

國立交通大學

電子工程學系 電子研究所

博士論文

視訊嵌入轉碼器之演算法與其硬體架構設計空間探討



An Algorithm and Its Architecture Design Space

Exploration of a Video Embedding Transcoder

研究生：李志鴻

指導教授：蔣迪豪 教授

中華民國九十七年六月



視訊嵌入轉碼器之演算法與其硬體架構設計空間探討

An Algorithm and Its Architecture Design Space

Exploration of a Video Embedding Transcoder

研究生：李志鴻

Student : Chih-Hung Li

指導教授：蔣迪豪

Advisor : Dr. Tihao Chiang

國立交通大學

電子工程學系 電子研究所



A Dissertation

Submitted to Department of Electronics Engineering and
Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electronics Engineering

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月



視訊嵌入轉碼器之演算法與其硬體架構設計空間探討

研究生：李志鴻

指導教授：蔣迪豪 博士

國立交通大學

電子工程學系暨電子研究所

摘要

視訊嵌入服務在今日多元化的多媒體應用中越來越廣泛，由於多媒體的龐大資料量，現今大部分的視訊資料都以壓縮的格式儲存與傳遞，在眾多壓縮標準中，H.264/AVC 已成為目前視訊壓縮的主流，因此針對 H.264/AVC 標準的視訊嵌入轉碼器將更加重要，以達到資料儲存以及網路傳輸的高效率。本論文主要是針對視訊嵌入轉碼器的演算法發展與硬體架構設計探索：第一，我們所發表的 H.264/AVC 視訊嵌入快速轉碼演算法乃為目前文獻上第一篇在 H.264/AVC 標準下的視訊嵌入轉碼技術。第二，關於硬體架構設計，我們利用最低的成本，成功結合了 H.264/AVC 轉碼與編碼功能於單一個硬體架構中，此乃文獻中第一個實現此技術的設計。第三，關於硬體設計探索(Design Space Exploration)，則是文獻上第一篇以資料交換層級(Transaction Level Modeling)做系統效能模擬分析的 H.264/AVC 相關之硬體設計。

本論文第一部份著重於多視窗視訊嵌入轉碼器(Multiple-Window Video Embedding Transcoder)之低複雜度演算法的發展。為解決傳統上串聯式像素值域轉碼器(Cascaded Pixel Domain Transcoder)的高複雜度難題，我們採用部份重新壓縮(Partial Re-encoding)的概念來降低轉碼時所需的運算量，並減少因重新量化(Re-quantization)所造成的轉碼視訊品質下降。針對預測不協調(Prediction Mismatch)的區塊，我們利用原始壓縮位元流內的資訊來幫助預測微調(Prediction Refinement)，即幅內模式轉換(Intra Mode Switching)以及運動向量重新映射(Motion Vector Re-mapping)，如此，我們可以完全去除壓縮器中複雜度最高的兩個模組：模式決策(Mode Decision)和運動估計(Motion Estimation)。針對殘餘值不協調(Residue Mismatch)的區塊，我們從理論與實驗數據的推導，有效率的找出最需要做錯誤修正(Error Correction)的區塊，從實驗數據顯示，我們只需針對極

少部份區塊作錯誤修正，即可將轉碼視訊品質大幅提高 2dB 左右。比起串聯式像素值域轉碼器，我們可以將轉碼速率提高 25 倍，更令人驚訝的，我們的低複雜度演算法最多可以有將近 1.5dB 的 PSNR 改善。

本論文第二部份著重於設計空間探索(Design Space Exploration)。基於所提出的低複雜度演算法，我們將之實現於平台式(Platform-based Design)系統設計並以最節省的成本將 H.264/AVC 轉碼器與解碼器結合在一個平台上。在所提出的高效能系統硬體架構中，我們針對幾個重要的系統參數做探索：硬體平行度(Parallelism)、資料交換精細度(Data Exchange Granularity)以及設計平衡(Design Balancing)。不同於傳統由下而上的設計哲學(Bottom-Up Design Methodology)，我們採用新穎的由上而下、逐步精確的設計哲學(Top-Down and Refinement-based Design Methodology)以獲得較優異的探索效能。我們主要採用電子系統層級(Electronic System Level)來做系統模擬以及探索，其模擬平台大都是操作在資料交換層級(Transaction Level Modeling)，其模擬的效能較傳統的暫存器傳輸級(Register Transfer Level)快上三個數量級左右。因此，比起傳統的系統設計，我們的設計提供相當大的自由空間來針對不同的設計限制作最佳化：針對硬體成本最佳化，我們的設計選擇(Design Alternative)可以將硬體成本降低至原本的 25%。針對速度最佳化，我們的設計選擇可以將速度增加為原本的兩倍。在 135MHz 的操作時脈下，我們可以針對 1920x1088 每秒 60 幅的高畫質視訊提供即時的轉碼或解碼輸出。

本論文第三部份著重於低成本高效能的硬體模組設計開發，我們著重在兩個核心區塊：像素預測(Pixel Prediction)和去邊濾波器(Deblocking Filter)。第一，我們成功結合了 H.264/AVC 中幅內與幅間預測於單一個硬體架構中，除了增加硬體使用效率外，亦大幅減少資料匯流排上的傳輸。第二，我們提出了一個具有同步細緻可調(Fine-Grained Synchronization Capability)的去邊濾波器，如此可以讓視訊資料管線(Video Pipe)的效能在不同的資料交換精細度(Level of Granularity)中，都能獲得提升。

總結，本論文提出一個低複雜度、高效率之視訊嵌入轉碼器。在演算法上，我們著重於快速有效率的微調與修正。在硬體設計空間探索上，我們有效率且量化地分析了各個系統參數的影響，以期在不同的設計限制下都能獲得最佳化。在硬體設計上，我們著重於硬體使用效率的增加以及整體系統效能的增加。

An Algorithm and Its Architecture Design Space

Exploration of a Video Embedding Transcoder

Student: Chih-Hung Li

Advisor: Dr. Tihao Chiang

Department of Electronics Engineering & Institute of Electronics
National Chiao Tung University

ABSTRACT

As the H.264/AVC standard is receiving worldwide adoption, the video embedding service is an important feature and thus this thesis presents an H.264/AVC multiple-window video embedding transcoder in three parts including an algorithm, a system architecture design space exploration, and two novel micro-architectures.

The first part describes a low-complexity algorithm as compared to the traditional cascaded pixel domain transcoder. The partial re-encoding is adopted to reduce the complexity and quality degradation due to re-quantization. Specifically, the intra mode switching and motion vector re-mapping techniques are used to eliminate the need for the mode decision and motion estimation modules. Moreover, with the theoretical analysis, only 5% of the total blocks needs error correction. The proposed approach can improve the quality up to 1.5 dB and enhance the throughput by 25 times as compared to the traditional cascaded transcoder.

The second part describes architecture for platform-based video embedding transcoder and its design space exploration from several system aspects: hardware parallelism, data exchange granularity and design load balancing using transaction level modeling. The top-down refinement-based design methodology provides effective exploration with high degree of freedom to optimize for various design constraints. Further, it identifies which critical module and how it can be optimized in terms of speed, memory and bandwidth for improving the overall performance. Our best design alternative can reduce the cost by four times or speed up by two times such that our design can achieve 1920x1088 @ 60 Hz video transcoding at 135MHz.

The third part describes the two novel micro-architectures designed for the prediction and the deblocking filter modules. The prediction is unified with a systolic architecture to improve the hardware utilization and the transmission bandwidth. The de-blocking filter is implemented with a multi-level fine-grain synchronization granularity to improve the system performance at finer level of granularity.

誌 謝

進入研究所的七個年頭，累積了太多的感謝。

首先我要感謝我的指導教授蔣迪豪老師在這七年中給我的諸多指導，不僅在學術研究的方向上不斷地引領我走在專業領域的先端，更讓我培養了獨立解決問題以及清楚表達想法的能力。每次與蔣老師的討論都像是腦力激盪，一點一滴激發我向前進步的潛能。除此之外，我也非常感謝蔣老師這些年來在各方面給我的關心鼓勵與幫助。

實驗室中優秀的學長學弟們則是研究生涯當中另一個幫助我進步的動力。王俊能學長是最早鼓勵我攻讀博士學位的，在我最青澀的時期，不厭其煩地修改我殘破的論文寫作，更在我中途最灰心的時期再次給予我堅持到底的信心。彭文孝學長則是在我博士班後期帶領我走進硬體設計的相關領域並教導我一些研究方法與論文寫作上的經驗。黃項群學長清晰敏銳的思路以及對於研究的熱情，一直是我所景仰與學習的榜樣。王世豪學長在硬體設計與伺服器架設的熟練經驗，也常是我打擾請教的對象。親切熱心的李俊毅學長總像個大哥哥般似的幫我解決在實驗室遇到的任何問題。而與思考敏銳的治傑共同研究則是充滿了愉快。其他不管是已畢業或是仍在實驗室打拼的學長同學與學弟們，峯誠、家揚、崑健、健霖、耀中、秉玉、沛昀、宗延、鑑明、偉倫、孝強、子良、振韋、朝雄、志凱、世騫、鴻志、世焯、德宣，謝謝你們一同營造這個讓我感動的實驗室。

在此我也要感謝張隆紋老師、黃仲陵老師、李鎮宜老師、王聖智老師、李國君老師、蔡宗漢老師百忙之中撥冗前來參予我的口試，給予我在求學生涯上的最後一堂課，因為有你們的寶貴意見使得論文能夠更加完備。尤其是李國君老師在過去這一年多的時間不厭其煩地給予我在系統層級設計上的一些專業意見。

謝謝這些在學術研究上不斷幫助我的貴人，讓我能夠以更謙卑的心態來看待這個學位，期待這個學位能夠成為往後不斷督促我進步的動力。

最後，我要感謝我的家人，尤其是媽媽在我的求學過程中，總是給我無條件的支持，可以讓我無後顧之憂做自己想做的事。而我的女友楓珮總是直接且深刻地感受我在研究與撰寫論文時的種種煎熬與壓力，在我低潮失意的時候，更是給予我最大的鼓勵與幫助，陪伴我完成這個人生中重要的里程碑。

簡短的文字實在難以完全紓發內心的由衷感激，沒有你們這些人，也許沒有今天的我，也謝謝老天給我足夠多的運氣，謹以此論文獻給所有關心我的人，希望我的努力沒有辜負你們的期望。

李志鴻


謹誌於台灣新竹交通大學

西元 2008 年 7 月

To My Mother and Girlfriend



Contents

Abstract in Chinese		i
Abstract		iii
Acknowledgements		iv
Contents		vi
List of Tables		x
List of Figures		xii
List of Notations		xviii
1 Introduction		1
1.1 Overview of Dissertation		1
1.1.1 Motivation		3
1.1.2 1 st Focus: Low-Complexity Algorithm Development		4
1.1.3 2 nd Focus: System Architecture Design Space Exploration		5
1.1.4 3 rd Focus: Highly Efficient Micro-Architecture Design		7
1.1.5 Attractive Applications of Video Embedding Transcoding		8

1.2	Organization and Contribution	9
2	Background and Related Work on Video Embedding Transcoding	14
2.1	Introduction	14
2.2	Realization of Video Embedding Service	15
2.3	Problem Statement of Video Transcoding	16
2.4	Wrong Reference Problem Formulation	19
2.5	Related Work on Video Embedding Transcoding	20
2.5.1	Cascaded Pixel Domain Transcoder (CPDT)	20
2.5.2	DCT Domain Transcoding with Motion Vector Re-mapping	21
2.5.3	DCT Domain Transcoding with Backtracking	22
2.6	The Challenge in H.264/AVC-based PIP Transcoding	22
2.7	Summary	25
3	Low-Complexity Algorithm of MW-VET	26
3.1	Introduction	26
3.2	Slice-Group-Based Transcoding	28
3.3	No Frame Memory Transcoding (NFMT)	30
3.3.1	Architecture of NFMT	30
3.3.2	Auxiliary Bitstream Generation	32
3.4	Reduced Frame Memory Transcoding (RFMT)	32
3.4.1	Intra Mode Switching (IMS)	37
3.4.2	Motion Vector Remapping (MVR)	43
3.4.3	Syntax Level Bypassing (SLB)	50
3.5	Simulation Results	52
3.6	Summary	61
4	System Architecture Design Space Exploration	65
4.1	Introduction	65
4.2	Algorithm to Architecture Mapping	67
4.3	Highly Efficient System Architecture	68
4.3.1	Memory Hierarchy	72
4.3.2	Video Decoding Pipe	73
4.3.3	Video Embedding Transcoding Pipe	75

4.3.4	Memory Sub-system	75
4.3.5	Task Scheduling	78
4.4	Effective Design Space Exploration	80
4.5	Pruned Design Space	84
4.5.1	Exploration of the Synchronization Granularity	84
4.5.2	Exploration of the Design Combination and Balancing	87
4.6	Evaluation of the System Performance	89
4.6.1	Evaluation Metric	89
4.6.2	Simulation Infrastructure	91
4.6.3	Pareto-Based Multi-Objective Optimization	94
4.7	Simulation Results and Analysis	95
4.7.1	Pareto Analysis for the Exploration of Synchronization Granularity	95
4.7.2	Pareto Analysis for the Exploration of Design Combination within the Video Pipe	101
4.7.3	Area-Weighted Hardware Utilization for the Exploration of Design Bal- ancing of the Video Pipe	102
4.8	Summary	103
5	A Highly Efficient Micro-Architecture Design	106
5.1	Introduction	106
5.2	An Efficient Memory Sub-System	108
5.2.1	Background	108
5.2.2	Interleaved Data Arrangement	109
5.2.3	External Memory Interface	112
5.2.4	Synchronization Buffer	114
5.2.5	Effect of Synchronization Granularity of Memory Sub-system on Off- Chip Transmission	116
5.3	Combined Inter and Intra Prediction	121
5.3.1	Motivation	122
5.3.2	Overall Architecture	124
5.3.3	Data Flow of the Inter Prediction	129
5.3.4	Data Flow of Intra Prediction	133
5.3.5	Analysis and Comparison	140

5.4	Efficient Deblocking Filter with Fine-Grained Synchronization Capability . . .	146
5.4.1	The Proposed Architecture of Deblocking Filter	153
5.4.2	The Proposed Filtering Order with Fine-Grained Synchronization Ca- pability	154
5.5	Summary	155
6	Conclusions	157
6.1	Summary of Contributions	158
6.1.1	Improvement of Rate-Distortion Performance	158
6.1.2	Design Space Exploration	159
6.1.3	Improvement of Area-Speed Efficiency	160
6.2	Suggestions for Future Works	162
	Bibliography	163
	Appendix	174
A	Why Transform Domain Approaches are Inefficient for H.264 Transcoding	174
A.1	Integer Transform with Quantization Scaling	175
A.2	Directional Intra Prediction	176
A.3	In-the-loop De-blocking Filtering	178
A.4	Sub-pixel Interpolation	178
B	Constant-Rate Bumping Process	182
B.1	Bumping Process in H.264/AVC	183
B.2	Proposed Constant-Rate Bumping Process	184

List of Tables

3.1	The Symbol Definitions	29
3.2	The Cases of the Intra4 Mode Switching	39
3.3	The Cases of the Intra16 Mode Switching	39
3.4	The Corresponding Operations of the RFMT for Each Block Type During the VET Transcoding	52
3.5	The Detailed Refinement during the VET Transcoding	53
3.6	The Encoder Parameters for the Experiments	56
3.7	The Improvement of Execution Time and Quality as Compared to the CPDT ⁽¹⁾	57
3.8	The Effectiveness of Error Correction (EC) for Different Kinds of p-blocks	58
4.1	The Transcoding Throughput Performance of Proposed Algorithm of a Software Implementation	66
4.2	The Transcoding Throughput Performance of Proposed Algorithm of a Software Implementation	78
4.3	The Bitrate of Each Long Sequence with Different Resolution	98
4.4	The Required Clock For Each Resolution When the Size of Display Buffer is of 32Mb.	101

5.1	Analysis of the Different Levels of Granularities Based on the Worst Case Assumption	116
5.2	The Comparison of the Intra Prediction	140
5.3	Comparison of Inter Prediction.	143
5.4	The Execution Cycle of the Inter and Intra Prediction in All Cases	144
5.5	The Corresponding Data Path of Each Filtered Edge	156
A.1	Computational Complexity of Each Intra Prediction Mode For Both Operation Domain	177



List of Figures

1.1	The Applications of the Video Embedding Service: (a) Live TV Program of CBBC. (b) Leatek’s WinFast Series Product. (c) Disney Supports "Full Motion, Picture-in-Picture Bonus Features" in Newly Released Blu-Ray DVD. (d) Multiple Stream Media Processor of DiscoveryBiz. (e) TV Setup Box of Allthings. (f) Sony Debuts In-Car Navigation with PIP in Japan. (g) Videoconferencing System of Wirered Company. (h) The PIP Functionality in Viliv X2 AIO. (i) YouTube Overlays Ads during Video Streaming (InVideo). (j) Video Surveillance System.	2
1.2	One of the Applications of Video Embedding Transcoder: Transcoding Server.	9
1.3	The Design Flow in This Dissertation	13
2.1	Illustration of a Novel Transcoder: (a) The Simplified Transcoding Process. (b) The Simplified Transcoder When the Prediction Blocks Are the Same. (c) The Fast Transcoder That Bypasses the Input Transform Coefficients.	18
2.2	Illustration of the Wrong Reference Problem	20
2.3	The Architecture of the CPDT	21
3.1	The Example of Slice Group That Can Be Used in the VET Transcoding	30
3.2	The Architecture of the No Frame Memory Transcoder (NFMT)	31

3.3	The Generation of the Auxiliary Bitstream Based on the Reconstruction of a RDO Encoder	33
3.4	The Initial Architecture of the RFMT with RDO Refinement Based on the Concept of the Partially Re-Encoding	34
3.5	The Intermediate Architecture of the RFMT with the MVR and the IMS Refinement	35
3.6	The Final architecture of the RFMT with Shared Frame Memory and the Constrained FG Bitstreams	36
3.7	The Transcoding Scheme for the Channel Preview	37
3.8	The Wrong Intra Reference Problem within a Macroblock Depending on the Intra Modes	38
3.9	The Relative Position of Each Case in the Intra Mode Switching Technique	38
3.10	An Example of the Intra Prediction Chain	40
3.11	Illustration of the Motion Vector Re-mapping Technique. (a) The Original Coding Mode and Motion Vectors. (b) Refinement by Using Inter4x4 Mode and Re-mapped Motion Vectors.	44
3.12	An Example of the Inter Prediction Chain	47
3.13	The Flow Chart of the Proposed RFMT	53
3.14	The Extended Wrong Reference Problem when Multiple Reference Frame is Used	54
3.15	The Visual Illustration after Each Refinement Step during the VET Transcoding	55
3.16	The Percentage of the Macroblock Types and the Block types during the VET Transcoding	57
3.17	The Rate-Distortion Performance of the Luminance Component When One Foreground Carphone_QCIF is Embedded in Table_SD: (a) Table_SD_Carphone_QCIF_1_1. (b) Table_SD_Carphone_QCIF_33_20.	59
3.18	The Rate-Distortion Performance of the Luminance Component When One Foreground Foreman_QCIF is Embedded in Mobile_SD: (a) Mobile_SD_Foreman_QCIF_1_1. (b) Mobile_SD_Foreman_QCIF_33_20.	60

3.19	The Rate-Distortion Performance of the Luminance Component by Four Fore- grounds Embedding with the Single-Generation Transcoding: (a) Table_SD_MD_QCIF_1_1 _Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20. (b) Mobile_ SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF _33_20.	62
3.20	The Rate-Distortion Performance of the Luminance Component by Four Fore- grounds Embedding with the Multi-Generation Transcoding: (a) Table_SD_MD_QCIF_1_1 _Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20. (b) Mobile_ SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF _33_20.	63
4.1	The Top Level Data Flow of RFMT and Its System Partitioning	69
4.2	The Refinement for Each Processing Macroblock and Its Hardware Partitioning which is of Five- Staged Pipeline	70
4.3	The System Architecture of the Proposed Video Embedded Transcoder and De- coder	72
4.4	The subjective Quality Comparison of the 100th Frame: (a) Decoded Picture without Refinement Update (b) Decoded Picture with Refinement Update (c) Transcoded Picture without Refinement Update (d) Transcoded Picture with Refinement Update.	76
4.5	The scheduling of the Video Pipe Where P Means the CPU Programs the Indi- vidual Module	79
4.6	The Scheduling of the Architecture A	79
4.7	The Scheduling of the Architecture B	80
4.8	The Scheduling of the Architecture C	81
4.9	The Scheduling of the Architecture D	82
4.10	The Scheduling of the Architecture E	83
4.11	The Scheduling of the Architecture F	84
4.12	The Flow of Our Design Space Exploration	85
4.13	The Exploration of the Synchronization Granularity	87
4.14	The SystemC Implementation with the Annotated Timing	93
4.15	The Proposed H.264/AVC Video Embedding Transcoder and Decoder Imple- mented by the Platform Architect of CoWare ConvergenSC	94

4.16	The Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count at Different Levels of Synchronization Granularity and for the Different Designs of the Inter and Intra Prediction. The Combination of the Level of Synchronization Granularity to be Explored Includes 16x16_16x16, 16x16_8x8, 16x16_4x4, 8x8_8x8, 8x8_4x4, and 4x4_4x4 where the Terms before and after the Underscore Indicate the GM and the GV, Respectively	96
4.17	The Combined Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count at Different Levels of Synchronization Granularity and for the Different Designs of the Inter and Intra Prediction	97
4.18	The Execution Cycle of Architecture 16_16_SA3 and The Minimized Clock for Real-Time When the Size of Display Buffer is 32Mb. (a) 240x144. (b) 480x272. (c) 960x544. (d) 1920x1088.	100
4.19	The Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count for Each Alternative where MB and B8 Denote the MB-based and B8-based DB, Respectively	102
4.20	The Area-Weighted Hardware Utilization for Each Design Alternative	103
4.21	Normalized Distance to Utopia Point Considering Execution Cycle Count, Hardware Cost, and Area-Weighted Utilization	104
5.1	The Interleaved Data Arrangement for the Stored Pictures	111
5.2	The Functional Block Diagram of the External Memory Interface	112
5.3	The Command FIFO for the Detection of the Row Miss	114
5.4	The Finite State Machine Designed for the Mobile DDR SDRAM	115
5.5	The Efficiency of DRAM Access for Motion Compensation. The Efficiency is Defined as (# of Data Read from the External DRAM) / (Actual Data Required for the Motion Compensation)	118
5.6	The Efficiency of DRAM Access for the Decoder. The Efficiency is Defined as (# of Data Read from the External DRAM) / (Actual Data Required for Decoder)	119
5.7	The Average Cycle Counts per MB	120
5.8	The Amount of Data Transfer	120
5.9	The Power Consumption in DRAM	121
5.10	The Architecture of Inter and Intra Prediction	125

5.11	The 2-D Interpolation for the Motion Compensation with Sub-Pel Precision. Note that the 2-D Filtering Can Be Separated Into Two 1-D Filtering.	126
5.12	(a) The Unified Systolic Array for Both Inter and Intra Interpolation. (b) The Block Diagram of Functional Block W. (c) The Weighting Mode of Functional Block W. (d) The Combination Mode of Functional Block W.	128
5.13	The Configuration of Inter Prediction for Luminance Component	130
5.14	The Weighting Factor of Each Input for Consecutive Execution of the 6-tap Filtering	131
5.15	Input Scheduling of the Proposed Systolic Array that Uses Two-Input Broad- casting	131
5.16	Separated Filterings of the Inter Prediction for Chrominance Component	133
5.17	The Configuration of Inter Prediction for Chrominance Component	134
5.18	Intra Prediction by Adaptive Filtering	136
5.19	The Configuration of Intra Prediction for the Direction Modes. (a) The Two (1 ,2, 1) Filters. (b) The (1, 1) and (1, 2, 1) Filter.	137
5.20	The Progression Property of Plane Mode	139
5.21	The Configuration of Intra Prediction for the DC Mode and the Plane Mode. (a) The Data Path for Accumulation. (b) The Generation of Gradient Values "b" and "c". (c) The Pixel Prediction of Plane Mode at Odd Cycles. (d) The Pixel Prediction of Plane Mode at Even Cycles.	141
5.22	The Execution Cycle of the Inter and Intra Prediction for the Blue_Sky Sequence	145
5.23	The Execution Cycle of the Inter and Intra Prediction for the Pedestrian_Area Sequence	146
5.24	The Execution Cycle of Inter and Intra Prediction for Riverbed Sequence.	147
5.25	The Execution Cycle of Inter and Intra Prediction for Rush_Hour Sequence.	147
5.26	The Execution Cycle of the Inter and Intra Prediction for the Station2 Sequence	148
5.27	The Execution Cycle of the Inter and Intra Prediction for the Sunflower Sequence	148
5.28	The Execution Cycle of the Inter and Intra Prediction for the Tractor Sequence	149
5.29	The Data Transmission via AHB Data Bus for the Blue_Sky Sequence	149
5.30	The Data Transmission via AHB Data Bus for the Pedestrian_Area Sequence	150
5.31	The Data Transmission via AHB Data Bus for the Riverbed Sequence	150
5.32	The Data Transmission via AHB Data Bus for the Rush_Hour Sequence	151

5.33	The Data Transmission via AHB Data Bus for the Station2 Sequence	151
5.34	The Data Transmission via AHB Data Bus for the Sunflower Sequence	152
5.35	The Data Transmission via AHB Data Bus for the Tractor Sequence	152
5.36	The Architecture of the Proposed Deblocking Filter	154
5.37	The Proposed Edge Filtering Order with Fine-Grained Synchronization Capability	155
6.1	The Improvement of Rate-Distortion Performance by the Proposed Low-Complexity Algorithm	159
6.2	The Improvement of Throughput and Hardware Cost by the Proposed Effective Design Space Exploration	160
6.3	The Improvement of Area-Speed Efficiency by the Proposed High-Efficient Architecture	161
B.1	The flow chart of the bumping process in H.264/AVC. The “smallest_poc” stands for the smallest picture order count (POC) among the non-output pictures in the DPB.	183
B.2	The flow chart of the proposed constant-rate bumping process. The “smallest_poc” stands for the smallest picture order count (POC) among the non-output pictures in the DPB and the “RB” denotes the regulation buffer.	185
B.3	Example for Comparing the Variable-Rate Bumping Process and the Proposed Constant-Rate Bumping Process	186

List of Notations

$HT(\cdot)$	Operation of integer transform
$IHT(\cdot)$	Operation of inverse integer transform
$Q(\cdot)$	Operation of quantization
$DQ(\cdot)$	Operation of de-quantization
$PRED(\cdot)$	Operation of pixel prediction
P_e	Encoding process from pixel domain to transform domain
P_d	Decoding process from transform domain to pixel domain
$MC(\cdot)$	Operation of motion compensation
$IP_n(\cdot)$	Operation of intra prediction with mode n
\overline{r}_n	Original residue of the n -th block
\overline{x}_n	Decoded pixels of the n -th block
\overline{r}_n'	Refined residue of the n -th block
\overline{x}_n'	Decoded pixels of the n -th block after refinement
$\overline{\overline{x}}_n$	Decoded pixels of the n -th block without refinement

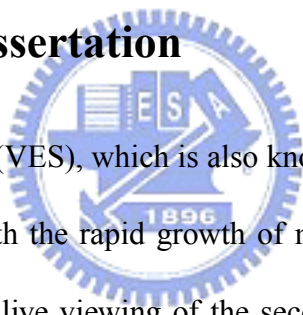
e_n	Quantization error of n -th block
$P[i]$	Cycle count of the system at i -th pipelined stage
$E_j[i]$	Cycle count of the j -th component at i -th pipelined stage
$u_j[i]$	Hardware utilization of the j -th component at i -th pipelined stage
μ_X	Mean value of the random variable X



CHAPTER 1

Introduction

1.1 Overview of Dissertation



The Video Embedding Service (VES), which is also known as the Picture-in-Picture (PIP) feature, attracts wide attention with the rapid growth of multimedia applications. With the PIP functionality, the VES enables live viewing of the second channel, advanced video mosaics, next-generation video programming guides, and user-configurable multi-view screens. The applications are summarized as follows and the visual experience of these applications is illustrated in Figure 1.1.

- Internet protocol television service (IPTV) such as video over IP (VoIP) [1]
- Internet/Mobile interactive applications such as video on demand (VoD)
- Commercial insertion [2]
- Video conferencing [3]
- Video surveillance
- Entertainment [4]
- Live news [5]



Figure 1.1: The Applications of the Video Embedding Service: (a) Live TV Program of CBBC. (b) Leatek's WinFast Series Product. (c) Disney Supports "Full Motion, Picture-in-Picture Bonus Features" in Newly Released Blu-Ray DVD. (d) Multiple Stream Media Processor of DiscoveryBiz. (e) TV Setup Box of Allthings. (f) Sony Debuts In-Car Navigation with PIP in Japan. (g) Videoconferencing System of Wireded Company. (h) The PIP Functionality in Viliv X2 AIO. (i) YouTube Overlays Ads during Video Streaming (InVideo). (j) Video Surveillance System.

The video compression is essential to delivery and distribution of multimedia information. In most multimedia applications, video content is stored in the compressed format to minimize transmission bandwidth and storage requirement. With the superior coding efficiency and network friendliness, the H.264/AVC [6] is regarded as the multimedia standard for service providers to deliver digital video contents over Local Access Networks (LAN), Digital Subscriber Line (DSL), Integrated Services Digital Network (ISDN) and third generation (3G) mobile systems [7]. Particularly, the next generation Internet protocol television service (IPTV) could be realized with H.264/AVC over very-high-bit-rate DSL (VDSL), which can support higher transmission rates up to 52 Mbps [8]. The high transmission bandwidth facilitates the development of video services with more functionalities and higher interactivity for video over DSL applications.

1.1.1 Motivation



To address the compressed video embedding technique over Internet and wireless channels, the idea of multiple-window video embedding transcoding (MW-VET) is proposed to deliver selected video contents that are encapsulated as one single bitstream. In such applications, the video may be transmitted over error-prone channels with limited bandwidth. To transmit the video contents via the single channel, the transcoder embeds downsized video frames into another frame with a specified resolution as the foreground pictures.

The most straightforward implementation of transcoding is the conventional cascaded pixel domain transcoder (CPDT) which is nothing more than a concatenation of several decoders and an encoder. It offers drift free performance with the highest computational cost. However, the complexity of the CPDT approach is relatively high at both algorithm and architecture levels. Furthermore, from the video compression perspective, the CPDT is inefficient because the

existing correlations between input and output bitstream are not utilized at all.

To address the high complexity issue of CPDT, our goal is to implement an H.264/AVC based VET transcoder while simultaneously (1) increasing throughput, (2) improving quality, (3) increasing rate-distortion (R-D) performance, (4) reducing cost, and (5) increasing cost-effective. First of all, a low-complexity transcoding algorithm is proposed while the R-D performance can also be improved as compared with the CPDT approach. Second, a highly efficient architecture is presented for combining H.264/AVC based video embedding transcoder (VET) and decoder such that the throughput performance can be further improved. Third, the design space is explored at a higher level of abstraction to find the design the most cost effective alternative. Fourth, the micro-architectures are designed for the three hot spot modules within our system architecture. Specifically, this dissertation will focus on several aspects as follows.

1.1.2 1st Focus: Low-Complexity Algorithm Development

To maintain transcoded picture quality and to reduce the overall complexity, three transcoding techniques are presented for a multiple-window video embedding transcoder (MW-VET) at the algorithm level: 1) slice-group-based transcoding (SGT), 2) reduced frame memory transcoding (RFMT) and 3) syntax level bypassing (SLB). The application of each transcoding technique depends on the data partitions of the archived bitstreams and the paths of error propagation. For the slice-aligned data partitions, the SGT that composes the VET bitstreams at the bitstream level can provide the highest throughput. For the region-aligned data partitions, the RFMT efficiently refines the prediction mismatch and increases the throughput while maintaining better R-D performance. For the blocks that are not affected by the drift error, the SLB de-multiplexes and multiplexes the bitstreams into a VET bitstream at the syntax element level.

To exploit the correlations among bitstreams before and after transcoding while maintaining

the coding efficiency, we use information from the incoming bitstream to make some simplification during re-encoding. Particularly, we propose the motion vector re-mapping and intra mode switching to eliminate the motion vector estimation and mode decision which are the most computational intensive functional blocks. On the other hand, to alleviate the quality degradation due to drift error while minimizing the computation complexity, we perform the error correction for some key blocks which occupies a small part of frame.

Our simulation results show that the proposed algorithm significantly reduces processing complexity by 25 times in terms of the time required to execute with similar or even higher R-D performance as compared to the conventional cascaded pixel domain transcoder. Particularly, the proposed algorithm can achieve up to 1.5 dB quality improvement in Peak Signal to Noise Ratio (PSNR).



1.1.3 2nd Focus: System Architecture Design Space Exploration

To further increase the throughput performance, we implement the proposed low complexity algorithm by a highly efficient architecture that combines H.264/AVC VET transcoder and decoder. We use the ARM platform based design and partition the system into several dedicated modules to provide task level parallelism. In addition, the proposed architecture can perform video decoding and transcoding alternatively or simultaneously. Such VET enabled decoders can find their applications in a peer-to-peer service community where every user enjoys versatile video embedded services while contributing partial computation power.

To solve the problems in traditional platform-based design, we implement most functional blocks by dedicated modules interconnected with the shared ping-pong memories to provide massive parallelism. First, all the computational intensive function block are partitioned as the hardware module such that the workload of ARM core can be significantly alleviated. Sec-

ond, we connect each module such that most of data communications take place through the video pipe instead of between ARM core and AHB bus. Thirdly, we allocate ping-pong buffer between adjacent pipeline stages such that computation and communication cycle can be overlapped.

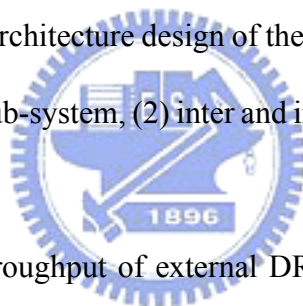
The system performance of a pipelined system mainly depends on the synchronization overhead. The pipeline is limited by the slowest module in the video pipe, while each module consumes certain cycles at each pipeline stage according to the various coding characteristics such that the synchronization overhead is introduced. However, the traditional bottom-up design methodology focuses on the individual performance of each module rather than considering the impact of synchronization overhead on the system level performance. Thus, in this thesis, we exploit the top-down design methodology to examine the effects of different design combinations with respect to system performance in order to explore the design space and ensure good tradeoffs between cost and performance. Such design space exploration enables us to find an optimized tradeoff between cost and performance. In addition, we analyze the load balancing by normalizing resource utilization weighted by the associated cost. Another factor of synchronization overhead is the level of synchronization granularity. In general, coarser granularity of synchronization conducts less synchronization overhead at the expense of more memory. Most designs adopt macroblock-level pipeline for the inter prediction and deblocking filter for best throughput performance. In this dissertation, we will exploit the impact of different level of synchronization granularity on the synchronization overhead.

Lastly, the proposed architecture is verified and simulated at system level using transaction level modeling (TLM) technique. We implement the architecture to ensure the correctness of data flow at system level and perform the design space exploration for the two system parameters: (1) the level of synchronization granularity and (2) design combinations within the video

pipe. These models are evaluated at the TLM level and thus it minimizes the modeling effort and increases the simulation speed so as to explore 185 design alternatives. We then evaluate the system performance in terms of the average cycle count, the equivalent gate count and the cost weighted hardware utilization. From the system level simulation with TLM, the design alternative optimized for cost can reduce the area of previous design up to 25%. The design alternative optimized for speed can increase the throughput of previous design by 2 times at most such that our design can fulfill the real-time requirement for 1920x1080 @ 60 Hz videos when clocking at 135MHz.

1.1.4 3rd Focus: Highly Efficient Micro-Architecture Design

We further focus on the micro-architecture design of the three key modules in our proposed system architecture: (1) memory sub-system, (2) inter and intra prediction (IIP), and (3) deblocking filter (DF).



To efficiently utilize the throughput of external DRAM, we propose an efficient memory sub-system, including (1) an interleaved data arrangement scheme for improving the efficiency of DRAM access, (2) an external memory interface for the control of mobile DDR SDRAM, (3) a synchronization buffer for data cache. Particularly, a synchronization buffer is employed as a bridge for reformatting the read/write data exchanged between the on-chip hardware and the off-chip DRAM. In addition, we optimize the issues of read/write commands and adaptively enable the auto-precharge function by monitoring the motion information of the input bitstream.

To increase the efficiency and the utilization, we propose a unified filtering architecture for the inter and intra prediction. First, the data paths are shared for both the inter and intra prediction so as to increase hardware utilization and reduce hardware cost. Second, to minimize redundant computations in the pixel predictions, the FIR filtering is implemented by a

re-programmable systolic architecture (SA). Thirdly, the proposed systolic architecture is fully utilized for any kind of interpolation and block partition. Fourthly, a local FIFO and memory is allocated for temporally buffering the motion-compensated data and the intermediate data such that the motion-compensated data of a block partition is transferred without redundant transmission. According to the simulation results, our combined, systolic based architecture for the inter and intra prediction to achieve throughput up to 4.5 times while decreasing up to 60% of the bus bandwidth.

To reduce the memory and the latency for buffering and to explore the level of synchronization granularity, we propose a novel deblocking filter with fine-grained synchronization capability (FGSC). In the finer granularity such as 8x8 or 4x4 block, our proposed deblocking filter can more efficiently process the input data at each pipeline stage than the other deblocking filter designs with macroblock-based processing order.

1.1.5 Attractive Applications of Video Embedding Transcoding

Figure 1.2 illustrates an application scenario for the VET that can be realized in practice application such as video on demand (VoD). Upon a client's request, the video server can use transcoding technique to compose two or more previously compressed bitstreams into one and then send the VET bitstream to the specific client. The bit-rate of transmission bitstream becomes much less because the secondary bitstreams are embedded directly instead of being carried additionally along the primary bitstream. Implementing the VES at the client side is neither cost effective as it requires multiple decoders at the client nor bandwidth-efficient as it needs to transmit multiple bit streams to the receiver. In addition, the VET can provide compliant bitstreams for facilitating the VES video playback in a way transparent to the normal decoder at the client side.

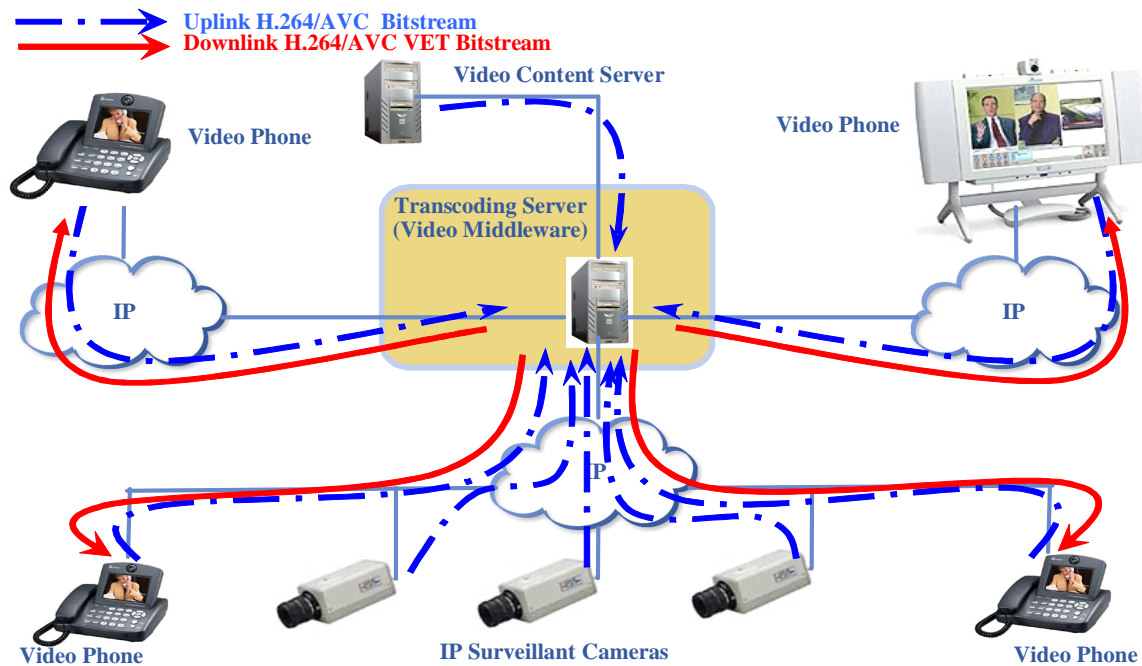


Figure 1.2: One of the Applications of Video Embedding Transcoder: Transcoding Server.

1.2 Organization and Contribution

In this dissertation, a low-complexity transcoding algorithm is developed to deliver higher coding efficiency with limited drifting errors. Moreover, a highly efficient system architecture is proposed with hybrid pipelined scheme to further improve transcoding throughput. Although the proposed schemes are mainly developed for Baseline Profile, these techniques are shown to be extended and tailored for the Main Profile and High Profile in H.264/AVC. For more details of each part, the rest of this thesis is organized as follows:

- Chapter 2 presents the related works on video embedding technique and shows the challenges in H.264/AVC based video transcoding.
- Chapter 3 proposes low-complexity algorithm that reduces the complexity of traditional transcoding scheme while improving the rate-distortion performance during transcoding.

Specifically, the contributions in the algorithm development include the following:

- All the inference mismatches of H.264/AVC syntax elements are refined during the

VET transcoding.

– The methodology of partially re-encoding is utilized and it not only reduces the complexity of transcoding, but also preserves the picture quality.

– Each coded block is classified into three types according to the predictor of a block.

With this classification, an efficient algorithm is employed such that the wrong reference problem can be significantly alleviated.

– To exploit the correlations among bitstreams before and after transcoding while maintaining the coding efficiency, we propose the intra mode switching and motion vector re-mapping to eliminate the motion vector estimation and mode decision which are the most computational intensive functional blocks.

– For the blocks that are affected by the drift error, only key blocks are effectively refined rather than all the affected blocks such that the complexity can be further reduced.

– For the blocks that are not affected by the drift error, the original syntax elements are bypassed in the original bitstreams into a VET bitstream to eliminate the re-coding process that is computation-intensive and harmful for transcoded quality.

– As compared to the conventional cascaded pixel-domain transcoder, our algorithm increases the transcoding throughput by 25 times while providing 0.3 ~1.5dB PSNR improvement.

- Chapter 4 demonstrates the problem of architecture design and presents the proposed highly efficient system architecture where hardware efficiency and utilization are significantly increased. In addition, we utilize the top-down design methodology for design space exploration. Our contributions in the system architecture design space exploration include the following:

- A system architecture for H.264/AVC is proposed to combine video embedding transcoder (VET) and decoder.
 - An architectural exploration on synchronization granularity.
 - A design space exploration on design balancing of video pipe.
 - As compared to the traditional hardware design, more system design factor are explored such that the best alternative according to the design constraints can be chosen effectively.
 - As compared to the previous system design, the system performance can be increased by 2 times or reduce the memory requirement to 25% depending on the optimization object. Otherwise, we can obtain the design alternative with highest speed-cost efficiency by tracing the Pareto frontier.
 - The design alternative with highest speed can achieve the transcoding (or decoding) throughput of 1920x1088@60Hz while clocking at 135MHz.
- Chapter 5 details the micro-architecture design for the three hot spot modules within our system architecture. Specifically, the contributions include the following:
 - To efficiently utilize the throughput of external DRAM, a synchronization buffer is employed as a bridge for reformatting the read/write data exchanged between the on-chip hardware and the off-chip DRAM.
 - To increase the efficiency and the utilization, a unified filtering architecture is proposed for the inter and intra prediction that consumes less execution cycle on the average while the hardware cost is relatively low than the state-of-the-art designs. Furthermore, we provide an exploration on the parallelism of systolic based inter and intra prediction.
 - To increase the efficiency of deblocking filter at finer synchronization granularity,

we propose a novel processing order for the deblocking filter that provides fine-grained synchronization capability. In addition, the memory requirement of proposed deblocking filter is lower than macroblock-based deblocking filter.

– As compared to the conventional adder-tree based inter and intra prediction, our systolic-based design increases the transcoding throughput up to 4 times while the amount of data transmission via AHB data bus can be reduced by 65%.

- Chapter 6 summarizes our works and illustrates the research activities in the future.
- Lastly, Appendix A shows why transform domain approaches are inefficient for H.264/AVC based transcoding. Appendix B shows the constant-rate bumping process for displaying the decoded pictures during transcoding.

In summary, this dissertation focuses on the exploration at both algorithm and architecture level. Figure 1.3 illustrates the techniques to be discussed in this work and provides an ordered step by step of the process.



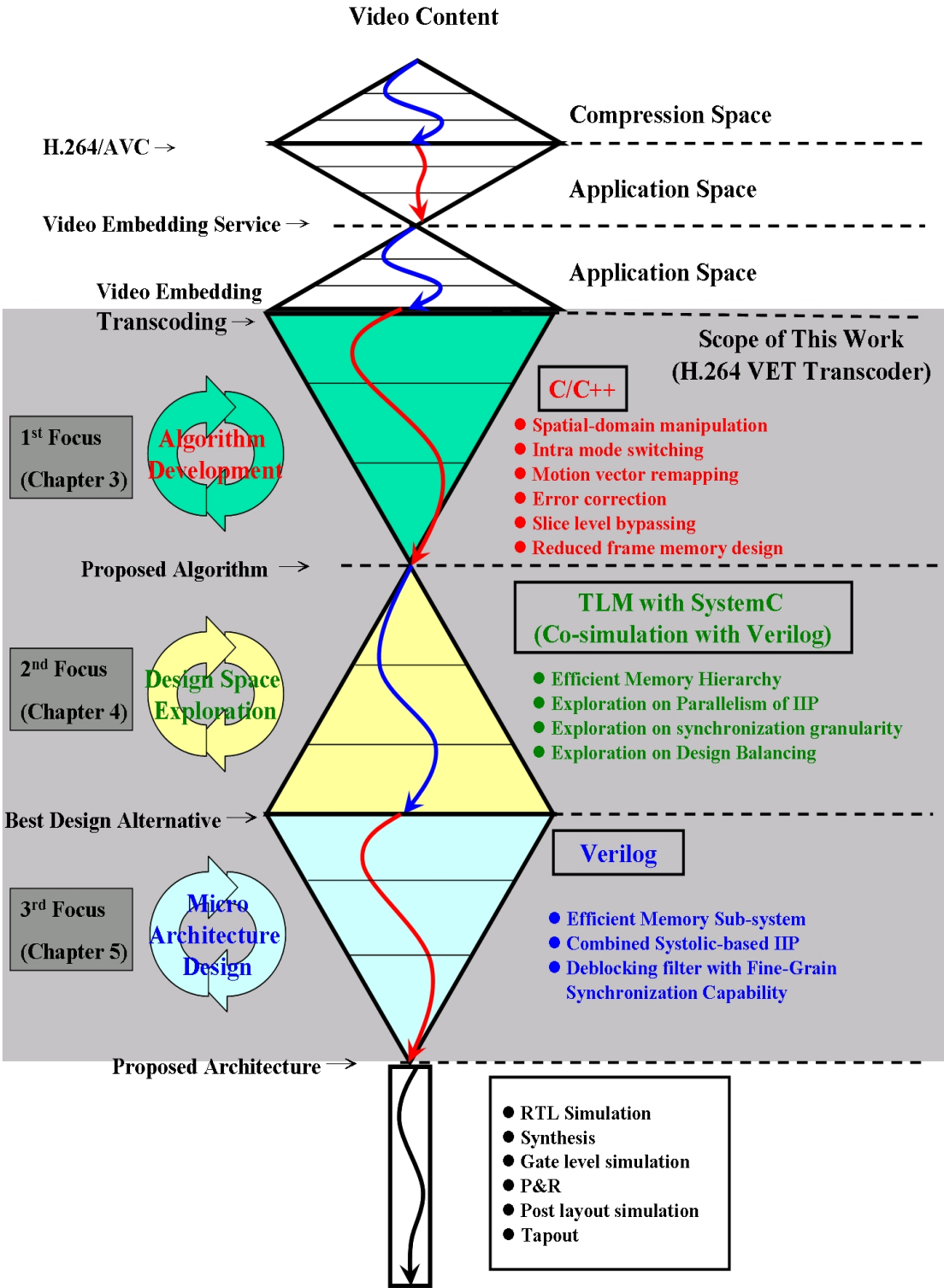


Figure 1.3: The Design Flow in This Dissertation

CHAPTER 2

Background and Related Work on Video Embedding Transcoding



2.1 Introduction

This chapter presents the background and related work on video embedding technique and then shows the challenges in H.264/AVC based video transcoding. Video information embedding technique is essential to several multimedia applications such as picture-in-picture (PIP), multi-channel mosaic, screen-split, pay-per-view, channel browsing, commercials and logo insertion and other visual information embedding services. For video embedding applications, the video embedding transcoding (VET) is essential to deliver multiple-window video services over one

transmission channel. The rest of this chapter is organized as follows: Section 2.2 explains why we realize the video embedding service by video embedding transcoding. Section 2.3 describes the generalized problem of video transcoding. Section 2.4 formulates the wrong reference problem in VET transcoding. Section 2.5 reviews some related works on VET. Section 2.6 elaborates the challenges in H.264/AVC based VET. Lastly, Section 2.7 summarizes this chapter.

2.2 Realization of Video Embedding Service

The video embedding service can be realized at the client side where multiple sets of tuners and video decoders acquire video content of multiple channels to for one frame. The content delivery side sends all the bitstreams of selected channels to the client while the client side reconstructs the pixels with an array of decoders in parallel and then re-composes the pixels into single frame in the pixel domain at the receivers. Each receiver needs N decoders running with a powerful picture composition tool to tile the varying size pictures from N channels. Thus, the overall cost is increased as N is increased. To reduce the cost of the VET service, fast pixel composition and less memory access can be achieved based on the architecture design [9][10][11][12][12]. To realize the VET feature at the client side, the key issues are inefficient bandwidth utilization and high hardware complexity that hinders the multiple-window embedding applications deployment.

To increase the bandwidth efficiency of bitstream transmission and reduce hardware complexity at client side, the video embedding service can be alternatively realized at the server/studio side to deliver selected video contents that are encapsulated as one bitstream. To transmit the video contents via the unitary channel, the MW-VET embeds downsized video frames into another frame with a specified resolution as the foreground areas. It can provide preview frames

or thumbnail frames by tiling a two-dimensional array of video frames from multiple television channels simultaneously. With the MW-VET, users can acquire multiple-channel video contents simultaneously. Moreover, the MW-VET bitstreams are compliant to H.264/AVC and it can facilitate the multiple-window video playback in a way transparent to the decoder at the client side.

The challenges are to simultaneously maintain the best picture quality after transcoding, to increase the picture insertion flexibility, to minimize the archival space of bitstreams and to reduce computational complexity. To optimize rate-distortion (R-D) performance, the bits of the newly covered blocks at the background picture are replaced by the bits of the blocks at the foreground pictures. To increase the flexibility of picture insertion, the foreground pictures are inserted at the macroblock boundaries of processing units. To minimize the bitstream storage space, H.264/AVC coding standard is adopted as the target format. To decrease the computational complexity, a low-complexity algorithm for composition is needed. Therefore, we proposed a fast H.264/AVC based multiple-window VET (MW-VET), which encapsulates on-the-fly multiple channels of video content with a set of pre-compressed bitstreams into one bitstream before transmission.

For real-time applications, video transcoding should retain R-D performance with the lowest complexity, minimal delay and the smallest memory requirement [13]. Particularly, the MW-VET should maintain good quality after multi-generation transcoding that may aggravate the quality degradation. An efficient VET transcoder is critical to address the issue of quality loss.

2.3 Problem Statement of Video Transcoding

Generally, transcoding process could be viewed as the modification process of incoming residue according to the changes in the prediction. As shown in Figure 2.1 (a), the output of transcoding

is represented by

$$\begin{aligned}
 \overline{r'_n} &= Q [HT (\overline{r'_n})] = Q \{HT [\overline{x_n} - PRED_2 (\overline{x_j})]\} \\
 &= Q \{HT [\overline{r_n} + PRED_1 (\overline{x_i}) - PRED_2 (\overline{x_j})]\}
 \end{aligned} \tag{2.1}$$

, where the symbols HT and Q indicate an integer transformation and quantization respectively. The symbols $\overline{r_n}$ and r'_n denote the residue before and after the transcoding. The symbols $PRED_1 (\overline{x_i})$ and $PRED_2 (\overline{x_j})$ represent the predictions from the reference data $\overline{x_i}$ and $\overline{x_j}$ respectively. In this paper, we use the symbol “bar” above the variables to denote the reconstructed values after decoding and the symbol “prime” to denote the refined values after transcoding. The suffix of each variable represents the index of block. The process to embed the foreground videos onto the background can incur drift error in the prediction loop since the reference frames at the decoder and the encoder are not synchronized.

When the predictions before and after the transcoding are identical, Figure 2.1 (a) can be simplified to Figure 2.1 (b). The quantized data $\overline{r_n}$ has no further quantization distortion with the same quantization step. Thus, the transcoded bitstream has almost identical R-D performance with the original bitstream as represented in Eq.(2.2).

$$P_d \cdot P_e \cdot \overline{r_n} = IHT \{DQ \{Q [HT (\overline{r_n})]\}\} = \overline{r_n} \tag{2.2}$$

, where the symbol P_e denotes the encoding process from the pixel domain to the transform domain. The symbol P_d denotes the decoding process from the transform domain back to the pixel domain. The symbols IHT and DQ mean an inverse integer transformation and de-quantization respectively.

By Eq.(2.2), the transcoding process in Figure 2.1 (b) can be further simplified to that in Fig-

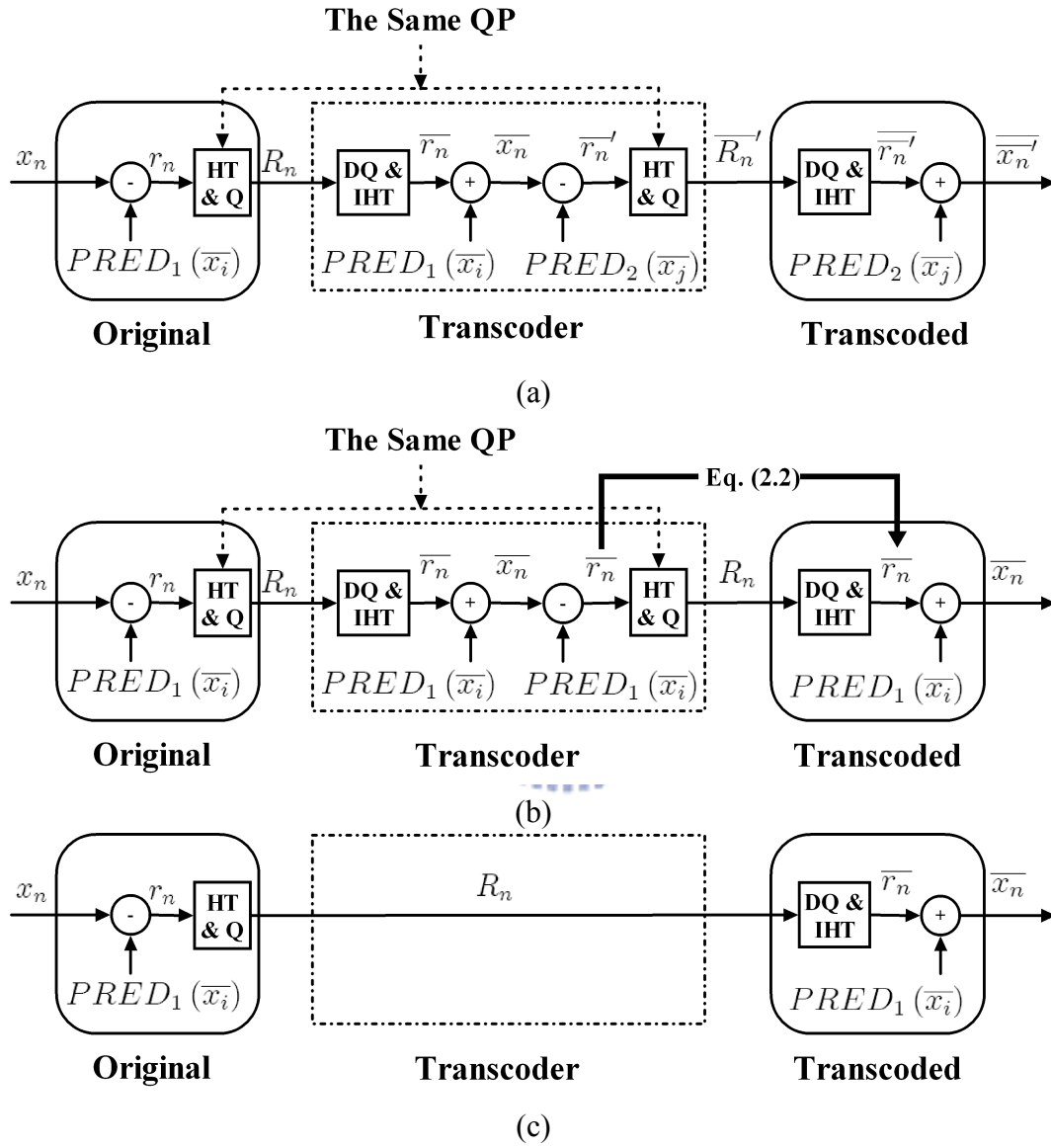


Figure 2.1: Illustration of a Novel Transcoder: (a) The Simplified Transcoding Process. (b) The Simplified Transcoder When the Prediction Blocks Are the Same. (c) The Fast Transcoder That Bypasses the Input Transform Coefficients.

ure 2.1 (c) where the data of the original bitstreams can be bypassed without any modification. It leads to a transcoding scheme with the highest R-D performance and the lowest complexity.

Video transcoding is intended to maximize R-D performance with the lowest complexity. Therefore, the remaining issue is to transcode efficiently the incoming data such that picture quality is maximized with the lowest complexity. Specifically, the incoming data are refined only when the reference pixels are modified to alleviate the propagation error. To reduce computational cycles and preserve picture quality, the residue data with identical reference pixels are bypassed.

