

國立交通大學

電子工程學系電子研究所

博士論文

適用於功率受限視訊編碼系統之運動估測演算法

與積體電路架構設計

Algorithm and Architecture Design of Motion
Estimation for Power Constrained Video Coding
Systems

研究生：王士豪

指導教授：蔣迪豪

中華民國九十六年九月

適用於功率受限視訊編碼系統之運動估測演算法與積體

電路架構設計

Algorithm and Architecture Design of Motion Estimation
for Power Constrained Video Coding System

研究生：王士豪

Student : Shih-Hao Wang

指導教授：蔣迪豪博士

Advisor : Dr. Tihao Chiang

國立交通大學

電子工程學系電子研究所



Submitted to Department of Electronics Engineering & 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

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

推薦函

主旨：推薦電子工程學系博士班學生王士豪，舉行博士班學位口試。

說明：本人所指導之博士班學生王士豪，業已通過資格考試，並完成本校電子工程學系電子研究所博士班規定之學科課程及論文研究訓練。王君主要從事視訊編、解碼演算法與積體電路架構設計之研究工作，其論文「適用於功率受限視訊編碼系統之運動估測演算法與積體電路架構設計」(Algorithm and Architecture Design of Motion Estimation for Power Constrained Video Coding System) 主要包含兩項應用於功率受限編碼系統上的運動估測技術。其中第一項技術『低功率、低頻寬需求之二元化運動估測』(Low power and bandwidth efficient binary motion estimation)，利用二元化的影像作為運動估測準則，並提出其最佳化的硬體架構設計，此技術可達到低於 1 mW 的功率消耗需求，並節省 I/O 頻寬存取需求達 54.3% 以上。第二項技術為『具功率感知之功率可調適性之疊代二元化運動估測』(Power adaptive motion estimation using iterative binary searches)，此技術為一套新穎的疊代二元化運動估測，基於第一項二元化搜尋技術，發展出可自我感知、自我調節之功率可調適運動估測。相較於其他可調適運動估測技術，所提出的方法具有較佳的功率可調適能力、低頻寬需求，與較佳的功率-失真(Power-Distortion)特性曲線。這兩項技術業已發表或投稿於著名國際期刊及會議論文。其中，第二項技術並完成晶片中心(CIC)的晶片下線與測試。

王君在視訊編、解碼演算法與架構設計之研究工作並有多項貢獻與論文發表。在 H.264/AVC 解碼器系統設計方面，其提出一套最佳化的 H.264/AVC 解碼器參考軟體，架構於以 ARM RISC 為核心的硬體平台上，實現軟、硬體共設計之 QCIF 即時解碼器系統設計。此方面的研究，在 Google 學術搜尋，兩篇論文被引用共 21 次。在視訊轉碼器設計方面，其提出一套『低複雜度碼率-失真最佳化之多層位元流技術』，應用於異質轉碼(heterogeneous transcoding)系統上，可達到 1.4-8.6 dB PSNR 改善。在數位浮水印技術方面，王君提出一個新的數位浮水印嵌入技術，作為數位影像版權的保護。此方法的貢獻在於利用統計性的能量差異法，在面對各種不同的幾何與非幾何攻擊時，仍能有效維持其統計上能量差異，以保存所嵌入的數位浮水印，且不需要原始影像作為解出數位浮水印的憑據。此部分研究，刊登於著名期刊 IEEE Trans. Image Processing，並於 Google 學術搜尋，被引用達 21 次。王君另於 2001-2003 期間，積極參與 MPEG 國際標準會議的參照軟體(Optimized MPEG-4 Simple Profile reference software)撰寫與維護工作，提出多項 MPEG 貢獻文件，並獲 MPEG 會議的書面表揚。

總體而言，王君論文有相當之學術貢獻與重要性，且其積極而深入的參與國際標準會議之參照軟體撰寫，使其貢獻成為標準的一部份，兼具實用之價值。以下詳列其所發表之論文（依代表性排序）：

期刊論文

1. **S.-H. Wang**, W.-H. Peng, Y.-W. 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 decoder with block level pipelining," *Journal of VLSI Signal Processing Systems*, vol. 41, no. 1, pp. 93-110, Jan. 2005. **[Google Citation: 7]**
2. **S.-H. Wang**, W.-L. Chen, and Tihao Chiang, "An efficient FGS to MPEG-1/2/4 single layer transcoder with R-D optimized multi-layer streaming technique for video quality improvement," *Journal of the Chinese Institute of Engineers*, vol. 31, 2008. **(to be appeared)**
3. S.-C. Chang, W.-H. Peng, **S.-H. Wang**, and T. Chiang, "A Platform based Bus-interleaved Architecture for Deblocking Filter in H.264/MPEG-4 AVC," *IEEE Trans. Consumer Electronics*, vol. 51, no. 1, pp. 249-255, Feb. 2005.
4. **S.-H. Wang**, and Y.-P. Lin, "Wavelet tree quantization for copyright protection watermarking," *IEEE Trans. Image Processing*, vol. 13, no. 2, pp. 154-165, Feb. 2004. **[Google Citation: 21]**

審查中國際期刊

1. **S.-H. Wang**, and T. Chiang, "A power adaptive motion estimation IP core design using iterative binary search," *IEEE Trans. Circuits and Systems for Video Technology*, 2006.
2. **S.-H. Wang**, S. -H. Tai, and T. Chiang, "A low power and bandwidth efficient motion estimation IP core design using binary search," *IEEE Trans. Circuits and Systems for Video Technology*, 2006.

國際會議論文

1. **S.-H. Wang**, W.-L. Tao, W.-H. Peng, C.-N. Wang, and T. Chiang, "Platform based design of all binary motion estimation (ABME) with bus interleaved architecture," *Proc. IEEE International Symposium on VLSI Technology, System and Applications*, Hsinchu, April 2005.
2. **S.-H. Wang**, C.-N. Wang, and T. Chiang, "A complexity aware variable-bit-depth motion estimation," *Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, Jan. 2005.
3. S.-C. Chang, W.-H. Peng, **S.-H. Wang**, and T. Chiang, "A platform-based de-blocking filter design with bus interleaved architecture for H.264," *Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, Jan. 2005. **[Google Citation: 17]**
4. **S.-H. Wang**, W.-H. Peng, Y. He, G.-Y. Lin, C.-Y. Lin, S.-C. Chang, C.-N. Wang, and T. Chiang, "A Platform Based MPEG-4 Advanced Video Coding Decoder with Block Level Pipelining," *Proc. IEEE ICICS-PCM*, Singapore, Nov. 2003.

5. **S.-H. Wang**, and Y.-P. Lin, “Blind watermarking using wavelet tree quantization,” *Proc. IEEE International Conference on Multimedia and Expo*, Lausanne, August, 2002.

MPEG 視訊標準會議文件

1. **S.-H. Wang**, C.-N. Wang, Yi-Shin Tung, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M9951: AHG report on editorial convergence of MPEG-4 reference software,” Oct. 2003.
2. **S.-H. Wang**, C.-N. Wang, Y.-S. Tung, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M9632: AHG report on editorial convergence of MPEG-4 reference software,” July 2003.
3. **S.-H. Wang**, C.-N. Wang, Y.-S. Tung, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M9355: AHG report on editorial convergence of MPEG-4 reference software,” March 2003.
4. **S.-H. Wang**, C.-N. Wang, G.-Y. Lin, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M9073: AHG report on editorial convergence of MPEG-4 reference software,” Dec. 2002.
5. **S.-H. Wang**, C.-N. Wang, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M8886: Proposed text of proposed draft technical reports of ISO/IEC PDTR 14496-7 for optimized simple profile reference software,” Oct. 2002.
6. **S.-H. Wang**, C.-N. Wang, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M8884: AHG report on editorial convergence of MPEG-4 reference software,” Oct. 2002.
7. **S.-H. Wang**, C.-N. Wang, Tihao Chiang, and H.F. Sun, “ISO/IEC JTC1/SC 29/WG 11 M8603: AHG report on editorial convergence of MPEG-4 reference software,” July 2002.
8. **S.-H. Wang**, Y.-C. Lin, C.-N. Wang, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M8408: AHG report on editorial convergence of MPEG-4 reference software,” May 2002.
9. **S.-H. Wang**, C.-N. Wang, T. Chiang, and H. Sun, “ISO/IEC JTC1/SC 29/WG 11 M8041: AHG report on editorial convergence of MPEG-4 reference software,” March 2002.

專利

1. M.-Y. Huang, T.-L. Su, **S.-H. Wang**, C.-N. Wang and T. Chiang, “MPEG-4 streaming system with adaptive error concealment,” 美國專利，專利號 20060104366.

審查中專利

1. **S.-H. Wang**, L. Kohn, and T. Chiang, “Mode decision using approximate 1/2 pel interpolation,” 美國專利. (Filed on Nov. 23, 2005)
2. S.-C. Chang, W.-H. Peng, **S.-H. Wang**, and T. Chiang, “A Platform Based

Bus-interleaved Architecture for Deblocking Filter in H.264/MPEG-4 AVC," 美國專利.
(Filed on March 24, 2005)

總結，王君在論文中所提出之技術顯著的改善了運動估測的功率消耗與可調適能力，加強了功率受限編碼系統上的應用範圍與效能。相關的視訊編、解碼系統研究並有多篇的期刊與會議論文發表。王君在修業期間，已滿足本所在課業和學業及研究上之要求，並具備團隊合作和獨立研究之能力，因此特以推薦之。

推薦人

國立交通大學 電子工程系教授

蔣迪家

中華民國 96 年 7 月 30 日

國立交通大學

論文口試委員會審定書

本校電子工程學系電子研究所博士班王士豪君

所提論文適用於功率受限視訊編碼系統之運動估測演算法與積體電路架構設計

合於博士資格標準、業經本委員會評審認可。

口試委員：

黃仲陵

黃 仲 陵

張隆紋

張 隆 紋

杭學鳴

杭 學 鳴

任建葳

任 建 葳

張文鐘

張 文 鐘

王聖智

王 聖 智

蔡宗漢

蔡 宗 漢

蔣迪豪

蔣 迪 豪

指導教授：

蔣迪豪

教授

蔣 迪 豪

所 長：

邱碧秀

教授

邱 碧 秀

系 主 任：

周世傑

教授

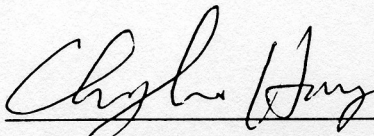
周 世 傑

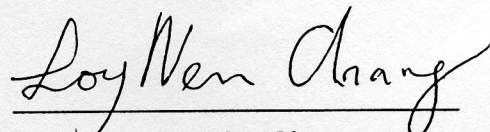
中華民國九十六年九月十二日

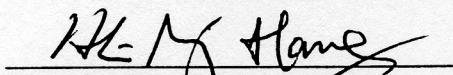
Department of Electronics Engineering
& Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

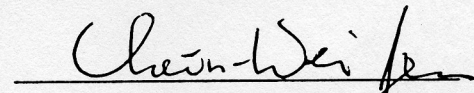
Date : 2007/09/12

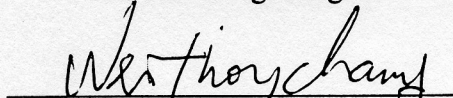
We have carefully read the dissertation entitled Algorithm and
Architecture Design of Motion Estimation for Power Constrained Video Coding Systems
submitted by Shih-Hao Wang in partial fulfillment of the requirements of
the degree of DOCTOR OF PHILOSOPHY and recommend its acceptance.

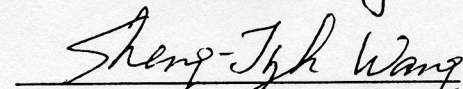

Chung-Lin Huang

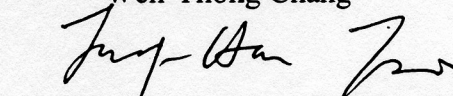

Long-Wen Chang

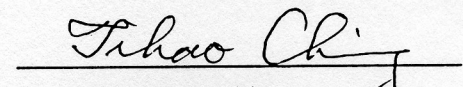

Hsueh-Ming Hang

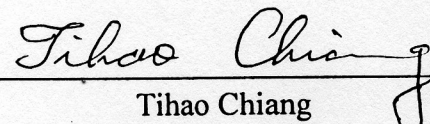

Chein-Wei Jen

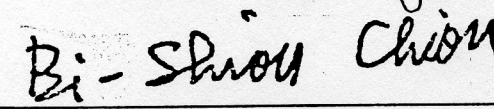

Wen-Thong Chang

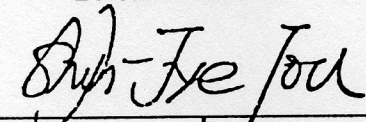

Sheng-Jyh Wang


Tsung-Han Tsai


Tihao Chiang

Thesis Advisor : 
Tihao Chiang

Director : 
Bi-Shiou Chiou

Chairman : 
Shyh-Jye Jou

國立交通大學

博碩士論文全文電子檔著作權授權書

(提供授權人裝訂於紙本論文書名頁之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學電子工程系所 系統 組，96 學年度第 1 學期取得博士學位之論文。

論文題目：適用於功率受限視訊編碼系統之運動估測演算法與積體電路架構設計
指導教授：蔣迪豪

■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：

本校及台灣聯合大學系統區域網路	<input checked="" type="checkbox"/> 立即公開
校外網際網路	<input checked="" type="checkbox"/> 立即公開

■ 全文電子檔送交國家圖書館

授權人：王士豪

親筆簽名：王士豪

中華民國 96 年 10 月 1 日

國立交通大學

博碩士紙本論文著作權授權書

(提供授權人裝訂於全文電子檔授權書之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學電子工程系所 系統 組，96 學年度第 1 學期取得博士學位之論文。

論文題目：適用於功率受限視訊編碼系統之運動估測演算法與積體電路架構設計
指導教授：蔣迪豪

■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

本論文為本人向經濟部智慧局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為：_____，請將論文延至_____年_____月_____日再公開。

授權人：王士豪

親筆簽名：王士豪

中華民國 96 年 10 月 1 日

國家圖書館 博碩士論文電子檔案上網授權書

(提供授權人裝訂於紙本論文本校授權書之後)

ID:GT009011843

本授權書所授權之論文為授權人在國立交通大學電子工程
系所 96 學年度第 1 學期取得博士學位之論文。

論文題目：適用於功率受限視訊編碼系統之運動估測演算
法與積體電路架構設計

指導教授：蔣迪豪

茲同意將授權人擁有著作權之上列論文全文（含摘要），
非專屬、無償授權國家圖書館，不限地域、時間與次數，
以微縮、光碟或其他各種數位化方式將上列論文重製，並
得將數位化之上列論文及論文電子檔以上載網路方式，提
供讀者基於個人非營利性質之線上檢索、閱覽、下載或列
印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應
依著作權法相關規定辦理。

授權人：王士豪

親筆簽名：王士豪

民國 96 年 10 月 | 日

適用於功率受限視訊編碼系統之運動估測演算法與積體電路 架構設計

研究生：王士豪

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

受限於可攜式行動設備之有限的電池容量，功率受限之視訊編碼系統設計逐漸受到了重視，在此之中，以低功率和功率可調適設計為最熱門的研究主題，本論文將以此兩主題為研究中心，以二元化搜尋技術為中心，逐步發展出兩項適用於低功率與功率可調適視訊編碼系統之運動估測技術，這兩項技術皆包含了演算法與積體電路架構設計。

本論文第一部份為提出一個具低功率與低頻寬需求之低功率全二元化搜尋之運動估測(Low Power-All Binary Motion Estimation, LP-ABME)積體電路設計。低功率與低頻寬需求為應用於行動視訊編碼應用上的兩大重要設計因素。為達到低功率與低頻寬需求，本技術架構於一個全二元化的運動估測(ABME)演算法上，藉由使用二元化的影像來完成運動估測，並將二元化的搜尋技術實現於金字塔式搜尋架構(pyramid search)下，以大量地降低了運動估測運算複雜度，且二元化的影像也降低了在 I/O 頻寬上的存取需求。為達成全二元化的運動估測(ABME)於積體電路實現，我們提出了一個基於原二元化的運動估測(ABME)之新的低功率全二元化搜尋之運動估測(LP-ABME)演算法與硬體架構設計。此設計具有四項重要的特色：

- (1)基於 MB 管狀設計的前處理器設計，
- (2)高硬體運算效率的二元化搜尋架構，
- (3)

平行化的 8x8 與 16x16 搜尋架構，(4)可平行處理雙向預測搜尋架構。第一項技術降低了對 I/O 存取頻寬上的需求，另三項則降地了運算複雜度與運算功率消耗。此積體電路架構設計在 I/O 存取頻寬、效能、與功率消耗上表現出很好的效能。功率消耗方面，執行 IPPPP CIF 30fps ，功率消耗為 763 微瓦(uW)，IPBPB CIF 30fps 則為 896 微瓦。I/O 存取頻寬方面，則可節省 54.3 至 67.1%.

本論文第二部份為提出一個具功率感知能力的功率可調適疊代二元化搜尋 (Power Adaptive Iterative Binary Search, PA-IBS)技術，目的在改善: (1)功率可調適能力，(2)高硬體閒置，與(3)功率-失真(Power-Distortion)效能。舊有功率可調適運動估測設計，使用了硬體遮罩的方式實現功率可調適性，卻也延伸出許多問題，如:多餘的 I/O 存取頻寬浪費，多餘的記憶體頻寬浪費，與高硬體閒置等問題，導致功率可調適能力降低，與不好的功率-失真效能。為解決這些問題，本論文延伸了二元化搜尋技術的應用，發展出一套具功率可調適能力的演算法與積體電路架構。此演算法稱之為功率可調適疊代二元化搜尋(PA-IBS)，其包含了: (1)疊代二元化搜尋技術，與(2)內容感知之疊代迴圈控制器。疊代二元化搜尋技術使用了最多八個迴圈的二元化搜尋，藉由疊代迴圈的應用，達到不同層次的預測品質與運算複雜度。內容感知之疊代迴圈控制器，則藉由運動向量(motion vector)來偵測視訊影像的運動複雜層度，以調整疊代迴圈數，並達到利用最少的迴圈達到最佳的預測品質與運算複雜度。積體電路設計方面，則使用頻率延展(frequency scaling)技術，將疊代迴圈數與功率消耗作一連結，藉由調整疊代迴圈數，來控制功率消耗與功率可調適能力，並解決高硬體閒置問題。實驗結果證明，相較於既有的功率可調適設計，PA-IBS 可改善功率可調適能力達 19-125%，I/O 存取頻寬需求最高則可降低 87.5%，同時具有較佳的功率-失真曲線。

總結，本論文提出兩個適用於低功率與功率可調適視訊編碼系統之運動估測技術。第一個技術達成了低於 1 毫瓦(mW)的功率消耗，和高於 50%的 I/O 存取頻寬節省。第二個技術則改善了現有功率可調適設計在功率可調適能力、高硬體開置，與不好的功率-失真(Power-Distortion)效能等方面的問題。在功率受限視訊編碼系統上面的應用，提供了顯著的改善與更大的應用空間。



Algorithm and Architecture Design of Motion Estimation for Power Constrained Video Coding Systems

Student: Shih-Hao Wang

Advisor: Dr. Tihao Chiang

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

Abstract

The design of power constrained video coding systems has drawn attentions in mobile devices or portable terminals due to the limited battery energy. Among the power constrained video coding applications, low power and power adaptive designs are two of the most attractive design topics. Inside the video coding system, motion estimation (ME) takes most of computation powers, and becomes the design bottleneck of the low power and power adaptive video coding systems. This thesis contains 2 major parts to address the design issues of low power and power adaptive motion estimation.

The first part is to propose a new Low Power-All Binary Motion Estimation (LP-AMBE) hardware design for motion estimation to achieve low power and bus bandwidth efficiency. Low power and high bus bandwidth efficiency are the two key issues for portable video applications. To address such issues, we first study an efficient algorithm called all binary motion estimation (ABME), and analyze its architecture issues in operational flow and bus access. Then, we propose an hardware architecture for ABME with four new features (1) macroblock level pre-processing (2) efficient binary pyramid search structure (3) parallel processing of 8x8 and 16x16 block searches (4) parallel processing of bi-directional search. Such architecture leads to a superior performance in bus access, speed and power. The experiments show that the power

consumption is as low as 763uW for IPPPP CIF 30fps and 896uW for IPBPB CIF 30fps. The bus bandwidth savings are 54.3% for P-frame search and 67.1% for B-frame search.

The second part is to propose a new Power Adaptive Iterative Binary Search (PA-IBS) design for motion estimation to improve the power adaptation performance. In the prior power adaptive ME designs that use the hardware masking approach, there exist design overheads such as redundant bus access, unnecessary on-chip memory access, and poor hardware utilization that lead to poor power adaptation performance. Our proposed power adaptive solution addresses these issues with a new ME algorithm called Iterative Binary Search (IBS) and the associated hardware architecture called PA-IBS. The IBS uses eight binary searches where each search can be either an independent search or one of the eight joint searches. Hence, redundant bus and on-chip memory access are eliminated. A Content Adaptive Mechanism (CAM) is used to dynamically select the number of iterations on a macroblock basis. The PA-IBS uses the frequency scaling technique to provide a link between the number of iterations and the power consumption level. Therefore, it reduces hardware idling and enhances hardware utilization. Experiments show that the PA-IBS delivers lower peak power consumption, better power adaptation performance and lower bus bandwidth requirement as compared to the prior hardware masking based designs such as sub-sampling or least significant bits truncation methods. As compared to those approaches, the power adaptation performance is improved up to 19-125% and bus bandwidth is saved up to 87.5%.

In conclusion, we have presented two algorithm and architecture designs of motion estimation for different power constrained video coding applications, and showed the advantages in low power consumption and bus bandwidth requirements as compared to

prior works. The proposed power adaptive design is also shown to have better power adaptation ability and better power-distortion performance. Moreover, the proposed low power and power adaptive ME designs can be applied to upcoming Scalable Video Coding (SVC) standard for further complexity and power reduction.



誌 謝

從博士班入學到現在拿到學位，回想這幾年的點點滴滴，彷彿又回到這些記憶的時光隧道。一開始為了從事多媒體視訊方面的研究，轉換了研究領域，在蔣教授的指導與俊能學長的帶領下，開始了我的博士班研究。接著，進入了 commlab 這個大家庭，認識了很多厲害的學長姐、學弟妹，大家一起作研究、討論功課、游泳、烤肉、參加電子週的比賽..等等。還有那超強的 commlab 影音資料庫，提供了在無數研究作不下去的夜晚時，有打發時間的娛樂。在文孝學長的幫助下，慢慢走向了作 IC design 這個領域。在那無數的週末，跟自己帶的碩士班學弟們，討論如何作研究，同時也學習了自己帶 project 的經驗。

能完成博士學位，首先必須要感謝蔣迪豪教授。從收我進 group 開始，蔣老師就提供了我無數的機會去擴展自己的研究領域與國際視野，同時也提供了我無數的學習機會與成長。從參加國際會議、到園區公司去作計畫結案報告、自己帶學弟作計畫等等，這些都會是很寶貴的經驗。而在學校的研究之外，工作的選擇上也給我許多建議。非常謝謝蔣老師這幾年來在各方面的幫助及鼓勵。

其次，我要謝謝 commlab 的各位成員們，特別是俊能與文孝兩位學長。俊能學長從我進入蔣老師 group 開始，不論是在做研究或者參與計畫，都給了我很多的幫忙與建議。從完全不懂什麼是 MPEG 到可以在上面發展演算法、作研究，俊能學長給了我最大的幫助。在博士班後期，開始從事 IC design 這個領域，文孝學長則是從帶領我們作 H.264/AVC 解碼器計畫開始，給了我很多的意見與想法，提醒我很多忽略的細節。此外，項群、俊毅、鑑明、志鴻、以及所有 commlab 學長姐、同學們，大家在學業上的討論、生活上的幫忙，都給了我對 commlab 最好的回憶。

另外，我也要謝謝我的口試委員：交大電子系的杭學鳴教授、任建葳教授、王聖智教授、交大電信系的張文鐘教授、清大資工系的張隆紋教授、電機系的黃仲陵教授、中央電機系蔡宗漢教授。感謝您們在百忙之中能抽空給予我指導，也因你們的寶貴建議使得論文能更加完備。

最後，我要感謝我的家人，包括了我們父母親以及未婚妻秋女英。感謝你們在這幾年來的照顧、協助與包容。

謹以此論文獻給所有愛我與我所愛的人。

王士豪 2007/10/01

Contents

Table of Contents	1
List of Figures	4
List of Tables	7
1 Introduction	1
1.1 Motivations	1
1.1.1 Importance of Low Power Designs	2
1.1.2 Importance of Power Adaptive Designs	6
1.2 Power Constrained Motion Estimation Designs	7
1.2.1 Low Power Motion Estimation	8
1.2.2 Power Adaptive Motion Estimation	11
1.3 Organization and Contribution	14
2 Review of Power Constrained Motion Estimation Designs	16
2.1 Block Motion Estimation	16
2.1.1 Block Matching Criterion	16
2.1.2 Design Metrics Evaluation	17
2.1.3 Motion Estimation Hardware Design	22
2.1.4 Memory Hierarchy	23
2.2 Low Power Motion Estimation Designs	27
2.2.1 Low Power by Fast ME Algorithms	28
2.2.2 Low Power by Simplified Block Matching Criteria	29
2.2.3 Low Power by Efficient Hardware Architecture	30
2.3 Power Adaptive Motion Estimation Designs	31
2.3.1 Power Adaptive by Fast Algorithms	31
2.3.2 Power Adaptive by Pixel Number for Block Matching	38
2.3.3 Power Adaptive by Pixel Bit Precision	39
2.3.4 Summary of Power Adaptive Design Schemes	40
3 Bi-directional Binary Motion Estimation (BBME)	41
3.1 Introduction	41

3.2	Problem Statement	44
3.2.1	Review of ABME Algorithm	44
	A. Frame Level of Pre-processing	44
	B. Three Levels of Binary Pyramid Search	45
3.2.2	Design Issues of ABME Algorithm	46
	A. Pre-processing for Binary Images Generation	48
	B. Sequential LV2 Binary Pyramid Search Structure	49
	C. Support of B-frame and 8x8 Block Search	50
3.3	BBME Algorithm	50
3.3.1	Macroblock Pre-processing Unit (MBPPU)	53
3.3.2	Efficient LV2 Search	54
3.3.3	Parallel Processing of 8×8 and 16×16 Block Searches	54
3.4	Hardware Architecture	58
3.4.1	System Architecture	58
3.4.2	Macroblock Pre-processing Unit (MBPPU)	61
3.4.3	Three Levels of Binary Search	62
3.5	Experimental Results and Analysis	62
3.5.1	Rate-Distortion (R-D) Performance Evaluation	62
3.5.2	Hardware Design Performance	66
3.5.3	Bus Bandwidth Analysis	77
3.5.4	Comparison of ABME and BBME	81
3.6	Summary	85
4	Power Adaptive Iterative Binary Search (PA-IBS)	86
4.1	Introduction	86
4.2	Power Adaptation Performance	89
4.3	Power Adaptive Iterative Binary Search (PA-IBS) Algorithm	91
4.3.1	Binary Image Preprocessor (BIP)	92
4.3.2	Iterative Binary Search (IBS)	92
4.3.3	Content Adaptive Mechanism (CAM)	95
4.4	Hardware Design Issues	96
4.4.1	Power Adaptation	96
4.4.2	SOD Accumulation	97
4.4.3	Binary Image Preprocessor	99
4.5	PA-IBS Hardware Architecture	102
4.5.1	System Architecture	102
4.5.2	8×1 Line Search	103
4.5.3	Pipelined Buffers	104
4.6	Experimental Results and Analysis	107
4.6.1	Evaluation of Algorithmic Performance	107
	A. R-D Performance for PA-IBS without CAM	107
	B. R-D Performance for PA-IBS with CAM	108

4.6.2	Evaluation of Hardware Performance	109
	A. Chip Specification	109
	B. Evaluation on Bus Bandwidth	113
	C. Evaluation on Power Adaptation Performance	114
	D. Evaluation on Peak Power Consumption	114
	E. Evaluation on Power-Distortion Performance	115
4.7	Summary	122
5	Conclusions and Future Work	126
5.1	Conclusions	126
5.2	Future Work	128
	Bibliography	130



List of Figures

1.1	Power consumption of encoder modules [7].	4
1.2	Estimated ME power consumption for different video resolutions.	5
1.3	I/O bandwidth for different video resolutions.	5
1.4	Battery discharging curve [13].	7
1.5	Lifetime improvement by power adaptive designs.	8
1.6	Block diagram of portable multimedia player based on TI DaVinci platform [14].	9
1.7	Power management IC for DaVinci technology based portable electronics [15].	9
2.1	Block motion estimation.	18
2.2	Video encoder hardware architecture with ME module.	23
2.3	Motion estimation hardware architecture.	24
2.4	Search window data reuse. (a)Level C (b)Level D.	26
2.5	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Miyama's work [36] (b) Chao's work [30].	34
2.6	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Chen's work [26] (b) Huang's work [37].	35
2.7	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Wang's work [35] (b) Shen's work [39].	36
2.8	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67].	37
3.1	Motion predictors for LV2 search.	47
3.2	Functional block diagram of a generic video encoder by adopting ABME algorithm.	51
3.3	The processing flow of pre-processing module in ABME algorithm.	51
3.4	The processing procedure for ABME and BBME flow. (a) ABME flow [41] (b) BBME flow.	52
3.5	The pre-processing flow in macroblock pre-processing unit. (a) $K=30$ (b) $K=18$. (The shadow area is padding pixels)	56

