz maximum operating frequency on IBM 65nm technology. In [77] and [78], the authors implemented the HEVC intra-prediction modes on 4×4 and 8×8 PU sizes using pixel equality based computation reduction (PECR) to reduce the energy consumption.

The motivation for this research comes from the need to provide an efficient implemen-

tation for all modes using a single circuit. The proposed approach uses unified reference sample indexing that avoids wasting cycles on re-arrangement or filling-in reference samples to a new buffer. A pipelined approach combines components from several modes to reduce the overall hardware footprint. Furthermore, the proposed pipelined circuit allows parallel processing of the different prediction modes that facilitates rate-distortion optimization that can be used to select an optimal mode for each block.

The rest of the chapter is organized as follows. In section 6.3, we describe the proposed unified approach. In section III, we discuss the pipelined implementation and system integration. We provide verification and implementation results on Virtex 5 FPGA in section IV. Concluding remarks are given in section V.

6.3 Unified Reference Sample Indexing and Accessing

HEVC intra prediction includes 33 angular direction modes (modes 2-34), an intra planar mode (mode 0), and an intra DC mode (mode 1) [79]. In the reference design provided by the standard draft and model software HM, intra prediction is implemented by re-arranging the reference samples into a new reference buffer according to the mode and then do the prediction. In the proposed approach, we will compute each prediction sample directly from original reference sample buffer using a unified indexing scheme and thus avoid the use of an intermediate buffer. The basic idea is that we can create a parallel and pipelined architecture for our unified indexing approach. Several modes were allowed in the pipeline to be computed at the same time for different stages without a lot of initial delay overhead.

6.3.1 Unified reference sample indexing

We set the reference samples as shown in Fig. 6.1. For each (x, y) sample in the predicted block, we use a reference indexing equation (or equations) to map back to the reference

sample(s) based on the prediction mode. For generality, we allow for different PU size ($nT \times nT$, nT=4,8,16,32,64), and map the *left/left-bottom* column and *upper/upper-right* row reference samples to the input buffer as shown in Fig. 6.1. To accommodate for the largest possible prediction unit, we allocate 257 bytes to the input buffer.

In what follows, let P_{y*nT+x} denotes the (x, y)-th prediction sample. We let R_i to index samples from the 1D input/reference buffer along the four sides and the upper-left corner pixel of the prediction block. Depending on the previous encoding/decoding conditions, the top-right row and left-bottom column pixels can be generated by extrapolation from the other two sides, and the $4nT+1$ reference pixels will always exist. Thus, our unified indexing approach computes P_{y*nT+x} using 1D indices from R_i.

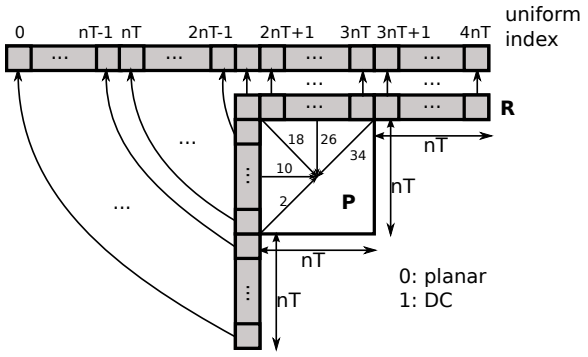


Figure 6.1: Unified reference sample indexing for PU size of $nT \times nT$. Here, $4nT+1$ reference pixels R are used to obtain $nT \times nT$ predicted samples P. The prediction directions are shown for several prediction modes (2,10,18,26,34).

6.3.2 Planar mode

In the planar mode, we use bilinear interpolation to generate P_{y*nT+x} using weighted-averaging over the 4 reference samples as given by:

$$\begin{aligned} P_{y*nT+x} &= ((nT-1-x)\cdot R_{2nT-1-y} \\ &\quad +(x+1)\cdot R_{3nT+1} \\ &\quad +(nT-1-y)\cdot R_{2nT+1+x} \\ &\quad +(y+1)\cdot R_{nT-1}+nT) \\ &\quad \gg (log2(nT)+1). \end{aligned} \tag{6.1}$$

where $x, y \in [0, nT-1]$ to cover all of the prediction block pixels.

6.3.3 DC mode

In the DC mode, the DC value is computed based on the average of all of the reference pixels except for corner pixels. Then, all of the internal pixels ($x, y \in [2, nT-1]$) are given by:

$$\begin{aligned} dcVal &= (\textstyle\sum_{x=0}^{nT-1} R_{2nT+1+x} + \sum_{y=0}^{nT-1} R_{2nT-1-y} \\ &\quad +nT) \gg (log2(nT)+1) \\ P_{y*nT+x} &= dcVal. \end{aligned} \tag{6.2}$$

The first row and first column pixels use weighted averaging of the DC value and the nearest neighbour as given by:

$$\begin{aligned} P_{0*nT+0} &= (R_{2nT-1}+2*dcVal+R_{2nT+1}+2)\gg 2 \\ P_{0*nT+x} &= (R_{2nT+1+x}+3*dcVal+2)\gg 2 \\ &\quad x\in[1,nT-1] \\ P_{y*nT+0} &= (R_{2nT-1-y}+3*dcVal+2)\gg 2 \\ &\quad y\in[1,nT-1]. \end{aligned} \tag{6.3}$$

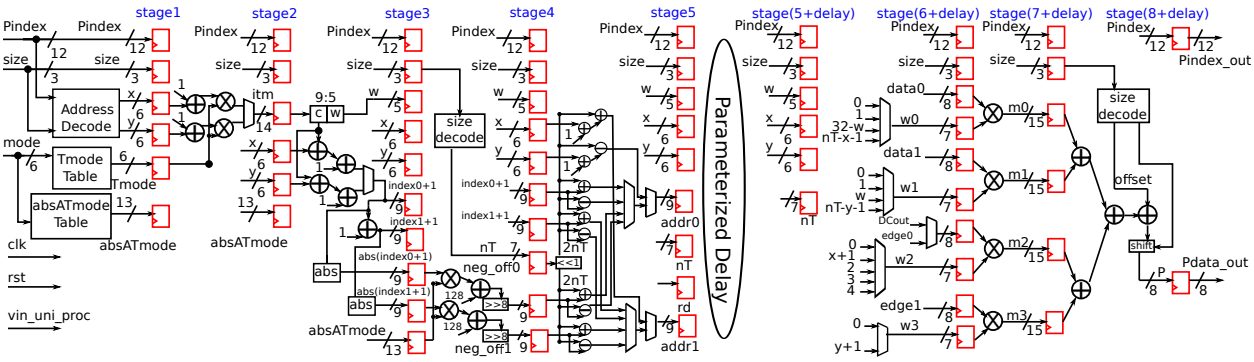


Figure 6.2: Datapath for the pipelined *uni_proc* circuit using $size = log2(nT) - 2$, $Pindex = y \times nT + x$, and $mode \in [0, 34]$. Parameter *delay* is used to notify number of cycles for RAM operation between address assertion and data to be ready. When using BRAM on virtex 5 FPGA, $delay = 2$.

6.3.4 Angular mode

In the angular modes, the predicted pixels use at most two reference samples depending on the direction. Refer to Table 6.1 for the correspondence between the integer modes and the prediction angle parameters. A lookup table is used for computing the trigonometric parameters needed in the prediction. Directional modes are defined in terms of the prediction angle A_{mode}. Horizontal modes are defined for mode numbers $2 - 17$. For prediction, horizontal modes use **main reference** samples from the left column and **side reference** samples from the top row. Vertical modes are defined for mode numbers 18-34. Similar to horizontal modes, vertical modes use **main reference** samples from the top row and **side reference** samples from the left column. Extrapolation is needed when the prediction angle satisfies $tan(A_{mode}) < 0$ (modes $16 - 25$). The primary angular mode equation based on weighted average is given by:

$$\begin{aligned} P_{y*nT+x} &= ((32 - w) * R_{tId0} + w * R_{tId1} + 16) \gg 5 \\ tId0 &= 2 * nT + off0 \\ tId1 &= 2 * nT + off1. \end{aligned} \tag{6.4}$$

To compute the weighting factor w, we use:

$$\begin{aligned} itm &= \begin{cases} (x+1) * T(mode) & \text{hor. mode} \\ (y+1) * T(mode) & \text{ver. mode} \end{cases} \\ c &= itm \gg 5 \\ w &= itm \;\&\&\; 31. \end{aligned} \tag{6.5}$$

For computing the indices $tId0, tId1$ of (6.4), we first compute the intermediate indices $index0, index1$ using:

$$\begin{aligned} index0 &= \begin{cases} y + c & \text{hor. mode} \\ x + c & \text{ver. mode} \end{cases} \\ index1 &= index0 + 1 \\ invA &= abs(AT(mode)). \end{aligned} \tag{6.6}$$