2.4 Wrong Reference Problem Formulation

Based on the occurrence of modified reference pixels and the paths of error propagation at the prediction loop, the MBs are classified into three types: w -MB, p -MB and n -MB. As shown in Figure 2.2, the small rectangle denotes the foreground picture (FG) and the large rectangle denotes the background picture (BG). Each small square within the rectangle represents one MB. The w -MBs represent the blocks whose reference samples are entirely or partially replaced by the newly inserted pictures. The p -MBs represent the blocks whose reference pixels are composed of the pixels at w -MBs. The remaining MBs of the background pictures are denoted as n -MBs for the un-affected MBs. We observe that most of the MBs within the processing picture are p -MBs and only a small percentage of MBs are w -MBs. As for w -MBs, the coding modes or motion vectors of the original bitstream are modified to fix the wrong reference problem. For the p -MBs, the wrong reference problem is inherited from the w -MBs. Thus, the coding modes and motion vectors are better to be refined for each p -MB. All n -MBs' information in the original bitstream can be bypassed because the predictors before and after transcoding are identical.

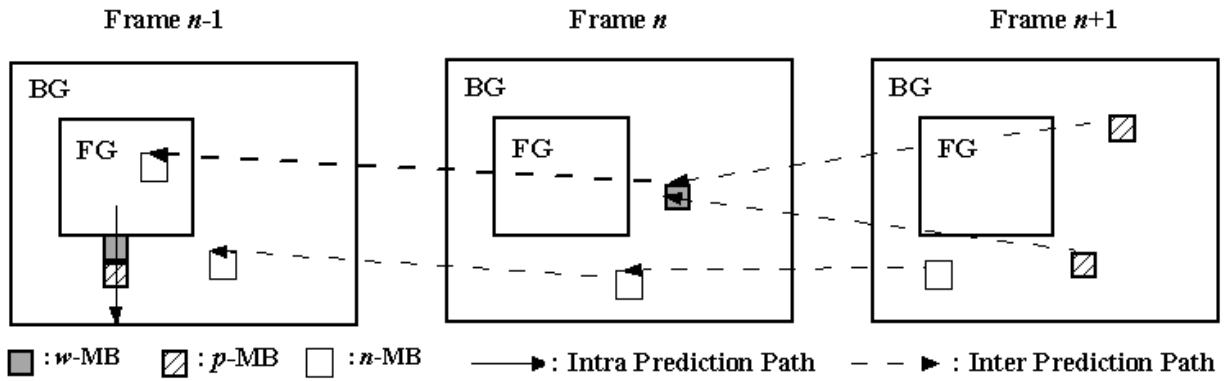


Figure 2.2: Illustration of the Wrong Reference Problem

2.5 Related Work on Video Embedding Transcoding

Depending on the operating domain, the transcoders can be classified as either pixel domain or transform domain approaches. In the following, we will review the most straightforward implementation of VET transcoder and two fast VET transcoding algorithms for MPEG-2 standard.

2.5.1 Cascaded Pixel Domain Transcoder (CPDT)

The cascaded pixel domain transcoder (CPDT) cascades multiple decoders, a pixel domain composer and an encoder, as shown in Figure 2.3. It decompresses multiple bitstreams, composes the decoded pixels into one picture, and re-compresses the picture into a new bitstream. It offers drift free performance since the exhaustive re-encoding process of CPDT can avoid drift errors from propagating to the whole group of pictures.

However, the CPDT suffers from noticeable visual quality degradation and high complexity. Specifically, the re-quantization process decreases quality of the original bitstreams. The irretrievable quality degradation exacerbates especially when the foreground pictures are inserted at different time using the CPDT with multiple iterations. In addition, the cost of a CPDT is relatively high at both algorithm and architecture level since the computation-intensive re-encoding

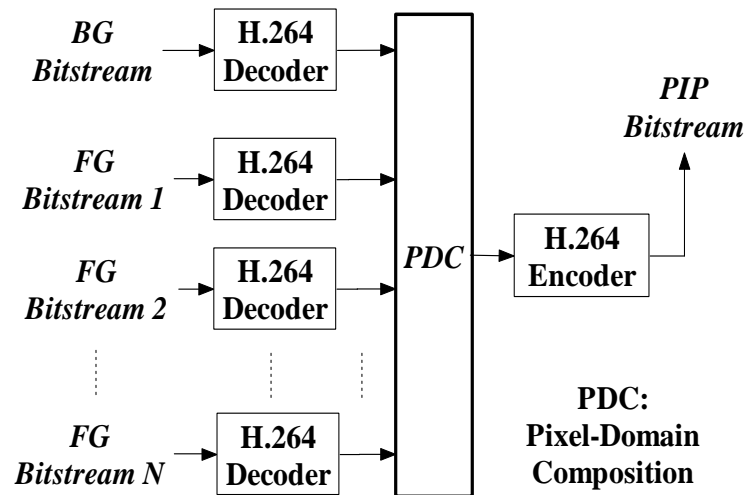


Figure 2.3: The Architecture of the CPDT

process makes the significant complexity increase of the CPDT too costly for real-time video content delivery. From the video compression perspective, the CPDT is not efficient because the existing correlations between input and output bitstream are not utilized at all. The complexity and memory requirement of the CPDT could be reduced with fast algorithms that exploit the correlations to remove transformation and motion estimation.

2.5.2 DCT Domain Transcoding with Motion Vector Re-mapping

The discrete cosine transform (DCT) domain inverse motion compensation (IMC) approach, which is proposed by Chang et al. [14][15][16], contributes to the DCT domain transcoding for MPEG-2 standard. The matrix translation manipulations are used to synthesis a DCT block that is not aligned to the boundaries of 8x8 blocks in the DCT domain. The DCT domain IMC takes less complexity than forward and inverse transform such that the transcoder can manipulate the bitstreams in the DCT domain to avoid transform back and forth. Chang's approach could achieve 10% to 30% speed-up over the CPDT. There are other algorithms to speed up the DCT domain IMC in [17][18][19].

The motion estimation can be eliminated with motion vector re-mapping (MVR) where the new motion vectors are obtained by examining only two most likely candidate motion vectors located at the edges outside the foreground picture. It simplifies the re-encoding process with negligible picture quality degradation. In addition, Chang refines the residue of w -MBs and p -MBs to correct all the drift error caused by refinement error.

2.5.3 DCT Domain Transcoding with Backtracking

A DCT domain transcoder based on a backtracking process is proposed by Yu et al. [20] to further improve the transcoding throughput. The backtracking process finds the affected macroblocks (MBs) of the background pictures in the motion prediction loop. Since only a small percentage of the MBs at the background are affected, only the damaged MBs are fixed and the unaffected MBs are bypassed.

In practice, for most effective backtracking, the future motion prediction path of each affected MB needs to be analyzed and stored in advance. To construct the motion prediction chains, Chang [14][15][16] completely reconstructs all the refined reference frames in the DCT domain for each group-of-picture (GOP). With the motion prediction chains, the transcoder decodes minimum number of MBs to render the correct video contents. The speedup of motion compensation is up to 90% at the cost of the buffering delay of the transcoder for one GOP period. The impact of the delay on the real-time applications depends on the length of a GOP in the original bitstream.

2.6 The Challenge in H.264/AVC-based PIP Transcoding

As compared to the previous standards, the H.264/AVC poses more challenges for video embedding transcoding as follows:

1. *Inference mismatch*: Since the mode and motion information of current block is inferred from the neighboring blocks, the foreground insertion will incur the inference mismatch and induce serious visual artifacts. Therefore, the syntax elements in the original bitstreams should be fully decoded, re-inferred, and re-encoded into the VET bitstream by an entropy encoder.
2. *More sophisticated wrong reference problem*: Due to the advanced prediction tools such as the variable block size, the various intra prediction modes and improved skip mode, the wrong reference problem becomes more complicated.
3. *More complicated mode decision at the re-encoder side*: Due to the variable block size and the various intra prediction modes, the mode decision at the re-encoder becomes more complicated.
4. *More sophisticated motion vector re-mapping*: When performing motion vector re-mapping technique, the H.264/AVC based VET transcoder should consider the impacts of 6-tap interpolation and the deblocking filter across the boundary between foreground and background picture.
5. *Transform-domain (HT-domain) IMC is inefficient*: The existing approaches [17][18][19][20] convert the bitstreams that are of MPEG-2 standard in the transform domain for complexity reduction. However, application of the transform domain techniques to H.264/AVC is not feasible since the advanced coding tools including in-the-loop de-blocking filter, directional intra prediction and 6-tap sub-pixel interpolation all operate in the pixel domain. The transform domain inverse motion compensation becomes inefficient when the motion compensation uses quarter-pixel resolution combined with 6-tap interpolation. In addition, the transformation and quantization processes in H.264/AVC are so optimized that traverse back to the pixel domain is not as expensive as before. Consequently, the

transform domain techniques have higher complexity as compared to the pixel domain techniques. As shown in the Appendix the pixel domain transcoder actually takes less complexity than the transform domain transcoder. The detail derivations are given in the Appendix A for brevity.

6. *Transform-domain (HT-domain) operations cause the drift error*: The transform domain manipulation introduces drift since the motion compensation is based on the filtered pixels which are the output of the in-the-loop deblocking filter. The filtering operation is defined in the pixel domain and cannot be performed in the transform domain due to its nonlinear operations [21][22][23]. In addition, the transform domain transcoding requires an inverse quantization process that introduces additional rounding error due to the operation of division. As a result, the transform domain transcoder for the H.264/AVC standard typically leads to an unacceptable level of error as shown in [24].

7. *Backtracking method has slight benefit while introducing a delay of one GOP*: The backtracking method proposed by Yu [20] has no use for the H.264/AVC based transcoder due to the deblocking filter, the directional intra prediction and interpolation filter. Particularly, to track the prediction paths of H.264/AVC bitstreams, almost 100% of the blocks need decoding, which is over the 10% reported in [20]. Thus, the expected complexity reduction is limited. Furthermore, it introduces an extra delay of one GOP period.

In summary, to speed up the CPDT, there are many fast algorithms to manipulate the incoming bitstreams in the transform domain. However, this is not the case for the H.264/AVC standard. To our best knowledge, all the state-of-the-art transcoding schemes with H.264 as input bitstream format perform the fast algorithms in the pixel domain [21][25][22][23][26][27][28][29][30].

2.7 Summary

In this chapter, we present the background, related work, and the challenges in H.264/AVC based video embedding technique. First of all, we realize the video embedding service by video embedding transcoding so as to (1) increase the information acquisition, (2) lower the bandwidth requirement, (3) reduce the consuming power of client's devices, (4) provide H.264/AVC-compliant bitstreams, and (5) minimize the storage space of video archive. Second, we formulate the wrong reference problem based on the occurrence of modified reference pixels and the paths of error propagation at the prediction loop. With this formulation, we are roughly aware of the problems in the video embedding transcoding. Third, we review the related work on fast transcoding algorithm for MPEG-2. As compared to the previous standards, the H.264/AVC poses more challenges for video embedding transcoding. Lastly, Section 2.6 elaborates several reasons to manifest the necessity of pixel domain manipulation for the H.264/AVC bitstreams. Therefore, we conclude that the spatial domain technique is a more realistic approach for H.264/AVC based transcoding. To resolve issues of low computational cost, less drift error and small memory bandwidth, we will develop the fast algorithm of a H.264/AVC-based video embedding transcoding in the spatial domain.

CHAPTER 3

Low-Complexity Algorithm of MW-VET



3.1 Introduction

This chapter proposes three low-complexity algorithm schemes of H.264/AVC multiple-window video embedding transcoder (MW-VET) for various interactive and non-interactive applications that require video embedding services including picture-in-picture (PIP), multi-channel mosaic, screen-split, pay-per-view, channel browsing, commercials and logo insertion, and other visual information embedding services. In particular, the MW-VET embeds multiple foreground pictures that are of smaller spatial resolution at macroblock-aligned positions. As the foreground bitstreams are encoded as full resolution, a downsizing transcoding [21][31][25] is needed prior to the VET transcoding. The spatial resolution adaptation transcoding has been widely inves-

tigated in the literatures and are not studied herein. In addition, we impose restrictions on the foreground bitstreams to remove the dependencies to the background. In particular, the foreground bitstream do not use un-restricted motion vectors and DC intra mode for the blocks at the first column or the first row of the foreground. The loss of R-D performance is negligible. In addition, we re-scale the DC coefficient of the first DC block within an intra-coded frame based on the reconstructed values of neighboring pixels in the background. Except for the first block, the foreground bitstream can be directly inserted into the new one.

According to the type of data partition and the encoding scenario of the archived bitstream, the MW-VET adopts one of three algorithm schemes including (1) slice group based transcoding (SGT), (2) no frame memory transcoding (NFMT), and (3) reduced frame memory transcoding (RFMT). As the prediction is applied to slice-aligned data partitions within the original bitstream, the SGT simply merges the bitstreams into VET bitstream by parsing and concatenating the syntax elements and provide the highest transcoding throughput. As the prediction is applied to the region-aligned data partitions and a corresponding auxiliary bitstream is available, the NFMT can compose VET bitstream by parsing, patching and concatenating the syntax elements from the original bitstreams and the auxiliary bitstream. If there is neither slice-aligned data partitions within the original bitstream nor an auxiliary bitstream, the RFMT can efficiently transcode the input bitstreams by three block level adaptive techniques based on the concept of partial re-encoding. The application of each transcoding scheme depends on the data partitions and encoding scenario of the archived bitstreams. Particularly, both the SGT and the NFMT are serviceable only in some specific conditions thus restricting their application. Therefore, we focus on the algorithm of the RFMT in this chapter.

To maintain transcoded picture quality and to reduce the overall complexity, we present three transcoding techniques in RFMT: (1) intra mode switching (IMS), (2) motion vector re-

mapping (MVR), and (3) syntax level bypassing (SLB). For region-aligned data partitions, the RFMT efficiently refines the prediction mismatch so as to adapt the prediction schemes compliant with the H.264/AVC standard and to increase the transcoding throughput while maintaining better R-D performance. Specifically, when the prediction comes from the newly covered area without slice-group data partitions, the pixels at the affected macroblocks are transcoded with the RFMT based on the concept of partial re-encoding to minimize the number of refined blocks. The RFMT employs intra mode switching (IMS) and motion vector remapping (MVR) to handle intra coded blocks and inter coded blocks respectively. For the pixels outside the macroblocks that are affected by newly covered reference frame, the SLB de-multiplexes and multiplexes the bitstreams into a VET bitstream at the bitstream level. Experimental results show that, as compared to the cascaded pixel domain transcoder (CPDT) with the highest complexity, our RFMT can significantly reduce the processing complexity by 25 times and retain the rate-distortion performance close to the CPDT. At certain bit rates, the RFMT can achieve up to 1.5 dB quality improvement in Peak-Signal-to-Noise-Ratio (PSNR).

In Table 3.1, we list all the symbol definitions used in this chapter. The rest of this chapter is organized as follows: Section 3.2 and Section 3.3 presents the slice group based transcoding (SGT) and no frame memory transcoding (NFMT). Section 3.4 proposes the reduced frame memory transcoding (RFMT). Section 3.5 shows the simulation results. Finally, Section 3.6 summaries this chapter.

3.2 Slice-Group-Based Transcoding

The slice group based transcoding (SGT) is used when the prediction within the original bitstream of background picture uses the slice-aligned data partitions [31]. Based on the slice-aligned data partitions, the SGT operates at the bitstream level to provide the highest through-

Table 3.1: The Symbol Definitions

Symbol	Meaning
CAVLD	Content adaptive variable length decoding
CAVLC	Content adaptive variable length coding
LB	Line buffer
FM	Frame memory
DB	De-blocking filter
IP	Intra prediction
MC	Motion compensation
ME	Motion estimation
HT & Q	Integer transform and quantization
DQ & IHT	De-quantization and inverse integer transform
PDC	Pixel domain composition
RDO MD	Rate-distortion optimized mode decision
MUX	Multiplexer (syntax element selector)

put with the lowest complexity. The rationale is that H.264/AVC defines a set of MBs to the slice group map types according to the adaptive data partition [6]. The concept of slice group is to separate the picture into isolated regions to prevent error propagation from leading error resiliency and random access. Each slice is regarded as an isolated region as defined in H.264/AVC standard. For each region, the encoder performs the prediction and filtering processes without referring to the pixels of the other regions.

For the video embedding feature using static slice groups, the large window denotes a background slice and the embedded small windows denote foreground slices. For example, a frame can be split into: a background slice and several rectangular foreground slices as shown in Figure 3.1. After video embedding transcoding, all the slices are encoded separately at the slice level and encapsulated to one bitstream at the slice level. Based on archived H.264/AVC bitstreams with the slice groups, a VET can replace the syntax elements of MBs in the foreground slices with the syntax elements of other bitstreams with identical spatial resolutions. Therefore, all the syntax elements are directly forwarded as-is to the final bitstream via an entropy coder.

Based on the pre-defined slice groups, the SGT is effective for non-interactive applications

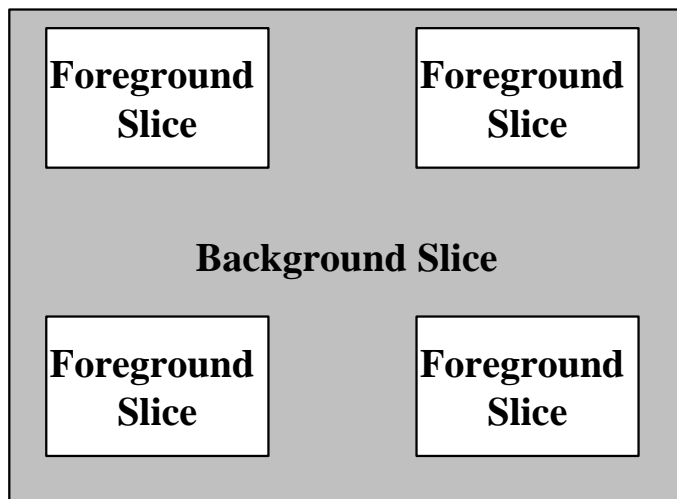
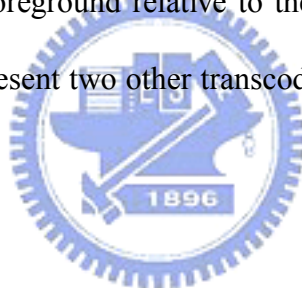


Figure 3.1: The Example of Slice Group That Can Be Used in the VET Transcoding

with multiple static windows. However, for interactive services, the SGT can not allow the change of location and size of the foreground relative to the background. To further enable movement of the window, we will present two other transcoding schemes in the next two sections.



3.3 No Frame Memory Transcoding (NFMT)

3.3.1 Architecture of NFMT

For interactive services, we propose a PIP transcoding scheme using an auxiliary bitstream, which we refer to it as the no reference frame transcoding (NRFT) approach. To achieve best coding efficiency at low complexity, the auxiliary bitstream is computed in such a way that this bitstream can substitute all the MBs that have wrong prediction when the foreground area is overwritten with the new content. Thus, the NRFT approach directly substitutes syntax elements of the affected blocks with the auxiliary bitstream and constitutes a simple frame composition in the transform domain. The channel bandwidth is efficiently used with only minimum

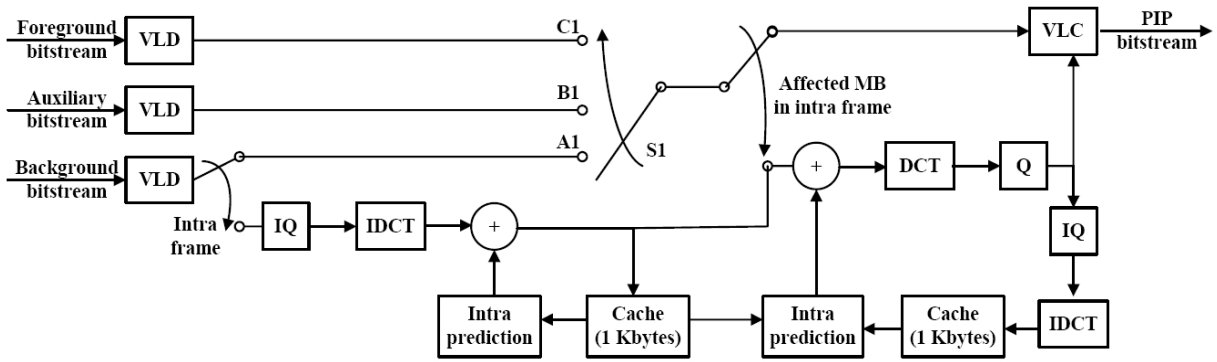


Figure 3.2: The Architecture of the No Frame Memory Transcoder (NFMT)

hardware. However, such a simple bitstream switching scheme requires additional storage for the auxiliary bitstreams.

NFMT composes a VET bitstream on transform domain based on an auxiliary bitstream as shown in Figure 3.2. The auxiliary bitstream is generated along with the primary bitstream to provide the syntax elements including the transform coefficients and motion information for a rapid frame composite. To balance the transcoding speed and the Rate-Distortion (R-D) performance, the PIP transcoding with the auxiliary bitstream is applied to all inter-coded frames and the partially re-encoding transcoding is used for the intra-coded frames. By removing the high complexity modules including spatial interpolation, motion compensation and inverse spatial prediction from the transcoding process of all inter-coded frames, the NFMT can significantly increase the transcoding throughput with very low memory requirement.

As compared to the CPDT, for all inter-coded frames, the proposed NRFT approach can simultaneously reduce execution time and memory by eliminating complex modules such as intra prediction, sub-pel interpolation and in-the-loop deblocking filter. However, NRFT approach suffers from error drift when using prediction residuals from the auxiliary bitstream. Our experimental results show that the error drift is not serious for sequences using periodic intra-frames with GOP size of 15 or 30.

To provide a balanced tradeoff of coding efficiency and complexity, the intra-coded frames does not use the auxiliary bitstream because the intra-frame mismatch can propagate to subsequent MBs spatially and temporally. To retain the R-D performance, the affected MBs of intra frames are transcoded using the IMS approach in the next section.

For the transcoding of inter-coded frames, all three bitstreams are first passed through VLD module to extract the syntax elements. With the original and the auxiliary bitstreams, the syntax elements that can retain the better R-D performance are put into the transcoded bitstream in the NRFT approach. For each affected MB in an inter-coded frame, we change the residual coefficients of affected 4x4 blocks with the syntax elements of the auxiliary bitstream. With the selected syntax elements, we can start to re-encode the PIP bitstream using VLD.

3.3.2 Auxiliary Bitstream Generation

The auxiliary bitstream generation is shown in Figure 3.3 where we use the reconstructed frames as the reference for prediction. Since each MB may contain multiple objects, we use a 4x4 block as the basic processing unit so that drift can be minimized.

In order to simplify the switching, we need to place restrictions on the formation of predictions of the auxiliary bit stream. To ensure that the patched MB of the auxiliary bitstream does not refer to the MBs in the unexpected region, we restrict all MBs to use inter4x4 mode with zero motion vector. To minimize error accumulation, the auxiliary bitstream is encoded using identical reconstructed coefficients of the background bitstream.

3.4 Reduced Frame Memory Transcoding (RFMT)

When the prediction is applied to the region-aligned data partitions, the specified pixels at the background picture are replaced by the pixels of the foreground pictures. For the pixels in

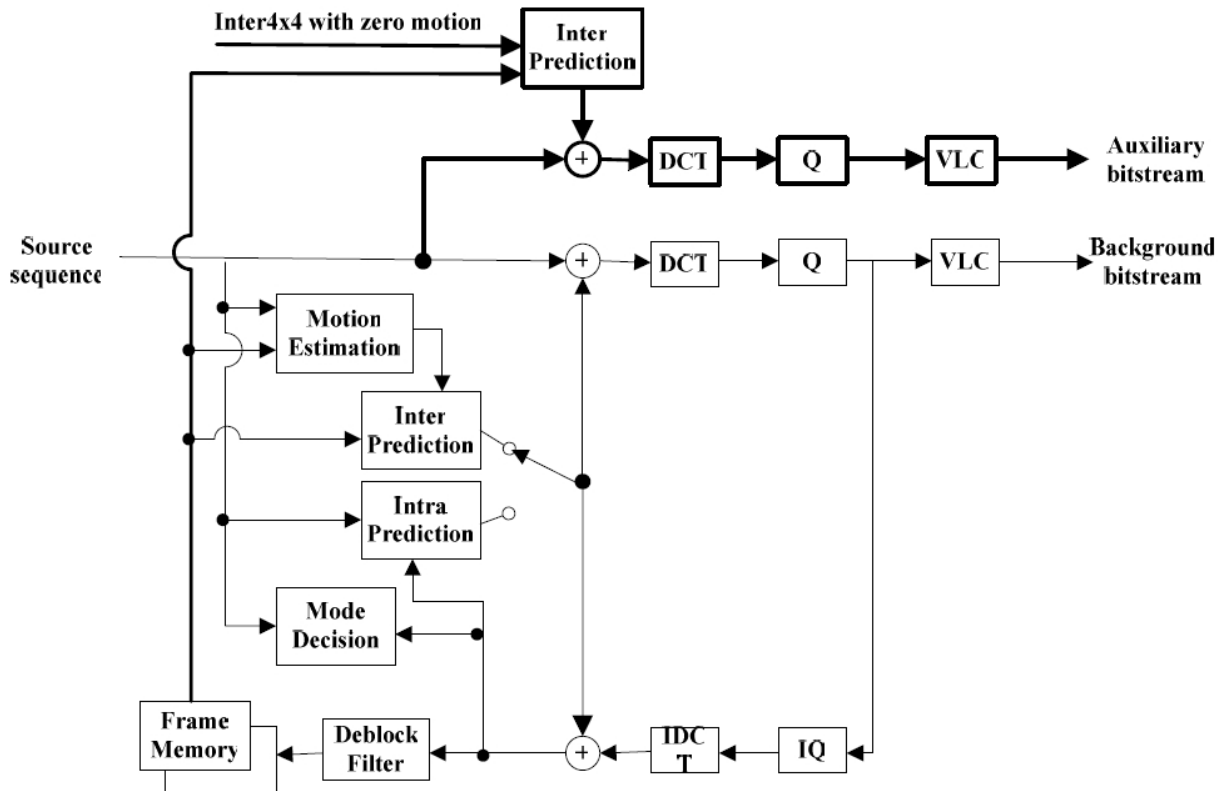


Figure 3.3: The Generation of the Auxiliary Bitstream Based on the Reconstruction of a RDO Encoder

the affected MBs, the RFMT can minimize the total number of refined blocks by partially re-encoding only those MBs. The RFMT employs both motion vector remapping (MVR) for inter coded blocks and intra mode switching (IMS) for intra coded blocks respectively. The pixels within the un-affected MBs are transcoded by the SLB that passes the syntax elements from the original bitstreams to the transcoded bitstream.

Based on the partially re-encoding techniques, the initial RFMT architecture is shown in Figure 3.4. After decoding all the bitstreams into pixel domain with multiple H.264/AVC decoders and composing all the decoded pictures into one frame by the PDC, the re-encoder side only refines the residue of the affected MBs rather than re-encoding all the decoded pixels as the CPDT architecture. For those unaffected MBs, the syntax elements are bypassed from each CAVLD and are sent to the MUX which selects the corresponding syntax elements based on

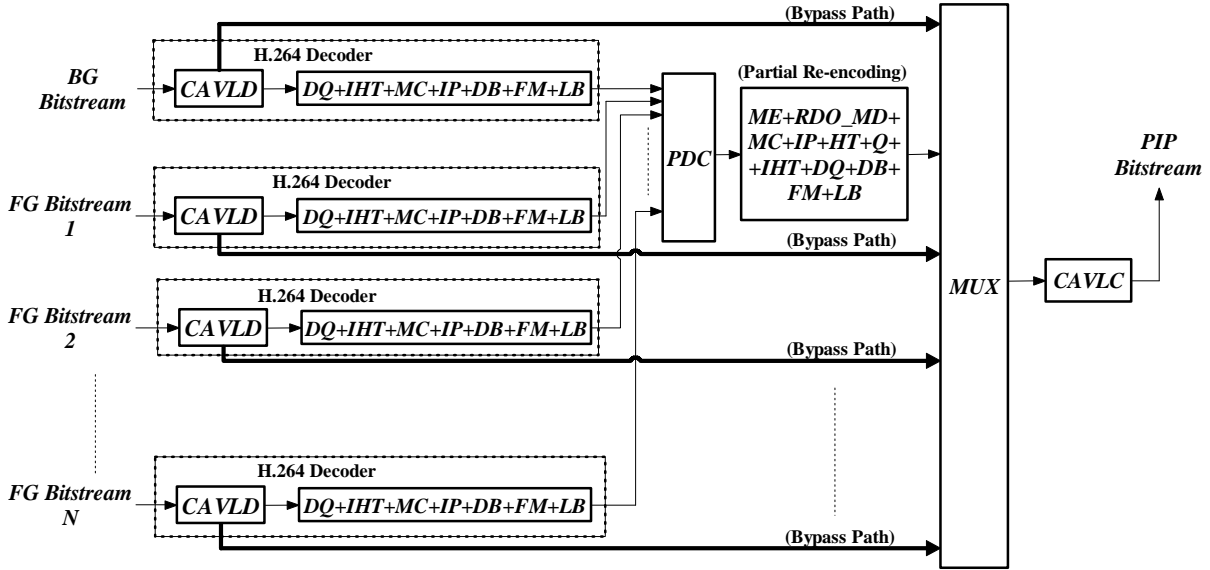


Figure 3.4: The Initial Architecture of the RFMT with RDO Refinement Based on the Concept of the Partially Re-Encoding

the PIP scenario. Lastly, the CAVLC encapsulates all the re-used syntax elements and the new syntax elements of refined blocks into the transcoded bitstream.

To increase the throughput, the R-D optimized mode decision and motion vector re-estimation within the re-encoder side of Figure 3.4 are replaced with the intra mode switching (IMS) and motion vector re-mapping (MVR) as shown in Figure 3.5 [32]. Specifically, the re-encoder as enclosed by the dashed line stores the decoded pixels into the FM. Then, the MVR and IMS modules retrieve the intra modes and the motion vectors from the original bitstreams to predict the characteristics of motion and the spatial correlation of the source. With such information, we examine only a subset of possible motion vectors and intra modes to speed up the refinement process. According to the refined motion vectors and coding modes, the MC and IP modules perform motion compensation and intra prediction from the data in the FB and LB. The reconstruction loop including HT, Q, DQ, IHT, and DB generates the reconstructed data of the refined blocks which are further stored in the FB to avoid the drift during the transcoding. In conclusion, other than the IMS and MVR modules all the modules in Figure 3.5 are the same

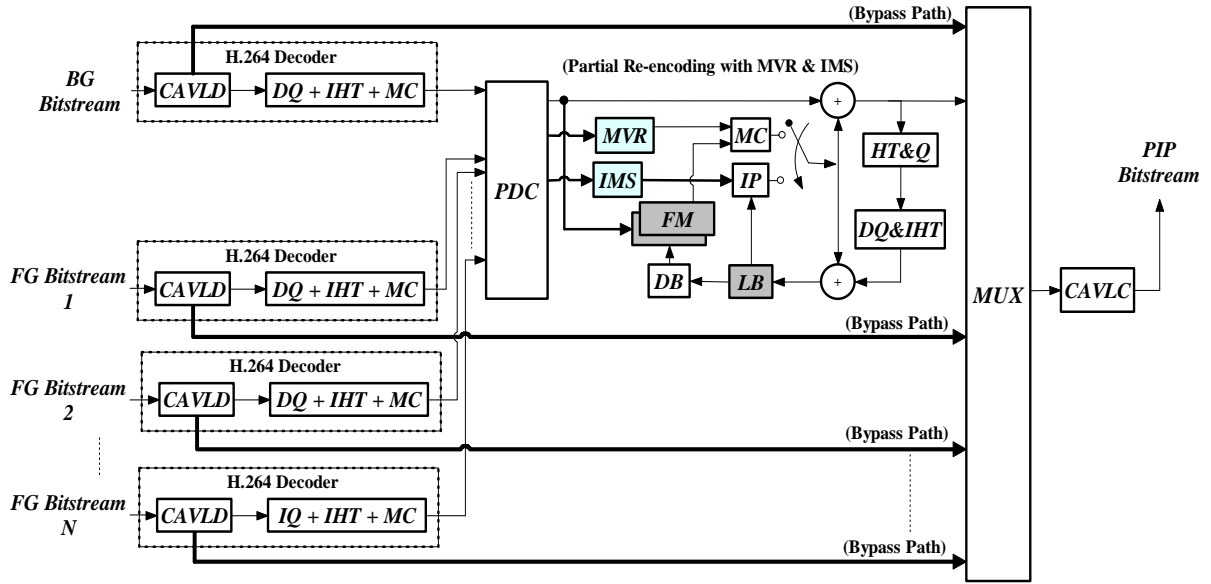


Figure 3.5: The Intermediate Architecture of the RFMT with the MVR and the IMS Refinement

as those in Figure 3.4.

To de-couple the dependency between the foreground and the background, there is an encoding constraint for the foreground bitstream that the unrestricted motion vectors and the intra DC modes are not used for the blocks at the first column or the first row. When the foreground video is from an archived bitstream or an encoder of live video, the unrestricted motion vectors and the intra DC mode can be modified and the loss of R-D performance is negligible according to our experiment. Particularly, we re-scale the DC coefficient of the first DC block within an intra coded frame based on the neighboring reconstructed pixels in the background. Except the first block, the foreground bitstreams can be multiplexed directly into the transcoded bitstream.

With the constrained foreground bitstreams, the final architecture of the MW-VET is simplified as shown in Figure 3.6. The highly efficient MW-VET adopts only the content adaptive variable length decoding (CAVLD) for the foreground bitstreams and uses one shared frame memory for the background bitstream. At first, two frame memories are dedicated for the decoder and the re-encoder in Figure 3.5 to store the decoded pixels and the reconstructed pixels

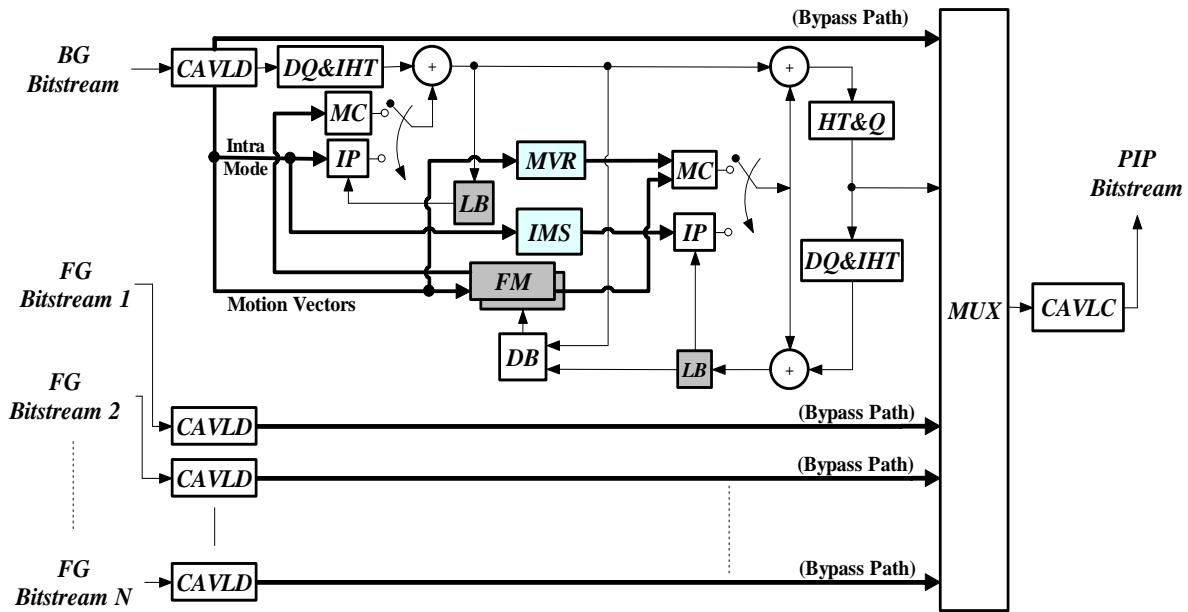


Figure 3.6: The Final architecture of the RFMT with Shared Frame Memory and the Constrained FG Bitstreams

respectively. However, the decoded data of affected blocks are no longer useful and could be replaced with the reconstructed pixels after the refinement. Therefore, we use a shared frame memory to buffer the reference pixels for both the decoding and re-encoding process. Specifically, the operation of the transcoder begins with the decoding by the CAVLD. The MC and the IP modules in the left hand side use the original motion vectors and intra modes to decode the source bitstream into pixels stored in the FM and used for the coefficient refinement. On the other hands, the MC and the IP modules in the right hand side use the refined motion vectors and intra modes to refine the decoded pixels of the affected blocks. In addition to one shared FM, the transcoding process is the same as that in Figure 3.5.

In case the PIP scenario generates the background block with top and left pixels next to the foreground pictures, our RFMT needs to decode each foreground bitstreams. Then, the transcoder switches the mode of this block to DC mode and computes the new residue according to the reconstructed values of two foreground pictures. Moreover, if the foreground pictures

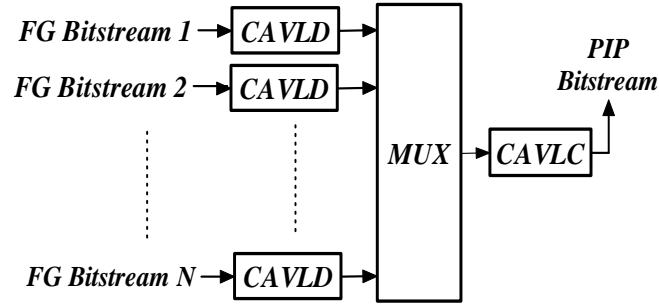


Figure 3.7: The Transcoding Scheme for the Channel Preview

occupy the whole frame, the feature of channel preview is realized with the degenerated architecture of Figure 3.7. The remaining issues are how the IMS and the MVR modules deal with the wrong reference problem of background bitstream. There are two goals: refining the affected blocks efficiently and deciding the minimal subset of refined block while retaining the visual quality of transcoded bitstream.

3.4.1 Intra Mode Switching (IMS)

For the intra-coded w -MBs, we need to change the intra modes to fix the wrong reference problem since the intra prediction is performed in the spatial domain. The neighboring samples of the already encoded blocks are used as the prediction reference. Thus, when we replace parts of the background picture with the foreground pixels, the MBs around the borders may have visual artifacts due to the newly inserted samples. Without drift error correction, the distortion propagates spatially all over the whole frame via the intra prediction process in a raster scanning order. A straightforward refinement approach is to apply the R-D optimized (RDO) mode decision to find the best intra mode from the available pixels and then re-encode new residue.

To reduce complexity we propose an intra mode switching (IMS) technique for the intra-coded w -MBs since the best reference pixels should come from the same region. The mode

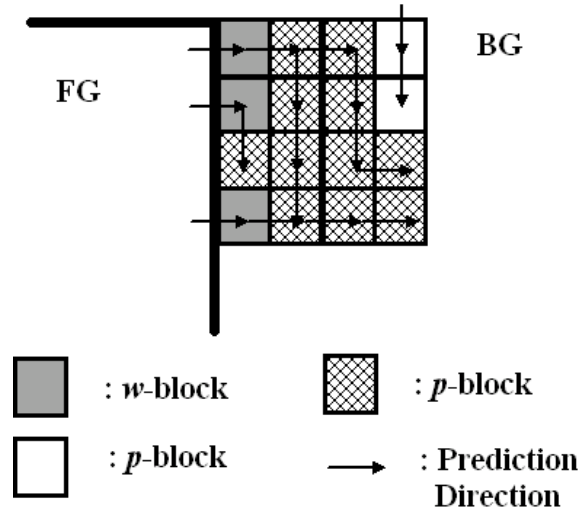


Figure 3.8: The Wrong Intra Reference Problem within a Macroblock Depending on the Intra Modes

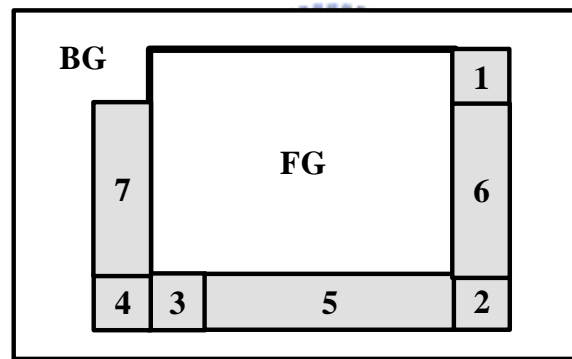


Figure 3.9: The Relative Position of Each Case in the Intra Mode Switching Technique

switching approach selects the best mode from the more probable intra prediction modes.

Each block within a MB could be classified according to the intra modes as shown in Figure 3.8. Similarly, the mode of the w -block should be refined while the modes of p -blocks are unchanged. For the w -blocks, the IMS is performed according to the relative position with respect to the foreground pictures as shown in Figure 3.9. To speed up the IMS process, a table lookup method is used to select the new intra mode according to the original intra mode and the relative position. Table 3.2 and Table 3.3 enumerate the IMS selection exhaustively.

Table 3.2: The Cases of the Intra4 Mode Switching

Case	Corresponding 4x4 block	Original Mode*	Switched Mode*
1	Left column of blocks	1, 2, 4, 5, 6, 8	0
2	Top left of block	4, 5, 6	2
3	Top row of blocks	0, 2, 3, 4, 5, 6, 7	1
4	Top right of block	3, 7	0
5	Top row of blocks	0, 2, 3, 4, 5, 6, 7	1
6	Left column of blocks	1, 2, 4, 5, 6, 8	0
7	Right column of blocks	3, 7	0

- * 0: Intra_4x4_Vertical
 1: Intra_4x4_Horizontal
 2: Intra_4x4_DC
 3: Intra_4x4_Diagonal_Down_Left
 4: Intra_4x4_Diagonal_Down_Right
 5: Intra_4x4_Vertical_Right
 6: Intra_4x4_Horizontal_Down
 7: Intra_4x4_Vertical_Left
 8: Intra_4x4_Horizontal_Up

Table 3.3: The Cases of the Intra16 Mode Switching

Case	Original Mode*	Switched Mode*
1,6	1, 2, 3	1
2	3	2
3,5	0, 2, 3	1

- * 0: Intra_16x16_Vertical
 1: Intra_16x16_Horizontal
 2: Intra_16x16_DC
 3: Intra_16x16_Plane

In the following, we will interpret the intra refinement process mathematically. Take Figure 3.10 for example where the block x_w is the w -block since it is predicted from the to-be-covered block \bar{x}_c and the other blocks including $x_{p,0}, x_{p,1}, x_{p,2}, x_{p,3}, x_{p,4}, x_{p,5}$ are the p -blocks. In this dissertation, we use the symbol "bar" above the variables to denote the reconstructed values after decoding, the symbol "prime" to denote the refined values after transcoding, and the symbol "hat" to denote the values after transcoding without performing refinement. In the original bitstream, the decoded signal of the block x_w can be represented in Eq.(3.1) where \bar{r}_w is the original residue.

$$\bar{x}_w = \bar{r}_w + IP_1(\bar{x}_c) \quad (3.1)$$

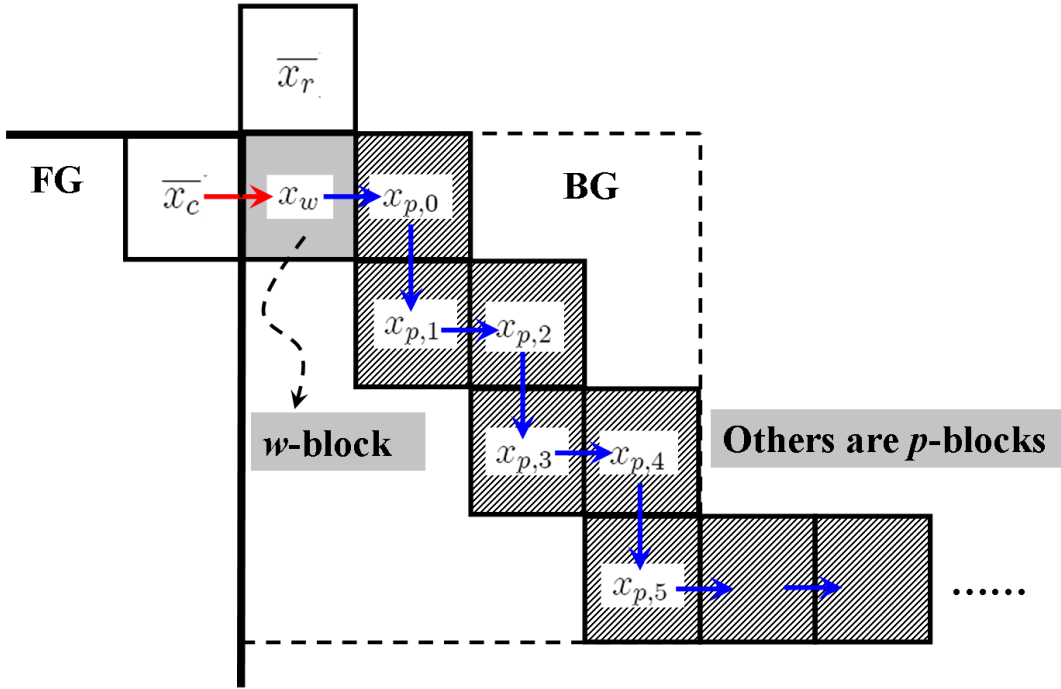


Figure 3.10: An Example of the Intra Prediction Chain

If we do not refine the block x_w at all, the decoded signal of the block x_w at client side is obtained from the original residue $\overline{r_w}$ and the reference block $\overline{y_c}$ from foreground region as represented in Eq.(3.2).

$$\widehat{x_w} = \overline{r_w} + IP_1(\overline{y_c}) = \overline{x_w} - IP_1(\overline{x_c}) + IP_1(\overline{y_c}) = \overline{x_w} + IP_1(\overline{y_c} - \overline{x_c}) \quad (3.2)$$

, where the symbol $IP_1(\overline{y_c} - \overline{x_c})$ indicates the serious mismatch of wrong reference propagated via intra prediction since the blocks in the foreground and background region are quite different in terms of pixel values. There fore, to improve the video quality, we refine the residue of w-block after the fully decoding and the intra mode switching. With the refined intra mode, we compute the new residue and coded block patterns. It should be noted that only the reconstructed values $\overline{x_w}$ is used as the original video is unavailable. The refinement of the block x_w

is defined by

$$\overline{r}_w' = \overline{x}_w - IP_2(\overline{x}_r) = \overline{r}_w + IP_1(\overline{x}_c) - IP_2(\overline{x}_r) \quad (3.3)$$

, where the symbols $IP_1(\overline{x}_c)$ and $IP_2(\overline{x}_r)$ denote intra prediction from the reference pixels \overline{x}_c and \overline{x}_r by using the original mode and the new mode respectively. The symbol \overline{r}_w is the decoded residue extracted from the source bitstream. Then, the refined residue is re-quantized and de-quantized as

$$\begin{aligned} \overline{r}_w' &= P_d \cdot P_e \cdot \overline{r}_w' = P_d \cdot P_e \cdot [\overline{r}_w + IP_1(\overline{x}_c) - IP_2(\overline{x}_r)] \\ &= P_d \cdot P_e \cdot \overline{r}_w + P_d \cdot P_e \cdot [IP_1(\overline{x}_c) - IP_2(\overline{x}_r)] \\ &= \overline{r}_w + IP_1(\overline{x}_c) - IP_2(\overline{x}_r) + e_w \end{aligned} \quad (3.4)$$

, where the symbol e_w denotes the quantization error of $[IP_1(\overline{x}_c) - IP_2(\overline{x}_r)]$. Lastly, the reconstructed data of the block x_w is shown in Eq.(3.5)

$$\overline{x}_w' = \overline{r}_w' + IP_2(\overline{x}_r) = \overline{r}_w + IP_1(\overline{x}_c) + e_w = \overline{x}_w + e_w \quad (3.5)$$

Therefore, the symbol e_w also denotes the refinement error due to the additional quantization process with respect to the original reconstructed value \overline{x}_w . As compared to Eq.(3.2), we can significantly reduce the mismatch from $IP_1(\overline{y}_c - \overline{x}_c)$ to e_w because and $[IP_1(\overline{x}_c) - IP_2(\overline{x}_r)]$ is smaller than $IP_1(\overline{y}_c - \overline{x}_c)$ and the quantization error of $[IP_1(\overline{x}_c) - IP_2(\overline{x}_r)]$ is much smaller than $[IP_1(\overline{x}_c) - IP_2(\overline{x}_r)]$ itself.

For the p -blocks, If we do not refine the residue of the block $x_{p,0}$, the decoded signal of the block $x_{p,0}$ at client side is obtained from the original residue $\overline{r}_{p,0}$ and the reference block \overline{x}_w'

instead of $\overline{x_w}$ as represented in Eq.(3.6)

$$\begin{aligned}\overline{\widehat{x_{p,0}}} &= \overline{r_{p,0}} + IP_1(\overline{x'_w}) = \overline{x_{p,0}} - IP_1(\overline{x_w}) + IP_1(\overline{x'_w}) \\ &= \overline{x_w} + IP_1(\overline{x'_w} - \overline{x_w}) = \overline{x_w} + IP_1(e_w)\end{aligned}\quad (3.6)$$

Therefore, the refinement of w -blocks may incur drift error that is amplified and propagated to the subsequent p -blocks by the intra prediction process. In order to alleviate the error propagation, we re-calculate the coefficients of p -blocks with the refined samples of w -blocks and the original intra modes as shown in Eq.(3.7), where we assume the block $x_{p,0}$ is the intra-coded p -block that uses the decoded data of the block x_w as prediction.

$$\begin{aligned}\overline{r_{p,0}'} &= \overline{x_{p,0}} - IP_1(\overline{x'_w}) = \overline{r_{p,0}} + IP_1(\overline{x_w}) - IP_1(\overline{x'_w}) \\ &= \overline{r_{p,0}} + IP_1(\overline{x_w} - \overline{x'_w}) = \overline{r_{p,0}} + IP_1(e_w)\end{aligned}\quad (3.7)$$

Similarly, the refined residue should be re-quantized and de-quantized as represented in Eq.(3.8) where the symbol $e_{p,0}$ denotes the drift error in the block $x_{p,0}$ and is identical to the quantization error of intra prediction of refinement error e_w in the n -th 4×4 block.

$$\begin{aligned}\overline{x'_{p,0}} &= \overline{r_{p,0}'} + IP_1(\overline{x'_w}) = P_d \cdot P_e \cdot \overline{r_{p,0}} + P_d \cdot P_e \cdot IP_1(e_w) + IP_1(\overline{x'_w}) \\ &= \overline{r_{p,0}} + IP_1(e_w) + e_{p,0} + IP_1(\overline{x'_w}) \\ &= \overline{x_{p,0}} - IP_1(\overline{x_w}) + IP_1(\overline{x'_w}) + IP_1(e_w) + e_{p,0} \\ &= \overline{x_{p,0}} + IP_1(\overline{x'_w} - \overline{x_w} + e_w) + e_{p,0} = \overline{x_{p,0}} + e_{p,0}\end{aligned}\quad (3.8)$$

Similarly, the next p -block can be derived:

$$\overline{x'_{p,m+1}} = \overline{x_{p,m+1}} + e_{p,m+1},$$

$$\text{where } e_{p,m+1} = P_d \cdot P_e \cdot IP(e_m) - IP(e_m), m = 0, 1, 2, 3, \dots \quad (3.9)$$

The generalized projection theory says that consecutive projections onto two non-convex sets will reach a trap point beyond which future projections do not change the results [33]. After several iterations of error correction, the drift error can not be further compensated. Therefore, we only perform error correction to the p -blocks within intra-coded w -MB rather than all the subsequent p -blocks. We observe that error correction for the p -blocks within intra-coded w -MB improves the averaged R-D performance up to 1.5dB. However, error correction for the intra-coded p -MBs has no significant quality improvement (less than 0.2dB).

3.4.2 Motion Vector Remapping (MVR)

The motion information of inter-coded w -MBs needs to be re-encoded since the motion vectors of the original bitstreams point to wrong reference samples after the embedding process. Since only the motion vector difference is encoded instead of the full scale motion vector. Owing to such prediction dependency, the new foreground video creates the wrong reference problem.

To solve the wrong reference issue, re-encoding the motion information is necessary for the surrounding MBs near the borders between foreground and background videos. In H.264/AVC, the motion vector difference is encoded according to the neighboring three motion vectors rather than the motion vector itself. Hence an identical motion vector predictor is needed for both encoder and decoder. However, due to foreground picture insertion, the motion compensation of background blocks may have wrong reference blocks from the new foreground pictures. Con-

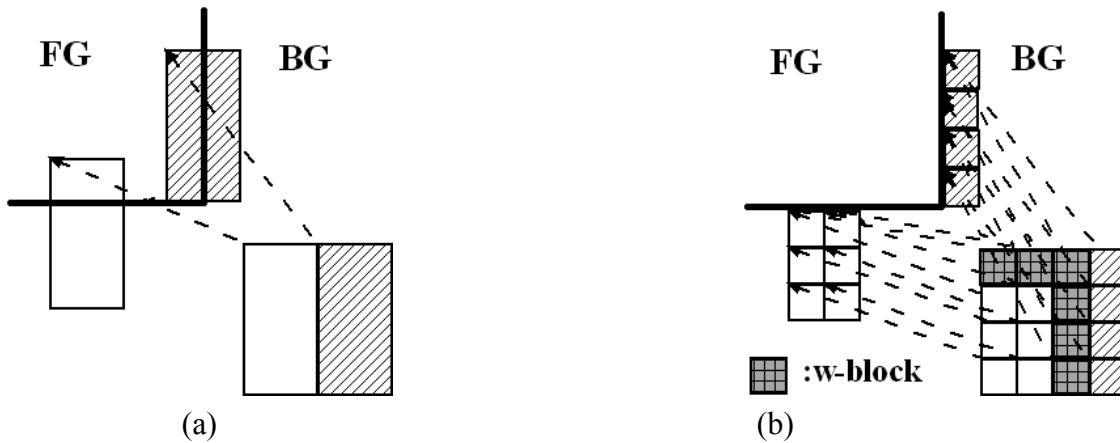


Figure 3.11: Illustration of the Motion Vector Re-mapping Technique. (a) The Original Coding Mode and Motion Vectors. (b) Refinement by Using Inter4x4 Mode and Re-mapped Motion Vectors.

sequently, the incorrect motion vectors cause serious prediction error propagated to subsequent pictures through the motion compensation process.

Within the background pictures, the reference pixels pointed by the motor vector may be lost or changed. For the MBs with wrong prediction reference, the motion vectors need to be refined for correct reconstruction at the receiver. To provide good tradeoff between the R-D performance and complexity, only the MBs using the reference blocks across the picture borders are refined. The refinement process can be done with motion re-estimation, mode decision and entropy coding. It takes significant complexity to perform exhaustive motion re-estimation and RDO mode decision for every MB with wrong prediction reference. Therefore, we use a motion vector re-mapping method (MVR) that has been extensively studied for MPEG-1/2/4 [14][15][16]. Before applying the MVR to the inter-coded w -MBs, we select the Inter4x4 mode as indicated in Figure 3.11. The MVR modifies the motor vector of every 4×4 w -block with a new motion vector pointing to the nearest of the four boundaries at the foreground picture. With the newly modified motion vectors, the prediction residue is re-computed and the HT transform is used to generate the new transform coefficients. Finally, the new motion vector and refined transform coefficients of w -blocks are entropy encoded as the final bitstream.

In the following, we will interpret the inter refinement process mathematically and explain why we can reduce the frame memory in the RFMT. Take Figure 3.12 for example where the blocks $x_{w,1}$ and $x_{w,2}$ are the w -blocks at frame 1 and frame 2 since they are predicted from the to-be-covered block $\overline{x_{c,0}}$ and $\overline{x_{c,1}}$, respectively. In the original bitstream, the decoded signal of the block $x_{w,1}$ can be represented in Eq.(3.10) where $\overline{r_{w,1}}$ is the original residue.

$$\overline{x_{w,1}} = \overline{r_{w,1}} + MC(\overline{x_{c,0}}) \quad (3.10)$$

If we do not refine the residue of the block $x_{w,1}$, the decoded signal of the block $x_{w,1}$ at client side is obtained from the original residue $\overline{r_{w,1}}$ and the reference block $\overline{y_{c,0}}$ from foreground region as represented in Eq.(3.11).

$$\begin{aligned} \widehat{\overline{x_{w,1}}} &= \overline{r_{w,1}} + MC(\overline{y_{c,0}}) = \overline{x_{w,1}} - MC(\overline{x_{c,0}}) + MC(\overline{y_{c,0}}) \\ &= \overline{x_{w,1}} + MC(\overline{y_{c,0}} - \overline{x_{c,0}}) \end{aligned} \quad (3.11)$$

, where the symbol $MC(\overline{y_{c,0}} - \overline{x_{c,0}})$ indicates the serious mismatch of wrong reference propagated via inter prediction since the blocks in the foreground and background region are quite different in terms of pixel values. Therefore, to improve the video quality, we refine the residue of w -block after the fully decoding and the motion vector re-mapping. The refinement of residue can be represented by Eq.(3.12) where the symbol $MC(\overline{x_{r,0}})$ denotes motion compensation from the new reference block $\overline{x_{r,0}}$ in the same region.

$$\begin{aligned} \overline{r_{w,1}}' &= \overline{x_{w,1}} - MC(\overline{x_{r,0}}) = \overline{r_{w,1}} + MC(\overline{x_{c,0}}) - MC(\overline{x_{r,0}}) \\ &= \overline{r_{w,1}} + MC(\overline{x_{c,0}} - \overline{x_{r,0}}) \end{aligned} \quad (3.12)$$

The refined residue data is re-quantized and de-quantized as

$$\begin{aligned}
 \overline{\overline{r_{w,1}}} &= P_d \cdot P_e \cdot \overline{r_{w,1}'} = P_d \cdot P_e \cdot [\overline{r_{w,1}} + MC(\overline{x_{c,0}} - \overline{x_{r,0}})] \\
 &= P_d \cdot P_e \cdot \overline{r_{w,1}} + P_d \cdot P_e \cdot MC(\overline{x_{c,0}} - \overline{x_{r,0}}) \\
 &= \overline{r_{w,1}} + MC(\overline{x_{c,0}} - \overline{x_{r,0}}) + e_{w,1}
 \end{aligned} \tag{3.13}$$

, where the symbol $e_{w,1}$ is the quantization error of $MC(\overline{x_{c,0}} - \overline{x_{r,0}})$. As mentioned in Eq.(2.2), the quantized data $\overline{r_{w,1}}$ has no further quantization distortion with the same quantization step.