3.6	The LV2 processing flow. (a) original LV2 flow in ABME (b) new LV2 flow in BBME. The number represents the processing order.	57
3.7	System architecture for the BBME design.	63
3.8	Architecture of macroblock based pre-processing unit.	64
3.9	Shared processing unit for three levels of binary pyramid searches.	65
3.10	R-D curves for full search (FS), ABME [?], and BBME designs with Foreman sequence. (a) IPPP (M=1) (b) IBPBP (M=2).	68
3.11	R-D curves for full search (FS), ABME [?], and BBME designs with Mobile sequence. (a) IPPP (M=1) (b) IBPBP (M=2).	69
3.12	Visual quality comparison for full search and BBME with Foreman 32 th frame. (Left: 34.73dB for full search, Right: 34.39dB for BBME.)	70
3.13	Visual quality comparison for full search and BBME with Foreman 99 th frame. (Left: 34.74dB for full search, Right: 34.36dB for BBME.)	70
3.14	Visual quality comparison for full search and BBME with Foreman 148 th frame. (Left: 34.99dB for full search, Right: 34.63dB for BBME.)	71
3.15	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Miyama's work [36] (b) Chao's work [30].	73
3.16	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Chen's work [26] (b) Huang's work [37].	74
3.17	Hexagonal plot of 6 design metrics for the propose BBME design.	75
3.18	Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67].	76
3.19	Pipeline timing of data movement and motion search.	82
4.1	Power adaptation curves for prior arts [56, 57, 58].	91
4.2	Functional block diagram of a generic video encoder with power adaptive-iterative binary search (PA-IBS).	94
4.3	Processing procedure of the power adaptive-iterative binary search (PA-IBS) algorithm.	94
4.4	Partitioned 8×8 regions for z-th binary search window under search range of ±16.	99
4.5	Block diagram of PA-IBS architecture.	105
4.6	Architecture of line based search engine.	106
4.7	Output data timing from line based search engine to the pipelined buffers for ϕ iterations of binary searches.	106
4.8	Temporal distribution of the target number of iterations (ϕ) for the PA-IBS algorithm. (a) Foreman with 300 fames (b) Flower&Garden with 250 frames.	109
4.9	Temporal distribution of PSNR and coding bits for the PA-IBS algorithm with and without the CAM on Foreman sequence.	110
4.10	Chip photo.	115
4.11	Comparison of bus bandwidth for the PA-IBS and conventional power adaptive designs with the hardware masking approach.	116

- 4.12 Comparison of the power adaptation performance for the PA-IBS and prior power adaptive designs. 117
- 4.13 Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67]. 120
- 4.14 Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) PA-IBS (CAR=1/8) (b) Chen's work [26]. 121
- 4.15 Power-Distortion curves for power adaptive designs. 123



List of Tables

1.1	Analysis on the power consumption for the state-of-the-art video encoder chip designs.	3
1.2	Data access bandwidth in one macroblock (MB) for full search. (SR=search range).	4
1.3	I/O access bandwidth for different image resolutions.	4
2.1	Required memory size and data reuse ratio for data reuse schemes of Level C and Level D.	27
2.2	Evaluation of low power designs using design metrics.	28
2.3	Evaluation of design metrics for low power designs (Group 1 and 2).	32
2.4	Evaluation of design metrics for low power designs (Group 3).	33
2.5	Summary of three groups of complexity adaptive ME works.	40
3.1	Evaluation of the pre-processing unit design schemes for pyramid based search.	49
3.2	Comparison of different K values for macroblock based pre-processing unit.	53
3.3	Evaluation of the ABME and BBME flow for block 8×8 and 16×16 searches in LV3.	55
3.4	Comparison for serial and parallel architecture.	61
3.5	R-D performance for full search (FS), ABME algorithm (ABME)[35] and the BBME algorithm at the bitrate of 256 kilo bps (N=300).	66
3.6	R-D performance for full search (FS), ABME algorithm [35] and the BBME algorithm at the bitrate of 512 kilo bps (N=300).	67
3.7	R-D performance for full search (FS), ABME algorithm [35] and the BBME algorithm at the bitrate of 1024 kilo bps (N=300).	67
3.8	Gate count and execution cycles for each module of our design.	72
3.9	Summary of cycles of data movement for P-frame and B-frame searches.	77
3.10	Performance comparison with state-of-the-art designs.	78
3.11	Design metrics evaluation for the state-of-the-art low power designs.	79
3.12	Design metrics evaluation for the state-of-the-art low power designs.	80

3.13	Bus bandwidth analysis for conventional 8-bit ME scheme and the proposed design (Search range = [-16, +15]).	83
3.14	Comparison of prior ABME design [35] and the proposed BBME design.	84
4.1	A summary of power adaptive motion estimation designs.	88
4.2	Filters used to generate binary images.	95
4.3	Analysis of hardware requirement for four region sizes.	100
4.4	Memory index for the partitioned regions in search range of ± 16 (z = iteration index from 0 to 7). Each word is 64 bits.	100
4.5	Evaluation of the pre-processing unit design schemes for PA-IBS.	101
4.6	R-D performance for three power adaptive algorithms as compared to full search (LSB = LSB truncation, SUB = sub-sampling, IBS = iterative binary search).	111
4.7	Complexity reduction for PA-IBS algorithm with CAM as compared to PA-IBS without CAM.	112
4.8	Chip design specification of PA-IBS.	116
4.9	Comparison of the power adaptive designs.	117
4.10	Comparison of H.264/AVC ME hardware designs.	118
4.11	Evaluation of design metrics for low power designs (Group 3).	119
4.12	Power-Distortion performance for power adaptive designs.	124



Chapter 1

Introduction

1.1 Motivations



With the rapid development of communication techniques and the popularity of mobile devices, vendors are providing more and more content services for end users. These content services include Digital TeleVision (DTV), Multimedia Messaging Services (MMS), mobile TV programs, MP4 movie/MP3 music playing, and video recording applications on Digital Still Camera (DSC), etc. Among these applications, video encoding/decoding applications are boosting the demands since the mobile devices are more powerful now than ever before to process heavier duty tasks such as video recording and compression. However, these devices are still powered by batteries and the battery power is still limited. Hence, developing the video coding application under power constraints to have efficient power usage for longer battery life is becoming important.

1.1.1 Importance of Low Power Designs

The video coding applications for mobile devices draw attentions in designing the video coding system in a more power efficient way. The most popular applications are to design a video coding system by minimizing its power for longer battery life. This design approach for power minimization is referred to as *low power designs*[34, 35, 36, 37, 38, 39, 40]. When the power consumption is reduced, the battery life can be lifted.

We survey the MPEG-related encoder chip designs in the academic or industrial areas [19, 7, 8, 9, 10, 11, 12] and summarize their design information in Table 1.1. The power consumption for the encoder chips is in the range of 600 milli-Watt (mW) to 4.2 Watt(W). Inside the encoder, the power consumption for the ME module will also increase to 1.2 W when the video resolutions increase to HDTV 720P (1280×720) or 1080I (1920× 1080). This shows the importance of low power ME designs, especially for the high resolutions of video coding applications.

Kumaki *et al.* [7] further analyze the power consumption of each module in their encoder chip design. The power consumption for motion estimation (ME), DCT/Q/IQ/IDCT, VLC, video in/out, SDRAM interface, and clock distribution are 39, 15, 5, 3, 8, and 22%, respectively, in which ME and the DRAM access take around 50% of the total encoder power (i.e. 750mW for D1 720x480 30fps under 0.35 μ m process). Fig. 1.1 shows the distribution of power consumption for each of the modules in [7]. After scaling it image resolution from D1 (720×480) to CIF, the power consumption for the ME is around 15 mW.

When image resolution and search range become larger, the power consumption for the ME becomes unacceptable. Fig. 1.2 plots the estimated power consumption of ME for

Table 1.1: Analysis on the power consumption for the state-of-the-art video encoder chip designs.

	Mizuno [19]	Kumaki [7]	Huang [46]	
Video standard	MPEG-2	MPEG-2	H.264/AVC	
Resolution	720×480	720×480	720×480 and 1280×720	
Encoder power	1.5W/0.35 μ m	0.7W/0.13 μ m	581mW/0.18 μ m (D1 with 4 references), and 785mW/0.18 μ m (720P with 1 reference)	
	Fujitsu [9]	Zoran [10]	Sanyo [11]	TI [12]
Video standard	H.264/AVC	H.264/AVC	H.264/AVC	H.264/AVC
Resolution	1080I	720P	1080P	720×480
Encoder power	750mW/0.09 μ m	n.a. ¹	4.2W/n.a.	n.a.

¹not available

different search ranges and image resolutions. From this Figure, when image resolution increases from CIF to HDTV 1080I60, the power consumption will increase from 15 mW to 19.6 W, which is a huge power consumption.

When image resolution and search range become larger, the I/O bandwidth also increases very quickly. Table 1.2 analyzes the I/O access bandwidth in one macroblock. Table 1.3 lists the I/O bandwidth for different search range and image resolution, and the I/O bandwidth is plotted in Fig. 1.3. From this Figure, when image resolution increases from CIF to HDTV 1080I60, the I/O bandwidth will increase from 13.2 mega bytes/sec to 642.5 mega bytes/sec. The huge I/O bandwidth cause more power consumption and become another ME design issue.

The issues of high power consumption and high I/O bandwidth shows the motivations for the low power ME designs.

Table 1.2: Data access bandwidth in one macroblock (MB) for full search. (SR=search range).

Scheme	In/out data	Full search
P-frame	current block	16×16
	reference block	$16 \times (16 + (SR + 2) \times 2)$
	Motion vectors	$4 \text{ (bytes/MV)} \times 5 \text{ (MVs)}$
	Total (bytes/MB)	$596 + 32 \times SR$
B-frame	current block	16×16
	reference block	$(16 \times (16 + (SR + 2) \times 2)) \times 2$
	Motion vectors	$4 \text{ (bytes/MV)} \times 5 \text{ (MVs)}$
	Total (bytes/MB)	$936 + 64 \times SR$

Table 1.3: I/O access bandwidth for different image resolutions.

I/O bandwidth (Mbytes/s)	CIF 30fps, SR= ± 16	CIF 30fps, SR= ± 32	D1 30fps, SR= ± 32	HDTV 720P 30fps, SR= ± 32	HDTV 1080I 60fps, SR= ± 64
Full search	13.2	19.3	65.6	175.0	642.5

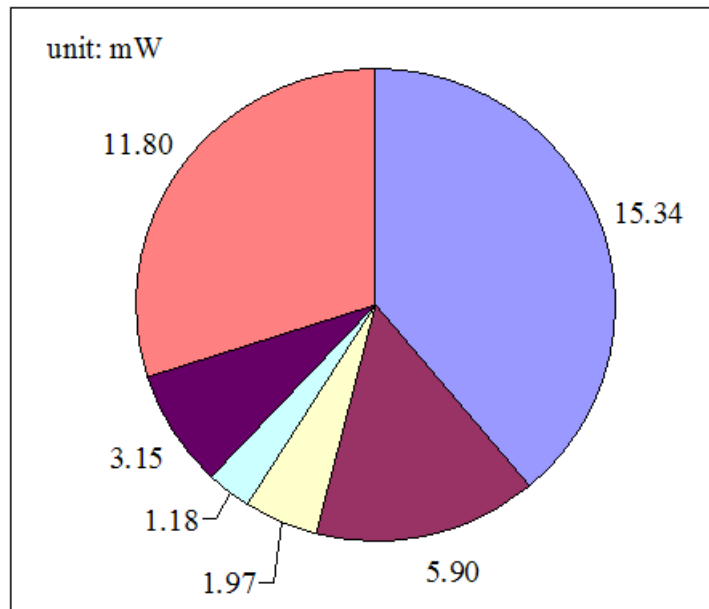


Figure 1.1: Power consumption of encoder modules [7].

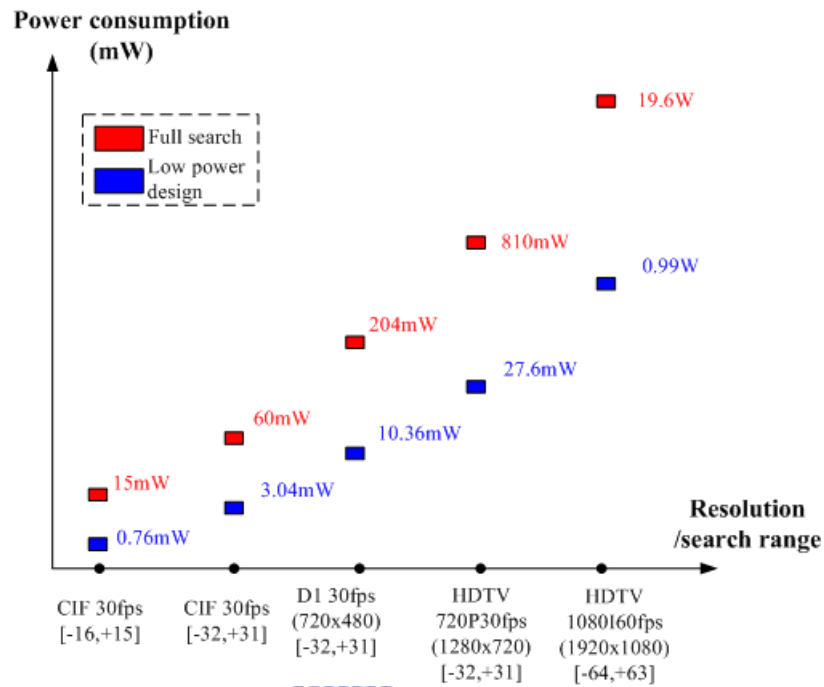


Figure 1.2: Estimated ME power consumption for different video resolutions.

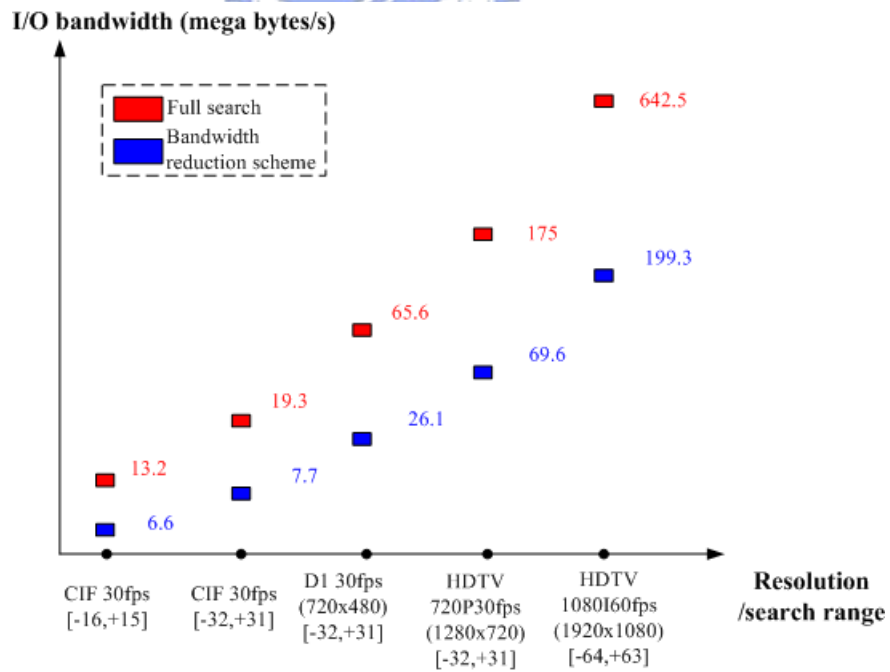


Figure 1.3: I/O bandwidth for different video resolutions.

1.1.2 Importance of Power Adaptive Designs

Another approach for power constrained video applications is to design with configurable power modes for different demands from low power to high quality applications. Such a design approach is referred to as *power adaptive designs* [45, 46, 47] or *power aware designs* [59, 5, 56]. The power adaptive designs can adapt the configuration to a suitable battery status manually or automatically considering the video content variations, and the power and video quality tradeoff [5]. That means the power adaptive designs have wider application range to adapt the power consumption and video coding quality to different application scenarios.

Fig. 1.4 shows the discharging curve of the 700 mAh AA NiCad battery [13]. Similar discharging curves can be observed for different types of batteries [13]. With the constant loading, the power discharging curve is shown in Fig. 1.5. Under this power discharging curve, a low power design will stop at T_1 since the remaining power is under its operational power. However, a power adaptive design can adapt to the real battery power status by adjusting its power consumption to extend its lifetime to T_2 with ΔT lifetime improvement. The lifetime improvement show the motivations for the power adaptive ME designs.

Fig. 1.6 shows how an power adaptive design works on a battery operated multimedia system. This system is built on TO DaVinci platform. For power adaptive applications, this system builds an power management mechanism to control the power consumption of LCD, DSP/CPU, and I/O peripherals by detecting the battery status. As shown in Fig. 1.7, the power management mechanism uses 3 step-down converters to adjust the voltage of DSP/CPU, memory and I/O peripherals for 32 levels of power consumptions (i.e. 0.8V 1.6V, step=0.025V). The power control signal is via I²S communication protocol. The

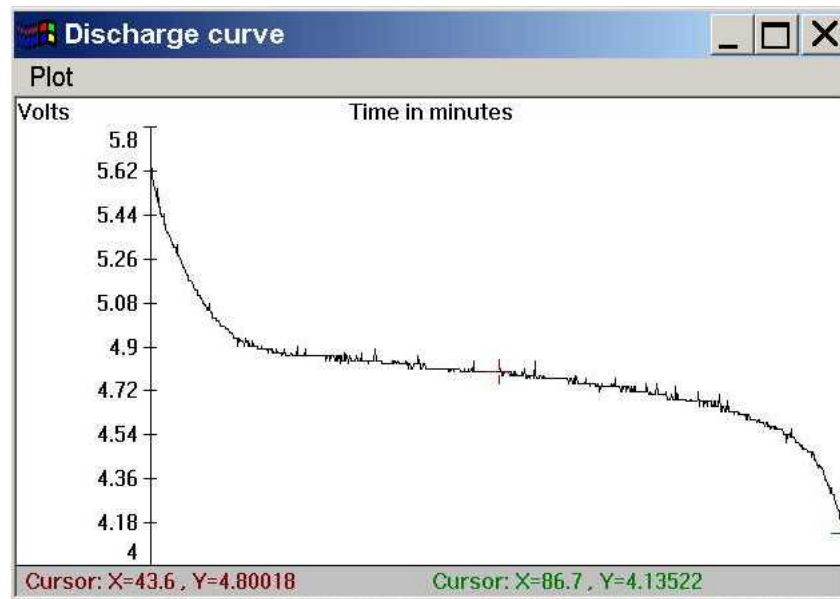
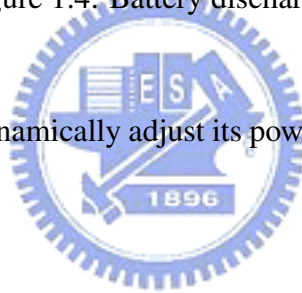


Figure 1.4: Battery discharging curve [13].

power adaptive ME can dynamically adjust its power consumption according to the control signal.



1.2 Power Constrained Motion Estimation Designs

The power constrained motion estimation design has two major applications: (1) low power design, and (2) power adaptive design. Low power design is to minimize the power consumption for ME with acceptable quality loss. On the other hand, the power adaptive design is to achieve a good power and video quality tradeoff to provide best quality under the same power consumption or minimal power consumption under fixed video quality. In this thesis, we will discuss the design challenges and our solutions.

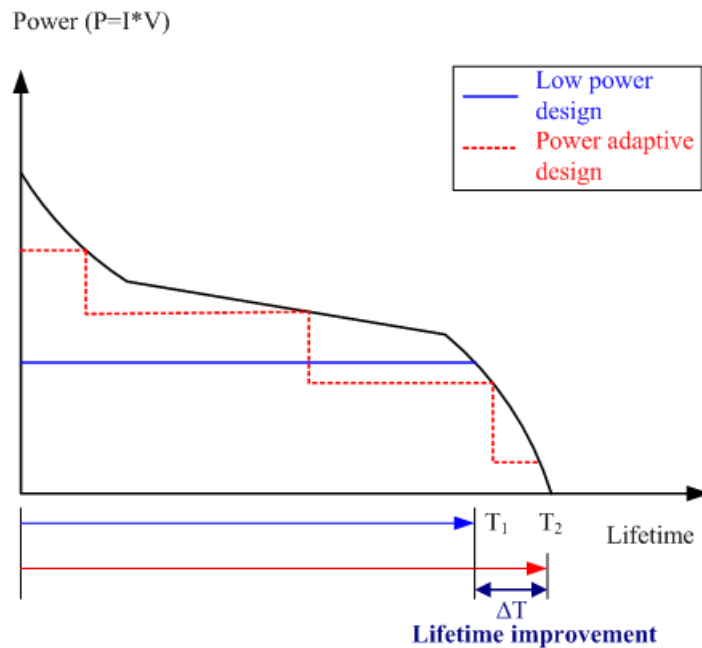


Figure 1.5: Lifetime improvement by power adaptive designs.

1.2.1 Low Power Motion Estimation

Motion estimation (ME) is the most computationally expensive module in multimedia compression standards such as MPEG-1/2/4 and H.26x. For portable video applications, low power and efficient bus access are two major design goals [42]. The huge power consumption from ME needs to be reduced to extend battery life. The ME is also a data intensive module. It moves huge data for pattern matching to find the best predicted block with minimal distortion for motion compensation. The amount of data movement increases proportionally to the square of search range, and becomes the performance bottleneck for System-on-Chip (SoC) designs due to the limited available bus bandwidth. Hence, an efficient data movement scheme via bus is another key design issue for portable video applications.

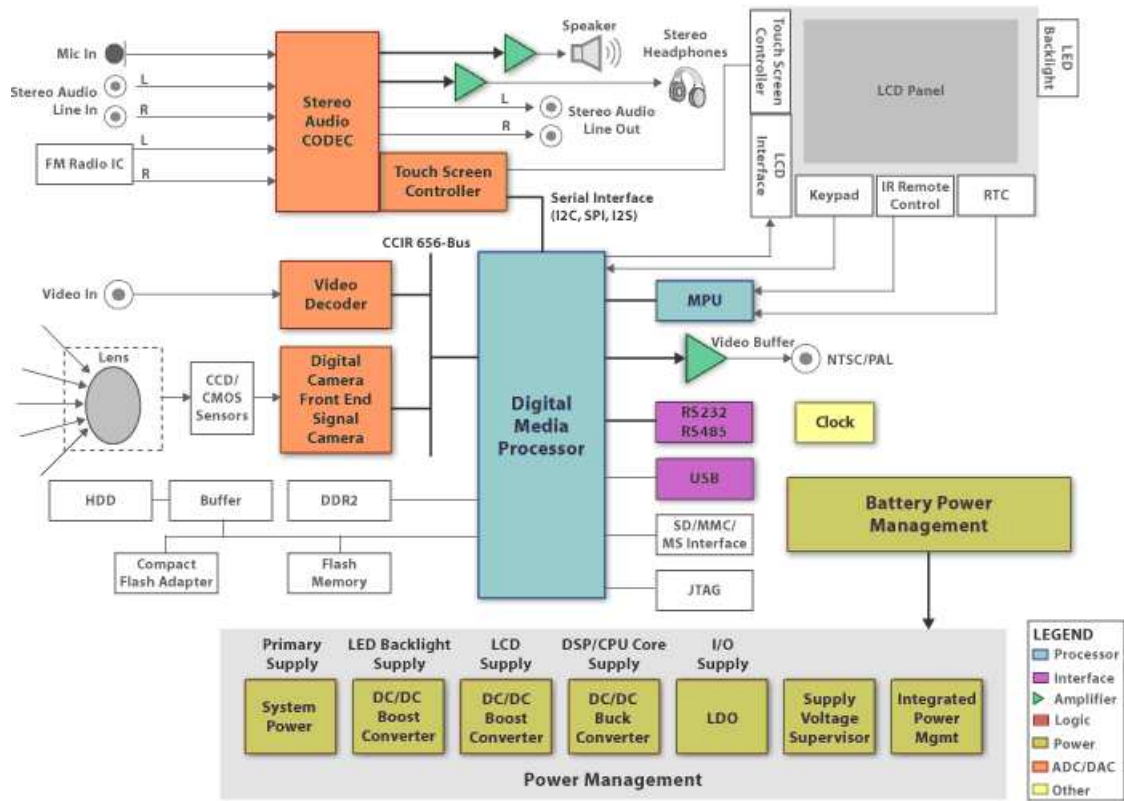
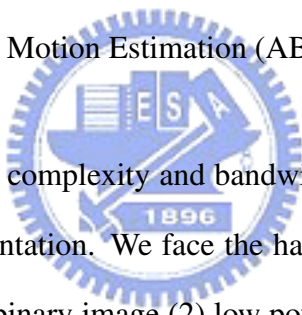


Figure 1.6: Block diagram of portable multimedia player based on TI DaVinci platform [14].



Figure 1.7: Power management IC for DaVinci technology based portable electronics [15].

To investigate low power ME designs, most of them are developed based on fast search algorithms [34, 35, 36, 37, 38] or low cost full search architectures [39, 40]. These fast algorithms include three-step search (TSS) [38], gradient descent search (GDS) [36], global elimination search [37], binary search [34, 35], etc. For low power or low cost full search architectures, one-dimensional (1-D) systolic array is the most widely used architecture [33]. Among these designs, binary search [34, 35] has both advantages of low computational complexity and low bus bandwidth requirement. The reason is it reduces the pixel precision from eight bits to one bit for block matching. Such a search strategy can also be viewed as a kind of feature matching with binary images. Therefore, in this paper, we develop our low power and bandwidth efficient ME design based on a binary pyramid search algorithm called All Binary Motion Estimation (ABME) [41].



Although ABME [41] is a low complexity and bandwidth efficient algorithm, it is not well optimized for VLSI implementation. We face the hardware design challenges in (1) image pre-processing to form the binary image (2) low power and bus bandwidth efficient architecture for binary pyramid search (3) support of bi-directional (or called B-frame) and 8x8 block searches. We solve these design issues in the proposed hardware architecture. Firstly, we propose a new pre-processing flow for binary image generation at the macroblock (MB) level instead of the original frame level. The binary image generation and binary motion search are integrated as an MB level pipelining to simplify redundant bus access. Secondly, we re-examine the data flow in the three levels of binary pyramid search structure, and modify the algorithm to remove the data dependency and inefficient operations for the second level of search. Finally, we address the design issues in B-frame search scheme and optimize the hardware architecture to enhance the processing throughput.

The contributions of this work include:

- Modified ABME algorithm (referred to as BBME) for efficient VLSI implementation. The BBME algorithm is developed based on a low complexity ABME algorithm [41]. We modify the ABME in the binary image generation and search method for efficient VLSI implementation. The power consumption for video encoding with CIF 30 fps and search range of [-16, +15] only needs less than 1 mW.
- MB level pipelining architecture for efficient bus access. We propose a new pre-processing flow for binary image generation at MB level as opposed to frame level. The new processing flow integrates both binary image generation and binary motion search using MB level pipelining to avoid repeated bus access. The bus bandwidth saving can achieve up to 67.1%.
- Bi-direction binary search architecture. We design our hardware to be able to handle B-frame search in parallel. It reuses the same current search data to save on-chip memory access and power. Thanks to the simple binary image matching, the gate counts have increased twice but not as much as the conventional 8-bit designs.

1.2.2 Power Adaptive Motion Estimation

The power adaptive designs have become an important feature especially for portable video applications [59]. Unlike the low power designs that aim for minimized power consumption, the power adaptive design targets on the efficient allocation of power resources with equal video quality and longer battery life. In multimedia compression systems such as MPEG-1/2/4 and H.26x, the motion estimation (ME) that dominates the power con-

sumption of the video encoder plays a key role in the power adaptive design. We will present a power adaptive ME design to improve power allocation and power efficiency.

In the power adaptive or complexity adaptive ME algorithms and designs [56, 57, 58, 47, 48, 49, 50, 45], we can roughly categorize them into two types according to their implementation methods. The first type is to achieve power adaptation by integration of multiple search strategies. This type adopts 2 to 3 search strategies such as three-step search, diamond search or full search to deliver different levels of search complexity. For example, the authors in [47] proposed a three-mode complexity adaptive method by using three-step search and enhanced four-step search for low power applications, and full search for high quality applications. Although this type of method can provide large scale of complexity differences, the coding quality for low power modes usually has significant quality loss.

The second type is to achieve power adaptation by simplified matching criterion. The simplified criterion include bit-depths truncation, pixel decimation, etc. By keeping different bit-depths or decimated pixel resolutions for block matching, the design can achieve different levels of computational complexity and power consumption. For example, the authors in [57, 58, 49] adopt the least-significant-bit truncation method to design their power adaptive ME. Pixel bit-depth of 1 or 2 is served for low power mode, and bit-depth of 8 is served for high quality mode. This type can provide the significant power reduction by dynamically adjusting the bit-depths, but it still suffers from significant quality loss in low power mode.

For both of the above 2 methods, they have the same problems of significant quality loss in low power modes. The bit-depth truncation method also has the issues in limited pixel bit-depths and bit-plane dependency. Limited pixel bit-depths cause the difficulty for

fine-granularity of power adaptation. Bit-plane dependency causes the inefficiency for data access and processing. To address these issues, a new power adaptive ME algorithm and hardware architecture called Power Adaptive-Iterative Binary Search (PA-IBS) is proposed with four key features:

To address such issues, a new power adaptive ME design called Power Adaptive-Iterative Binary Search (PA-IBS) is proposed with two features:

- *Frequency decomposed bit-planes design*: PA-IBS algorithm adopts the frequency decomposition method for bit-planes design. The new bit-plane design method generates directional and gradient image features in binary format, and can provide better rate-distortion performance as compared to using pixel bit-planes.
- *Finer granularity of power adaptation*: The number of frequency decomposed bit-planes is not limited to pixel bit-depths. This allows finer granularity of power adaptation for smooth power and video quality adjustment.
- *Independent bit-plane processing*: The frequency decomposed bit-planes can be individually stored in the memories and independently processed. Therefore, we can avoid unnecessary memory access and data processing to those unrelated bit-planes.
- *Frequency scaling based hardware architecture*: The independent bit-plane processing provides the advantage to design the hardware for processing single bit-plane instead of all bit-planes. To full use this hardware design for single bit-plane processing, the frequency scaling technique scales the working frequency with the number of bit-planes to be processed. Such hardware architecture reduces the overheads to design the hardware for the worst case of all bit-planes, and enhances the hardware

utilization and power adaptation performance.