Then, we have the negative offsets given by:

$$\begin{aligned} neg\_off0 &= (128 + abs(index0 + 1) * invA) \gg 8 \\ neg\_off1 &= (128 + abs(index1 + 1) * invA) \gg 8, \end{aligned} \tag{6.7}$$

which can be used to compute $off0, off1$ using:

$$off0 = \begin{cases} neg\_off0; & index0 \text{ <-1, hor. mode} \\ -1 - index0; & index0 \text{ $\geq$-1, hor. mode} \\ -neg\_off0; & index0 \text{ <-1, ver. mode} \\ 1 + index0; & index0 \text{ $\geq$-1, ver. mode} \end{cases} \tag{6.8}$$

$$off1 = \begin{cases} neg\_off1; & index1 \text{ <-1, hor. mode} \\ -1 - index1; & index1 \text{ $\geq$-1, hor. mode} \\ -neg\_off1; & index1 \text{ <-1, ver. mode} \\ 1 + index1; & index1 \text{ $\geq$-1, ver. mode.} \end{cases} \tag{6.9}$$

Table 6.1: Look up table for angle A_{mode} parameters for angular prediction mode. $T(mode) = 32 \times tan(A_{mode})$, $AT(mode) = 256 \times actan(A_{mode})$

mode	2	3	4	5	6	7	8
$T(mode)$	32	26	21	17	13	9	5
$AT(mode)$	—	—	—	—	—	—	—
mode	9	10	11	12	13	14	15
$T(mode)$	2	0	-2	-5	-9	-13	-17
$AT(mode)$	—	—	-4096	-1638	-910	-630	-482
mode	16	17	18	19	20	21	22
$T(mode)$	-21	-26	-32	-26	-21	-17	-13
$AT(mode)$	-390	-315	-256	-315	-390	-482	-630
mode	23	24	25	26	27	28	29
$T(mode)$	-9	-5	-2	0	2	5	9
$AT(mode)$	-910	-1638	-4096	—	—	—	—
mode	30	31	32	33	34		
$T(mode)$	13	17	21	26	32		
$AT(mode)$	—	—	—	—	—		

6.4 Methodology and Implementation

6.4.1 Pipeline Organization

We unify all of the modes using a pipelined approach. The number of stages of the pipeline will depend on the RAM type used to save the reference pixels. We provide the pipelined architecture in Fig.6.2 and summarize the pipeline stages below:

Stage 1: Compute $T(mode)$ and $AT(mode)$ using the lookup table. Determine x and y from 1D index P_{index}.

Stage 2: Compute itm as given in eq.6.5.

Stage 3: Compute c and w as given in eq.6.5.

Generate $index0$ and $index1$ as given in eq.6.6.

Stage 4: Compute neg_off0 and neg_off1 as given in eq.6.7.

Stage 5: Compute $addr0 = tId0$ and $addr1 = tId1$ as given in eq. 6.4.

In angular mode, compute $off0$ and $off1$

as given by eq. 6.8 and 6.9.

In DC or planar mode compute:

$off0 = -1 - y$ and $off1 = 1 + x$.

Stage(s) $6 : 5 + $ delay: Based on RAM type, we need:

One cycle (delay=1) for register-based RAM or

Two cycle delay (delay=2) for dual-port BRAM.

Stage $6 + $ delay: Use $data0 = R_{addr0}$ and $data1 = R_{addr1}$.

in the following computations.

For Planar or DC mode:

Select $edge0$, $edge1$ or $dcVal$ based on the mode.

Select the weights for input pixels $w0, w1, w2, w3$.

Stage $7 + $ delay: Compute $m(i) = w(i) \times R_{tId(i)}, i \in [0 - 3]$.

Stage $8 + $ delay: Compute predicted pixel P_{data_out} as:

$P_{data\_out} = (m0 + m1 + m2 + m3 + offset) \gg shift$.

6.4.2 System Integration

The integrated system is given in Fig. 6.3. The system consists of 4 components *BRAM*, *edge*, *DCval* and *uni_proc* circuit. The *BRAM* circuit is used to store the decoded reference samples. The *edge* circuit is only used in planar mode for accessing R_{nT-1} and R_{3nT+1}. The *DCval* circuit is only used in the DC mode for computing $dcVal$. The *uni_proc* circuit implements the pipeline stages described in subsection 6.4.1. The integrated system uses a finite state machine to control the 4 component circuits.

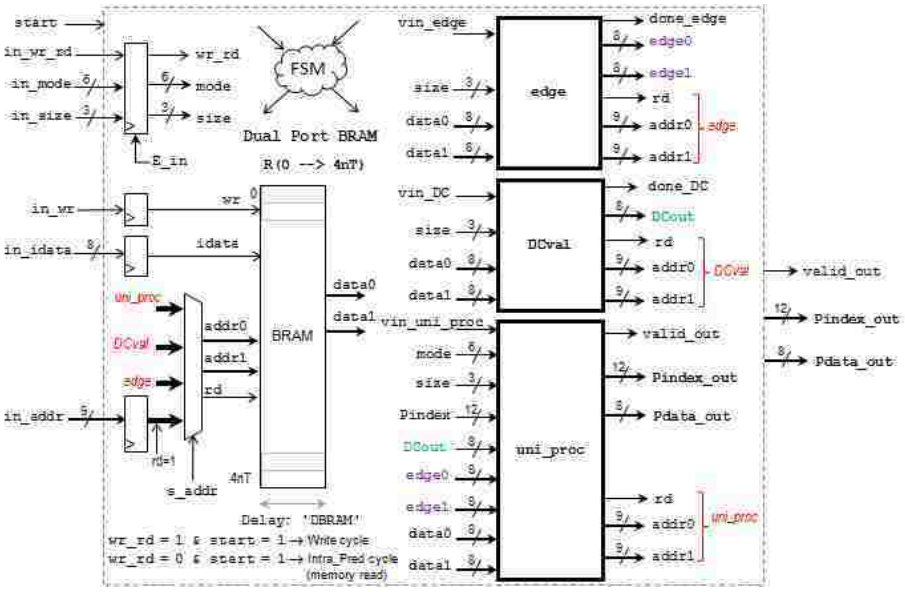


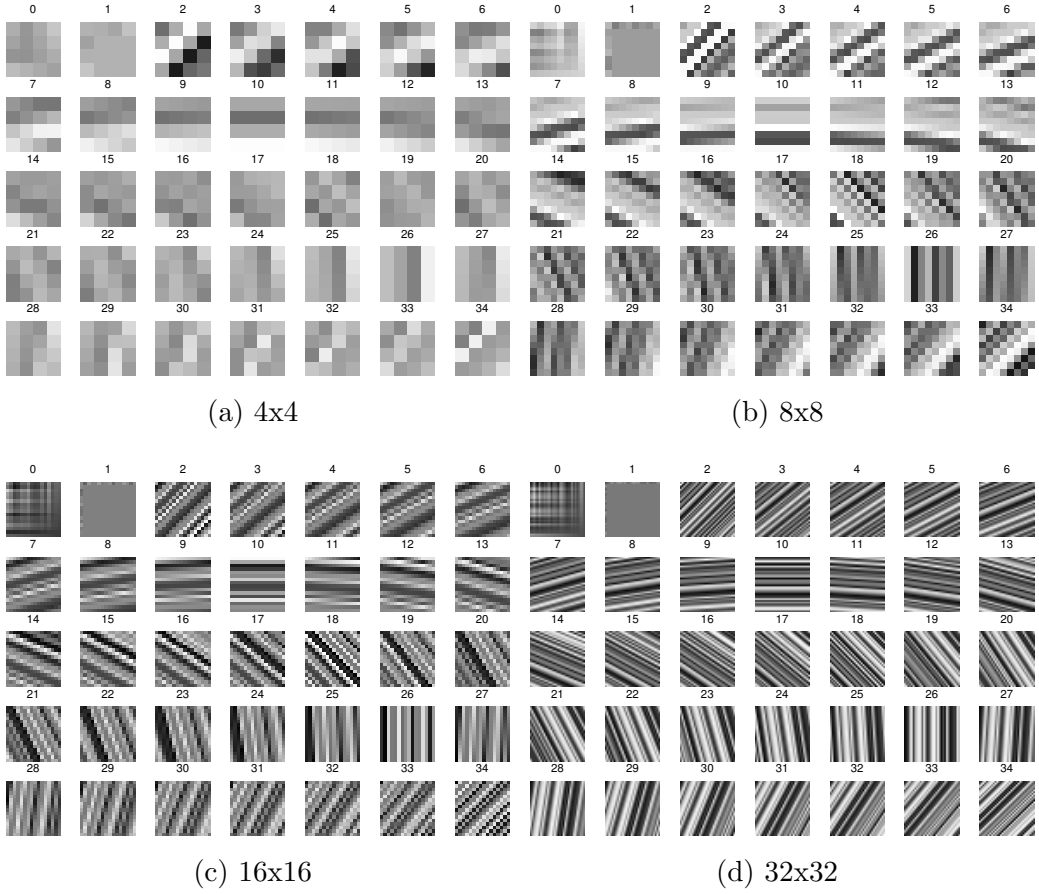
Figure 6.3: Integrated system integration using pipelined *uni_proc* circuit.

6.5 Results

6.5.1 Unified Reference Sample Indexing Verification

To verify the correctness of our unified indexing and accessing method, we generated the predicted blocks for each PU size $nT = 4, 8, 16, 32, 64$ for randomly generated reference samples. For each PU size and set of random reference samples, we confirm that the predicted Blocks were identical to the ones produced by the reference software.

We provide examples for PU sizes from 4×4 to 32×32 in Fig.6.4. From the results, we can see the fine directional selectivity of the HEVC standard.