In the transcoded bitstream, the decoded signal of this w -block is represented in Eq.(3.14).

$$\begin{aligned}
 \overline{x'_{w,1}} &= \overline{\overline{r_{w,1}}} + MC(\overline{x_{r,0}}) = \overline{r_{w,1}} + MC(\overline{x_{c,0}} - \overline{x_{r,0}}) + e_{w,1} + MC(\overline{x_{r,0}}) \\
 &= \overline{x_{w,1}} + e_{w,1}
 \end{aligned} \tag{3.14}$$

, where the symbol $e_{w,1}$ indicates the refinement error with respect to the original reconstructed value $\overline{x_{w,1}}$. As compared to Eq.(3.12), we can significantly reduce the mismatch from $MC(\overline{y_{c,0}} - \overline{x_{c,0}})$ to $e_{w,1}$ because $MC(\overline{x_{c,0}} - \overline{x_{r,0}})$ is smaller than $MC(\overline{y_{c,0}} - \overline{x_{c,0}})$ and the quantization error of $MC(\overline{x_{c,0}} - \overline{x_{r,0}})$ is much smaller than $MC(\overline{x_{c,0}} - \overline{x_{r,0}})$ itself.

We further consider the refinement of the next w -block $x_{w,2}$ whose the refined motion vector points to the block $x_{w,1}$. In the original bitstream, the decoded signal of the block $x_{w,1}$ can be represented in Eq.(3.15) where $\overline{r_{w,2}}$ is the original residue.

$$\overline{x_{w,2}} = \overline{r_{w,2}} + MC(\overline{x_{c,1}}) \tag{3.15}$$

The refinement of residue can be represented by Eq.(3.16) where the symbols $MC(\overline{x_{w,1}})$ denote

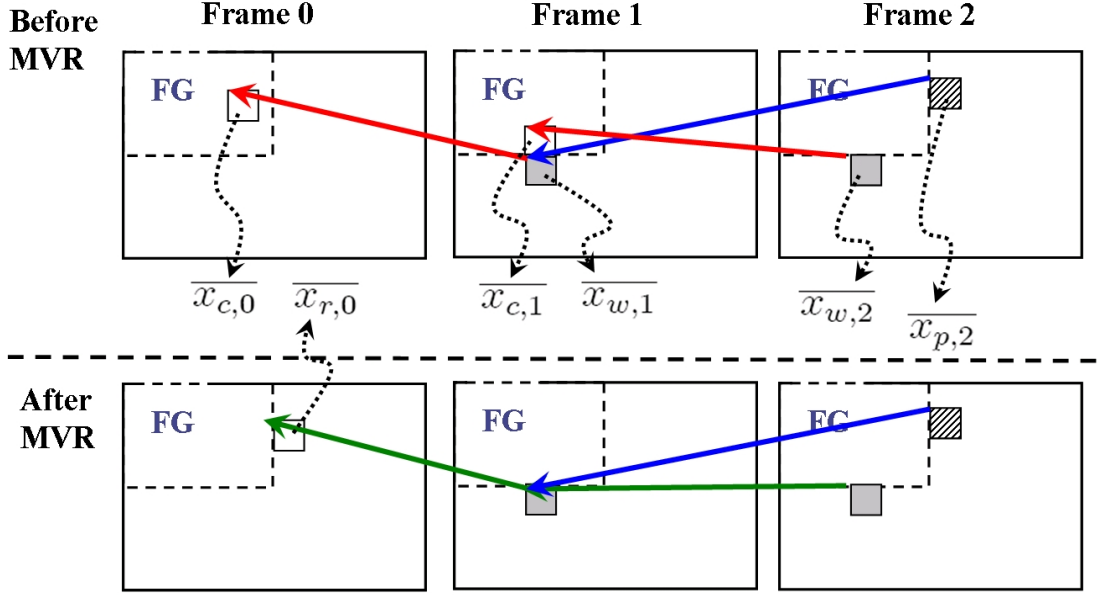


Figure 3.12: An Example of the Inter Prediction Chain

motion compensation from the reference block $\overline{x_{w,1}}$.

$$\begin{aligned}
 \overline{r_{w,2}''} &= \overline{x_{w,2}} - MC(\overline{x_{w,1}}) = \overline{r_{w,2}} + MC(\overline{x_{c,1}}) - MC(\overline{x_{w,1}}) \\
 &= \overline{r_{w,1}} + MC(\overline{x_{c,1}} - \overline{x_{w,1}})
 \end{aligned} \tag{3.16}$$

Similarly, the refined residue data is re-quantized and de-quantized as

$$\begin{aligned}
 \overline{\overline{r_{w,2}''}} &= P_d \cdot P_e \cdot \overline{r_{w,2}''} = P_d \cdot P_e \cdot [\overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x_{w,1}})] \\
 &= P_d \cdot P_e \cdot \overline{r_{w,2}} + P_d \cdot P_e \cdot MC(\overline{x_{c,1}} - \overline{x_{w,1}}) \\
 &= \overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x_{w,1}}) + e'_{w,2}
 \end{aligned} \tag{3.17}$$

, where the symbol $e'_{w,2}$ is the quantization error of $MC(\overline{x_{c,1}} - \overline{x_{w,1}})$. In the transcoded bit-

stream, the decoded signal of this w -block is represented in Eq.(3.18).

$$\begin{aligned}\overline{x''_{w,2}} &= \overline{\overline{r_{w,2}''}} + MC(\overline{x'_{w,1}}) = \overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x_{w,1}} + \overline{x'_{w,1}}) + e'_{w,2} \\ &= \overline{x_{w,2}} + MC(e_{w,1}) + e'_{w,2}\end{aligned}\quad (3.18)$$

As shown, the refinement error of the block $x_{w,1}$ propagates to the proceeding blocks that take the block $x_{w,1}$ as reference. To end off the error propagation in such circumstances, we use the reconstructed signal after refinement $\overline{x'_{w,1}}$ instead of the original reconstructed signal $\overline{x_{w,1}}$ to refine the block $x_{w,2}$ as represented by Eq.(3.19)

$$\begin{aligned}\overline{r_{w,2}'} &= \overline{x_{w,2}} - MC(\overline{x'_{w,1}}) = \overline{r_{w,2}} + MC(\overline{x_{c,1}}) - MC(\overline{x'_{w,1}}) \\ &= \overline{r_{w,1}} + MC(\overline{x_{c,1}} - \overline{x'_{w,1}})\end{aligned}\quad (3.19)$$

Then, the refined residue data is re-quantized and de-quantized as

$$\begin{aligned}\overline{\overline{r_{w,2}'}} &= P_d \cdot P_e \cdot \overline{r_{w,2}'} = P_d \cdot P_e \cdot [\overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x'_{w,1}})] \\ &= P_d \cdot P_e \cdot \overline{r_{w,2}} + P_d \cdot P_e \cdot MC(\overline{x_{c,1}} - \overline{x'_{w,1}}) \\ &= \overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x'_{w,1}}) + e_{w,2}\end{aligned}\quad (3.20)$$

, where the symbol $e_{w,2}$ is the quantization error of $MC(\overline{x_{c,1}} - \overline{x'_{w,1}})$. In the transcoded bit-stream, the decoded signal of this w -block is represented in Eq.(3.21).

$$\begin{aligned}\overline{x'_{w,2}} &= \overline{\overline{\overline{r_{w,2}'}}} + MC(\overline{x'_{w,1}}) = \overline{r_{w,2}} + MC(\overline{x_{c,1}} - \overline{x'_{w,1}} + \overline{x'_{w,1}}) + e_{w,2} \\ &= \overline{x_{w,2}} + e_{w,2}\end{aligned}\quad (3.21)$$

As shown, the symbol $MC(e_{w,1})$ in Eq.(3.18) is eliminated in Eq.(3.21). Therefore, for each w -block, the original reconstructed signal is no longer used for the future refinement and could be replaced with the reconstructed signal after the refinement. Therefore, we use a shared frame memory to buffer the reference pixels for both the decoding and re-encoding.

The refinement may occur at the border MBs with the skip mode. Since two neighboring motion vectors are used to infer the motion vector of a MB with the skip mode, the border MBs with the skip mode may be classified as two kinds of w -MBs due to the insertion of the foreground blocks. Firstly, for the w -MBs whose motion vectors that do not refer to a reference block covered by the foreground pictures, the skip mode is changed to inter 16×16 mode to compensate the mismatch of motion vectors by the motion inference. Secondly, for the w -MBs whose motion vectors point to reference blocks covered by the foreground pictures, the skip mode is changed to inter 16×16 mode and the motion vector is refined to new position by the MVR method. Then, the refined coefficients are computed according to the new prediction.

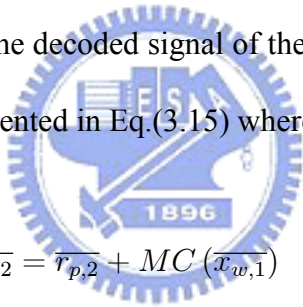
To fix the wrong sub-pixel interpolation after inserting the foreground pictures, the blocks whose motion vectors point to the wrong sub-pixel positions are refined. H.264/AVC supports finer sub-pixel resolutions such as 1/2-, 1/4- and 1/8-pixel. The sub-pixel samples do not exist in the reference buffer for motion prediction. To generate the sub-pixel samples, a 6-tap interpolation filter is applied to full-pixel samples for the sub-pixel location. The sub-pixel samples within 2-pixel range of picture boundaries are refined to avoid vertical and horizontal artifacts. The refinement is done by replacing the wrong sub-pixel motion vectors with the nearest full-pixel motion vectors and the new prediction residues are re-encoded.

3.4.3 Syntax Level Bypassing (SLB)

To minimize the transcoding complexity, the blocks within inter-coded p -MBs and n -MBs are bypassed at the syntax level after the CAVLD. Since the blocks within p -MBs and n -MBs are not affected by the picture insertion directly, the syntax data can be forwarded unchanged to the multiplexer.

As for the intra-coded frames, the affected blocks by video insertion are refined to compensate the drift error. We observe that the correction of p -blocks within the w -MBs can significantly improve the quality. While the correction of intra-coded p -MBs might get a bit of quality improvement with drastically increased complexity.

As for the inter-coded frames, we examine the effectiveness of error compensation in the following. In the original bitstream, the decoded signal of the block $x_{p,2}$, whose motion vector points to the block $x_{w,1}$, can be represented in Eq.(3.15) where $\overline{r_{p,2}}$ is the original residue.



$$\overline{x_{p,2}} = \overline{r_{p,2}} + MC(\overline{x_{w,1}}) \quad (3.22)$$

The residue can be re-computed with the refined pixel values $\overline{x'_{w,1}}$ by Eq.(3.23).

$$\begin{aligned} \overline{r'_{p,2}} &= \overline{x_{p,2}} - MC(\overline{x'_{w,1}}) = \overline{r_{p,2}} + MC(\overline{x_{w,1}}) - MC(\overline{x'_{w,1}}) \\ &= \overline{r_{p,2}} + MC(\overline{x_{w,1}} - \overline{x'_{w,1}}) \end{aligned} \quad (3.23)$$

Similarly, the transcoded data can be represented by Eq.(3.24) where the refinement error of the

w -block is propagated to the next p -block.

$$\begin{aligned}
 \overline{x'_{p,2}} &= \overline{r_{p,2}'} + MC(\overline{x'_{w,1}}) \\
 &= P_d \cdot P_e \cdot \overline{r_{p,2}} - P_d \cdot P_e \cdot MC(\overline{x_{w,1}} - \overline{x'_{w,1}}) + MC(\overline{x'_{w,1}}) \\
 &= \overline{r_{p,2}} - P_d \cdot P_e \cdot MC(e_{w,1}) + MC(\overline{x'_{w,1}}) \\
 &= \overline{x_{p,2}} - MC(\overline{x_{w,1}}) + MC(\overline{x'_{w,1}}) \\
 &= \overline{x_{p,2}} + MC(\overline{x'_{w,1}} - \overline{x_{w,1}}) = \overline{x_{p,2}} + MC(e_{w,1}) \tag{3.24}
 \end{aligned}$$

Let's assume the refinement of w -block performs well and the term of $MC(e_{w,1})$ is smaller than the quantization step size, it means that the quantization of $MC(e_n)$ becomes zero. If our assumption is valid, the term $P_d \cdot P_e \cdot MC(e_{w,1})$ in Eq.(3.24) can be removed. Therefore, the refinement error of the block $x_{w,1}$ still remains in the following p -blocks that take the block $x_{w,1}$ as reference. Alternatively, we use the original residue of the block $x_{p,2}$ without refinement in the transcoded bitstream, the decoded data at the client side is

$$\begin{aligned}
 \overline{\overline{x_{p,2}}} &= \overline{r_{p,2}} + MC(\overline{x'_{w,1}}) = \overline{x_{p,2}} - MC(\overline{x_{w,1}}) + MC(\overline{x'_{w,1}}) \\
 &= \overline{x_{p,2}} + MC(\overline{x'_{w,1}} - \overline{x_{w,1}}) = \overline{x_{p,2}} + MC(e_{w,1}) \tag{3.25}
 \end{aligned}$$

As shown, the client can derive the same results as Eq.(3.24). Thus, the drift compensation of inter-coded p -block has no quality improvement despite extra computations. In terms of complexity reduction, we bypass all the transform coefficients of p -MB and n -MB to the transcoded bitstream.

In summary, the proposed MW-VET deals with each type of block efficiently according to Table 3.4. Figure 3.13 summarizes the refinement flow during transcoding. In addition, the

Table 3.4: The Corresponding Operations of the RFMT for Each Block Type During the VET Transcoding

Block type		Operations
<i>w</i> -MB	Intra-coded <i>w</i> -block	IMS and CR*
	Inter-coded <i>w</i> -block	MVR and CR*
	Intra-coded <i>p</i> -block	CR*
	Inter-coded <i>p</i> -block	SLB
	<i>n</i> -block	SLB
<i>p</i> -MB		SLB
<i>n</i> -MB		SLB

* CR means coefficient re-calculation.

partially re-encoding method can preserve picture quality. For the applications requiring multi-generation transcoding, the deterioration caused by successive decoding and re-encoding of the signals can be eliminated with the reuse of the coding information from the original bitstreams. As the motion compensation with multiple reference frames is applied, the proposed algorithm is still valid. Specifically, it first classifies the type of each block (i.e. *n*-block, *p*-block, and *w*-block according to Figure 2.2). The classification is based on whether the reference block is covered by foreground pictures and it does not matter what reference picture is chosen. In other words, the wrong reference problem with multiple reference frame feature, as illustrated in Figure 3.14, is an extension of Figure 2.2. Then, the aforementioned MVR and SLB processes are applied to each type of inter-coded block.

3.5 Simulation Results

To support PIP functionality, we have made several refinements to guarantee the correctness of reconstructed picture of PIP bitstream under the consideration of transcoding complexity. For the clarity, we indicate the steps of these refinements by Figure 3.15. Table 3.5 explains what happens in each step and what refinement have been done in detail.

The R-D performance and execution time are compared based on the transcoding meth-

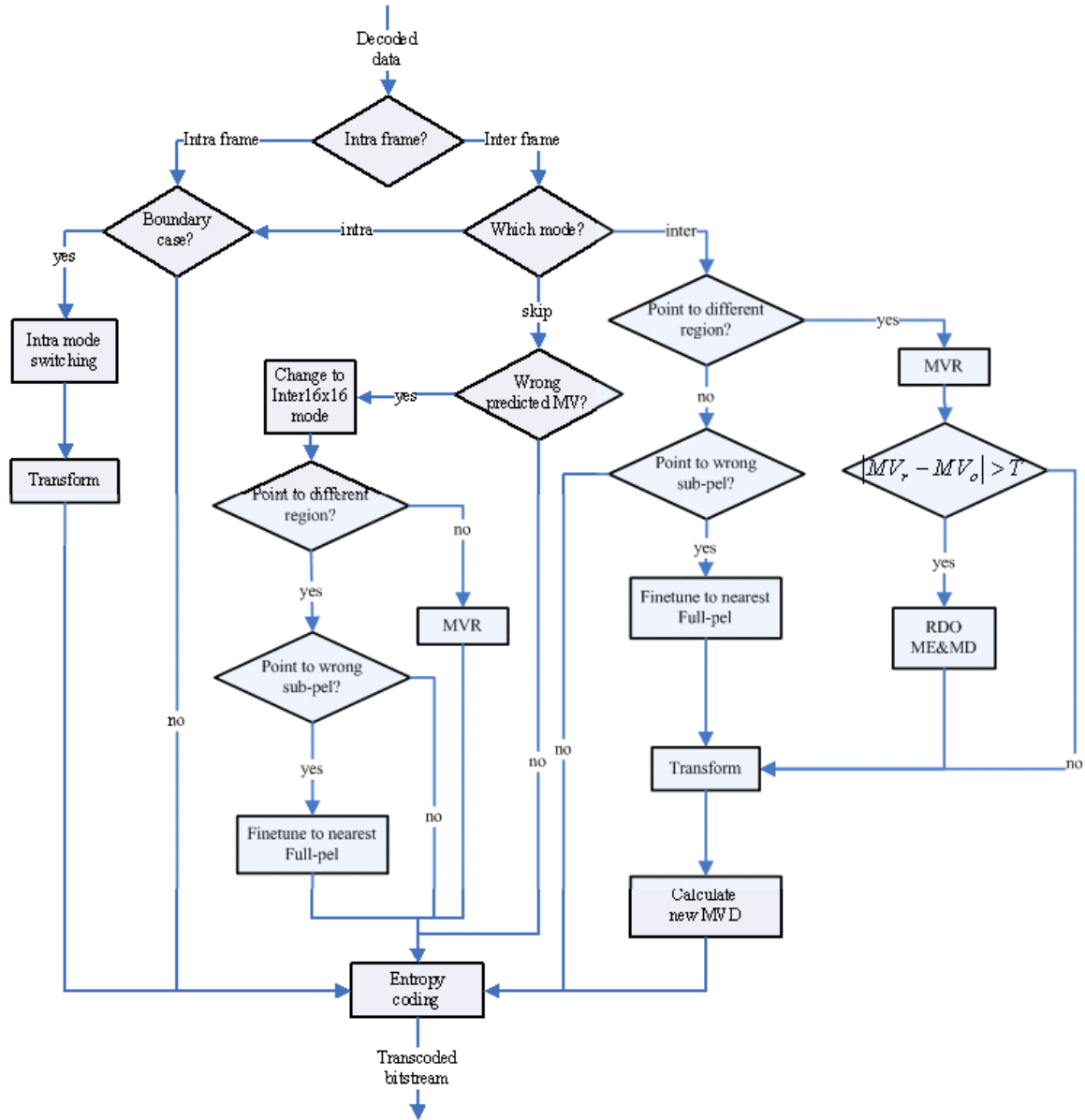

Figure 3.13: The Flow Chart of the Proposed RFMT

Table 3.5: The Detailed Refinement during the VET Transcoding

	Problem	Refinement
1	Intra prediction	Intra mode switching
2	Motion vector predictor	Re-calculate motion vector difference
3	Inter prediction	Motion vector re-mapping
4	Intra blocks in P-frame	Re-encoding the necessary cases
5	Residue mismatch	Re-calculate transform coefficients
6	Unrestricted MVs of FG	Fine tune to full-pixel position

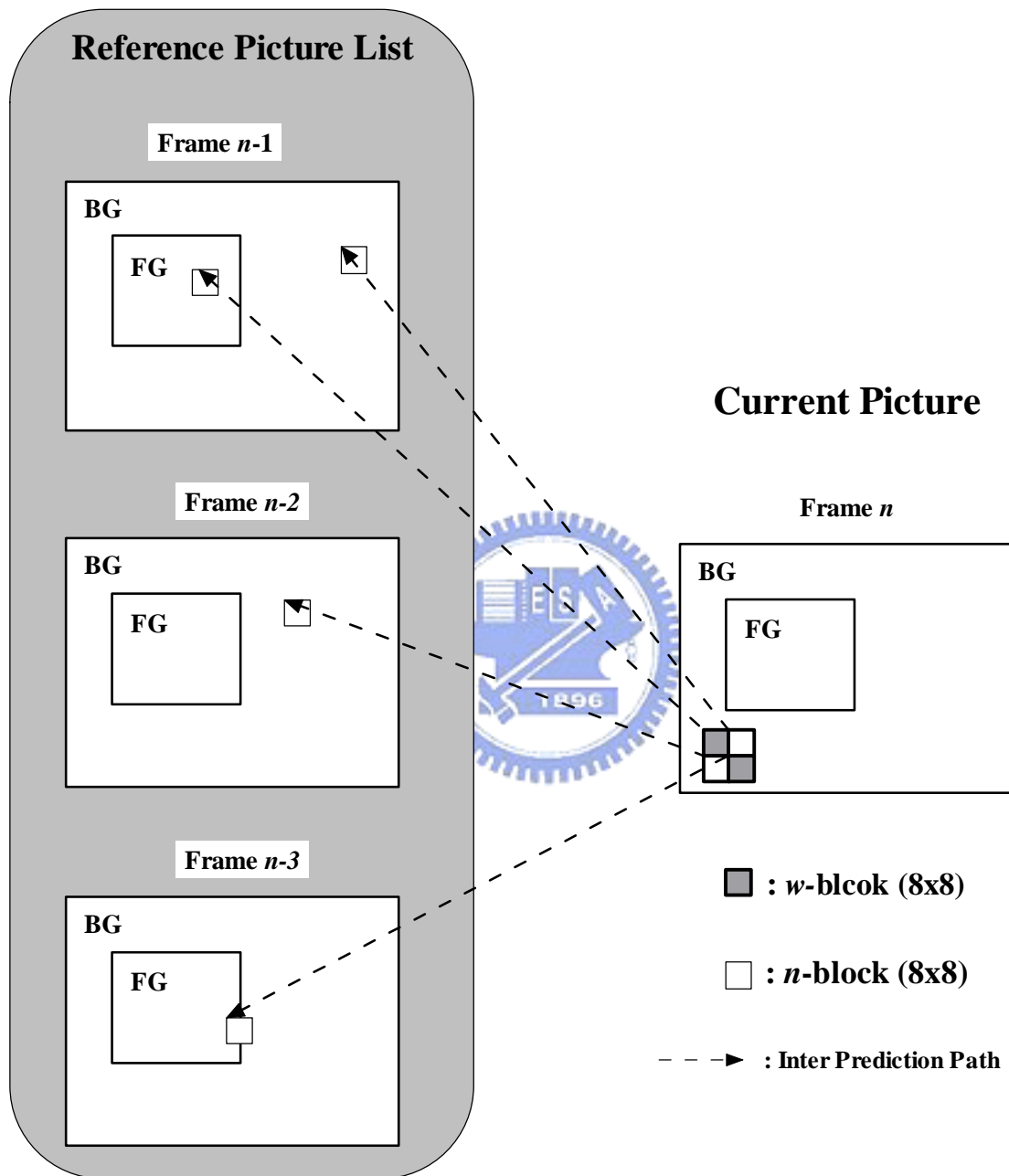


Figure 3.14: The Extended Wrong Reference Problem when Multiple Reference Frame is Used



Figure 3.15: The Visual Illustration after Each Refinement Step during the VET Transcoding

Table 3.6: The Encoder Parameters for the Experiments

Frame size	QCIF (176x144), SD (720x480)
Frame rate	30 frames/s
GOP structure	IPPP.....P
Total frame	100
Intra period	15
Reference frame number	1
Motion estimation range	16 for QCIF, 64 for SD
Quantization step size	17,21,25,29,33,37

ods, test sequences and picture insertion scenarios. For a fair comparison, all the transcoding methods have been implemented based on H.264/AVC reference software of version JM9.4. In addition, all the transcoders are built using Visual .NET compiler on a desktop with Windows XP, Intel P4 3.2 GHz and 2 Giga bytes DRAM. To further speed up the H.264/AVC based transcoding, the source code of the reference CAVLD module is optimized using a table lookup technique [34]. In the simulations, the test sequences are pre-encoded with the test conditions as shown in Table 3.6. The notation for each new transcoded bitstream is ‘background_foreground_x_y’, where x and y are the coordinates of the foreground picture. The values of x and y need to be on the MB boundaries within the background picture. To evaluate the picture quality of each reconstructed sequence, the two original source sequences are combined to be the reference video source for peak-signal-to-noise-ratio (PSNR) computation.

The percentage of each MB type and each 4×4 block type is shown in Figure 3.16. In general, the *p*-MBs occupy 30% to 80% of MBs and the percentage of the *w*-MBs is less than 15%. In addition, the *w*-blocks occupy only 5% of the 4×4 blocks. Bypassing all the *p*-blocks that are 95% of blocks accelerates the transcoding process as shown in Table 3.7. On the average, as compared to the CPDT, the MW-VET can achieve 25 times of speedup with improved picture quality.

Table 3.8 lists the PSNR comparison to show the effectiveness of error correction for differ-

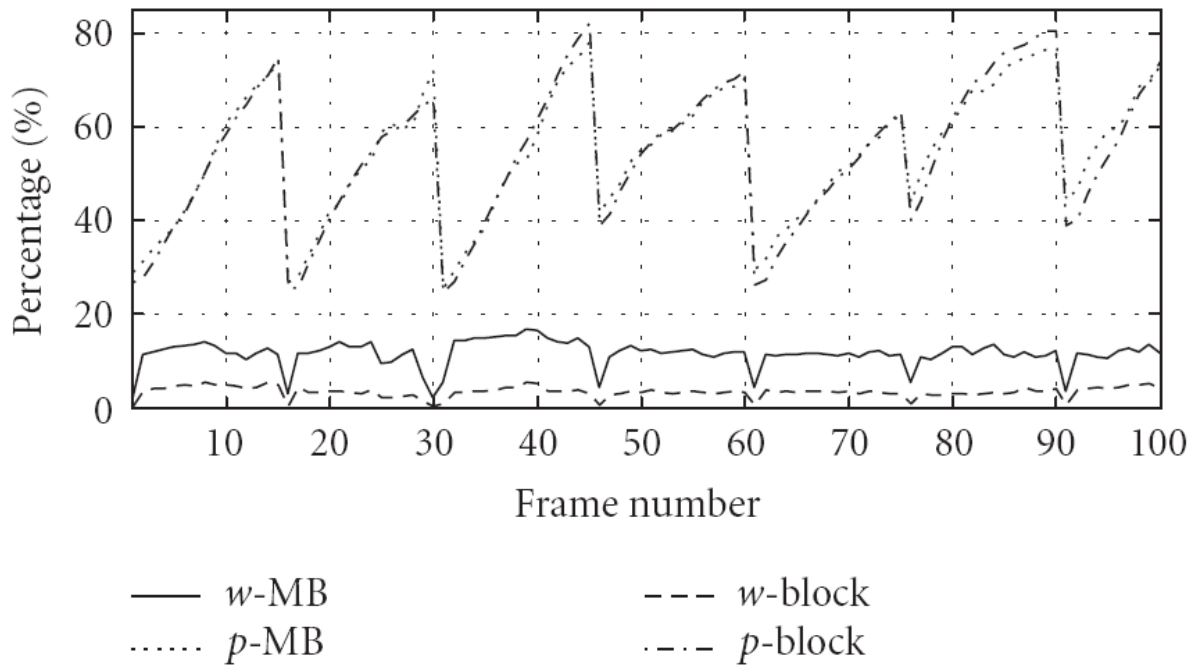


Figure 3.16: The Percentage of the Macroblock Types and the Block types during the VET Transcoding

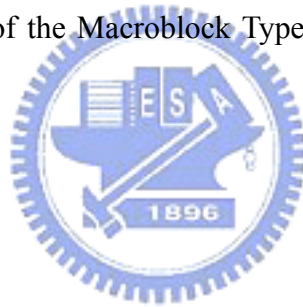


Table 3.7: The Improvement of Execution Time and Quality as Compared to the CPDT ⁽¹⁾

VET combination		Speed-up ratio	PSNR gain of Luma component
BG ⁽²⁾	FG ⁽³⁾ & lication		
Stefan	Mobile_1_1	25	+1.72 dB
Table	Carphone_1_1	28	+1.56 dB
Stefan	Mobile_1_1	28	+ 1.18 dB
	Foreman_33_1		
	News_1_20		
	Coastguard_33_20		
Table	MD_1_1	25	+ 1.15 dB
	Stefan_33_1		
	Carphone_1_20		
	News_33_20		

⁽¹⁾ Intel P4 3.2G, 2GB SDRAM, Windows XP and Visual .NET compiler.

⁽²⁾ All are in SD (720×480) resolution.

⁽³⁾ All are in QCIF (176×144) resolution.

Table 3.8: The Effectiveness of Error Correction (EC) for Different Kinds of p -blocks

Methods	PSNR
Golden	43.73
CPDT	42.02
RFMT w/o EC	41.18
RFMT with EC for the p -blocks in intra-coded w -MBs	43.16
RFMT with EC for all intra-coded p -blocks	43.33
RFMT with EC for all inter-coded p -blocks	43.14

ent kinds of blocks. The Golden method is not a transcoding scheme. The R-D curves of Golden method are obtained from encoding the original picture-in-picture source sequences. The inclusion of the R-D curves of Golden method is to highlight the upper bound of a transcoder. The error correction of p -blocks in the intra-coded w -MBs can obtain a significant gain in picture quality. However, the error correction for other p -blocks almost has no quality improvement while the complexity increases dramatically. Therefore, the results verify our derivations in Section 3.4.

The R-D performance of different approaches at various bit rates and different VET scenarios are compared. We embedded one foreground picture into one background picture at different positions in Figure 3.17 and Figure 3.18. The performance of RFMT is better than that of CPDT. At medium and high bit rates, the RFMT can offer up to 1.5 dB improvement in PSNR. Even through the mode and motion vectors obtained by our IMS and MVR is not always the optimal solution, the simulation results show that our IMS and MVR approaches provide a solution close to the optimal case. In the comparison, we have plotted the R-D curves named as RFMT_RDO to show the optimal R-D performance when the partial re-encoding is performed under RDO mode decision and motion vector re-estimation. It could be observed that the R-D performance of RFMT with IMS and MVR is very close to that of RFMT_RDO.

Figure 3.19 shows the R-D curve of transcoding bitstreams that embed four foreground pictures onto one background picture at the same time. As compared with the one-foreground

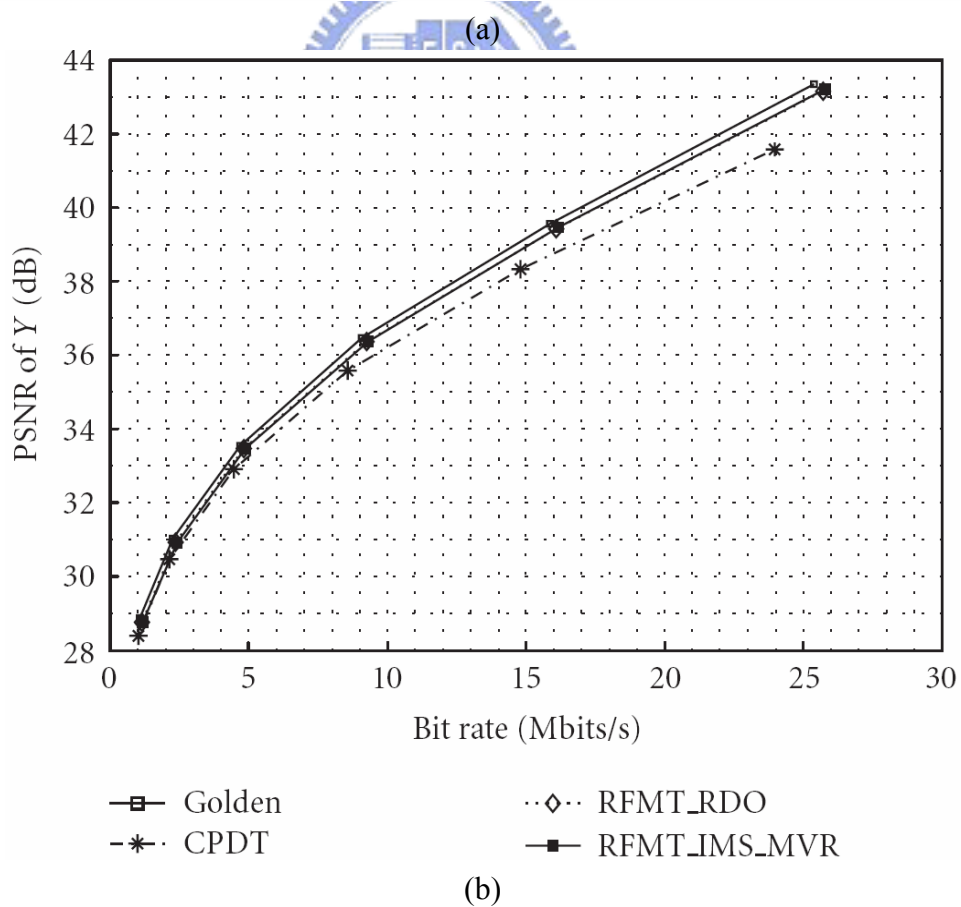
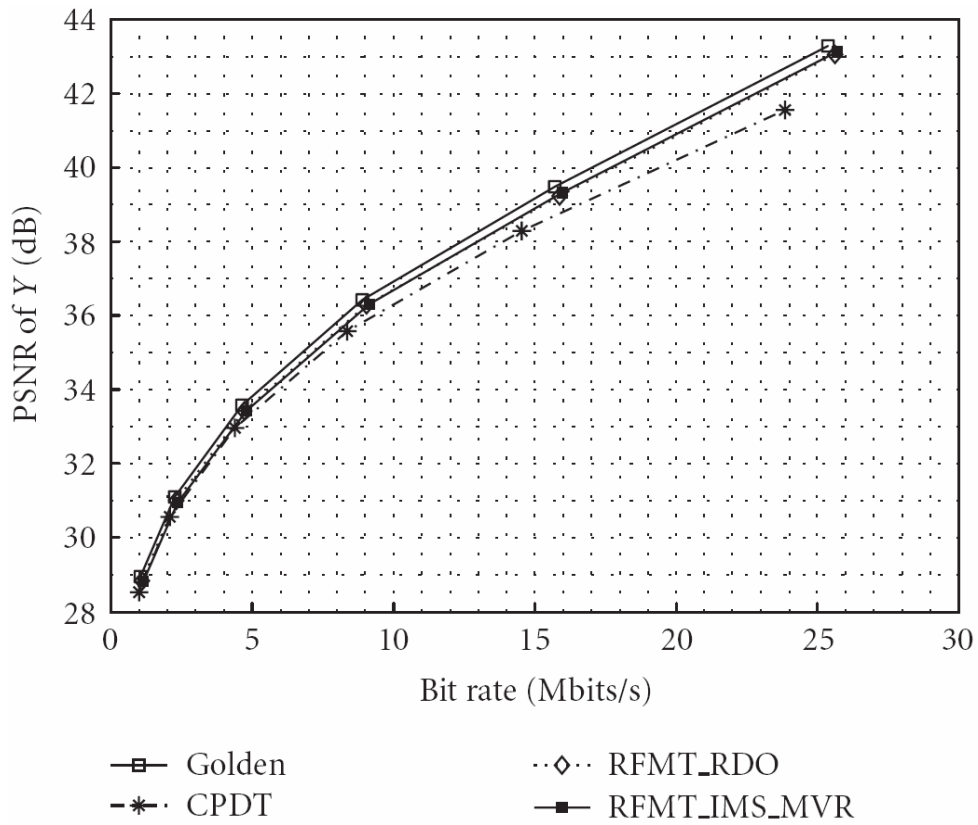
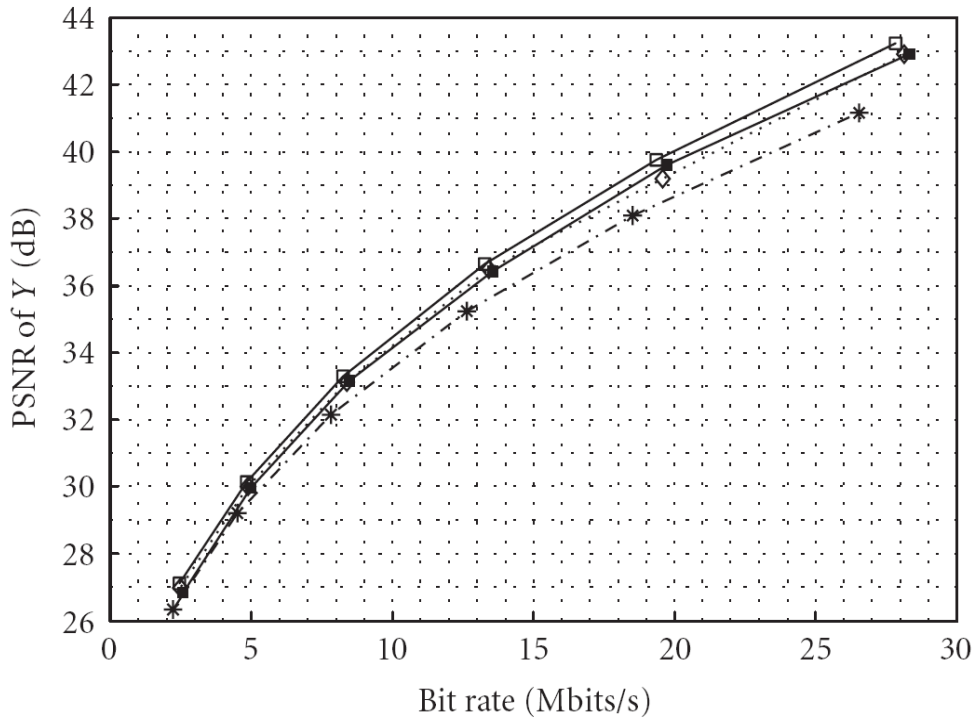
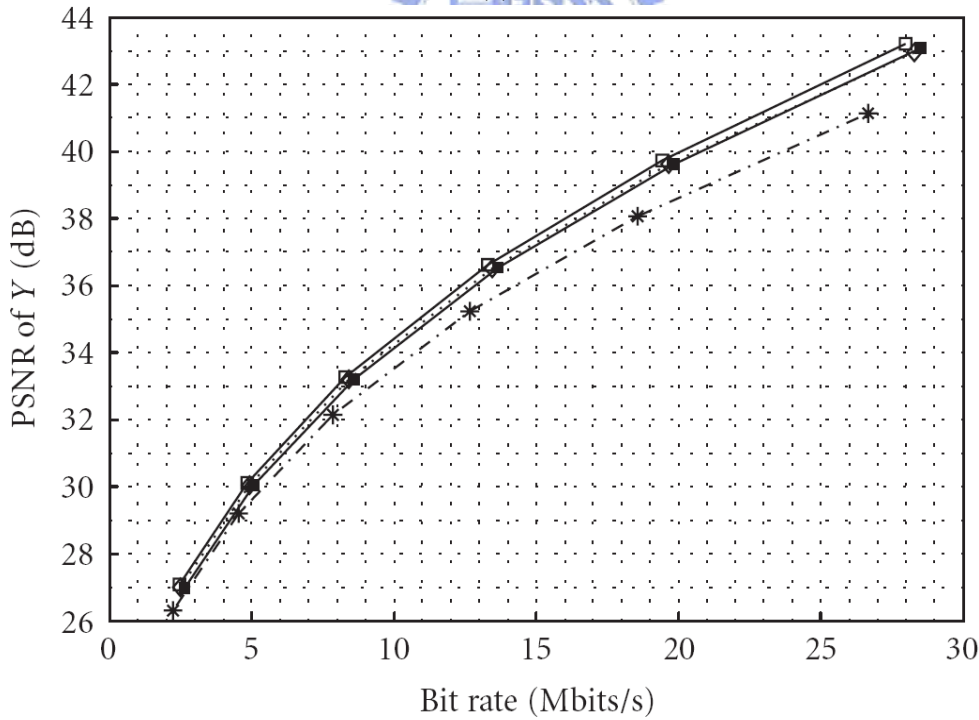


Figure 3.17: The Rate-Distortion Performance of the Luminance Component When One Fore-ground Carphone_QCIF is Embedded in Table_SD: (a) Table_SD_Carphone_QCIF_1_1. (b) Table_SD_Carphone_QCIF_33_20.



(a)



(b)

Figure 3.18: The Rate-Distortion Performance of the Luminance Component When One Fore-ground Foreman_QCIF is Embedded in Mobile_SD: (a) Mobile_SD_Foreman_QCIF_1_1. (b) Mobile_SD_Foreman_QCIF_33_20.

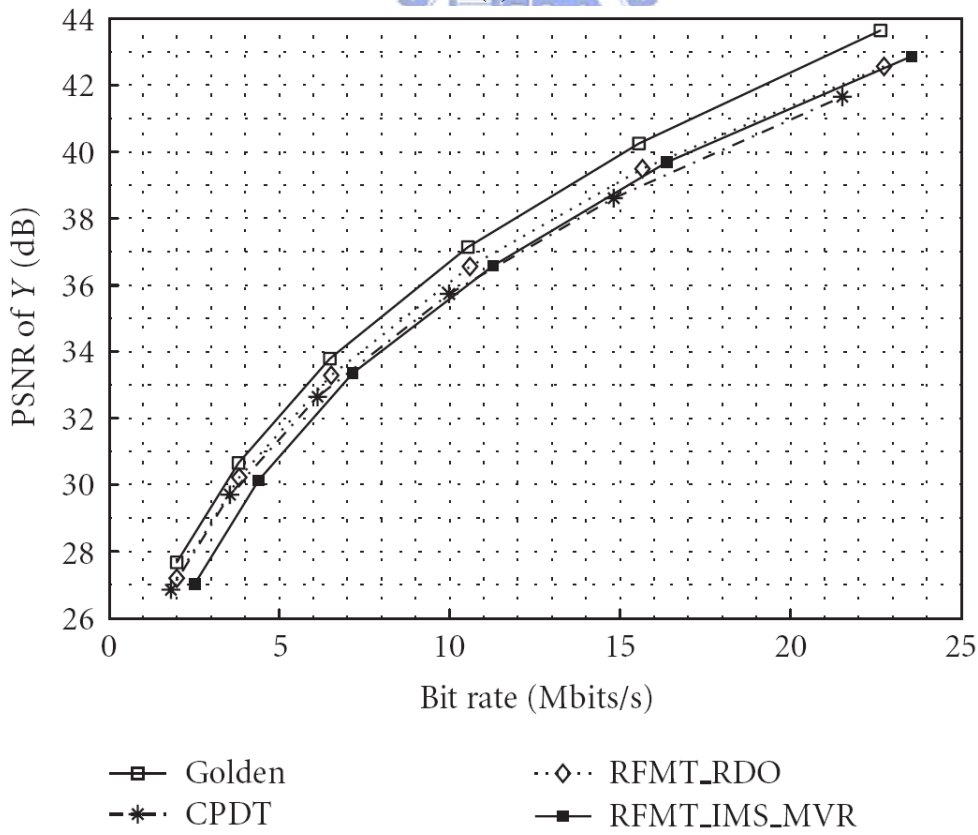
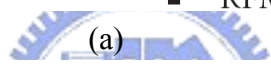
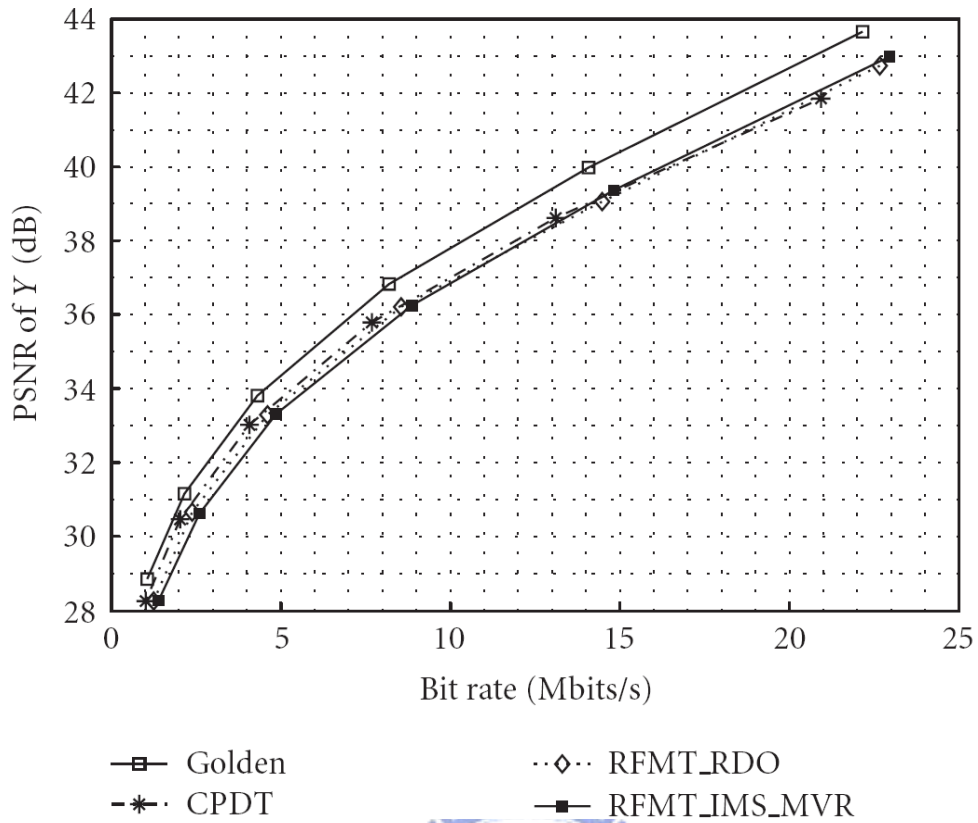
VET scenarios, the performance has a little degradation because that the ratio of w -blocks and p -blocks increases. Figure 3.20 shows the performance of multi-generation transcoding that embeds one foreground picture to the background picture every generation. Our MW-VET can retain the R-D performance while the CPDT degrades every generation. Thus, the proposed MW-VET is robust for the multi-generation transcoding.

3.6 Summary

In this chapter, we have proposed a low-complexity algorithm of a H.264/AVC multiple-window video embedding transcoder (MW-VET) to embed the multiple foreground videos into one background video. The pictures are inserted at the MB-aligned positions to retain high flexibility.

As the prediction is applied to the slice-aligned data partitions within the original bitstreams, the SGT parses and merges the bitstreams directly. When the prediction is applied to the region-aligned data partitions, the MBs with wrong prediction reference are processed with the RFMT that partially re-encodes the blocks to minimize the number of refined blocks. To handle inter-coded and intra-coded blocks that suffer from the wrong reference problem, the RFMT employs motion vector remapping (MVR) and intra mode switching (IMS) respectively. The un-affected MBs are handled by the syntax level bypassing (SLB) in terms of transcoding throughput and picture quality. Apart from a fully functional decoder, the proposed algorithm requires only minor extra complexity for foreground insertion.

To improve coding efficiency and alleviate drifting errors, every w -block, whose (inter or intra) reference samples are covered entirely or partially by the foreground images, is fine tuned such that the updated inter or intra reference samples are derived completely from the background region. Further, the residues of the p -blocks within intra-coded macroblocks, which are



(b)

Figure 3.19: The Rate-Distortion Performance of the Luminance Component by Four Foregrounds Embedding with the Single-Generation Transcoding: (a) Table_SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20. (b) Mobile_SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20.

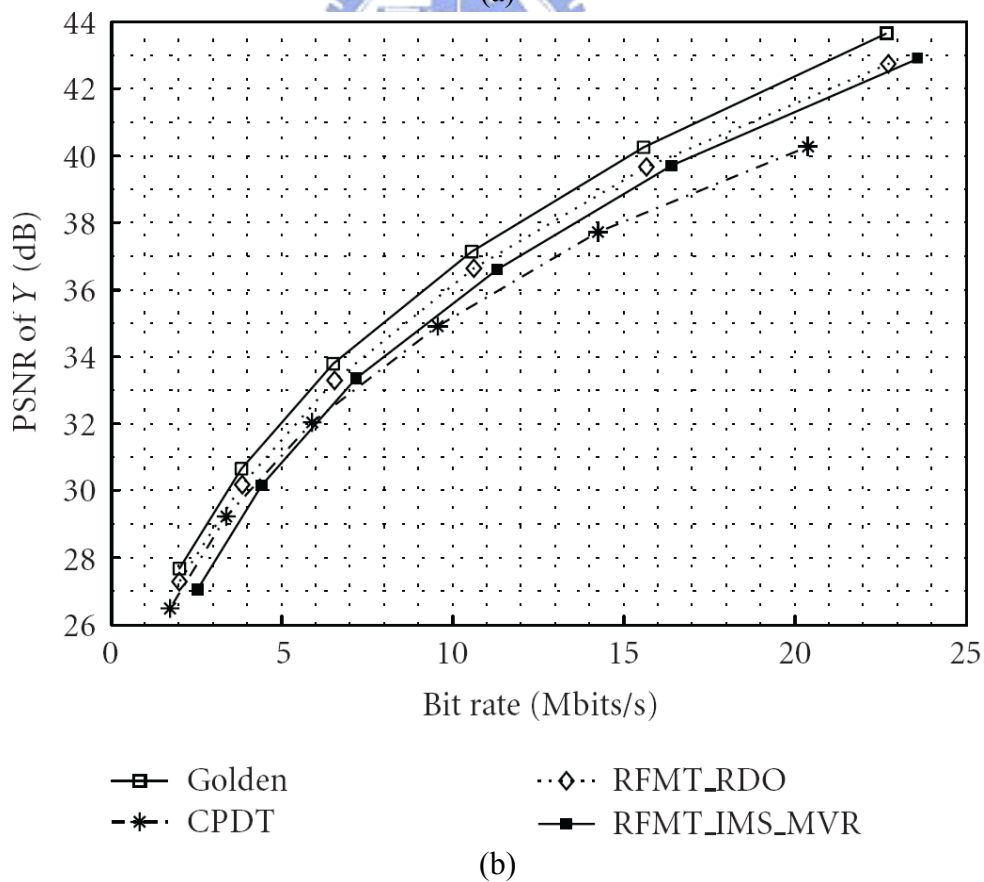
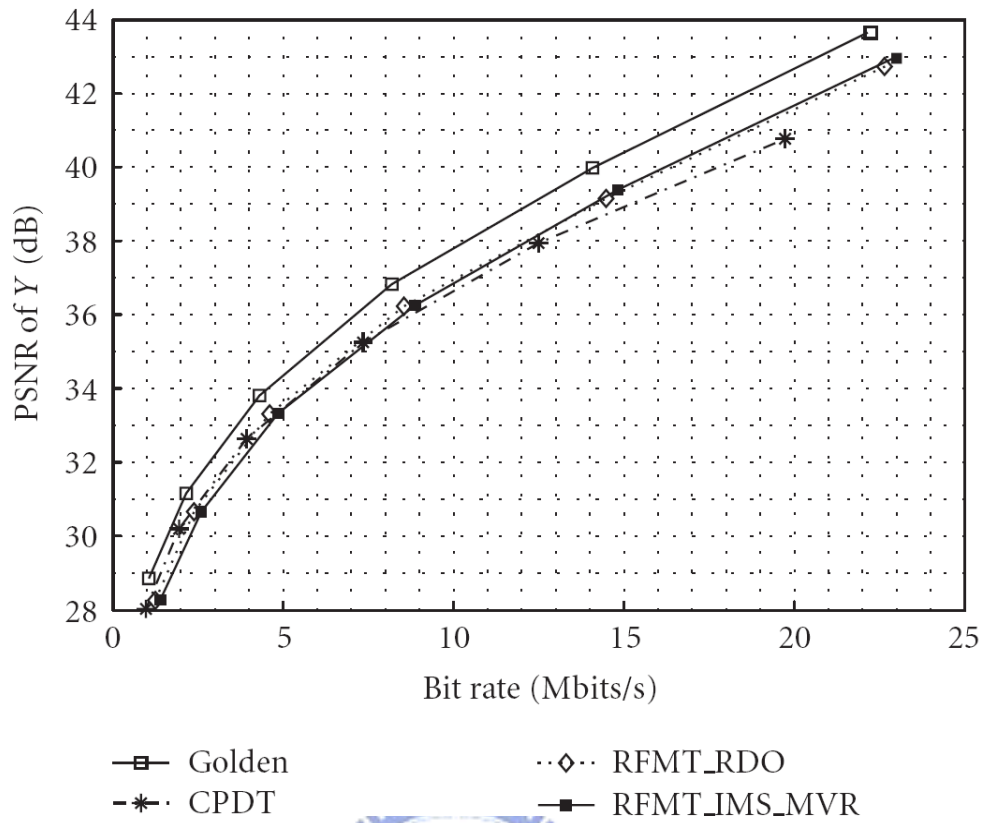


Figure 3.20: The Rate-Distortion Performance of the Luminance Component by Four Foregrounds Embedding with the Multi-Generation Transcoding: (a) Table_SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20. (b) Mobile_SD_MD_QCIF_1_1_Stefan_QCIF_33_1_Carphone_QCIF_1_20_News_QCIF_33_20.

subsequently predicted from the w -blocks, are refined to stop error propagation in the spatial domain. In addition, unnecessary computations are skipped by detecting unaffected blocks.

Our results show that the RFMT as compared to the cascaded pixel domain transcoder (CPDT) can significantly reduce the processing complexity by 25 times with similar or higher R-D performance. In addition, the RFMT can achieve up to 1.5 dB quality improvement in PSNR. Based on the RFMT, the quality improvement over the CPDT is significant for multi-generation transcoding.



CHAPTER 4

System Architecture Design Space Exploration



4.1 Introduction

In Chapter 3, a low-complexity multiple-window video embedding transcoder (MW-VET) is proposed based on an H.264/AVC decoder. The proposed algorithm can be implemented as software on a general purpose computer, however, the throughput of a pure software implementation is insufficient for real-time applications as shown in Table 4.1.

To improve the transcoding throughput, in this chapter a platform-based architecture is proposed for combining H.264/AVC VET transcoder and decoder based on the low complexity al-

Table 4.1: The Transcoding Throughput Performance of Proposed Algorithm of a Software Implementation

Resolution of BG	Resolution of FG	Throughput Performance ⁽¹⁾ (Frame per Second)
CIF (352x288)	QCIF (176x144)	3~4
SD (720x480)	QCIF (176x144)	1~1.5
HD (1920x1088)	QCIF (176x144)	0.1~0.3

⁽¹⁾ Intel P4 3.2G, 2GB SDRAM, Windows XP and Visual .NET compiler.

gorithm in Chapter 3. The platform-based design is adopted while the system is partitioned into several dedicated modules to provide task-level parallelism. A VET pipe is used to support the operations for the foreground insertion, which includes the decoding of foreground syntax, the refinement of the background macroblocks, and the re-encoding of the combined syntax elements. With such implementation, the proposed architecture can perform video decoding and transcoding alternatively or simultaneously. Such VET enabled decoders can find their applications in a peer-to-peer service community, in which every user enjoys versatile video embedding services while using only partial computational power.

In addition, this chapter advocates the simulation-based evaluation at a higher level of abstraction for the system architecture design space exploration. The system development needs to perform an early estimation and balance for some system-level parameters because the optimization of the individual module may not lead to the optimal performance for the whole system. In addition, the level of synchronization granularity for each layer between the memory hierarchies strongly influences the system performance.

The rest of this chapter is organized as follows: Section 4.2 describes the algorithm to architecture mapping. Section 4.3 presents the system architecture of the proposed video embedding transcoder. Section 4.4 describes the flow of the design space exploration. Section 4.5 presents the pruned design space for exploring the level of synchronization granularity and the design

combination of the video pipe. Section 4.6 shows how we evaluate the system performance. Then, Section 4.7 shows the simulation results using transaction level modeling. Finally, Section 4.8 summarizes this chapter.

4.2 Algorithm to Architecture Mapping

This section describes the mapping from algorithm to architecture and some very early design decisions such as architecture selection and system partitioning. First of all, we use the platform-based architecture to enable the hardware/software (HW/SW) partition with high flexibility and extensibility. This concept has been widely studied in the literature [35][36][37]. Although the implementation of the pure software results in the best flexibility, the required computational power is inadequate. Specifically, implementing all the tasks on a reduced instruction set computing (RISC) CPU consumes more than 46,000 cycles per macroblock [38][39][40], thus it makes it impossible to achieve the real time application for the 1920x1088 @ 60Hz videos, where the RISC CPU needs to operate at 23 GHz or higher.

To meet the real-time requirement, we partition all the computationally intensive tasks as the hardware components. Figure 4.1 shows the top level data flow. First of all, the coarse-grain functions, including the setting of the sequence parameter set (SPS), the picture parameter set (PPS), the slice header (SH), and the locations of foreground pictures, are performed in software on a host processor because these tasks require less computation and lower synchronization rate. For the macroblock-level refinement which requires higher synchronization rate and more computation, we partition this part as dedicated hardware. The data flow and data dependency of the macroblock-level refinement in RFMT can be further presented in Figure 4.2 where the targeted algorithm is decomposed into separated tasks with self-contained functionality. According to the complexity analysis in [41][42][43], the grayed blocks including the transform

module, the prediction module, the deblocking module, and the entropy coding module that take most of computation in an H.264/AVC decoder. In addition, the blue blocks represent the communicationally intensive tasks, i.e. the DRAM accesses. We distribute the computational and communicational workload by mapping these tasks onto the separated, dedicated hardware modules. To further increase the throughput performance, we use five-staged pipeline to provide task-level parallelism as shown in Figure 4.2. As compared to the tasks for decoding, the techniques used for the refinement in the RFMT consume relatively less computation. However, in order to alleviate the overhead due to the interrupt latency of the RISC CPU, we implement all the refinement techniques in the hardware component with a minor extra cost. Therefore, we implement the RFMT in a heterogeneous system including the RISC CPU, the communication bus, and the dedicated hardware components. The RISC CPU is mainly in charge of the high-level tasks such as the sequence-level or slice-level processing monitor, the location of foreground insertion, and other user interaction tasks. The hardware components address the low-level computationally intensive video signal processing.