1.3 Organization and Contribution

In **chapter 2**, we review the principle of motion estimation and the block matching criteria. Toward ME hardware design, we present the design metrics for evaluation of different ME works. The memory hierarchy architecture is also analyzed. Then, we survey low power and power adaptive ME designs, and categorize them into three major groups according to their implementation methods. The frequently cited ME works are also evaluated according to the design metrics, and used as the reference works for our low power and power adaptive binary motion search algorithm and architecture designs.

In **chapter 3**, a new Low Power ME algorithm and architecture design called Bi-directional Binary Motion Estimation (BMBE) is proposed to achieve low power and bus bandwidth efficiency. Low power and high bus bandwidth efficiency are the two key issues for portable video applications. To address such issues, we first study an efficient algorithm called all binary motion estimation (ABME), and analyze its architecture issues in operational flow and bus access. Then, we propose an hardware architecture called BBME with four new features (1) macroblock level pre-processing (2) efficient binary pyramid search structure (3) parallel processing of 8x8 and 16x16 block searches (4) parallel processing of bi-directional search. Such architecture leads to a superior performance in bus access, throughput and power.

In **chapter 4**, a new Power Adaptive Iterative Binary Search (PA-IBS) design for motion estimation is proposed to improve the power adaptation performance. In the prior power adaptive ME designs that use the hardware masking approach, there exist design

overheads such as redundant bus access, unnecessary on-chip memory access, and poor hardware utilization that lead to poor power adaptation performance. Our proposed power adaptive solution addresses these issues with a new ME algorithm called Iterative Binary Search (IBS) and the associated hardware architecture called PA-IBS. The IBS integrates a new frequency decomposed bit-plane design method to improve the rate-distortion curve and provide the flexibility for finer granularity of power adaptation. The IBS also executes the multiple bit-plane searches in an either individual or accumulated manner, thus redundant bus and on-chip memory access are eliminated. A Content Adaptive Mechanism (CAM) is used to dynamically select the number of iterations on a macroblock basis. The PA-IBS uses the frequency scaling technique to provide a link between the number of iterations and the power consumption level. Therefore, it reduces hardware idling and enhances hardware utilization.

In **chapter 5**, the concluding remarks and future works are addressed.



Chapter 2

Review of Power Constrained Motion

Estimation Designs

2.1 Block Motion Estimation



2.1.1 Block Matching Criterion

Motion estimation is to remove the temporal redundancy between neighboring frames for efficient video coding. For practical applications, a whole frame is partitioned into small blocks for motion compensated prediction. This block based method needs less data for one point of block matching, and thus is widely used in video compression standards such as MPEG-1/2/4 and H.26x.

Fig. 2.1 shows the block motion estimation flow to find the best matched block from previous frame. The current frame I_C is firstly partitioned into several $L \times L$ blocks. The motion estimation is done by checking all the candidate blocks in the search window of the

reference frame I_R , and the best matched block is found with minimal matching distortion. The distance from the current block to the best matched block is the Motion Vector (MV). The commonly used distortion metrics are Sum of Absolute Difference (SAD) or Sum of Square Difference (SSD). The SAD which is denoted as

$$SAD = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} |I_C(x, y) - I_R(x + x_0, y + y_0)| \quad (2.1)$$

is to calculate the difference in absolute values for each pixel data between current block $I_C(x, y)$ and candidate blocks $I_R(x + x_0, y + y_0)$ in the search window. The SSD which is denoted as

$$SSD = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} (I_C(x, y) - I_R(x + x_0, y + y_0))^2 \quad (2.2)$$

is to calculate the difference in squared values for each pixel data. The SSD can provide better prediction results but with higher computational complexity. The SAD which can also provide good prediction results with minor computational complexity is widely used in video coding.

2.1.2 Design Metrics Evaluation

For evaluation of ME designs, there are several important design metrics under consideration. For ME algorithm development, the design metrics for evaluation are: (1) number of operations, (2) quality of the algorithm in terms of PSNR, (3) memory access bandwidth [33]. However, to evaluate an ME hardware architecture, there are 6 major design metrics for consideration as follows [33].

- **Quality (Q):** The quality metric is to evaluate the motion search performance between full search (FS) and fast ME (FME) algorithm. The FME algorithms reduce

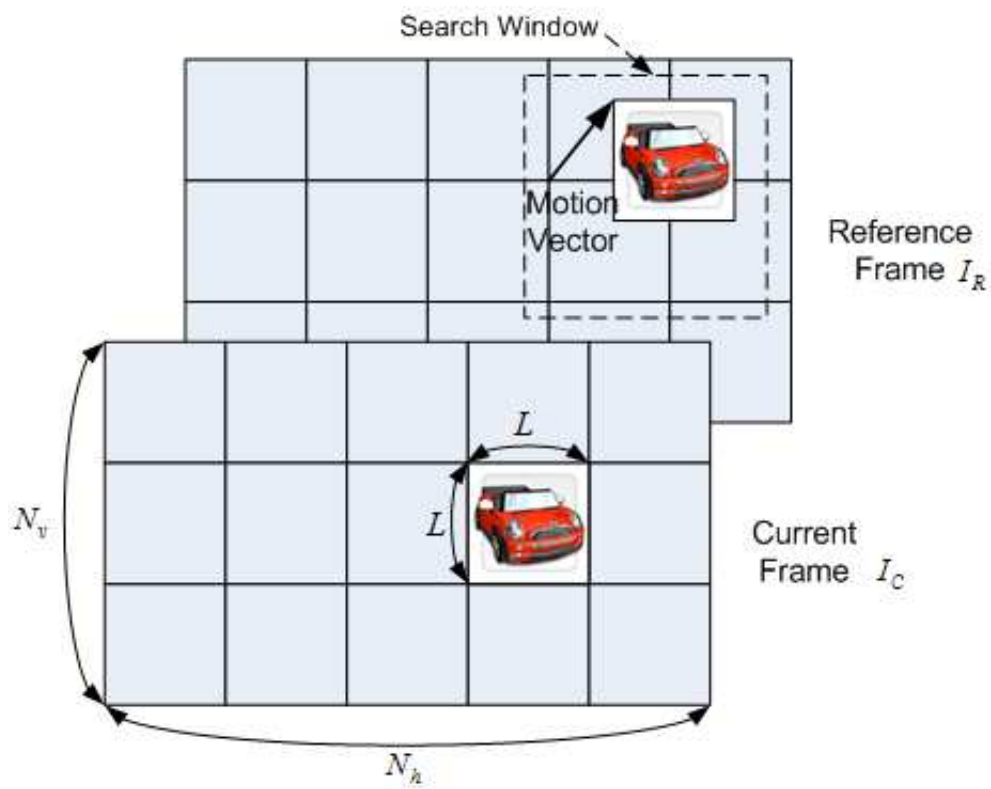


Figure 2.1: Block motion estimation.

the computation complexity by search candidates reduction or matching criterion simplification, etc. However, it suffers quality loss due to the complexity reduction for poor search performance. The quality loss is usually measured by Peak Signal Noise Ratio (PSNR) denoted as

$$PSNR = 10 \cdot \log_{10} \left(\frac{255^2}{\sum_{m=1}^{N_h} \sum_{n=1}^{N_v} (I_C(m, n) - \hat{I}_C(m, n))^2 / (N_h \cdot N_v)} \right) \quad (2.3)$$

where I_C is the current frame, $\hat{I}_C(m, n)$ is the reconstructed image of current frame, N_h and N_v are the frame width and height. Thus, to have a fair comparison of different ME algorithms or hardware architectures, the quality metric (Q) is defined as the PSNR difference between FS and FME denoted as

$$\text{Quality Metric (Q)} = PSNR_{FME} - PSNR_{FS} \text{ (dB)}. \quad (2.4)$$

To apply this metric for fair design evaluations in later sections, the PSNR is measured under search range of [-16, +15].

- **Throughput (T):** The throughput metric is to measure the processing speed of the hardware architecture to see if it can meet the real-time requirement. To quantify the processing speed, the throughput metric is defined as the required number of cycles for one macroblock (MB) of block matching denoted as

$$\text{Throughput Metric (T)} = NC_{MB} \text{ (cycles)} \quad (2.5)$$

, where NC_{MB} represents the required cycles for one MB of block matching. Although the throughput is application dependent, and there is no need to over-design the hardware, the throughput metric is still an important parameter to measure the

hardware design performance. The more cycles the design takes to meet the real-time requirement, the slower the processing speed. If the processing speed is too slow, the applications are limited to those devices target on processing low resolution of video sequences. To apply this metric for fair design evaluations in later sections, the throughput is measured in the number of clock cycles under search range of [-16, +15].

- **Silicon area (A):** The chip size is determined by the hardware silicon area and the VLSI technology. That means the chip size is not available until the chip is really designed. However, we can have a more efficient way to estimate the silicon area on the architectural level by measuring the equivalent gate counts for ME designs. The gate counts includes the number of logic gates for memories and design logics denoted as

$$\text{Area Metric (A)} = A_{\text{Memory}} + A_{\text{Logic}}. \quad (2.6)$$

, where A_{Memory} is the silicon area for memory including Synchronous Random Access Memory (SARM) and/or register files, and A_{Logic} is the silicon area for all the ME hardware logics except memories. In this thesis, we will use gate counts as the silicon area for the evaluation of the ME designs.

- **Hardware Utilization (U):** Hardware utilization or hardware efficiency [33] is to evaluate the hardware utilization ratio in percentage by calculating the active cycles and idle cycles in designs. This metric can also be used to evaluate the design overheads. The higher the utilization ratio, the lower the overheads in the ME hardware designs. That means a design is a highly efficient design if its hardware utilization ratio is high.

$$\text{Utilization Metric (U)} = \frac{NC_{\text{MB}}(\text{Active})}{NC_{\text{MB}}(\text{Active}) + NC_{\text{MB}}(\text{Idle})} = \frac{NC_{\text{MB}}(\text{Active})}{NC_{\text{MB}}} (\%). \quad (2.7)$$

- **I/O Bandwidth (B):** The I/O bandwidth is to evaluate the amount of data transmission between off-chip memories and ME processing core. Since most of the ME designs put frame buffers in off-chip memories, the access to the off-chip memories is unavoidable. Thus, the I/O bandwidth will directly affect the design throughput and hardware utilization ratio. If the bandwidth requirement is high, the ME design will take longer cycles in waiting data before the motion search begins. This will lead to poor throughput and poor hardware utilization since there are many hardware idle cycles existed. In this thesis, the metric for I/O bandwidth evaluation is defined as the number of read cycles and write cycles from off-chip memories denoted as

$$\text{Bandwidth Metric (B}_{\text{I/O}}) = NC_{\text{read}} + NC_{\text{write}} \text{ (bytes)} \quad (2.8)$$

, where NC_{read} and NC_{write} are read and write cycles respectively. The unit is the number of bytes required for the bus access under the search range of [-16, +15].

- **Power consumption (P):** Power consumption is the most critical issue in ME designs, and power constrained designs have wider applications in mobile or portable devices. To evaluate ME designs, the power consumption metric is defined as the total power consumption for memory and hardware logics and is denoted as

$$\text{Power Consumption Metric (P)} = P_{\text{Memory}} + P_{\text{Logic}} \text{ (mW)}. \quad (2.9)$$

, where P_{Memory} is the power consumption for memory, and P_{Logic} is the power consumption for the ME hardware logics. In this thesis, the ME designs is evaluated

using this power consumption metric with the unit of milli-watt (mW). To provide a fair comparison basis, we use the normalized power consumption [26] by mapping the original power consumption to the equivalent power consumption for 0.18 μm denoted as

$$\text{NormalizedPower Consumption Metric (P}_{\text{nom}}) = P \times \frac{0.18^2}{\text{Process}^2} \times \frac{1.8^2}{\text{Voltage}^2}. \quad (2.10)$$

2.1.3 Motion Estimation Hardware Design

Under the consideration of the design metrics in Section 2.1.2, we are able implement the block ME algorithm in Section 2.1.1 as an effective low power or power adaptive ME hardware designs. Fig. 2.2 shows the functional blocks of a generic video encoder hardware architecture with the ME module. The encoder architecture contains a RISC CPU to control the data and command flow, External Memory Interface (EMI) to access current and reference frames which are put in external memories, dedicated co-processors including ME module for processing acceleration.

Fig. 2.3 has detail views to the architecture of the ME module. This ME architecture contains several local memories to store current and reference search window data, a ME core for motion search, a control unit for data and search flow control, and the decision unit to determine the final motion vectors. Firstly, The current and reference data for motion search are received via memory interface (MEM_IF) and then stored in local memories (LM_CUR and LM_REF). After the data are ready in local memories, the controller sends commands to get data from local memories, and sends commands to ME core starts the motion search. The block matching results are sent to decision engine for final motion

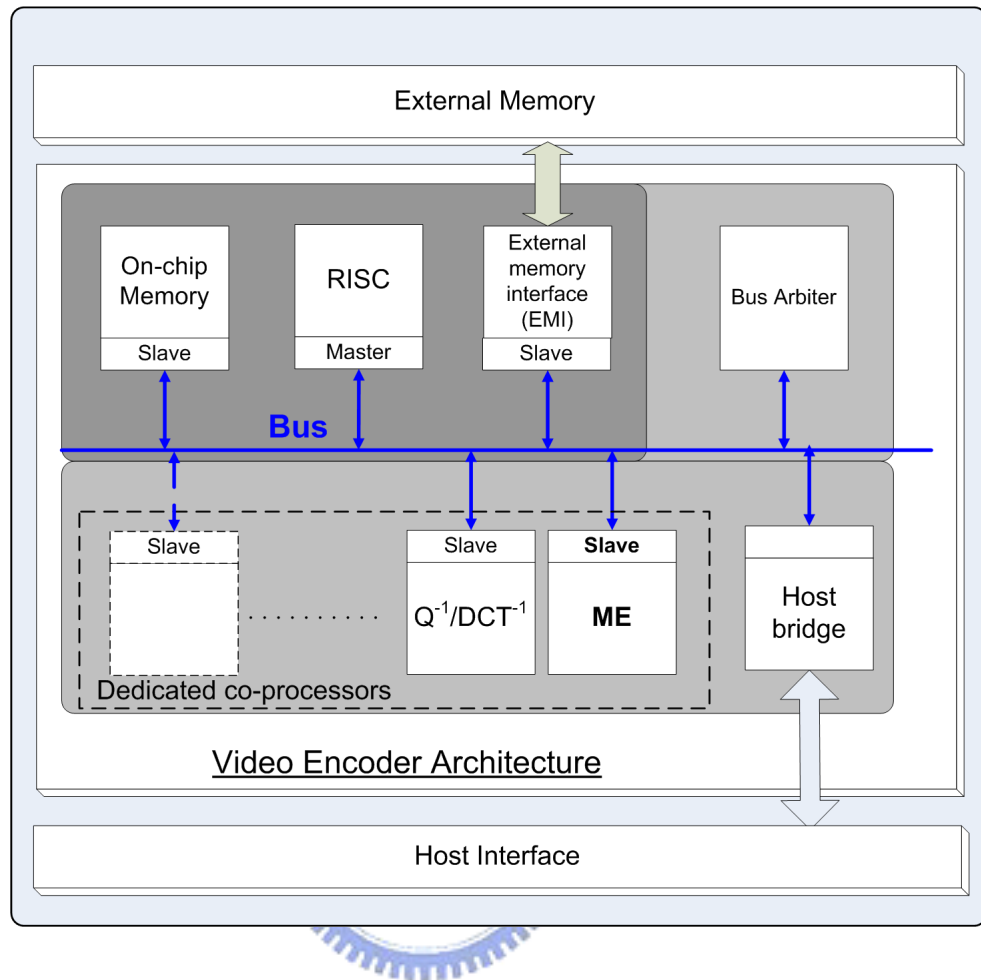


Figure 2.2: Video encoder hardware architecture with ME module.

vector determination.

2.1.4 Memory Hierarchy

In the ME hardware design, memory hierarchy is equally important as designing the motion estimation core. The motion estimation core determines the overall design throughput (T). However, the memory hierarchy architecture is related to the I/O bandwidth ($B_{I/O}$), local memory sizes (A_{Memory}), memory access power (P_{Memory}), and hardware utilization

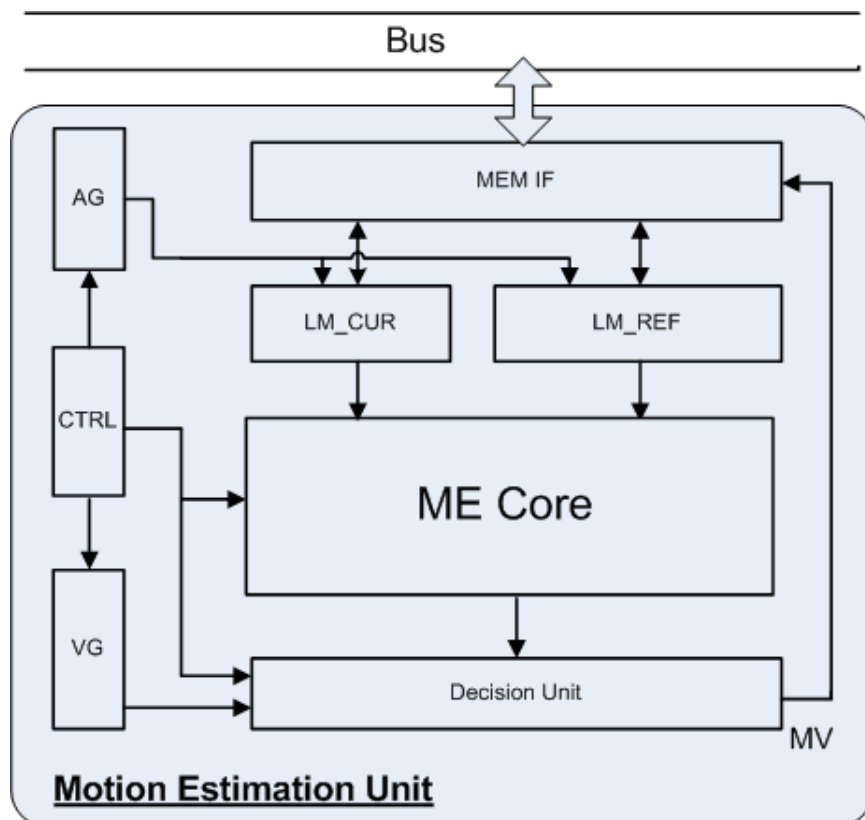


Figure 2.3: Motion estimation hardware architecture.

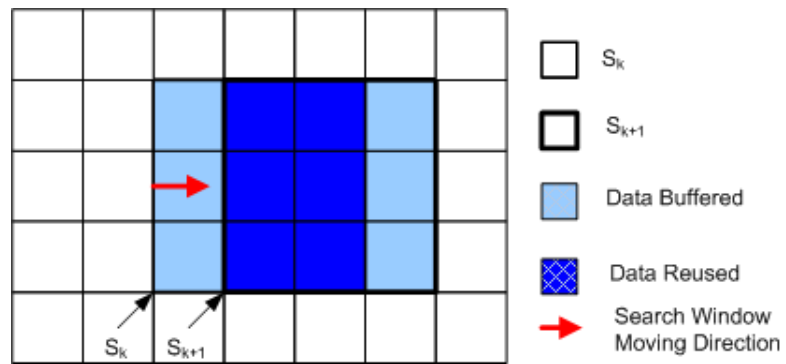
(U).

The memory hierarchy in ME hardware design is related to the data reuse of the search window [42, 16, 17]. The data reuse approaches can be roughly categorized into 4 levels (Level A to D) according to the reuse percentage from low to high [16]. In the 4 levels, the most commonly used approaches are Level C and Level D. They are depicted as below.

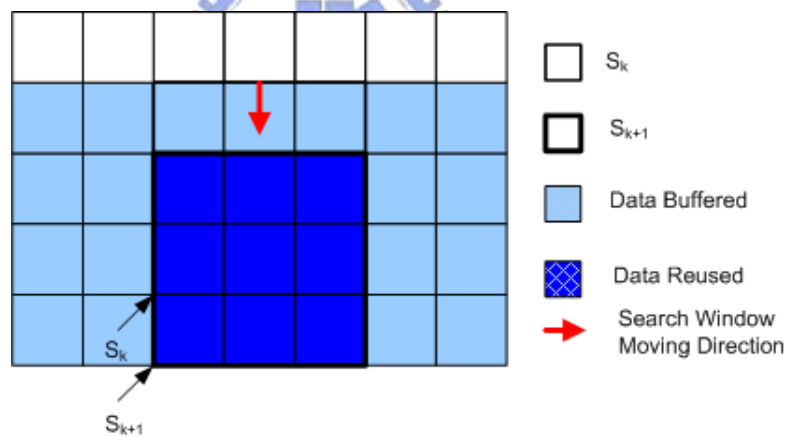
- **Level C:** The Level C approach is to buffer the search window data in the same MB row as the current MB. As shown in Fig. 2.4(a), suppose current block size is L and the search range is $[-SR, +(SR-1)]$, the data reuse ratio is $Ra = \frac{L \times (L + (SR \cdot 2))}{L \times L} = 1 + \frac{(SR \cdot 2)}{L}$. The required memory buffer size is $S = (L + (SR \cdot 2))^2$.
- **Level D:** The Level D approach is to buffer the whole MB row data in the same MB row as the current MB. As shown in Fig. 2.4(b), suppose the image width is N_h , the data reuse ratio is $Ra = \left(\frac{L \times L}{L \times L}\right) = 1$. The required memory buffer size is $S = (L + SR) \times N_h$.

Table 2.1 summarizes the required memory size and data reuse ratio under different search range and frame resolution for Level C and Level D schemes. Level D has better data reuse ratio, but requires larger memory buffers as compared to Level C. Level C is more commonly used in the application with larger frame sizes such as High-Definition Television (HDTV) to avoid huge memory requirements with median data reuse ratio.

In the following sections, we will survey the prior low power ME works in Section 2.2 and power adaptive ME works in Section 2.3. Then, these works are evaluated according to the design metrics depicted in Section 2.1.2.



(a)



(b)

Figure 2.4: Search window data reuse. (a)Level C (b)Level D.

Table 2.1: Required memory size and data reuse ratio for data reuse schemes of Level C and Level D.

Resolution	SR	Level C		Level D	
		Memory (bytes)	Ra	Memory (bytes)	Ra
CIF 352×288	16	2304	3	11264	1
	32	6400	5	16896	1
	64	20736	9	28160	1
D1 720×480	16	2304	3	23040	1
	32	6400	5	34560	1
	64	20736	9	57600	1
HDTV 720P 1280×720	16	2304	3	40960	1
	32	6400	5	61440	1
	64	20736	9	102400	1

2.2 Low Power Motion Estimation Designs

The low power ME designs from 1995 to 2007 are surveyed and categorized into 3 groups according to their design approaches. The first group is to achieve low power by fast ME algorithms [19, 36, 37, 38, 42, 20, 21, 26]. This group of designs apply fast ME algorithms such as three-step search (TSS), hierarchical search, etc. which reduce the search candidates for computational power reduction. The second group is to achieve low power by simplified block matching criterion [34, 35, 49, 52]. The commonly used block matching criterion is SAD (eqn. 2.1) or SSD (eqn. 2.2). Although they can provide good R-D performance, it takes lots of computational power. For power reduction, the approaches such as using Most Significant Bits (MSB) only for block matching, or pel-subsampling which takes partial pixel data in that block for block matching, etc. are able to reduce computational power of the block matching operations for power saving. The third group of designs is to archive low power by efficient hardware architectures [18, 39, 40, 22, 23, 24, 25, 26, 17, 27, 29]. For example, the one-dimensional (1-D) systolic array for

Table 2.2: Evaluation of low power designs using design metrics.

Groups	Q	T	A	U	B	P
Reduced candidates	X	O	-	-	-	O
Simplified Matching Criteria	X	O	-	-	-	O
Efficient Architectures	-	O	-	-	-	O

O: improved

X: degraded

-: case dependent

full search or an efficient memory hierarchy architecture can effectively reduce the power consumption. Table 2.2 shows the influences to the design metrics in Section 2.1.2 by using these three design approaches for low power hardware designs.

In the following, we will introduce the frequently cited low power design works, and summarize their design metrics evaluation in Table 2.3 and Table 2.4. These works are also used as the reference to the proposed BBME design in later chapters.

2.2.1 Low Power by Fast ME Algorithms

Miyama *et al.* [36] proposed a sub-mW motion estimation processor core by developing a Gradient Descent Search (GDS) algorithm with the optimized hardware architecture for mobile applications. The GDS algorithm is to reduce the required computational complexity and hardware operational cycles for ME. The Single Instruction Multiple Data (SIMD) data path is to reduce the required clock frequency by maximizing the parallel processing ability. The three-port SRAM acts as the data cache to reduce the power consumption. These features make this hardware core to be able to run QCIF 15fps at 0.85 MHz with 0.4 mW power consumption. The hexagon plot is shown in Fig. 2.5(a).

Chao *et al.* [30] proposed a hybrid motion estimation hardware architecture to support

Successive Elimination Algorithm (SEA) and Diamond Search (DS). The irregular flow between the two fast algorithms are solved to achieve different applications for high quality and low power. This design has 3 modes including: (1) SEA without early cut, (2) SEA with early cut (at cycle 4208 to meet CIF 30fps at 50MHz), (3) DS without early cut. Running on the third mode, the power consumption is 223.6 mW for CIF 30 fps with 50MHz clock frequency. The hexagon plot is shown in Fig. 2.5(b).

Chen *et al.* [26] proposed a an optimal low power IME engine with a parallel hardware architecture supporting fast algorithms and efficient data reuse (DR) called content adaptive parallel-VBS 4SS. This design has 3 modes to achieve different video quality and power consumption. These 3 modes are: (1)high quality mode, (2)low power mode, and (3) ultra low power mode. The first mode is with 2 reference frame and multiple iterations to achieve high quality. The second mode is with 1 reference frame and multiple iterations to achieve minor quality loss and low power consumption. The third mode is with one reference and single iteration to achieve ultra low power consumption. Running on the third mode, the power consumption is 2.13 mW for CIF 30 fps with 13.5 MHz clock frequency. The hexagon plot is shown in Fig. 2.6(a).

2.2.2 Low Power by Simplified Block Matching Criteria

Huang *et al.* [37] proposed a new block matching algorithm called Global Elimination Algorithm (GEA) and its optimized architecture to achieve the low power design. The GEA is developed from Successive Elimination Algorithm (SEA), but saves more SAD computations by calculating sub-sampled pixel data for early terminations. The early termination can save more unnecessary power consumption for SAD computations. This hardware de-

sign can achieve more than CIF 30 fps at 25 MHz with 189 mW power consumption. The hexagon plot is shown in Fig. 2.6(b).

Wang *et al.* [35] proposed a low power ME design by implementing All Binary Motion Estimation (ABME) algorithm and proposing an optimized hardware architecture for the binary bitplane of block matching. The images for search are firstly formatted as binary bitplane, and the block matching criterion is modified to use the binary data for pattern matching. The pattern matching using binary data can greatly reduce the computational complexity, thus the power consumption is saved. The power consumption for CIF 30fps is 2.2mW. The hexagon plot is shown in Fig. 2.7(a).

2.2.3 Low Power by Efficient Hardware Architecture

Shen *et al.* [39] proposed a low-power full-search block matching (FSBM) motion-estimation design for H.263+. To minimize power consumption, techniques such as gated-clock and dual-supply voltages are used. This design runs CIF 36fps at 60 MHz, and the power consumption is 423.8 mW. The hexagon plot is shown in Fig. 2.7(b).

Chen *et al.* [66] proposed an parallel-SAD tree with a shared reference buffer for H.264 integer motion estimation (IME). To solve the huge memory bandwidth required by H.264 IME, an efficient memory architecture is proposed to save 99.9% off-chip memory bandwidth and 99.22% on-chip memory bandwidth. This design can run 720P 30fps solution at 108 MHz with 330.2k gate count and 208k bits on-chip memory.

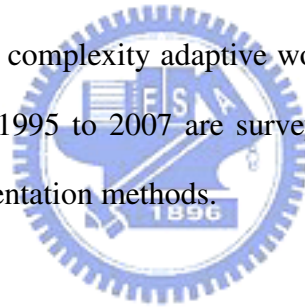
Yap *et al.* [40] proposed a new 1-D VLSI architecture for H.264 IME. The SAD computation is performed by reusing the results of smaller sub-block computations to save the computations and power. They are combined with a shuffling mechanism within each pro-

cessing element to process up to 41 MV sub-blocks in the same number of clock cycles. The design supports CIF 191fps processing rate with 294 MHz, and the power consumption is 0.008 mW/MB/fps. The hexagon plot is shown in Fig. 2.8(a).

Ou *et al.* [67] proposed a new 2-D VLSI architecture for H.264 IME. Using the 2-D systolic array architecture, this design is able to complete a MB search in 256 cycles with 100% PE utilization. The power consumption is 20.48mW. The hexagon plot is shown in Fig. 2.8(b).

2.3 Power Adaptive Motion Estimation Designs

The power adaptive or complexity adaptive works [44, 45, 46, 47, 48, 56, 49, 50, 57, 51, 52, 58, 30, 26] from 1995 to 2007 are surveyed, and categorized into three groups according to their implementation methods.