(a) 4x4 (b) 8x8

(c) 16x16 (d) 32x32

Figure 6.4: 35 modes of intra prediciton for HEVC, for PU sizes from 4x4 to 32x32, with random generated reference samples for each PU size.

6.5.2 Synthesis Results

The proposed architecture was coded in VHDL and synthesized using Xilinx ISE 13.2 speed grade -3 for the $xc5vlx110t$ device. Synthesis results are shown in Table 6.2.

From the synthesis report, we have the maximum delay path is 4.887ns that gives a maximum operating frequency is 204.61Mhz. For the planar mode, we have a setup latency of *delay* cycles (defined in section 6.4.1) for the *edge* circuit. DC mode has setup latency of $2nT - 1 + delay$ cycles due to the *DCval* circuit. All other modes have no additional

Table 6.2: synthesis results on xc5vlx110t with speed grade -3 using Xilinx ISE 13.2

	uni_proc	*whole system*	FPGA
# of slice registers	562	684	69120
# of slice LUTs	467	690	69120
# of DSP48Es	7	7	64
# of BRAM	0	1	148

setup delays. After setup, the pipeline in *uni_proc* is filled in and after $8 + delay$ cycles, the pipelined is filled up and the first output pixel is computed. For the remaining pixels, we need an additional $nT \times nT - 1$ cycles since we output one pixel per cycle. For the encoder to determine the best mode, it generates all 35 modes prediction results. Overall, it takes $3delay + 7 + 2nT + 35nT^2$ cycles to generate all $35 \cdot nT^2$ output pixels. Thus, on average, the setup delay is dominated by the number of output pixels $35 \cdot nT^2$. Based on the encoded mode, the decoder circuit will only need $setup\_latency + 8 + delay + nT^2$ cycles to compute all nT^2 output pixels. For each mode and PU size, we summarize the operation cycles in Table 6.3. At an operating frequency of 100Mhz, the fully realized system is simulated after place and route, and gives a dynamic power cost of 50.48mW.

We also provide comparisons of our approach to previously published results. Compared to [10], we provide more modes, PU sizes and higher operating frequency (204Mhz vs 150Mhz). In terms of resource usage, we are using less resources in Virtex 5 than the Virtex 6 resources reported in [77] and [78].

Table 6.3: Total cycles to generate one prediction block and average cycles for one prediction pixel, on decoder side, delay=2.

	DC		Plannar		Angular	
	total	avg.	total	avg.	total	avg.
4x4	35	2.19	28	1.75	26	1.63
8x8	91	1.42	76	1.19	74	1.16
16x16	299	1.17	268	1.05	266	1.04
32x32	1099	1.07	1036	1.01	1034	1.01
64x64	4235	1.03	4108	1.00	4106	1.00

Chapter 7

DRASTIC for HEVC intra-encoding at the Frame Level

7.1 Abstract

We introduce the use of a dynamically reconfigurable architecture system for time-varying image constraints (DRASTIC) and consider applications in HEVC intra encoding. DRASTIC provides a framework for jointly optimizing complexity-rate-distortion for different operating modes. DRASTIC optimization involves dynamically reconfiguring parameters (e.g., the quantization parameter, encoding configuration modes) in software for achieving fine optimization control. The fine control achieved with the use of the DRASTIC modes is shown to perform significantly better than the standard use of fixed profiles.

7.2 Introduction

HEVC is improved video coding standard with tools such as recursive coding/transforms units, complex intra prediction modes, and asymetric inter prediction unit division, etc. It aims at 50% bit rate reduction for equal perceptual video quality [80]. However, the performance comes with unbearable computing complexity as more computing and comparision are executed. To reduce the inter encoding complexity, several configuration modes were shown as in [81]. For intra encoding complexity, rough mode set(RMS, [82]), gradient based intra prediction [14] and coding unit(CU) depth control [83] were proposed. Problem comes when complexity is treated as a separate performance factor, you can't achieve scalablility in complexity with same RD performance.

In this chapter, we introduce a Dynamically Reconfigurable Architecture System for Time-varying Image Constraints (DRASTIC) to describe a multi-objective optimization framework for video compression that provides scalable and adaptive solutions that jointly optimize computational-complexity, rate, and distortion. To formally define this framework, let E denote the energy expended in the computation, Q denote an image quality metric, R denote the rate measured in the number of bits per pixel, and $\mathcal{C}$ denote the control space used for optimization. We have four design modes for DRASTIC:

- **Minimum complexity mode:** Used for conserving energy without sacrificing image quality or requiring excessive bitrate. Here, based on $\texttt{Energy} = \texttt{Power} \cdot \texttt{Time}$, we minimize energy by minimizing time complexity. In this case, the family of optimal designs solve:

$$\min_{\mathcal{C}} T \quad \text{subject to } (Q \geq Q_{min}) \;\&\; (R \leq R_{max}). \tag{7.1}$$

- **Minimum rate mode:** Used to minimize bandwidth requirements as constrained by the network, or to support multiple users, without sacrificing image quality or requiring

excessive energy. In this case, we have:

$$\min_{\mathcal{C}} B \quad \text{subject to } (Q \geq Q_{min}) \;\&\; (T \leq T_{max}). \tag{7.2}$$

- **Maximum video quality mode:** Used for providing the maximum possible image quality subject to limited bandwidth and available energy:

$$\max_{\mathcal{C}} Q \quad \text{subject to } (T \leq T_{max}) \;\&\; (R \leq R_{max}) \tag{7.3}$$

- **Typical mode:** Used for balancing all objective requirements subject to limited bandwidth, energy, and minimum image quality levels:

$$\begin{aligned} &\min_{\mathcal{C}} \quad -\alpha \cdot Q + \beta \cdot T + \gamma \cdot R \\ &\text{subject to } (Q \geq Q_{max}) \;\&\; (T \leq T_{max}) \;\&\; (R \leq R_{min}). \end{aligned} \tag{7.4}$$

The contributions of this paper include the introduction of a control space and controller that provide scalable solutions in terms of encoding complexity, bitrate and quality for HEVC intra encoding, and the realization of DRASTIC modes, with dynamic reconfiguration at the frame level. The current paper is focused on implementing DRASTIC based on scalable block compression for intra-encoding. This work complements earlier work on the use of DRASTIC modes for the DCT presented in [84].

The paper is organized as follows. In section II, we introduce the configuration space $\mathcal{C}$. The implementation of DRASTIC controller and modes is detailed in section III. Concluding remarks are given in section IV.

7.3 The configuration space: scalable block-compress for HEVC intra-encoding

The basic framework for implementing DRASTIC HEVC intra-encoding is shown in Fig. 7.1. Following video encoding for each video frame, the DRASTIC controller is provided

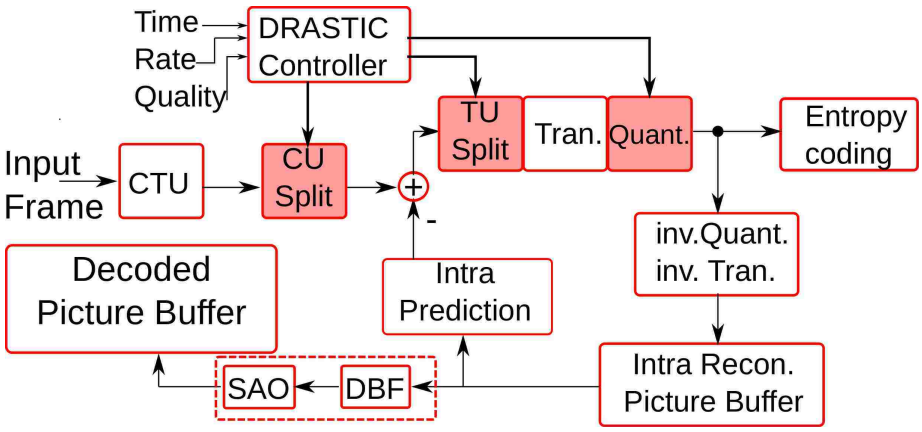


Figure 7.1: Diagram for DRASTIC HEVC Intra Encoding System

with: (i) Time taken to compress and encode of last frame. (ii) Rate achieved as bits per sample for the encoding of last frame. (iii) Quality based on the PSNR between the original and the encoded last frame. The goal of the DRASTIC controller is to use these objectives to adapt HEVC's intra-encoding configuration-performance model in order to provide optimal parameters for the different DRASTIC modes. To make this possible, it is necessary to define the configuration space $\mathcal{C}$ based on a scalable block-compress approach. This section details the parameters associated with $\mathcal{C}$. We begin with a summary of HEVC intra-prediction. The HEVC intra-encoding implementation is based on the reference software HM11.0 given in [85]. For luma prediction, the modes include the use of a rough mode set (RMS [82, 86], 8 modes for prediction units (PU) of sizes 4×4 and 8×8, and 3 modes for PU sizes of 16×16,32×32 and 64×64). The determination of the optimal luma mode is based on a two-step approach. First, the rough mode set is determined as a set of possible candidate modes that are assumed to include the optimal prediction mode. Fast computation of the rough mode set is based on the estimation of distortion using the sum of absolute Hadmard transform coefficients and the rate estimated based on the considered mode. Second, using the RMS, the optimal luma prediction mode is determined by estimating distortion using the mean-squared error in the reconstruction and rate estimated based on the use of the largest

Transform Unit (TU). For chroma prediction, the optimal prediction mode is selected from 5 available modes [85] according to their RD performance, as was done in the second step for the luma mode. Given the estimated, optimal luma and chroma CU modes, an exhaustive subdivision process is applied based on RD performance to determine the TU split (see Fig. 7.1). After TU spliting is applied, the resulting, quantized coefficients are entropy encoded. The reconstructed, intra-encoded part is further used to encode next CTU. To reduce reconstruction artifacts, a deblocking filter (DBF) and sample adaptive offset (SAO) is applied before the intra picture is put into decoded picture buffer. For the current paper, DBF and SAO are not used to enable fast intra-prediction performance. Here, note that DBF and SAO should be considered for low bitrates where compression artifacts can be significant. The DRASTIC controller is used for controlling how the coding tree units (CTU) is split into the coding units(CU) and how the residuals were splited into the transform units (TU), by replacing the above mentioned CU and TU subdivision process.