4.3 Highly Efficient System Architecture

The performance of a typical ARM platform-based design is constrained by high communications overhead and insufficient performance of the RISC processor for entropy coding [34]. According to the results in [34], the performance of dedicated hardware takes 1,280 cycles per macroblock in the worst case while the overall system performance is about 120,000 cycles per macroblock on the average. Specifically, each hardware accelerator works as a standalone co-processor of the RISC processor via only one 32-bit AHB bus. However, all the co-processor processes the data from the ARM core. In addition, data movement without direct memory access (DMA) incurs great penalty on the execution cycles. In addition to the massive data

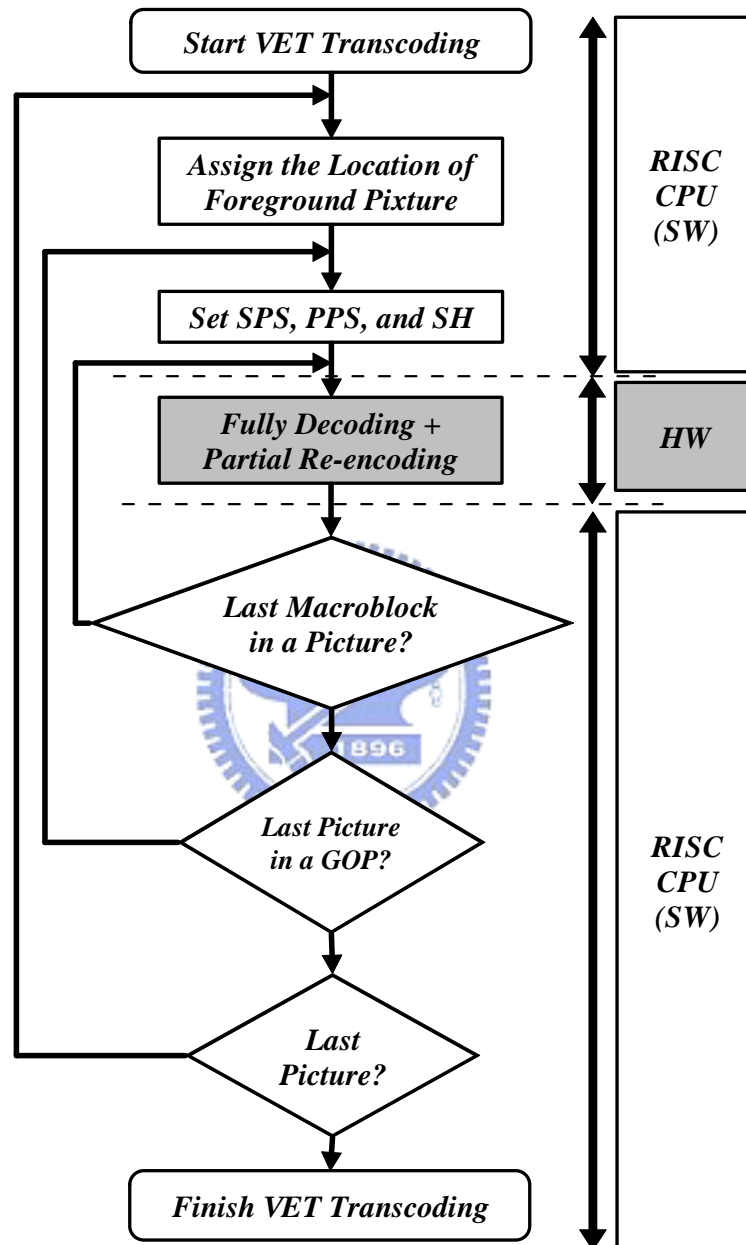


Figure 4.1: The Top Level Data Flow of RFMT and Its System Partitioning

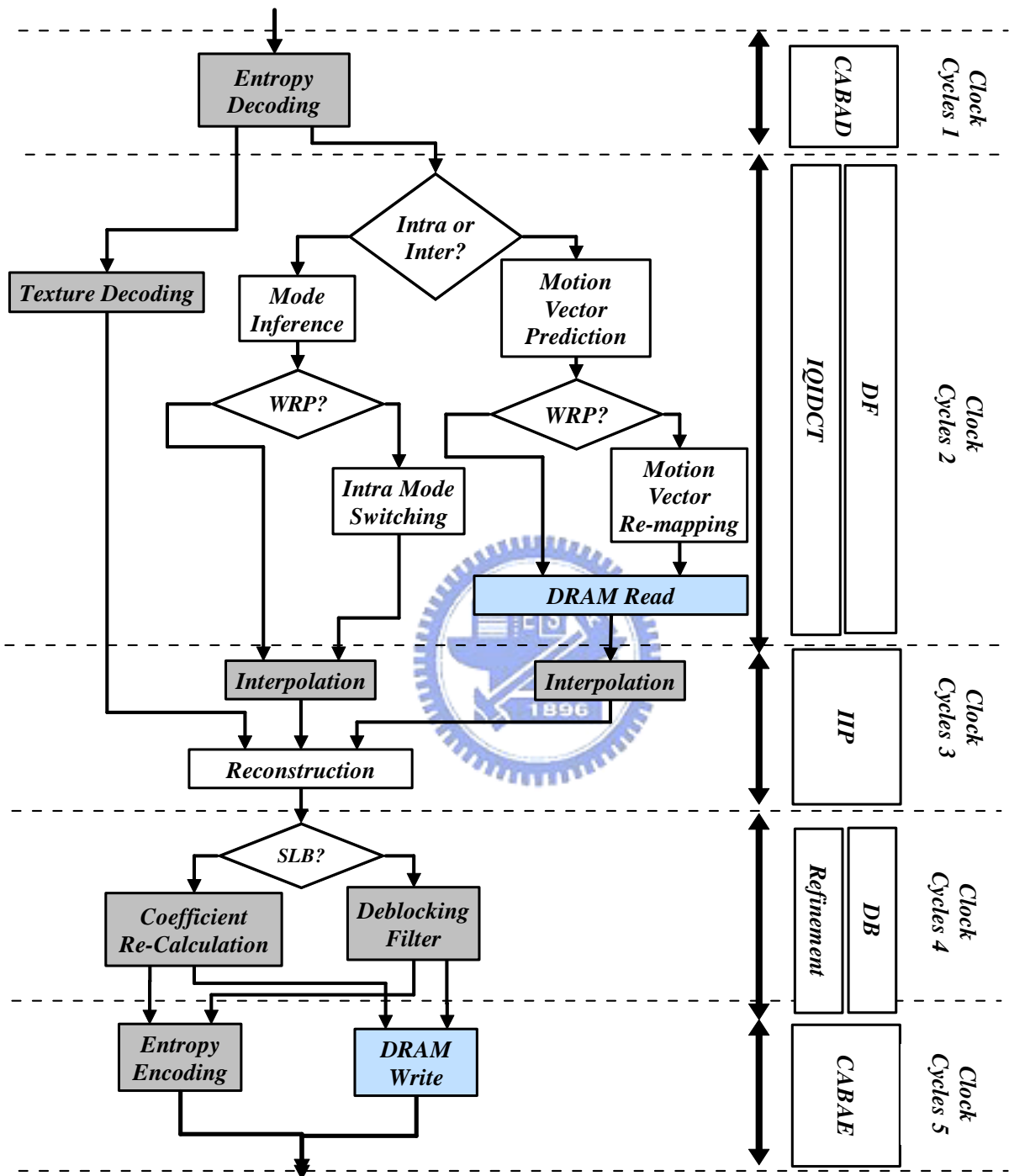


Figure 4.2: The Refinement for Each Processing Macroblock and Its Hardware Partitioning which is of Five-Stage Pipeline

transfer, the entropy coding places the heaviest computational load on the RISC processor. The general purpose processor cannot process the operations of entropy coding in real-time such that the execution cycle count of processor is considerably greater than that of dedicated hardware. Therefore, system performance and the utilization of each hardware accelerator are both significantly degraded due to synchronization overhead in the communication-centric platform-based designs.

To solve the problems addressed above, we implement most functional blocks by dedicated modules interconnected with the shared ping-pong memories to provide massive parallelism. First, the entropy coding is implemented as dedicated hardware such that the load of the RISC core can be significantly alleviated. Second, we connect each module such that most of data communications take place through the video pipe instead of passing information between the RISC core and the AHB bus. Third, we allocate several ping-pong buffers between adjacent pipeline stages such that the computation and communication cycles can be overlapped.

Figure 4.3 depicts the block diagram of the proposed system architecture, which mainly consists of six parts including (1) the RISC CPU, (2) the memory sub-system, (3) the video decoding pipe, (4) the VET pipe, (5) the 32-bit AHB control bus, and (6) the 32-bit AHB data bus. The RISC CPU acts as the data flow controller, which programs the other modules and constantly monitors their statuses via the 32-bit control bus. For the video transcoding, the decoding and the VET pipes represent the major data paths. The decoding pipe decodes the background video and adaptively maps the motion vectors and intra predictors depending on the location of the foreground video. Meanwhile, the VET pipe re-encodes the syntax elements decoded from the foreground or the background bitstream for the foreground insertion. The decoded background pictures are stored in the external DRAM for the temporal references and a dedicated 32-bit AHB data bus is employed for the intensive data transfer between the video

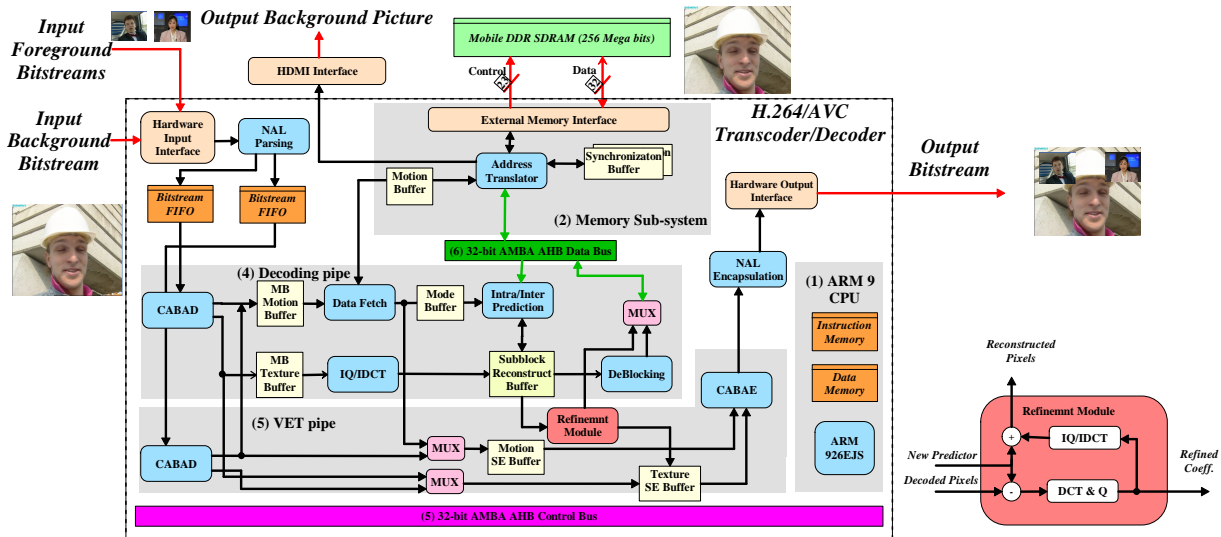


Figure 4.3: The System Architecture of the Proposed Video Embedded Transcoder and Decoder

pipe and the external DRAM.


4.3.1 Memory Hierarchy

For the data storage, we utilize the four-level memory hierarchies: (1) the external memory for the large memory space requirement of the decoded picture buffer, (2) the synchronization buffer in the memory sub-system for decoupling the latency of external memory and the operation of video pipe, (3) the ping-pong memories between modules for eliminating the communications overhead of the video pipe and reducing the on-chip bus bandwidth, and (4) the local memory within each module for reducing the input bandwidth.

First, we use double-data-rate (DDR) synchronous dynamic random access memory (SDRAM) as the external memory for increasing nearly twice the bandwidth of the high-density memory. Second, in order to increase the on-chip bus utilization while maximizing the effective throughput of DRAM accesses, a synchronization buffer is used in the memory sub-system so as to re-format the read/write data while performing the DRAM accesses. With the synchroniza-

tion buffer, the DRAM latency can be decoupled from the operation of video pipe such that the internal modules can fetch or load data with a lower latency. Third, the video pipe consists of a number of autonomous modules that communicate through ping-pong memories. The pipeline with interconnected memories frees the processor from spending clock cycles to cope with the intensive data movement. In addition, the use of the ping-pong memories avoids the communications overhead because each module is reading from (or writing into) a buffer rather than directly communicating with another module. Fourth, the local memories are employed as configuration register and pixel line buffer for storing the general parameter and reducing the frequent access to the external memory. The next two sections detail the operations of the decoding and VET pipes when the system is configured for transcoding.

4.3.2 Video Decoding Pipe



The decoding process of the background bitstream begins with the control information followed by the slice data. The control information such as sequence parameter set (SPS), picture parameter set (PPS), and slice header (SH) are firstly acquired by the RISC CPU for programming each module in the decoding pipe. Then, the slice data is decoded by feeding the remaining bitstream through the decoding pipe, which starts from the CABAC decoding (CABAD) and ends with the DeBlocking filtering. The CABAD module performs entropy decoding of the motion information and the transform coefficients. The decoded motion information is then used in the Data Fetch module to derive the motion vectors or the intra prediction modes. Meanwhile, the transform coefficients are processed by the IQ/IDCT module to reconstruct the prediction residues. While the memory sub-system fetches the data in preparation for the next execution stage, the Inter/Intra Prediction (IIP) module generates the inter or intra predictors by performing the sub-pixel interpolation or the directional prediction using the reference pixels

pre-fetched and stored in the synchronization buffer. After that, the block being decoded is reconstructed by adding the prediction residuals to the corresponding predictors and is updated using the deblocking filtering.

Each specific module within the proposed video pipe can be mapped into the well-designed micro-architecture. For instance, [44] and [45] are the design alternatives for the inverse transform module. However, the state-of-the-art designs of the pixel prediction, which includes the inter and intra prediction, pose some disadvantages in terms of efficiency and utilization. To that end, a unified filtering architecture is proposed for the inter and intra prediction in the Section 5.3.

In addition to the normal decoding process, the decoding pipe performs the prediction refinement with the minimal effort. First, the Data Fetch module determines the current decoded block according to the coding mode, motion vector, and the location for the foreground insertion. Second, if the current block is recognized as a w-block or a p-block within an intra-coded macroblock, a particular procedure is invoked for the motion vector remapping or the intra mode switching such that the wrong reference problem can be solved as described in Chapter 3. Therefore, the traffic-intensive motion estimation and computation-intensive mode decision are eliminated from the proposed architecture. Third, together with the remapped motion vectors or the refined intra prediction modes, the new prediction block and the current reconstructed block are passed to the VET pipe for the refinement of prediction residues. Particularly, the new prediction block is composed of the full-pixel block without consuming the execution cycles of the IIP module.

4.3.3 Video Embedding Transcoding Pipe

The VET pipe is started from the CABAC decoding of the foreground syntax elements and ended with the CABAC encoding (CABAE) of the syntax elements decoded from either the foreground bitstream or the background bitstream. Particularly, for the w-block or the p-block within an intra-coded macroblock, the Refinement module updates its prediction residues by subtracting the new prediction block from the current reconstructed block. Furthermore, the reconstruction of these refined w-/p-blocks overwrites the decoded background image so as to minimize the drifting errors in the output bitstream. For lower power consumption, the de-blocking filtering is skipped during the refinement of the w-/p-blocks to remove redundant computation. Although such hardware-oriented simplification may cause drifting errors for the current decoded picture, there is no noticeable difference in the visual quality at high bit rate while providing higher quality for the transcoded bitstream. With the low complexity algorithm in the absence of motion estimation and mode decision, we can implement the function of transcoding upon an H.264/AVC decoder by the VET pipe.

Figure 4.4 shows the comparison for the visual quality. As shown, there is no noticeable difference between Figure 4.4 (a) and Figure 4.4 (b). On the other hand, Figure 4.4 (c) has more visual artifacts than Figure 4.4 (d) especially around the boundary of the inserted foreground. Therefore, such hardware-oriented simplification can seamlessly integrate the normal decoding process and transcoding functionality while providing better transcoded visual quality.

4.3.4 Memory Sub-system

The memory sub-system consists of an address translator, a memory interface, and a synchronization buffer. The address translator takes the motion vectors from the Data Fetch module and generates the physical DRAM address for the external memory interface, which further issues



Figure 4.4: The subjective Quality Comparison of the 100th Frame: (a) Decoded Picture without Refinement Update (b) Decoded Picture with Refinement Update (c) Transcoded Picture without Refinement Update (d) Transcoded Picture with Refinement Update.

the DRAM commands and enables the auto-precharge function adaptively. Particularly, for the concurrent DRAM access and video decoding, all the read/write data required for the video pipe will be firstly fetched in the synchronization buffer. Such a buffer is also used for reformatting the read/write data exchanged between the video pipe and the off-chip DRAM so as to simultaneously increase the on-chip bus utilization while maximizing the effective throughput of the DRAM access. Specifically, for the implementation of the synchronization buffer, two SRAMs are used as the ping-pong buffers to overlap the off-chip and on-chip data movements. Section 5.2 will describe more details about the memory sub-system.

In addition, the irregular and variable-rate bumping process of the H.264/AVC increases the complexity of memory management. Different from prior video coding standards where the decoder can output pictures according to the time stamps in the picture or slice header, the network abstraction layers (NAL) in the H.264/AVC contain no timing information. Thus, the bumping process manages the output of the decoded pictures such that the output order conformance is guaranteed. Currently, the bumping process of the decoded pictures is invoked when a decoded picture is to be stored in the decoded picture buffer (DPB) while there is no empty space. Multiple decoded pictures could be output simultaneously while for some time instances none of the pictures are output. The variable output rate causes extremely varying bandwidth for the DRAM access. To ensure a constant output rate for the decoded pictures when simultaneously performing both the transcoding and the decoding, a regulation buffer with the size of the DPB is created and described as a modified bumping process in Appendix B.

Table 4.2: The Transcoding Throughput Performance of Proposed Algorithm of a Software Implementation

Architecture	Granularity of Memory Subsystem (GM)	Granularity of Video Pipe (GP)	Size of Synchronization Buffer N_1 (Bits)	Size of Internal Ping-Pong Buffer N_2 (Bits)	Total Buffer Size N_1+N_2 (Bits)
A	16x16	16x16	20160	41184	61344
B	16x16	8x8	20160	32480	52640
C	16x16	4x4	20160	30432	50592
D	8x8	8x8	5424	32480	37904
E	8x8	4x4	5424	30432	35856
F	4x4	4x4	2160	30432	32592

4.3.5 Task Scheduling

We devise a hybrid pipeline structure to schedule the video pipe, where the entropy coding modules operate at macroblock (MB) level while the other modules communicate at the sub-block level to yield a smaller synchronization buffer. In addition, our hybrid pipeline structure can harmonize both the VET and the decoding pipes such that the transcoded bitstream can be output in a timely manner while the background video is being decoded. For clarity, Figure 4.5 depicts the scheduling of our hybrid pipeline design, where the CABAD and CABAE modules operate at the macroblock level while the other modules communicate at the sub-block level. In particular, there are six architectural candidates according to the different levels of data exchange. Table 4.2 summaries these six architectures and Figure 4.6 to Figure 4.11 illustrate the corresponding task scheduling of each architecture. Lastly, the DeBlocking filter and the Refinement module are scheduled to operate at the same sub-block stage because only either one of them is activated for execution. This also suggests that the decoded background image is made of the output results of the normal decoding process and the particular refinement process.

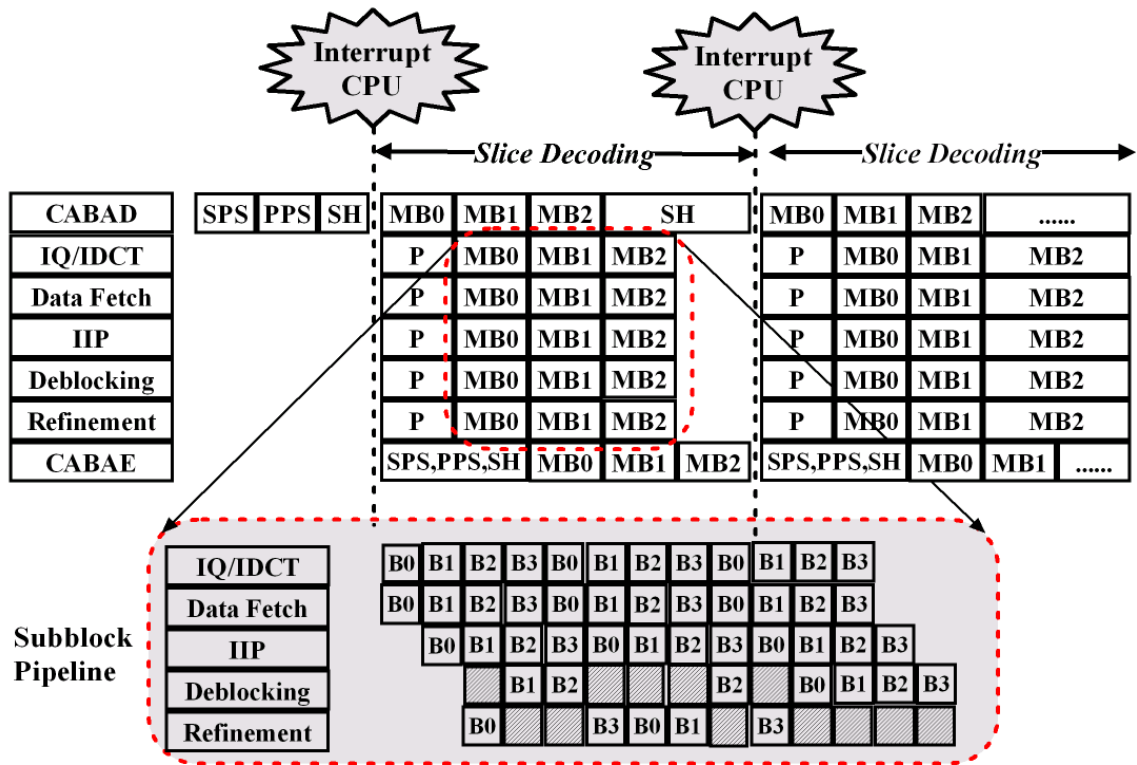


Figure 4.5: The scheduling of the Video Pipe Where P Means the CPU Programs the Individual Module



Clock Cycles	CABAD	IQIDCT	DRAM Read	Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE
0	MB_0									
1	MB_1	MB_0	MB_0							
2	MB_2	MB_1	MB_1	MB_0	MB_0					
3	MB_3	MB_2	MB_2	MB_1	MB_1	MB_0	MB_0	MB_0		
4	...	MB_3	MB_3	MB_2	MB_2	MB_1	MB_1	MB_1	MB_0	MB_0
5		MB_3	MB_3	MB_2	MB_2	MB_2	MB_1	MB_1
6				MB_3	MB_3	MB_3	MB_2	MB_2
7						MB_3	MB_3
8								

Figure 4.6: The Scheduling of the Architecture A

Clock Cycles	CABAD	IQIDCT	DRAM Read	Fetch Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE	
0	MB_0										
1	MB_1	MB_0,b8_0	MB_0								
2		MB_0,b8_1									
3		MB_0,b8_2									
4		MB_0,b8_3									
5	MB_2	MB_1,b8_0	MB_1	MB_0,b8_0	MB_0,b8_0						
6		MB_1,b8_1		MB_0,b8_1	MB_0,b8_1	MB_0,b8_1	MB_0,b8_1	MB_0,b8_1	MB_0,b8_1		
7		MB_1,b8_2		MB_0,b8_2	MB_0,b8_2	MB_0,b8_2	MB_0,b8_2	MB_0,b8_2	MB_0,b8_2		
8		MB_1,b8_3		MB_0,b8_3	MB_0,b8_3	MB_0,b8_3	MB_0,b8_3	MB_0,b8_3	MB_0,b8_3		
9	MB_3	MB_2,b8_0	MB_2	MB_1,b8_0	MB_1,b8_0	MB_0,b8_3	MB_0,b8_3	MB_0,b8_3			
10		MB_2,b8_1		MB_1,b8_1	MB_1,b8_1	MB_1,b8_1	MB_1,b8_1	MB_1,b8_1	MB_1,b8_1	MB_0	MB_0
11		MB_2,b8_2		MB_1,b8_2	MB_1,b8_2	MB_1,b8_2	MB_1,b8_2	MB_1,b8_2	MB_1,b8_2		
12		MB_2,b8_3		MB_1,b8_3	MB_1,b8_3	MB_1,b8_3	MB_1,b8_3	MB_1,b8_3	MB_1,b8_3		
13	...	MB_3,b8_0	MB_3	MB_2,b8_0	MB_2,b8_0	MB_1,b8_3	MB_1,b8_3	MB_1,b8_3			
14	...	MB_3,b8_1		MB_2,b8_1	MB_2,b8_1	MB_2,b8_1	MB_2,b8_1	MB_2,b8_1	MB_2,b8_1	MB_1	MB_1
15	...	MB_3,b8_2		MB_2,b8_2	MB_2,b8_2	MB_2,b8_2	MB_2,b8_2	MB_2,b8_2	MB_2,b8_2		
16	...	MB_3,b8_3		MB_2,b8_3	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3		
17	MB_3,b8_0	MB_3,b8_0	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3			
18	MB_3,b8_1	MB_3,b8_1	MB_3,b8_1	MB_3,b8_1	MB_3,b8_1	MB_3,b8_1	MB_2	MB_2	
19	MB_3,b8_2	MB_3,b8_2	MB_3,b8_2	MB_3,b8_2	MB_3,b8_2	MB_3,b8_2			
20	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3			
21	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3			
22	MB_3	MB_3	
23			
24			
25			
26			

Figure 4.7: The Scheduling of the Architecture B



4.4 Effective Design Space Exploration

Generally, the design space exploration is the process to find the optimal or the near optimal solution among the design alternatives to fulfill the design requirements in terms of performance and cost. More specifically, design space exploration involves (1) finding all the possible design alternatives, (2) evaluating the performance of each design alternative, and (3) determining the best design alternative among all options. Figure 4.12 shows the flow of our design space exploration. The system architecture is partitioned into several specialized components with the four-level memory hierarchies and interoperates as a pipeline to support parallelism. The problem is the optimization of the individual module may not lead to the optimal performance of the overall system. In addition, the level of synchronization granularity between the memory

Clock Cycles	CABAD	IQIDCT	DRAM Read	Fetch Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE	
0	MB_0										
1	MB_1	MB_0,b4_0	MB_0								
2		MB_0,b4_1									
...		...									
16		MB_0,b4_15									
17	MB_2	MB_1,b4_0	MB_1	MB_0,b4_0	MB_0,b4_0						
18		MB_1,b4_1		MB_0,b4_1	MB_0,b4_1	MB_0,b4_1	MB_0,b4_1	MB_0,b4_1	MB_0,b4_1		
...			
32		MB_1,b4_15		MB_0,b4_15	MB_0,b4_15		
33	MB_3	MB_2,b4_0	MB_2	MB_1,b4_0	MB_1,b4_0	MB_0,b4_15	MB_0,b4_15	MB_0,b4_15			
34		MB_2,b4_1		MB_1,b4_1	MB_1,b4_1	MB_1,b4_0	MB_1,b4_0	MB_1,b4_0	MB_0	MB_0	
...		MB_1,b4_1	MB_1,b4_1	MB_1,b4_1			
48		MB_2,b4_15		MB_1,b4_15	MB_1,b4_15			
49	...	MB_3	MB_3	MB_2,b4_0	MB_2,b4_0	MB_1,b4_15	MB_1,b4_15	MB_1,b4_15			
50	MB_3,b4_1			MB_2,b4_1	MB_2,b4_1	MB_2,b4_0	MB_2,b4_0	MB_2,b4_0	MB_1	MB_1	
...	MB_2,b4_1	MB_2,b4_1	MB_2,b4_1			
64	MB_3,b4_15			MB_2,b4_15	MB_2,b4_15			
65	MB_3,b4_0	MB_3,b4_0	MB_2,b4_15	MB_2,b4_15	MB_2,b4_15			
66	MB_3,b4_1	MB_3,b4_1	MB_3,b4_0	MB_3,b4_0	MB_3,b4_0	MB_2	MB_2	
...	MB_3,b4_1	MB_3,b4_1	MB_3,b4_1			
80	MB_3,b4_15	MB_3,b4_15			
81	MB_3,b4_15	MB_3,b4_15	MB_3,b4_15			
82	MB_3	MB_3	
...			
96			
97			
98			

Figure 4.8: The Scheduling of the Architecture C



hierarchies strongly influences the system performance. Therefore, we focus on the design space exploration on two system parameters: (1) the level of synchronization granularity and (2) the design combinations within the video pipe.

The second part of the design space exploration is the evaluation of the large number of potential design alternatives such that the design variable space can be translated into the performance space. In order to examine the effect on the whole system, we implement the design alternatives in the proposed architecture and present a method to evaluate the system performance. The performance evaluation is based on (1) throughput, (2) hardware cost including logic circuit and the memory unit, and (3) cost-normalized hardware utilization. Specifically, we utilize the transaction level modeling (TLM) to evaluate the throughput performance because exploiting a wide range of design trade-off at register transfer level (RTL) is impractical

Clock Cycles	CABAD	IQIDCT	DRAM Read	Fetch Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE
0	MB_0									
1	MB_1	MB_0,b8_0	MB_0,b8_0							
2		MB_0,b8_1	MB_0,b8_1	MB_0,b8_0	MB_0,b8_0					
3		MB_0,b8_2	MB_0,b8_2	MB_0,b8_1	MB_0,b8_1	MB_0,b8_0	MB_0,b8_0	MB_0,b8_0		
4		MB_0,b8_3	MB_0,b8_3	MB_0,b8_2	MB_0,b8_2	MB_0,b8_1	MB_0,b8_1	MB_0,b8_1	MB_0,b8_0	
5	MB_2	MB_1,b8_0	MB_1,b8_0	MB_0,b8_3	MB_0,b8_3	MB_0,b8_2	MB_0,b8_2	MB_0,b8_2	MB_0,b8_1	
6		MB_1,b8_1	MB_1,b8_1	MB_1,b8_0	MB_1,b8_0	MB_0,b8_3	MB_0,b8_3	MB_0,b8_3	MB_0,b8_2	
7		MB_1,b8_2	MB_1,b8_2	MB_1,b8_1	MB_1,b8_1	MB_1,b8_0	MB_1,b8_0	MB_1,b8_0	MB_0,b8_3	MB_0
8	MB_3	MB_1,b8_3	MB_1,b8_3	MB_1,b8_2	MB_1,b8_2	MB_1,b8_1	MB_1,b8_1	MB_1,b8_1	MB_1,b8_0	
9		MB_2,b8_0	MB_2,b8_0	MB_1,b8_3	MB_1,b8_3	MB_1,b8_2	MB_1,b8_2	MB_1,b8_2	MB_1,b8_1	
10		MB_2,b8_1	MB_2,b8_1	MB_2,b8_0	MB_2,b8_0	MB_1,b8_3	MB_1,b8_3	MB_1,b8_3	MB_1,b8_2	
11		MB_2,b8_2	MB_2,b8_2	MB_2,b8_1	MB_2,b8_1	MB_2,b8_0	MB_2,b8_0	MB_2,b8_0	MB_1,b8_3	MB_1
12	...	MB_2,b8_3	MB_2,b8_3	MB_2,b8_2	MB_2,b8_2	MB_2,b8_1	MB_2,b8_1	MB_2,b8_1	MB_2,b8_0	
13		MB_3,b8_0	MB_3,b8_0	MB_2,b8_3	MB_2,b8_3	MB_2,b8_2	MB_2,b8_2	MB_2,b8_2	MB_2,b8_1	
14		MB_3,b8_1	MB_3,b8_1	MB_3,b8_0	MB_3,b8_0	MB_2,b8_3	MB_2,b8_3	MB_2,b8_3	MB_2,b8_2	
15	...	MB_3,b8_2	MB_3,b8_2	MB_3,b8_1	MB_3,b8_1	MB_3,b8_0	MB_3,b8_0	MB_3,b8_0	MB_2,b8_3	MB_2
16		MB_3,b8_3	MB_3,b8_3	MB_3,b8_2	MB_3,b8_2	MB_3,b8_1	MB_3,b8_1	MB_3,b8_1	MB_3,b8_0	
17		MB_3,b8_3	MB_3,b8_3	MB_3,b8_2	MB_3,b8_2	MB_3,b8_2	MB_3,b8_1	
18	MB_3,b8_3	MB_3,b8_3	MB_3,b8_3	MB_3,b8_2	
19						MB_3,b8_3	MB_3
20									...	
21										
22										
23										

Figure 4.9: The Scheduling of the Architecture D



due to the poor simulation performance. The hardware cost is obtained from the reports of our RTL synthesis and reference designs.

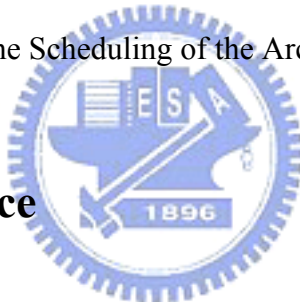
The third part of the design space exploration is to determine the optimal solution within the explored design space. We use the hierarchical design space exploration to reduce the design space variables. More specifically, we firstly search the optimal configuration for the high level of system parameter, i.e., the level of synchronization granularity. We then find the optimal design combination for a specified level of synchronization granularity. After acquiring the data of each performance metric, we use the Pareto analysis to solve the multi-objective optimization and to facilitate decision-making in the system architecture development [62].

Clock Cycles	CABAD	IQIDCT	DRAM Read	Fetch Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE
0	MB_0									
1	MB_1	MB_0,b4_0	MB_0,b8_0							
...		...								
4		MB_0,b4_3								
5		MB_0,b4_4	MB_0,b8_1	MB_0,b4_0	MB_0,b4_0					
...		MB_0,b4_0	MB_0,b4_0	MB_0,b4_0		
8		MB_0,b4_7		MB_0,b4_3	MB_0,b4_3		
9		MB_0,b4_8	MB_0,b8_2	MB_0,b4_4	MB_0,b4_4	MB_0,b4_3	MB_0,b4_3	MB_0,b4_3		
...		MB_0,b4_4	MB_0,b4_4	MB_0,b4_4	MB_0,b8_0	
12		MB_0,b4_11		MB_0,b4_7	MB_0,b4_7		
13		MB_0,b4_12	MB_0,b8_3	MB_0,b4_8	MB_0,b4_8	MB_0,b4_7	MB_0,b4_7	MB_0,b4_7		
...		MB_0,b4_8	MB_0,b4_8	MB_0,b4_8	MB_0,b8_1	
16		MB_0,b4_15		MB_0,b4_11	MB_0,b4_11		
17	MB_2	MB_1,b4_0	MB_1,b8_0	MB_0,b4_12	MB_0,b4_12	MB_0,b4_11	MB_0,b4_11	MB_0,b4_11		
...		MB_0,b4_12	MB_0,b4_12	MB_0,b4_12	MB_0,b8_2	
20		MB_1,b4_3		MB_0,b4_15	MB_0,b4_15		
21		MB_1,b4_4	MB_1,b8_1	MB_1,b4_0	MB_1,b4_0	MB_0,b4_15	MB_0,b4_15	MB_0,b4_15		
...		MB_1,b4_0	MB_1,b4_0	MB_1,b4_0	MB_0,b8_3	MB_0
24		MB_1,b4_7		MB_1,b4_3	MB_1,b4_3		
25		MB_1,b4_8	MB_1,b8_2	MB_1,b4_4	MB_1,b4_4	MB_1,b4_3	MB_1,b4_3	MB_1,b4_3		
...		MB_1,b4_4	MB_1,b4_4	MB_1,b4_4	MB_1,b8_0	
28		MB_1,b4_11		MB_1,b4_7	MB_1,b4_7		
29		MB_1,b4_1	MB_1,b8_3	MB_1,b4_8	MB_1,b4_8	MB_1,b4_7	MB_1,b4_7	MB_1,b4_7		
...		MB_1,b4_8	MB_1,b4_8	MB_1,b4_8	MB_1,b8_1	
32		MB_1,b4_15		MB_1,b4_11	MB_1,b4_11		
33	MB_3	MB_2,b4_0	MB_2,b8_0	MB_1,b4_12	MB_1,b4_12	MB_1,b4_11	MB_1,b4_11	MB_1,b4_11		
...		MB_1,b4_12	MB_1,b4_12	MB_1,b4_12	MB_1,b8_2	
36		MB_2,b4_3		MB_1,b4_15	MB_1,b4_15		
37		MB_2,b4_4	MB_2,b8_1	MB_2,b4_0	MB_2,b4_0	MB_1,b4_15	MB_1,b4_15	MB_1,b4_15		
...		MB_2,b4_0	MB_2,b4_0	MB_2,b4_0	MB_1,b8_3	MB_1
40		MB_2,b4_7		MB_2,b4_3	MB_2,b4_3		
41		MB_2,b4_8	MB_2,b8_2	MB_2,b4_4	MB_2,b4_4	MB_2,b4_3	MB_2,b4_3	MB_2,b4_3		
...		MB_2,b4_4	MB_2,b4_4	MB_2,b4_4	MB_2,b8_0	
44		MB_2,b4_11		MB_2,b4_7	MB_2,b4_7		
45		MB_2,b4_1	MB_2,b8_3	MB_2,b4_8	MB_2,b4_8	MB_2,b4_7	MB_2,b4_7	MB_2,b4_7		
...		MB_2,b4_8	MB_2,b4_8	MB_2,b4_8	MB_2,b8_1	
48		MB_2,b4_15		MB_2,b4_11	MB_2,b4_11		
49	...	MB_3,b4_0	MB_3,b8_0	MB_2,b4_12	MB_2,b4_12	MB_2,b4_11	MB_2,b4_11	MB_2,b4_11		
...		MB_2,b4_12	MB_2,b4_12	MB_2,b4_12	MB_2,b8_2	
52		MB_3,b4_3		MB_2,b4_15	MB_2,b4_15		
53		MB_3,b4_4	MB_3,b8_1	MB_3,b4_0	MB_3,b4_0	MB_2,b4_15	MB_2,b4_15	MB_2,b4_15		
...		MB_3,b4_0	MB_3,b4_0	MB_3,b4_0	MB_2,b8_3	MB_2
56		MB_3,b4_7		MB_3,b4_3	MB_3,b4_3		
57		MB_3,b4_8	MB_3,b8_2	MB_3,b4_4	MB_3,b4_4	MB_3,b4_3	MB_3,b4_3	MB_3,b4_3		
...		MB_3,b4_4	MB_3,b4_4	MB_3,b4_4	MB_3,b8_0	
60		MB_3,b4_11		MB_3,b4_7	MB_3,b4_7		
61		MB_3,b4_12	MB_3,b8_3	MB_3,b4_8	MB_3,b4_8	MB_3,b4_7	MB_3,b4_7	MB_3,b4_7		
...		MB_3,b4_8	MB_3,b4_8	MB_3,b4_8	MB_3,b8_1	
64		MB_3,b4_15		MB_3,b4_11	MB_3,b4_11		
65		MB_3,b4_12	MB_3,b4_12	MB_3,b4_11	MB_3,b4_11	MB_3,b4_11		
...		MB_3,b4_12	MB_3,b4_12	MB_3,b4_12	MB_3,b8_2	
68				MB_3,b4_15	MB_3,b4_15		
69				MB_3,b4_15	MB_3,b4_15	MB_3,b4_15		
...				MB_3,b8_3	MB_3
72										
73										
...										
76										
77										
...										
80										
81										
...										
84										
85										
...										

Figure 4.10: The Scheduling of the Architecture E

Clock Cycles	CABAD	IQIDCT	DRAM Read	Fetch Data via AHB	IIP	Deblocking	Refinement	Send Data via AHB	DRAM Write	CABAE
0	MB_0									
1	MB_1	MB_0,b4_0	MB_0,b4_0							
2		MB_0,b4_1	MB_0,b4_1	MB_0,b4_0	MB_0,b4_0					
...		MB_0,b4_1	MB_0,b4_1	MB_0,b4_0	MB_0,b4_0	MB_0,b4_0		
16		MB_0,b4_15	MB_0,b4_15	MB_0,b4_1	MB_0,b4_1	MB_0,b4_1	MB_0,b4_0	
17	MB_2	MB_1,b4_0	MB_1,b4_0	MB_0,b4_15	MB_0,b4_15	MB_0,b4_1	
18		MB_1,b4_1	MB_1,b4_1	MB_1,b4_0	MB_1,b4_0	MB_0,b4_15	MB_0,b4_15	MB_0,b4_15	...	
...		MB_1,b4_1	MB_1,b4_1	MB_1,b4_0	MB_1,b4_0	MB_1,b4_0	MB_0,b4_15	MB_0
32	MB_1,b4_15	MB_1,b4_15	MB_1,b4_1	MB_1,b4_1	MB_1,b4_1	MB_1,b4_0		
33	MB_3	MB_2,b4_0	MB_2,b4_0	MB_1,b4_15	MB_1,b4_15	MB_1,b4_1	
34		MB_2,b4_1	MB_2,b4_1	MB_2,b4_0	MB_2,b4_0	MB_1,b4_15	MB_1,b4_15	MB_1,b4_15	...	
...		MB_2,b4_1	MB_2,b4_1	MB_2,b4_0	MB_2,b4_0	MB_2,b4_0	MB_1,b4_15	MB_1
48	MB_2,b4_15	MB_2,b4_15	MB_2,b4_1	MB_2,b4_1	MB_2,b4_1	MB_2,b4_0		
49	...	MB_3,b4_0	MB_3,b4_0	MB_2,b4_15	MB_2,b4_15	MB_2,b4_1	
50		MB_3,b4_1	MB_3,b4_1	MB_3,b4_0	MB_3,b4_0	MB_2,b4_15	MB_2,b4_15	MB_2,b4_15	...	
...		MB_3,b4_1	MB_3,b4_1	MB_3,b4_0	MB_3,b4_0	MB_3,b4_0	MB_2,b4_15	MB_2
64	MB_3,b4_15	MB_3,b4_15	MB_3,b4_1	MB_3,b4_1	MB_3,b4_1	MB_3,b4_0		
65	MB_3,b4_15	MB_3,b4_15	MB_3,b4_1		
66	MB_3,b4_15	MB_3,b4_15	MB_3,b4_15	...		
...	MB_3,b4_15	MB_3	
80		
81		
82		
...	

Figure 4.11: The Scheduling of the Architecture F



4.5 Pruned Design Space

After the architecture is partitioned as a pipelined system with multi-level memory hierarchy, the granularity of synchronization buffer within the memory hierarchy and the various combinations of the internal modules are the system level design challenges. Thus, these two design concerns derive lots of design alternatives.

4.5.1 Exploration of the Synchronization Granularity

The granularity of the synchronization buffer immediately affects the pipelined system performance because it is relevant to the performance for the DRAM access and the motion compensation of variable block size. First, the granularity of the synchronization buffer is a dominant factor to the occurrence of row-miss for DRAM access. The video pipe module repeatedly

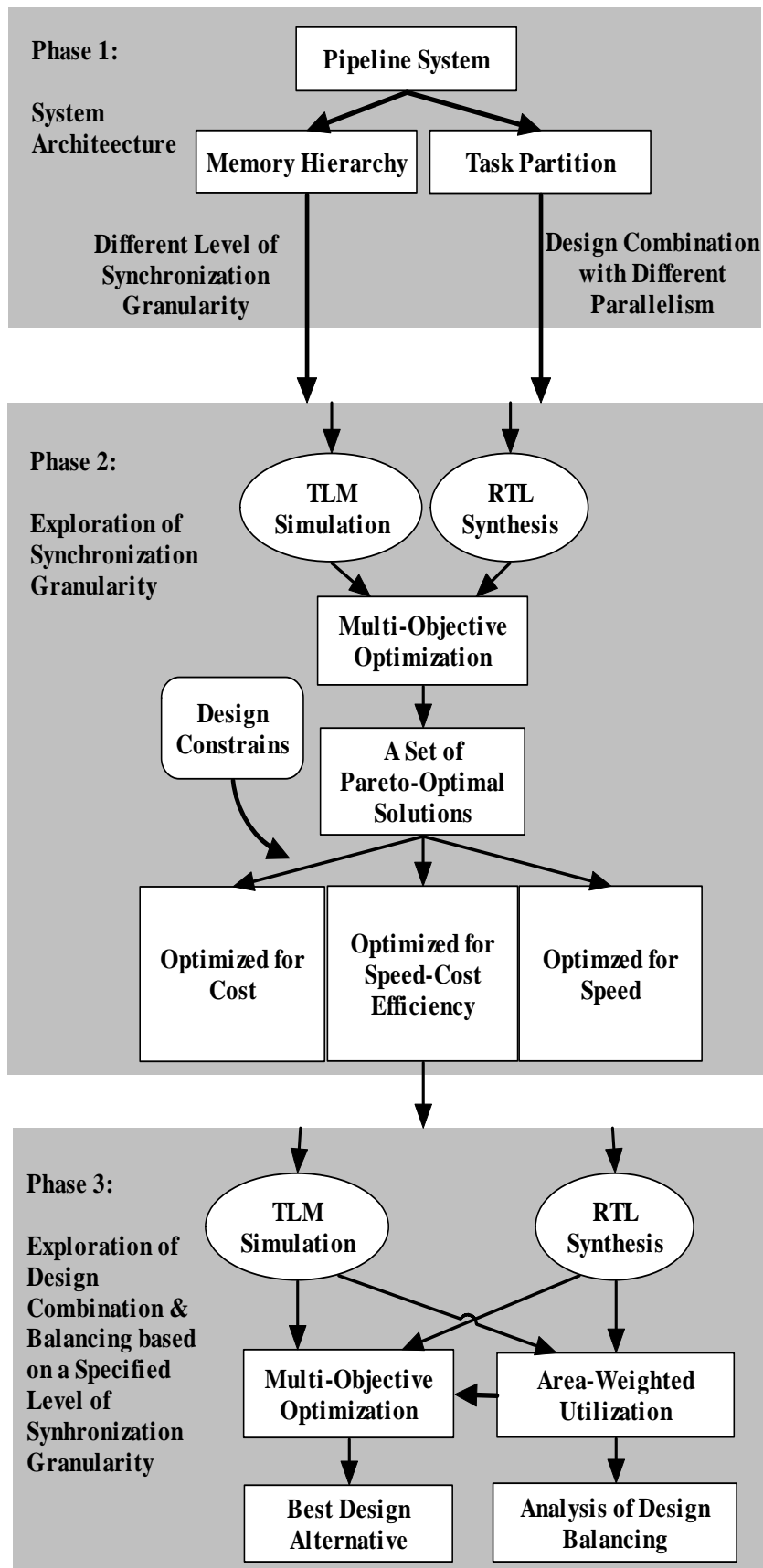


Figure 4.12: The Flow of Our Design Space Exploration

fetches the motion-compensated pixels from a certain row of DRAM and stores the filtered pixels into another row of DRAM such that the row-miss always occurs at the transition from data fetch to data storage. In the case of the row-miss, extra DRAM cycles are introduced because of the pre-charge of the original row, the activation of another row, and the CAS latency [46]. In addition to the DRAM cycles, the granularity of the synchronization buffer affects the amount of data transmission via AHB data bus because the motion compensation is of variable block size and requires 6-tap interpolation. Particularly, if the granularity of the video pipe is finer than the size of a block partition, redundant transmission is required due to the operations of the 6-tap interpolation. Generally, coarsening the granularity of the synchronization buffer decreases the synchronization overhead at the expense of more memory.

To quantitatively analyze the tradeoff between the synchronization overhead and the memory cost, we explore the granularity of the memory-subsystem and the video pipe at multiple levels such as 4x4, 8x8, and 16x16. During the transcoding process, the data exchange mainly takes place at the two levels of the proposed memory hierarchies as shown in Figure 4.13. The synchronization buffer in the memory sub-system is responsible for the off-chip data exchange. The ping-pong memories between modules are responsible for the on-chip data exchange. The frequency of data exchange has great impact on the efficiency of data access. For this reason, we explore the granularity of memory-subsystem (GM) and the granularity of video pipe (GV) at multiple levels such as 4x4, 8x8, and 16x16. In particular, the GV should be finer than the GM because the GM is at the higher level in the memory hierarchy. Therefore, by analyzing the level of synchronization granularity, we explore six architectural candidates as mentioned in Table 4.2.

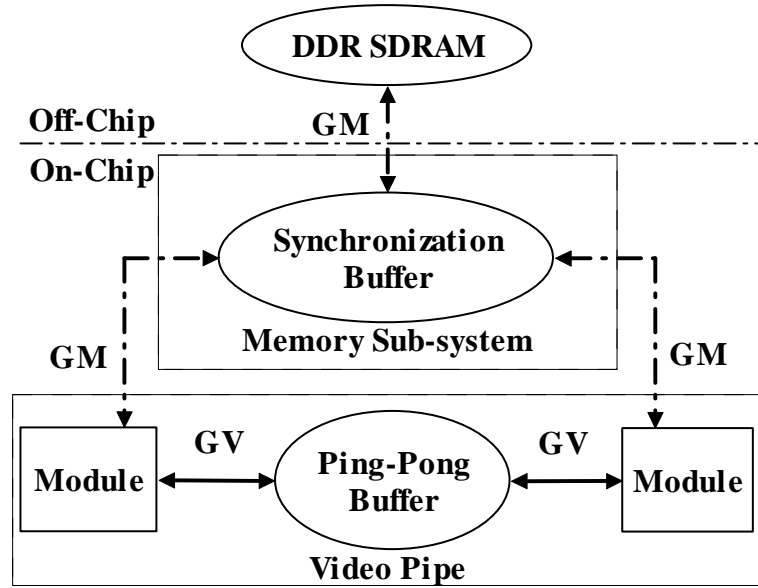


Figure 4.13: The Exploration of the Synchronization Granularity

4.5.2 Exploration of the Design Combination and Balancing

The various design alternatives among the modules in the video pipe can have impact to the system performance in terms of the bus utilization, the hardware utilization, and the system throughput due to the bus arbitration and the pipeline synchronization issues. Specifically, the pipeline is synchronized by the slowest module in the video pipe as follows.

$$P[i] = \arg \max_{1 \leq j \leq M} (E_j[i]) \quad (4.1)$$

where $P[i]$ is the cycle count of the system at the i -th pipelined stage and $E_j[i]$ is the execution cycle count of the j -th component at the i -th pipelined stage. In particular, each module consumes different cycles at each pipeline stage according to various coding characteristic such

that the synchronization overhead is introduced as follows.

$$S = \frac{\sum_{i=1}^N P[i]}{N} \geq \arg_{1 \leq j \leq M} \left(\frac{\sum_{i=1}^N E_j[i]}{N} \right) \quad (4.2)$$

where S is the throughput performance of the system and N is the number of the pipelined stage.

The gains of task-level parallelism could be reduced by the imbalanced design combination. In the prior designs, the individual performance is the main focus with the assumption that the individual module can always have immediate and unlimited access to the data bus, which is not realistic in the design with only one data bus. Therefore, we examine the effects of different design combinations with respect to system performance in consideration of the transform module, the prediction module, and the deblocking filter.

We explore 2 design alternatives for the transform module (IQIDCT) with different processing rates. Specifically, the design in [44] processes 4 pixels per cycle, while the other design can increase the processing rate up to 8 pixels per cycle at the expense of more hardware cost [45]. In the simulations, we use the terms of LT and HT to denote the low-throughput and the high-throughput designs respectively.

We explore 4 design alternatives for the prediction module (IIP), which can be classified into separated adder-tree (AT) based [44][47] and combined systolic-array (SA) based approaches. The separated AT-based design implements inter Luma/Chroma prediction and intra prediction by three separated circuits with more hardware cost. Although the AT-based architecture has higher processing throughput for the 2-D interpolation, it suffers from the poor hardware utilization for the 1-D interpolation. On the other hand, our SA-based design combines inter and intra prediction into one unified circuit with less hardware cost. Furthermore, our scalable architec-

ture has several levels of parallelisms where the number of systolic arrays can be reconfigured as one, two, or three levels of parallel structure. In the following, we represent the 4 alternatives as AT, SA_1, SA_2, and SA_3 respectively.

We explore 20 design alternatives for the deblocking filter (DB), where the exploration includes the filtering order, the processing granularity, and the memory hierarchy. Firstly, the filtering order can be categorized as either the MB-based and or the B8-based. Secondly, for more efficient DRAM access, we can transfer the filtered pixels of one MB in the same pipelined stage at the expense of more memory. Thirdly, to further reduce the bus bandwidth and the processing latency, a single-port SRAM is used as a line buffer to store a frame-wide row of 4x4 blocks. Therefore, there are 20 alternatives among which 17 of them are MB-based [48][49][50][51][52][53][54][55][56][57][58][59] approaches and the other 3 are B8-based approaches. In summary, our design space exploration covers a total of 160 ($2*4*20$) combinations.



4.6 Evaluation of the System Performance

This section presents how the system performance is evaluated. In particular, the several evaluation metrics are introduced in our simulation infrastructure. The optimized design alternative is decided based on the concept of Pareto-based multi-objective optimization.

4.6.1 Evaluation Metric

To evaluate the system performance, we consider the following design metrics: (1) processing speed, (2) silicon area, and (3) hardware utilization. The processing speed is used to measure the system throughput. To measure the speed, we use the average execution cycle counts of one pipelined stage which is dominated by the slowest module in the video pipe. In particular, the

pipeline is dominated by the following four modules: IQIDCT, IIP, DB, and DRAM access.

The silicon area is used to measure the hardware cost. Although the actual chip size is not available until the chip is taped out, we use the equivalent gate count to estimate the silicon area at the system level. In addition, the equivalent gate count includes the cost of the local memories and the logic circuits. One of the problems in [48][49][50][51][52][53][54][55][56][57][58][59] is that the cost of memory units and the cost of logic circuit are not normalized to the same scale. Specifically, the cost of memory is measured by the number of storage bits while the cost of logic circuit is measured by the synthesized gate count. In some cases, the smaller logic circuit could be obtained at the expense of more memory cost [51][57]. Therefore, we should normalize these two cost factors to the equivalent gate count for a fair comparison.

The hardware utilization is used to evaluate the efficiency of a design and to determine how the explored modules are utilized. The higher the hardware utilization, the lower the overheads in the individual design. Originally, the utilization is the ratio of the computational cycles to the execution cycles of the system. However, the problem is that the factor is not scaled according to its cost. The unbalanced design alternative increases the utilization for the low-cost module while the other high-cost modules are significantly underutilized. Because the multiple components jointly determine the system utilization, the cost of each module should also consider the impact of the utilization on the overall performance.

We use the average utilization normalized by the cost to represent the balance of a design combination. Let X be the random variable that represents how the hardware cost is utilized at the i -th cycle among the M components to be explored.

$$x[i] = \frac{\sum_{j=1}^M C_j \cdot u_j[i]}{\sum_{j=1}^M C_j} = \frac{\sum_{j=1}^M C_j \cdot u_j[i]}{C} \quad (4.3)$$

where the symbols C_j and $u_j[i]$ represent the hardware cost and utilization of the j -th component at the i -th cycle, respectively. The symbol C represents the overall cost of the M components. We further calculate the mean value of the symbol X throughout the total execution period of N cycles as follows.

$$\mu_X = \frac{\sum_{i=1}^N x_i[i]}{N} = \frac{\sum_{i=1}^N \sum_{j=1}^M C_j \cdot u_j[i]}{C \cdot N} = \sum_{j=1}^M \left[\left(\frac{C_j}{C} \right) \cdot \frac{\sum_{i=1}^N u_j[i]}{N} \right] = \sum_{j=1}^M \left[\left(\frac{C_j}{C} \right) \cdot \mu_{U_j} \right] \quad (4.4)$$

where $\left(\frac{C_j}{C} \right)$ is the proportional cost of the j -th component. The average hardware utilization μ_{U_j} is the hardware utilization for each individual component. A designer should keep the balance among modules in a pipelined system such that the hardware can be more efficiently utilized for improving the system throughput. Therefore, we use the area-weighted hardware utilization, which is the summation of product of proportional cost and utilization, to represent the overall balanced system performance.



4.6.2 Simulation Infrastructure

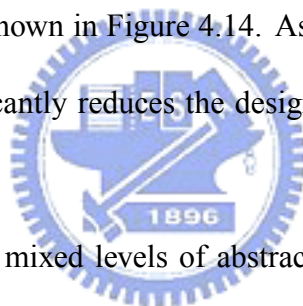
To effectively evaluate the system performance for the numerous design alternatives, we employ the transaction level modeling (TLM) technique to increase the simulation speed. Particularly, the entire system is modeled at mixed levels of abstraction and simulated using CoWare ConvergenSCTM [60].

4.6.2.1 System Simulation with Mixed Levels of Abstraction

Traditionally, the coding effort of the lower level of abstraction, such as register transfer level (RTL), is costly and its simulation is very time consuming. While we need to check if the DRAM access violates the timing constraints at each cycle, the memory sub-system including

DRAM and DRAM controller is simulated at cycle-accurate RTL level using Verilog.

On the other hand, the remaining parts of the system are modeled at a higher level of abstraction, known as the transaction level, with SystemC [61] to achieve faster simulation speed. In the SystemC implementation, the interface between modules and the granularity for data exchange are defined. To synchronize the modules at different levels of abstraction, the data exchanges are triggered by the call of the interface functions bound to the ports of a module and carried out by the implementation of the channels that connects the ports of different modules. In addition, the functionalities within a module are described by a set of concurrent processes with annotated timing information. Specifically, within a module, the task modeling is accomplished by the task execution followed by N waiting cycles which is annotated from the associated hardware architecture as shown in Figure 4.14. As compared to the RTL, the transaction level modeling (TLM) significantly reduces the design regression cycle while offering sufficient simulation accuracy.



In summary, the simulation with mixed levels of abstraction allows a system designer to find the best trade-off between the simulation speed and the modeling accuracy. The mixed level of abstraction not only considers the impact of external memory but also has a faster simulation speed than a pure RTL simulation such that we can effectively perform the architectural evaluation.