2.3.1 Power Adaptive by Fast Algorithms

The first group is to achieve different computational complexity or power consumption by switching between different fast ME algorithms [44, 45, 46, 47, 48, 30, 26]. Different fast ME algorithms contain different candidate locations for search so as to deliver different complexity level. The commonly used fast ME algorithms include full search (FS), three-step search (TSS), sub-sampling search (SUB), diamond search, etc. For search range of [-16, +15], FS needs to check 1024 candidates, but TSS only checks 25 candidates with around 40 times accelerations.

For hardware design, this kind of method has implementation difficulties in switching

Table 2.3: Evaluation of design metrics for low power designs (Group 1 and 2).

Designs	Design Metrics	
Miyama [36]	Q (dB)	-0.10
	T (cycles/MB)	568 ¹
	A (logic,memory) ²	250/40
	U (%)	n.a. ³
	B (bytes/MB)	1024 ⁴
	P (mW, μ m)	2.5/0.13
	P _{nom} (mW)	15.53 ⁵
Chao [30]	Q (dB)	-0.00/-0.01/-0.72 (mode 1-3)
	T (cycles/MB)	5886/2879/437 (mode 1-3) ⁶
	A (logic,memory)	69/36
	U (%)	n.a.
	B (bytes/MB)	1024
	P (mW, μ m)	223.6/0.35
	P _{nom} (mW)	17.60
Chen [26]	Q (dB)	-0.00/-0.07/-0.55 (mode 1-3)
	T (cycles/MB)	1136 (mode 3) ⁷
	A (logic,memory)	131.2/64
	U (%)	n.a.
	B (bytes/MB)	1024 (mode 1), 256 (mode 2,3)
	P (mW, μ m)	16.72,4.83,2.13/0.18 (mode 1-3)
	P _{nom} (mW)	4.08 ⁸
Huang [37]	Q (dB)	-0.08
	T (cycles/MB)	1784 ⁹
	A (logic,memory)	89.39/24.08
	U (%)	n.a.
	B (bytes/MB)	1024
	P (mW, μ m)	160/0.35
	P _{nom} (mW)	12.59
Wang [35]	Q (dB)	-0.19
	T (cycles/MB)	283
	A (logic,memory)	68.5/9.80
	U (%)	n.a.
	B (bytes/MB)	1086 ¹⁰
	P (mW, μ m)	2.2/0.18
	P _{nom} (mW)	2.20

¹CIF 30fps@6.75MHz.²kilo gates/kilo bits³Not available.⁴Memroy hierarchy architecture :Level C (=16 \times 16 + 16 \times 48).⁵Voltage=1.0V.⁶The average case.⁷CIF 30fps@13.5MHz.⁸Voltage=1.3V.⁹CIF 30fps@21.2MHz.¹⁰32 \times 30 + (16 \times 48 + 8 \times 24 + 4 \times 12)/8 = 960 + 126 = 1086

Table 2.4: Evaluation of design metrics for low power designs (Group 3).

Designs	Design Metrics	
Shen [39]	Q (dB)	-0.00
	T (cycles/MB)	4209 ¹
	A (logic,memory)	66.8/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	353/0.60
	P _{nom} (mW)	4.12
Chen [66]	Q (dB)	n.a.
	T (cycles/MB)	1000
	A (logic,memory)	330.2/208
	U (%)	100
	B (bytes/MB)	1024
	P (mW, μ m)	n.a./0.18
	P _{nom} (mW)	n.a.
Yap [40]	Q (dB)	-0.00
	T (cycles/MB)	4096
	A (logic,memory)	61/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	95.04/0.13
	P _{nom} (mW)	409.97
Ou [67]	Q (dB)	-0.00
	T (cycles/MB)	1024
	A (logic,memory)	597/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	20.48/0.18
	P _{nom} (mW)	20.48

¹CIF 36fps@60MHz.

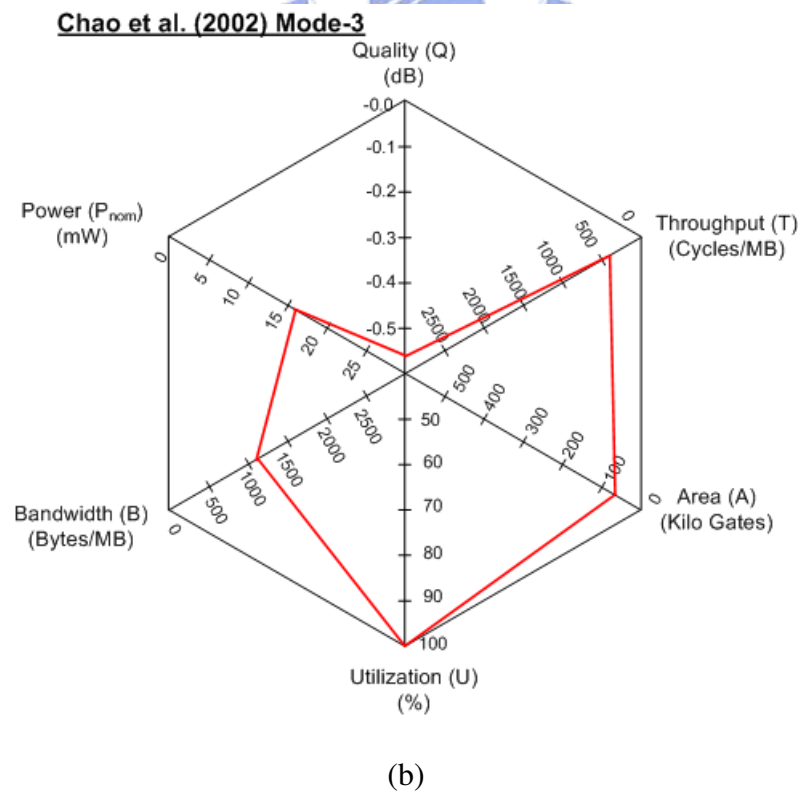
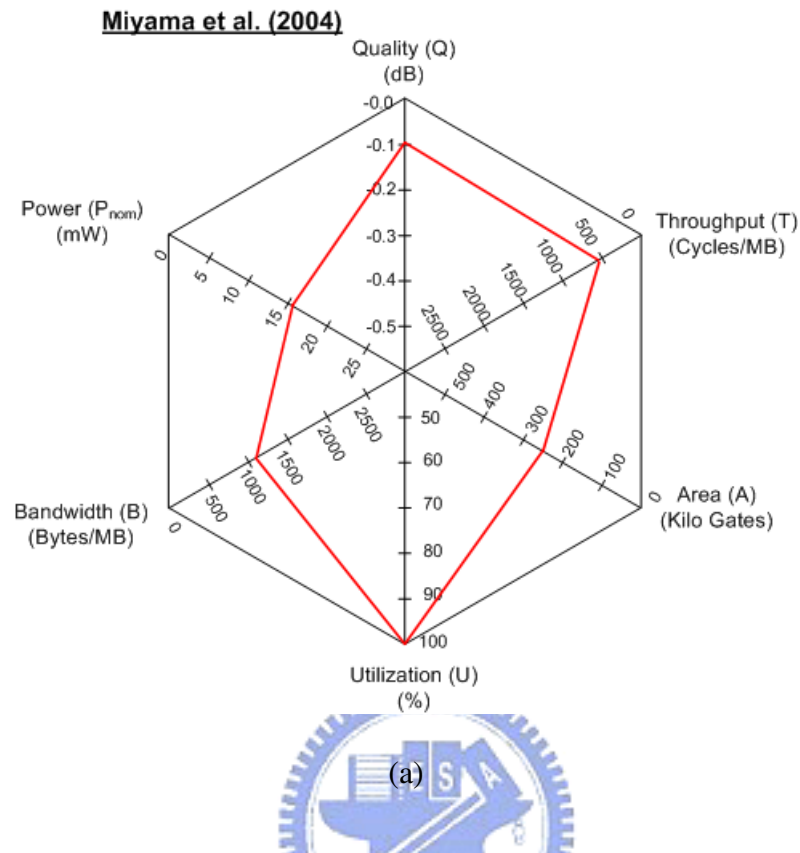


Figure 2.5: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Miyama's work [36] (b) Chao's work [30].

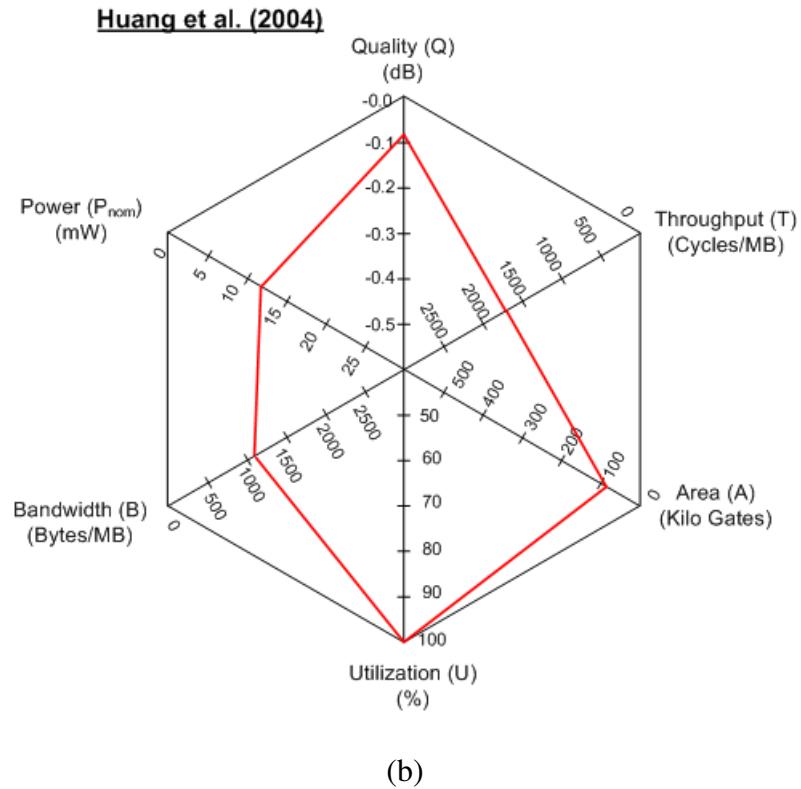
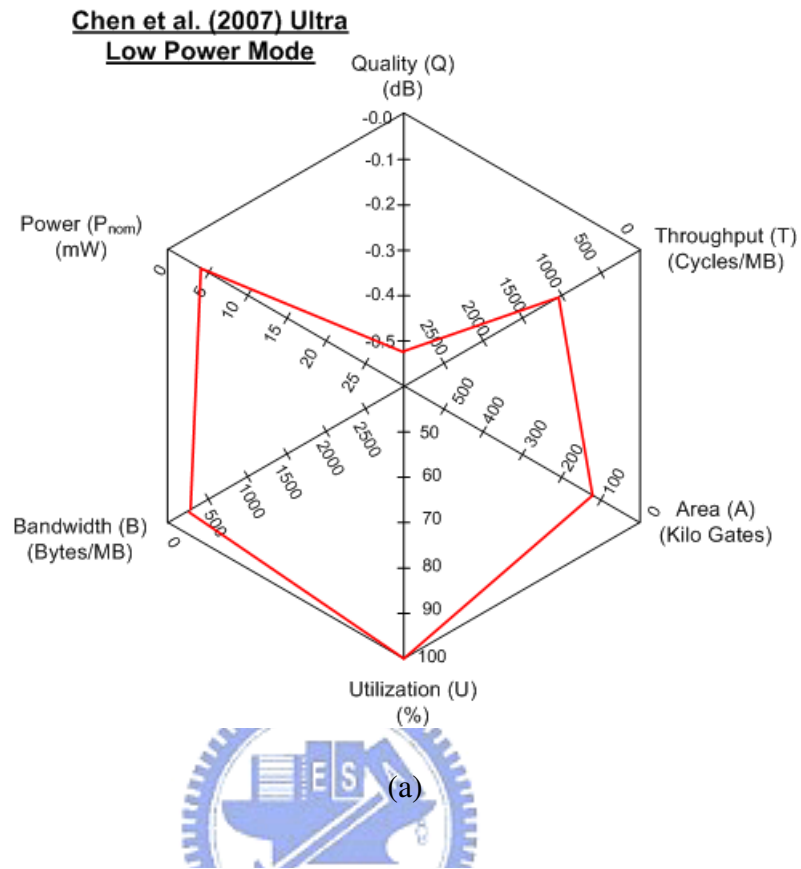


Figure 2.6: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Chen’s work [26] (b) Huang’s work [37].

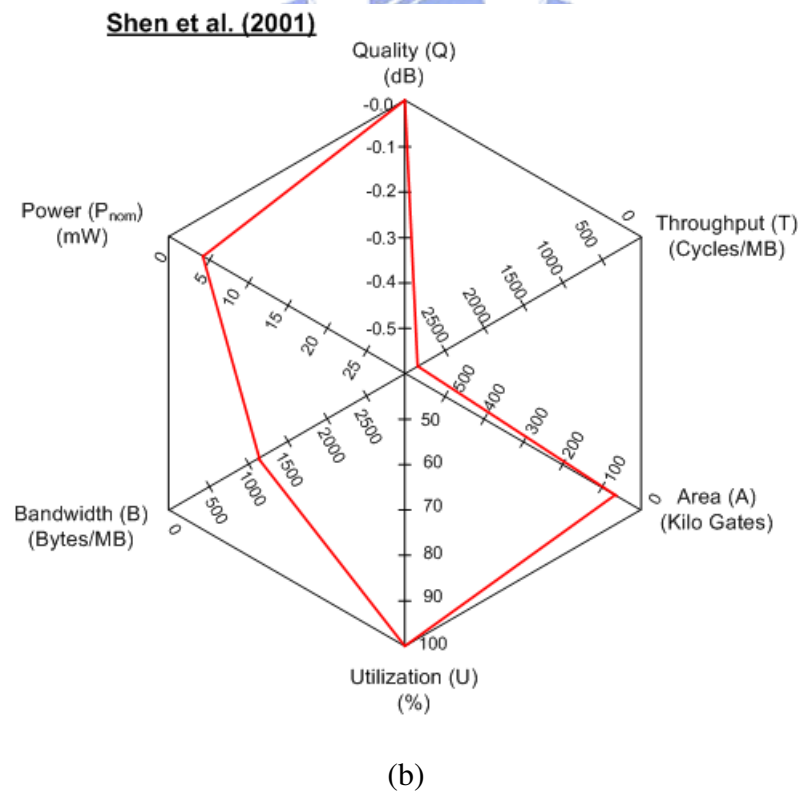
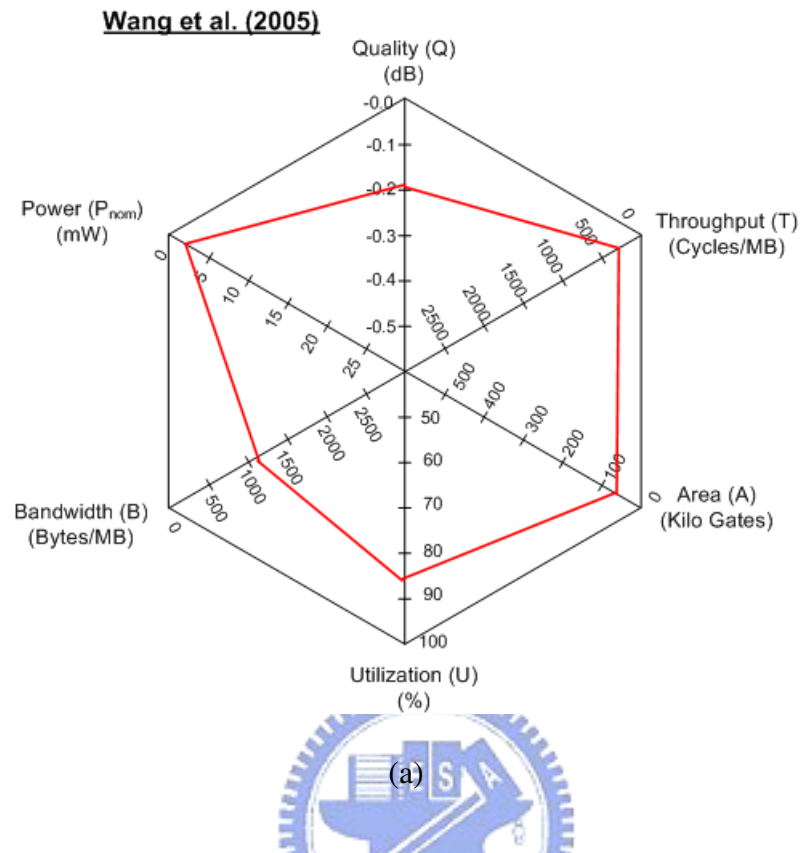


Figure 2.7: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Wang's work [35] (b) Shen's work [39].

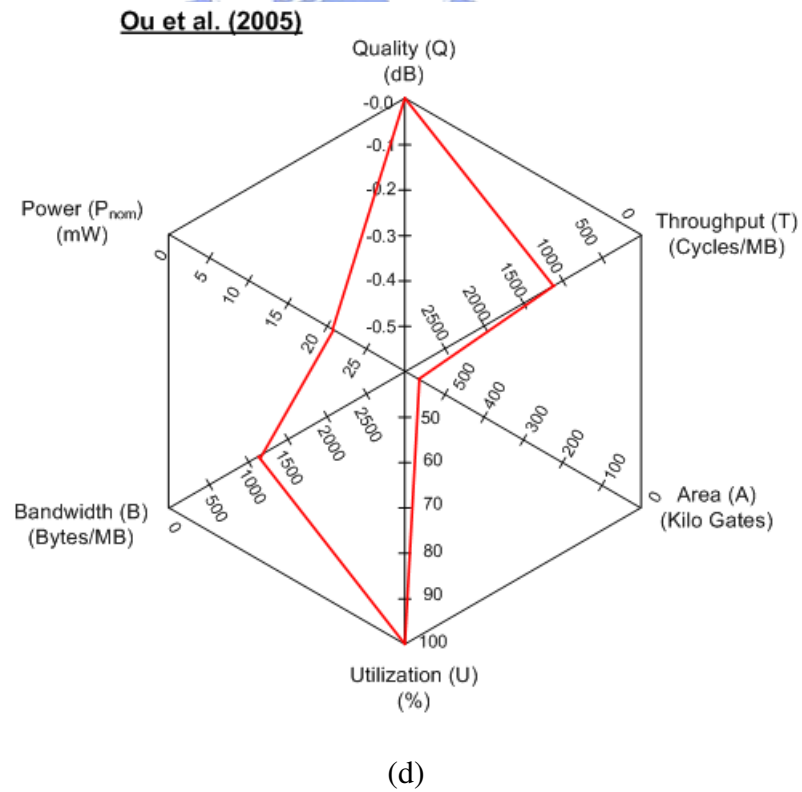
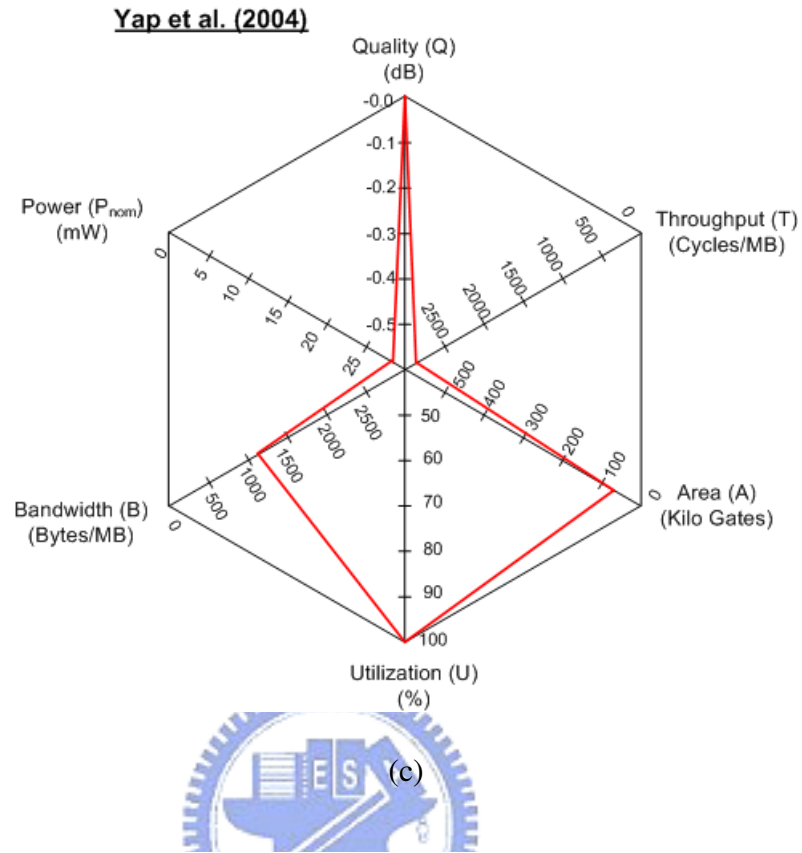


Figure 2.8: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67].

between different fast ME algorithms which may cause serious design penalties in hardware pipelining and memory access. To solve this problem, the solution is to use ME algorithms with similar search strategies such as TSS and 4SS. For example, a configurable architecture in [48] is proposed to allow flexible switching between Predictive Motion Vector Filled Adaptive Search Technique (PMVFAST) [53] and Enhanced Predictive Zonal Search (EPZS) [54].

2.3.2 Power Adaptive by Pixel Number for Block Matching

The second group is to achieve power adaptation by using different pixel number for block matching [56, 49, 50]. In conventional block matching formula such as SAD (eqn. 2.1) or SSD (eqn. 2.1), it needs 256 subtraction and 255 addition operations to complete a 16×16 SAD calculation. However, this type of method is to dynamically adjust the number of pixels for SAD calculations. For example, a commonly used algorithm is $n : 1$ sub-sampling search, which only uses $1/n$ pixels for the SAD calculations. With $n : 1$ sub-sampling search, $\frac{n-1}{n}$ subtraction and addition operations can be saved for power consumption reduction.

To achieve a better switching, Pre-analyze the video content can provide good information about the motion activities for current MB search. For low motion activities, only a few pixels for block matching can provide similar R-D performance as the full search. However, for complex motion activities, we need more data pixels to provide more detail block information for better R-D performance. One example is [56], the authors explore several feature/edge extraction methods as the masking criterion to decide the parameter for $n : 1$ sub-sampling search for power adaptation.

2.3.3 Power Adaptive by Pixel Bit Precision

The third group is to achieve power adaptation by using different pixel bit precision for block matching [57, 51, 52, 58]. The conventional block matching criterion is to use 8 bits for SAD calculations. However, similar to group 2, we can use fewer bits for low motion activities and more bits for complex motion activities to achieve similar R-D performance as FS.

A commonly used method is least significant bit (LSB) truncation which keeps the MSBs only for SAD calculation. Such a method doesn't benefit too much in software implementation for general purpose processors, but gains more in dedicated hardware designs. The hardware architecture can provide more efficient processing to the dedicated binary format SAD calculations. One example is [57], the authors use three control lines to mask the LSBs to have different bit precision for SAD calculations. However, according to the prior works, using single bit of MSBs for SAD calculation usually leads to poor R-D performance [57, 52, 58]. To improve this problem, the image preprocessing skills are used to find the image features, and reconstruct it as the new MSB-LSB structure for SAD calculation with reduced pixel bit precision. One example is [51], a complexity adaptive algorithm is proposed by extracting the image features as the new MSB-LSB structure and use the results of the video content analysis to decide the bit precisions for SAD calculation. Such a method can improve the serious PSNR loss in conventional LSB truncation methods.

Table 2.5: Summary of three groups of complexity adaptive ME works.

Type	Features	Pro and Con
Fast Algorithms	<ul style="list-style-type: none"> • Switch search ranges or fast algorithms 	<ul style="list-style-type: none"> • Pro: More flexible for power adaptation. • Con: different hardware integration.
Reduced Pixel Number	<ul style="list-style-type: none"> • Switch pixel number for block matching. 	<ul style="list-style-type: none"> • Pro: Simple hardware implementation. • Con: Hardware masking causes idling.
Reduced Bit Precision	<ul style="list-style-type: none"> • Switch bit precision for block matching 	<ul style="list-style-type: none"> • Pro: Simple hardware implementation. • Con: Hardware masking causes idling.

2.3.4 Summary of Power Adaptive Design Schemes

Table 2.5 summarizes the pros and cons for the aforementioned three groups of complexity adaptive or power adaptive methods. From this table, the first group is more flexible for power adaptation but is difficult in hardware integration of different algorithms. The second and third groups are simpler in hardware implementation, but the adopted hardware masking approach usually causes serious hardware idling problem and the overhead for power adaptation.

Chapter 3

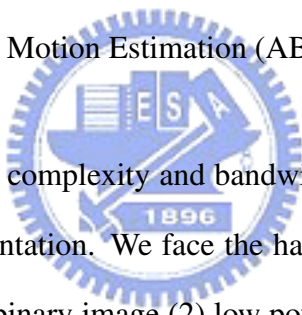
Bi-directional Binary Motion Estimation (BBME)

3.1 Introduction



Motion estimation (ME) is the most computationally expensive module in multimedia compression standards such as MPEG-1/2/4 and H.26x. For portable video applications, low power and efficient bus access are two major design goals [42]. The huge power consumption from ME needs to be reduced to extend battery life. The ME is also a data intensive module. It moves huge data for block matching to find the best predicted block with minimal distortion for motion compensation. The amount of data movement increases proportionally to the square of search range, and becomes the performance bottleneck for System-on-Chip (SoC) designs due to the limited available bus bandwidth. Hence, an efficient data movement scheme via bus is another key design issue for portable video applications.

To investigate low power ME designs, most of them are developed based on fast search algorithms [34, 35, 36, 37, 38] or low cost full search architectures [39, 40]. These fast algorithms include three-step search (TSS) [38], gradient descent search (GDS) [36], global elimination search [37], binary search [34, 35], etc. For low power or low cost full search architectures, one-dimensional (1-D) systolic array is the most widely used architecture [33]. Among these designs, binary search [34, 35] has both advantages of low computational complexity and low bus bandwidth requirement. The reason is it reduces the pixel precision from eight bits to be one bit for block matching. Such a search strategy can also be viewed as a kind of feature matching with binary images. Therefore, in this paper, we develop our low power and bandwidth efficient ME design based on a binary pyramid search algorithm called All Binary Motion Estimation (ABME) [41].



Although ABME [41] is a low complexity and bandwidth efficient algorithm, it is not well optimized for VLSI implementation. We face the hardware design challenges in (1) image pre-processing to form the binary image (2) low power and bus bandwidth efficient architecture for binary pyramid search (3) support of bi-directional (or called B-frame) and 8x8 block searches. We solve these design issues in the proposed hardware architecture. Firstly, we propose a new pre-processing flow for binary image generation at the macroblock (MB) level instead of the original frame level. The binary image generation and binary motion search are integrated as an MB level pipelining to simplify redundant bus access. Secondly, we re-examine the data flow in the three levels of binary pyramid search structure, and modify the algorithm to remove the data dependency and inefficient operations for the second level of search. Finally, we address the design issues in B-frame search scheme and optimize the hardware architecture to enhance the processing throughput.

The contributions of this work include:

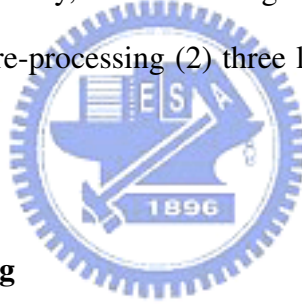
- Modified ABME algorithm called bi-directional binary motion estimation (BBME) for efficient VLSI implementation. The BBME algorithm is developed based on a low complexity ABME algorithm [41]. We modify the ABME in the binary image generation and search method for efficient VLSI implementation. The power consumption for video encoding with CIF 30 fps and search range of [-16, +15] only needs less than 1 mW.
- MB level pipelining architecture for efficient bus access. We propose a new pre-processing flow for binary image generation at MB level as opposed to frame level. The new processing flow integrates both binary image generation and binary motion search using MB level pipelining to avoid redundant bus access. The bus bandwidth saving can achieve up to 67.1%.
- Parallel B-frame search architecture. We design our hardware to be able to handle B-frame search in parallel. It reuses the same current search data to save on-chip memory access and power. Thanks to the simple binary image matching, the gate counts have increased twice but not as much as the conventional 8-bit designs.

The remainder of this paper is organized as follows. Section 3.2 depicts the design issues in ABME algorithms. According to the design issues described in Section 3.2, a modified algorithm is developed to address these issues in Section 3.3 and the hardware architecture based on the modified algorithm is proposed in Section 3.4. In Section 3.5, experiments show the improved performance in power consumption and bus bandwidth loading. Section 3.6 gives the summary of this chapter.

3.2 Problem Statement

3.2.1 Review of ABME Algorithm

Fig. 3.2 shows the functional block diagram of a generic video encoder system with ABME algorithm [41]. The current video frame is processed as binary image in the pre-processing module, and is stored back into binary frame buffer as reference picture for next video frame. The binary ME module accesses the current and reference data from pre-processing module and binary frame buffer respectively to start the binary search. At the end of search, it outputs the final motion vectors for entropy encoding in VLC and for motion compensation in MC. Basically, the ABME algorithm is composed of two major components: (1) frame level of pre-processing (2) three levels of binary pyramid search. They are described as below.



A. Frame Level of Pre-processing

The frame level of pre-processing is used to generate binary images for the current block in frame level. Fig. 3.3 shows the generation flow of the binary images for block matching in the pre-processing module. The original image $I_{M \times N}$ is binarized as $\hat{I}_{M \times N}$ (referred to as LV3). Then, the original image $I_{M \times N}$ is down-sampled by 2 to be $I_{\frac{M}{2} \times \frac{N}{2}}$ that is a quarter size of the original image. The second level of binary image is generated after applying binarization (BIN) to $I_{\frac{M}{2} \times \frac{N}{2}}$ to be $\hat{I}_{\frac{M}{2} \times \frac{N}{2}}$ (referred to as LV2). Following the same down-sampling and binarization process, a one-sixteenth resolution of the binary image $\hat{I}_{\frac{M}{4} \times \frac{N}{4}}$ (referred to as LV1) is generated.