Recent bitrate, image quality (based on PSNR), and encoding time are used by the DRASTIC controller to determine future control parameters for jointly optimizing rate-distortion-energy performance. Here, note that jointly-optimal requires Pareto-optimality. To provide Pareto-optimality, we consider an approach based on the quantization parameter (QP varies from 0 to 51) and a custom configuration mode that determines the allowed CU and TU sizes used in determining the optimal prediction mode. The custom configuration mode is given by:

$$\texttt{config} = 5 * \texttt{DepthConfig} + \texttt{FinerDepthConfig},$$

where $\texttt{DepthConfig} = 0, \ldots, 3$, is used for selecting the possible sizes for CU and TU as described in Table 7.1, and $\texttt{FinerDepthConfig} = 0, \ldots, 4$, provides finer control over the allowed CU and TU sizes as described in Table 7.2. The basic idea for `DepthConfig` is to use nested configurations where larger configuration modes include all the options available in lower modes, and thus are expected to provide reasonable configuration scalability grain. For `FinerDepthConfig`, the nested configurations control the percentage of CTUs that is

required to use the minimum CU and TU sizes.

Table 7.1: Depth control for CU and TU candidate sizes based on `DepthConfig`.

`DepthConfig`	Allowed Luma CU size	Allowed Luma TU size
0	64,32	32
1	64,32,16	32,16
2	64,32,16,8	32,16,8
3	64,32,16,8,4	32,16,8,4

Table 7.2: Finer depth control for the CU, TU sizes using `FinerDepthConfig`.

`FinerDepthConfig`	Allowed Luma CU size	Allowed Luma TU size
0	20% minimum size	20% minium size
1	40% minimum size	40% minimum size
2	60% minimum size	60% minimum size
3	80% minimum size	80% minimum size
4	100% minimum size	100% minimum size

7.3.1 Rate-distortion-energy space results

We will next provide results on the estimation of the rate-distortion-energy space and demonstrate that our approach leads to Pareto-optimal results. The results are demonstrated on the RaceHorses video [87] in Figures 7.2, 7.3 and 7.4. To generate the space, `QP` was varied in the range of [0, 51) with a step of 3 and `config` takes on all of the values in $0, 1, 2, \ldots, 19$. Furthermore, to estimate the space, we take the average of 6 video frames. The approach generates 340 Pareto-optimal points where higher values of `config` generate better RD performance at higher complexity (or energy consumption).

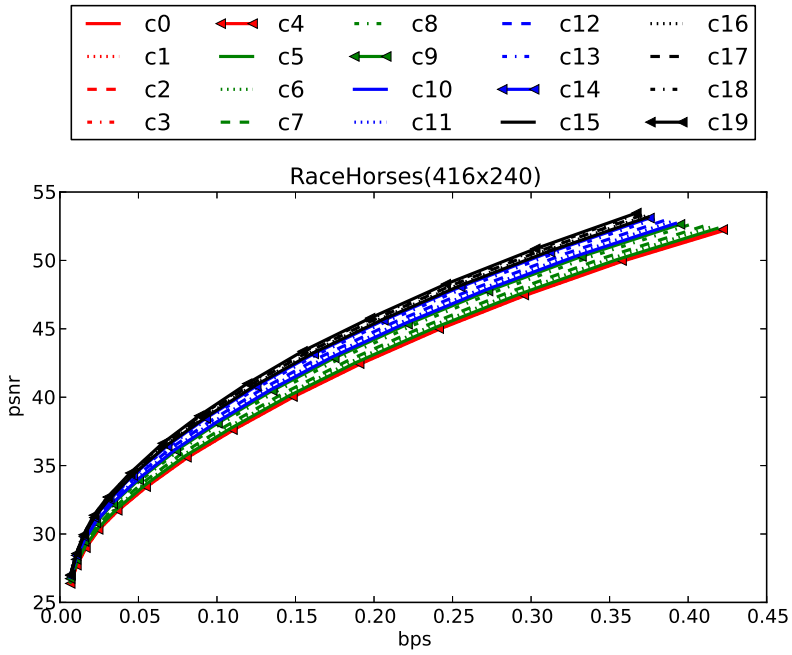


Figure 7.2: The projection of the rate-distortion-energy space on the rate-distortion space for the RaceHorses video. Here, bps refers to the number of bits-per-sample. Note that all of the configurations are Pareto-optimal in the sense that it takes more energy (time) to provide better rate-distortion performance. The video frame is of size 432×240.

7.4 DRASTIC Control

7.4.1 Simulation Setup

To simulate time-varying constraints, we consider the transition from a `low` to a `medium` to a `high` constraint profile. Transitions occur every 40 frames. The `low` constraint profile uses lower values for PSNR, bitrate, and complexity. Then, the `medium` and `high` constraint profiles are used to define the four DRASTIC modes described in the introduction. The time-varying constraint profile evolution is shown in Fig. 7.5 plotted using red lines. For the typical mode, all of the constraints are active. For all other DRASTIC modes, one of the constraints, chosen from the PSNR, BPS, SPS is allowed to vary so as to: (i) maximize

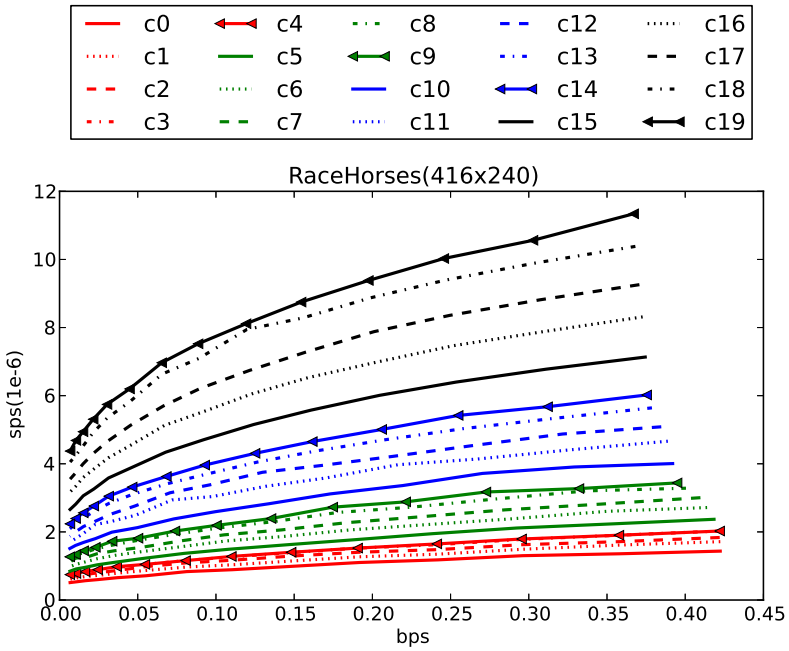


Figure 7.3: The projection of the rate-distortion-energy space on the rate-complexity space where complexity is measured in terms of the number of seconds per sample (sps) and it is assumed to be proportional to the consumed energy (from $E = Pt$). The space is Pareto-optimal in the 3-dimensional space in that longer computation times increase the PSNR.

quality, (ii) minimize bitrate, or (iii) minimize computational complexity as detailed in the introduction.