4.6.2.2 Platform Creation

After the model creation, all the modules in the proposed H.264/AVC decoder are assembled and simulated on the Platform Architect of Coware's ConvergenSC™, as shown in Figure 4.15. Particularly, the external memory interface modeled in the Verilog is instantiated in the SystemC-based platform using a proxy module, which acts as a wrapper and deals with the

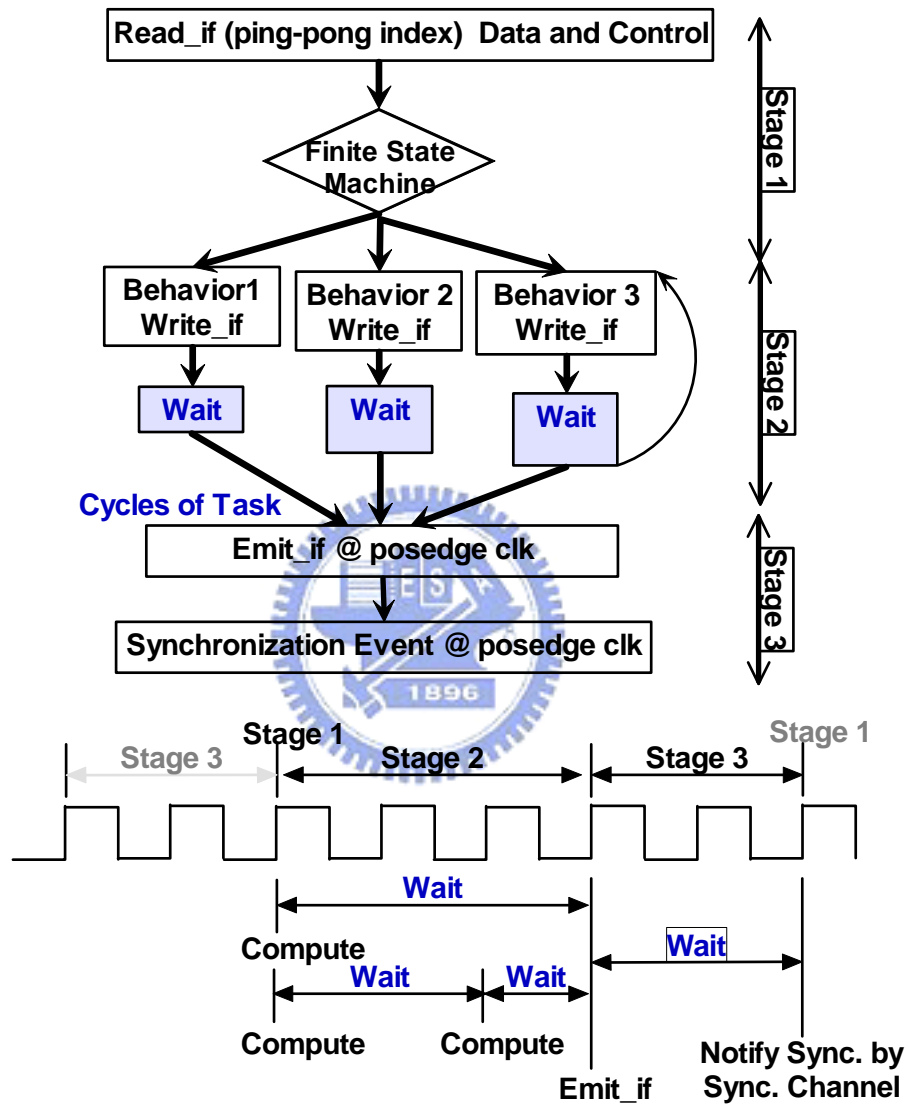


Figure 4.14: The SystemC Implementation with the Annotated Timing

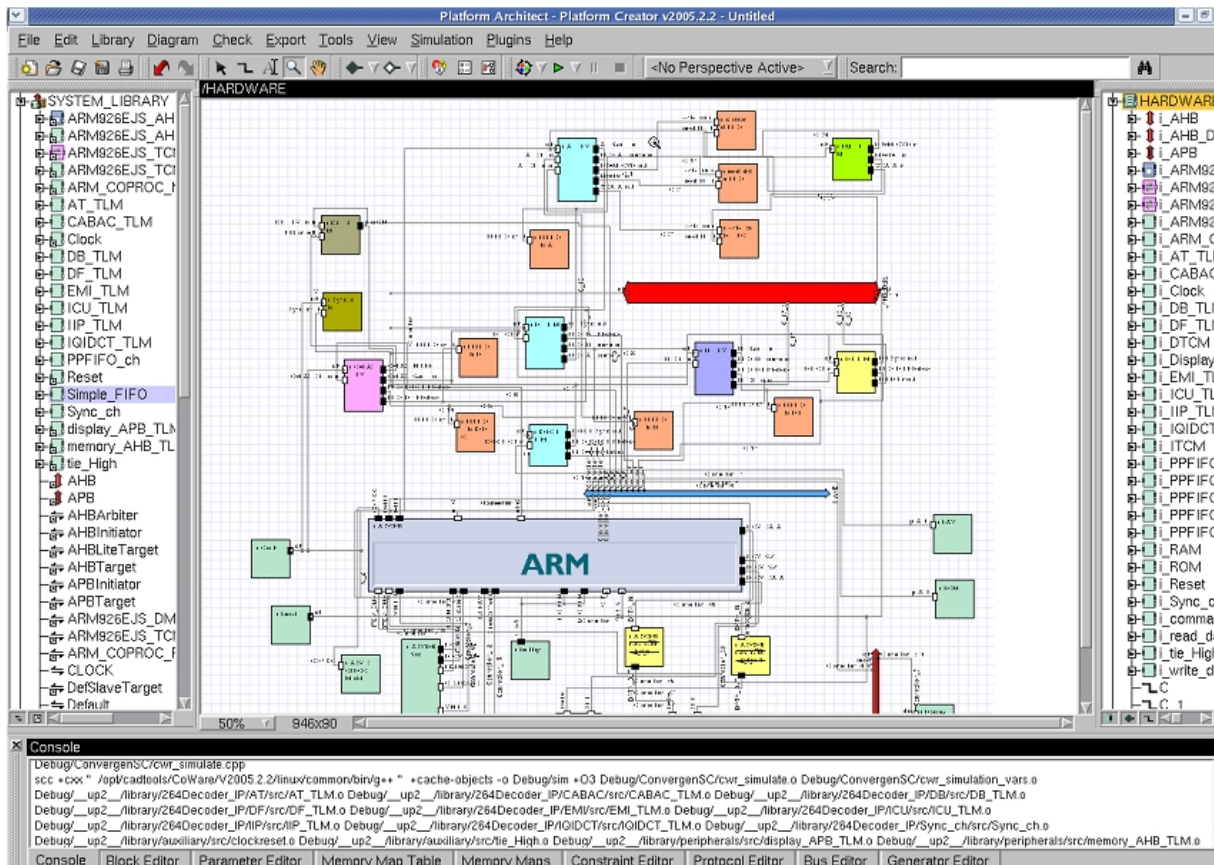


Figure 4.15: The Proposed H.264/AVC Video Embedding Transcoder and Decoder Implemented by the Platform Architect of CoWare ConvergenSC

signal exchange across the two different simulators. Compared with traditional system simulation at register transfer level, the transaction level modeling significantly reduce the design regression cycle while offering sufficient simulation accuracy.

4.6.3 Pareto-Based Multi-Objective Optimization

With these performance metrics, which can be viewed as the design objectives, we are able to translate the design space variable into performance space and use the Pareto-based multi-objective optimization to determine the optimality of a given performance space. Generally, the goal of the design tradeoff is to find an optimal solution that has the lowest hardware cost and smallest execution cycle count. However, the processing speed is typically improved at

the expense of more hardware cost. Moreover, some poor design alternatives lead to the more execution cycle and hardware cost. To resolve the conflicting objectives, we use Pareto analysis to find a set of Pareto-optimal alternatives with multi-dimensional optimization in terms of the execution cycle count, the hardware cost, and the normalized hardware utilization. These Pareto-optimal alternatives can be further pruned when considering certain design constraints such as required processing throughput and limited hardware cost.

4.7 Simulation Results and Analysis

In this section, the simulation results of our design space exploration are provided. We use the simulation-based exploration because we would like to optimize the system performance for the average case instead of the worst case. Our TLM simulation framework creates a large volume of simulation data to explore the system at a high level of abstraction with annotated timing and use the metrics described in Section 4.6 to evaluate the system performance.

4.7.1 Pareto Analysis for the Exploration of Synchronization Granularity

To find the optimized granularity of the synchronization, we depict the Pareto analysis in Figure 4.16 and Figure 4.17 where 25 alternatives are placed on the plot according to the cycle count and the equivalent gate count. In our analysis, the cycle count is obtained by our TLM simulation framework while the gate count is obtained from the synthesis of a very rough RTL coding and the reports in [47]. The equivalent gate count only includes the cost of hardware to be explored, i.e. the ping-pong memories, the synchronization buffer, and the inter and intra prediction. The Pareto frontier which defines a set of Pareto-optimal solutions is drawn by connecting the points that are not dominated by other points and the utopia point is defined as the point in performance space that is best in every objective [62].

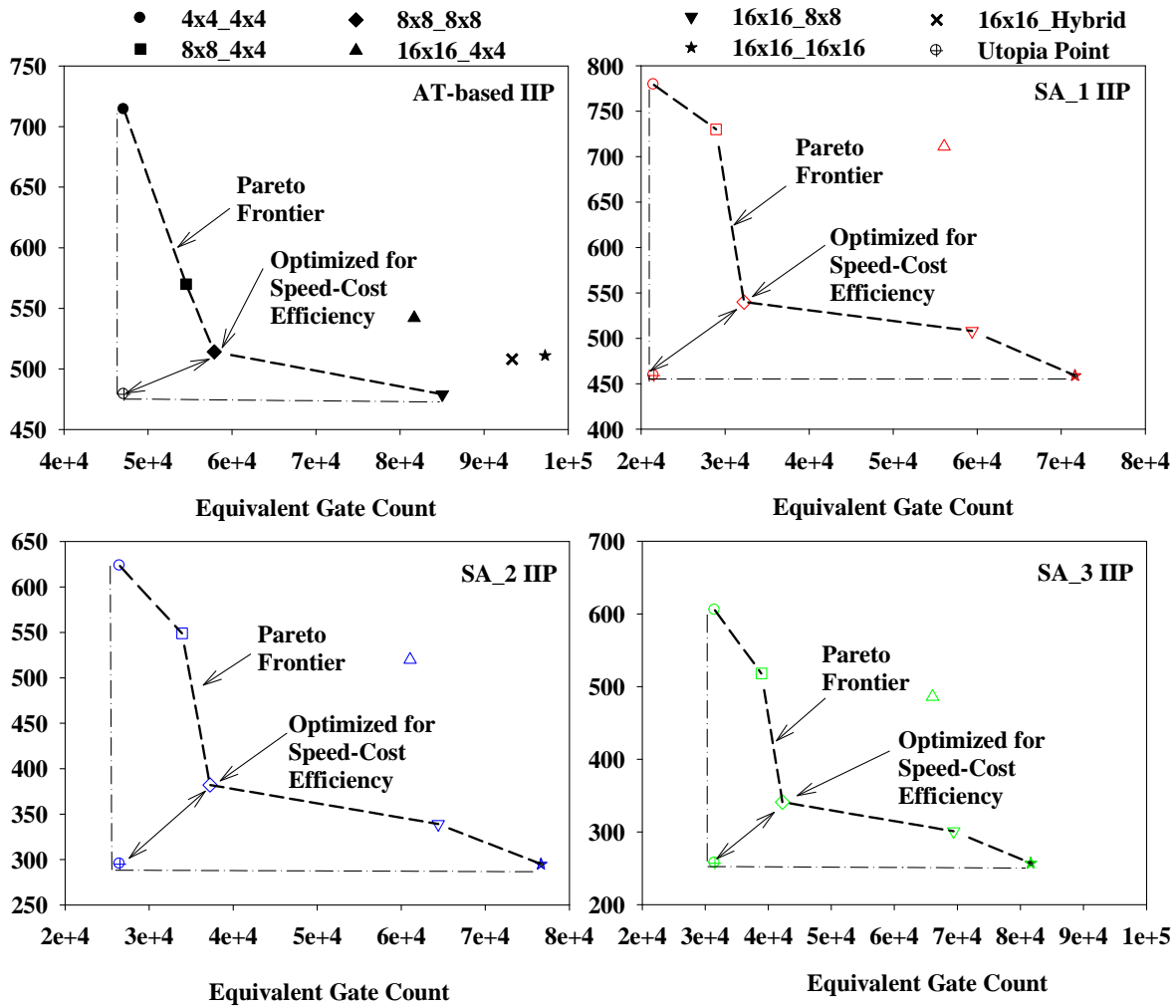


Figure 4.16: The Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count at Different Levels of Synchronization Granularity and for the Different Designs of the Inter and Intra Prediction. The Combination of the Level of Synchronization Granularity to be Explored Includes 16x16_16x16, 16x16_8x8, 16x16_4x4, 8x8_8x8, 8x8_4x4, and 4x4_4x4 where the Terms before and after the Underscore Indicate the GM and the GV, Respectively

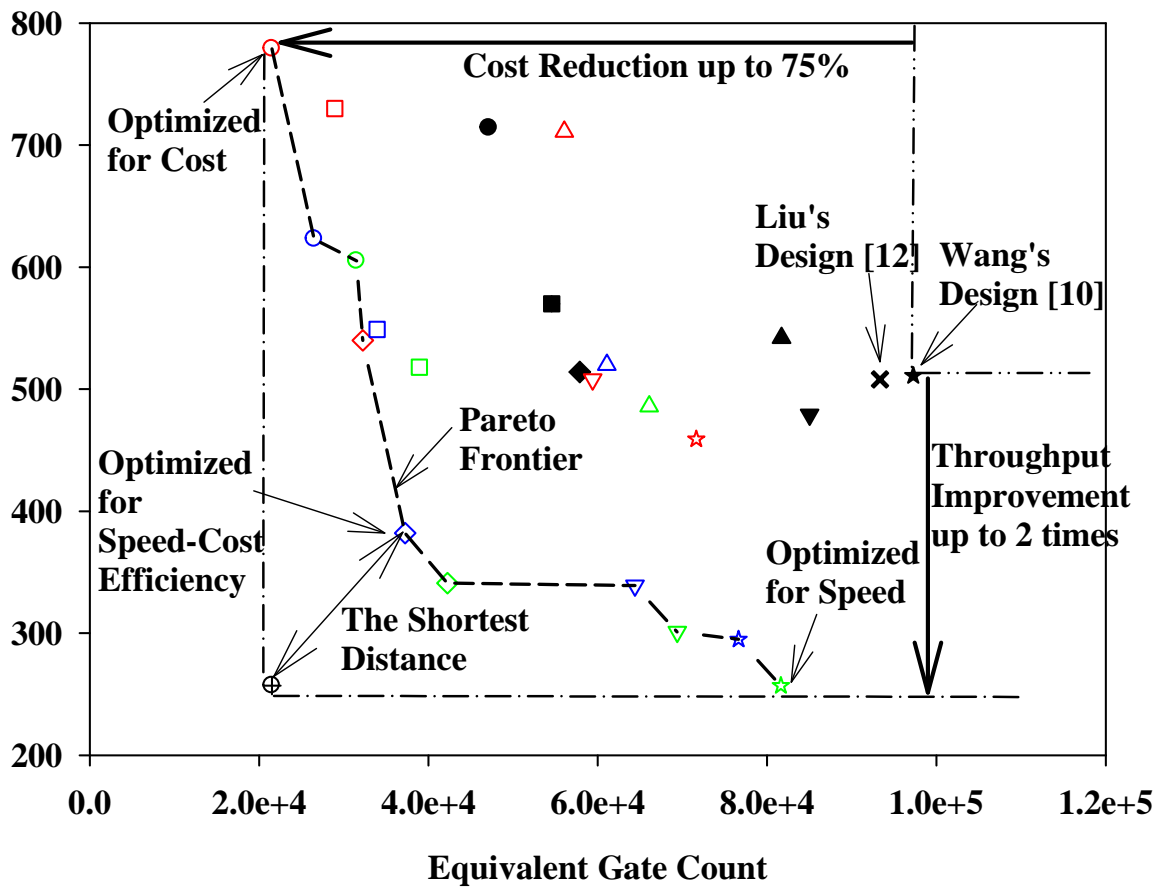


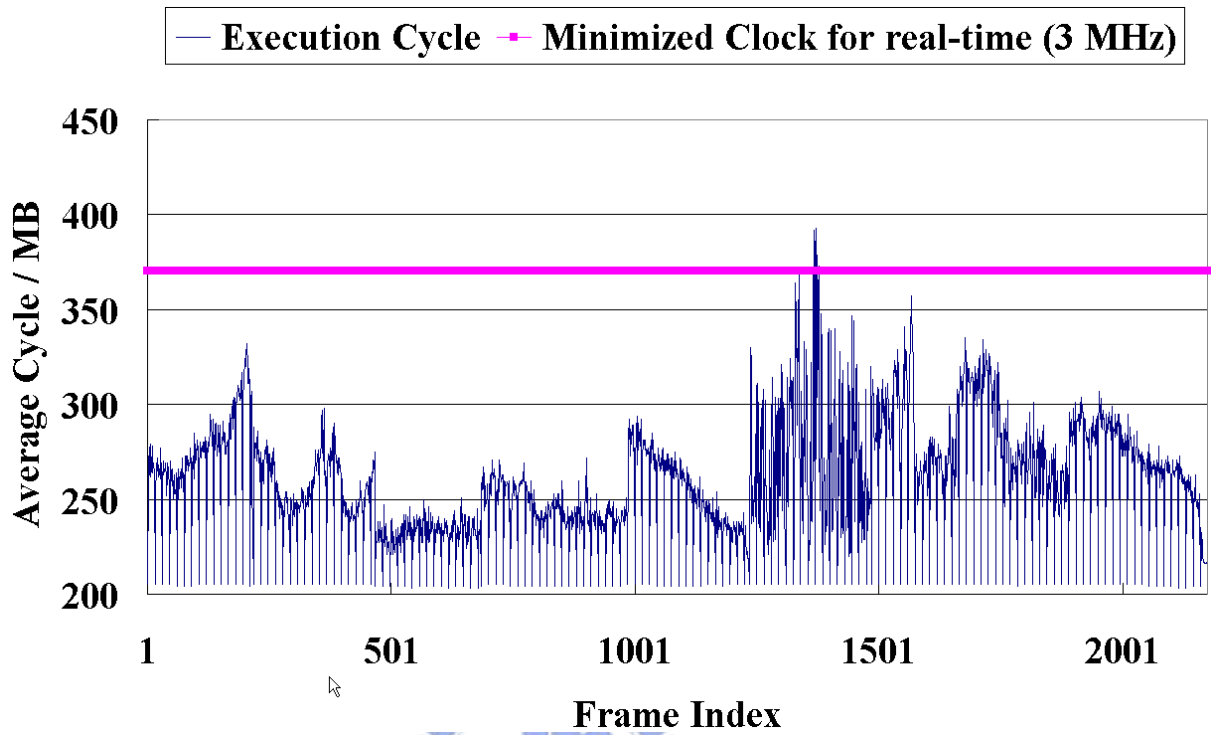
Figure 4.17: The Combined Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count at Different Levels of Synchronization Granularity and for the Different Designs of the Inter and Intra Prediction

Table 4.3: The Bitrate of Each Long Sequence with Different Resolution

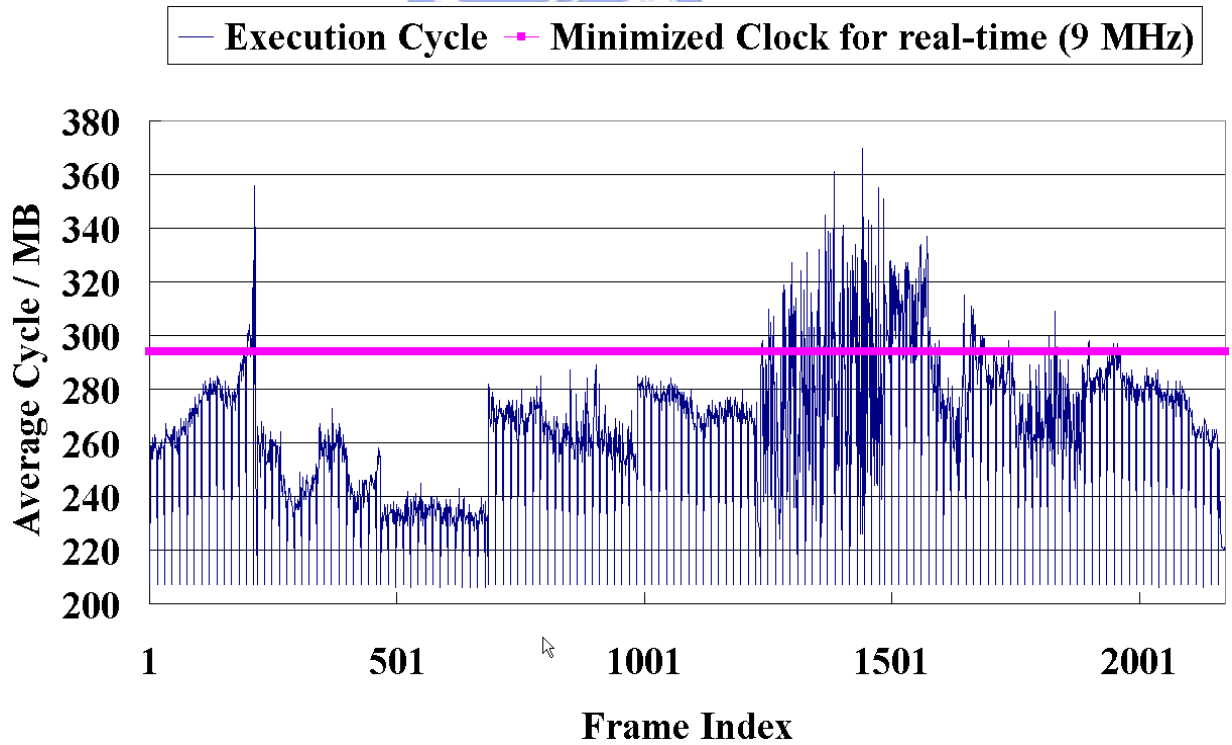
Frame Index	Sequence	1920x1088 @ 60Hz	960x544 @ 60Hz	480x272 @ 60Hz	240x144 @ 60Hz
1 - 217	Blue_Sky	13.39	5.78	2.74	1.14
218 - 517	Pedestrian_Area	12.83	5.24	2.16	0.85
518 - 767	Riverbed	13.81	3.19	0.97	0.24
768 - 1067	Rush_Hour	14.51	5.53	2.12	0.78
1068 - 1367	Station2	15.45	5.23	2.29	0.71
1368 - 11667	Sunflower	16.19	6.54	2.95	1.31
1668 - 2357	Tractor	13.98	5.50	2.23	0.81
Average		14.30	4.94	2.05	0.77

The design alternative with the highest speed-cost efficiency is the point that has the shortest distance to the utopia point and locates on the Pareto frontier. Therefore, when the GM and GV are both synchronized at the level of 8x8 block, the system performance can be optimized for both the speed and cost simultaneously no matter what kind of design for the prediction module is used.

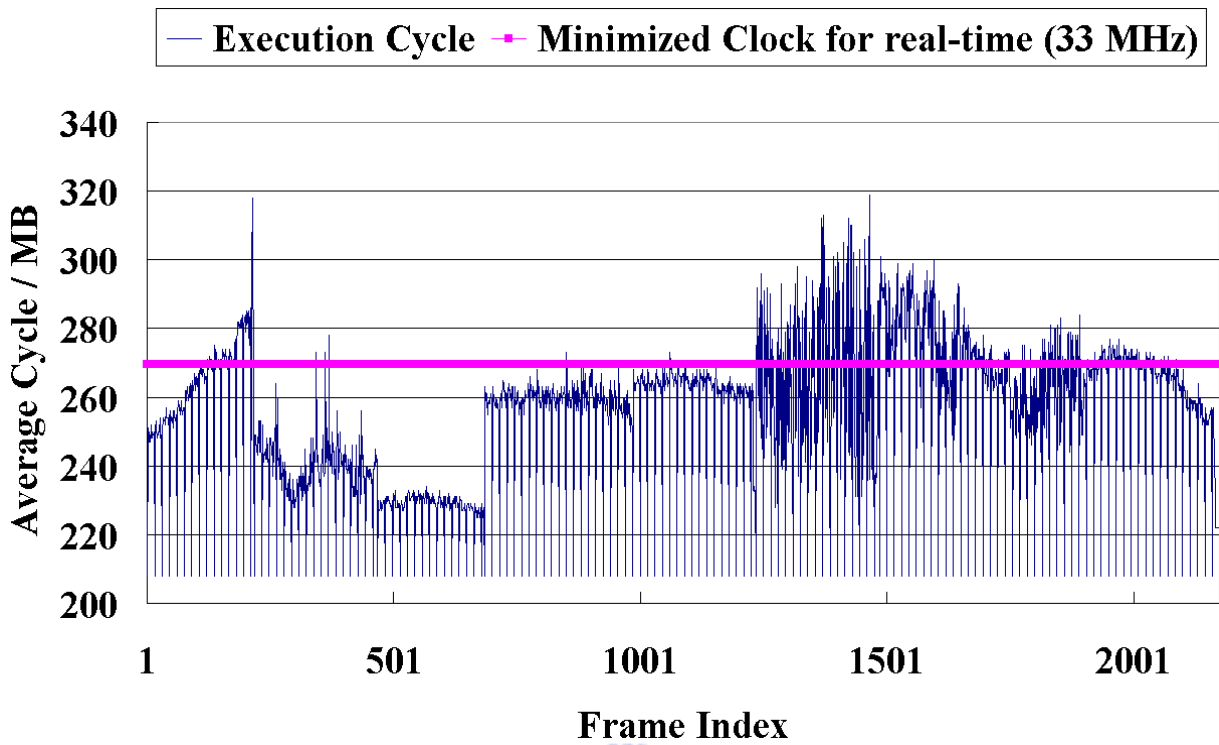
Our design space exploration is able to attain robust system performance in the face of varying application constraints. Specifically, the design alternative optimized for area can reduce the gate count of previous design up to 25%. The design alternative optimized for speed can increase the cycle count of previous design by 2 times at most such that the design with highest throughput performance can fulfill the real-time requirement for 1920x1088 @ 60 Hz videos when clocking at 135MHz. To show this claim, we simulate a long sequence for different resolutions and at certain bitrates as listed in Table 4.3. Figure 4.18 shows the average execution cycle count and the corresponding budget for the average cycle count at the minimal clock for real time requirement. Table 4.4 lists the minimal clock for real-time for various resolutions. In general, the internal clock frequency of a chip is the multiple of 27 MHz. Therefore, we round the required clock frequency to the multiple of 27 MHz and show the ratio of the slower than real-time frames when the operating clock frequency is lower than the required clock frequency.



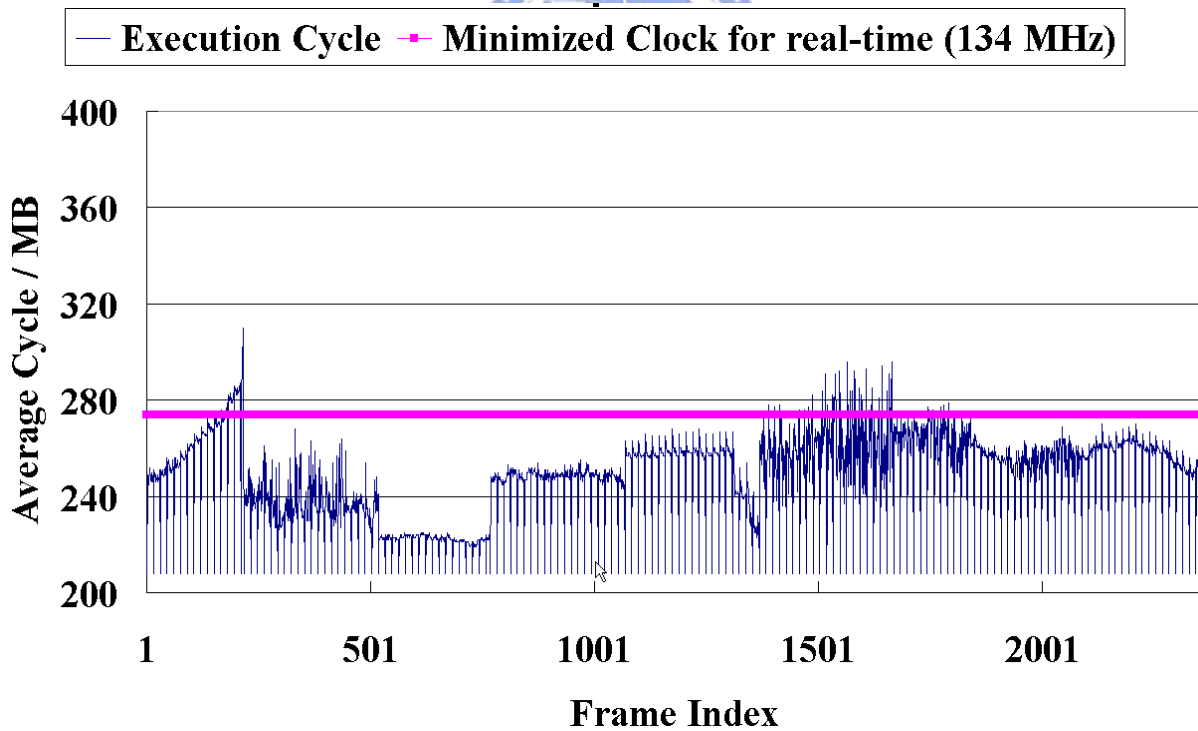
(a)



(b)



(c)



(d)

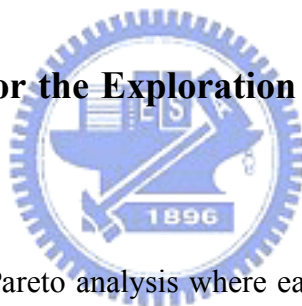
Figure 4.18: The Execution Cycle of Architecture 16_16_SA3 and The Minimized Clock for Real-Time When the Size of Display Buffer is 32Mb. (a) 240x144. (b) 480x272. (c) 960x544. (d) 1920x1088.

Table 4.4: The Required Clock For Each Resolution When the Size of Display Buffer is of 32Mb.

Resolution	Minimal Clock for Real-Time (MHz)	Minimal Clock of 27MHz for Real-Time (MHz)	Operating Clock N (MHz)	Ratio of Non-Real-Time Frame at Clock N (%)
240x144	3	27	27	0
480x272	9	27	27	0
960x544	33	54	27	15.03
1920x1088	134	135	108	12.82

In addition, our work explores a larger design space as compared to the previous design. Particularly, our design space provides a 3 times cycle count range and a 5 times gate count range.

4.7.2 Pareto Analysis for the Exploration of Design Combination within the Video Pipe



In Figure 4.19, we depict the Pareto analysis where each alternative is placed on the plot according to the cycle count and the equivalent gate count. Similarly, the cycle count is obtained by our TLM simulation framework while the gate count is obtained from the synthesis of a very rough RTL coding and the reports in the literatures. Observed from the left-down shift of the Pareto curves toward the origin, the SA-based IIP design and proposed FGSC-based deblocking filter significantly improve the area-speed efficiency of the system. While the high-throughput design of the IQIDCT has no distinct improvement upon the system throughput. Thus, it shows that the system performance may not be optimized while the individual module is improved twice.

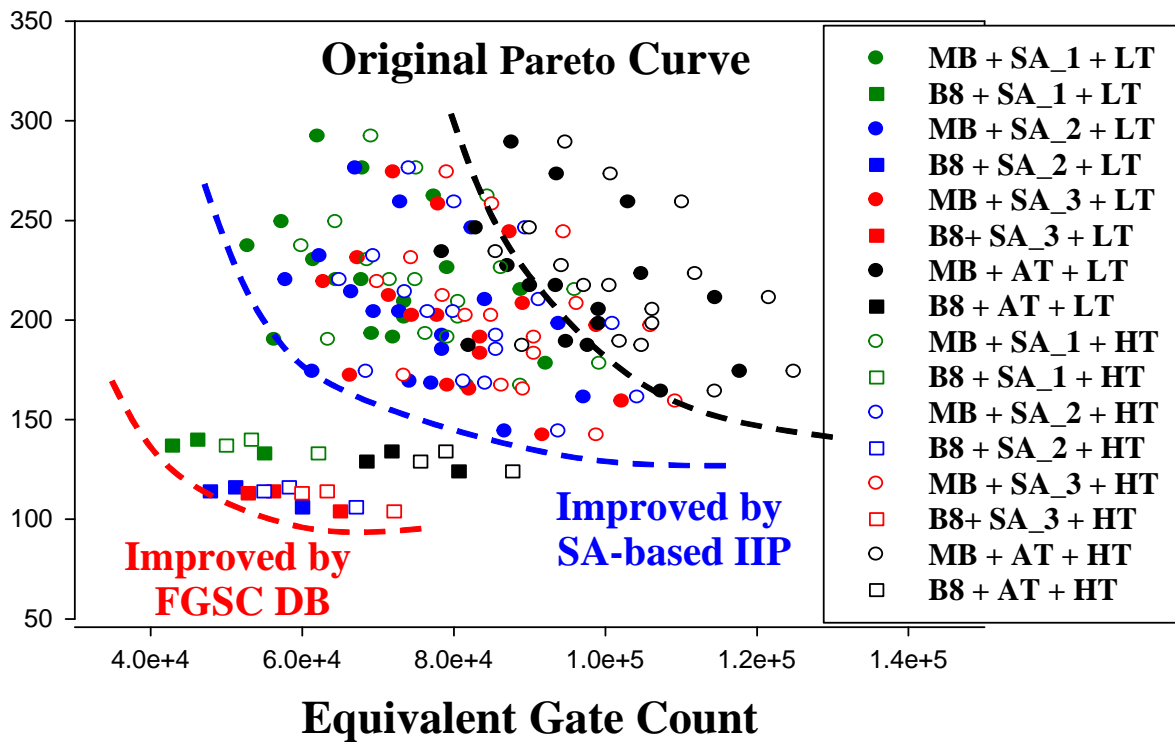
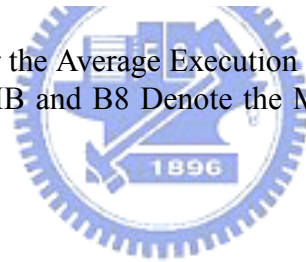


Figure 4.19: The Pareto Analysis for the Average Execution Cycle Count and Equivalent Gate Count for Each Alternative where MB and B8 Denote the MB-based and B8-based DB, Respectively



4.7.3 Area-Weighted Hardware Utilization for the Exploration of Design

Balancing of the Video Pipe

Figure 4.20 shows the area-weighted hardware utilization that reveals four system-level arguments. Firstly, each peak in the bar chart shows that the proposed FGSC-based deblocking filter remarkably increase the utilization. Secondly, the high-throughput design of IQIDCT decreases the utilization since it does not improve the system throughput at the expense of more cost. Thirdly, the combined SA-based IIP designs significantly improve the utilization as compared to the separated AT-based design. Lastly, increasing the parallelism of SA-based design has a bounded improvement for the system performance. As long as the DRAM access utilization is close to 100%, the off-chip bandwidth eventually becomes the performance bottleneck

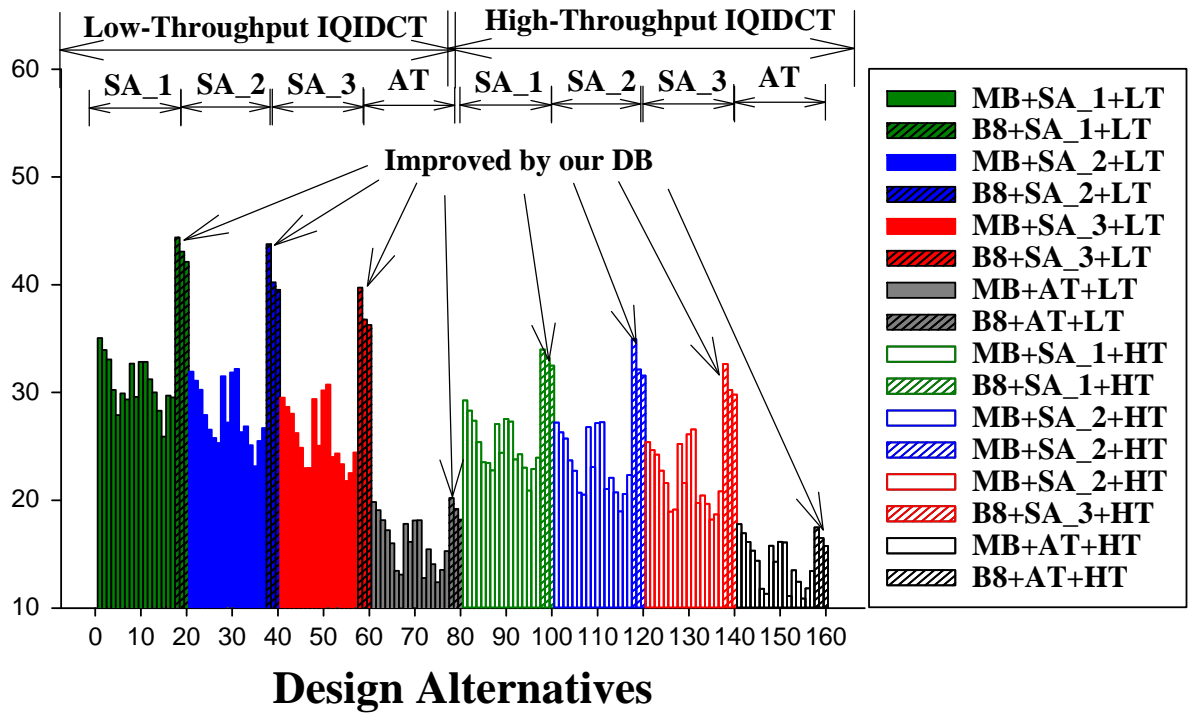


Figure 4.20: The Area-Weighted Hardware Utilization for Each Design Alternative

for higher throughput. Therefore, extra efforts on the SA-based designs, e.g. SA_3, have no distinct improvement upon the system throughput but the utilization is decreased.

Furthermore, among these alternatives, Figure 4.21 measures the normalized distance to the utopia point of each alternative based on the theory of multi-objective optimization. The shortest distance indicates the optimal solution for the three performance metrics simultaneously, i.e. execution cycle count, hardware cost, and area-weighted utilization. Therefore, the optimal solution among these alternatives is the combination of the low-throughput transform module, the 2-parallel SA-based prediction, and the FGSC-based deblocking filter.

4.8 Summary

This chapter proposes a highly efficient and platform-based system architecture for combining an H.264/AVC based video embedding transcoder and decoder. We firstly partition the system

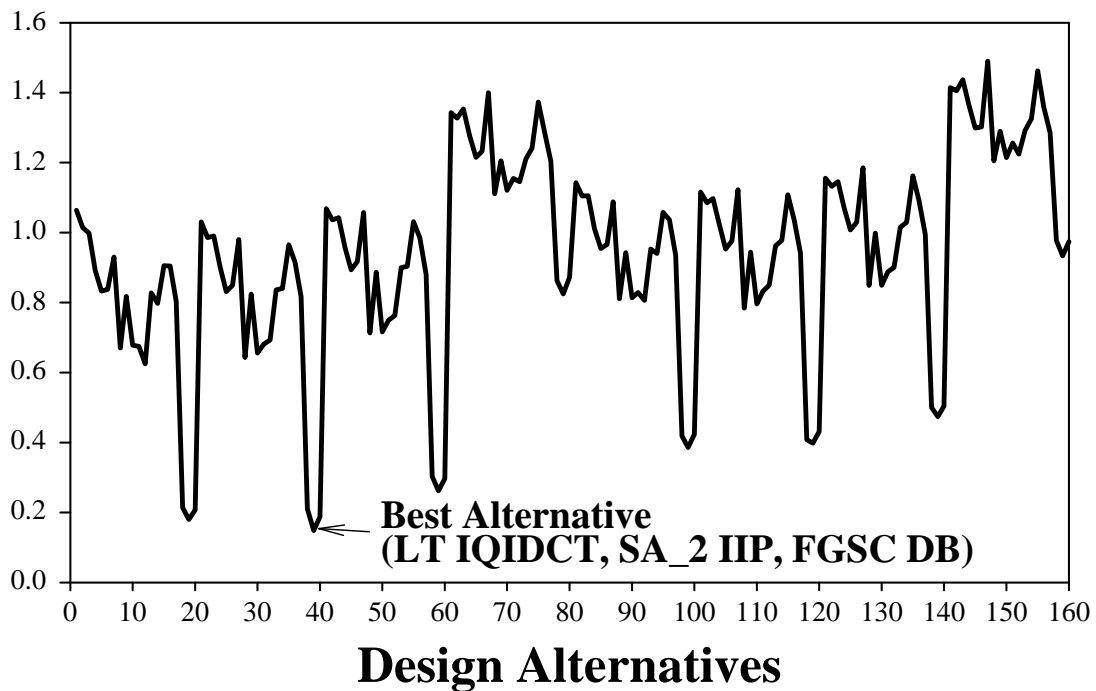
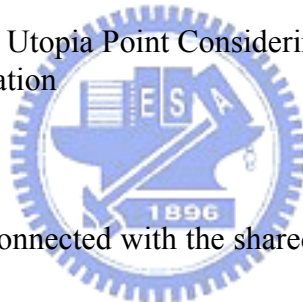


Figure 4.21: Normalized Distance to Utopia Point Considering Execution Cycle Count, Hardware Cost, and Area-Weighted Utilization



into several dedicated modules interconnected with the shared ping-pong memories to provide massive parallelism and facilitate efficient communication. Then, the architecture is implemented to ensure the correctness of data flow at system level and the design space exploration is performed for two system parameters: (1) the level of synchronization granularity and (2) design combinations within the video pipe. These models are evaluated at the TLM level, thus they minimize the modeling effort and increase the simulation speed so as to explore the 185 design alternatives.

The system performance are computed in terms of the average cycle count, the equivalent gate count and the area-weighted hardware utilization. From the system level simulation with the transaction level modeling, the best design alternative can reduce the cost of previous design up to 25% or can increase the speed of previous design by 2 times at most such that our design can fulfill the real-time requirement for 1920x1088 @ 60 Hz videos when clocking at 135MHz.

With the transaction level modeling, architectures can be modeled at higher abstraction level to obtain an initial estimate and subsequently refined to lower abstraction levels.



CHAPTER 5

A Highly Efficient Micro-Architecture Design



5.1 Introduction

For the micro-architecture design, it is important to increase the efficiency and the utilization of the dedicated hardware. As compared with the existing standards such as H.261/2/3 and MPEG-1/2/4, the H.264/AVC-based applications demand higher bandwidth for external memory due to intensive data transfer and less efficient DRAM access. With the intensive and irregular DRAM access, the memory sub-system that acts as a bridge for exchanging data between the on-chip hardware and the off-chip DRAM becomes the performance bottleneck for the real-time

applications. To increase the on-chip bus utilization while maximizing the effective throughput of the DRAM access, we will propose an efficient memory sub-system with synchronization buffer.

The prediction techniques used in the H.264/AVC can save up to 50% bit rates while providing similar perceptual quality as compared with the previous standards [68]. However, the coding gain is obtained at the cost of additional computations that require intensive filtering operations, thus posing challenges for the real-time applications. Moreover, the highly data-dependent irregular filtering makes hardware implementation more difficult. To increase the efficiency and the utilization, we will propose a systolic-based filtering architecture for combining the inter and intra prediction. The inter and intra predictions can actually share the processing elements because each macroblock is either coded in the inter mode or the intra mode.

The in-the-loop deblocking filter is another key feature as compared to the existing standards. Most of the architecture design of the deblocking filter used the macroblock-based filtering orders, which are inefficient for the finer granularity of pipeline synchronization. To reduce the processing latency and memory requirement, we will propose a novel deblocking filter with fine-grained synchronization capability.

The rest of this chapter is organized as follows: Section 5.2 presents an efficient memory sub-system. Section 5.3 details an unified systolic architecture for the inter and intra prediction. Section 5.4 proposes a novel deblocking filter with fine-grained synchronization capability. Lastly, Section 5.5 summaries this chapter.

5.2 An Efficient Memory Sub-System

5.2.1 Background

The H.264/AVC, which incorporates advanced inter predictions, demands higher bandwidth for the external memory due to the intensive data transfer and less efficient DRAM access. The intensive data transfer is introduced by more accurate sub-pel interpolation, motion compensation of smaller block size, as well as motion field inference of temporal direct modes in the B slices. On the other hand, the inefficient DRAM access is caused by motion compensation of the variable block size and adaptive temporal prediction with multiple reference frames. As compared with MPEG-1/-2/-4, the frequency of DRAM access is increased by 2x to 16x. Moreover, the ratio of row-miss in the external DRAM is increased considerably.

With the intensive and irregular DRAM access, the memory sub-system that acts as a bridge for exchanging data between the on-chip hardware and the off-chip DRAM becomes the performance bottleneck for the real-time applications. Several literatures are dedicated to improve the efficiency for the DRAM access. In [63][64], the probability of page miss is minimized by storing reference frames in the DRAM with checkerboard pattern. That is, the adjacent groups of pixels are stored in different banks with interleaved order. Furthermore, in [65] the luminance and chrominance components are stored in two separate DRAMs so as to gain sufficient bandwidth. However, the auto precharge function is enabled after each DRAM access; consequently, two to four DRAMs must be used to match the bandwidth of the on-chip modules. To effectively schedule the DRAM commands, in 5.1 a history-based prediction scheme is used to enable auto precharge function according to the run-time behavior of the memory access. However, the prediction accuracy is reduced significantly by the irregular data access of the H.264/AVC.

To increase the on-chip bus utilization while maximizing the effective throughput of the DRAM access, a synchronization buffer is employed for re-formatting the read/write data exchanged between the on-chip modules and the off-chip DRAM. In addition, we optimize the issues of DRAM commands and adaptively enable the auto-precharge function by monitoring the motion information of a submacroblock. Further, for lower probability of row miss, the reference frames are partitioned into multiple blocks of size 32x32 and stored in the DRAM with checkerboard pattern. In the following subsections, we will present the memory sub-system in detail, including (1) an interleaved data arrangement scheme for improving the efficiency of the DRAM access, (2) an external memory interface for the control of mobile DDR SDRAM, and (3) a synchronization buffer for the data cache.

5.2.2 Interleaved Data Arrangement

The data arrangement in the external memory is critical to the effective bandwidth that can be achieved as well as the power consumption incurred for the DRAM access. An inefficient data arrangement increases the probability of row miss, for which the data in the sense amplifier must be firstly precharged before another row is activated causing longer latency for DRAM access and lower effective bandwidth. In addition, the row activation and the precharge operation induce more power consumption than normal read/write access. Eq.(5.1) illustrates the energy consumption during the DRAM access [66], where N_{active} , $N_{precharge}$, N_{rw} and are the number of active, precharge, and read/write access respectively.

$$E_{dynamic} = N_{active} \times E_{active} + N_{precharge} \times E_{precharge} + N_{rw} \times E_{rw} \quad (5.1)$$

According to the data sheet in [46], each active or precharge command typically expends 14,000 pJ while each read/write access consumes only 2,000 pJ. Thus, the active and precharge com-

mands require 7x more energy consumption than read/write access. As a result, an efficient data arrangement must simultaneously minimize the probability of row miss and reduce the number of read/write access.

In this subsection, we propose an interleaved data arrangement based on the checkerboard pattern to minimize the probability of row miss. As shown in Figure 5.1, the luminance component of each reference picture is partitioned into multiple blocks of size 32x32 (i.e., four macroblocks) and the spatially adjacent blocks are stored respectively in the 4 different banks of DRAM. When accessing a block for the motion compensation as the example in Figure 5.1, at most one row is activated for each bank such that the number of row-miss is minimized. In addition, the row activations for different banks can be overlapped thereby the latency can be reduced. Similarly, the same arrangement is applied to the chrominance components. In particular, because of lower resolution, the Cb and Cr components are mixed together when forming the 32x16 block. Moreover, since the chrominance block is processed right after a luminance block, the interlaced memory mapping allocates the luminance and chrominance pixels and motion data of a 32x32 block into the same row of a bank such that the row activated for the data access of a luminance block remains open during the processing of the following chrominance block. Additional precharge and RAS (Row Address Strobe) latency could be eliminated due to such arrangement.

To fulfill the requirements, we define a group of 4 spatially adjacent macroblocks as a luminance row block and the pixels in a luminance row block are stored in the same row of a bank. Similarly, the same arrangement is applied for the associated chrominance components. In particular, because of lower resolution, each Cb and Cr block of size 16x16 are mixed together to form a chrominance row block of size 32x16. The samples in a chrominance row block are appended after those in the associated luminance row block. Moreover, the remaining space of

Micron Mobile DDR SDRAM (MT46H16M16LF)

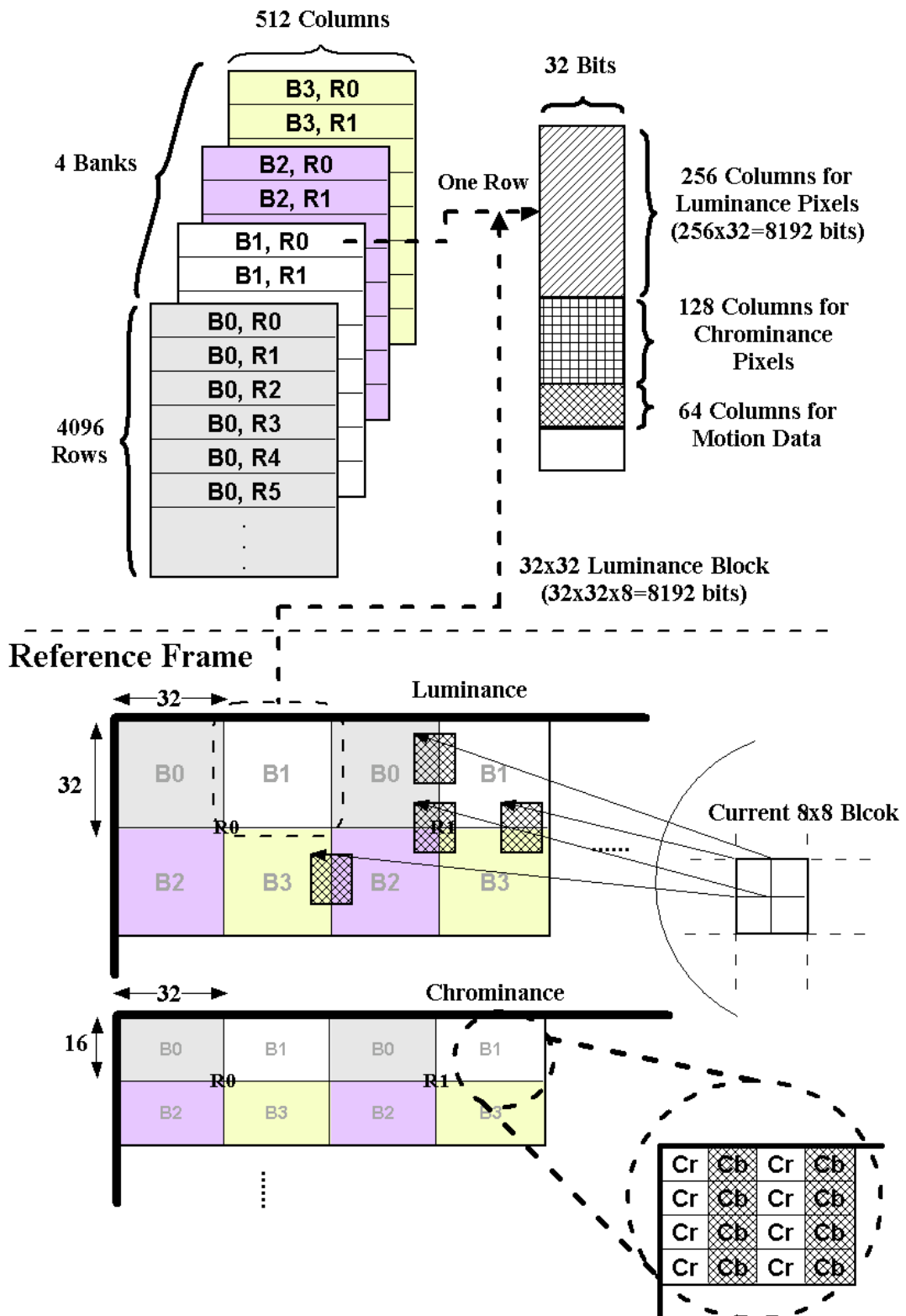


Figure 5.1: The Interleaved Data Arrangement for the Stored Pictures

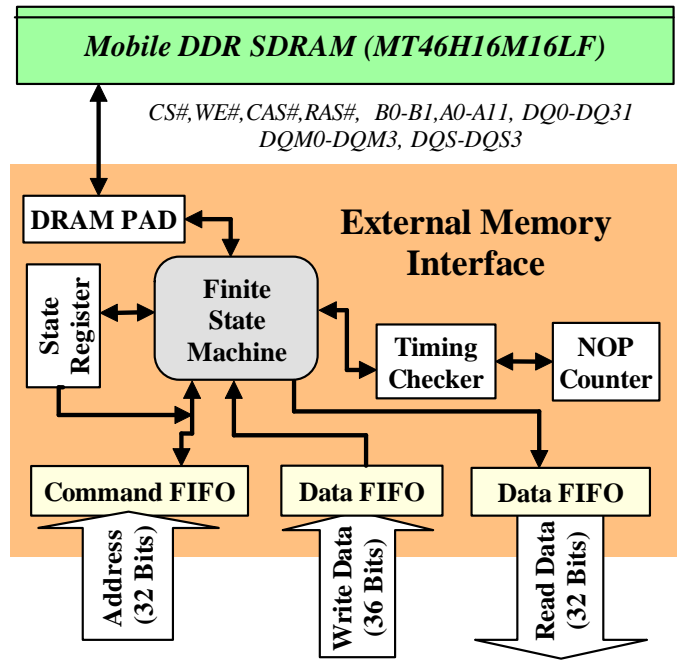


Figure 5.2: The Functional Block Diagram of the External Memory Interface

a row is allocated for the storage of motion information including motion vectors and reference indexes.



5.2.3 External Memory Interface

To minimize the extra cycles induced by the row miss and the bank miss, an external memory interface optimized for the control of the mobile DDR SDRAM is proposed. As shown in the functional block diagram of Figure 5.2, the physical addresses from the address translator are stored in a specific command FIFO (First-In-First-Out) such that auto precharge function can be used more efficiently. The finite state machine (FSM) combined with the state register, timing checker, and counter generates the corresponding DRAM commands under the considerations of the timing constraints and the power consumption. The read/write data FIFO are included to synchronize the latency of row activation, burst data, and video processing operations.

5.2.3.1 Adaptive Auto Precharge

The auto precharge function of DRAM allows the memory controller to perform the read/write commands in a more compact way so that the latency of DRAM access can be reduced. Normally, during the DRAM access, the row being accessed will be remained activated for the subsequent accesses unless a refresh command is received. With the auto precharge function, one can decide whether the row being accessed should be precharged automatically at the end of a read/write burst. Generally, the decision for enabling the auto precharge function is dependent on consecutive commands. For instance, the auto precharge function will be a benefit when the row miss occurs. However, it is redundant when the next DRAM access actually points to the same row of a bank. In that case, an unnecessary active command and extra RAS latency are introduced.

To adaptively enable the auto precharge function for minimizing the latency of DRAM access, a command FIFO is allocated for the detection of row miss between consecutive DRAM accesses. As shown in Figure 5.3, the FIFO is firstly initialized with the address information of the data to be accessed, including the row and column of the target banks. Moreover, a hit flag is associated with each entry to distinguish whether the address of the current DRAM access is different from the one of the next access. By default, the hit flag is set to 1. When the target address of the incoming DRAM access is different from the current one, the hit flag of the current entry will be reset to 0. According to the hit flag, the auto precharge function can be adaptively enabled.

5.2.3.2 Finite State Machine (FSM)

Figure 5.4 shows the finite state machine (FSM) of the external memory interface, which processes the DRAM access and generate the DRAM commands along with the consideration

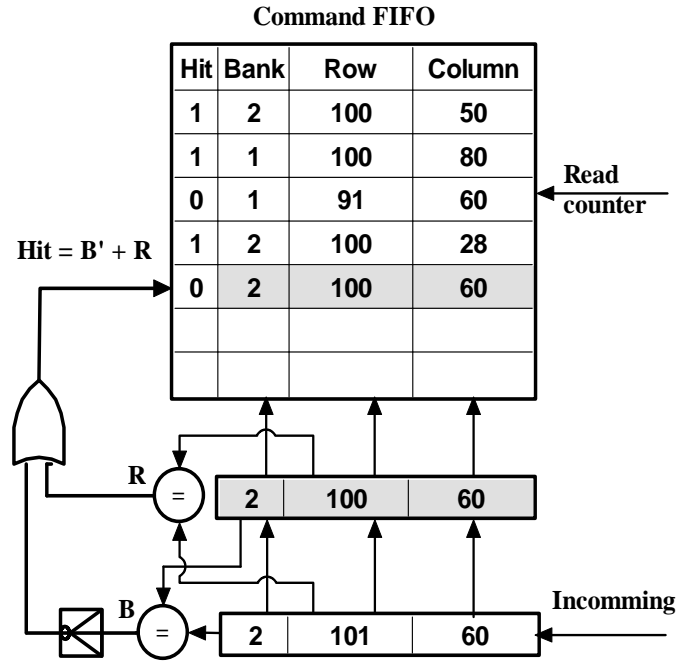


Figure 5.3: The Command FIFO for the Detection of the Row Miss

of the timing constraints while providing the minimal latency. The timing checker and the NOP counter are used to check the timing constraints of each state. The state register that records the status of each bank is used for the scheduling of read/write commands so as to minimize the number of cycles between commands. In addition, the operations of all the banks are controlled by one unified FSM instead of multiple FSMs such that the cost the memory controller is minimized.

5.2.4 Synchronization Buffer

The size of the synchronization buffer is critical to the efficiency of the DRAM access. A synchronization buffer of larger size allows multiple DRAM accesses to be optimized jointly. For instance, if the synchronization buffer is designed in such a way that the data required for the decoding of a slice can be fetched in advanced, then one can schedule the data access of a slice according to the target addresses to minimize the row miss and thus increase the

Table 5.1: Analysis of the Different Levels of Granularities Based on the Worst Case Assumption

Level of Granularity of Memory Sub-system	4x4	8x8	One Macroblock	Two Macroblocks
Synchronization Buffer (Bits)	2160	5424	20160	40320
Cycles/Macroblock (one DRAM)	2096	940	745	454

Configuration for 1920x1088@30Hz

Number of DRAM (N)	8*	4	2	1
Cycles/Macroblock (N DRAMs)	1136	532	595	454
Bandwidth Utilization	7.3%	16.8%	28.1%	78.2%
AHB Bus (Bits)	128	128	64	32

*Not Able to Meet Real-Time Requirement

As shown, smaller block size causes poor DRAM bandwidth utilization because of more DRAM active and precharge operations. Equivalently, for the same real-time requirement and clocking rate, the configuration with smaller block size demands more DRAMs and wider AHB data bus. In TABLE I, when one DRAM is used, only the granularity of 2MBs can fulfill the real-time requirement when clocking at 162MHz, i.e., 660cycles/MB.

However, in H.264/AVC, designing for the worst case may not be the best policy since the worst case could only occur in an irrational sequence. Moreover, the bandwidth increase by using more DRAMs could be constrained by the limited pin counts. In the next subsection, we will show the performance of the proposed memory sub-system when the synchronization buffer is designed at different levels of granularity.

5.2.5 Effect of Synchronization Granularity of Memory Sub-system on Off-Chip Transmission

Generally, increasing the size of the synchronization buffer improves the efficiency of DRAM access; however, it also introduces higher cost for the on-chip hardware. To find the best design

trade-off, the efficiency of DRAM access and the associated power dissipation as well as the hardware cost are analyzed when the synchronization buffer is designed at different levels of granularity. The efficiency of DRAM access and the associated power dissipation as well as the hardware cost are analyzed when the synchronization buffer is designed at different levels of granularity, including 4x4, 8x8, and 16x16 (1 macroblock).