The binarization process contains two operational steps: (1) filtering, and (2) compara-

tor. The filtering operation is to calculate a low-pass or high-pass filtered data for comparator to produce the binary pattern. The filter adopted in [41] is a 3×3 two dimensional filter H_A as shown in

$$H_A = \frac{1}{4} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (3.1)$$

The filtered data $I_{M \times N}^F(x, y)$ by H_A is denoted as

$$I_{M \times N}^F(x, y) = H_A(I_{M \times N}(x, y)) = \left(\sum_{i=1,-1} \sum_{j=1,-1} (I_{M \times N}(x+i, y+j)) + 1 \right) \gg 2 \quad (3.2)$$

, where 1 is for rounding control. The comparator operation then compares the filtered data and original data to construct the three levels of binary pyramid data as shown in

$$\hat{I}_{M \times N}(x, y) = \begin{cases} 1 & \text{if } I_{M \times N}(x, y) \geq I_{M \times N}^F(x, y) \\ 0 & \text{otherwise} \end{cases}. \quad (3.3)$$

B. Three Levels of Binary Pyramid Search

The ABME algorithm adopts a three-level of binary pyramid search structure in which LV1 is the top level of search, LV2 is the middle level of search, and LV3 is the bottom level of search. Similar to the conventional 8-bit pyramid search structure, the search starts from LV1 to LV3 sequentially.

Fig. 3.4(a) shows the three-level binary pyramid search for ABME. The LV1 search is a small range of binary full search with search range of $[-SR/4, +SR/4-1]$, and the final motion vectors are called MV_{LV1} . The LV2 search contains multiple candidates search. As shown in Fig. 3.1, the LV2 search checks 6 candidates from motion predictors for top MB (MV_{top}), left MB (MV_{left}), top right MB ($MV_{topright}$), previous co-located MB ($MV_{co-located}$), the final MV from LV1 (MV_{LV1}) and zero MV at (0,0) (MV_{zero}). If the 6

candidates are all zero MVs, then a ± 2 cross pattern of search is started from (0,0). If not, a ± 1 cross pattern of search is started from the one of the 6 MV predictors with minimal block matching distortion. The final motion vector for LV2 search is called MV_{LV2} . The LV3 search starts a ± 2 full search from the center of MV_{LV2} , and the final motion vectors with minimal distortion is MV_{LV3} .

For the block matching criterion, due to the data for search are all in binary format, the matching criterion can be simplified. In the formula of sum absolute difference (SAD) as shown in eqn. (3.4), the subtraction and addition operations are all 8-bit operations. However, if the data for search is binary format, the results for the subtraction operations between current and reference search window data are equal to the results using exclusive OR (XOR) operations, but XOR operation is simpler. Hence, the SAD matching criterion can be simplified to sum of difference (SOD) as in eqn. (3.5) with the same results.

$$SAD = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} |I_C(x, y) - I_R(x + x_0, y + y_0)| \quad (3.4)$$

$$SOD = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} |\hat{I}_C(x, y) \oplus \hat{I}_R(x + x_0, y + y_0)| \quad (3.5)$$

, where the symbol \hat{I}_C is the current binary frame, the symbol \hat{I}_R is the reference binary frame, the symbol \oplus denotes the XOR operation, L is 16 for LV3, 8 for LV2 and 4 for LV1.

3.2.2 Design Issues of ABME Algorithm

Although ABME [41] is a low complexity and bandwidth efficient algorithm, it is not well optimized for hardware implementation. We need to adapt it to address the following issues: (1) pre-processing for binary image generation before the search (2) sequential LV2

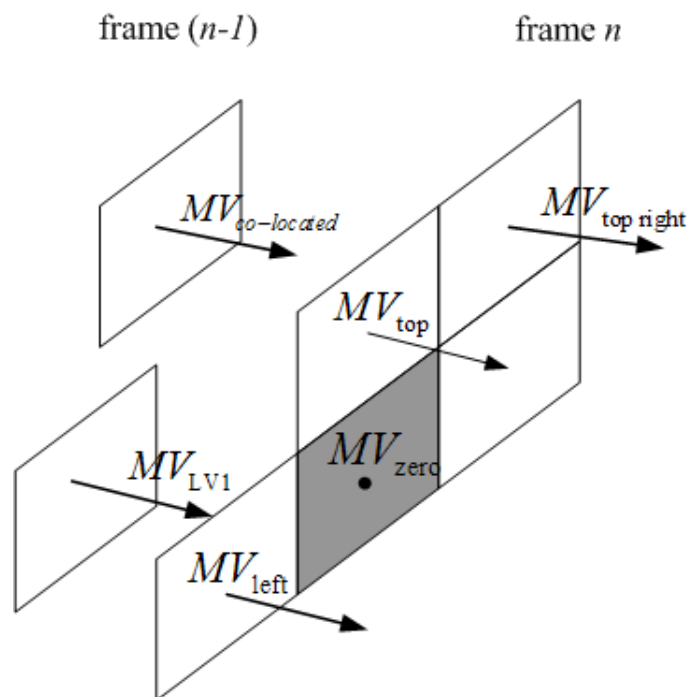


Figure 3.1: Motion predictors for LV2 search.

binary pyramid search that makes it difficult to achieve low power and efficient on-chip memory access (3) support of B-frame and 8x8 block searches.

A. Pre-processing for Binary Images Generation

The ABME has a pyramid based search structure that requires a pre-processing unit (PPU) to generate three different resolutions of binary images for search. For most pyramid based designs, the pre-processing module is implemented outside the ME unit (MEU) because of the longer cycles to each pipeline stage. The longer cycles for each pipeline stage will cause longer latency to system pipelining and other hardware modules.

As shown in Table 3.1, we analyzed 3 design schemes for the PPU module. Scheme 1 is the conventional 8-bit pyramid search, and such a design scheme puts PPU out of MEU to avoid longer latency and redundant bus transmission. However, ABME is a binary format of pyramid search which has different influences to the cycles in one pipeline stage (NC_{system}). Scheme 2 analyzes the bus transmission and pipeline cycles for 1-bit pyramid search with separated PPU and MEU. The required bus transmission is reduced to 36.6%¹, but no influences to the pipeline cycles. In Scheme 3, if we apply MB pipelining scheme to MEU/PPU, the bus transmission is further reduced to 32.1%² with increased pipeline cycles from NC_{MEU} to $NC_{\text{MEU}} + NC_{\text{PPU}}$. However, the required cycles $NC_{\text{MEU}-1\text{bit}}$ for 1-bit pyramid search is far less than the running cycles for 8-bit pyramid search $NC_{\text{MEU}-8\text{bit}}$ (i.e. $NC_{\text{MEU}-1\text{bit}} \ll NC_{\text{MEU}-8\text{bit}}$), this provides the design space to integrate PPU with MEU to have reduced bus bandwidth with minor or no influences to the system pipelining.

¹ $(256+42+42)/(256+336+336) \times 100\% = 36.6\%$.

² $(256+42+0)/(256+336+336) \times 100\% = 32.1\%$.

Table 3.1: Evaluation of the pre-processing unit design schemes for pyramid based search.

Scheme	Bit Precision	Pipeline	Bus Bandwidth (Bytes)		Pipeline Cycles (NC_{system})
1	8-bit	MEU/PPU: Frame MEU/MEU: MB	E-MEM→PPU	256 ¹	NC_{MEU}
			PPU→E-MEM	336 ²	
			PPU→MEU	336	
2	1-bit	MEU/PPU: Frame MEU/MEU: MB	E-MEM→PPU	256	NC_{MEU}
			PPU→E-MEM	42 ³	
			PPU→MEU	42	
3	1-bit	MEU/PPU: MB	E-MEM→PPU	256	NC_{MEU} NC_{PPU}
		MEU/MEU: MB	PPU→E-MEM	42	
			PPU→MEU	0	

¹ 16×16 .

² $16 \times 16 + 8 \times 8 + 4 \times 4$.

³ $(16 \times 16 + 8 \times 8 + 4 \times 4) / 8$.



B. Sequential LV2 Binary Pyramid Search Structure

To examine the three levels of binary pyramid search structure, the ABME algorithm is composed of a sequential fine tuning process in level 2 (LV2) and two levels (LV1 and LV3) of small range full search. The two levels of full search have regular data flow that is easy for VLSI implementation. However, the sequential fine tuning process as shown in Fig. 3.6(a) contains redundant data access between the selected candidate for ± 1 cross pattern search and the 6 candidates for fine tuning search. The selected candidate for ± 1 search accesses the same search data from the memory as one of the 6 candidates in the tuning process. With optimized hardware architecture, we can remove the redundant cycles and power in accessing the repeated memory in LV2 search without sacrificing R-D performance.

C. Support of B-frame and 8x8 Block Search

The B-frame search and 8x8 block search are commonly adopted in latest compression standards such as MPEG-4 [31] or H.263+ [32] to enhance coding efficiency. In our reference MPEG-4 software, the ± 2 of 8x8 block search is applied after 16x16 search is completed at integer pixel resolution. It means that such 8x8 block search can extend the search range to $[-18, +17]$ with the search range of $[-16, +15]$ for 16x16 search. Such a simple step increases the bus access by 17%³. In hardware point of view, the sequential processing of block 8×8 and 16×16 also has longer processing cycles, but often accesses the same data from memory for video sequences with slow or no motion which is redundant. Hence, an optimized hardware flow to avoid the redundant memory access and longer processing cycles is needed.

For the B-frame search, search with two reference frames doubles the execution cycles and bus access to cause the system pipelining difficult. Hence, an optimized hardware architecture to parallel processing of B-frame search without cause system pipelining difficult is needed.

3.3 BBME Algorithm

The BBME algorithm addresses the three design issues as described in Section 3.2.2. Fig. 3.4(b) shows the processing procedure of BBME algorithm. The first modification is to replace the original frame pre-processing with macroblock level pre-processing. The second modification is to simplify the LV2 search flow of ABME. The third modification

³ $\frac{3(18 \cdot 2 + 16)^2}{(16 \cdot 2 + 16)^2} = 117\%$

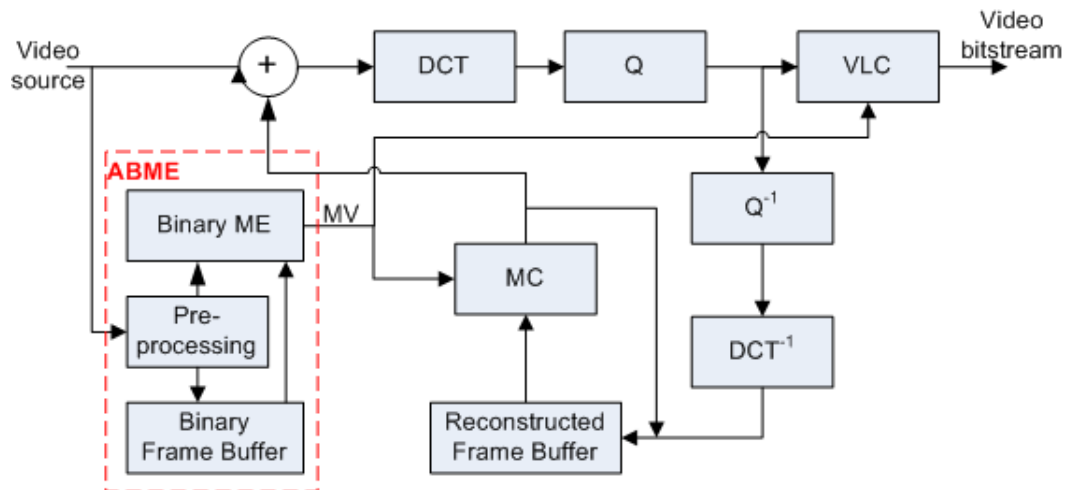


Figure 3.2: Functional block diagram of a generic video encoder by adopting ABME algorithm.

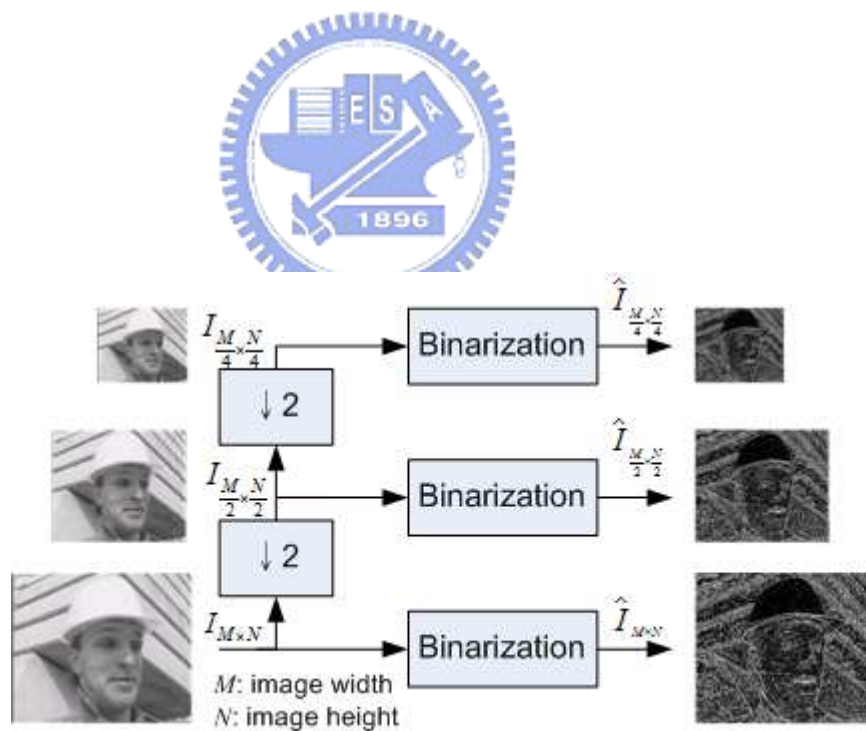


Figure 3.3: The processing flow of pre-processing module in ABME algorithm.

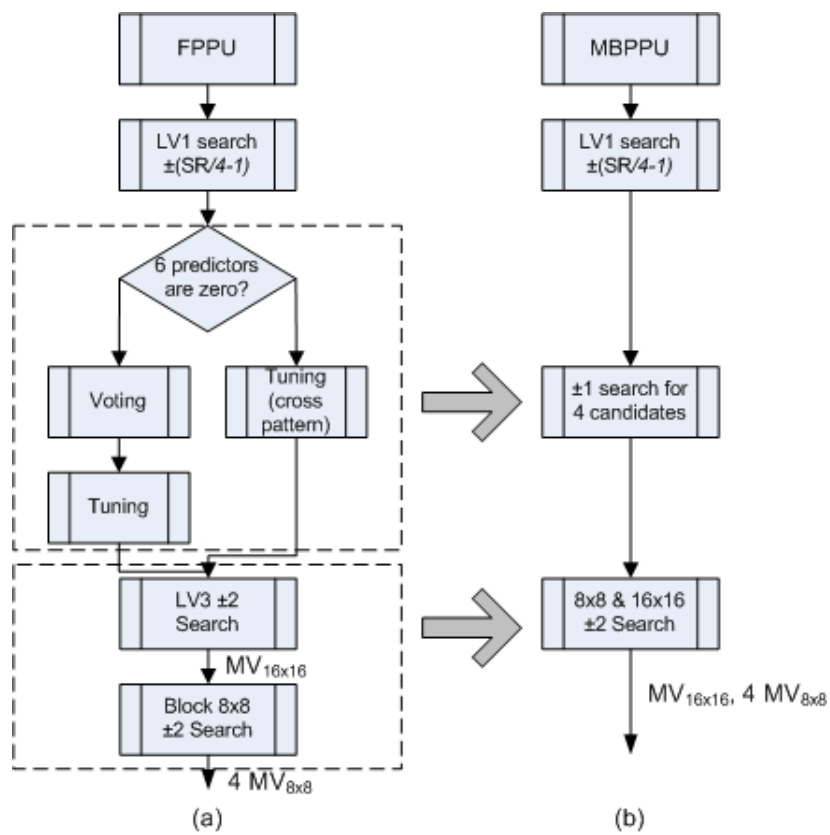


Figure 3.4: The processing procedure for ABME and BBME flow. (a) ABME flow [41] (b) BBME flow.

Table 3.2: Comparison of different K values for macroblock based pre-processing unit.

Block Size ($K \times K$)	16×16	18×18	20×20	30×30
Required Data (bits)	256	324	400	900
PSNR Loss (dB)	-0.50	-0.14	-0.10	-0.00

is to support parallel processing of 8×8 and 16×16 LV3 block search.

3.3.1 Macroblock Pre-processing Unit (MBPPU)

The MBPPU removes the repeated bus access. As opposed to frame level implementation of the pre-processing module, the MBPPU is integrated with the binary search module to generate MB level binary search block for current frame. The three levels of binary search blocks are then stored back to external memory as reference picture for the next frame. To integrate the pre-processing module with the MB level pipelining, a straightforward approach is to implement at macroblock level as shown in Fig. 3.5(a). Such a method needs to transmit 30×30^4 image data to generate the 4×4 LV1 binary block. An alternative approach is to replace the 30×30 image data with smaller $K \times K$ image data and pads the missing image pixels. The value of K has no effect on the search range because such a simplification is applied for the current search block. Fig. 3.5(b) shows an example for $K = 18$. We pad the boundary pixels to fill the missing pixels at LV2 and LV1 to generate three levels of binary search blocks. We experiment with various K values, and Table 3.2 shows the PSNR results. From this Table, $K = 18$ is selected as it has a PSNR loss around 0.1dB on the average that is a tolerable penalty in quality.

$$^4(((L_{LV3} + 2) \times 2 + 2) \times 2 + 2) = (((4 + 2) \times 2 + 2) \times 2 + 2) = 30$$

3.3.2 Efficient LV2 Search

The modified LV2 search is to remove the redundant on-chip memory access in the LV2 flow of the original ABME algorithm. Fig. 3.6 shows the LV2 processing flow. In Fig. 3.6(a), the original LV2 flow is a software efficient flow that checks each candidate sequentially and then performs ± 1 search. Repeated on-chip memory access happens for the first candidate if the first candidate is selected for ± 1 search finally. The modified LV2 flow is efficient for hardware as shown in Fig. 3.6(b). It reduces the number of checked candidates to avoid longer processing cycles. It also changes the processing order as the labeled numbers in Fig. 3.6(b) to improve parallelism and avoid repeated on-chip memory access for the selected candidate.

3.3.3 Parallel Processing of 8×8 and 16×16 Block Searches

In our MPEG-4 software [31], ± 2 of 8×8 block search is performed after 16×16 search is completed at integer pixel resolution. To support 8×8 block search, the memory bandwidth will be increased by 17% while it is beneficial for area with complex motion. To balance the tradeoff between quality and bus bandwidth, a modified search method is used. To minimize bus access, we restrict the 8×8 block search to start at the same search center as the 16×16 LV3 search so that the search range is still within $[-16, +15]$. It enhances parallelism, and only suffers PSNR loss of 0.1dB for the worst case.

Table 3.3 analyzes the original and modified methods for 8×8 and 16×16 block search in LV3. One 16×16 block search requires 256 subtractions and 256 absolute value operation to calculate the absolute differences, 255 additions to sum up the differences, 1 comparator to decide if this is best block with minimal distortion. Similar to 16×16 block

Table 3.3: Evaluation of the ABME and BBME flow for block 8×8 and 16×16 searches in LV3.

Method	Processing	Arithmetic Operations			
		16×16 search	8×8 search	Total	Percentage
ABME	Sequential	768 ¹	192 ²	38400 ³	100%
BBME	Parallel	4 ⁴	192	19300 ⁵	50.3%

¹=256 sub + 256 abs + 255 add + 1 cmp.

²=64 sub + 64 abs + 63 add + 1 cmp.

³=25(points) \times 768 + 4(8x8blocks) \times 25(points) \times 192

⁴3 add + 1 cmp.

⁵=25(points) \times 4 + 4(8 \times 8blocks) \times 25(points) \times 192

search, the 8×8 block search requires 64 subtractions and 64 absolute value operation to calculate the absolute differences, 63 additions to sum up the differences, 1 comparator to decide if this is best block with minimal distortion. The total operational counts for the original method is 38400 as shown in Table 3.3.

For the proposed new method, we parallel processing the 8×8 and 16×16 block searches. Since both the searches starts from the same locations, the operational results for block matching is reusable. Thus, 16×16 block search just need 3 additions to sum up the results for four 8×8 blocks, and 1 comparator to decide if this is the best block with minimal distortion. The total operational count is 19300 with 49.7% saving. The new method is also optimized for hardware operations since it can allow parallel processing of 8×8 and 16×16 block searches. Therefore, the new method is better for hardware operations than the old one.

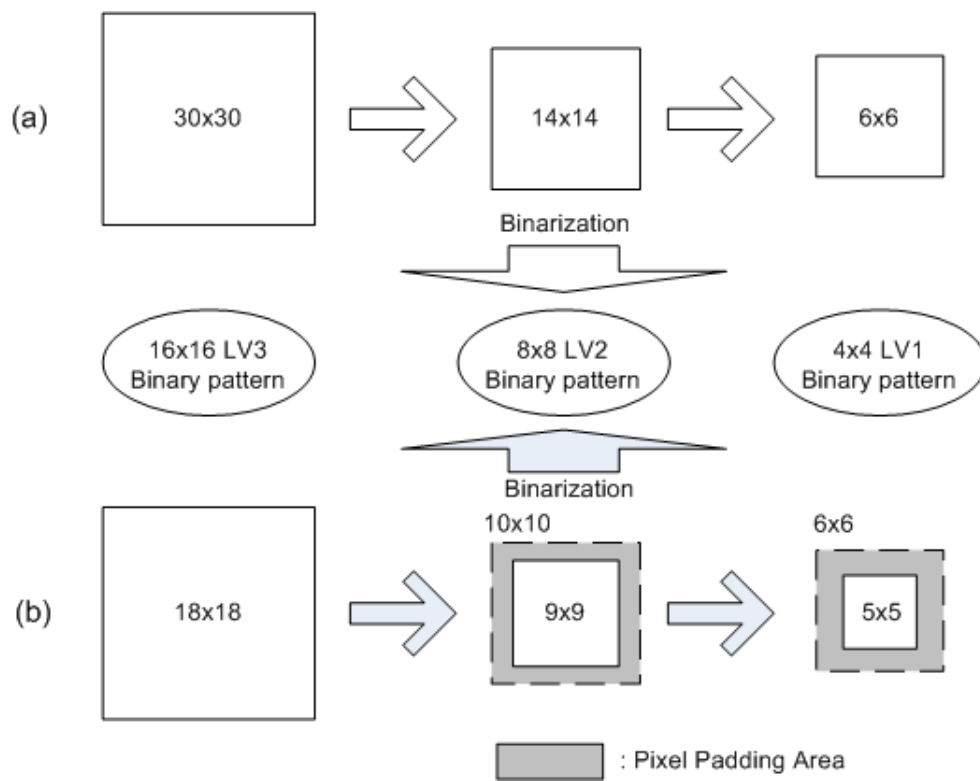


Figure 3.5: The pre-processing flow in macroblock pre-processing unit. (a) $K=30$ (b) $K=18$. (The shadow area is padding pixels)

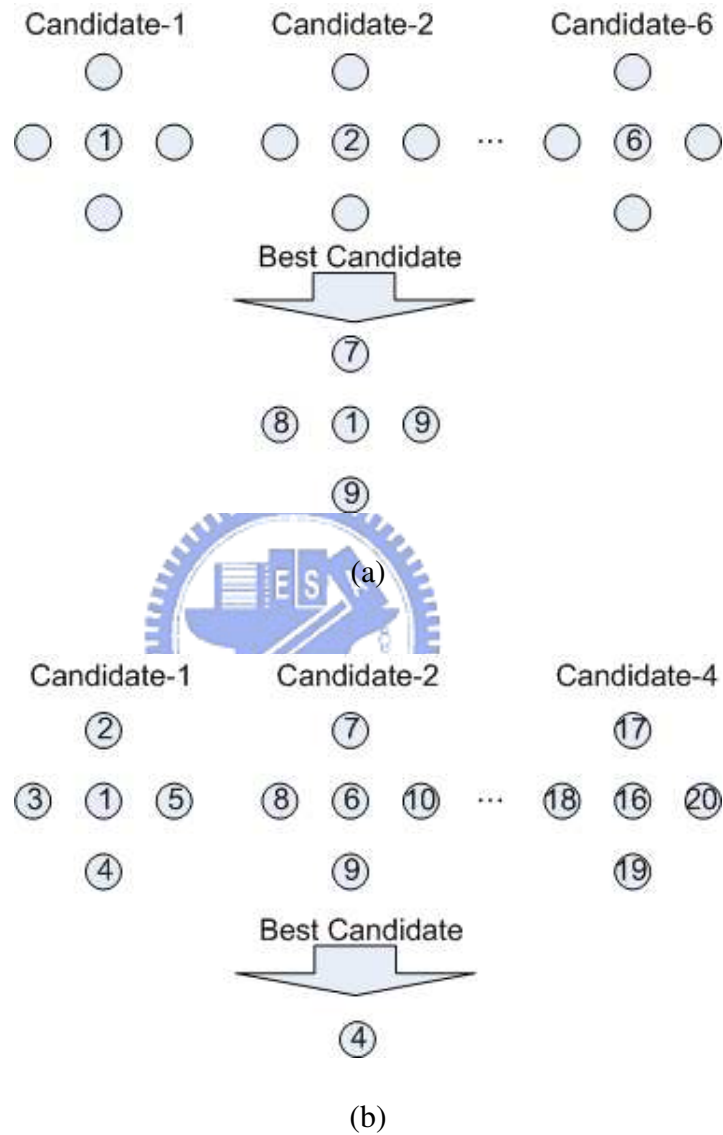


Figure 3.6: The LV2 processing flow. (a) original LV2 flow in ABME (b) new LV2 flow in BBME. The number represents the processing order.

3.4 Hardware Architecture

3.4.1 System Architecture

Fig. 3.7 shows the system architecture of the proposed design that implements the BBME algorithm with the parallel hardware support for B-frame search as described in Section III. For B-frame search, two sets of hardware are used to enable forward and backward search in parallel. The 8-bit data of current search block is passed to the pre-processing module (MBPPU) to generate three levels of binary search blocks. The three levels of binary data are stored in the on-chip memories C1-C3. The binary data of forward and backward reference frames are stored in two on-chip memories, S01-S03 and S11-S13, respectively. The memory blocks of C1, S01 and S11 are for LV1. The memory blocks of C2, S02 and S12 are for LV2. The memory blocks of C3, S03 and S13 are for LV3.

For each level of block search, the address generator (AG) controls the access of C0-C3, S01-S03, S11-S13 to provide the necessary current and target block data to the shared SOD processing units for block matching (SOD1 for forward and SOD2 for backward). The shared processing units firstly decide which level of binary data to be used for calculation according to the control signal from controller (CTRL). Then, it computes the SOD between selected current and reference search blocks. The matching results are sent to the comparator for final motion vector selection considering motion vector cost input from the motion vector generator module (VG).

For the forward only P-frame search, the parallel architecture leaves half the hardware idle. Such an issue is addressed with a parallel P-frame search scheme. In the parallel P-frame search mode, the forward search data from S01-S03 are mirrored to S11-S13. The

original forward search path through SOD1 module handles search for the odd positions while the backward search path through SOD2 module handles search for the even positions. The AG/CTRL/VG/Comparator modules will send appropriate addresses and signals for each path. Thus, both paths are busy with half execution cycles.

Compared to the conventional serial architecture which processes forward and backward search sequentially, this parallel architecture enjoys five major advantages including:

- *Less on-chip memory access:* Parallel architecture reuses the current block search data, removes redundant on-chip memory access, and thus saves power. This is particularly important for pyramid search structure since the current block data are changed for each level of search.
- *Higher overall hardware utilization:* In addition to full hardware utilization for ME, all the other modules in the pipeline enjoy higher utilization. Typically ME module takes the longest execution cycles as compared to other modules such as transform or motion compensation (MC), so it can become the design critical path in the overall system. The execution cycles are doubled for B-frame search which leads to more idle cycles for the other modules such as entropy coding or transform. Thus, parallel architecture can not only halve the ME execution cycles but also reduce the idling of other modules. Compared to serial architecture, our parallel architecture decreases execution cycles for both P-frame search and B-frame search. As shown in Table 3.8, the P-frame and B-frame searches take 177 cycles and 148 cycles, respectively.
- *Lower working frequency:* Lower working frequency is a key factor leading to a low power design. Similar to the previous item, the B-frame ME search is typically the slowest module. In that case, it is the dominant factor to decide the system frequency.

The parallel B-frame architecture as opposed to serial architecture improves the worst case scenario leading to the lowest system frequency. Combing with the voltage scaling technique, parallel architecture can achieve further power reduction.

- *Less penalty in hardware cost:* It is less expensive to use parallel architecture for the binary search. If the system were to use full pixel (8-bit) for matching, parallel architecture suffers more increase of hardware. Although the increase in percentage is the same, the binary search algorithm has smaller increase.
- *Flexibility for joint optimization of B-frame search:* Joint optimization of B-frame search is a widely used encoding technique that jointly considers cost and distortion based on the forward and backward search results to provide better motion vectors. For a serial architecture, it needs to finish the forward and backward searches first. Then, the joint optimization can start. Parallel architecture can save cycles and the memory for storing first pass results.

Table 3.4 lists a brief summary to compare serial architecture and our parallel architecture. Although the on-chip memory is doubled, the necessary local memory size for the 1-bit parallel architecture is still only one quarter of the 8-bit sequential design. The peak memory bandwidth of the proposed parallel 1-bit architecture is $(B_1 + 2 \times B_2)$ in which B_1 represents the current frame on-chip memory bandwidth and $(2 \times B_2)$ as the bandwidth for two parallel forward and backward reference frames. The execution cycles are algorithms dependent, but the ME with 8-bit data type may need more cycles to complete one location of search. The reason is that binary search can complete one search location matching in one cycle with its simple XOR operation. On the contrary, it is difficult for most ME designs with 8-bit data type to achieve single cycle execution except when two-dimensional

Table 3.4: Comparison for serial and parallel architecture.