To provide baseline comparisons, we will compare the fine adaptation of DRASTIC with a set of three fixed profiles. Furthermore, the basic profiles will be used for initializing the search for the local, Pareto-optimal solution. The three fixed configuration profiles are given by: (i) *low*-profile uses `Config=4, QP=42` for the lowest-quality, minimum bitrate, and minimum complexity, (ii) *medium*-profile uses `Config=11, QP=28`, and (iii) *high*-profile uses `Config=18, QP=14` for the highest-quality, maximum bitrate, and maximum complexity. The corresponding constraints that can be achieved are estimated using the average quality, bitrate, and time complexity based on the original configuration (`Config`, `QP`)

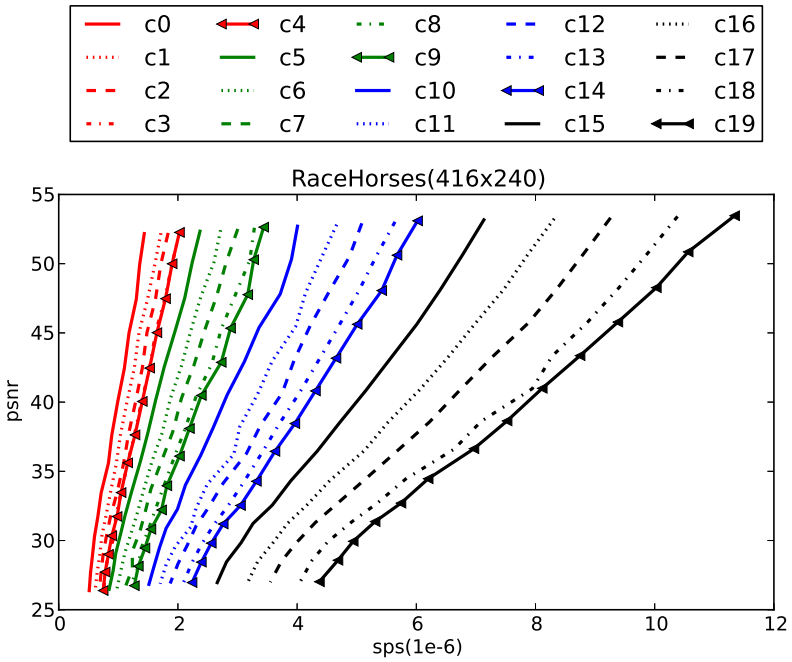


Figure 7.4: The projection of the rate-distortion-energy space on the distortion-complexity space where complexity is assumed to be proportional to the consumed energy (from $E = Pt$). The space is Pareto-optimal in the 3-dimensional space.

and the three neighboring configurations given by: (`Config` + 2, `QP`), (`Config`, `QP` − 2), and (`Config` + 2, `QP` − 4).

7.4.2 Initialize and Hold Control

A simple solution for controling the encoding system to meet initialized constraints is just to initialize and hold. We change the profile constraints every 40 frames. The initialization and hold policy is shown in fig.7.5, this method will create a smooth performance sequence, we know that the performance will not change too much as long as there is no scene change. Problem with initialization and hold control policy is that the control space is not used to achieive optimized performance, also it can't adapt to image contents, scene change will lead

to performance ripple.

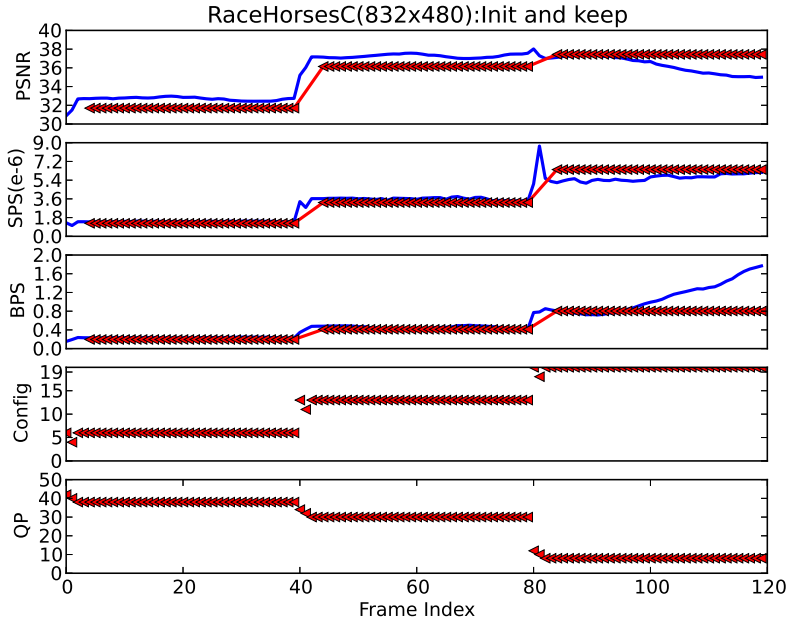


Figure 7.5: Results based on *initialize and hold*.

7.4.3 Prediction Model and Model Update

We model the local Pareto front using a linear model given by:

$$\begin{aligned} Q &= a_1 \cdot \texttt{QP} + b_1 \cdot \texttt{Config} + c1 \\ T &= a_2 \cdot \texttt{QP} + b_2 \cdot \texttt{Config} + c2 \\ R &= a_3/2^{(\texttt{QP}-4)/6} + b_3 \cdot \texttt{Config} + c3 \end{aligned} \tag{7.5}$$

where Q is measured by PSNR, T denotes the time required for processing a single pixel, and R denotes the number of bits per sample. All of the coefficients in (7.5) can be estimated by using the last three independent measurements. For example, to estimate a_1, b_1, c_1 using

a set of three measurements, we would simply use

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} \texttt{QP}_1 & \texttt{Config}_1 & 1 \\ \texttt{QP}_2 & \texttt{Config}_2 & 1 \\ \texttt{QP}_3 & \texttt{Config}_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix}. \tag{7.6}$$

Naturally, matrix inversion requires three independent measurements. The search-space for moving from the current configuration (given by (Config, QP)) is restricted in the neighborhood defined by $[\texttt{Config} - 2, \texttt{Config} + 2] \times [\texttt{QP} - 5, \texttt{QP} + 5]$. We expect that the selected configuration ($\texttt{Config}_\texttt{n}$, $\texttt{QP}_\texttt{n}$) will generate ($\texttt{Q}_\texttt{n}$, $\texttt{T}_\texttt{n}$, $\texttt{R}_\texttt{n}$) accurately, based on the local linear model. If the linear model provides predictions that miss the measured time-rate-PSNR by more than 5%, (Config, QP) are adjusted by adding random offsets chosen from $[-1, +1]$ for both of them.

7.5 DRASTIC Implementation And Results

7.5.1 Minimum Complexity Mode

For each constraint-profile, the minimum complexity mode starts with the corresponding profile (low, medium, or high) and uses the controller to determine the next configuration. The basic idea is to determine the configuration that provides the minimum value of T within the search space. In the case where the constraints cannot be satisfied, the basic approach is to solve the unconstrained problem given by

$$\begin{aligned} \min_{\texttt{Config}, \texttt{QP}} \quad & \alpha \cdot \frac{T_n}{T_{max}} + \beta \cdot \text{abs}\left(\frac{Q_n - Q_{min}}{Q_{min}}\right) \\ & + \gamma \cdot \text{abs}\left(\frac{R_n - R_{max}}{R_{max}}\right) \end{aligned} \tag{7.7}$$

with $\alpha = 1$, $\beta = 20$ and $\gamma = 20$. As shown in Fig. 7.6, compared to the initialize and hold mode, the minimum complexity mode requires far less times T, especially for the high-profile.

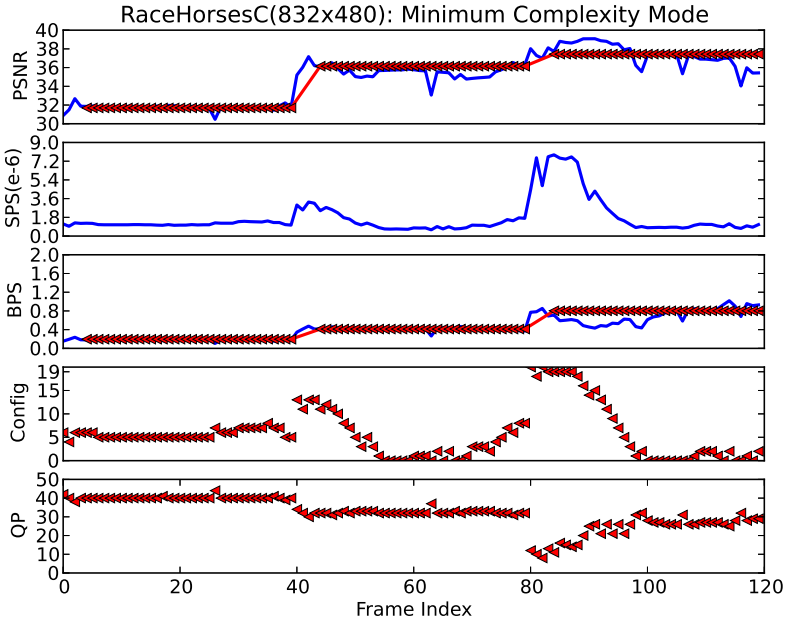


Figure 7.6: Results from *minimum complexity mode*.

7.5.2 Minimum Rate Mode

The minimum rate mode looks for the configuration pair that minimizes the bitrate. In the case that the constraints cannot be satisfied, the configuration that solves the unconstrained problem given by

$$\min_{\texttt{Config, QP}} \quad \alpha \cdot \text{abs}\left(\frac{T_n - T_{max}}{T_{max}}\right) + \beta \cdot \text{abs}\left(\frac{Q_n - Q_{min}}{Q_{min}}\right) + \gamma \cdot \frac{R_n}{R_{max}} \tag{7.8}$$

with $\alpha = 20$, $\beta = 20$ and $\gamma = 1$, is chosen. As shown in Fig. 7.7, compared to the initialize and hold mode, the minimum rate mode tends to require lower nitrates while satisfying the quality and complexity constraints.