In this subsection, the DRAM access is statistically analyzed when the synchronization buffer is designed at different levels of granularities, including 4x4, 8x8, and 16x16. The analysis is conducted by deploying one external DRAM. To capture the characteristics for different scenarios, the testing sequences include four QCIF sequences (Stefan, Mobile, Table, Coast-guard) and 4 HD sequences (Pedestrian, Rush_hour, Sunflower, and Station2.). Each sequence is further coded at low, medium, and high bit rates using the Qp of 10, 28, and 45 respectively. In addition, to minimize the number of active and precharge operations, the data mapping in the external DRAM is based on the checkerboard arrangement as presented in Subsection 5.2.2.

Figure 5.5 shows the memory efficiency for the motion compensation, which dominates the access of DRAM. As shown, the access of DRAM has higher efficiency when using larger block size. By firstly observing the block partition and motion vectors, we can prevent redundant data from being transferred. Figure 5.6 further shows the memory efficiency from the system perspective; that is, the DRAM access for the on-chip modules is simultaneously considered. Again, we have similar observation as in Figure 5.5. However, the difference between the 8x8 and 16x16 is not significant at medium or low bit rates.

To evaluate the DRAM latency, Figure 5.7 illustrates the average cycle counts for DRAM access per MB. Here, we consider all the data transfer for the on-chip modules. As expected, larger block size require less DRAM cycles because of higher memory efficiency. Moreover, as shown in Figure 5.8, larger block size also has less memory overhead due to less frequent active

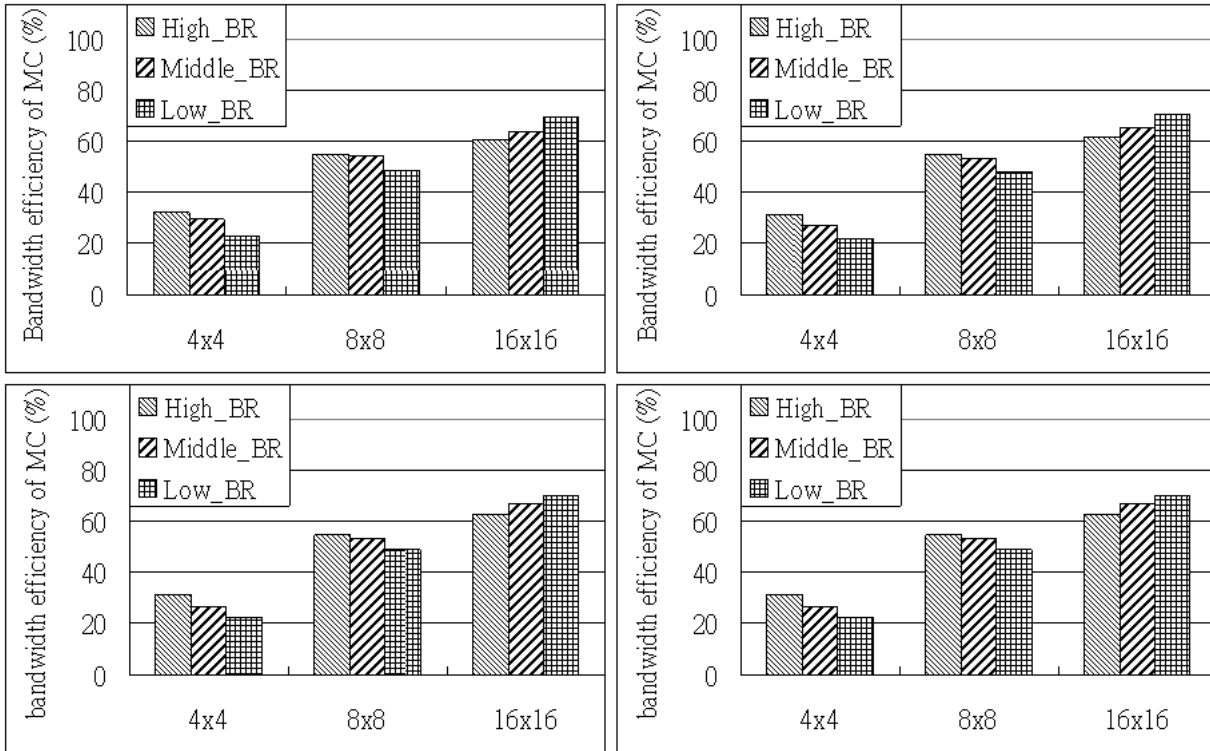
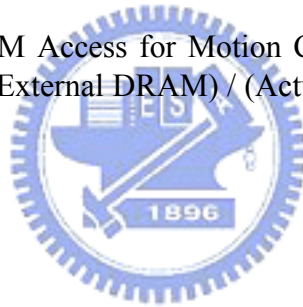


Figure 5.5: The Efficiency of DRAM Access for Motion Compensation. The Efficiency is Defined as (# of Data Read from the External DRAM) / (Actual Data Required for the Motion Compensation)



and precharge operations.

The efficiency of DRAM access also affects the power dissipation. Figure 5.9 compares the estimated power dissipation for different block granularities according to the DRAM specification [46]. As shown, smaller granularity dissipates higher power due to more DRAM commands. Moreover, since each active or precharge command spends 7x power consumption than the read/write commands, larger block size, which has less active and precharge operations, is more efficient in terms of power dissipation.

From the experimental results, we obtain the conclusions as follows: (1) with one DRAM, only the granularity of two macroblocks guarantees the design for the worst case. However, (2) using the granularity of one macroblock with one DRAM is sufficient for the average case. Moreover, (3) using the granularity of 8x8 block with one DRAM consumes more power while

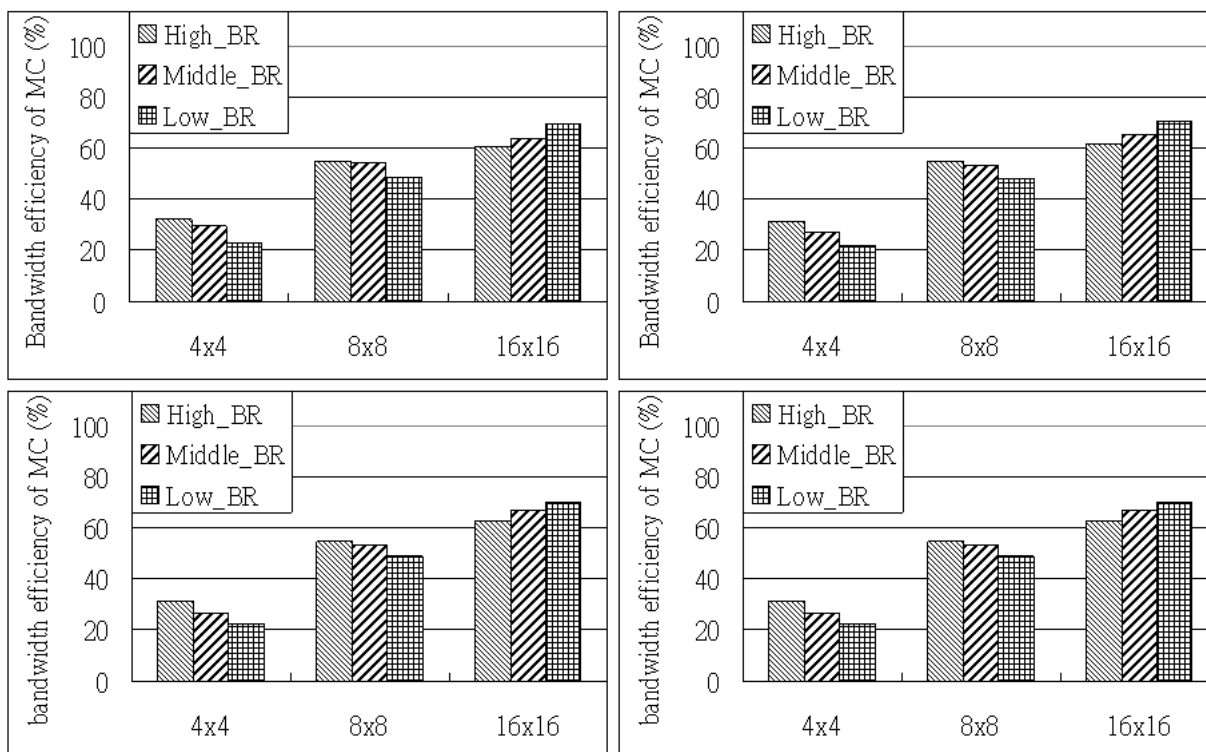


Figure 5.6: The Efficiency of DRAM Access for the Decoder. The Efficiency is Defined as (# of Data Read from the External DRAM) / (Actual Data Required for Decoder)

the synchronous buffer can be reduced. In addition, (4) using the granularity of 4x4 block requires least internal buffer, but needs more external DRAMs to provide higher bandwidth. In summary, the synchronization buffer with larger block size has higher memory efficiency, and thus, less DRAM access cycles and power dissipation. However, the granularity of 8x8 block provides better trade-off among cost, efficiency, power, and real-time requirement.

In summary, the intensive, irregular DRAM access in the H.264/AVC decoder makes the memory sub-system become the performance bottleneck for the real-time applications. Moreover, the variable-rate bumping process further complicates the management of decoded picture buffer. To tackle these issues, an efficient memory sub-system and a constant-rate bumping process are proposed. Firstly, the memory interface is designed to optimize the issues of read/write commands and the auto-precharge function. Moreover, an interleaved data arrange-

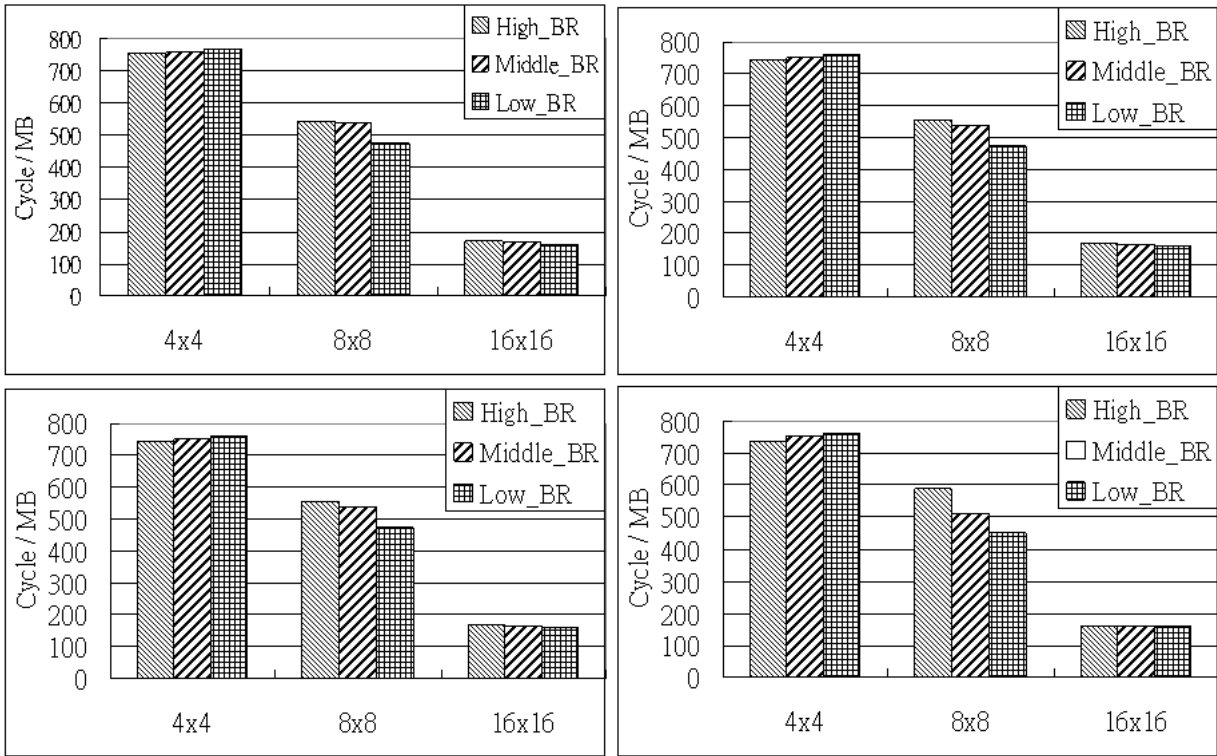


Figure 5.7: The Average Cycle Counts per MB

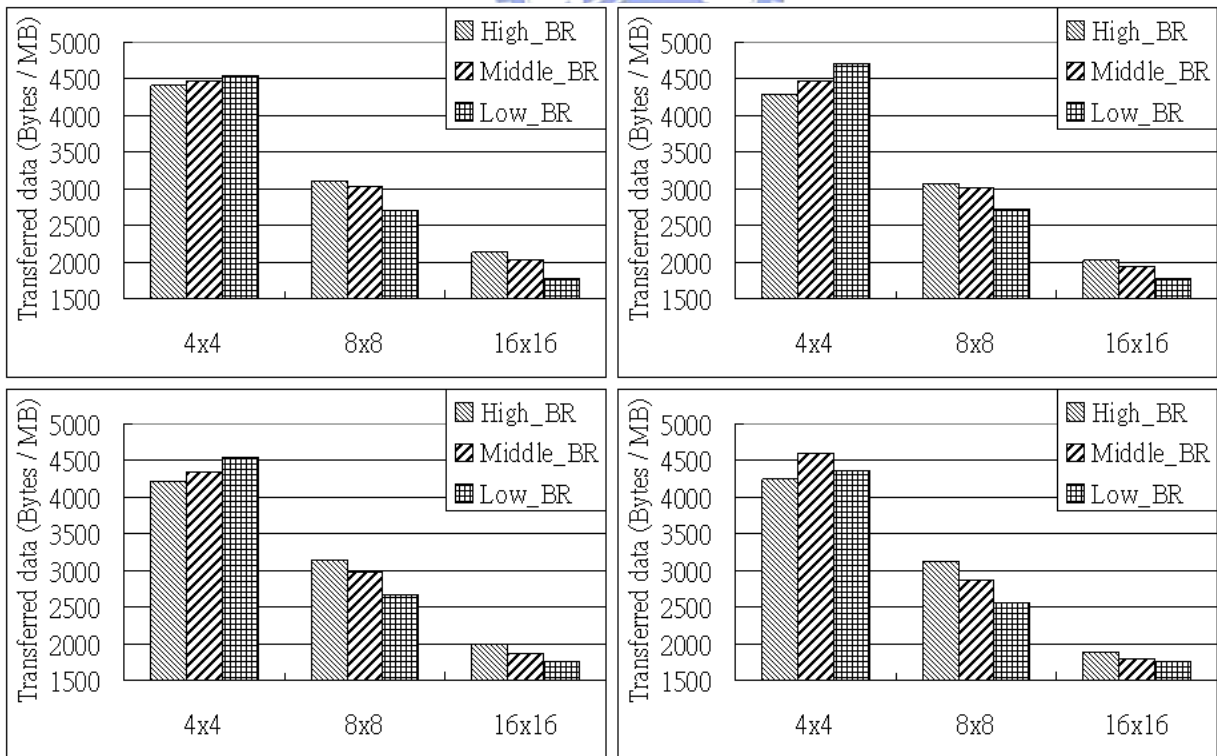


Figure 5.8: The Amount of Data Transfer

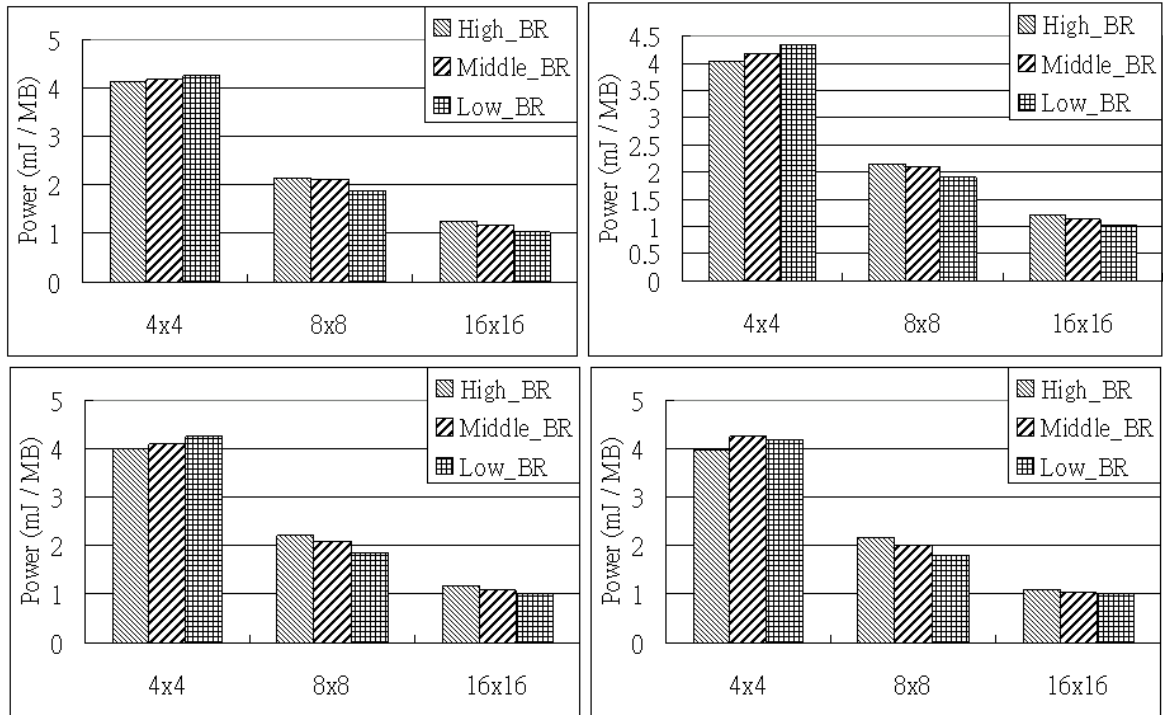


Figure 5.9: The Power Consumption in DRAM

ment is proposed to reduce the probability of row miss. Secondly, a regulation buffer with size being equal to the decoded picture buffer is created to ensure a constant output rate of decoded pictures for any conformed prediction structures. Thirdly, a synchronization buffer at the granularity of 8x8 is used to maximize the DRAM bandwidth while minimizing the extra cost.

5.3 Combined Inter and Intra Prediction

In this section, we implement the inter and intra prediction with a reconfigurable systolic architecture. First, we map all kinds of computations into the systolic operations such that the different predictions can be performed in a similar way. Second, we synthesize a unified systolic architecture from the description of prediction algorithm. Systolic architecture includes a number of regular and modular processing elements (PEs) that simultaneously process and pass data in a similar way. All PEs regularly pump data in and out such that a regular data

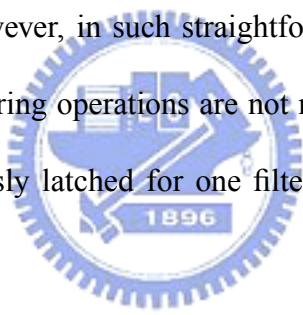
flow is maintained [67]. Thirdly, we use several multiplexers to configure the data paths according to the coding modes and the motion vectors. Therefore, we can share the array of PEs for all the prediction modes including the inter prediction of luminance component, the inter prediction of chrominance component, and the intra prediction of both components. For the inter prediction, the 2-D interpolation is conducted through two separable 1-D filterings. For the intra prediction, the boundary pixels are reshuffled before feeding into the systolic array. In the following subsections, we firstly elaborate the disadvantages of the related work on the design of inter and intra prediction. After that, we present a unified architecture that efficiently combines the inter and intra prediction. Then, we detail the operations of the architecture at different configurations. Lastly, we compare the proposed architecture with the state-of-the-art designs.

5.3.1 Motivation



The spatial and temporal predictions are essential to video coding efficiency. The H.264/AVC [6] simultaneously incorporates the inter and intra predictions to remove temporal and spatial redundancy. Comparing with the existing standards H.261/2/3 and MPEG-1/2/4, these prediction techniques save up to 50% bit rates while providing similar perceptual quality [68]. However, the coding gain is at the cost of additional computations. In the intra prediction, the mode-adaptive predictor is generated by a 1-D filtering, which is conducted along with the boundary pixels of a block. Similarly, the half-/quarter-pel predictor in the inter prediction is produced through a separable 2-D filtering with the motion compensated blocks of variable size. Both predictions require intensive filtering operations that poses challenges for the real-time applications. Moreover, the adaptive and irregular filtering makes hardware implementation more difficult. Therefore, there are many related work on the design for the inter and intra prediction.

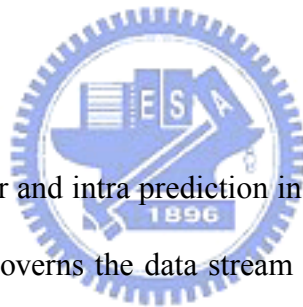
However, the state-of-the-art designs of the pixel prediction, which includes the inter and intra prediction, pose some disadvantages in the aspect of efficiency and utilization as following:

1. The inter and intra predictions are always implemented as two separated modules due to the difference in their operations [69][70][71][72][73][74][75][76]. However, in the application of decoder and transcoder, the prediction mode of each macroblock in the input bitstream is known in advance. Thus, using separated hardware resources for the inter and intra prediction causes poor hardware utilization.
2. For the interpolation in the inter and intra prediction, most of the prior works implement the finite impulse response (FIR) filter based on the traditional adder-tree (AT) structure [77][44][78][47][79], where the filtering is implemented by a number of tree-structured adders and shifters. However, in such straightforward implementation, common terms between consecutive filtering operations are not reused at all. Moreover, multiple input samples are simultaneously latched for one filtered output causing higher input bandwidth.
3. The number of FIR filter in the AT-based design [34][78][47] is designed for the worst case, i.e. all the 4x4 blocks are coded as Inter_4x4 mode that requires 2-D interpolation. However, the filter utilization is significantly decreased for the block partition that only requires 1-D interpolation. Furthermore, the design for the 4x4 block partition introduces redundant computations in case of larger block partition. In our simulation, the worst case occurs rarely in the actual bitstream such that the AT-based design performs worse system performance on the average.
4. In addition to the less efficient FIR filter design, the AT-based design is tightly coupled with the external memory [47] or the on-chip data bus [71]. The latency of the external memory could compromise the performance of the prediction module. Besides, the re-

dundant data transmission not only increases the transmission power but also degrades the system performance caused by serious bus contention.

To increase the efficiency and the utilization, we propose a unified filtering architecture for the inter and intra prediction. First, we share the data paths for both the inter and intra prediction so as to increase hardware utilization and reduce hardware cost. Second, to minimize redundant computations in the pixel predictions, the FIR filtering is implemented by a reconfigurable systolic architecture. Thirdly, our proposed systolic architecture is fully utilized for any kind of interpolation and block partition. Fourthly, we allocate a local FIFO and memory for temporarily buffering the motion-compensated data and the intermediate data such that the motion-compensated data of a block partition is transferred without redundant transmission.

5.3.2 Overall Architecture



The overview architecture of our inter and intra prediction is given in Figure 5.10. Depending on the coding mode, the controller governs the data stream entering and leaving the ports of the unified systolic array. In the H.264/AVC, the predictor of a block is created from image samples that are coded in either the previously decoded frames or the current frame. The inter prediction creates the predictor of a block from the previously-coded frames that are stored in the external memory. The motion-compensated data, which is determined by the motion vector and the size of block partition, is pre-fetched into the synchronization buffer before generating the predictor of the current block as mentioned in Section 4.3,. Thus, the source data of inter prediction comes from the on-chip data bus via AHB interface. Due to the conflict between 32-bit data bus and pixel-wise processing granularity of the systolic array, the internal pixel FIFOs are used to harmonize the bus transmission and the pixel interpolation. On the other hand, the intra prediction creates the predictor for a block using the boundary pixels in the

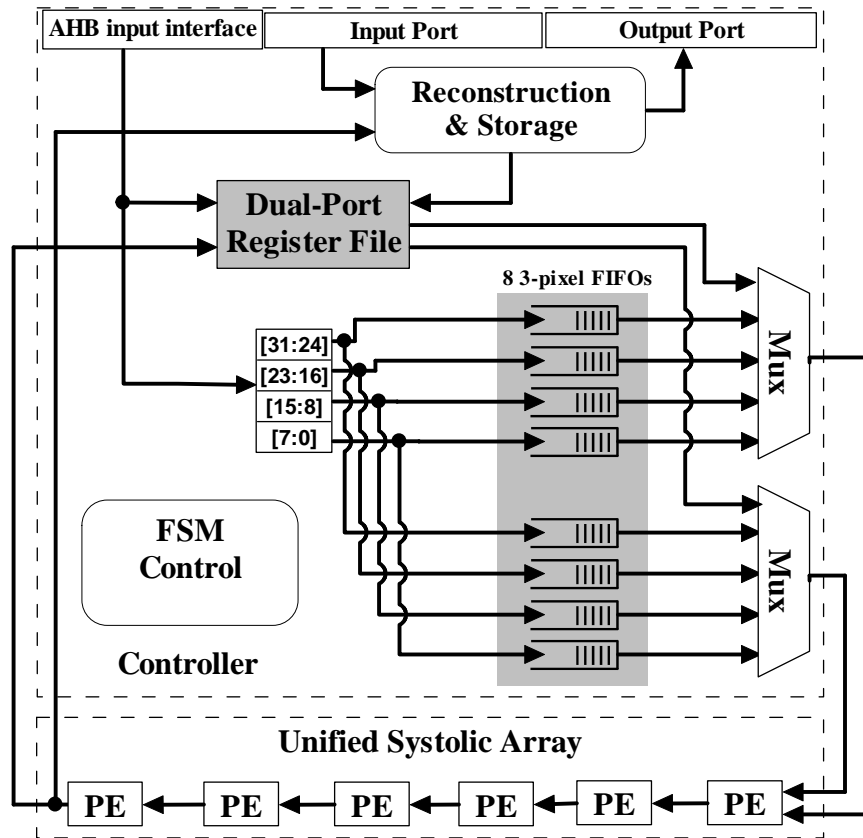


Figure 5.10: The Architecture of Inter and Intra Prediction

adjacent blocks. That means the systolic array get the data in the local memory that stores the boundary pixels of the adjacent blocks when intra prediction is performed. Therefore, the finite state machine (FSM) controller can select the source data as the input of systolic array by two 5-to-1 multiplexers as shown in Figure 5.10.

In addition to the selection of the input of systolic array, the FSM controller controls the data flow for each inter-coded block which is of variable size and requires various interpolations depended on the motion vector. Due to the sub-pixel resolutions of motion vectors such as 1/2-, 1/4-, and 1/8-pixel, the inter prediction requires intensive computations for the interpolation of motion-compensated full-pixel samples. Specifically, the 1/2-pixel samples are interpolated from full-pixel samples using the 6-tap FIR filter whose tap is (1, -5, 20, 20, -5, 1). Particularly, when the motion vector of a block points to a certain position, a 2-D interpolation may be

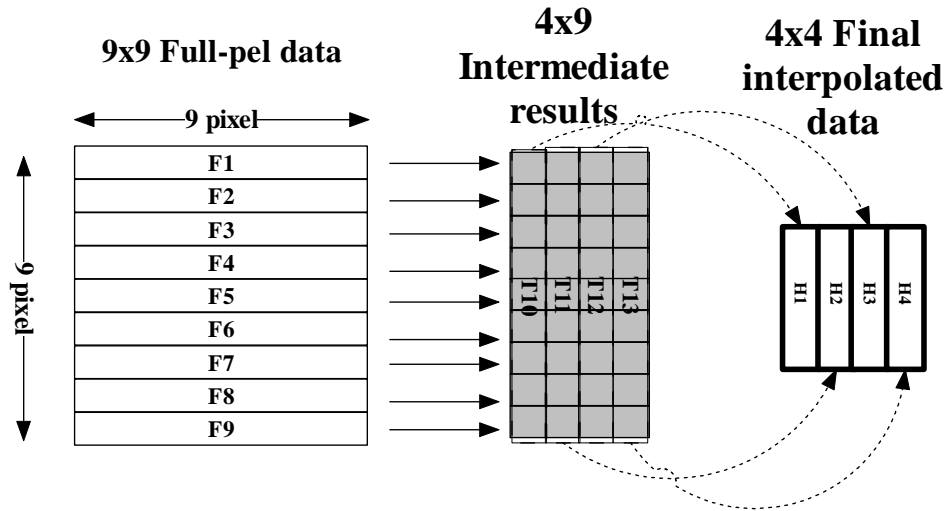


Figure 5.11: The 2-D Interpolation for the Motion Compensation with Sub-Pel Precision. Note that the 2-D Filtering Can Be Separated Into Two 1-D Filtering.

required which means that the second interpolation of one predictor is dependent on the results of the first interpolation. In order to increase the hardware utilization, the 2-D interpolation is done by two separable 1-D filterings as shown in Figure 5.11. For instance, to get a 4x4 block whose motion vector points to the position that requires a 2-D interpolation, the filter firstly processes a 9x9 block into a 9x4 block. The local memory buffers the intermediate results and transposes them when performing the second filtering. Lastly, the filter processes the 9x4 block to produce the predictor of 4x4 block. We use a dual-port register file as the local memory and its size depends on the level of synchronization granularity. For instance, a 104x8 register file is allocated to buffer the intermediate 13x8 block of pixels at the granularity of 8x8 block.

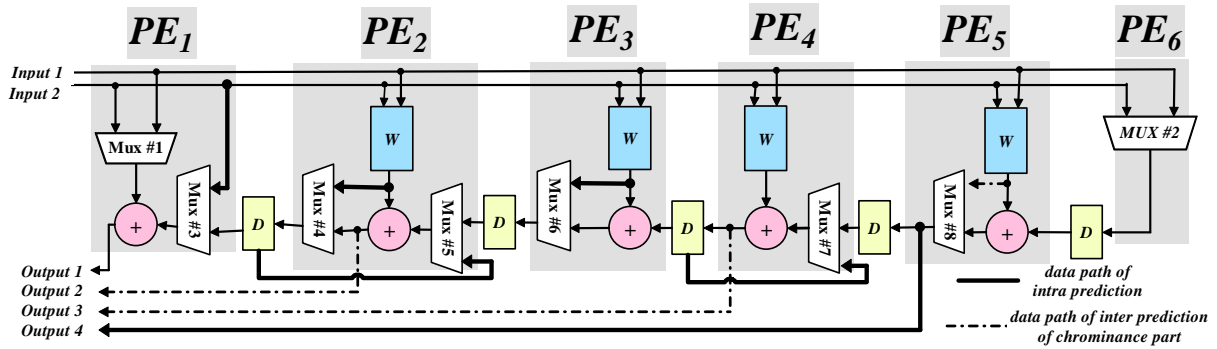
Similarly, the FSM controller controls the data flow for each intra-coded block which is of variable size and requires variable predictions depended on the coding mode. In the main profile of H.264/AVC [6], each intra-coded macroblock can have one of the two prediction types, which are Intra_4x4 and Intra_16x16. For each type of predictions, the macroblock is firstly partitioned into multiple sub-blocks (with size being NxN where N can be 4 and 16). Then,

each sub-block can be further assigned with directional modes, DC mode, or plane mode where different operations are required. While the FSM controller adequately inputs the boundary pixels into the systolic array, all different operations can be mapped into the FIR filtering.

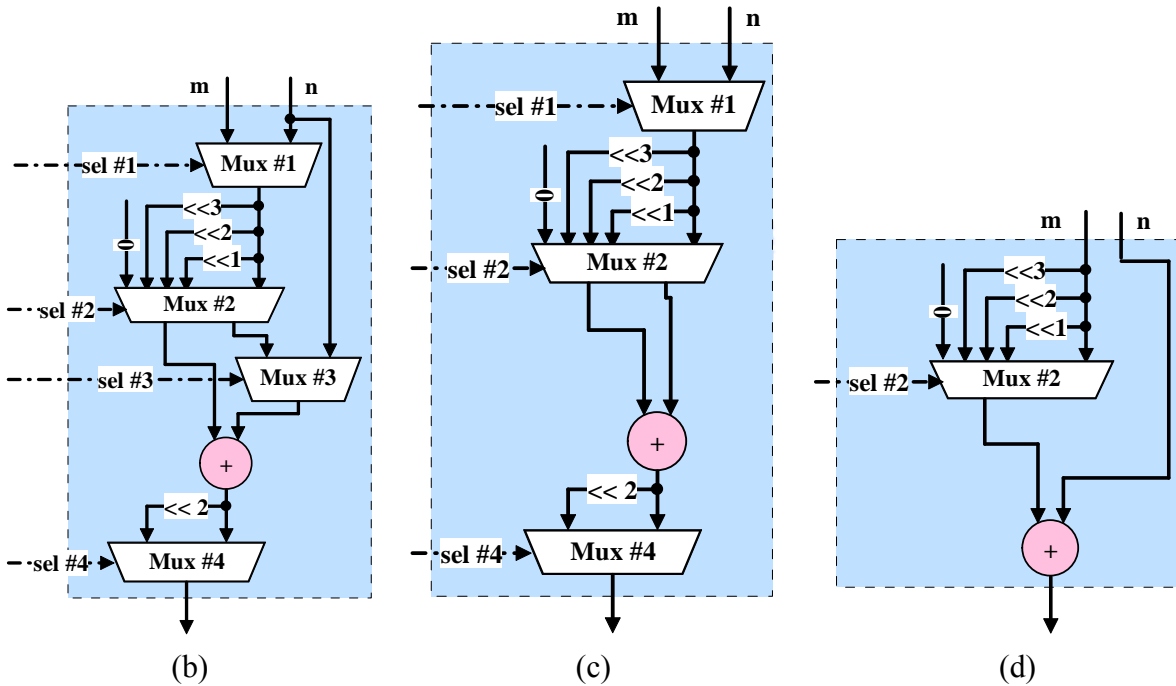
After the predictor of a block is generated, the reconstruction unit adds the predictor to the residue from the input port and then outputs the reconstructed pixels to the next module, i.e. the deblocking filter. In particular, the boundary pixels of the current block need to be stored in the local memory for the subsequent intra-coded blocks.

The FIR filtering is performed using a unified systolic array which pumps the data from the controller in a pipeline fashion. For details, Figure 5.12(a) shows the pipelined network arrangement of the PEs within our unified systolic array. It consists of 6 PEs and these PEs perform a sequence of operations on data that flows between them. Each PE is responsible for the multiplication and addition of a filter tap: one multiplies its input by a factor, adds the data from the right, and passes the result to the left. The outcome of each PE is buffered in a D flip-flop such that each pixel is not only used when it is input but also is reused as it moves through the pipelines in the array. The size of the D flip-flop is 18-, 18-, 18-, 16- and 13-bit respectively from left to right. Except for the initial cycles, the systolic array pumps in two pixels and generate 1 to 4 predictor(s).

The multiplexers in Figure 5.12(a) can configure the data paths so as to provide various filters. Specifically, the Mux #1 and #2 are used to select one of two inputs while performing the inter prediction for the luminance component. The Mux #4, #6, and #7 are used to partition the PEs as two concurrent filters while performing the inter prediction for the luminance component and intra prediction. The Mux #5 and #7 are used to implement the accumulation for the intra DC and plane modes. Lastly, the PE1 can perform the addition of two inputs by the Mux #1 and #3 for the intra plane mode.



(a)



(b)

(c)

(d)

Figure 5.12: (a) The Unified Systolic Array for Both Inter and Intra Interpolation. (b) The Block Diagram of Functional Block W. (c) The Weighting Mode of Functional Block W. (d) The Combination Mode of Functional Block W.

The blocks denoted as W in Figure 5.12(a) implement the multiplications of FIR filtering using shift and addition. Moreover, these W blocks can dynamically re-program the filter tap so as to provide various filters. For details, Figure 5.12(b) shows the block diagram of the W block. Specifically, the Mux #3 can configure the W block as two modes: (1) The W block selects one of two input signals using the Mux #1 and weights it by a factor between 1 and 20 using the Mux #2 and #4 as shown in Figure 5.12(c). (2) The W block performs the addition of one input and the other input with weights of 1, 2, 4, or 8 as shown in Figure 5.12(d). Therefore, the multiplexers and the W blocks enable the programmability and reconfigurability of our unified systolic array so as to support all the predictions. In the following, we will detail the data flow for different prediction modes.

5.3.3 Data Flow of the Inter Prediction

For the inter prediction, the luminance and chrominance components are processed differently. However, all the computations are done by the same systolic array. The luminance block requires a fixed 6-tap filter while the chrominance block needs a dynamic 2-tap filter. The actual filter tap used for the chrominance part is determined by the value of motion vectors. Particularly, to increase the throughput, our reconfigurable architecture can be divided into two parts for simultaneously processing Cb and Cr.

5.3.3.1 Luminance Component

The systolic array implementation of luminance interpolation, which requires 9 adders and 5 registers, constructs the whole framework of unified systolic array as shown in Figure 5.13. It is the basic form of a systolic array to implement a fixed, 6-tap FIR filter. The weights of 5 and 20 can be implemented by the W blocks in Figure 5.12(a) as described above.

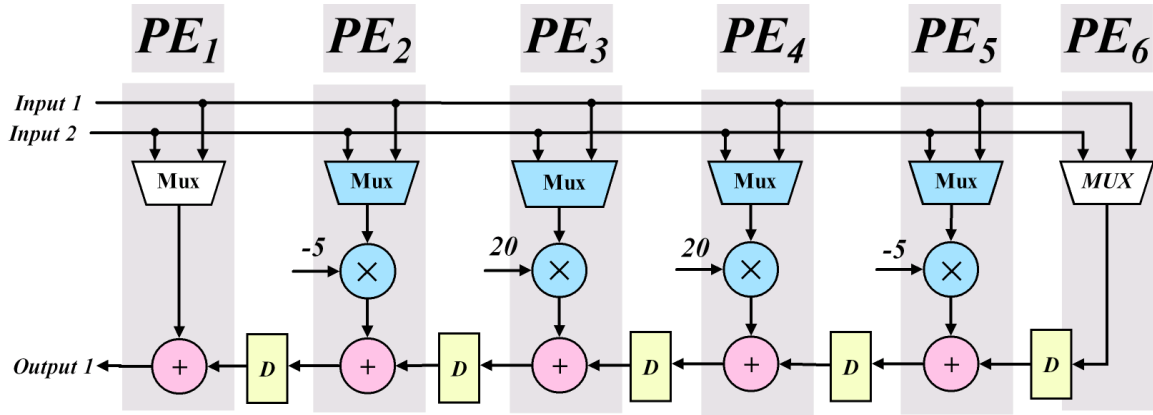


Figure 5.13: The Configuration of Inter Prediction for Luminance Component

Using systolic implementation can efficiently re-use the intermediate terms between consecutive filtering operations and take less bus bandwidth. For better understanding, the consecutive operations of the 6-tap filtering is presented in Figure 5.14,, where $X_{m,n}$ denotes the motion-compensated full-pixel samples, the suffix m specifies the row index, and the suffix n indicates the column index. As shown, the Outputs #3 and #4 have the common term ($X_{0,5}$ times 20). Using the conventional adder-tree architectures, the common terms will not be reused and it introduces redundant computations and higher power consumption. Therefore, by mapping the 6-tap filtering into the systolic array, the intermediate results can be passed through the registers and reused in different PEs. In addition, since the adder-tree designs require 6 neighboring pixels to generate one predictor per cycle, there are 5 redundant pixels transferred via the data bus between the two consecutive executions. In our implementation, however, each motion-compensated pixel in the synchronization buffer is transferred once and is sent to the systolic array one by one. Therefore, the systolic architecture has the advantage of lower bus bandwidth.

Among a variety of systolic arrays, the form of input broadcast has higher throughput by overlapping the filtering operations of the two adjacent rows. Several bubble cycles with invalid data output is conducted between the two adjacent rows if pixel data is fed into the systolic

5.3.3.2 Chrominance Component

Similarly, the chrominance component can be processed with the same manner as the luminance part except that the filtering is of 2-tap. When the motion vector points to the sub-pixel position, the predictor of one sub-pixel a from four full-pixel positions, i.e. A , B , C , and D as shown in Eq.(5.2).

$$a = \left[\begin{array}{l} (8 - dx) \cdot (8 - dy) \cdot A + dx \cdot (8 - dy) \cdot B + \\ (8 - dx) \cdot dy \cdot C + dx \cdot dy \cdot D + 32 \end{array} \right] \gg 6 \quad (5.2)$$

, where the terms dx and dy , whose value is an integer between 1 and 7, indicates the relative location of the motion vector to the full-pixel positions. Instead of the straightforward implementation in [47], we re-write Eq.(5.2) by factor decomposition as follows.

$$a = \left\{ \begin{array}{l} (8 - dy) \cdot [(8 - dx) \cdot A + dx \cdot B] + \\ dy \cdot [(8 - dx) \cdot C + dx \cdot D] + 32 \end{array} \right\} \gg 6$$

$$= [(8 - dy) \cdot a_1 + dy \cdot a_2 + 32] \gg 6 \quad (5.3)$$

, where

$$a_1 = (8 - dx) \cdot A + dx \cdot B$$

$$a_2 = (8 - dx) \cdot C + dx \cdot D$$

From Eq.(5.3), we realize the interpolation of the chrominance component by three separable 2-tap filterings as illustrated in Figure 5.16. First, the pixels "A" and "B" are processed by the 2-tap filter whose tap is $(8 - dx, dx)$ to generate the intermediate result " a_1 ". Second, the pixels "C" and "D" are processed by the same 2-tap filter to generate the intermediate result " a_2 ". Lastly, two intermediate terms " a_1 " and " a_2 " are processed by another 2-tap filter whose tap is

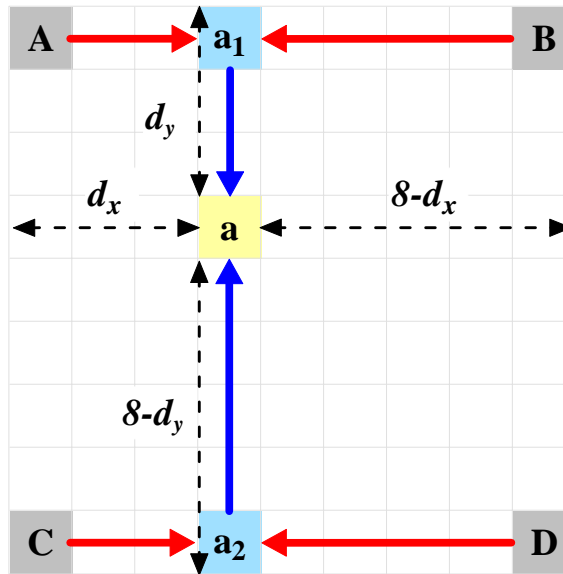


Figure 5.16: Separated Filterings of the Inter Prediction for Chrominance Component

$(8 - d_y, d_y)$ to generate the predictor. Therefore, the interpolation of chrominance component can be implemented by our systolic array.

To increase the hardware utilization, we partition the systolic array into two parts so as to simultaneously process Cb and Cr. For clarity, Figure 5.17 shows the data path for the chrominance interpolation. As shown, the samples of Cb and Cr blocks are fed into the systolic array simultaneously via the two inputs and each color component is separately filtered by the reconfigurable, 2-tap filters. Specifically, each PE can realize the various filter tap between 1 and 7 according to the value of motion vector.

5.3.4 Data Flow of Intra Prediction

For the intra prediction, the computations involved for each mode is different. However, all the computations are done by the same systolic arrays. Generally, the intra prediction requires the accumulator and the two fixed filters whose tap are of $(1, 1)$ and $(1, 2, 1)$. Particularly, to increase the throughput, our reconfigurable architecture can be divided into two parts for

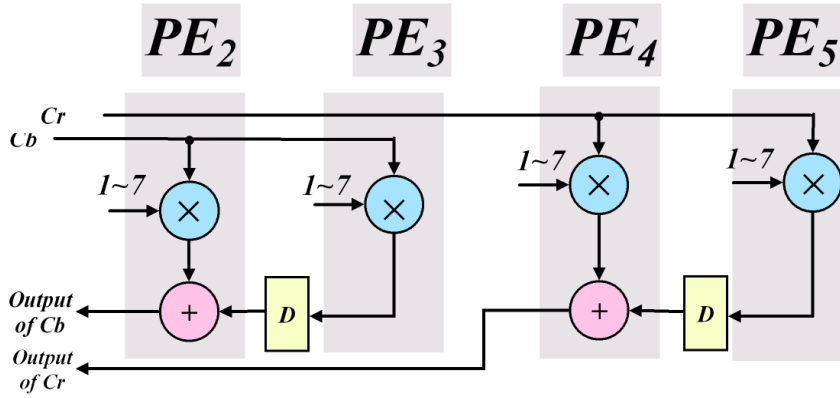


Figure 5.17: The Configuration of Inter Prediction for Chrominance Component

realizing two filters at the same time.

5.3.4.1 Directional Mode

Except for the horizontal and vertical mode, other 6 directional modes require the FIR filtering to generate the predictor. The predictor is actually constructed by a linear combination of the boundary pixels. For example, Eq.(5.4) lists the corresponding formulas for the two predictors in the Vertical_Right mode.

$$\begin{aligned}
 PRED[1,0] &= (P[0,-1] + P[1,-1] + 1) \gg 1 \\
 PRED[0,3] &= (P[-1,2] + 2 \times P[-1,1] + P[-1,0] + 2) \gg 2 \quad (5.4)
 \end{aligned}$$

, where $P[x,y]$ indicates the boundary pixel at the (x,y) position related to the current block.

By reordering the boundary pixels, all the predictors of a sub-block can be obtained by adaptively filtering part of boundary pixels. Different modes simply differ in how the filtering is applied as shown in Figure 5.18. For each mode, part of the boundary pixels are pumped into the systolic array in a fixed order so that the data can be continuously processed to minimize stalls and bubbles. In particular, 2-tap (1, 3) filtering can be extended as 3-tap (1, 2, 1) filtering

when the last pixel is input repeatedly. For clarity, Figure 5.18 shows the data path of the intra prediction for the direction modes. As shown in Figure 5.19(a), the systolic array is configured as two (1, 2, 1) filters for the Diagonal_Down_Left and Diagonal_Down_Right mode because they only require (1, 2, 1) filtering. Otherwise, the systolic array is configured as (1, 1) and (1, 2, 1) filters for other directional modes as shown in Figure 5.19(b) since they require both types of filterings.

5.3.4.2 DC Mode and Plane Mode

In addition to the directional modes, the DC mode and the plane mode require the accumulation operation to sum up the boundary pixels in different ways. The DC mode is useful for prediction in the low pass regions. Therefore, all the predictors of a block with the DC mode are the average value of the boundary pixels. The plane mode is effective for the regions with directional gradient. Each predictor are represented as a function of gradient and spatial coordinate. Specifically, the predictors of plane mode can be calculated by Eq. (5.5).

$$PRED[x, y] = ClipY [(a + b \times (x - 7) + c \times (y - 7) + 16) \gg 5] \quad (5.5)$$

, where

$$a = 16 \times (P[-1, 15] + P[15, -1]),$$

$$b = (5 \times H + 32) \gg 6,$$

$$c = (5 \times V + 32) \gg 6,$$

$$H = \sum_{x'=0}^7 (x' + 1) \times (P[8 + x', -1] - P[6 - x', -1]),$$

$$V = \sum_{y'=0}^7 (y' + 1) \times (P[-1, 8 + y'] - P[-1, 6 - y']).$$

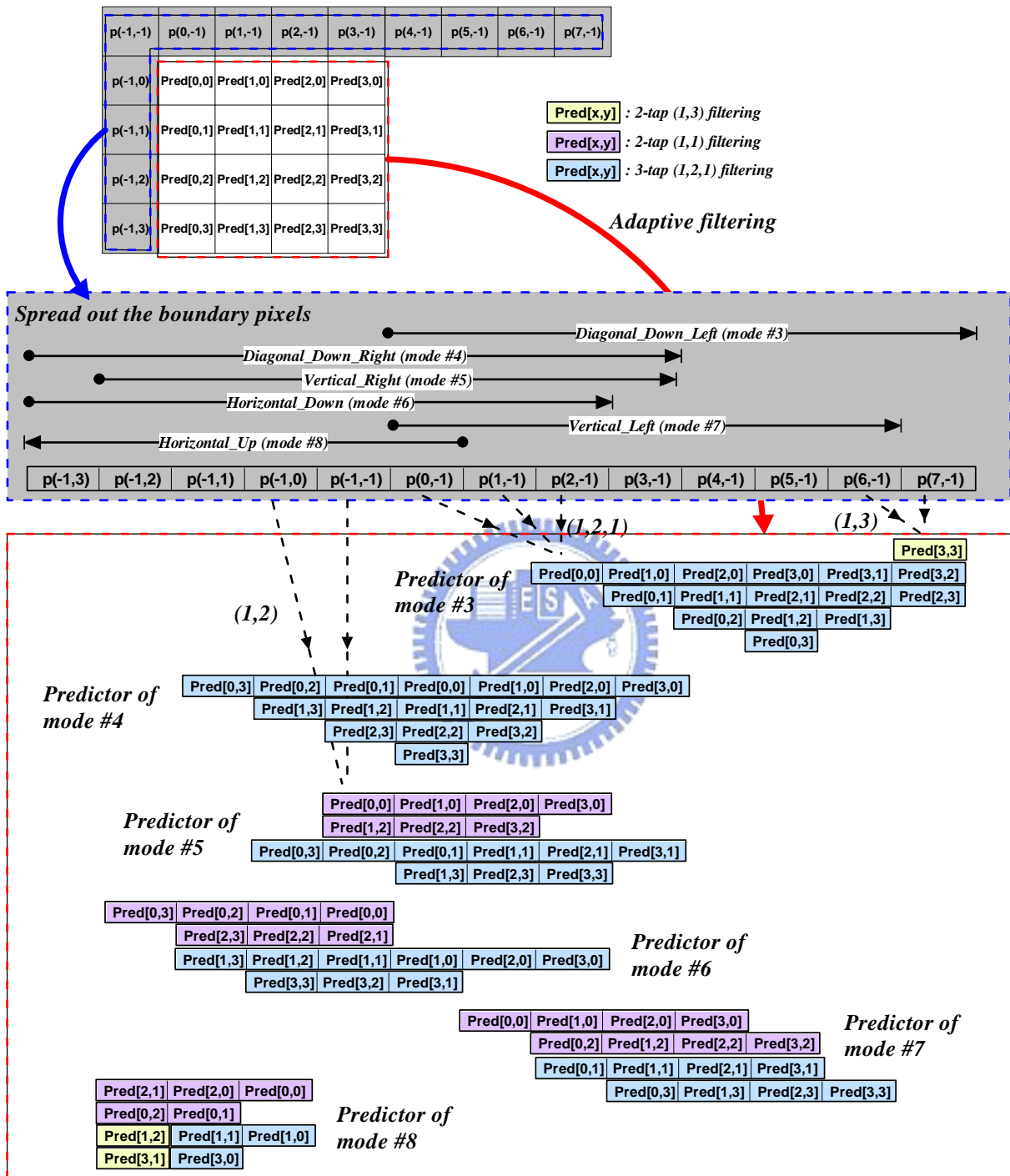


Figure 5.18: Intra Prediction by Adaptive Filtering

The item " H " and " V " are the sum of weighed difference of pixel pairs. The item " b " and " c " can be viewed as the weighted version of " H " and " V " respectively so as to represent the horizontal gradient and vertical gradient. After that, each predictor can be directly calculated from the value " a ", " b ", " c ", and the spatial coordinate. Particularly, the plane mode requires the most computations among all the intra prediction modes. However, in this paper, we generate the predictor in another way to minimize the number of operations.

To remove the operational redundancy in Eq. (5.5), we use the property of arithmetic progression to generate the predictors as shown in Figure 5.20. As shown, the difference of any two successive predictors in the horizontal direction is a constant " b ". Similarly, the difference of any two successive predictors in the vertical direction is a constant " c ". To obtain the predictor of a macroblock, the boundary pixels are firstly weighted and accumulated to obtain " H " and " V ". Then, all the predictors can be calculated from an initial value " a ". The values of predictors in the same row are incremented by a factor of " b ". On the other hand, the predictors in the same column are incremented by a factor of " c ". Therefore, each predictor in the macroblock with plane mode is generated by accumulating " a ", " b ", and " c " without the formula of Eq. (5.5) for each predictor.

Figure 5.21 shows the data path of the intra prediction for the DC mode and the plane mode. Specifically, the feedback loops $F1$ and $F2$ in Figure 5.21(a) are created for the accumulation to generate the average value and the values of " H " and " V " from all the boundary pixels. The gradient values " b " and " c " are calculated by the configuration in Figure 5.21(b). After obtaining the " a ", " b ", and " c ", the prediction of a macroblock with plane block can be generated at the rate of 4 predictors per cycle. In particular, the systolic array has two different configurations at the odd and even cycles as shown in Figure 5.21(c) and Figure 5.21(d). In particular, 5 of 6 PEs operate independently at the odd cycles to generate 4 predictors in the same row as shown

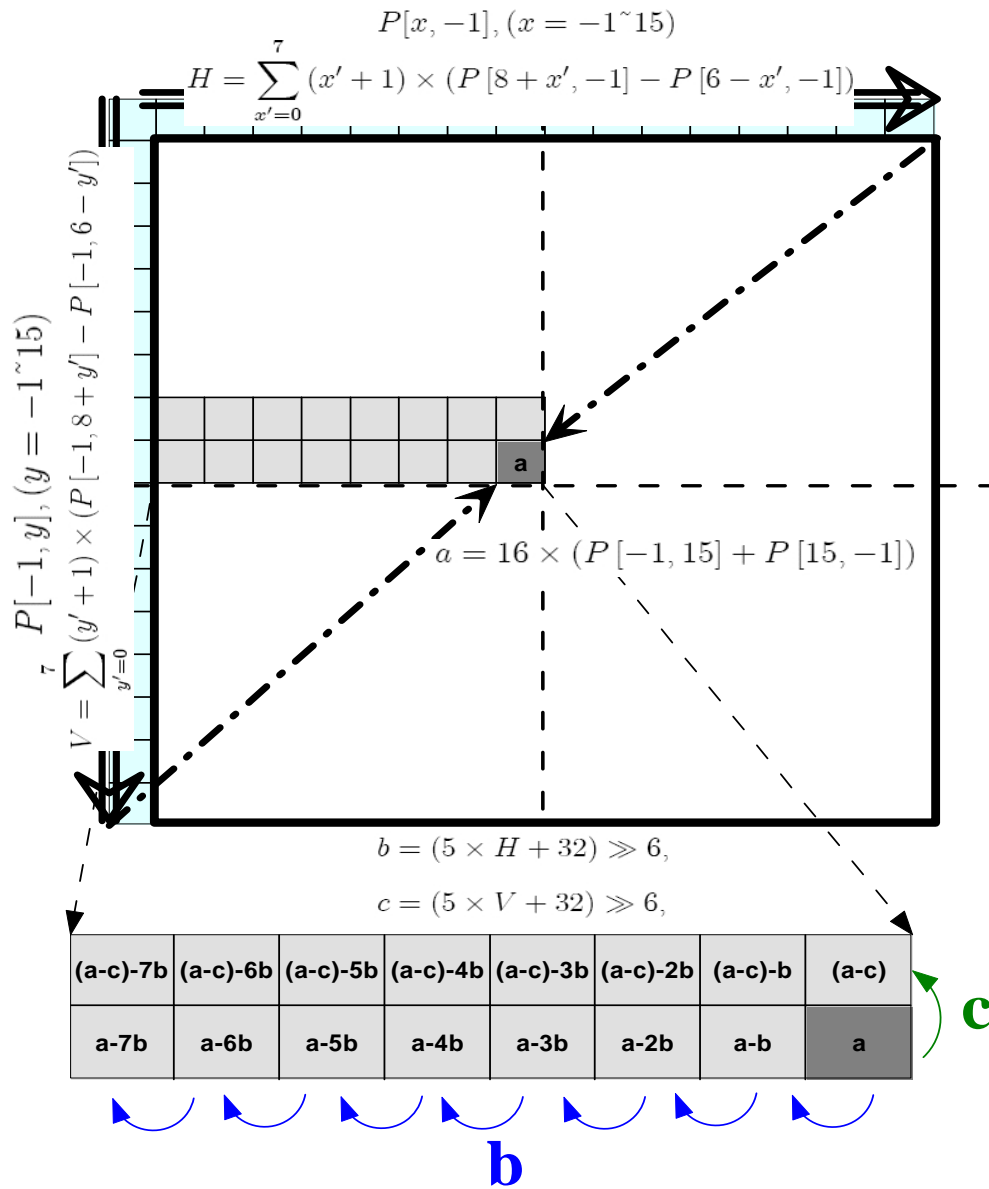


Figure 5.20: The Progression Property of Plane Mode

Table 5.2: The Comparison of the Intra Prediction

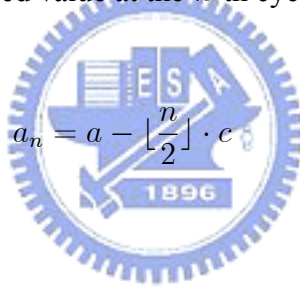
	Huang '04 [77][44]	Proposed
Architecture	Adder Tree (AT)	Separates 1-D Systolic Array (SA)
Component	AT x 4	SA. x 1
# of Adder	$> 4 \times 3 = 12$	$2 \times 3 = 6$
Gate Count	12945	8000
Execution Cycle ¹	> 64 cycles/MB ²	$18+64=82$ cycles/MB ³
# of Input Wires	$8 \times 4 + 13 + 3 + 4 = 52$	2

¹The execution cycle count in worst case (e.g. Intra16x16 Plane Mode)

²Without including computation of H, V, a, b, c

³18 cycles are for the computation of H, V, a, b, c and 64 cycles are for the computation of predictor

in Figure 5.21(c). While the PEs reuse the outcome of neighboring PE at the even cycles to generate 3 predictor in the same row and 1 predictor in the neighboring row as shown in Figure 5.21(d). The item a_n represents the seed value at the n -th cycle as follows.

$$a_n = a - \left\lfloor \frac{n}{2} \right\rfloor \cdot c \quad (5.6)$$


5.3.5 Analysis and Comparison

This subsection compares our systolic architecture with several state-of-the-art designs for the pixel prediction. Table 5.2 shows the comparison of our combined architecture with one level of parallelism and Huang's architecture for for intra prediction [44]. Leaving the computation of " H ", " V ", " a ", " b ", and " c " aside, both designs require equal execution cycles in the worst case, i.e. the intra16 plane mode, while our architecture has lower equivalent gate count. Note that in Huang's design, the inter and intra predictions are separated into two modules [70]. In addition, our combined systolic architecture significantly reduces the number of input wires which leads to lower input bandwidth and cost.