Architecture	Serial		Parallel	
	8-bit	1-bit	8-bit	1-bit
Memory size	$(8 \cdot A_M)$	A_M	$2 \cdot (8 \cdot A_M)$	$2 \cdot A_M$
Peak memory bandwidth	$8 \cdot (B_1 + B_2)$	$(B_1 + B_2)$	$8 \cdot (B_1 + 2 \cdot B_2)$	$(B_1 + 2 \cdot B_2)$
Running cycles	$\geq 2 \cdot NC$	$2 \cdot NC$	$\geq NC$	NC
Hardware cost	$8 \cdot A_L$	A_L	$8 \cdot (2 \cdot A_L)$	$2 \cdot A_L$

systolic array is used. The binary search is more suitable for parallel architecture due to its simplicity.

3.4.2 Macroblock Pre-processing Unit (MBPPU)

The three levels of binary pyramid data for search are generated by MBPPU. The binarization processing elements (PE) are the key modules to convert the 8-bit image data into binary format. Inside the binarization PE, it contains a 3×3 filtering operation and a comparator. To support the 3×3 filter design as shown in eq. (3.1), three rows of line buffers are needed to output a line of binary data. Fig. 3.8 shows the architecture of MBPPU. For each level of pre-processing, there are three rows of data buffers, several binarization (BIN) PEs, and the row rotators. The three rows of data buffers are used to store three rows of 8-bits data before entering binarization process. The filter operation is to implement filter H_A . The number of BINs is designed according to the processing data rate at each level, and the processing rate depends on the width of each row buffer (18 for LV3, 10 for LV2 and 6 for LV1). So, the number of PEs for LV3 to LV1 is 9, 5, and 2. The row rotator outputs correct data for binarization.

3.4.3 Three Levels of Binary Search

The three levels of binary searches are done by three steps of operations: (1) CTRL informs AG about the current level index for search (2) AG sends addresses to local memories (3) local memories output data to SOD1 and SOD2 for search. Fig. 3.9 shows the operational flow of processing units of SOD1 and SOD2. The processing unit is basically a 256-bit XOR operation followed by a 256-bit adder tree. To be able to support three search block sizes for LV1 to LV3, the 256 bits XOR operations are partitioned into 16 blocks of 16-bit XOR operations to provide 16 4x4 SOD results $S_i^{4 \times 4} \{i = 0 \sim 15\}$. Then, the sixteen 4x4 SODs can be accumulated as four 8x8 SODs $S_i^{8 \times 8} \{i = 0 \sim 3\}$ or one 16x16 SOD $S_0^{16 \times 16}$. To make sure the input data and the output motion vectors are moved correctly, the CTRL sends a control signal to inform the processing unit about which level is being processed.

For LV2, we need to process 5 search points in parallel. However, the processing unit can only handle four 8x8 SOD operations in parallel. Thus, the search point opposite to the motion vector direction is abandoned.

3.5 Experimental Results and Analysis

3.5.1 Rate-Distortion (R-D) Performance Evaluation

Table 3.5-Table 3.7 shows the PSNR with bit rate of 256, 512 and 1024 kbps for full search (FS), ABME [35] and BBME. Fig. 3.10 and Fig. 3.11 plot the R-D curves for Foreman and Mobile sequences. There are totally five commonly used MPEG test sequences tested with GOP structures of IPPPP (M=1) and IPBPB (M=2). The search range is [-16,

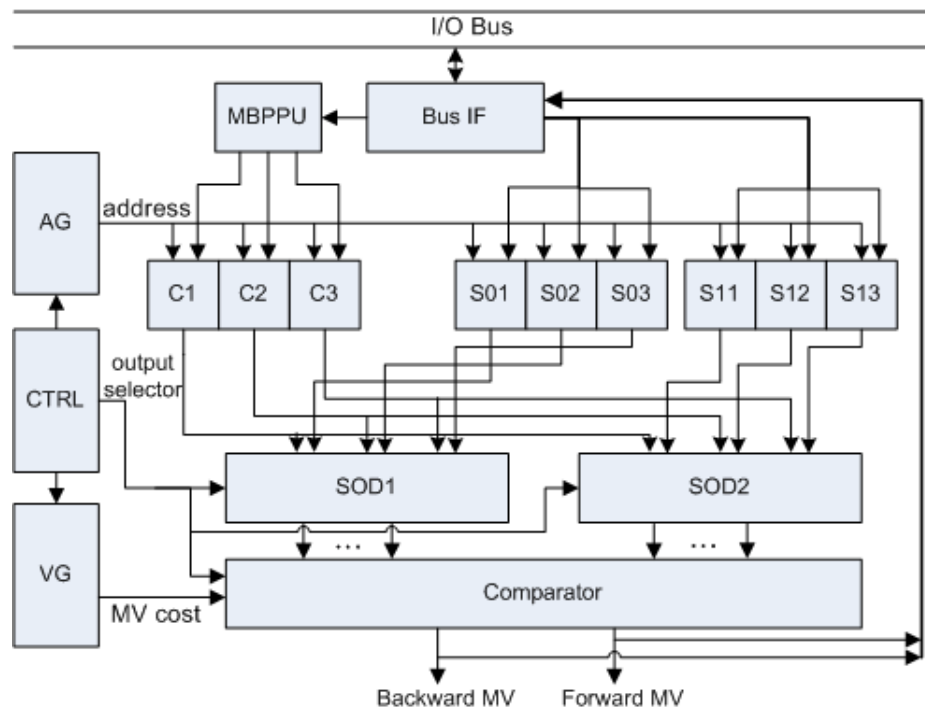


Figure 3.7: System architecture for the BBME design.

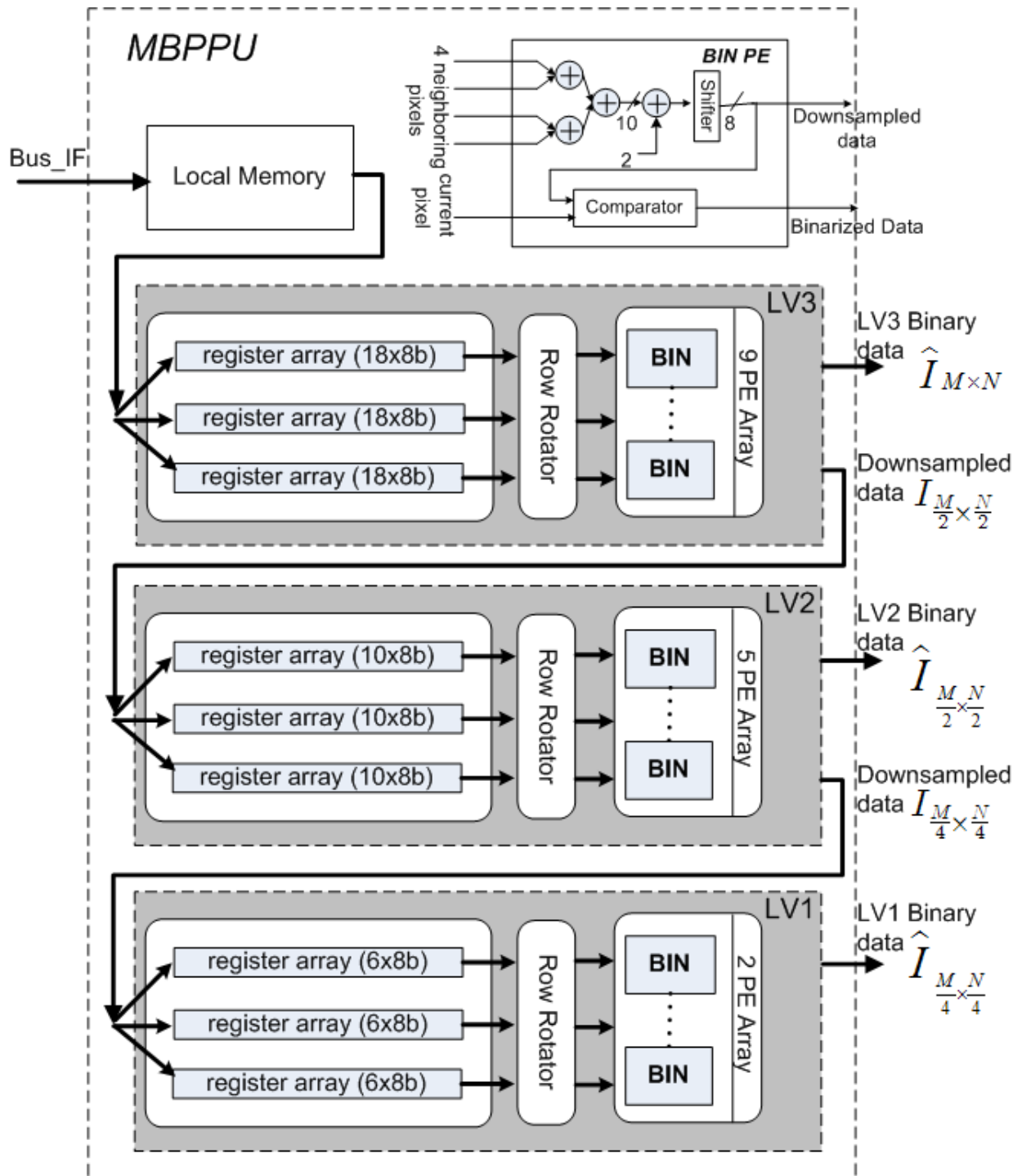


Figure 3.8: Architecture of macroblock based pre-processing unit.

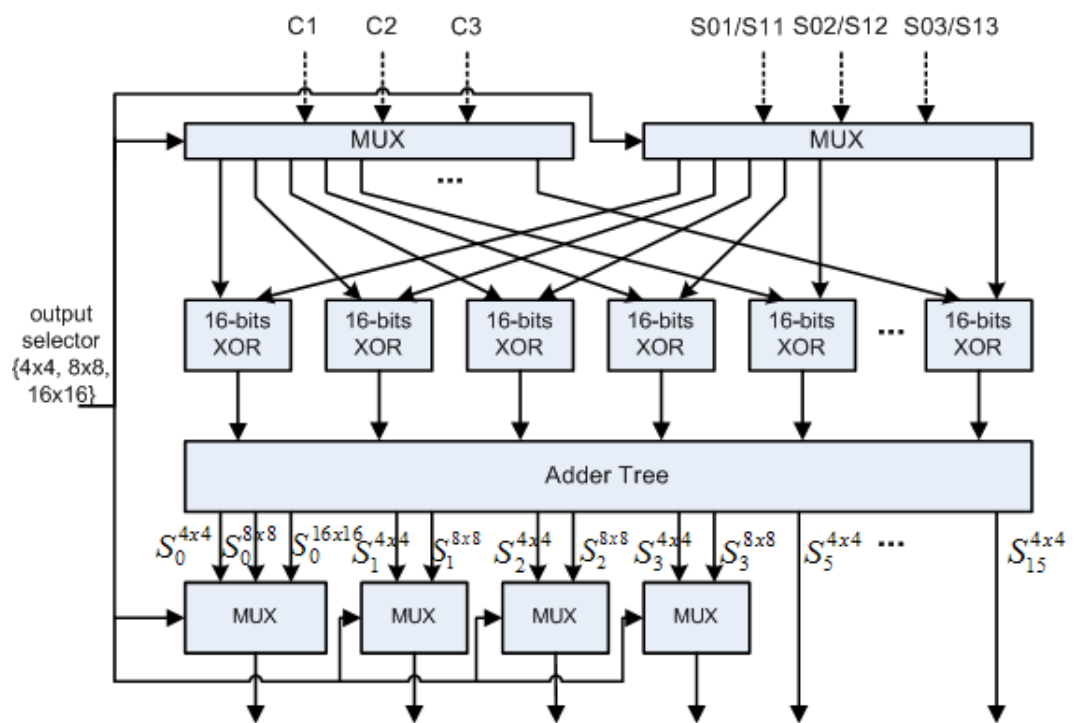


Figure 3.9: Shared processing unit for three levels of binary pyramid searches.

Table 3.5: R-D performance for full search (FS), ABME algorithm (ABME)[35] and the BBME algorithm at the bitrate of 256 kilo bps (N=300).

Sequence	Method	IPP 30fps (M=1)		IBPBP 30fps (M=2)	
		PSNR_Y (dB)	Δ PSNR	PSNR_Y (dB)	Δ PSNR
Foreman	FS	30.84		30.49	
	ABME	30.31	-0.53	30.17	-0.32
	BBME	30.23	-0.61	29.99	-0.50
Akiyo	FS	41.61		41.85	
	ABME	41.50	-0.11	41.90	+0.05
	BBME	41.59	-0.02	41.83	-0.02
Flower	FS	23.86		23.88	
	ABME	23.70	-0.16	23.72	-0.16
	BBME	23.69	-0.17	23.72	-0.16
Mobile	FS	23.38		24.26	
	ABME	23.37	-0.01	24.19	-0.10
	BBME	23.38	-0.00	23.74	-0.52
Tempete	FS	26.04		27.00	
	ABME	25.95	-0.09	26.60	-0.40
	BBME	25.98	-0.06	26.47	-0.53

+15]. From these 3 Tables, the PSNR loss for the modified ABME algorithm is up to 0.45dB for IPPPP and 0.75dB for IPBPB compared to full search. Compared with ABME, up to 0.14 dB PSNR loss is observed for IPBPB.

For subjective quality evaluation, Fig. 3.12 to Fig. 3.14 shows the encoded pictures of Foreman 32nd, 99th, and 148th frames. The left side of Fig. 3.14 used full search and its PSNR is 34.99dB. The right side used BBME and its PSNR is 34.63dB. The two pictures have minor visual difference in detailed area. The other 2 Figures have similar results.

3.5.2 Hardware Design Performance

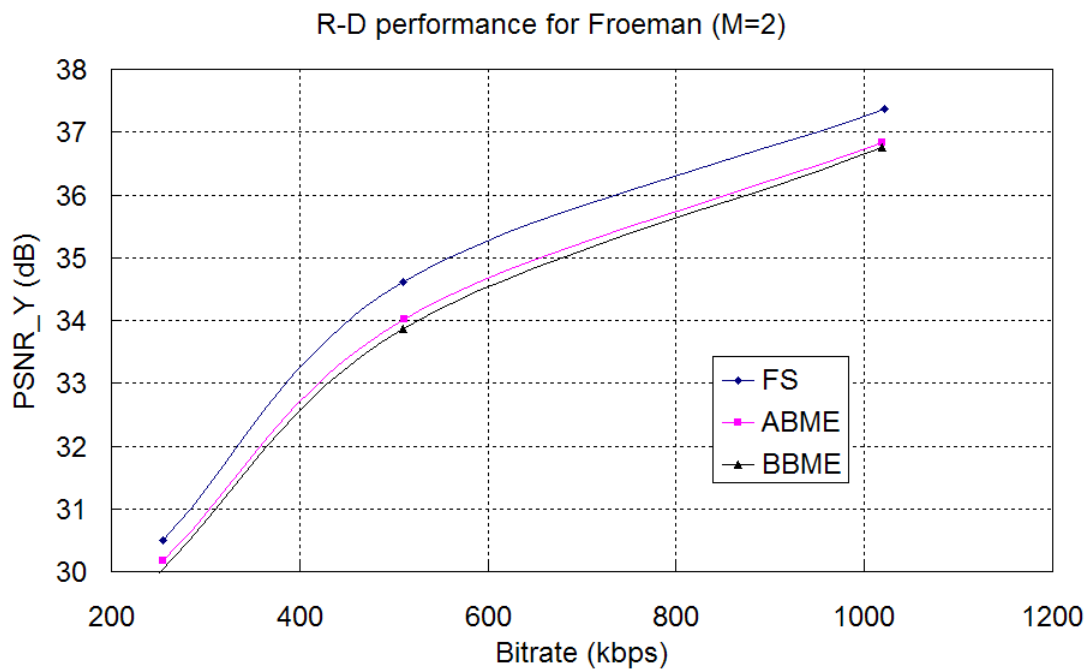
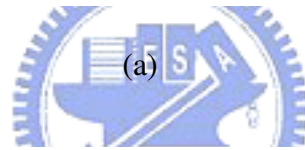
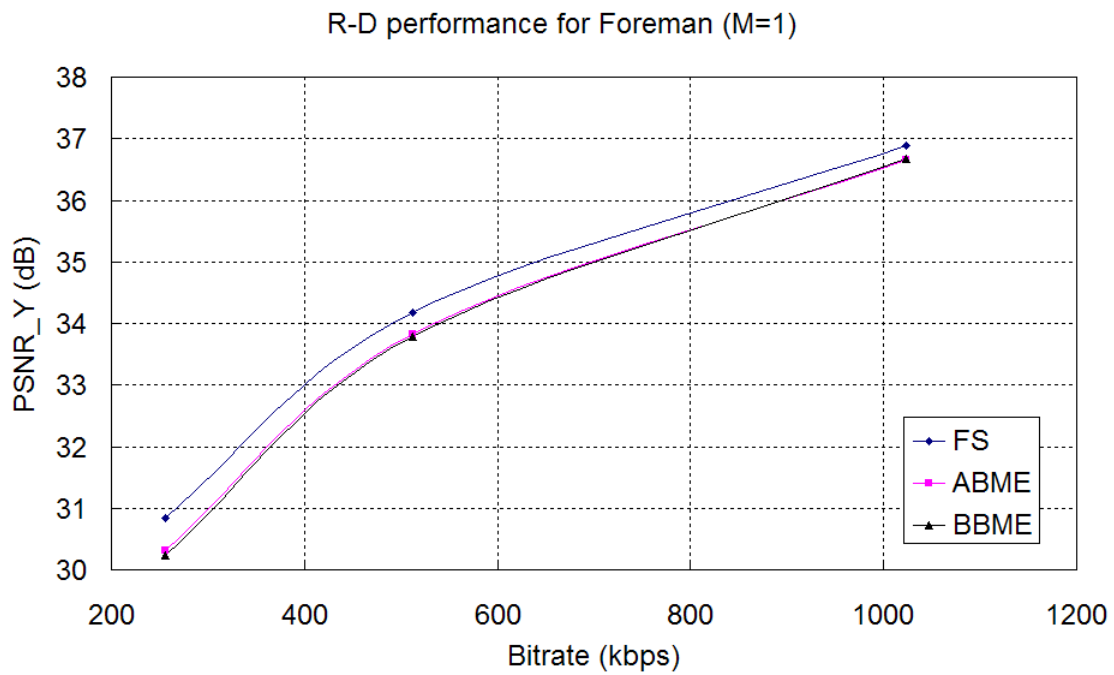
Table 3.8 summarizes the execution cycles and gate counts for P-frame and B-frame block search. For each MB, totals of 148 and 177 cycles are used for P-frame and B-frame search respectively. It is found that more than 60% (107 cycles) of overall execution cycles

Table 3.6: R-D performance for full search (FS), ABME algorithm [35] and the BBME algorithm at the bitrate of 512 kilo bps (N=300).

Sequence	Method	IPPP 30fps (M=1)		IBPBP 30fps (M=2)	
		PSNR_Y (dB)	Δ PSNR	PSNR_Y (dB)	Δ PSNR
Foreman	FS	34.18		34.62	
	ABME	33.82	-0.36	34.01	-0.61
	BBME	33.79	-0.39	33.87	-0.75
Akiyo	FS	43.33		43.47	
	ABME	43.31	-0.02	43.39	-0.08
	BBME	43.33	-0.00	43.52	+0.05
Flower	FS	26.11		26.56	
	ABME	25.75	-0.36	26.29	-0.27
	BBME	25.66	-0.45	26.30	-0.26
Mobile	FS	26.18		27.69	
	ABME	26.10	-0.08	27.59	-0.10
	BBME	26.07	-0.11	27.33	-0.36
Tempete	FS	28.78		29.89	
	ABME	28.78	-0.00	29.67	-0.22
	BBME	28.79	+0.01	29.55	-0.34

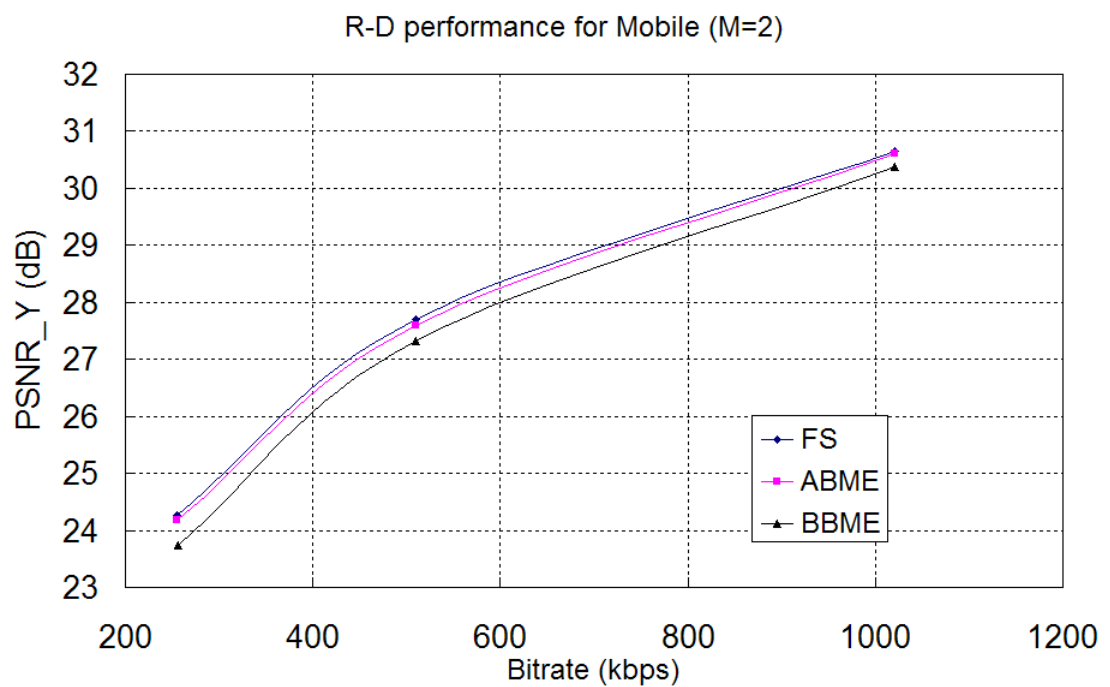
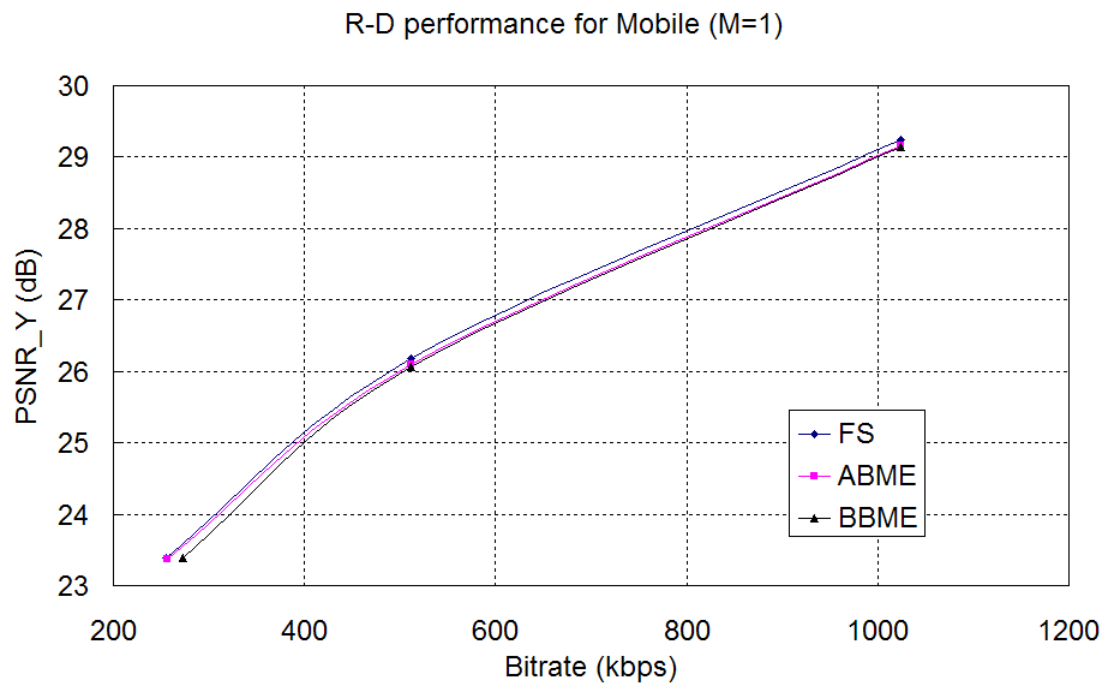
Table 3.7: R-D performance for full search (FS), ABME algorithm [35] and the BBME algorithm at the bitrate of 1024 kilo bps (N=300).

Sequence	Method	IPPP 30fps (M=1)		IBPBP 30fps (M=2)	
		PSNR_Y (dB)	Δ PSNR	PSNR_Y (dB)	Δ PSNR
Foreman	FS	36.89		37.37	
	ABME	36.66	-0.23	36.84	-0.53
	BBME	36.67	-0.22	36.75	-0.62
Akiyo	FS	44.43		44.95	
	ABME	44.41	-0.02	44.91	-0.08
	BBME	44.44	+0.01	44.94	+0.05
Flower	FS	29.30		29.71	
	ABME	29.05	-0.25	29.57	-0.14
	BBME	29.01	-0.29	29.60	-0.11
Mobile	FS	29.25		30.65	
	ABME	29.16	-0.09	30.60	-0.05
	BBME	29.14	-0.11	30.38	-0.27
Tempete	FS	31.55		32.55	
	ABME	31.60	+0.05	32.38	-0.17
	BBME	31.60	+0.05	32.34	-0.21



(b)

Figure 3.10: R-D curves for full search (FS), ABME [?], and BBME designs with Foreman sequence. (a) IPPP (M=1) (b) IBPBP (M=2).



(b)

Figure 3.11: R-D curves for full search (FS), ABME [?], and BBME designs with Mobile sequence. (a) IPPP (M=1) (b) IBPBP (M=2).



Figure 3.12: Visual quality comparison for full search and BBME with Foreman 32th frame. (Left: 34.73dB for full search, Right: 34.39dB for BBME.)



Figure 3.13: Visual quality comparison for full search and BBME with Foreman 99th frame. (Left: 34.74dB for full search, Right: 34.36dB for BBME.)



Figure 3.14: Visual quality comparison for full search and BBME with Foreman 148th frame. (Left: 34.99dB for full search, Right: 34.63dB for BBME.)

(148 or 177 cycles) are in MBPPU. With limited bus bandwidth, the data movement takes longer than ME search. To avoid over-design of MBPPU, we slow down the processing throughput in MBPPU. With a 32-bit bus, the data movement takes 136 cycles for P-frame search and 171 cycles for B-frame search. As shown in Table 3.9, it takes 90 cycles to move current block data of $(18 \times 18 \times 8)$ bits for the MBPPU. It takes 32 cycles to move single reference search window data. It takes 11 cycles to move binary pyramid data that will be used as reference frame after MBPPU. It takes 3 cycles to move five motion vectors (MV) including one MV for 16x16 search and four MVs for 8x8 search. For the B-frame search, the cycles are doubled to be 6 for two sets of forward and backward MVs. To enable the pipelining of data movement and motion search as shown in Fig. 3.19, the proposed work is designed to meet the pipeline timing.

The hardware gate count is 62.6 kilo gates and the on-chip memory size is 8.64 kilo bits using TSMC 0.18 μ m CMOS technology. For CIF 30fps with GOP of IPPPP, the working frequency is 1.67MHz and the power consumption is 763 μ W measured by Synopsys

Table 3.8: Gate count and execution cycles for each module of our design.

	Cycle Count for P-Search	Cycle Count for B-Search	Gate Count (kilo)
MBPPU	107	107	14.2
LV1	4	6	7.2
LV2	6	8	17.6
LV3	27	54	22.5
Total	148	177	62.6

PrimePower. The power consumption for CIF 30fps with GOP of IPBPB is $896 \mu\text{W}$ with a working frequency of 1.94MHz. The functionality of this IP core design has been verified on FPGA.



To compare with the state-of-the-art designs, the proposed design shows obvious advantage in power consumption. Table 3.10 summarizes the design information for [35, 36, 37, 39, 43, 67, 26, 40]. The normalized power consumption is also listed to provide a fair comparison.

Table 3.11 and Table 3.12 shows the design metrics evaluation for the designs in Table 3.10 [35, 36, 37, 39, 67, 26, 40]. The design metrics starts the analysis from 6 dimensions including quality (Q), throughput (T), silicon area (A), I/O bandwidth (B), hardware utilization (U), and power consumption (P). Fig. 3.15 to Fig. 3.18 plots the hexagon plots for these designs. From the Figures, we can find the proposed bi-directional binary motion estimation (BBME) has lowest power consumption and lowest bus bandwidth as compared to other low power designs.