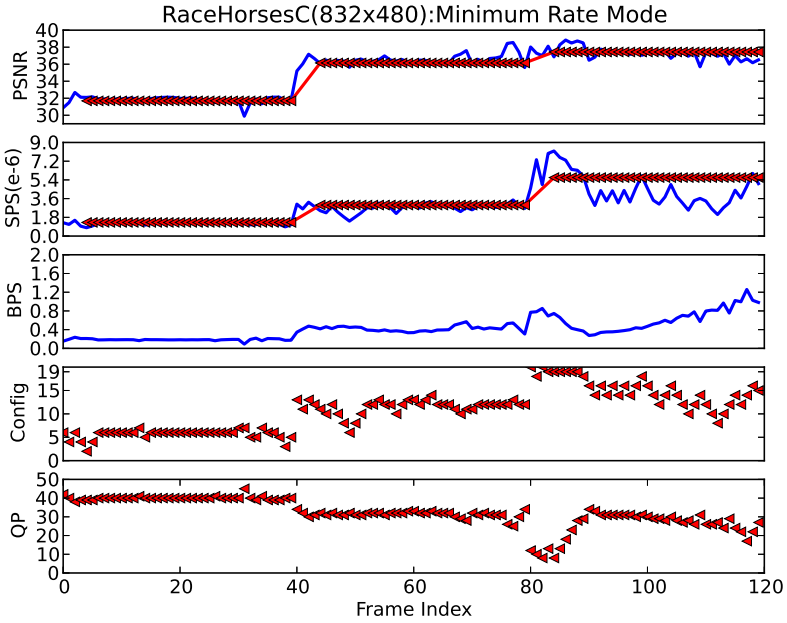


Figure 7.7: Results for *minimum rate mode*.

7.5.3 Maximum Quality Mode

Similarly, for the maximum quality mode, in the case that the constraints are not satisfied, the configuration that solves

$$\min_{\texttt{Config, QP}} \quad \alpha \cdot \text{abs}\left(\frac{T_n - T_{max}}{T_{max}}\right) + \beta \cdot \frac{Q_n}{Q_{min}} + \gamma \cdot \text{abs}\left(\frac{R_n - R_{max}}{R_{max}}\right) \tag{7.9}$$

with $\alpha = 20$, $\beta = -1$ and $\gamma = 20$ is chosen instead. As demonstrated in Fig. 7.8, the maximum quality mode maintains higher quality than the initialize and hold mode.

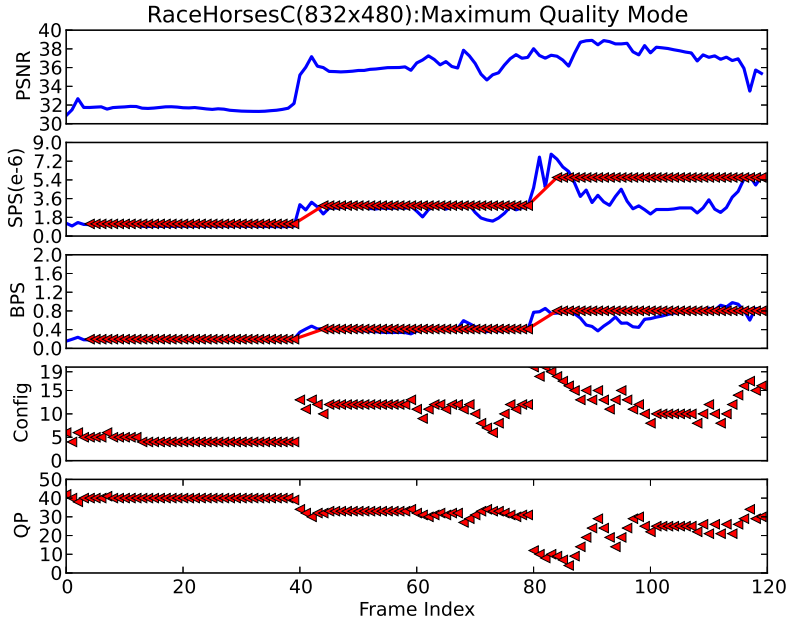


Figure 7.8: Results for *maximum quality mode*.

7.5.4 Typical Mode

For the typical mode, failure to meet the constraints leads to problem of finding the configuration that solves

$$\min_{\texttt{Config, QP}} \quad \alpha \cdot \text{abs}\left(\frac{T_n - T_{max}}{T_{max}}\right) + \beta \cdot \text{abs}\left(\frac{Q_n - Q_{min}}{Q_{min}}\right) + \gamma \cdot \text{abs}\left(\frac{R_n - R_{max}}{R_{max}}\right) \tag{7.10}$$

with $\alpha = 20$, $\beta = 20$, and $\gamma = 20$. From the results in Fig. 7.9, compared to initialize and hold, the typical mode provides a more consistent performance.

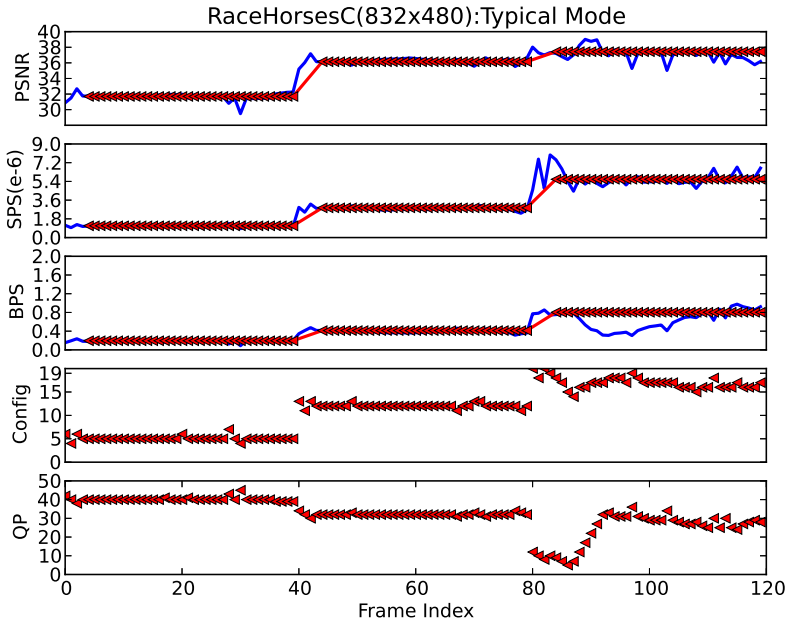


Figure 7.9: Results for *typical mode*.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

A framework for achieving optimal image and video compression performance in multiple objectives has been developed in the dissertation. The developed framework provides optimal configurations that can meet time-varying constraints. The framework jointly considers optimization over bitrate, reconstruction quality, and dynamic power / energy (or computational complexity). In terms of optimization modes, effective approaches have been developed for: 1) minimum dynamic power / energy (or computational complexity), 2) minimum rate, 3) maximum image quality, a 4) typical mode.

Instead of traditional RD optimization, the dissertation introduced a rate-quality optimization approach that used stochastic optimization to compute optimal quantization tables. This first approach was based on multi-pass compression that is possible for offline compression. For real-time video communication, multi-pass compression will introduce unacceptable delays. Computational complexity is considered as a measurement that is directly related to energy consumption for real-time implementations. Thus, DRASTIC is considered for both software-hardware as well as software only configurations. The dissertation considered appli-

cations in MJPEG, H.264/AVC, and H.265/HEVC by providing DRASTIC implementations of the DCT, Deblocking filtering and intra-prediction respectively. Joint software-hardware optimization uses hardware reconfiguration to control dynamic power while software-based quantization control allows for an effective method for controlling bitrate (which is also affected by the hardware configuration). For HEVC, intra-encoding modes are considered as well the use of different CU depths.

For hardware dynamic partial reconfiguration, DRASTIC requires scalabiltiy to reduce the reconfiguration overhead. A scalable reconfiguration controller has been developed and implemented for MJPEG. For software-only configurations, DRASTIC uses more precise feedback by modeling the encoding process and updating model parameters after each encoding unit to ensure precise estimation of encoding performance.

Currently, there are two journal papers that are directly associated with the dissertation. First, the DRASTIC DCT implementation will be submitted as a full journal paper to the *IEEE Transactions on Circuits and Systems for Video Technology*. Second, the HEVC intra RD optimization based on the DRASTIC framework is expected to be completed and submitted for publication before the end of the semester.

8.2 Future Work

In what follows, I provide a set of recommendations for future work:

- Hardware implementations need to consider power issues when significant computations involve heterogeneous environments that include both CPUs and reconfigurable hardware components. The CPU will have less work to do and video processing can be performed in effective hardware modules. DRASTIC can effective control dynamic power allocation.
- DRASTIC needs to be extended to allow for finer control. The dissertation research

was focused on dynamic reconfiguration at the frame level. Reconfiguration between CTU or Macroblocks will lead to finer performance control which will in turn yield to smooth transitions in image quality, bitrate, and computational complexity levels. Smoother transitions are needed for minimizing the possibility that humans will detect decoded video artifacts due to configuration mode changes. Furthermore, finer control modes will provide for better DRASTIC reconfiguration control.

- DRASTIC needs to be extended to cover inter-mode coding. Inter-mode extensions should consider the use of different video frame formats (e.g., P and B video frame types). By considering different video frame types, inter-mode can yield significant bitrate savings over intra-mode encoding.

- The design of effective configuration spaces for different video standards remains a challenge. It will be interesting to investigate the use of several different DRASTIC configuration spaces for different video coding standards. This research would first analyze several different coding tools independently and then jointly to determine the most effective configuration spaces.

- DRASTIC control requires accurate models for estimating bitrate, quality, and complexity based on the selected configuration. More accurate models that use the configuration parameters to predict bitrate, quality, and complexity need to be considered in future research. This research should also consider the development of better mechanisms for updating the models.