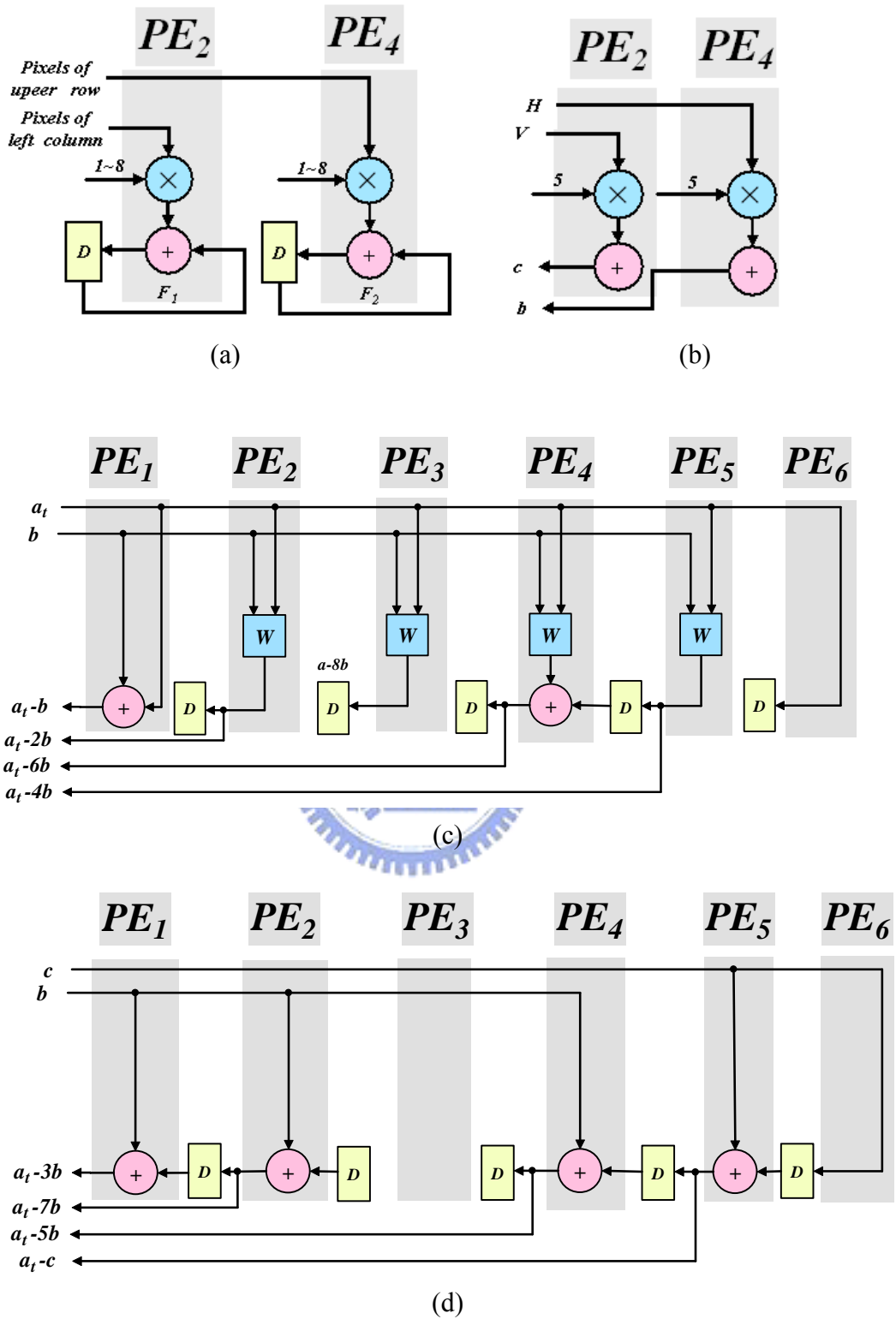


Figure 5.21: The Configuration of Intra Prediction for the DC Mode and the Plane Mode. (a) The Data Path for Accumulation. (b) The Generation of Gradient Values "b" and "c". (c) The Pixel Prediction of Plane Mode at Odd Cycles. (d) The Pixel Prediction of Plane Mode at Even Cycles.

)

Table 5.3 further compares our architecture with the state-of-the-art designs for the inter prediction. As shown, our systolic architecture has lower cost in terms of equivalent gate count. The higher throughput has been achieved by increasing the parallelism of the systolic array, however, our design with three levels of parallelism systolic array still consumes more cycles than Wang's design [47] for the worst case where all the sub-blocks are coded as the Inter_4x4 mode that requires the 2-D interpolation. Wang designed the inter prediction module for the worst case, while our systolic architecture has better throughput performance on the average as listed in Table 5.4. In the following two subsections, we will compare the performance for the general case.

5.3.5.1 The Effect of Parallelism on the Systolic-Based Inter and Intra Prediction

The throughput performance are evaluated and compared for different designs of inter and intra prediction with a wide range of bitstreams as shown in Figure 5.22, Figure 5.23, Figure 5.25, Figure 5.26, Figure 5.27, and Figure 5.28. The sequences including Blue_Sky, Pedestrian_Area, Riverbed, Rush_Hour, Station2, Sunflower, and Tractor are compared at a wide range of bit-rate for four different resolutions covering 240x144, 480x272, 960x544, and 1920x1088. The performance of SA_1 lies in between that of Wang's design and Deng's design while the performance of SA_2 and SA_3 outperform all the other reference designs. As compared to Wang's design [40], our systolic architecture can provide up to 4.5X throughput improvement.

5.3.5.2 Data Transmission via AHB Data Bus at Different Levels of Synchronization Granularity

We continuously analyze the amount of motion-compensated data transmission via AHB data bus at different levels of granularity of video pipe as shown in Figure 5.29, Figure 5.30, Figure

Table 5.3: Comparison of Inter Prediction.

	Wang '03 [40] [34]	Deng '04 [78]	Wang '05 [47]	Proposed		
Architecture	1-D Adder Tree (AT)	Pipelined 2D Adder Tree (AT)	Separated 1D Adder Tree (AT)	Separated 1-D Systolic Array (SA)		
MVG	Software	N/A	Hardware	Hardware		
Component	FIR x 2	Pipelined FIR x 9	Horizontal FIR x 9 & Vertical FIR x 4	SA x 1	SA x 2	SA x 3
	Chroma FIR x 3	N/A	Chroma FIR x 2			
	Bilinear	Bilinear	Bilinear	Bilinear		
# of Adder	$6 \times 2 + 1 + 3 \times 3 = 22$	$> 7 \times 9 = 63$	$6 \times 13 + 23 \times 2 = 124$	$9 \times 1 + 3 = 12$	$9 \times 2 + 3 = 21$	$9 \times 3 + 3 = 30$
Gate Count	11172 ¹	16910 ²	20686	8000	13000	18000
Execution Cycle ³	1616 cycles/MB	800 cycles/MB	576 cycles/MB	1248 cycles/MB	752 cycles/MB	672 cycles/MB
Critical Path	4 adders ⁴	1 18-bit adders	4 adders ⁵	2 adders ⁶	2 adders ⁶	2 adders ⁶
# of input wires	$6 \times 2 + 2 \times 3 = 18$	$> 13 \times 2 = 26$	$6 \times 4 + 4 \times 2 = 32$	2	4	6

¹Without including the cost of SRAM

²Without including the cost of chroma filter

³The execution cycle count in worst case

(e.g. All are blocks are coded as Inter4x4 mode that requires 2-D interpolation)

⁴A 18-bit adder + two 16-bit adders + a 13-bit adder + a 13-bit multiplexer + a 18-bit multiplexer

⁵A 18-bit adder + two 16-bit adders + a 13-bit adder

⁶A 18-bit adder + a 15-bit adder + a 13-bit multiplexer + two 16-bit multiplexers + a 18-bit multiplexer

Table 5.4: The Execution Cycle of the Inter and Intra Prediction in All Cases

Mode	A ¹	B ²	C ³	SA_1 ⁴	SA_2 ⁴	SA_3 ⁴
Intra_4x4_DC	32	32	32	80	80	80
Intra_4x4_Diagonal_Down_Left	64	64	64	80	80	80
Intra_4x4_Diagonal_Down_Right	64	64	64	80	80	80
Intra_4x4_Vertical_Right	64	64	64	112	112	112
Intra_4x4_Horizontal_Down	64	64	64	112	112	112
Intra_4x4_Vertical_Left	64	64	64	112	112	112
Intra_4x4_Horizontal_Up	64	64	64	112	112	112
Intra_16x16_DC	4	4	4	17	17	17
Intra_16x16_Plane	64	64	64	64	64	64
Inter_4x4 at 2-D	1616	800	576	1248	752	672
Inter_4x4 at 2 1-D	592	800	576	1008	672	544
Inter_4x4 at Horizontal 1-D	272	800	336	464	288	288
Inter_4x4 at Vertical 1-D	272	800	288	464	256	256
Inter_4x8 at 2-D	1528	640	568	960	544	504
Inter_4x8 at 2 1-D	504	640	568	944	552	472
Inter_4x8 at Horizontal 1-D	280	640	240	416	240	240
Inter_4x8 at Vertical 1-D	232	640	712	376	208	208
Inter_8x4 at 2-D	1528	640	448	1120	568	480
Inter_8x4 at 2 1-D	504	640	448	880	552	512
Inter_8x4 at Horizontal 1-D	232	640	328	376	232	232
Inter_8x4 at Vertical 1-D	280	640	280	416	240	240
Inter_8x8 at 2-D	1452	512	444	824	484	348
Inter_8x8 at 2 1-D	428	512	444	740	432	348
Inter_8x8 at Horizontal 1-D	236	512	324	356	200	200
Inter_8x8 at Vertical 1-D	236	512	708	356	196	172
Inter_16x8 at 2-D	1418	128	444	852	456	332
Inter_16x8 at 2 1-D	394	128	444	588	308	204
Inter_16x8 at Horizontal 1-D	214	128	324	338	174	142
Inter_16x8 at Vertical 1-D	234	128	708	298	162	158
Inter_8x16 at 2-D	1418	128	444	852	456	332
Inter_8x16 at 2 1-D	394	128	444	588	308	204
Inter_8x16 at Horizontal 1-D	214	128	324	338	174	142
Inter_8x16 at Vertical 1-D	234	128	708	298	162	158
Inter_16x16 at 2-D	1391	128	444	760	390	75
Inter_16x16 at 2 1-D	367	128	444	680	342	235
Inter_16x16 at Horizontal 1-D	213	128	324	333	169	112
Inter_16x16 at Vertical 1-D	213	128	708	333	169	112
Average	469	334	365	485	282	234

¹A Indicates the Combination of Huang '04 [44] Intra Prediction and Wang '03 [40] Inter Prediction

²B Indicates the Combination of Huang '04 [44] Intra Prediction and Deng '04 [78] Inter Prediction

³C Indicates the Combination of Huang '04 [44] Intra Prediction and Wang '05 [47] Inter Prediction

⁴SA Indicates the Systolic Array

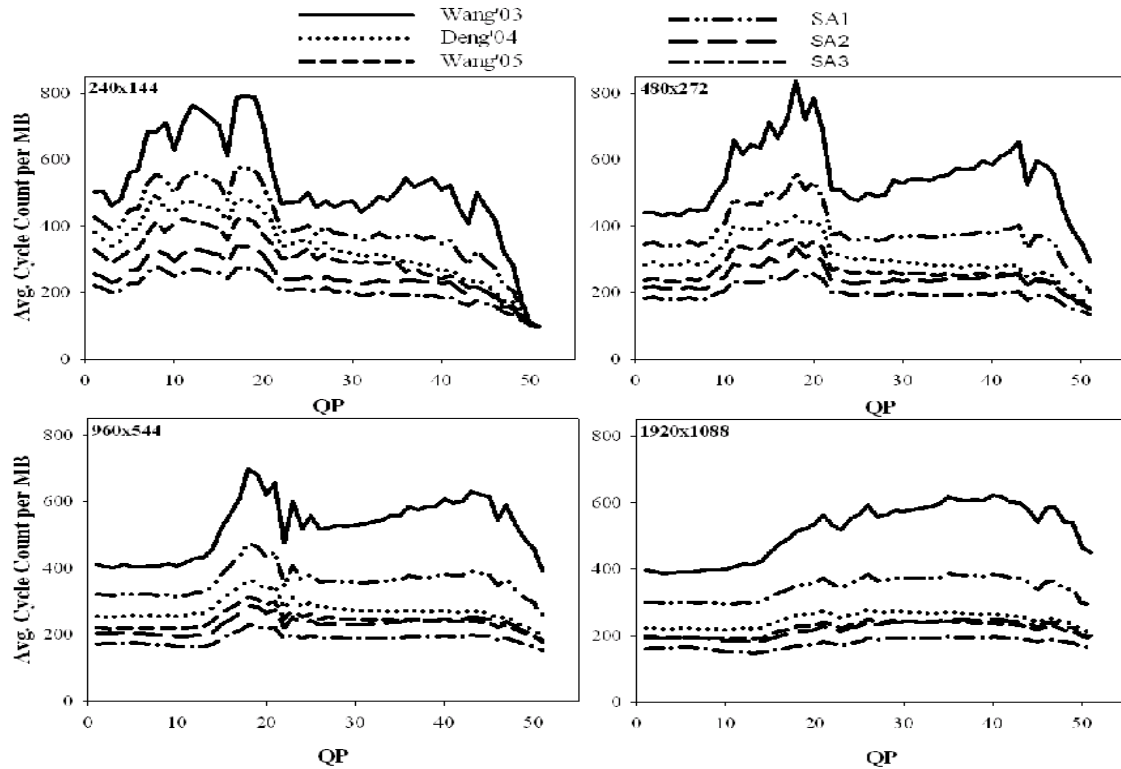


Figure 5.22: The Execution Cycle of the Inter and Intra Prediction for the Blue_Sky Sequence

5.32, Figure 5.33, Figure 5.34, and Figure 5.35. As shown, the amount of data transmission in our systolic designs is significantly decreased when the granularity is moved from 4x4 block to 8x8 block. While the decrease is limited as the granularity moves from 8x8 block to 16x16 block. However, our designs always conduct less data transmission than AT-based design because the motion-compensated data in a pipeline stage is transferred once without redundant transmission in our design. Particularly, our design can reduce up to 60% of the data transmission.

In summary, an efficient FIR implementation is proposed to combine the inter and intra predictions for intensive computation. The key concept of combined inter and intra prediction is to allow flexible reconfiguration of filter so as to create different filter taps on-the-fly. Using the control path, the unified systolic array can be configured as a 2-, 3-, or 6-tap filter and the filter tap can be selected on-the-fly to implement the adaptive filtering. We can achieve

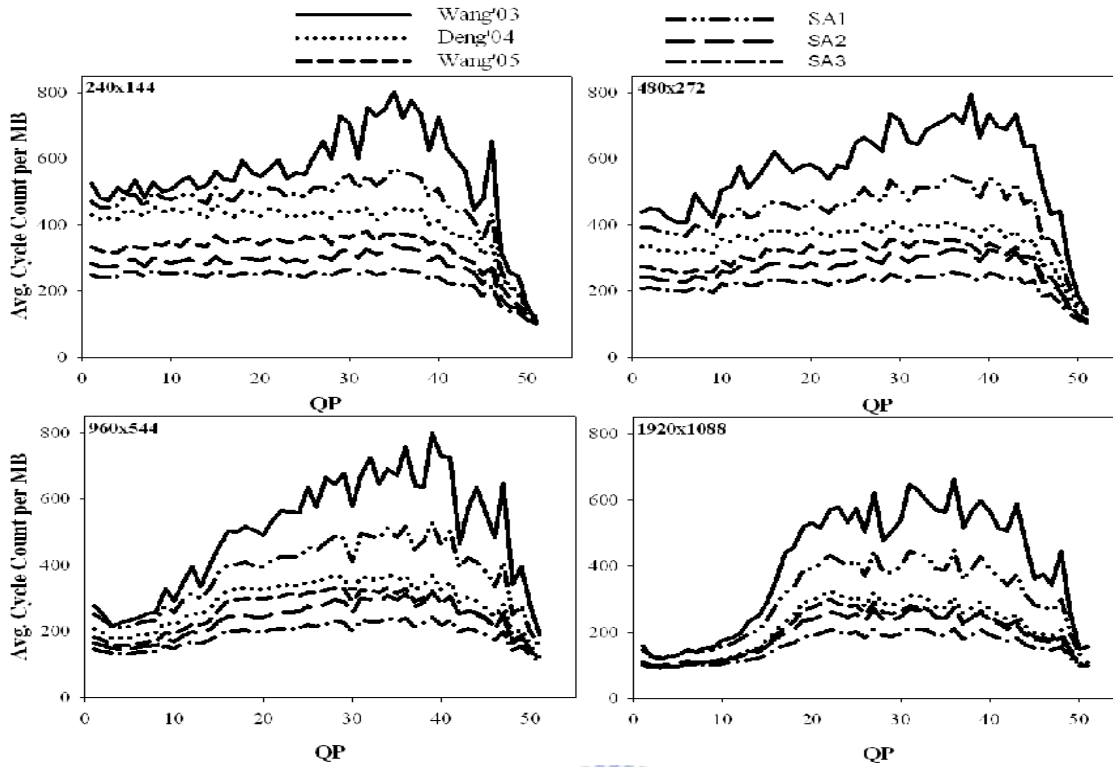
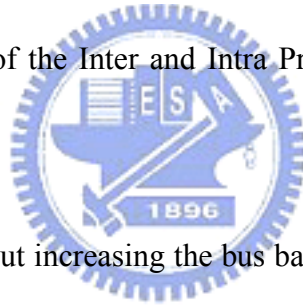


Figure 5.23: The Execution Cycle of the Inter and Intra Prediction for the Pedestrian_Area Sequence



higher computation throughput without increasing the bus bandwidth and memory bandwidth. Moreover, the data paths are shared between inter and intra predictions which is more efficient from the system perspective.

5.4 Efficient Deblocking Filter with Fine-Grained Synchronization Capability

In this section, we propose an efficient deblocking filter that provides a fine-grained synchronization capability (FGSC). Most deblocking filter designs focus on the data transportation and the filtering order instead of the kernel filter design [48][49][50][51][52][53][54][55][56][57][58][59]. However, all of them used the macroblock-based filtering orders, which are inefficient for

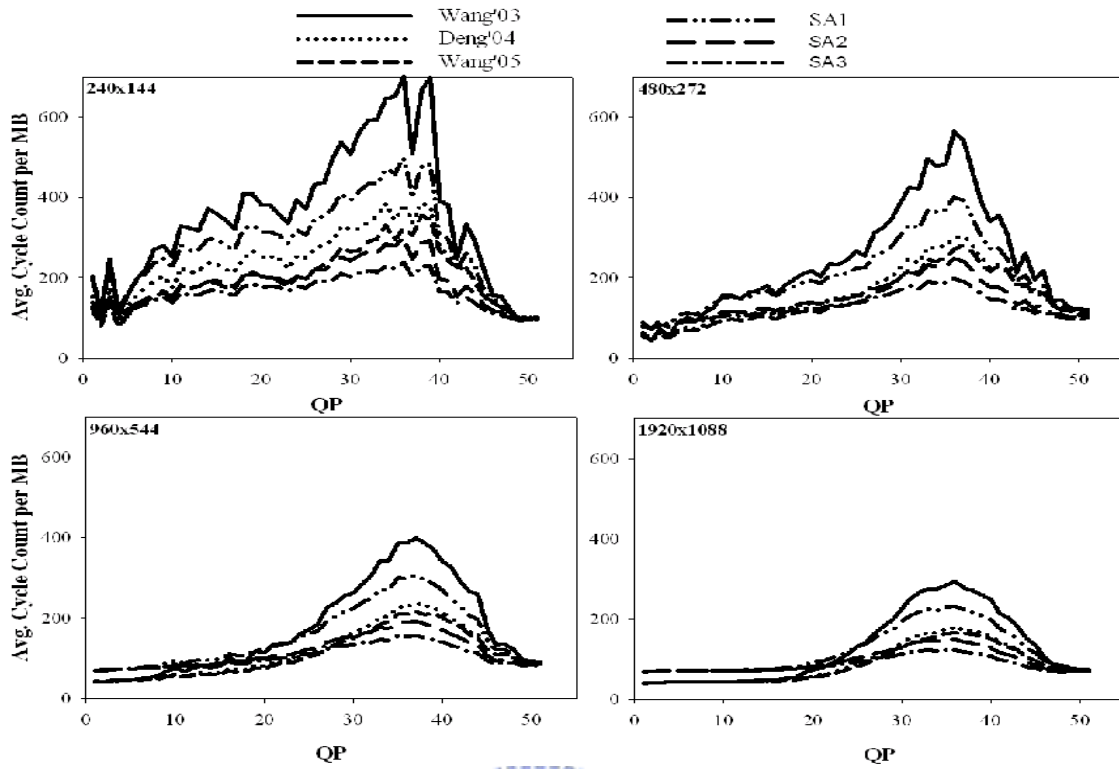


Figure 5.24: The Execution Cycle of Inter and Intra Prediction for Riverbed Sequence.

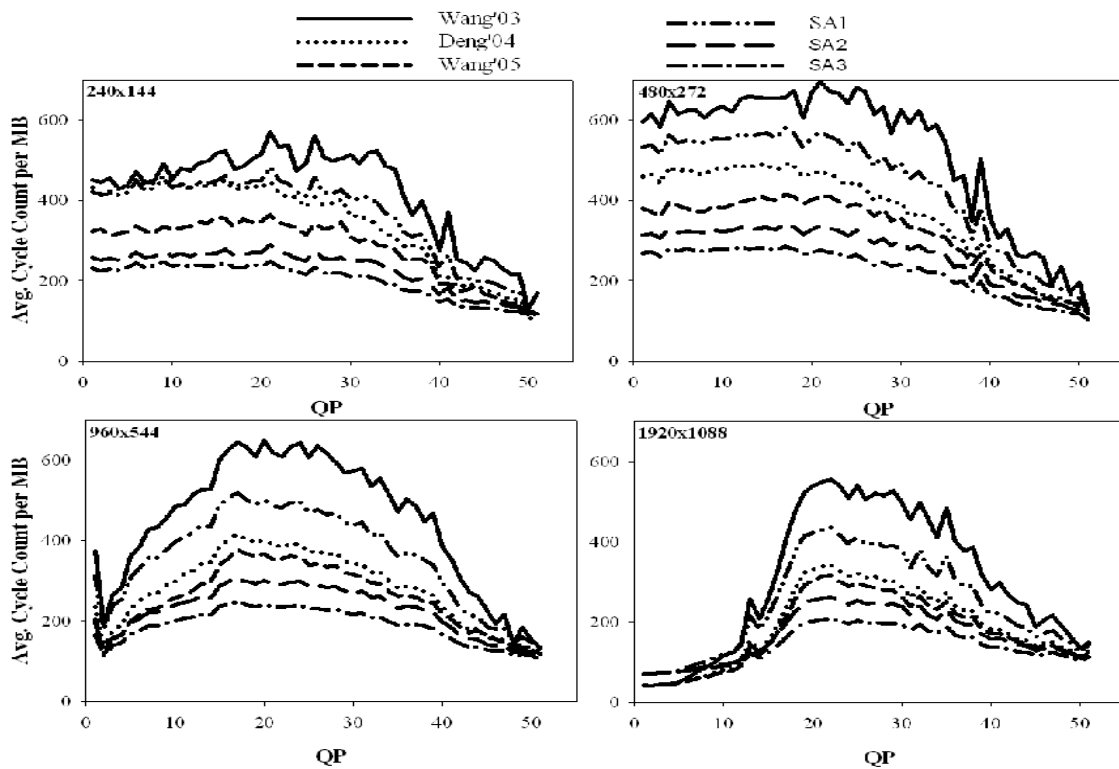


Figure 5.25: The Execution Cycle of Inter and Intra Prediction for Rush_Hour Sequence.

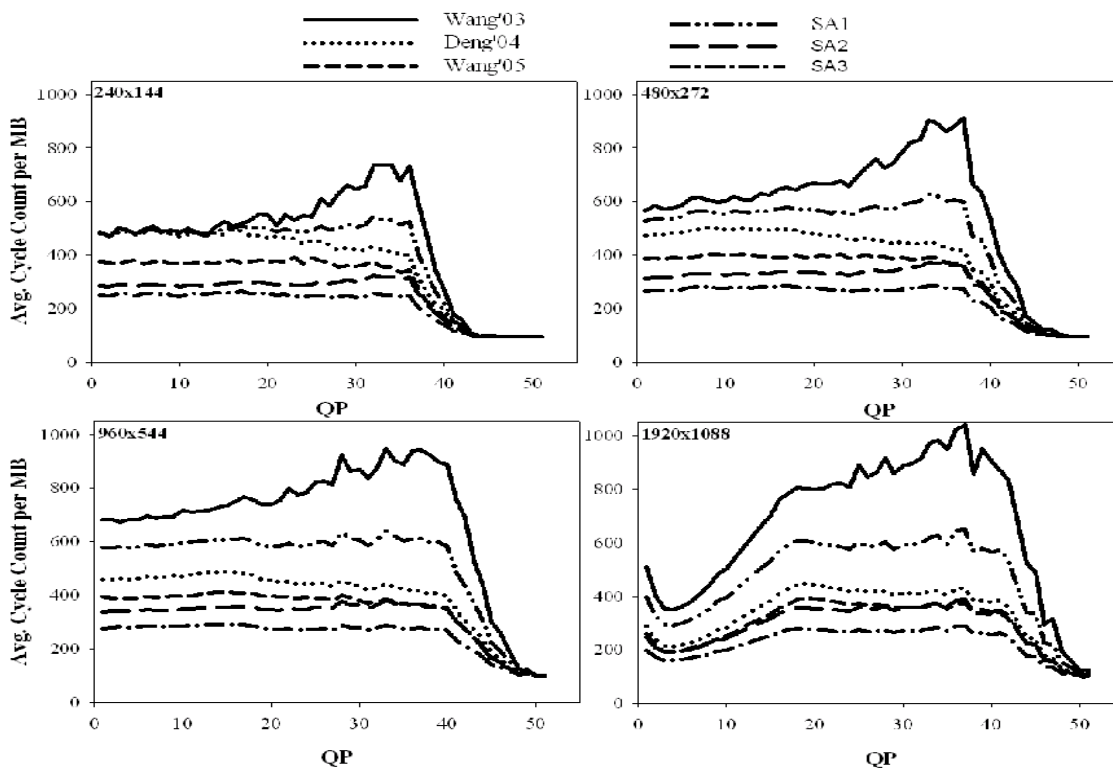


Figure 5.26: The Execution Cycle of the Inter and Intra Prediction for the Station2 Sequence

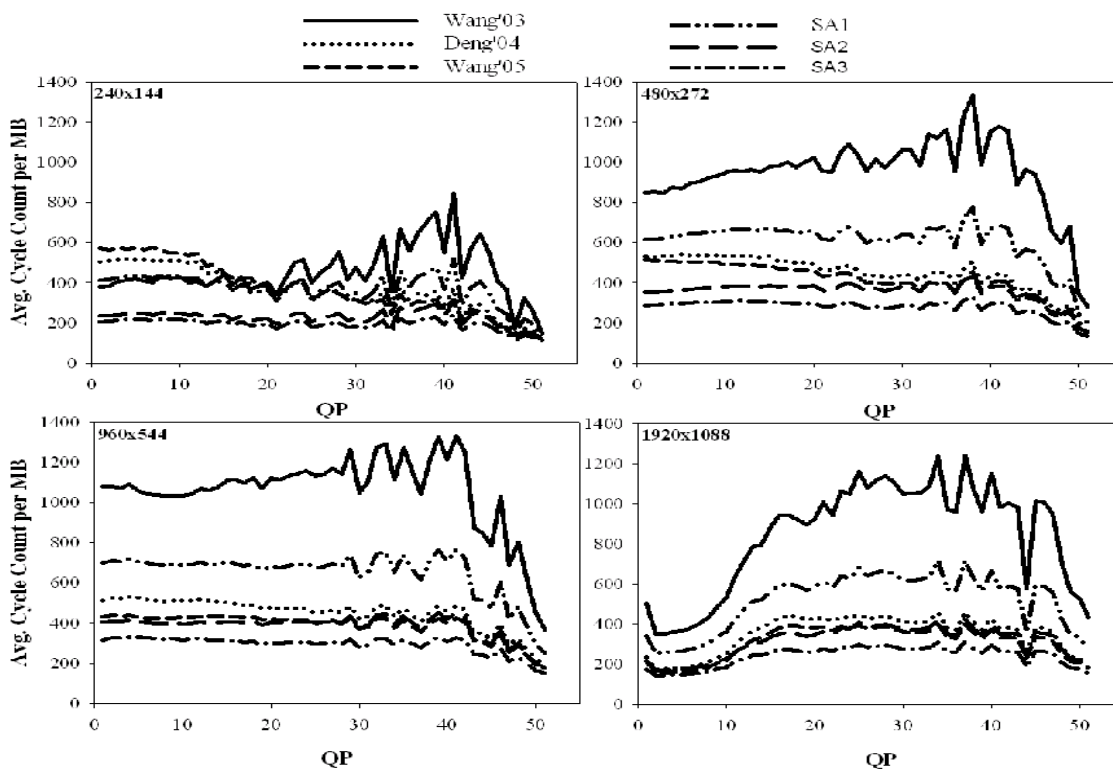


Figure 5.27: The Execution Cycle of the Inter and Intra Prediction for the Sunflower Sequence

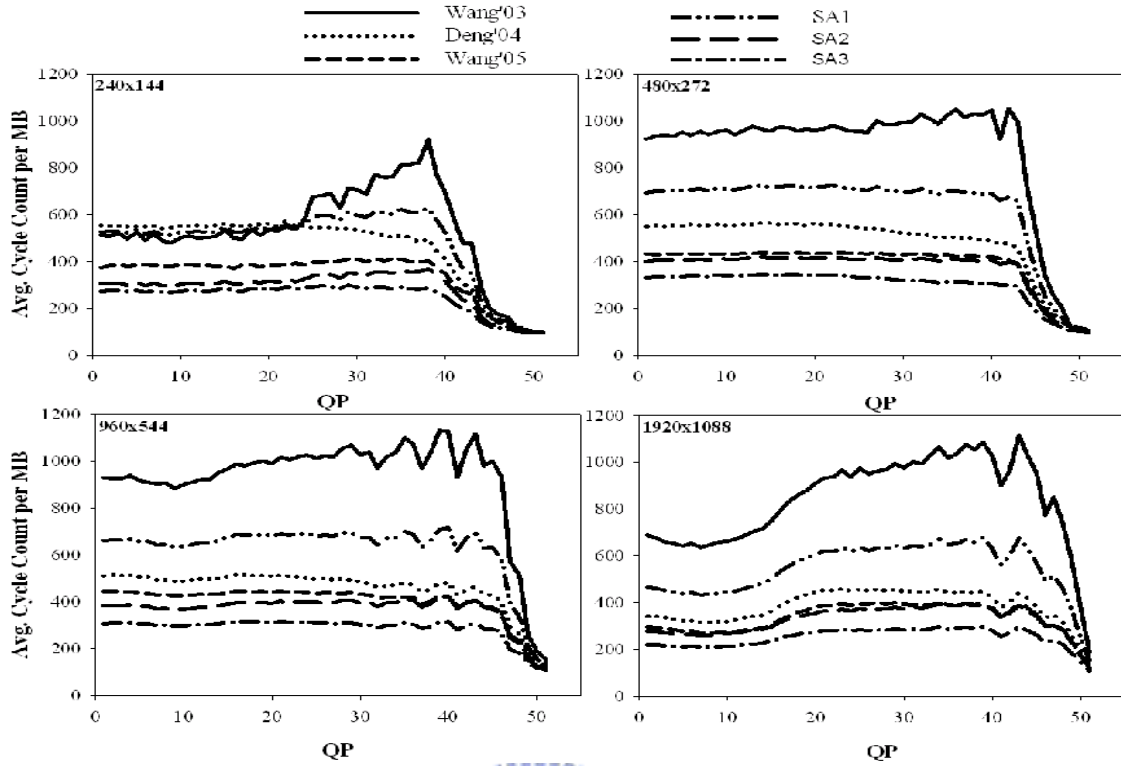


Figure 5.28: The Execution Cycle of the Inter and Intra Prediction for the Tractor Sequence

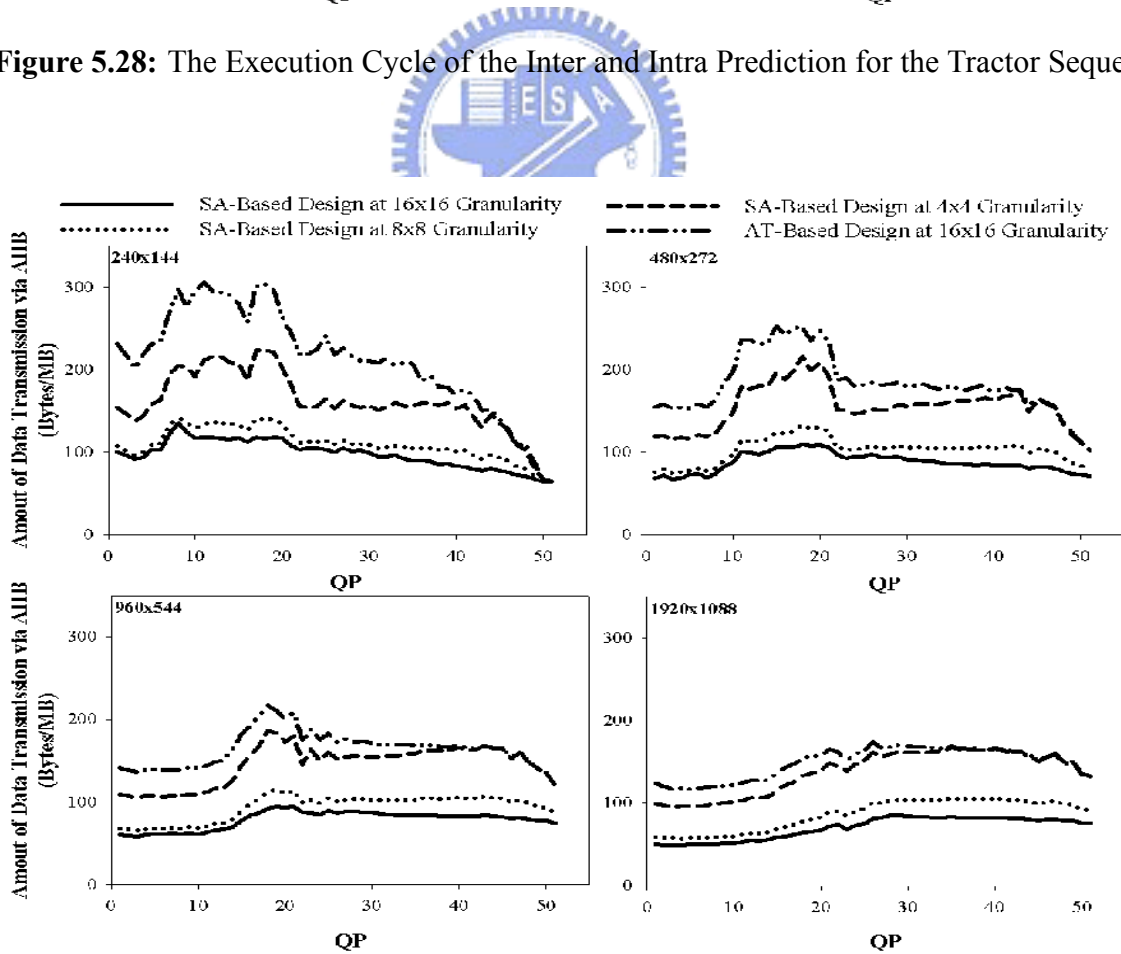


Figure 5.29: The Data Transmission via AHB Data Bus for the Blue_Sky Sequence

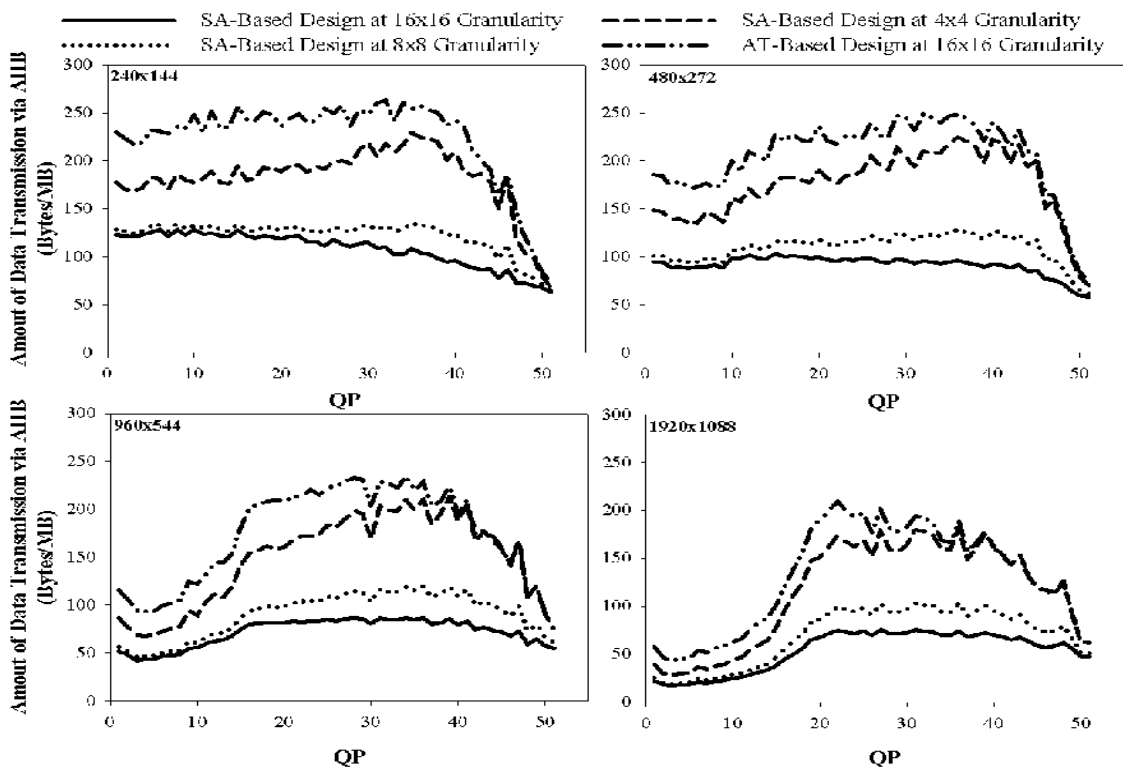


Figure 5.30: The Data Transmission via AHB Data Bus for the Pedestrian_Area Sequence

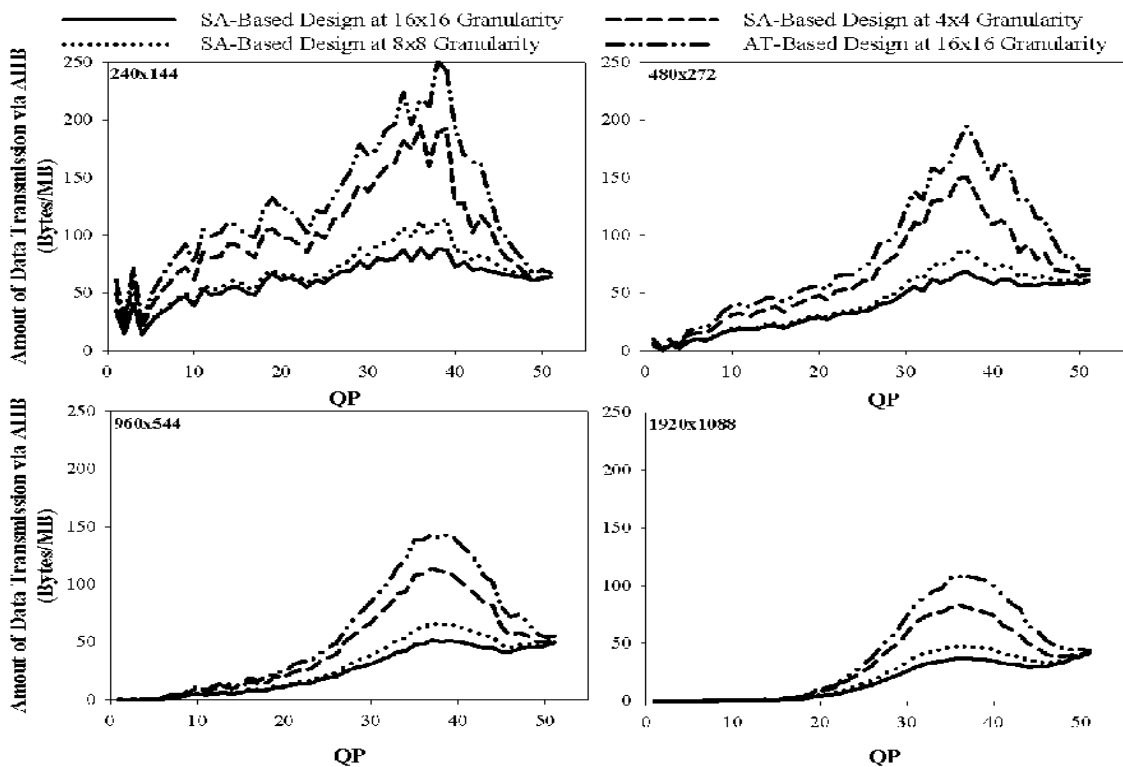


Figure 5.31: The Data Transmission via AHB Data Bus for the Riverbed Sequence

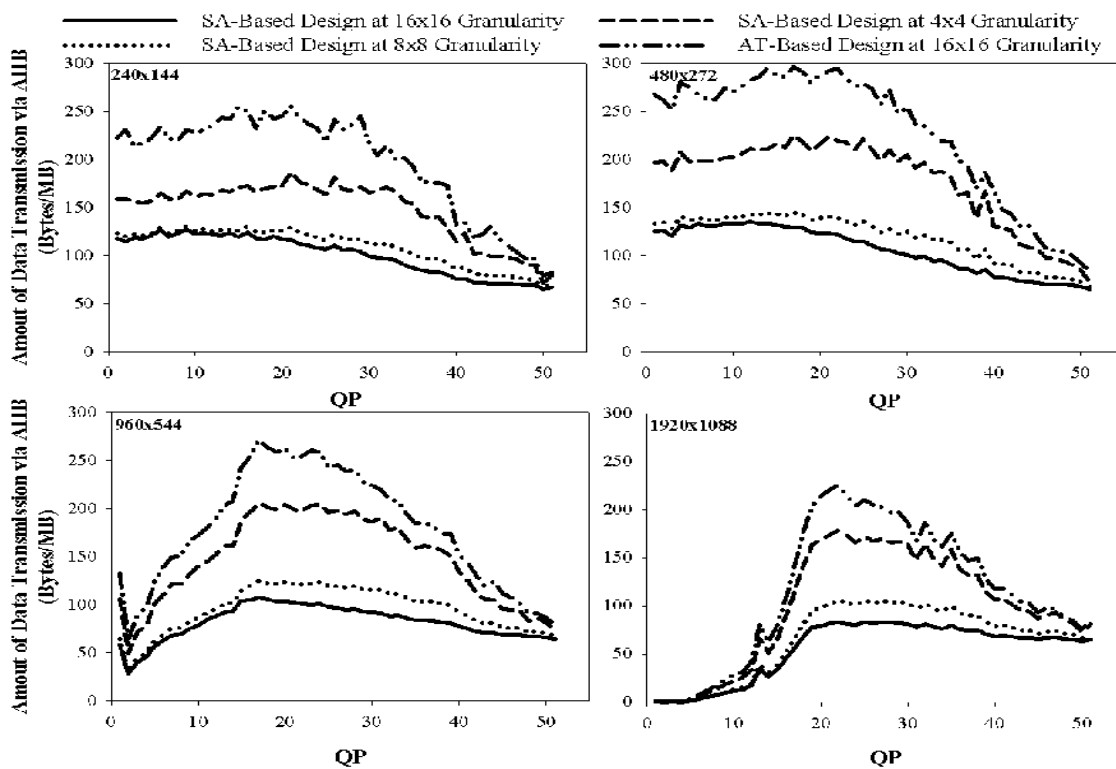


Figure 5.32: The Data Transmission via AHB Data Bus for the Rush_Hour Sequence

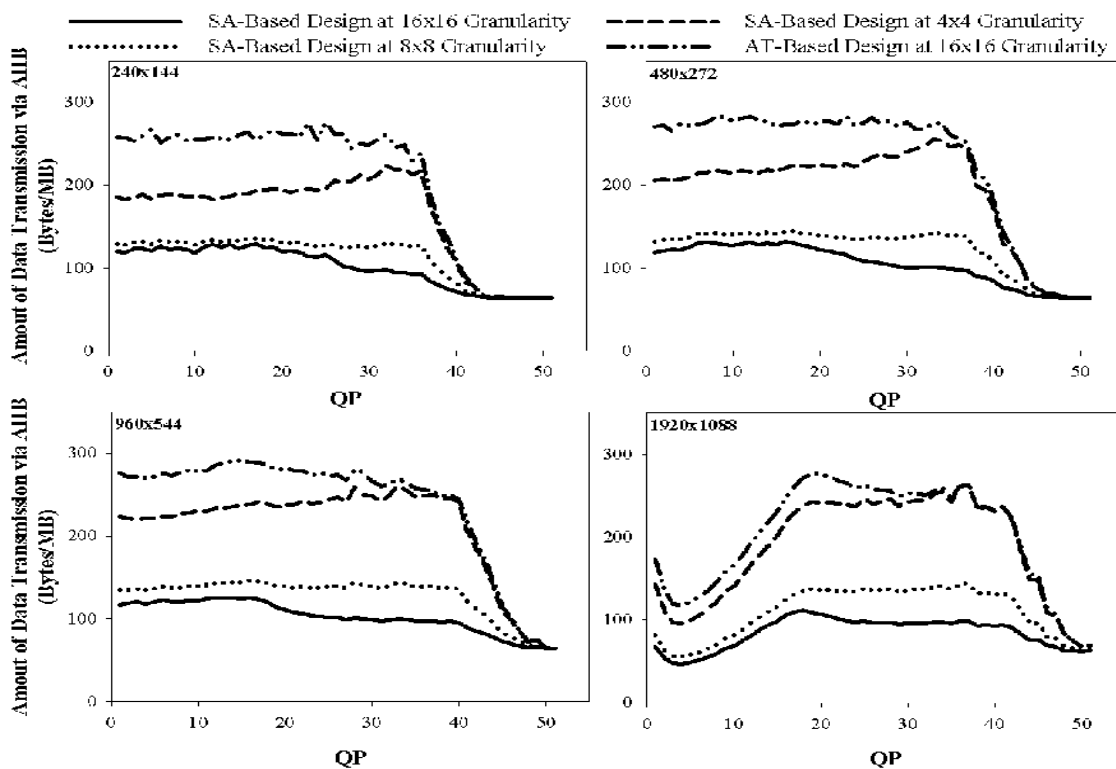


Figure 5.33: The Data Transmission via AHB Data Bus for the Station2 Sequence

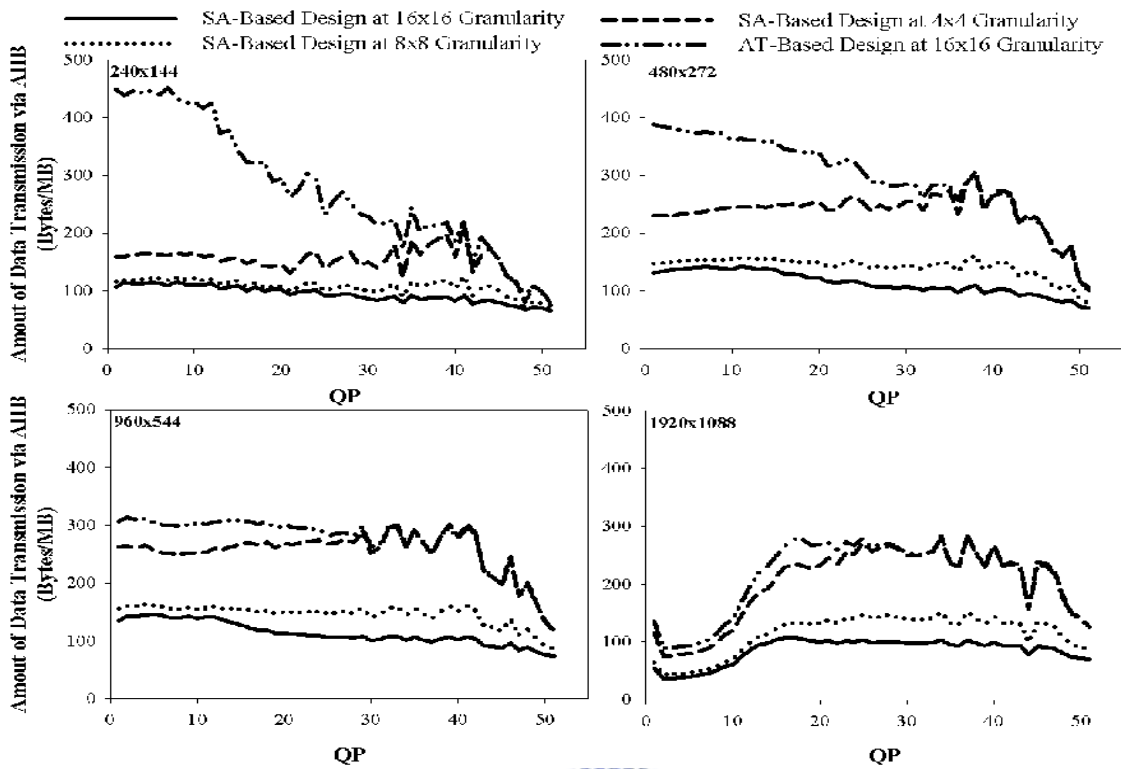


Figure 5.34: The Data Transmission via AHB Data Bus for the Sunflower Sequence

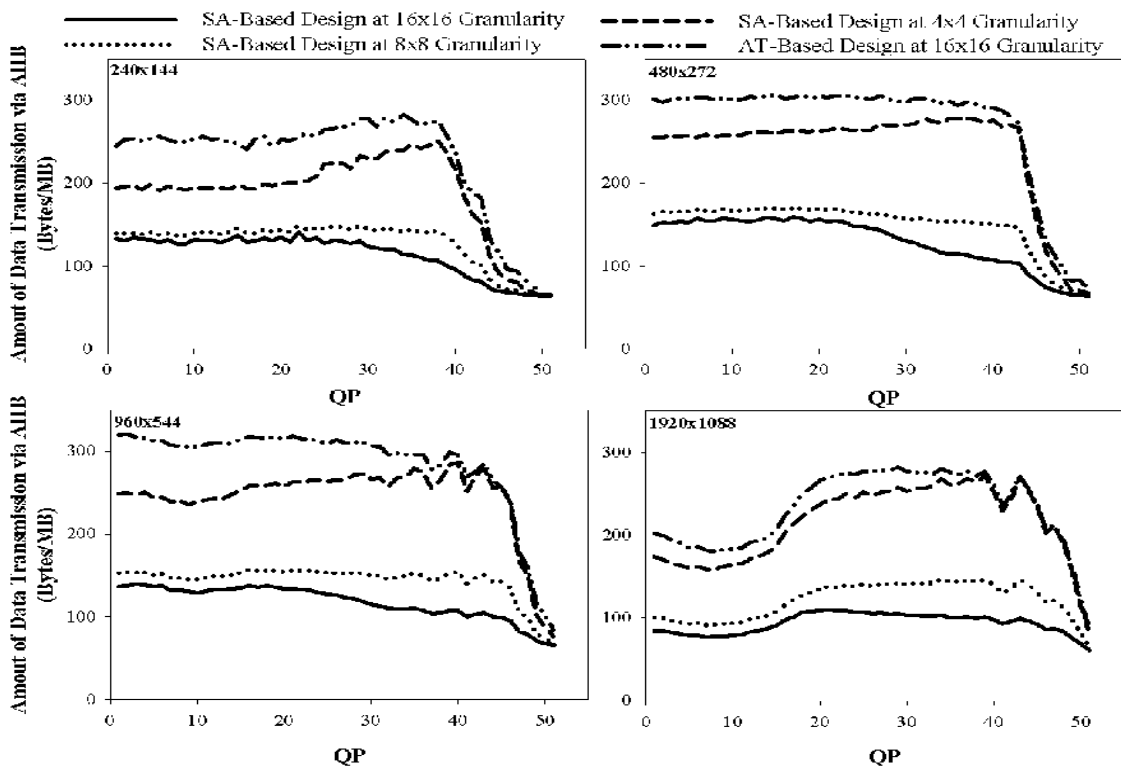


Figure 5.35: The Data Transmission via AHB Data Bus for the Tractor Sequence

the finer granularity of pipeline synchronization. The deblocking filter processes the reconstructed data produced by the previous modules in the different pipelined stages. Therefore, the macroblock-based deblocking filter needs to suspend the computation and buffer the reconstructed data until the whole macroblock is received. The integration of the macroblock-based deblocking filter into a finer granularity pipelined system increases both the memory requirement and the processing latency. In the following discussions, we will describe the architecture, the data flow, and the filtering order of our proposed deblocking filter.

5.4.1 The Proposed Architecture of Deblocking Filter

To reduce the memory and the latency for buffering, we propose a novel architecture for fine-grain synchronization as shown in Figure 5.36. The 1-D deblocking filter implements the conditional filtering as specified in the H.264/AVC standard. It performs one filtering operation per cycle according to the coding information in the control register. The two pixel arrays buffer the output of the kernel filter and adaptively transpose the pixels. Specifically, the pixels of a 4x4 block are transposed from row major order to column major order after the horizontal filtering. When the pixels are fully filtered on both directions, they are transposed back to row major order to be consistent with the data arrangement in the external DRAM. Meanwhile, the two internal memory units store the pixels until they have been fully filtered. Similarly, the size of the internal memories depends on the granularity of synchronization. For instance, in the granularity of 8x8 block, the size of two register files is 36x32 and 64x32 respectively. The intermediate pixels are appropriately separated into two register files such that the 1-D filter can input 8 pixels and output 8 pixels at each cycle. To this end, the controller can configure the data path according to the specific edges.

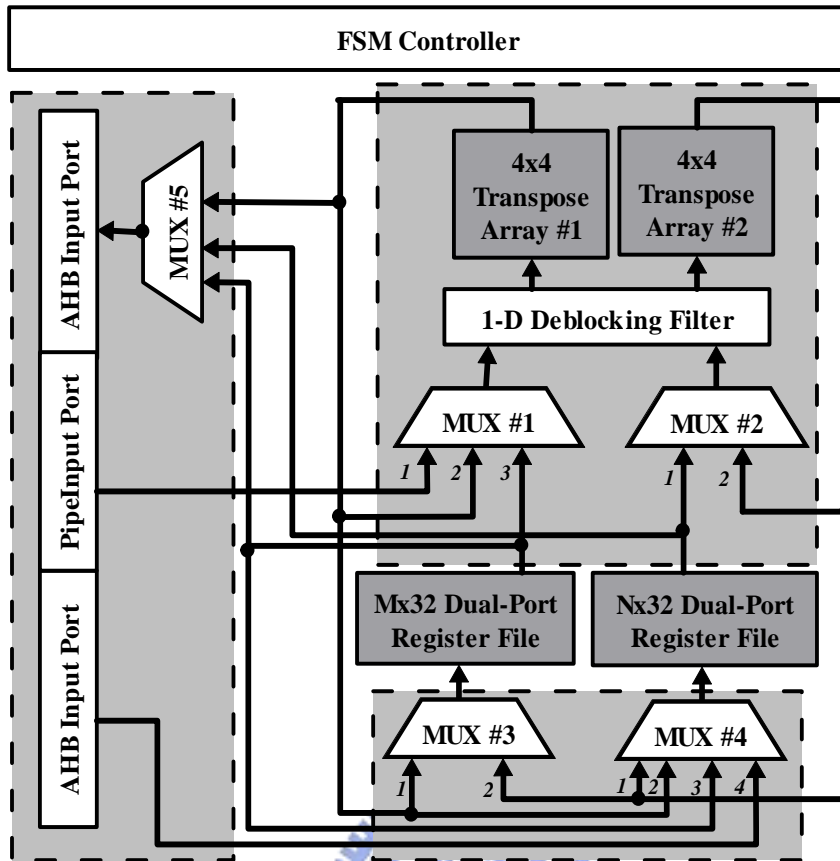


Figure 5.36: The Architecture of the Proposed Deblocking Filter

5.4.2 The Proposed Filtering Order with Fine-Grained Synchronization Capability

The proposed filtering scheme adopts the fine-grain processing order as shown in Figure 5.37 where the numbers in the circles stand for the order of the edge filtering between the two adjacent 4x4 blocks and the corresponding index of data path of each edge is illustrated by different shading. To reduce the overhead with the reloaded data from the memory units, we interleave the horizontal filtering and the vertical filtering while the filtered results are still compliant.

As compared to the macroblock-based order, the proposed filter order can reduce both memory requirement and processing latency. In the finer granularity such as 8x8 block or 4x4 block, our proposed deblocking filter can efficiently process the input data at each pipelined stage.

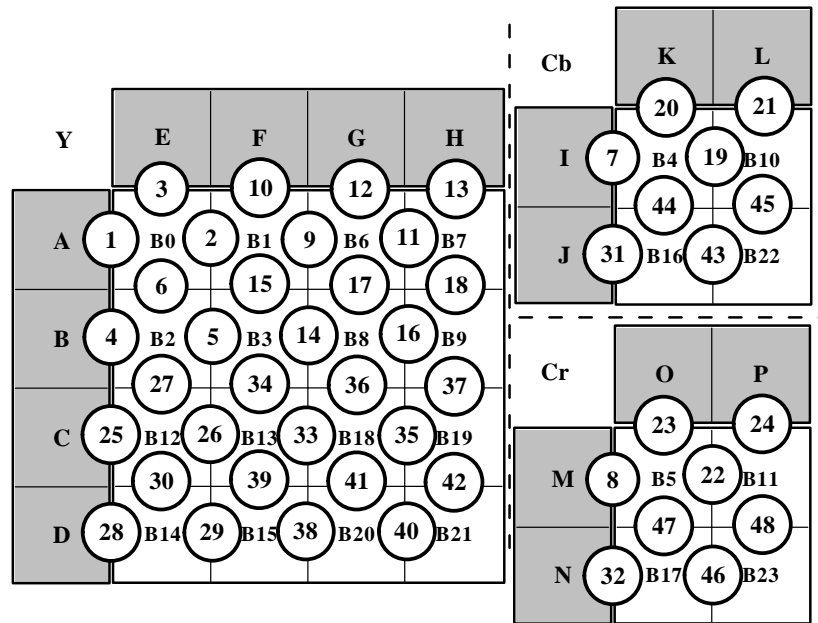


Figure 5.37: The Proposed Edge Filtering Order with Fine-Grained Synchronization Capability

That is, the proposed processing order is able to filter pixels as soon as the data comes in.