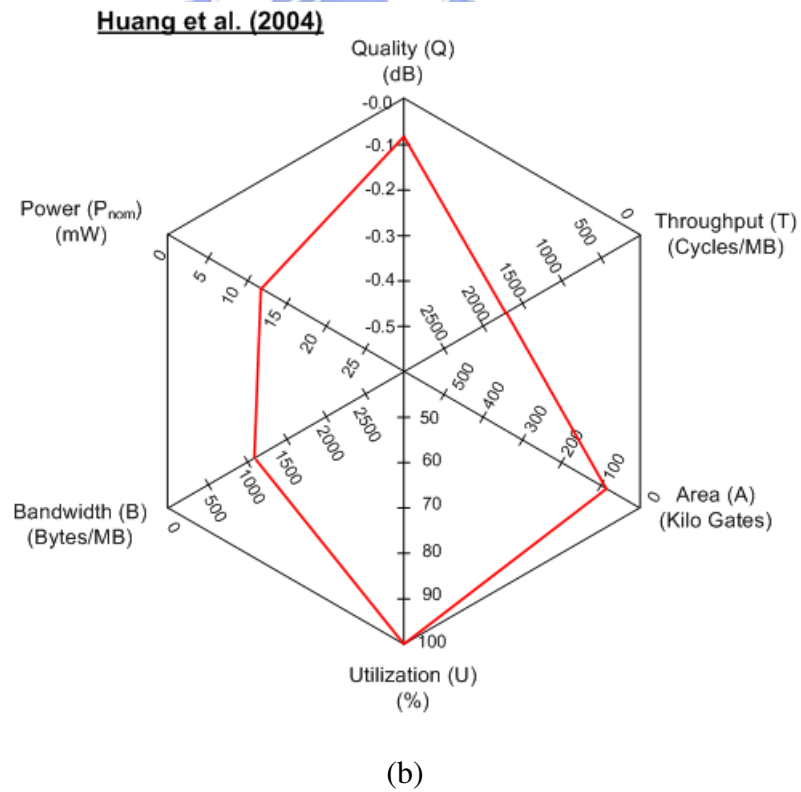
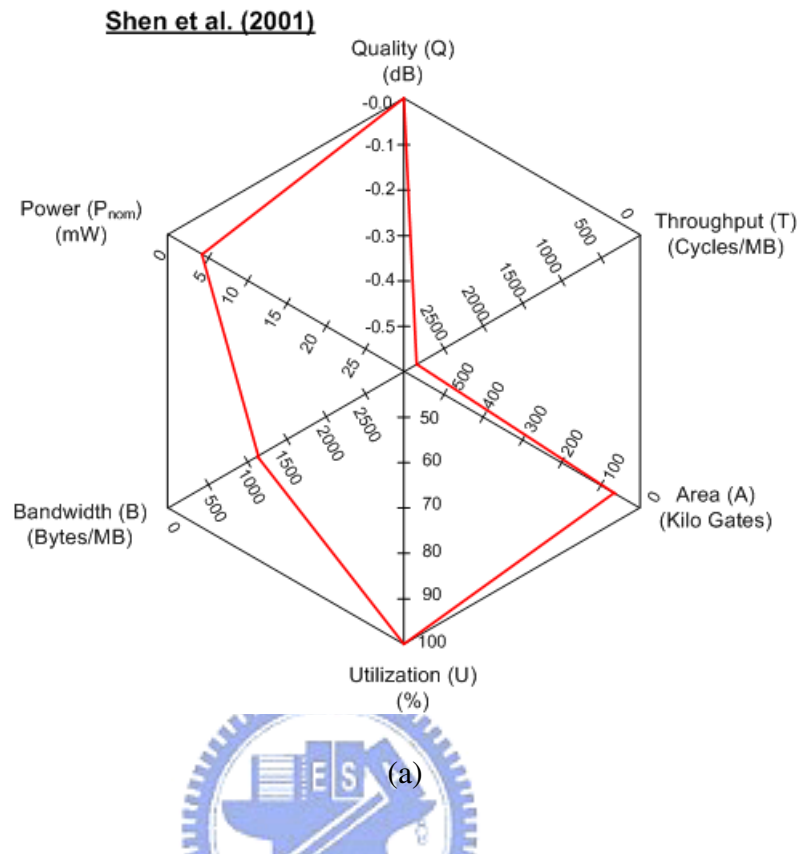


Figure 3.15: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Miyama’s work [36] (b) Chao’s work [30].

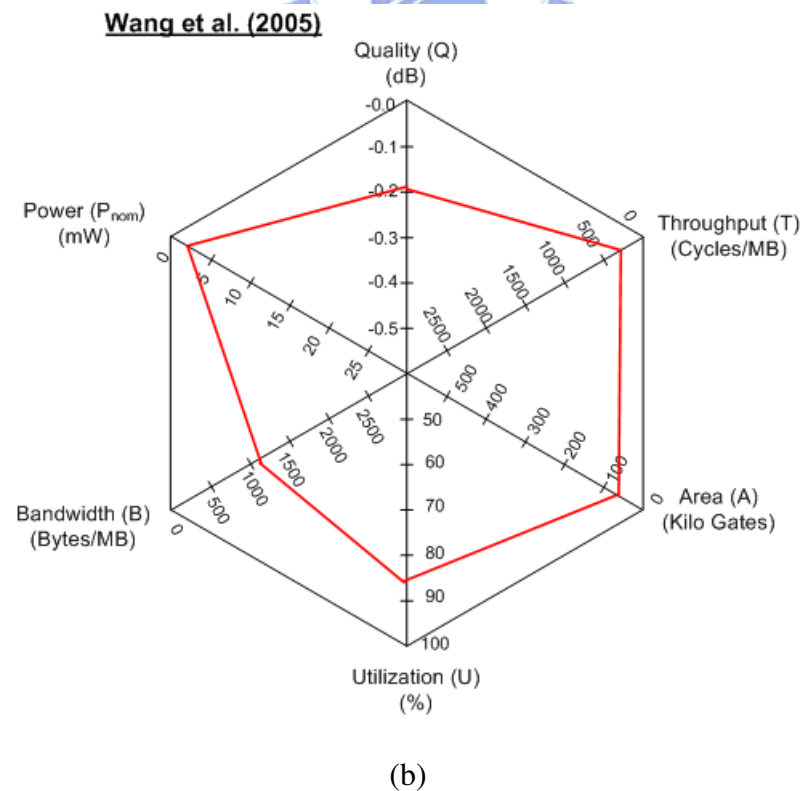
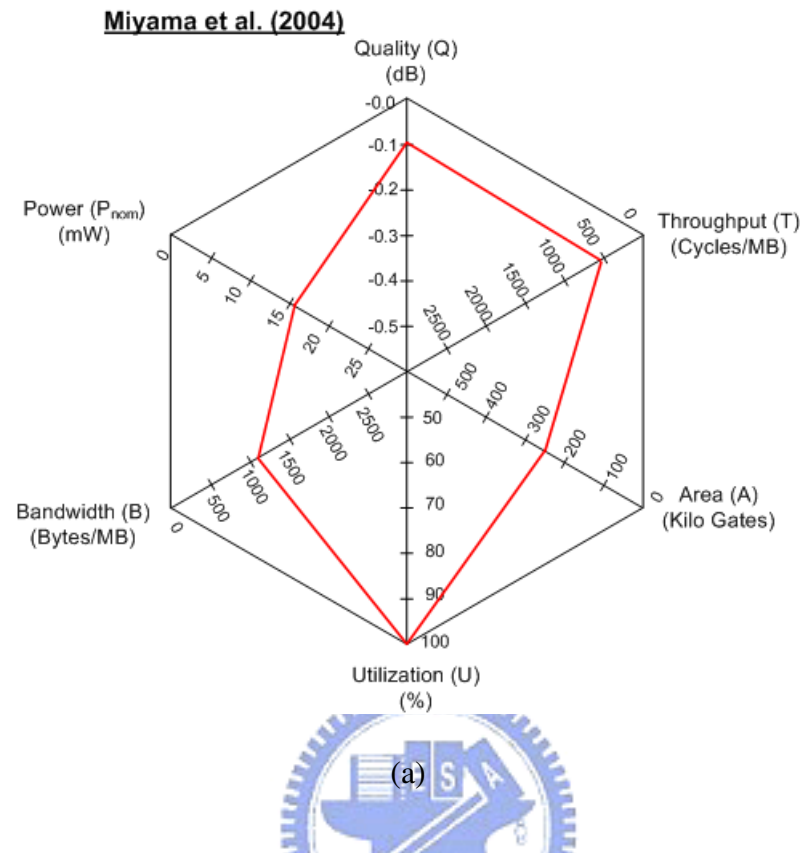
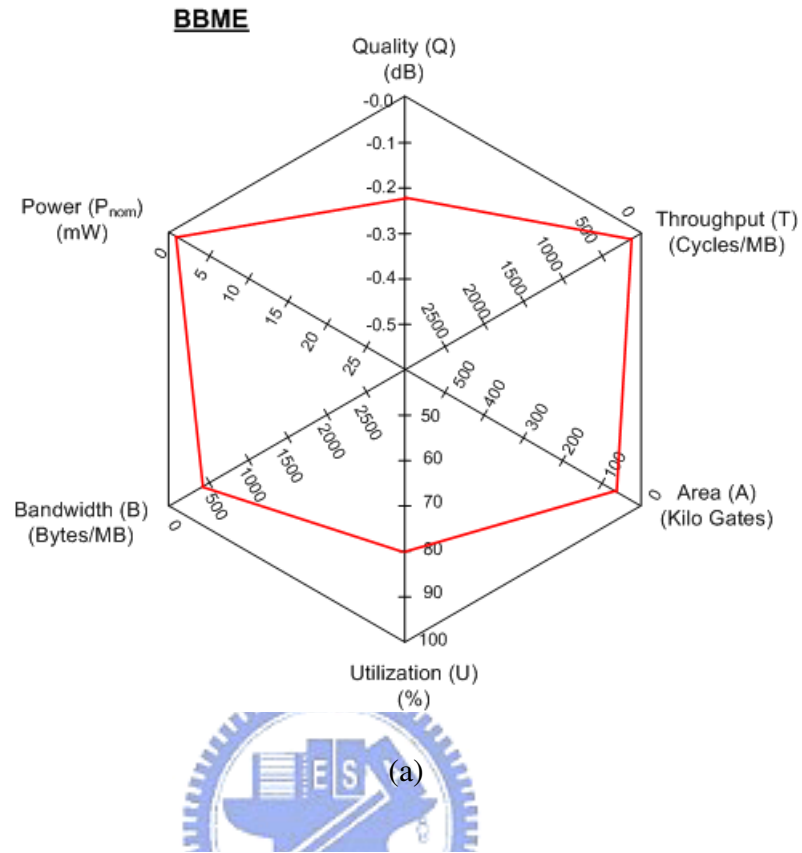


Figure 3.16: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Chen's work [26] (b) Huang's work [37].



Chen et al. (2007) Ultra Low Power Mode

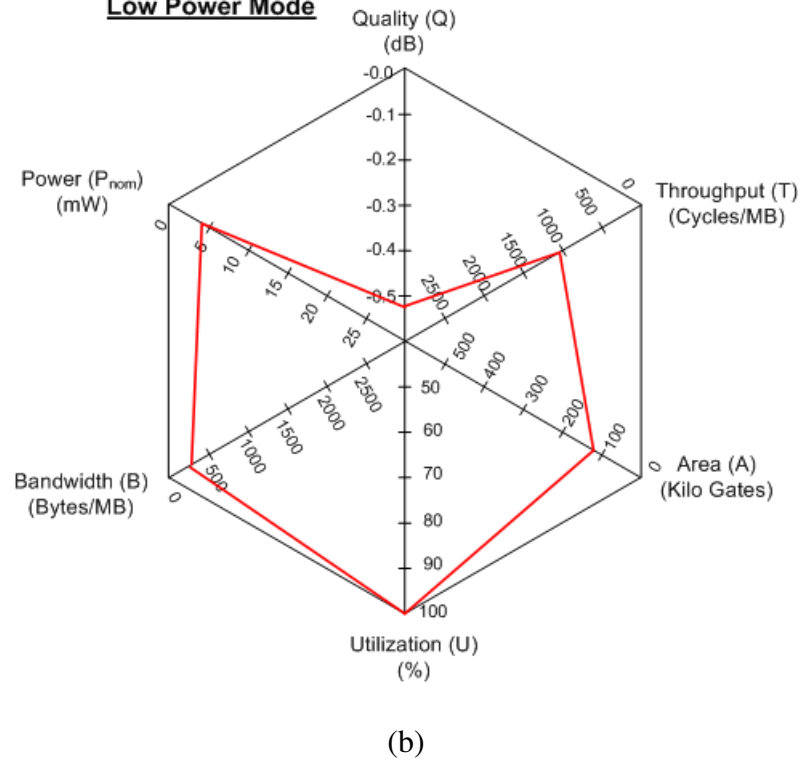


Figure 3.17: Hexagonal plot of 6 design metrics for the propose BBME design.

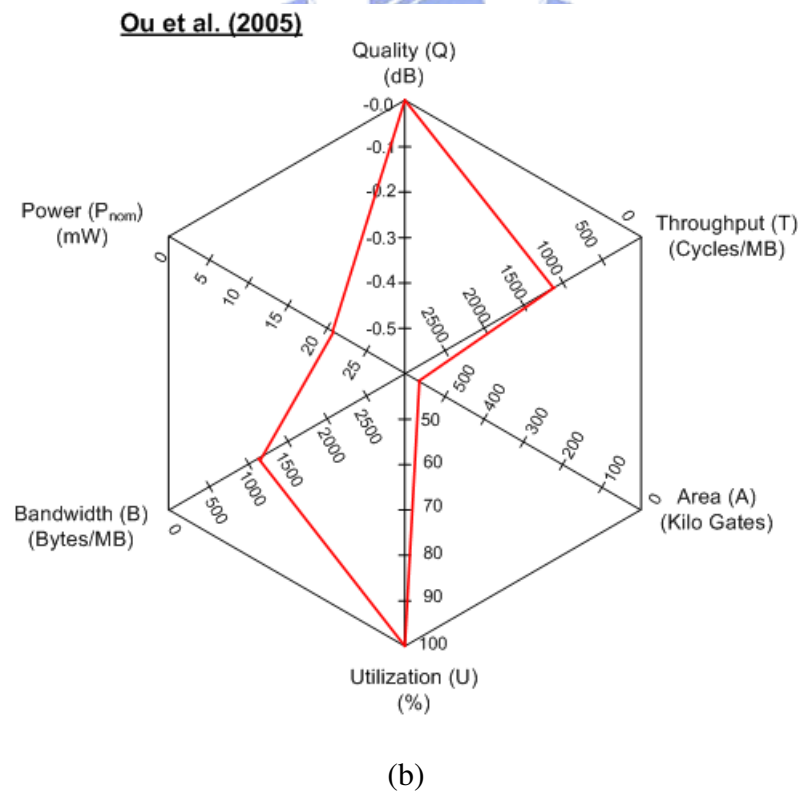
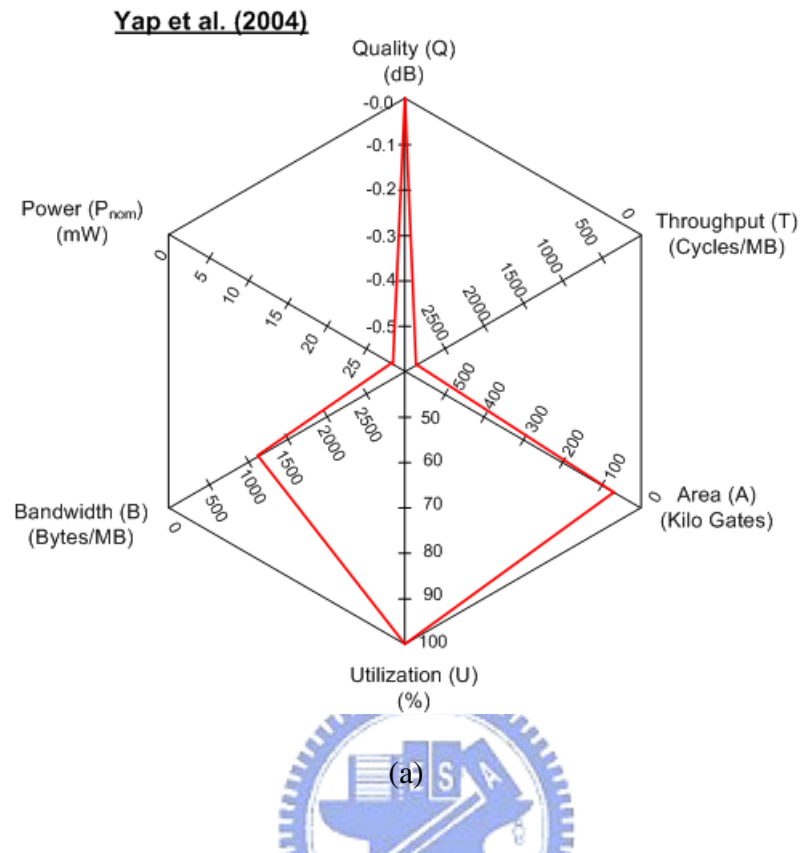


Figure 3.18: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67].

Table 3.9: Summary of cycles of data movement for P-frame and B-frame searches.

	P-Search	B-Search
Current Block	90	90
Reference Blcok	32	64
Binarized Data	11	11
Motion Vectors	3	6
Total	136	171

3.5.3 Bus Bandwidth Analysis

The bus bandwidth has been the bottleneck for modern SoC design, especially for data intensive designs such as video. The conventional 8-bit ME designs need whole search window to complete the motion search such as [36, 37, 39]. They proposed solutions to reduce gate counts or on-chip memory bandwidth, but not the bus bandwidth. In our case, the required data for movement are as follows and are also summarized in Table 3.13.

- Input current block data: $16 \times 16 \times 8$ bits.
- Input reference block data: for search range of $[-SR, +(SR - 1)]$.
 - $(16 + 2 \times SR) \times (16 + 2 \times SR) \times 8$ bits if search window reuse is not considered.
 - $16 \times (16 + 2 \times SR) \times 8$ bits if search window reuse is considered [42].
 - $16 \times (16 + 2 \times (SR + 2)) \times 8$ bits if one reuses search window and applies 8x8 block search with ± 2 range. If B-frame search is supported, it is doubled.
- Output motion vectors: 80 bits for 5 pairs of motion vectors if 16 bits are assumed for a pair of motion vectors

The total needed data movement for one block search is 8784 bits for P-search and 14336 for B-search under SR is 16.

Table 3.10: Performance comparison with state-of-the-art designs.

Design	Shen [39]	Huang [37]	Miyama [36]	ABME [35]	BBME
Standard	H.263+	MPEG-4	MPEG-4	MPEG-4	MPEG-4
Architecture	1-D systolic	Global elimination	Gradient search	Binary search	Parallel binary search
Cycle/MB	4096	5187	n.a. ¹	283	148
On-chip memory (kilo bits)	n.a.	24.08	40	9.80	8.64
PSNR loss (dB)	0.00	0.08	0.10	0.19	0.23
Search range	[-16, +15.5]	[-16, +15]	[-16, +15.5]	[-16, +15]	[-16, +15]
Process (μm)	0.60	0.35	0.13	0.18	0.18
Gate Count (kilo gates)	67	33.3	250	68.5	62.6
Power (mW) ²	353	149	2.5	2.2	0.76
Normalized power (mW)	4.12	12.6	15.5	2.2	0.76

Design	Song [43]	Ou [67]	Chen [26]	Yap [40]
Standard	MPEG-2	H.264/AVC	H.264/AVC	H.264/AVC
Architecture	Multi-resolution search	2-D systolic	Four step search	1-D systolic
Cycles/MB	210	1024	1136	4096
On-chip memory (kbits)	20	n.a.	64	n.a.
PSNR loss (dB)	n.a.	0.00	0.55	0.00
Search range	[-192.0, +191.5]	[-8, +7]	[-16, +15]	[-16, +15]
Process (μm)	n.a.	0.18	0.18	0.13
Gate count (kilo gates)	140	597	131.2	597
Power (mW)	n.a.	20.48	2.13/1.3V	95.04
Normalized power (mW)	n.a.	20.48	4.08	409.97

¹not available²For CIF 30 fps, GOP=IPPP..

Table 3.11: Design metrics evaluation for the state-of-the-art low power designs.

Designs	Design Metrics	
Shen [39]	Q (dB)	-0.00
	T (cycles/MB)	4209 ⁵
	A (logic,memory)	66.8/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	353/0.60
	P _{nom} (mW)	4.12
Huang [37]	Q (dB)	-0.08
	T (cycles/MB)	1784 ⁶
	A (logic,memory)	89.39/24.08
	U (%)	n.a.
	B (bytes/MB)	1024
	P (mW, μ m)	160/0.35
	P _{nom} (mW)	12.59
Miyama [36]	Q (dB)	-0.10
	T (cycles/MB)	568 ⁷
	A (logic,memory) ⁸	250/40
	U (%)	n.a. ⁹
	B (bytes/MB)	1024 ¹⁰
	P (mW, μ m)	2.5/0.13
	P _{nom} (mW)	15.53 ¹¹
Wang [35] (ABME)	Q (dB)	-0.19
	T (cycles/MB)	283
	A (logic,memory)	68.5/9.80
	U (%)	n.a.
	B (bytes/MB)	1086 ¹²
	P (mW, μ m)	2.2/0.18
	P _{nom} (mW)	2.20
BBME	Q (dB)	-0.23
	T (cycles/MB)	148
	A (logic,memory)	62.6/8.64
	U (%)	80.5
	B (bytes/MB)	486 ¹³
	P (mW, μ m)	0.763/0.18
	P _{nom} (mW)	0.763

Table 3.12: Design metrics evaluation for the state-of-the-art low power designs.

Designs	Design Metrics	
Yap [40]	Q (dB)	-0.00
	T (cycles/MB)	4096
	A (logic,memory)	61/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	95.04/0.13
	P _{nom} (mW)	409.97
Ou [67]	Q (dB)	-0.00
	T (cycles/MB)	1024
	A (logic,memory)	597/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	20.48/0.18
	P _{nom} (mW)	20.48
Chen [26]	Q (dB)	-0.00/-0.07/-0.55 (mode 1-3)
	T (cycles/MB)	1136 (mode 3) ¹⁴
	A (logic,memory)	131.2/64
	U (%)	n.a.
	B (bytes/MB)	1024 (mode 1), 256 (mode 2,3)
	P (mW, μ m)	16.72,4.83,2.13/0.18 (mode 1-3)
	P _{nom} (mW)	4.08 ¹⁵

¹⁴CIF 30fps@ 13.5MHz.¹⁵Voltage=1.3V.

Considering our proposed design, the required data for movement are summarized in Table 3.13. The input current block data are $(18 \times 18 \times 8)$ bits for BBME and $(30 \times 30 \times 8)$ for ABME. The input reference block data with a reuse scheme are $16 \times (16 + 2 \times SR) \times 1$ bits for LV3, $8 \times (8 + 2 \times \frac{SR}{2}) \times 1$ bits for LV2, and $4 \times (4 + 2 \times \frac{SR}{4}) \times 1$ bits for LV1. If B-frame search is supported, the total bits are doubled. For outputting binarized data as reference picture for the next frame, there are 4×4 bits for LV1, 8×8 bits for LV2, and 16×16 bits for LV3. The output motion vectors (MV) are 80 bits for the 5 motion vectors if 16 bits are used for a pair of motion vectors. For the B-frame search, the MV bits are doubled due to the forward and backward directions.

In our design, the required data movement for one MB search is 4,016 bits that is 45.7% of the bandwidth for the conventional 8-bit ME designs for the P-frame search. For the B-frame search, the data movement is 5,104 bits that is 32.9% of the bandwidth for conventional 8-bit ME designs. Table 3.13 summarizes the bandwidth analysis results. It shows that the bus bandwidth savings for the proposed design are 54.3% and 67.1% for P-frame and B-frame searches, respectively

3.5.4 Comparison of ABME and BBME

Table 3.14 summaries the feature difference between ABME [35] and BBME. ABME implements serial search architecture, but BBME uses parallel search architecture. BBME has five new important features including (1) MB based pre-processing (2) support of B-frame parallel search (3) parallel processing of 8x8 and 16x16 LV3 block search (4) shared processing units for each level to reduce the implementation cost (5) efficient LV2 search to reduce the processing latency. Experiments show BBME has significant performance

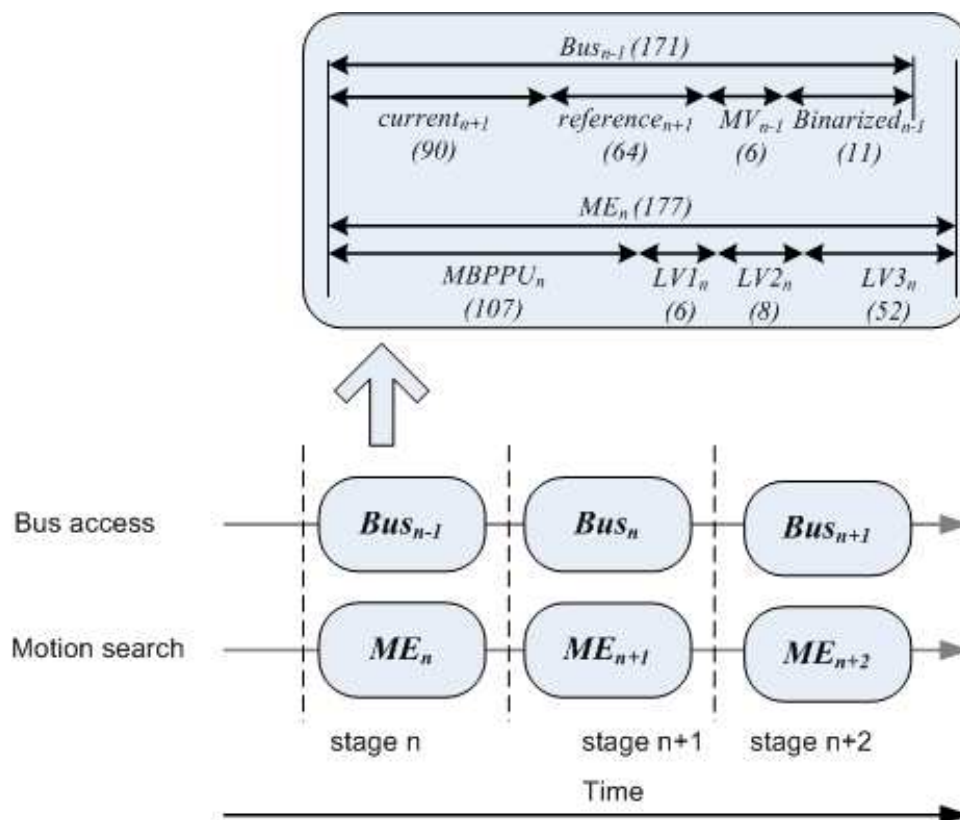


Figure 3.19: Pipeline timing of data movement and motion search.

Table 3.13: Bus bandwidth analysis for conventional 8-bit ME scheme and the proposed design (Search range = [-16, +15]).

Scheme	In/Out Data	Conventional 8-bit	AMBE[35]	BBME
P-frame	Current Block	2048	7200	2592
	Reference Block	6656	1032	1008
	Binarized Data	0	336	336
	Motion Vectors	80	80	80
	Total Bandwidth ¹	8784	8648	4016
	Percentage ²	100%	98.5%	45.7%
B-frame	Current Block	2048	7200	2592
	Reference Block	13312	2064	2016
	Binarized Data	0	336	336
	Motion Vectors	160	160	160
	Total Bandwidth	15520	9760	5104
	Percentage	100%	62.9%	32.9%

¹Required data in bits for one macroblock search = (current block + reference block + binarized data + motion vectors)

²Bandwidth of this work / Bandwidth of conventional 8-bit

improvement over ABME. The bandwidth savings are 52.8% and the power consumption is reduced by 1.44 mW. However, the BBME suffers from 0.1dB loss in quality.

The low power advantage is attributed to the following:

- The low complexity using binary SOD in ABME.
- Hardware efficient binary pyramid search structure, especially the hardware oriented modification for the LV2 search.
- Parallel processing of 8x8 and 16x16 LV3 block searches to minimize additional search cycles for 8x8 block search.
- Hardware support of B-frame parallel processing to reuse the current search block data and remove repeated on-chip memory access.

Table 3.14: Comparison of prior ABME design [35] and the proposed BBME design.

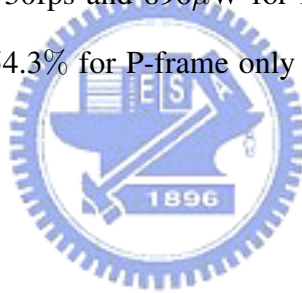
Feature	ABME[35]	BBME
Data structure	Binary pyramid	Binary pyramid
Pre-Processing	Frame Level	Macroblock Level
P-frame search	Spatially sequential	Spatially parallel for even and odd locations
B-frame Search	Temporally sequential for forward and backward directions	Temporally parallel for forward and backward directions
16×16 and 8×8 search	Temporally sequential	Temporally parallel
Hardware for each pyramid level	Separated SOD and control unit for each level	Shared SOD and comparator unit with separated control unit for each level
LV2 search	- Sequential search for six candidates - Redundant on-chip memory access for selected candidate	- Parallel search for adaptively select four candidates - No redundant on-chip memory access
PSNR loss (dB)	0.02-0.36	0.00-0.45
Power consumption (mW)	2.21	0.768
Bus bandwidth saving	1.5 %	54.3%
Gate count (kilo)	68.5	62.6

On the other hand, the high bandwidth efficiency is achieved by the following:

- Binary search structure to use binary data instead of 8-bit data for search.
- MB level pre-processing unit to reduce the amount of bus access for generation of the three levels of binary pyramid structure.
- Parallel processing of 8x8 and 16x16 LV3 block searches to save additional 17% bus bandwidth for search range of [-16, +15].

3.6 Summary

In this paper, we have proposed new motion estimation hardware architecture to achieve low power and high bandwidth efficiency. The proposed design is developed from a very low complexity motion estimation algorithm called all binary motion estimation [11]. It integrates several important features including (1) MB based pre-processing (2) support of B-frame parallel search (3) parallel processing of 8x8 and 16x16 LV3 block searches (4) shared processing units to reduce the hardware cost (5) efficient LV2 search to reduce the latency. We also analyze how low power and high bandwidth efficiency can be achieved with the proposed design. Experiments show that the power consumption can reach as low as $763\mu\text{W}$ for IPPPP CIF 30fps and $896\mu\text{W}$ for IPBPB CIF 30fps. The bus bandwidth saving can achieve up to 54.3% for P-frame only forward search and 67.1% for B-frame search.