Appendix

This section provides a list of published and to be published work from the author.

[1] Murray, V., Llamocca, D., Jiang, Y., Lyke, J., Pattichis, M., Achramowicz, S. , and Avery, K., "Adaptive Wiring Panels using Cell-based Architectures: A First Approach", *Reconfigurable Systems Workshop*, Albuquerque, Nov, 2010.

[2] Murray Victor, Llamocca Daniel, Yuebing Jiang, Lyke J., Pattichis, Marios S., Achramowicz S., and Avery K., "A Cell-based Architecture for Adaptive Wiring Panels: A First Approach," *2011 Reinventing Space Conference*, Los Angeles, May, 2011.

[3] Yuebing Jiang and Pattichis, M.S., "JPEG image compression using quantization table optimization based on perceptual image quality assessment," *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp.225-229, 6-9 Nov. 2011.

[4] Yuebing Jiang and Pattichis, M., "A dynamically reconfigurable DCT architecture for maximum image quality subject to dynamic power and bitrate constraints," *2012 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pp.189-192, 22-24 April 2012.

[5] Yuebing Jiang and Pattichis, M., "Dynamically reconfigurable DCT architecture based on Bitrate, Power, and Image Quality Considerations," *2012 IEEE International Conference*

on Image Processing (ICIP), 3-6 Oct. 2012

[6] Yuebing Jiang and Marios S. Pattichis, "A Dynamically Reconfigurable Deblock- ing Filter for H.264/AVC Codecs," *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, Nov., 2012.

[7] Yuebing Jiang and Marios S. Pattichis, "A Configurable System For Role-specific Video Imaging During Laparoscopic Surgery," *International Conference On BioInformatics and Bioengineering*, Larnaca, Cyprus, Nov., 2012.

[8] Murray Victor, Llamocca Daniel, Lyke J., Avery K., Yuebing Jiang and Marios S. Pattichis, "A Cell-based Architecture for Adaptive Wiring Panels: A First Prototype," accepted for publication, *AIAA Journal of Aerospace Computing, Information and Communication.*

[9] Llamocca, D., Murray, V., Jiang, Y., Pattichis, M., Lyke, J., and Avery, K., "A Scalable, Open-Source Architecture for Real-Time Monitoring of Adaptive Wiring Panels", accepted for publication in *AIAA Journal of Aerospace Information Systems.*

[10] Yuebing Jiang, Gangadharan Esakki, Marios Pattichis, "Dynamically Reconfigurable Architecture System for Time-varying Image Constraints (DRASTIC) for HEVC Intra Encoding". *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, Nov., 2013.

[11] Yuebing Jiang, Daniel Llamocca, Marios Pattichis, and Gangadharan Esakki , A Unified and Pipelined Hardware Architecture for Implementing Intra Prediction in HEVC, accepted for publication in *IEEE Southwest Symposium on Image Analysis and Interpretation*

References

[1] *Draft ITU-T Recommendation and Final Draft International Standard of JointVideo Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC*, Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG Std., May 2003.

[2] W.-J. H. Gary J. Sullivan, Jens-Rainer Ohm and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE TRANS. ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, p. to be published, 2012.

[3] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhring, M. Winken, and T. Wiegand, "Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 12, pp. 1676 –1687, dec. 2010.

[4] *Virtex Series Configuration Architecture user guide*, Xilinx, San Jose,CA, October 20, 2004.

[5] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 13, no. 7, pp. 688 – 703, july 2003.

[6] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 15, no. 5, pp. 645 – 658, may 2005.

[7] Z. He, W. Cheng, and X. Chen, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 18, no. 5, pp. 596 –608, may 2008.

[8] J. Kim, J. Yang, H. Lee, and B. Jeon, "Fast intra mode decision of hevc based on hierarchical structure," in *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, dec. 2011, pp. 1 –4.

[9] R. Vanam, J. Chon, E. A. Riskin, R. E. Ladner, F. M. Ciaramello, and S. S. Hemami, "Rate-distortion-complexity optimization of an h.264/avc encoder for real-time video-conferencing on a mobile device," 5th International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM 2010), January 2010.

[10] P. Li, Y. Chen, and W. Ji, "Rate-distortion-complexity analysis on avs encoder," in *Advances in Multimedia Information Processing - PCM 2010*, ser. Lecture Notes in Computer Science, G. Qiu, K. Lam, H. Kiya, X.-Y. Xue, C.-C. Kuo, and M. Lew, Eds. Springer Berlin / Heidelberg, 2011, vol. 6298, pp. 73–83.

[11] X. Li, M. Wien, and J.-R. Ohm, "Rate-complexity-distortion optimization for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 21, no. 7, pp. 957 –970, july 2011.

[12] D. Llamocca and M. , "A dynamically reconfigurable pixel processor system based on power/energy-performance-accuracy optimization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[13] Y. Jiang and M. Pattichis, "Jpeg image compression using quantization table optimization based on perceptual image quality assessment," in *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, nov. 2011, pp. 225 –229.

[14] ——, "A dynamically reconfigurable dct architecture for maximum image quality subject to dynamic power and bitrate constraints," in *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, april 2012, pp. 189 –192.

[15] ——, "Dynamically reconfigurable dct architecture based on bitrate, power, and image quality considerations," in *2012 International Conference on Image Processing*, Oct 2012, p. to be published.

[16] S. Channappayya, A. Bovik, C. Caramanis, and R. Heath, "Design of linear equalizers optimized for the structural similarity index," *IEEE Transactions on Image Processing,*, vol. 17, no. 6, pp. 857 –872, 2008.

[17] S. Channappayya, A. Bovik, and R. Heath, "Rate bounds on ssim index of quantized images," *IEEE Transactions on Image Processing,*, vol. 17, no. 9, pp. 1624 –1639, 2008.

[18] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing,*, vol. 13, no. 4, pp. 600 –612, 2004.

[19] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[20] S. Channappayya, A. Bovik, C. Caramanis, and R. Heath, "Ssim-optimal linear image restoration," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008.*, 312008-april4 2008, pp. 765 –768.

[21] S. M.Ross, *Simulation(Third Edition).* Academic Press, 2002.

[22] L. C. H.R. Sheikh, Z.Wang and A. Bovik. Live image quality assessment database release 2. [Online]. Available: http://live.ece.utexas.edu/research/quality

[23] H. Sheikh, M. Sabir, and A. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *Image Processing, IEEE Transactions on*, vol. 15, no. 11, pp. 3440 –3451, nov. 2006.

[24] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis, fast algorithm, and vlsi architecture design for h.264/avc intra frame coder," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 3, pp. 378 – 401, march 2005.

[25] L. Chen, N. Shashidhar, and Q. Liu, "Scalable secure mjpeg video streaming," in *2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2012, pp. 111–115.

[26] Q. Du, H. Qin, J. Tang, and X. Li, "Design of the arm based remote surveillance system," in *2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM)*, vol. 1, 2012, pp. 336–338.

[27] H.-Y. Ko, J.-H. Lee, and J.-O. Kim, "Implementation and evaluation of fast mobile vnc systems," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 4, pp. 1211–1218, 2012.

[28] D. Llamocca and M. Pattichis, "A dynamically reconfigurable pixel processor system based on power/energy-performance-accuracy optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 3, pp. 488–502, 2013.

[29] J. Huang, M. Parris, J. Lee, and R. F. Demara, "Scalable fpga-based architecture for dct computation using dynamic partial reconfiguration," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 1, pp. 1–18, 2009.

[30] J. Huang and J. Lee, "A self-reconfigurable platform for scalable dct computation using compressed partial bitstreams and blockram prefetching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 11, pp. 1623 –1632, 2009.

[31] ——, "Reconfigurable architecture for zqdct using computational complexity prediction and bitstream relocation," *Embedded Systems Letters, IEEE*, vol. 3, no. 1, pp. 1 –4, march 2011.

[32] C. Kannangara, I. Richardson, and A. J. Miller, "Computational complexity management of a real-time h.264/avc encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 9, pp. 1191–1200, 2008.

[33] A. Madisetti and J. Willson, A.N., "A 100 mhz 2-d 8x8 dct/idct processor for hdtv applications," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 5, no. 2, pp. 158 –165, apr 1995.

[34] Y.-P. Lee, T.-H. Chen, L.-G. Chen, M.-J. Chen, and C.-W. Ku, "A cost-effective architecture for 8x8 two-dimensional dct/idct using direct method," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 7, no. 3, pp. 459 –467, jun 1997.

[35] S.-F. Hsiao, W.-R. Shiue, and J.-M. Tseng, "A cost-efficient and fully-pipelinable architecture for dct/idct," *Consumer Electronics, IEEE Transactions on*, vol. 45, no. 3, pp. 515 –525, aug 1999.

[36] K.-H. Cheng, C.-S. Huang, and C.-P. Lin, "The design and implementation of dct/idct chip with novel architecture," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4, 2000, pp. 741 –744 vol.4.

[37] S.-F. Hsiao, W.-R. Shiue, and J.-M. Tseng, "Design and implementation of a novel linear-array dct/idct processor with complexity of order log2n," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 147, no. 5, pp. 400 –408, oct 2000.

[38] L. Agostini, I. Silva, and S. Bampi, "Pipelined fast 2d dct architecture for jpeg image compression," in *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, 2001, pp. 226 –231.