5.5 Summary

In this chapter, we focus on the three hot spot micro-architecture designs including the memory sub-system, the prediction module and the deblocking filter. First, an efficient memory sub-system is proposed to feature the interleaved data arrangement, the unified FSM with content-adaptive auto precharge, and the internal synchronization buffer. Second, the inter and intra prediction is combined in a highly utilized systolic architecture to achieve throughput up to 4.5 times while decreasing up to 65% of the bus bandwidth as compared to the conventional adder-tree based designs. Third, a novel deblocking filter is proposed with a fine-grained synchronization granularity such that the efficiency can be improved at different levels of granularity.

Table 5.5: The Corresponding Data Path of Each Filtered Edge

# of Data Path	Corresponding Edge Index	Selected Input of MUX#1	Selected Input of MUX#2	Selected Input of MUX#3	Selected Input of MUX#4	Configure of TA ¹ #1	Configure of TA ¹ #2
1	1,4,7,11,14,16,25,28,31,33,35,40	1	1	1	1	T ²	N ³
2	2,5,26,29	1	2	1	x ⁴	N	N
3	3,6,10,15,27,30,34,39	2	1	x	1	T	N
4	8,32	1	1	1	1	N	N
5	12,17,36,41	2	1	2	x	T	T
6	13,18,21,24,37,42,45,48	3	1	1	1	T	N
7	19,22	1	1	2	2	T	N
8	44,47	2	1	2	3	T	T
9	38	1	1	1	1	N	T
10	43,46	1	1	1	1	T	T

¹Transpose array

²T indicates transposed configuration.

³N indicates normal configuration without transposing the pixel array.

⁴x means don't care.

CHAPTER 6

Conclusions



To implement the video embedding service while meeting the technical challenges of the high complexity of the cascaded transcoder, we have proposed a low-complexity video embedding transcoder at both algorithm and architecture level in this dissertation.

In Chapter 2, we introduce the background of this work and describe the technical challenges of the H.264/AVC-based transcoding. To reduce the complexity while improving the R-D performance, a low-complexity algorithm was proposed in Chapter 3. Moreover, system architecture of the proposed VET algorithm and its design space exploration were presented in Chapter 4. Lastly, in Chapter 5, three highly utilized micro-architecture designs were proposed. In the following, we summarize this dissertation in a number of major contributions and show how this work can be further improved.

6.1 Summary of Contributions

The main contributions of this dissertation are summarized in the following subsection.

6.1.1 Improvement of Rate-Distortion Performance

In this dissertation, a low-complexity VET algorithm is firstly developed to minimize complexity while delivering high coding efficiency with limited drifting errors. The proposed reduced frame memory transcoding (RFMT) reduces half of the frame memory and employs three refinement techniques including the intra mode switching (IMS), the motion vector re-mapping (MVR) and the syntax level bypassing (SLB).

- All the inference and reference mismatch are refined during the VET transcoding such that the visual artifacts can be significantly removed.
- Two low-complexity refinement techniques including the intra mode switching and the motion vector re-mapping are employed such that the wrong reference problem can be significantly alleviated.
- Only a few blocks are effectively refined to correct the drift error such that the video quality can be further improved.
- The proposed refinement techniques result in a negligible quality loss as compared to the rate-distortion-optimized refinement.
- The methodology of partial re-encoding is utilized and it not only reduces the complexity of transcoding, but also preserves the picture quality. As compared to the cascaded transcoder, the proposed algorithm increases the transcoding throughput by 25 times while providing 0.3~1.5dB PSNR improvement.

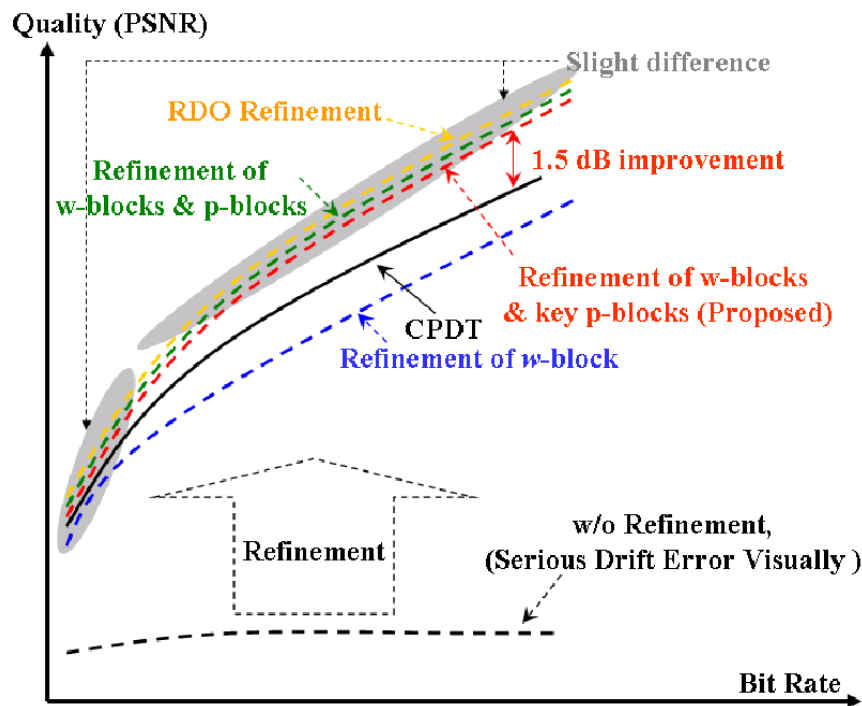


Figure 6.1: The Improvement of Rate-Distortion Performance by the Proposed Low-Complexity Algorithm



6.1.2 Design Space Exploration

The second part of this dissertation describes the top-down design methodology for the design space exploration. The proposed architecture is explored in terms of some system parameters including hardware parallelism, data exchange granularity, and design combination. Moreover, the exploration is performed at a higher level of abstraction using transaction level modeling (TLM) and it can significantly increase the simulation speed. Therefore, the proposed system architecture has a higher degree of freedom such that it can be optimized for the different design constraints. Specifically, the contributions of the design space exploration include the following:

- The design alternative that is optimized for cost can reduce the gate count of previous design up to 25%.

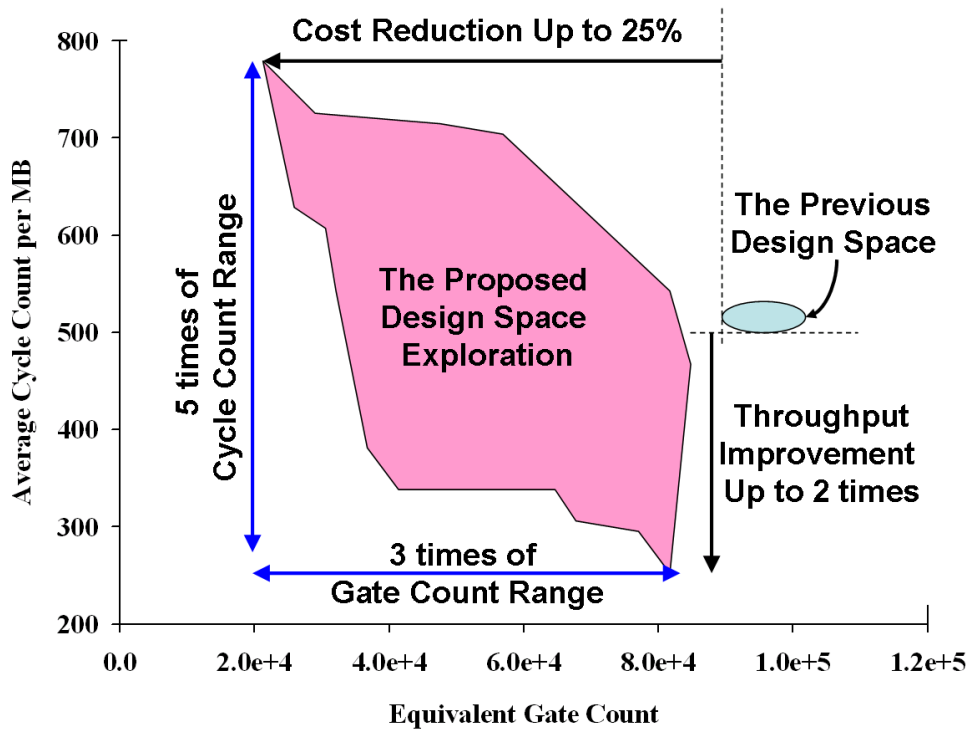


Figure 6.2: The Improvement of Throughput and Hardware Cost by the Proposed Effective Design Space Exploration

- The design alternative that is optimized for speed can reduce the cycle count of previous design up to 25% such that the proposed system architecture can fulfill the real-time requirement for 1920x1088 @ 60 Hz videos when clocking at 135MHz.
- This work has explored broad design space that includes 3 times of cycle count range and 5 times of gate count range as shown in Figure 6.2.

6.1.3 Improvement of Area-Speed Efficiency

The last part of this dissertation focuses on the micro-architecture design for the three hot spot modules within our system architecture. Specifically, the contributions at architecture level include the following:

- As compared to the conventional adder-tree based prediction module, the proposed systolic-

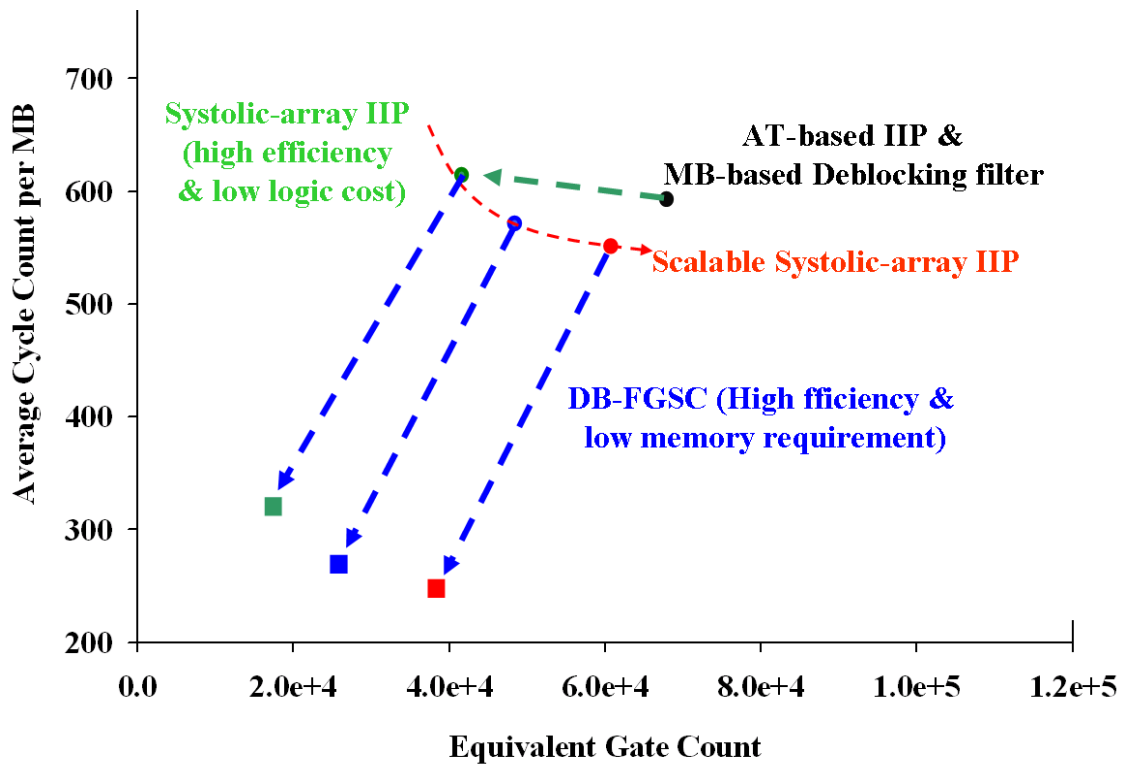


Figure 6.3: The Improvement of Area-Speed Efficiency by the Proposed High-Efficient Architecture

based design can increase the transcoding throughput up to four times while the amount of data transmission via AHB data bus can be reduced by 65%.

- To increase the efficiency of deblocking filter at a finer synchronization granularity, the proposed deblocking filter with fine-grained synchronization capability not only increases the cycle count but also reduces the memory cost as compared to the macroblock-based deblocking filter design.
- Lastly, Figure 6.3 shows the improvement of area-speed efficiency by each micro-architecture design.

6.2 Suggestions for Future Works

In this dissertation, the design exploration of an H.264/AVC VET transcoder is performed at both algorithm level and architecture level. In the future, the algorithm can be further developed to support other advanced coding tools in the H.264/AVC like bi-direction prediction, picture adaptive frame/field coding (PAFF), macroblock adaptive frame/field coding (MBAFF), flexible macroblock order (FMO), and so on.

In addition, the architecture design in this work follows the refinement-based methodology. Therefore, more performance evaluation metrics can be added to the design space exploration under the same simulation infrastructure. On the other hand, the architecture can be further refined to a lower level of abstraction, that is, the modules in the proposed architecture could be implemented as the RTL models, combined into the system platform, and verified for the functionality.

Eventually, the design flow of whole system will step forward to the RTL simulation, the RTL synthesis, the gate level simulation, the P&R (place and route), the post layout simulation, and the tape out.

Bibliography



- [1] *WinFast Series product of LeadTek*. [Online]. Available: <http://www.leadtek.com>
- [2] *YouTube Website*. [Online]. Available: <http://www.youtube.com>
- [3] *Multiple point videoconferencing system of Wiredred*. [Online]. Available: <http://www.wiredred.com/video-conferencing/multipoint-video-conference.html>
- [4] *"Walt disney studios home entertainment rolls into 2008 with blu-ray"*. [Online]. Available: "<http://corporate.disney.go.com>"
- [5] *CBBC TV programs*. [Online]. Available: <http://www.bbc.co.uk/cbbc/>
- [6] *Advanced video coding for generic audiovisual services*, 4th ed., ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG4-AVC), July 2005.
- [7] S. Wenger, "H.264/AVC over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.

- [8] M. D. Nava and C. Del-Toso, "A short overview of the VDSL system requirements," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 82–90, 2002.
- [9] S. Naimpally, L. Johnson, T. Darby, R. Meyer, L. Phillips, and J. Vantrease, "Integrated digital IDTV receiver with features," *IEEE Transactions on Consumer Electronics*, vol. 34, no. 3, pp. 410–419, 1988.
- [10] D. Gillies, R. Schweer, and H. Zibold, "VSLI realizations for picture in picture and flicker free television display," *IEEE Transactions on Consumer Electronics*, vol. 34, no. 1, pp. 253–261, 1988.
- [11] M. Burkert, F. Frieling, U. Langenkamp, U. Libal, M. Mende, and G. Scheffler, "IC set for a picture-in-picture system with on-chip memory," *IEEE Transactions on Consumer Electronics*, vol. 36, no. 1, pp. 23–31, 1990.
- [12] C. A. Mancini and C. Markhauser, "Microprocessor controlled picture in picture system," *IEEE Transactions on Consumer Electronics*, vol. 36, no. 3, pp. 375–379, 1990.
- [13] I. Ahmad, X. Wei, Y. Sun, and Y. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, 2005.
- [14] S.-F. Chang and D. G. Messerschmitt, "Compositing motion-compensated video within the network," in *Proceedings of IEEE 4th Workshop on Multimedia Communications*, Monterey, CA, USA, April 1992, pp. 40–56.
- [15] —, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, 1995.

- [16] D. G. Messerschmitt, Y. Noguchi, and S.-F. Chang, "MPEG video compositing in the compressed domain," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Atlanta, Ga, USA, May 1996, pp. 596–599.
- [17] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT domain inverse motion compensation," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, Ga, USA, May 1996, pp. 2307–2310.
- [18] J. Song and B.-L. Yeo, "A fast algorithm for DCT-domain inverse motion compensation based on shared information in a macroblock," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 767–775, 2000.
- [19] S. Liu and A. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, 2002.
- [20] B. Yu and K. Nahrstedt, "Internet-based Interactive HDTV," *Multimedia Systems*, vol. 9, no. 5, pp. 477–489, 2004.
- [21] Y.-P. Tan and H. Sun, "Fast motion re-estimation for arbitrary downsizing video transcoding using H.264/AVC standard," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 3, pp. 887–894, 2004.
- [22] J. Bialkowski, M. Barkouwsy, F. Leschka, and A. Kaup, "Low-Complexity transcoding of inter coded video frames from H.264 to H.263," in *Proceedings of IEEE International Conference on Image Processing*, Atlanta, Ga, USA, October 2006, pp. 837–840.

- [23] J. Hur and Y. Lee, "H.264 to MPEG-4 transcoding using block type information," in *Proceedings of IEEE International Conference Region 10*, Melbourne, Australia, November 2005, pp. 1–6.
- [24] D. Lefol, D. Bull, and N. Canagarajah, "Performance evaluation of transcoding algorithms for H.264," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 215–222, 2006.
- [25] H. Shen, X. Sun, F. Wu, H. Li, and S. Li, "A fast downsizing video transcoder for H.264/AVC with rate-distortion optimal mode decision," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Toronto, Ontario, Canada, July 2006, pp. 2017–2020.
- [26] P. Zhang, Y. Lu, Q. Huang, and W. Gao, "Mode mapping method for H.264/AVC spatial downscaling transcoding," in *Proceedings of IEEE International Conference on Image Processing*, Singapore, October 2004, pp. 2781–2784.
- [27] I.-H. Shin, Y.-L. Lee, and H.-W. Park, "Motion estimation for frame-rate reduction in H.264 transcoding," in *Proceedings of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, Vienna, Austria, May 2004, pp. 63–67.
- [28] D. Lefol and D. Bull, "Mode refinement algorithm for H.264 inter frame requantization," in *Proceedings of IEEE International Conference on Image Processing*, Atlanta, Ga, USA, October 2006, pp. 845–848.
- [29] J. Zhang, A. Perkis, and N. Georganas, "H.264/AVC and transcoding for multimedia adaptation," in *Proceedings of the 6th COST 276 Workshop*, Thessaloniki, Greece, May 2004.

- [30] X. Xui, L. Zhuo, and L. Shen, "A H.264 bit rate transcoding scheme based on PID controller," in *Proceedings of IEEE International Symposium on Communications and Information Technologies*, vol. 2, Beijing, China, October 2005, pp. 1074–1077.
- [31] C.-H. Li, C.-N. Wang, and T. Chiang, "A fast downsizing video transcoding based on H.264/AVC standard," in *Proceedings of IEEE Pacific Rim Conference on Multimedia*, Tokyo, Japan, November–December 2004, pp. 214–223.
- [32] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang, "A fast H.264-based picture-in-picture (PIP) transcoder," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, June 2004, pp. 1691–1694.
- [33] A. Levi and H. Stark, "Restoration from phase and magnitude by generalized projections," in *Image Recovery Theory and Application*. Orlando, Fla, USA: Academic Press, 1987, pp. 277–319.
- [34] S.-H. Wang, W.-H. Peng, Y. He, G.-Y. Lin, C.-Y. Lin, S.-C. Chang, C.-N. Wang, and T. Chiang, "A software-hardware co-implementation of MPEG-4 advanced video coding (AVC) decoder with block level pipelining," *Journal of VLSI Signal Processing*, vol. 41, no. 1, pp. 93–110, 2005.
- [35] K. Keutzer, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: orthogonalization of concerns and platform-based design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1523–1543, 2000.
- [36] H. Chang, *Surviving the SoC Revolution: A Guide to Platform-Based Design*. Kluwer Academic Publishers, 1999.

- [37] S. Huang, C. Chen, and H. Chen, "Hardware architecture design of video compression for multimedia communication systems," *Communications Magazine, IEEE*, vol. 43, no. 8, pp. 122–131, 2005.
- [38] H. Hubert, B. Stabernack, and K.-I. Wels, "Performance and Memory Profiling for Embedded System Design," in *Proceedings of the 2nd IEEE International Symposium on Industrial Embedded Systems (SIES07)*, Lisbon, Portugal, July 2007, pp. 94–101.
- [39] B. Sheshadri, "Optimization Of H. 264 Baseline Decoder On ARM9TDMI Processor," 2005.
- [40] S. H. Wang and T. C. et al., "A platform-based MPEG-4 advanced video coding (AVC) decoder with block level pipeling," vol. 1, Singapore, December 2003, pp. 51–55.
- [41] M. Ravasi and M. Mattavelli, "High-abstraction level complexity analysis and memory architecture simulations of multimedia algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 673–684, 2005.
- [42] M. Horowitz, A. J. F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, 2003.
- [43] V. Lappalainen, A. Hallapuro, and T. Hamalainen, "Complexity of H.26L optimized video decoder implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 717–725, 2003.
- [44] Y. W. Huang, B. Y. Shieh, T. C. Chen, and L. G. Chen, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 13, pp. 378–401, March 2005.

- [45] H. Lin, Y. Chiao, B. Liu, and J. Yang, "Combined 2-D Transform and Quantization Architectures for H.264 Video Coders," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 1802–1805.
- [46] *The datasheet of Micron's 256Mb Mobile DDR SDRAM*. [Online]. Available: <http://download.micron.com/pdf/datasheets/dram/mobile/MT46H16M16LF.pdf>
- [47] S. Wang, T. Lin, T. Liu, and C. Lee, "A new motion compensation design for H.264/AVC decoder," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 5, Kobe, Japan, June 2005, pp. 4558–4561.
- [48] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-H. Chang, and L. Chen, "Architecture design for deblocking filter in H.264/JVT/AVC," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Baltimore, Maryland, USA, July 2003, pp. 693–696.
- [49] B. Sheng, W. Gao, and D. Wu, "An implemented architecture of deblocking filter for H.264/AVC," in *Proceedings of IEEE International Conference on Image Processing*, Singapore, October 2004, pp. 665–668.
- [50] C.-C. Chen and T.-S. Chang, "An hardware efficient deblocking filter for H.264/AVC," in *Proceedings of IEEE International Conference on Consumer Electronics*, Las Vegas, Nevada, USA, January 2005, pp. 235–236.
- [51] L. Li, S. Goto, and T. Ikenaga, "An efficient deblocking filter architecture with 2-dimensional parallel memory for H.264/AVC," in *Proceedings of the 10th IEEE Asia and South Pacific Design Automation Conference*, Shanghai, China, January 2005, pp. 623–626.

- [52] T.-M. Liu, W.-P. Lee, T.-A. Lin, and C. Lee, "A memory-efficient deblocking filter for H.264/AVC video coding," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 2140–2143.
- [53] S. Shih, C. Chang, and Y. Lin, "An AMBA-compliant deblocking filter IP for H.264 AVC," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 4529–4532.
- [54] S.-Y. Shih, C.-R. Chang, and Y.-L. Lin, "A near optimal deblocking filter for H.264 advanced video coding," in *Proceedings of the 11th IEEE Asia and South Pacific Design Automation Conference*, Yokohama, Japan, January 2006, pp. 170–175.
- [55] G. Khurana and A. Kassim, "A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 536–540, 2006.
- [56] C.-C. Chen, T.-S. Chang, and K.-B. Lee, "An in-place architecture for the deblocking filter in h.264avc," *IEEE Transactions on Circuits and Systems*, vol. 53, no. 7, pp. 530–534, 2006.
- [57] H.-Y. Lin, J.-J. Yang, B.-D. Liu, and J.-F. Yang, "Efficient deblocking filter architecture for H.264 video coders," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Island of Kos, Greece, May 2006, pp. 2617–2620.
- [58] M. Bojnordi, O. Fatemi, and M. Hashemi, "An efficient deblocking filter with self-transposing memory architecture for H.264/AVC," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, May 2006, pp. II925–II928.

- [59] S. Lee and K. Cho, "An efficient architecture of high-performance deblocking filter for H.264/AVC," *IEEE Transactions on Fundamentals*, vol. E89, no. 6, pp. 1736–1739, 2006.
- [60] *CoWare's ConvergenSCTM*. [Online]. Available: <http://www.coware.com>
- [61] *Open SystemC Initiative, SystemC Version 2.0 Users Guide*. [Online]. Available: www.systemc.org
- [62] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1999.
- [63] H. Kim and I. C. Park, "High-performance and low-power memory-interface architecture for video processing applications," vol. 11, no. 11, pp. 1160–1170, 2001.
- [64] J. Zhu, L. Hou, R. Wang, C. Huang, and J. Li, "High performance synchronous DRAMs controller in H.264 HDTV decoder," in *Proceedings of IEEE International Conference Solid-State and Integrated Circuit Technology*, vol. 3, Beijing, China, October 2004, pp. 1621–1624.
- [65] R. Wang, J. Li, and C. Huang, "Motion Compensation Memory Access Optimization Strategies for H.264/AVC Decoder," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, USA, March 2005, pp. 97–100.
- [66] S. I. Park, Y. Yongseok, and I. C. Park, "High performance memory mode control for HDTV decoders," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 14, pp. 1348–1353, 2003.
- [67] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, 1999.

- [68] J. Joch, F. Kossentini, and P. Nasiopoulos, "A performance analysis of the ITU-T draft H.26L video coding standard," in *Proceedings of the 12th International Packet Video Workshop*, Pittsburgh, Pa, USA, April 2002.
- [69] H. Kang, K. Jeong, J. Bae, Y. Lee, and S. Lee, "MPEG4 AVC/H. 264 decoder with scalable bus architecture and dual memory controller," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Vancouver, Canada, May 2004, pp. 145–148.
- [70] T. Chen, Y. Huang, T. Chen, Y. Chen, C. Tsai, and L. Chen, "Architecture design of H.264/AVC decoder with hybrid task pipeling for high definition videos," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 2931–2934.
- [71] T.-A. Lin, S.-Z. Wang, T.-M. Liu, and C. Lee, "An H.264/AVC decoder with 4x4-block level pipeline," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 1810–1813.
- [72] T.-M. Liu, T.-A. Lin, S.-Z. Wang, W.-P. Lee, K.-C. Hou, J.-Y. Yang, and C.-Y. Lee, "An 865 uW H.264/AVC video decoder for mobile applications," in *Proceedings of IEEE Asia Solid-State Circuit Conference*, HsinChu, Taiwan, November 2005, pp. 301–304.
- [73] S. Park, H. Cho, H. Jung, and D. Lee, "An implemented of H.264 video decoder using hardware and software," in *Proceedings of IEEE Custom Integrated Circuits Conference*, San Jose, Calif, USA, September 2005, pp. 271–275.
- [74] T.-M. Liu, T.-A. Lin, S.-Z. Wang, W.-P. Lee, K.-C. Hou, J.-Y. Yang, and C.-Y. Lee, "A 125 uW, fully scalable MPEG-2 and H.264/AVC video decoder for mobile applications," *IEEE Journal on Solid-State Circuits*, vol. 42, no. 1, pp. 161–169, 2007.

- [75] K. Yang, C. Zhang, G. Du, J. Xie, and Z. Wang, "A hardware-software co-design for H.264/AVC decoder," in *Proceedings of IEEE Asia Solid-State Circuit Conference*, Hangzhou, China, November 2006, pp. 119–122.
- [76] Y. Lei, H. Li, K. Huang, Y. Leng, and Z. Zheng, "A H.264 video decoder with scheme of efficient bandwidth optimization for motion compensation," in *Proceedings of IEEE International Symposium on Communications and Information Technologies*, Sydney, Australia, October 2007, pp. 531–534.
- [77] Y. W. Huang, B. Y. Shieh, T. C. Chen, and L. G. Chen, "Hardware architecture design for H.264/AVC intra frame coder," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, Vancouver, BC, Canada, May 2004, pp. 269–272.
- [78] L. Deng, W. Gao, M. Hu, and Z. Ji, "An efficient VLSI architecture for MC interpolation in AVC video coding," in *Proceedings of International MultiConference in Computer Science and Computer Engineering*, Las Vegas, Nevada, USA, June 2004, pp. 564–568.
- [79] T. C. Chen, Y. Huang, and L. Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, Canada, May 2004, pp. 9–12.
- [80] C. Chen, P. Wu, and H. Chen, "Transform-domain intra prediction for H.264," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, Kobe, Japan, May 2005, pp. 1497–1500.

APPENDIX A

Why Transform Domain Approaches are Inefficient for H.264 Transcoding



In this appendix, we will show that it is inefficient to develop a transcoder in the transform domain as commonly proposed for previous standards such as MPEG-1/2/4. There are several reasons to support such a claim.

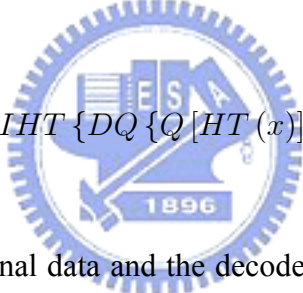
1. In H.264/AVC, the transformation and quantization processes are so optimized that traverse back to the pixel domain is not as expensive as before.
2. The intra prediction and de-blocking filter introduce stronger spatial domain error propagation although they are effective to exploit the spatial redundancy.

3. The IMC becomes inefficient when the motion compensation uses quarter-pixel resolution combined with 6-tap interpolation.

The following text will describe the detail study to support such a claim.

A.1 Integer Transform with Quantization Scaling

The transformation used in H.264/AVC is an integrated transform with quantization scaling, which means the scaling multiplication is merged with the quantization. The integer transform with quantization scaling is performed with simple integer operations such as shifting and addition, which indicates no rounding mismatch between the encoder and the decoder. The relationship between the pixel values at the encoder and the decoder can be represented by

$$IHT \{DQ \{Q [HT(x)]\}\} = \bar{x} \quad (\text{A.1})$$


, where x and \bar{x} mean the original data and the decoded data respectively. However, the data after de-quantization is not the original HT coefficients:

$$DQ \{Q [HT(x)]\} \neq HT(x) \quad (\text{A.2})$$

In order to obtain the transform coefficients at the transcoder side, an inverse operation of quantization is needed. The inverse quantization of HT coefficients is derived as follows. The following shows the quantization of HT coefficients $Y_{i,j}$.

$$Z_{i,j} = (Y_{i,j} \times MF_{i,j} + f) \gg S_1 \quad (\text{A.3})$$

, where

$$S_1 = 15 + \left\lfloor \frac{QP}{6} \right\rfloor, f = \frac{2^{S_1}}{3} \text{ or } \frac{2^{S_1}}{6}$$

The following shows that the data after the de-quantization is different from the original HT coefficients.

$$W_{i,j} = (Z_{i,j} \times V_{i,j}) \ll S_2 = \{[(Y_{i,j} \times MF_{i,j} + f) \gg S_1] \times V_{i,j}\} \ll S_2 \neq Y_{i,j} \quad (\text{A.4})$$

, where

$$S_2 = \left\lfloor \frac{QP}{6} \right\rfloor$$

The symbols $MF_{i,j}$ and $V_{i,j}$ are multiplication and rescaling factor respectively as defined in H.264/AVC standard. To obtain the HT coefficients, the de-quantization process should be replaced by converting quantized coefficients to de-quantized HT coefficients. The process is computed as

$$Y'_{i,j} = \frac{(Y_{i,j} \ll S_1)}{MF_{i,j}} \quad (\text{A.5})$$

However, Eq.(A.5) requires a division operation with higher complexity and additional rounding error.

A.2 Directional Intra Prediction

The intra prediction as defined in the spatial domain poses challenges to the transform domain transcoding. To implement intra prediction in the transform domain significantly increases complexity because the HT transform is not orthogonal, which means the transpose is not equal

A.3 In-the-loop De-blocking Filtering

The de-blocking filter defined in the spatial domain introduces mismatch error during HT domain transcoding. Particularly, the de-blocked pixels stored in the reference frame memory are used for motion compensation. Thus, mismatch error will propagate to the next frames via motion compensation until the subsequent intra-coded frame or slices at the decoder. To prevent the mismatch error, implementing the de-blocking filter in the HT domain is required. However, this kind of implementation increases the complexity and memory requirement.

A.4 Sub-pixel Interpolation

The complexity of HT domain IMC increases due to the 6-tap interpolator defined in H.264/AVC. Detailed derivations are given in the following. A 4×4 motion-compensated block can be represented as the summation of four blocks in the spatial domain.

$$B_{pred(4 \times 4)_{full_pel}} = \sum_{k=1}^4 V_{k(4 \times 4)} \cdot B_k \cdot H_{k(4 \times 4)} \quad (A.7)$$

, where

$$\begin{aligned} V_{1(4 \times 4)} &= V_{2(4 \times 4)} = \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}, & V_{3(4 \times 4)} &= V_{4(4 \times 4)} = \begin{bmatrix} 0 & 0 \\ I_{4-h} & 0 \end{bmatrix}, \\ H_{1(4 \times 4)} &= H_{3(4 \times 4)} = \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix}, & H_{2(4 \times 4)} &= H_{4(4 \times 4)} = \begin{bmatrix} 0 & I_{4-w} \\ 0 & 0 \end{bmatrix} \end{aligned}$$

We start the discussion of IMC from a block with integer motion vector. The HT coefficients of

prediction block can be calculated from four HT blocks as indicated by

$$\begin{aligned}
 & HT(B_{pred(4 \times 4)_{full_pel}}) \\
 &= HT\left(\sum_{k=1}^4 V_{k(4 \times 4)} \cdot B_k \cdot H_{k(4 \times 4)}\right) = \sum_{k=1}^4 HT(V_{k(4 \times 4)} \cdot B_k \cdot H_{k(4 \times 4)}) \\
 &= \sum_{k=1}^4 (C_f \cdot V_{k(4 \times 4)} \cdot C_f^{-1}) \cdot (C_f \cdot B_k \cdot C_f^{-1}) \cdot (C_f \cdot H_{k(4 \times 4)} \cdot C_f^T) \cdot (C_f \cdot V_{k(4 \times 4)} \cdot C_f^{-1}) \\
 &= \sum_{k=1}^4 HT(B_k) \begin{bmatrix} c & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & a \end{bmatrix} (C_f \cdot H_{k(4 \times 4)} \cdot C_f^T) \quad (A.8)
 \end{aligned}$$

The terms of $(C_f \cdot V_{k(4 \times 4)} \cdot C_f^{-1})$ and $(C_f \cdot H_{k(4 \times 4)} \cdot C_f^T)$ can be pre-computed and stored in memory. The computation of Eq.(A.8) needs 576 multiplications and 384 additions.

The sub-pixel interpolation filter increases the complexity of transform domain IMC. The half-pixel sample is interpolated from integral pixel samples by applying a 6-tap Finite Impulse Response (FIR) filter, whose weights are $(1, -20, 20, 5/8, -5, 1)/32$. The HT coefficients of a prediction block on the half-pixel position have to be calculated from nine blocks as indicated in the following equation:

$$\begin{aligned}
 & HT(B_{pred(4 \times 4)_{sub_pel}}) \\
 &= \sum_{k=1}^4 \left\{ HT(B_k) \begin{bmatrix} c & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & a \end{bmatrix} \cdot \left(C_f \cdot V_{avg(4 \times 9)} \cdot V'_{k(9 \times 4)} \cdot C_f^{-1} \right) \cdot \left(C_f \cdot H'_{k(4 \times 9)} \cdot H_{avg(9 \times 4)} \cdot C_f^T \right) \right\} \quad (A.9)
 \end{aligned}$$

, where

$$\begin{aligned}
 V'_{1(9 \times 4)} = V'_{2(9 \times 4)} = V'_{3(9 \times 4)} &= \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}, H'_{1(4 \times 9)} = H'_{2(4 \times 9)} = H'_{3(4 \times 9)} = \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix}, \\
 V'_{4(9 \times 4)} = V'_{5(9 \times 4)} = V'_{6(9 \times 4)} &= \begin{bmatrix} 0 \\ I_4 \\ 0 \end{bmatrix}, H'_{4(4 \times 9)} = H'_{5(4 \times 9)} = H'_{6(4 \times 9)} = \begin{bmatrix} 0 & I_4 & 0 \end{bmatrix}, \\
 V'_{7(9 \times 4)} = V'_{8(9 \times 4)} = V'_{9(9 \times 4)} &= \begin{bmatrix} 0 & 0 \\ I_{5-h} & 0 \end{bmatrix}, H'_{7(4 \times 9)} = H'_{8(4 \times 9)} = H'_{9(4 \times 9)} = \begin{bmatrix} 0 & I_{5-w} \\ 0 & 0 \end{bmatrix}, \\
 V_{avg(4 \times 9)} &= \frac{1}{32} \begin{bmatrix} 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 \\ 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 \\ 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 \\ 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 \end{bmatrix}, \\
 H_{avg(9 \times 4)} &\text{ is the transpose of } V_{avg(4 \times 9)}
 \end{aligned}$$

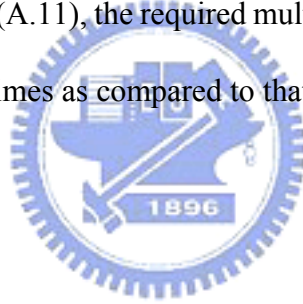
The computation of Eq.(A.9) requires 1296 multiplications and 864 additions. Assume that the frame size is M by N and the probability of full-pixel MV is α , the total amount of computation for the HT domain IMC involves with

$$\begin{aligned}
 576 \times \frac{M \cdot N}{4} \times \alpha + 1296 \times \frac{M \cdot N}{4} \times (1 - \alpha) &= \frac{M \cdot N}{4} \cdot (1296 - 720 \cdot \alpha) \\
 &\text{multiplications and} \\
 384 \times \frac{M \cdot N}{4} \times \alpha + 864 \times \frac{M \cdot N}{4} \times (1 - \alpha) &= \frac{M \cdot N}{4} \cdot (864 - 480 \cdot \alpha) \\
 &\text{additions} \tag{A.10}
 \end{aligned}$$

As for the spatial domain IMC, the total amount of computation covers

$$\begin{aligned}
 & 4 \cdot [M \cdot (N - 1) + N \cdot (M - 1) + (N - 1) \cdot (M - 1)] \\
 = & 12M \cdot N - 8(M + N) + 4 \quad \text{multiplications and} \\
 & 5 \cdot [M \cdot (N - 1) + N \cdot (M - 1) + (N - 1) \cdot (M - 1)] + 64 \cdot \frac{M \cdot N}{4} \cdot 2 \\
 = & 47M \cdot N - 10(M + N) + 5 \quad \text{additions} \quad \quad \quad (\text{A.11})
 \end{aligned}$$

On the average, the SD resolution bitstream has 25% of motion vectors pointing to full-pixel locations and 75% of motion vectors pointing to half-pixel locations. Therefore, the complexity increase by sub-pixel interpolation in the HT domain is not preferable for transcoding. From the results of Eq.(A.10) and Eq.(A.11), the required multiplications and additions of HT domain IMC are about 23 times and 3 times as compared to that of spatial domain IMC respectively.



APPENDIX B

Constant-Rate Bumping Process



The bumping process in H.264/AVC manages the output of decoded pictures such that the output order conforms to the display order. Different from the prior video coding standards in which the decoder can output the decoded pictures according to the timestamp information in the picture or slice header, the NAL units of H.264/AVC do not contain any timing information. Thus, the bumping process defines the rules for ensuring correct output of decoded pictures. It should be noted that the output of a decoded picture does not necessarily imply the removal of the picture from the DPB. For a reference picture, it may still be kept in the DPB for temporal prediction while it is output for display.

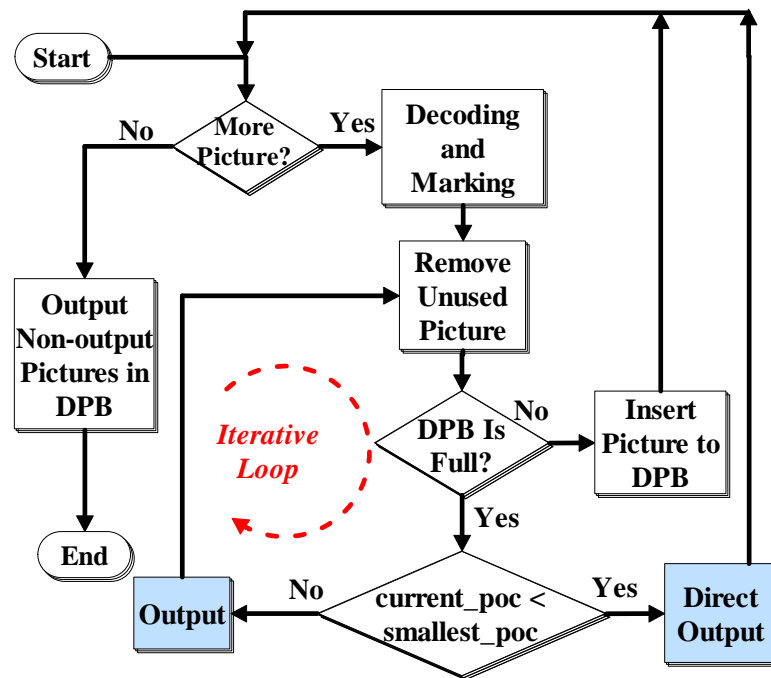


Figure B.1: The flow chart of the bumping process in H.264/AVC. The “smallest_poc” stands for the smallest picture order count (POC) among the non-output pictures in the DPB.

B.1 Bumping Process in H.264/AVC

Figure B.1 shows the flow chart of the current bumping process in H.264/AVC. As shown, after the decoding of a picture, those in the DPB that are marked as unused for reference and have been output for display will be removed from the buffer. Then, the bumping process is enabled by checking whether the DPB has more space for the current picture. In case that the DPB is still fully occupied, the non-output picture in the DPB that has the smallest picture order count (POC) will be firstly considered for output unless the current picture has an even smaller POC and needs to be output directly. After the output of a decoded picture in the DPB, it will be further removed if it is no longer used for reference. However, it could happen that the picture just output for display still needs to be kept for reference. Thus, such a process must be repeated until space is found or the current picture is output.

Due to the iterative bumping behavior in the current algorithm, multiple decoded pictures

could be output simultaneously while for some time instances none of the pictures are output. The irregular output rate of the bumping process causes extremely varying bandwidth during the DRAM access. The number of frame buffer in DRAM is double to regulate the burst of output. Thus, the maximum number of output pictures at a time is equal to DPB size plus one. In addition to the space of DPB, the regulation buffer is used for buffering the direct outputs that is not stored in the DPB. With a constant-rate bumping process, we can output one picture at a time without violating the output order conformance.

B.2 Proposed Constant-Rate Bumping Process

To provide a constant output rate and minimize the bandwidth variation for the DRAM access, an output regulation buffer is created in addition to the DPB. In particular, the bumping process in H.264/AVC may output all pictures in the DPB and the current decoding picture. Thus, a buffer with size being equal to the one of the DPB is allocated to regulate the burst of output. Figure B.2 shows the proposed constant-rate bumping algorithm using the regulation buffer. As highlighted by the gray blocks and the dotted lines, the major difference in contrast to the variable-rate bumping process is that there will be only 1 picture output for display after the decoding of a picture. As a result, when the DPB is still fully occupied after the output of a picture, the current picture will be stored in the regulation buffer. On the other hand, when space is available, the DPB keeps the current picture and the regulation buffer may output 1 picture according to the constraint on the output rate.

For better understanding, Figure B.3 uses an example to compare the two bumping processes, where the numbers in the parentheses indicate the display order, the number on the top of each picture denotes the coding order, and the arrowheads reveal the prediction structure with those pictures of gray color used for reference. According to the flow in Fig. 7, the variable-rate

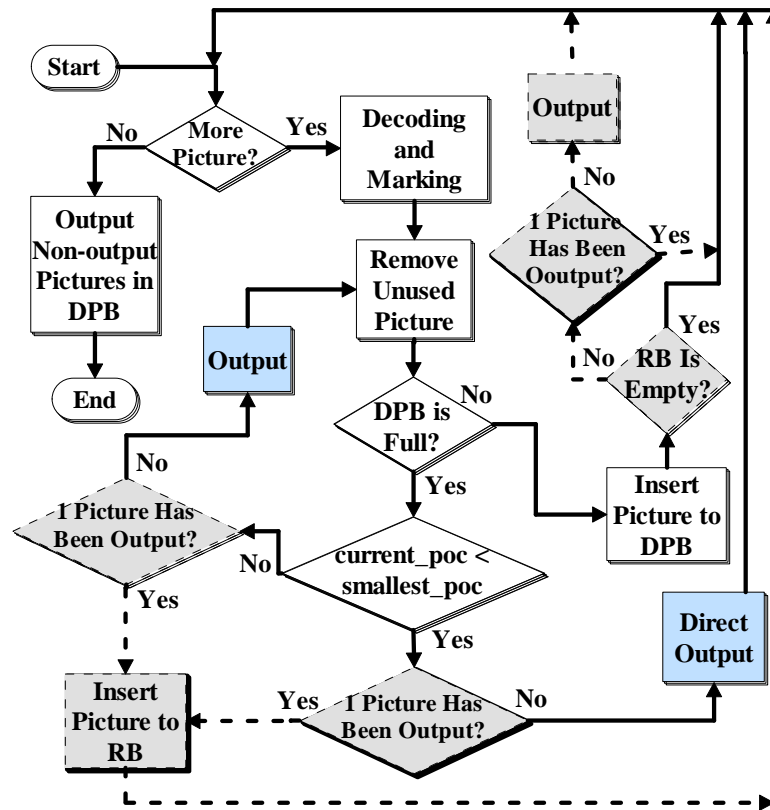


Figure B.2: The flow chart of the proposed constant-rate bumping process. The “smallest_poc” stands for the smallest picture order count (POC) among the non-output pictures in the DPB and the “RB” denotes the regulation buffer.

bumping process outputs 2 to 3 pictures when decoding the non-reference B pictures such as pictures 2, 6, and 10. Conversely, the proposed bumping process provides constant output rate while maintaining the conformance of output order.

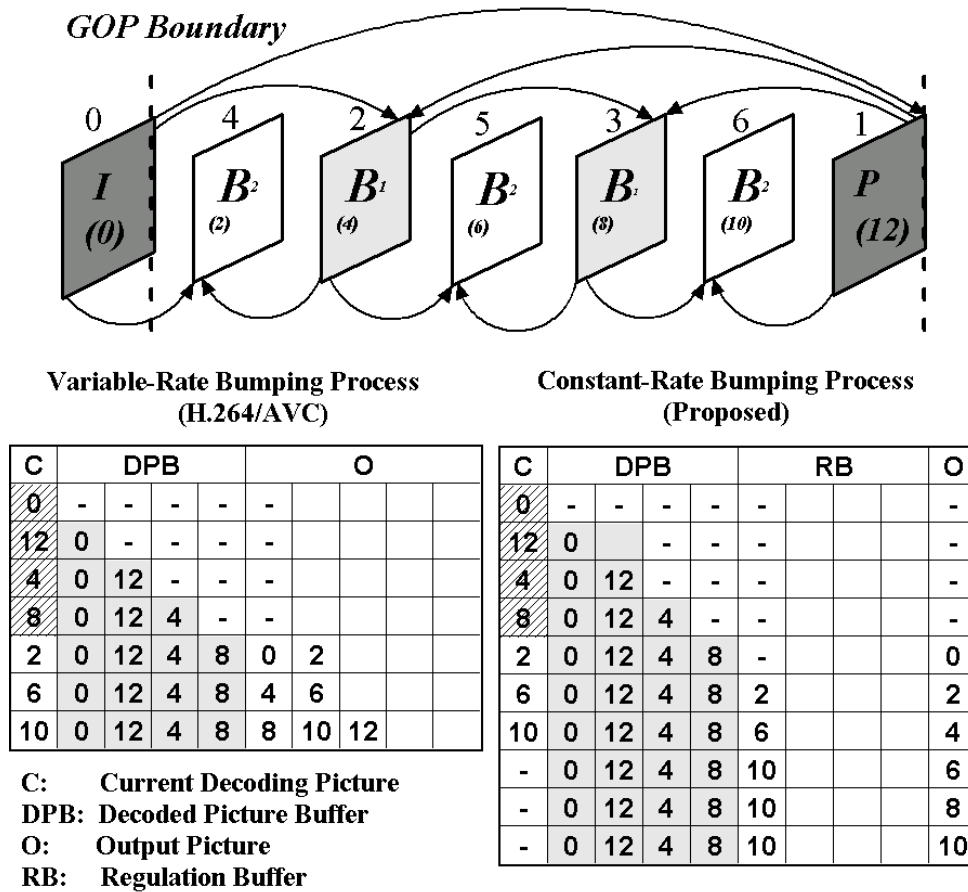


Figure B.3: Example for Comparing the Variable-Rate Bumping Process and the Proposed Constant-Rate Bumping Process

CHIH-HUNG LI

PERSONAL INFORMATION

- Ph.D Candidate, Institute of Electronics, National Chiao-Tung University
- 1001 Ta-Hsueh Road, HsinChu 30010, Taiwan.
- Office: ED422
- Tel: +886-(0)922-664-568
- E-mail: ChihHung.Li@gmail.com

EDUCATION

- **Ph.D Candidate**, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan, **Sept. 2002 - Present**.
- **M.S. Student**, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan, **Sept. 2001 - Aug. 2002**.
- **B.S. Degree**, Dept. of Electronics Engineering, National Chiao-Tung University, HsinChu, Taiwan, **June, 2001**.

RESEARCH INTEREST

- Video/Image Compression
 - MPEG1, 2, 4, 21 and H.264
 - Rate control.
 - Human visual system (Subjective quality evaluation).
 - Video transcoding.
 - Algorithm development of video coding.
 - Codec optimization.
- Embedded System Design for Video/Image Compression
 - Platform-based System-on-Chip design.
 - System architecture modeling and verification.
 - Software/hardware partition and co-design.
 - DRAM controller.
 - Dedicated hardware accelerators.
- Video/Image Streaming System

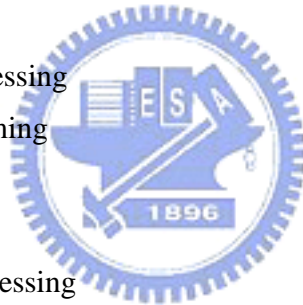
- MPEG-21 Part-12 Test-Bed streaming system.
- Rate control
- Universal multimedia access

PROGRAMMING SKILL

- C/C++
- SystemC
- Matlab
- Verilog, VHDL, Spice
- ARM Code
- XML, Tcl

SPECIALIZE COURSE

- (Grade) (Course)
- (90) Random Process
- (87) Video Signal Processing
- (90) Network programming
- (92) Channel Coding
- (92) Source Coding
- (92) Digital Image Processing
- (88) Digital Integrated Circuit
- (94) Digital Communication
- (89) Advance Discrete Signal Process
- (90) CAD



WORK EXPERIENCE

- **Involved Project**
 - July. 2006 - June. 2008, NSC SOC project, “遍佈即時多媒體系統與技術-子計畫二:可調視訊編碼之適應性動態精細改進和編碼”
 - July. 2005 - June. 2006, NSC SOC project “MPEG-21 多媒體通用存取架構下數位元件可適性之即時視訊轉碼系統之設計與製作”NSC 94-2220-E-009-012
 - July. 2005 - June. 2006, NSC SOC project “高等精細可調層次式視頻編碼技術之研究” NSC 93-2220-E-009-012

- Oct. 2003 - Sept. 2006, 學界科專”MPEG-7/21 多媒體特徵擷取，意涵理解及資訊探索技術子計畫－MPEG-7/21 標準之規範理解與追蹤”
- Sept. 2002 - Aug. 2003, MicroSoft Asia Research, “Advanced PIP (Picture-in-Picture) Application for WMV9 (Window Media Video 9) based Video Streaming”

- **Teaching Assistant**

- Jan. 2007 - June. 2007, System Model Design and Verification.
- Jan. 2006 - June. 2006, Information Theory.
- Jan. 2004 - June. 2004, Programming Language.
- Jan. 2003 - June. 2003, Programming Language.
- Jan. 2002 - June. 2002, Programming Language.
- Sept. 2001 - Feb. 2002, Discrete Signal Process.

PUBLICATION

- **Journal paper**

1. **C.-H. Li**, C.-N. Wang, and T. Chiang, “A Multiple-Window Video Embedding Transcoder based on H.264/AVC Standard,” *EURASIP Journal on Advanced Signal Processing*, vol. 2007, Article ID 13790, 17 pages, 2007. doi:10.1155/2007/13790.
2. **C.-H. Li**, W.-H. Peng, and T. Chiang, “Design of Memory Sub-System with Constant-Rate Bumping Process for H.264/AVC Decoder,” *IEEE Transaction on Consumer Electronics*, vol. 53, no. 1, pp. 209–217. Feb. 2007.
3. **C.-H. Li**, S.-H. Wu, W.-H. Peng, and T. Chiang, “Design Space Exploration of A Highly Efficient H.264/AVC-based Video Embedding Transcoder Using Transaction Level Modeling,” (in preparation)

- **International Conference paper**

1. **C.-H. Li**, W.-H. Peng, and T. Chiang, “Design Space Exploration of An H.264/AVC-based Video Embedding Transcoder Using Transaction Level Modeling,” to be appeared in ICME 2008.
2. **C.-H. Li**, W.-H. Peng, and T. Chiang, “A Reconfigurable Video Embedding Transcoder Based on H.264/AVC: Design Tradeoffs and Analysis,” to be appeared in ISCAS 2008.
3. **C.-H. Li**, C.-H. Chang, W.-H. Peng, W. Hwang and T. Chiang, “Design of Memory Sub-System in H.264/AVC Decoder,” in *Proceedings of IEEE International Conference on Consumer Electronics (ICCE’07)*, pp. 1–2, Las

Vegas, USA, Jan. 2007.

4. **C.-H. Li**, C.-C. Chen, W.-C. Su, M.-J. Wang, W.-H. Peng, G.-G. Lee and T. Chiang, "A Unified Systolic Architecture for Combined Inter and Intra Predictions in H.264/AVC Decoder," in *ACM International Conference on Wireless Networks, Communications and Mobile Computing (IWCMC'06)*, pp. 73–78, Vancouver, July 2006.
5. **C.-H. Li**, C.-N. Wang, and T. Chiang, "A Low Complexity Picture-in-Picture Transcoder for Video-on-Demand, " in *Proceedings of IEEE International Conference on Wireless Networks, Communications and Mobile Computing (WirelessCom'05)*, vol.2, pp. 1382–1387, Maui, Hawaii, USA, June 2005.
6. **C.-H. Li**, C.-N. Wang, and T. Chiang, "A Fast Downsizing Video Transcoding Based on H.264/AVC Standard," in *Proceedings of IEEE Pacific Rim Conference on Multimedia (PCM'04)*, pp.215–223, Tokyo, Japan, Dec. 2004.
7. **C.-H. Li**, H. Lin, C.-N. Wang, and T. Chiang, "A Fast H.264-Based Picture-In-Picture (PIP) Transcoder," in *Proceedings of IEEE International Conference on Multimedia and Expo. (ICME'04)*, vol.3, pp. 1691–1694, Taipei, Taiwan, June, 2004.
8. **C.-H. Li**, C.-N. Wang, and T. Chiang, "A VBR Rate Control Using MINMAX Criterion for Video Streaming," in *Proceedings of IEEE Pacific Rim Conference on Multimedia. (PCM'02)*, vol.2532, pp. 831–838, HsinChu, Taiwan, Dec. 2002.

● **Contribution document of ISO/IEC MPEG**

1. **C.-H. Li** et al "ISO/IEC JTC1/SC 29/WG 11 M12373: "Update to the FGS-Based Multimedia Resource Delivery Test Bed Software, " July 2005. (73rd, Poznan, Poland)
2. **C.-H. Li** et al "ISO/IEC JTC1/SC 29/WG 11 M11117: "FGS-Based Video Streaming Test Bed for Media Coding and Testing in Streaming Environments, " July 2004. (69th, Redmond, Washington, USA)

● **Patent**

1. C.-C. Chen, **C.-H. Li**, W.-H. Peng, and T. Chiang, "Prediction Module," US patent, Filed on 2006.
2. 陳治傑, **李志鴻**, 彭文孝, 蔣迪豪, "預測模組," 中華民國, 專利公開號 200808067, February 2008.
3. 蘇子良, 王俊能, 蔣迪豪, **李志鴻**, "使用於 MPEG-4 之具有 R-D 最佳化畫面內更新的容錯編碼器," 中華民國, 專利公告號 I263418, October 2006.

● **Technical report**

1. Project report of NSC MPEG SOC NSC 95-2221-E-009-074-MY3：可調視訊編碼之適應性動態精細改進和編碼.
2. Project report of NSC MPEG SOC NSC 93-2220-E-009-012：高等精細可調層次式視頻編碼技術之研究 .
3. Project report of NSC MPEG SOC NSC 94-2220-E-009-012：MPEG-21 多媒體通用存取架構下數位元件可適性之即時視訊轉碼系統之設計與製作.
4. Project report of 學界科專：MPEG-7/21 多媒體特徵擷取，意涵理解及資訊探索技術三年計畫：MPEG-7/21 標準之規範理解與追蹤第三年度報告.
5. Project report of 學界科專：MPEG-7/21 多媒體特徵擷取，意涵理解及資訊探索技術三年計畫：MPEG-7/21 標準之規範理解與追蹤第二年度報告.
6. Project report of 學界科專：MPEG-7/21 多媒體特徵擷取，意涵理解及資訊探索技術三年計畫：MPEG-7/21 標準之規範理解與追蹤第一年度報告.
7. Project report of MicroSoft Research Asia (MSRA)：Advanced PIP (Picture-in-Picture) Application for WMV9 (Window Media Video 9) based Video Streaming.

● **Others**

1. 文/彭文孝、李志鴻、陳治傑、蔣迪豪,“MPEG-4 AVC/H.264 視訊壓縮編碼簡介”, 電子月刊 2006/八月號.
2. 文/李鴻儒 李志鴻、王俊能/譯,“新寬頻生活型態”, 電子月刊 2002/元月號/第 78 期.

AWARDS

- Scholarship of Ministry of Education (2003 & 2004)
- Novatek Fellowship (2007)

REFERENCE

● **Professor Tihao Chiang**

- Professor, Dept. of Electronics Engineering, National Chiao-Tung University
- 1001 Ta-Hsueh Rd., HsinChu 30010, Taiwan.
- Office: ED506
- Tel :+886-351-31558

- Fax:+886-357-31791
- E-mail: tchiang@mail.nctu.edu.tw