[39] E. Kusuma and T. Widodo, "Fpga implementation of pipelined 2d-dct and quantization architecture for jpeg image compression," in *Information Technology (ITSim), 2010 International Symposium in*, vol. 1, june 2010, pp. 1 –6.

[40] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *Communications, IEEE Transactions on*, vol. 25, no. 9, pp. 1004 – 1009, sep 1977.

[41] T. Xanthopoulos and A. Chandrakasan, "A low-power dct core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 5, pp. 740 –750, may 2000.

[42] D. W. Kim, T. W. Kwon, J. M. Seo, J. K. Yu, S. K. Lee, J. H. Suk, and J. R. Choi, "A compatible dct/idct architecture using hardwired distributed arithmetic," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol. 2, may 2001, pp. 457 –460 vol. 2.

[43] S. Yu and J. Swartziander, E.E., "Dct implementation with distributed arithmetic," *Computers, IEEE Transactions on*, vol. 50, no. 9, pp. 985 –991, sep 2001.

[44] P. Chungan, C. Xixin, Y. Dunshan, and Z. Xing, "A 250mhz optimized distributed architecture of 2d 8x8 dct," in *ASIC, 2007. ASICON '07. 7th International Conference on*, oct. 2007, pp. 189 –192.

[45] H. Lim, C. Yim, and J. Swartzlander, E.E., "Finite word-length effects of an unified systolic array for 2-d dct/idct," in *Application Specific Systems, Architectures and Processors, 1996. ASAP 96. Proceedings of International Conference on*, aug 1996, pp. 35 –44.

[46] D. Chiper, M. Swamy, M. Ahmad, and T. Stouraitis, "Systolic algorithms and a memory-based design approach for a unified architecture for the computation of dct/dst/idct/idst," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 6, pp. 1125 – 1137, june 2005.

[47] P. Meher, "Systolic designs for dct using a low-complexity concurrent convolutional formulation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 9, pp. 1041 –1050, sept. 2006.

[48] Y. H. Hu and Z. Wu, "An efficient cordic array structure for the implementation of discrete cosine transform," *Signal Processing, IEEE Transactions on*, vol. 43, no. 1, pp. 331 –336, jan 1995.

[49] S. Yu and J. Swartzlander, E.E., "A scaled dct architecture with the cordic algorithm," *Signal Processing, IEEE Transactions on*, vol. 50, no. 1, pp. 160 –167, jan 2002.

[50] J.-I. Guo, R.-C. Ju, and J.-W. Chen, "An efficient 2-d dct/idct core design using cyclic convolution and adder-based realization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 4, pp. 416 – 428, april 2004.

[51] A. Madanayake, R. Cintra, D. Onen, V. Dimitrov, N. Rajapaksha, L. Bruton, and A. Edirisuriya, "A row-parallel 8 8 2-d dct architecture using algebraic integer-based exact computation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 6, pp. 915 –929, june 2012.

[52] K. Seshadrinathan and A. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *IEEE Transactions on Image Processing,*, vol. 19, no. 2, pp. 335–350, 2010.

[53] K. Seshadrinathan, R. Soundararajan, A. Bovik, and L. Cormack, "Study of subjective and objective quality assessment of video," *IEEE Transactions on Image Processing,*, vol. 19, no. 6, pp. 1427–1441, 2010.

[54] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack, "A subjective study to evaluate video quality assessment algorithms," in *SPIE Proceedings Human Vision and Electronic Imaging,*, Jan. 2010.

[55] Y.-F. Ou, Z. Ma, and Y. Wang, "Modeling the impact of frame rate and quantization stepsizes and their temporal variations on perceptual video quality: A review of recent works," in *2010 44th Annual Conference on Information Sciences and Systems (CISS),*, 2010, pp. 1–6.

[56] C.-H. Hsieh, "A zonal jpeg," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2, april 2005, pp. 756 – 757 Vol. 2.

[57] J. Park, J. H. Choi, and K. Roy, "Dynamic bit-width adaptation in dct: An approach to trade off image quality and computation energy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,*, vol. 18, no. 5, pp. 787 –793, may 2010.

[58] P. Seeling and M. Reisslein, "Video transport evaluation with h.264 video traces," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1142–1165. [Online]. Available: http://trace.eas.asu.edu/yuv/

[59] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, "A pipelined fast 2d-dct accelerator for fpga-based socs," in *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, march 2007, pp. 331 –336.

[60] V. Sharma, K. Mahapatra, and U. Pati, "An efficient distributed arithmetic based vlsi architecture for dct," in *Devices and Communications (ICDeCom), 2011 International Conference on*, feb. 2011, pp. 1 –5.

[61] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 614 –619, july 2003.

[62] M. Sima, Y. Zhou, and W. Zhang, "An efficient architecture for adaptive deblocking filter of h.264/avc video coding," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 1, pp. 292 – 296, feb 2004.

[63] S. Warrington, H. Shojania, S. Sudharsanan, and W.-Y. Chan, "Performance improvement of the h.264/avc deblocking filter using simd instructions," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, may 2006, pp. 4 pp. –2700.

[64] C. Arbelo, A. Kanstein, S. Lopez, J. Lopez, M. Berekovic, R. Sarmiento, and J.-Y. Mignolet, "Mapping control-intensive video kernels onto a coarse-grain reconfigurable architecture: the h.264/avc deblocking filter," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, april 2007, pp. 1 –6.

[65] M. Parlak and I. Hamzaoglu, "Low power h.264 deblocking filter hardware implementations," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 2, pp. 808 –816, may 2008.

[66] R. Khraisha and J. Lee, "A scalable h.264/avc deblocking filter architecture using dynamic partial reconfiguration," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, march 2010, pp. 1566 –1569.

[67] ——, "A bit-rate aware scalable h.264/avc deblocking filter using dynamic partial reconfiguration," *Journal of Signal Processing Systems*, pp. 1–10, 2011, 10.1007/s11265-011-0584-z. [Online]. Available: http://dx.doi.org/10.1007/s11265-011-0584-z

[68] C. Yim and A. Bovik, "Quality assessment of deblocked images," *Image Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 88 –98, jan. 2011.

[69] K. Hanke, P. Hosten, and F. Jager, "Content-adaptive encoder optimization of the h.264/avc deblocking filter for visual quality improvement," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*, nov. 2011, pp. 1 –4.

[70] X. Chen, W. Xia, and X. Lu, "A high-throughput low-power hardware architecture for h.264 deblocking filter," in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 2, april 2010, pp. V2–561 –V2–565.

[71] K. Xu and C.-S. Choy, "A five-stage pipeline, 204 cycles/mb, single-port sram-based deblocking filter for h.264/avc," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 18, no. 3, pp. 363 –374, march 2008.

[72] H. S. T. K. T. Jens-Rainer Ohm, Gary J. Sullivan and T. Wiegand, "Comparison of the coding efficiency of video coding standards - including high efficiency video coding (hevc)," *IEEE TRANS. ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, p. to be published, 2012.

[73] T. Wedi and T. Tan, "Ahg report - coding efficiency improvements, vceg-aa06," 2005.

[74] G. Sullivan and T. Wiegand, "meeting report for q.6 meeting in marrakech, morocco, vceg-ae01," Marrakech, Morocco, 15-16/Jan 2007.

[75] J. Lainema and K. Ugur, "Angular intra prediction in high efficiency video coding (hevc)," in *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, oct. 2011, pp. 1 –5.

[76] D. Palomino, F. Sampaio, L. Agostini, S. Bampi, and A. Susin, "A memory aware and multiplierless vlsi architecture for the complete intra prediction of the hevc emerging standard," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, 2012, pp. 201–204.

[77] E. Kalali, Y. Adibelli, and I. Hamzaoglu, "A high performance and low energy intra prediction hardware for high efficiency video coding," in *2012 22nd International Conference on Field Programmable Logic and Applications (FPL),*, 2012, pp. 719–722.

[78] ——, "A high performance and low energy intra prediction hardware for hevc video decoding," in *2012 Conference on Design and Architectures for Signal and Image Processing (DASIP),*, 2012, pp. 1–8.

[79] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, "High efficiency video coding (hevc) text specification draft 10," in *JCTVC-L1003*, Jan. 2013.

[80] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[81] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[82] L. Zhao, L. Zhang, S. Ma, and D. Zhao, "Fast mode decision algorithm for intra prediction in HEVC," in *2011 IEEE Visual Communications and Image Processing (VCIP),*, nov. 2011, pp. 1 –4.

[83] G. Correa, P. Assuncao, L. Agostini, and L. da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, November.

[84] Y. Jiang and M. Pattichis, "Dynamically reconfigurable DCT architectures based on bitrate, power, and image quality considerations," in *2012 19th IEEE International Conference on Image Processing (ICIP)*, 2012, pp. 2465–2468.

[85] I.-K. Kim, K. McCann, K. Sugimoto, B. Bross, and W.-J. Han, "High efficiency video coding (HEVC) test model 11 (HM11) encoder description,JCTVC-M1002," 2013.

[86] J. C. Yinji Piao, Junghye Min, "Encoder improvement of unified intra prediction, JCTVC-C207," 2010.

[87] F. Bossen, "HM 8 common test conditions and software reference configurations, JCTVC-J1100," 2012.