Chapter 4

Power Adaptive Iterative Binary Search (PA-IBS)



4.1 Introduction

¹The power adaptive designs have become an important feature especially for portable video applications [59]. Unlike the low power designs that aim for minimized power consumption, the power adaptive design targets on the efficient allocation of power resources with equal video quality and longer battery life. In multimedia compression systems such as MPEG-1/2/4 and H.26x, the motion estimation (ME) that dominates the power consumption of the video encoder plays a key role in the power adaptive design. We will present a power adaptive ME design to improve power allocation and power efficiency.

In the power adaptive or complexity adaptive ME algorithms and designs [56, 57, 58, 47, 48, 49, 50, 45], we can roughly categorize them into two types according to their imple-

¹The authors would like to thank National Chip Implementation Center(CIC) for chip fabrication.(Chip No: T18-95E-04A)

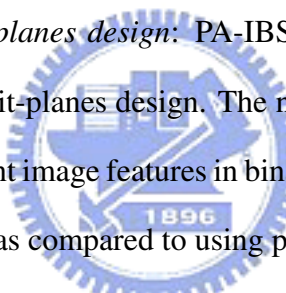
mentation methods. The first type is to achieve power adaptation by integration of multiple search strategies. This type adopts 2 to 3 search strategies such as three-step search, diamond search or full search to deliver different levels of search complexity. For example, the authors in [47] proposed a three-mode complexity adaptive method by using three-step search and enhanced four-step search for low power applications, and full search for high quality applications. Although this type of method can provide large scale of complexity differences, the coding quality for low power modes usually has significant quality loss.

The second type is to achieve power adaptation by simplified matching criterion. The simplified criterion include bit-depths truncation, pixel decimation, etc. By keeping different bit-depths or decimated pixel resolutions for block matching, the design can achieve different levels of computational complexity and power consumption. For example, the authors in [57, 58, 49] adopt the least-significant-bit truncation method to design their power adaptive ME. Pixel bit-depth of 1 or 2 is served for low power mode, and bit-depth of 8 is served for high quality mode. This type can provide the significant power reduction by dynamically adjusting the bit-depths, but it still suffers from significant quality loss in low power mode.

Table 4.1 summaries for the two types of complexity adaptive algorithms or power adaptive designs. Both of the methods have the significant quality loss in low power modes. The bit-depth truncation method also has the issues in limited pixel bit-depths and bit-plane dependency. Limited pixel bit-depths cause the difficulty for fine-granularity of power adaptation. Bit-plane dependency causes the inefficiency for data access and processing. To address these issues, a new power adaptive ME algorithm and hardware architecture called Power Adaptive-Iterative Binary Search (PA-IBS) is proposed with four key features:

Table 4.1: A summary of power adaptive motion estimation designs.

Methods	Type I	Type II
	Multiple search strategies	Simplified matching criteria
Examples	PMVFAST-EPZS (2 modes) [48]	Pixel sub-sampling [56]
	FS-3SS-E4SS (3 modes) [47] MV refinements [45]	Bit-depth truncation [57, 58, 49]
Pro and Con	Pro: (1) Large scale of complexity reduction Con: (1) Limited modes (2) Inflexibility for VLSI implementation (3) Significant quality loss	Pro: (1) Simple for VLSI implementation Con: (1) Limited pixel bit-depths (2) Significant quality loss (3) Bit-plane dependency

- 
Frequency decomposed bit-planes design: PA-IBS algorithm adopts the frequency decomposition method for bit-planes design. The new bit-plane design method generates directional and gradient image features in binary format, and can provide better rate-distortion performance as compared to using pixel bit-planes.
- Finer granularity of power adaptation:* The number of frequency decomposed bit-planes is not limited to pixel bit-depths. This allows finer granularity of power adaptation for smooth power and video quality adjustment.
- Independent bit-plane processing:* The frequency decomposed bit-planes can be individually stored in the memories and independently processed. Therefore, we can avoid unnecessary memory access and data processing to those unrelated bit-planes.
- Frequency scaling based hardware architecture:* The independent bit-plane processing provides the advantage to design the hardware for processing single bit-plane instead of all bit-planes. To full use this hardware design for single bit-plane process-

ing, the frequency scaling technique scales the working frequency with the number of bit-planes to be processed. Such hardware architecture reduces the overheads to design the hardware for the worst case of all bit-planes, and enhances the hardware utilization and power adaptation performance.

The remainder of this chapter is organized as follows. Section 2 describes our metric to measure the power adaptation performance. Section 3 describes the proposed power adaptive ME algorithm. The VLSI design issues for the proposed algorithm are addressed in Section 4. In Section 5, we show the hardware architecture, and its experimental results are demonstrated in Section 6. Section 7 gives the concluding remarks.

4.2 Power Adaptation Performance

An effective power adaptive ME design should deliver high power adaptation performance with lower or equal peak power consumption so as to achieve less energy consumption and higher power efficiency. Fig. 4.1 shows the power adaptation curves to measure the power adaptation performance for prior arts [56, 57, 58]. Each curve contains multiple points to represent the complexity-power relationship at different power modes.

In this paper, the power adaptation performance is defined as the ratio of Power Adaptation Ratio (PAR) to Complexity Adaptation Ratio (CAR). The PAR represents the power increasing percentage and the CAR represents the complexity increasing percentage. The reference point for PAR and CAR measurement is (0, 0). A higher PARCAR ratio indicates a poor power adaptation performance due to the overhead for extra power consumption. Moreover, the CAR is defined as the ratio of adapted operation counts to the total operation

counts. The CAR specifies complexity adaptation level in percentage and it is measured in data operation counts or instruction counts. For example, a 4:1 sub-sampling motion search has the CAR of 14. To provide a fair comparison basis for various hardware designs with different peak power consumption, we use the PAR as opposed to the actual power consumption. The PAR is defined as the ratio of adapted power consumption to peak power consumption for a specified CAR. A lower PAR indicates lower power increasing percentage and better power adaptation performance. In Fig. 4.1, the prior arts [56, 57, 58] show the higher PARCAR which means more overheads in their designs. The reasons to cause higher PARCAR can be summarized as follows:

- Redundant bus access from the external memory that retrieves unnecessary pixels for all CARs.
- Redundant search window data access from the on-chip memory due to inefficient data access strategy. A sizable portion of the retrieved data is unused for execution in the low CAR cases.
- Serious hardware idling problems for the hardware masking based approach.

To address these issues, we propose a new power adaptive ME algorithm and hardware architecture called Power Adaptive Iterative Binary Search (PA-IBS). The algorithm adopts a new bit-plane design solution by frequency decomposition to extract multiple image features for pattern matching. Each bit-plane can be stored and processed individually so as the redundant bus access and data computation to unused bit-planes can be saved. The algorithm also provides the hardware design the large flexibility to adopt different working frequency to the iterations used for search. Different working frequency can keep our

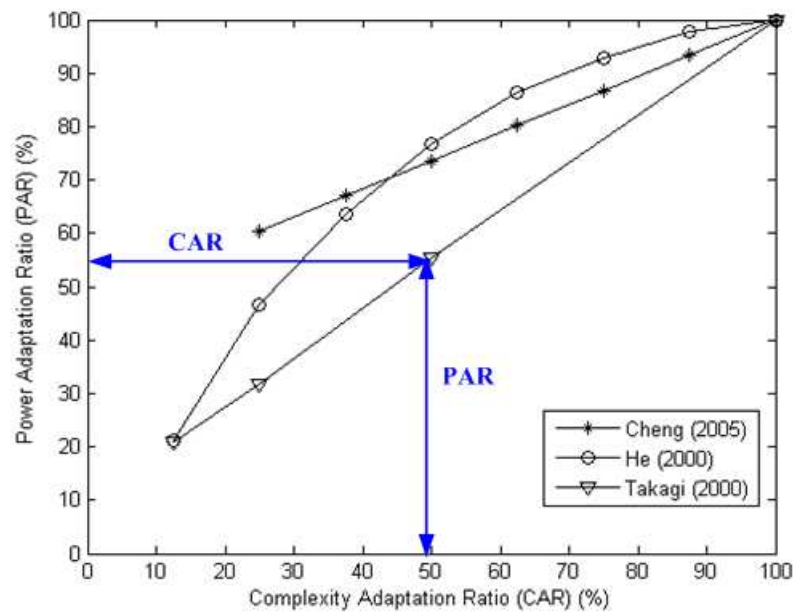
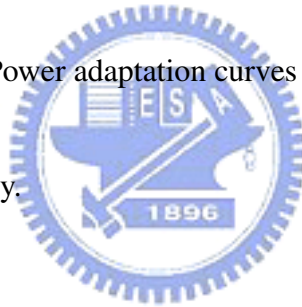


Figure 4.1: Power adaptation curves for prior arts [56, 57, 58].

hardware design 100% busy.



4.3 Power Adaptive Iterative Binary Search (PA-IBS) Algorithm

The PA-IBS contains the IBS algorithm to achieve power adaptive ME and the CAM to select the target iterations for IBS on an MB basis. Fig. 4.2 shows the functional block diagram of a generic video encoder system with the PA-IBS. The PA-IBS contains three major modules: (1) Binary Image Preprocessor (BIP) (2) Iterative Binary Search (IBS) (3) Content Adaptive Mechanism (CAM). The current video frame is sent to the BIP module to generate binary images stored in the frame buffer as the reference frame of the next current

frame. The IBS module processes total f iterations of binary bit-plane searches, where f represents the target iterations decided by CAM.

4.3.1 Binary Image Preprocessor (BIP)

The BIP is used to generate the frequency-decomposed binary bit-planes for search. The preprocessing operation in [41] provides us a good reference to generate single bit-plane by extracting the image feature in binary format, but this method is only limited to single bit-plane design. The BIP improve the bit-planes design by considering the directional and gradient image features. We adopt totally Φ filters to generate Φ binary bit-planes, in which each binary bit-plane represents one kind of directional or gradient image feature to provide more precise of pattern matching for motion search. In our design, Φ is not limited to pixel bit-depths, but the number of filters to apply for finer granularity of power adaptation.

The preprocessing operation is similar to [41], but the filters design is different. We design Φ FIR filters to generate Φ binary bit-planes. The k -th binary bit-plane $\hat{I}_k(x, y)$

$$\hat{I}_k(x, y) = \begin{cases} 1 & \text{if } FIR_k(I(x, y)) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

is generated by applying k -th FIR filter (FIR_k). The adopted filters shown in Table 4.2 are two-dimensional (2-D) 2nd or 3rd order of FIR filters due to the complexity issue.

4.3.2 Iterative Binary Search (IBS)

The IBS contains multiple iterations of binary searches, and each of the iterations is a full search on binary bit-planes. The authors in [60, 61, 62, 63] provide a good reference

for binary bit-plane of full search, but their method is limited to single bit-plane. Our algorithm can execute multiple iterations of binary bit-plane searches, and then link the number of iterations to different power consumption levels.

Fig. 4.3 shows the processing procedure for the IBS algorithm. The BIP module generates Φ binary images by frequency-decomposed filters. Then, the weighted prediction (w_k) is applied to adjust the weightings between the Φ binary images to achieve better motion search results. The number of iterations (ϕ) to be executed in IBS is decided by content adaptive pre-analysis mechanism (CAM) on the macroblock (MB) basis. The ϕ iterations of block matching results in Sum of Difference (SOD) are accumulated as the final search results for the decision of motion vectors. In our experiments in later section, Φ is set as eight to provide a similar comparison basis for the 8-bit full search.

The block matching criterion for multiple iterations of binary search is SOD accumulation

$$SOD(\phi) = \sum_{k=1}^{\phi} SOD_k = \sum_{k=1}^{\phi} \left(\sum_{x=0}^{15} \sum_{y=0}^{15} \left(\hat{I}_k^C(x, y) \oplus \hat{I}_k^R(x + x_0, y + y_0) \right) \cdot w_k \right). \quad (4.2)$$

The SOD accumulation is a function of which means we accumulate iterations of binary bit-plane matching results. For each of binary bit-plane search, we compare the current block $\hat{I}_k^C(x, y)$ and the reference block $\hat{I}_k^R(x + x_0, y + y_0)$ at the k -th binary bit-plane by XOR operation. The SOD operations can provide equal results as Sum of Absolute Difference (SAD) when image data is in binary format, but XOR operation is much simpler than subtraction and absolute operations.

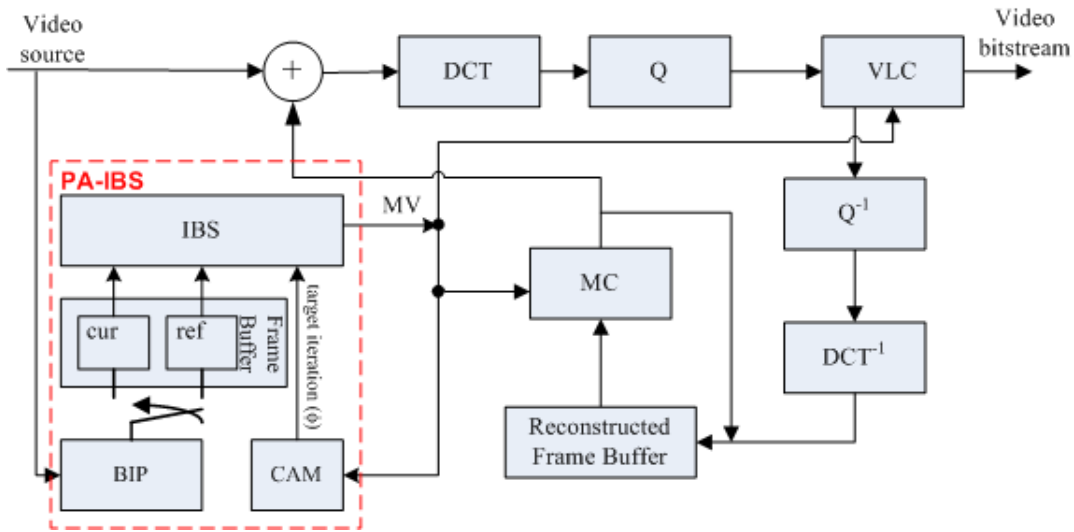


Figure 4.2: Functional block diagram of a generic video encoder with power adaptive-iterative binary search (PA-IBS).

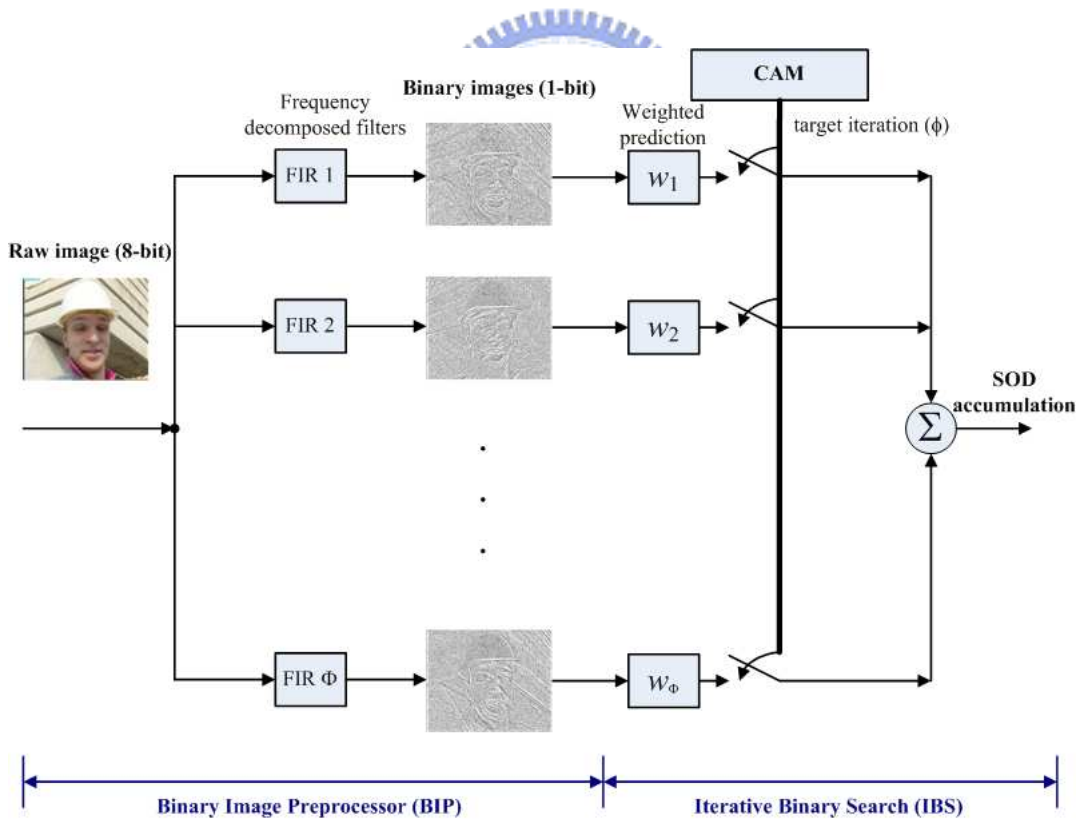


Figure 4.3: Processing procedure of the power adaptive-iterative binary search (PA-IBS) algorithm.

Table 4.2: Filters used to generate binary images.

k -th binary image	Filter Types	2-D Filter Coefficients
1	3×3 Laplacian operators	[1 1 1; 1 -8 1; 1 1 1]
2	3×3 Sobel operators along 0 degree	[1 0 -1; 2 0 -2; 1 0 -1]
3	3×3 Sobel operators along 90 degree	[1 2 1; 0 0 0; -1 -2 -1]
4	3×3 Sobel operators along 135 degree	[1 1 -2; 1 -2 1; -2 1 1]
5	3×3 Sobel operators along 45 degree	[-2 1 1; 1 -2 1; 1 1 -2]
6	3×3 Laplacian operators (diagonal)	[0 0 0; 1 -3 1; 0 1 0]
7	3rd order of high pass filter (horizontal)	[0 0 0 0; 0 0 0 0; -1 3 -3 1; 0 0 0 0]
8	3rd order of high pass filter (vertical)	[0 0 -1 0; 0 0 3 0; 0 0 -3 0; 0 0 1 0]

4.3.3 Content Adaptive Mechanism (CAM)

The CAM dynamically adjusts the target iterations (ϕ) according to video contents so as to achieve a better power allocation and power adaptation performance. For example, a complicated video scene needs more power to provide an accurate search for significant quality improvement. On the other hand, a coarse search with less power is sufficient for a simple scene such as slow motion to achieve similar performance. To select ϕ , we need to find a relationship between the activities in video content and ϕ . To provide more accurate prediction for the activities in video content, this decision criterion is implemented on an MB basis.

Our MB based decision criterion measures the deviation of motion vectors from the neighboring MBs denoted as φ and such criterion is used to map from φ to ϕ . The parameter φ is defined in eq. (4.3) where it measures the average difference between the horizontal

and vertical motion predictors from the top, top right and left MBs.

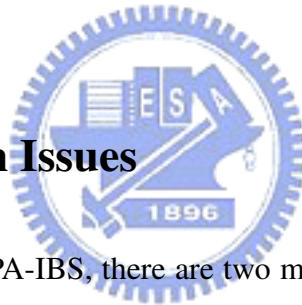
$$\varphi = \sum_{i \in x,y} \left(\left| MV_i^{\text{top}} - MV_i^{\text{top right}} \right| + \left| MV_i^{\text{top}} - MV_i^{\text{top left}} \right| \right) / 2 \quad (4.3)$$

Then the ϕ is determined by eq. (4.4), where T_k are decision thresholds and $T_0 = 0$, $T_9 =$ maximal allowable motion vectors in the design.

$$\phi = k, \text{ if } T_k \leq \varphi < T_{k+1} \text{ where } 1 \leq k \leq 8 \quad (4.4)$$

These decision thresholds T_k are defined by the users according to the search range and maximal number of iterations Φ .

4.4 Hardware Design Issues



For VLSI implementation of PA-IBS, there are two major design issues: power adaptation and SOD accumulation for the target number of iterations. The first issue of power adaptation is addressed by exploring hardware techniques so that the number of iterations corresponds to the power consumption level. The second issue of SOD accumulation for the target number of iterations is addressed by exploring hardware solutions to reduce the sizable memory required by the SOD accumulation.

4.4.1 Power Adaptation

To provide a link between the number of iterations and the power consumption level, there are two possible solutions, clock gating and frequency scaling [64].

- *Clock gating*: Clock gating is a commonly used technique to halt hardware operations and to save power consumption. The hardware with clock gating should be able to turn off the clock to stop power consumption before it is awaked.
- *Frequency scaling*: Frequency scaling is the other widely used technique coupled with the voltage scaling technique to achieve power adaptation. The system changes the operating frequency to complete operations of different complexity within the same time period. Thus, different power level is achieved.

If PA-IBS is implemented with the clock gating technique, a constant frequency clock is designed to meet the requirement of ϕ binary searches and the clock is turned off when ϕ binary searches are done. However, the hardware is idling for the remaining 7/8 of total operation cycles if only one iteration is executed. On the other hand, if PA-IBS is implemented with the frequency scaling technique, we can adjust the working frequency according to the target iterations (ϕ) as shown in eq. (4.5).

$$f_{\text{work}} = \frac{\phi}{\Phi} \cdot f_{\text{input}} = \frac{\phi}{8} \cdot f_{\text{input}} \quad (4.5)$$

The frequency scaling technique has fewer hardware idling cycles and higher hardware utilization. Thus, the frequency scaling technique is selected in our design.

4.4.2 SOD Accumulation

To implement SOD accumulation for the target iterations as shown in eq. (4.2), the hardware requires sizable on-chip memories to buffer the SOD from each search. For H.264/AVC [55], the ME that supports up to 7 search block sizes needs at least 16 4x4

SODs buffered for further SOD computation of larger block sizes. Suppose the 16 4×4 SODs for one search location takes M bits, the whole search window for search range of ± 16 needs a sizable buffer with $(32 \times 32 \times M)$ bits. The size increases with the search range.

To reduce the on-chip memory size, a region based search strategy is proposed to partition the original wider search window into several smaller search regions. The buffer size is limited to the region size rather than the whole search window size. The final motion results are selected region by region. Thus, each region search can reuse the same hardware.

The first design issue for the region search architecture is to select the region size. The region size affects the buffer size and the search locations for parallel processing. Table 4.3 summarizes the required hardware and memory access bandwidth for four region sizes. If region size of 16×16 is selected, for each search location at most 4 regions are needed and the on-chip memory access bandwidth is 256×4 bits. To allow maximal memory access efficiency, we need to design 256 SOD processing elements (PE) in order to complete the 16×16 region search in parallel. If region size of 8×8 is selected, 9 regions and 64 SOD PEs are needed. If the smaller region size is picked, the hardware design cost and memory access bandwidth is reduced. However, the overall processing cycles are increased and vice versa. The 8×8 region size is selected to meet the requirement of CIF 30 fps or higher frame rates. It achieves the processing throughput with the smallest hardware design area.

To support the 8×8 region search, the whole search window is partitioned into several 8×8 regions. Fig. 4.4 exemplifies how the search window with search range of ± 16 is partitioned into 36 8×8 blocks. For each 8×8 region search, we need to check totally 64 search locations for an MB that covers nine 8×8 blocks. So, 9 memory banks and the 64-bit

Search window for search range of +/-16

	-16	+0	+16	+32		
-16	00z	10z	20z	01z	11z	21z
+0	30z	40z	50z	31z	41z	51z
+16	60z	70z	80z	61z	71z	81z
+32	02z	12z	22z	03z	13z	23z
	32z	42z	52z	33z	43z	53z
	62z	72z	82z	63z	73z	83z

Region index : xyz
 x - bank index (x = 0~8)
 z - iteration index (z = 0~7)
 (4*z+y) - word index (y =0~3)

Figure 4.4: Partitioned 8×8 regions for z -th binary search window under search range of ± 16 .

data depth are designed. In Fig. 4.4, we label each partitioned 8×8 region with region index xyz . The index xyz represents the region belonging to the binary image for z -th iteration of search, and is stored in the $(y + 4 \times z)$ -th word of the x -th memory bank. The region search is processed region by region in a raster scanning order. Table 4.4 summarizes the region indexes in Fig. 4.4 and their associated memory banks. There are 9 banks of $4 \times 8 \times 2 \times 64$ two-port register files for LM_REF where the factor 4 is for the index y , the factor 8 is for the index z and the factor of 2 is for the ping-pong buffers. There are 4 banks of $8 \times 2 \times 64$ two-port register files for LM_CUR where the factor 8 is for the index z and the factor of 2 is for the double ping-pong buffer. The total memory size is 40 kilo bits.

4.4.3 Binary Image Preprocessor

Preprocessor is designed with ME module in BBME due to the significant bus bandwidth savings, but will be put outside ME module in PA-IBS. The reasons are: (1) PA-IBS

Table 4.3: Analysis of hardware requirement for four region sizes.

Region size	Number of regions for one point search	Memory access for one point search	Number of SOD PE
16×16	4	$256 \times 4 = 1024$	256
8×8	9	$64 \times 9 = 576$	64
8×4	15	$32 \times 15 = 480$	32
4×4	25	$16 \times 25 = 400$	16

Table 4.4: Memory index for the partitioned regions in search range of ± 16 ($z =$ iteration index from 0 to 7). Each word is 64 bits.

Bank index	No of words	The stored region index ($z=0-7$)
0	4×8	00z, 01z, 02z, 03z
1	4×8	10z, 11z, 12z, 13z
2	4×8	20z, 21z, 22z, 23z
3	4×8	30z, 31z, 32z, 33z
4	4×8	40z, 41z, 42z, 43z
5	4×8	50z, 51z, 52z, 53z
6	4×8	60z, 61z, 62z, 63z
7	4×8	70z, 71z, 72z, 73z
8	4×8	80z, 81z, 82z, 83z

Table 4.5: Evaluation of the pre-processing unit design schemes for PA-IBS.

Scheme	Iterations	Pipeline	Bus Bandwidth (Bytes/MB)		Pipeline Cycles (NC_{system})
1	$\phi=8$	MEU/BIP: Frame MEU/MEU: MB	E-MEM→BIP	256 ¹	NC_{MEU}
			BIP→E-MEM	256	
			BIP→MEU	256	
2	$\phi=8$	MEU/BIP: MB MEU/MEU: MB	E-MEM→BIP	256	$NC_{MEU} + NC_{BIP}$
			BIP→E-MEM	256	
			BIP→MEU	0	
3	$\phi=1$	MEU/BIP: Frame MEU/MEU: MB	E-MEM→BIP	256	NC_{MEU}
			BIP→E-MEM	256	
			BIP→MEU	32 ²	
4	$\phi=1$	MEU/BIP: MB MEU/MEU: MB	E-MEM→BIP	256	$NC_{MEU} + NC_{BIP}$
			BIP→E-MEM	256	
			BIP→MEU	0	

$$^1((16 \times 16)/8) \times (\phi=8)$$

$$^2((16 \times 16)/8) \times (\phi=1)$$

has longer worst processing cycles to cause longer pipelining stages, (2) the bus bandwidth saving is not significant in low CAR (or small ϕ) case.

In Table 4.5, we analyze 4 design schemes for the BIP module. Scheme 1 is PA-IBS with $\phi = 8$, and such a design scheme puts BIP out of MEU to avoid longer latency and redundant bus transmission. Scheme 2 is with separated BIP and MEU for $\phi = 8$. The required bus transmission for Scheme 2 is reduced to 66.7%², but the pipeline cycles is longer. Scheme 3 and 4 have similar analysis as Scheme 1 and 2 for $\phi = 1$. The bus saving from Scheme 3 to Scheme 4 is 5.9%³ which is very small.

From the worst case scenario, the pipeline cycles is more critical than bus bandwidth saving. Hence, unlike low power ABME design in Chapter 3, the PA-IBS implements frame level BIP instead of MB BIP.

$$^2(256+256)/(256+256+256) \times 100\% = 66.7\%.$$

$$^3(256+256)/(256+256+32) \times 100\% = 94.1\%.$$

4.5 PA-IBS Hardware Architecture

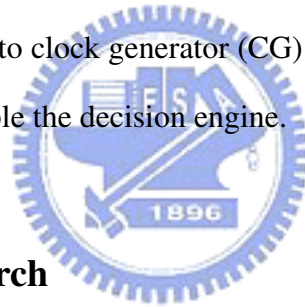
The PA-IBS architecture is a hardware-efficient power adaptive design based on both the IBS algorithm in Section 4.3 and the design approaches in Section 4.4. The design contains a low complexity binary matching architecture to reduce the peak power consumption, and the frequency scaling technique for power adaptation to prevent hardware idling.

4.5.1 System Architecture

The system architecture of PA-IBS as shown in Fig. 4.5 contains several on-chip memories for current and reference search data, a 8×1 line search engine to realize region search, pipelined buffers to store the accumulated SODs, and several control units for data and search flow control. The current and reference image data for ME are stored in two sets of local buffers, LM_CUR and LM_REF, respectively. The LM_CUR contains four banks (C0-C3) to store the current block data where each bank is a 16×64 two-port register file. The LM_REF contains nine banks (S0-S8) to store the reference data where each bank is a 64×64 two-port register file. The current and reference block data are received via memory interface (MEM_IF). For each access of search data according to the current 8×8 region search, the LM_CUR outputs 16×16 bits and LM_REF outputs 24×24 bits into two registers arrays, REG_CUR and REG_REF. The region search is realized by implementing 8 lines of 8×1 line searches. The 8×1 line search engine designed for low complexity binary matching gets 256 bits from REG_CUR and 16×24 bits from REG_REF for the parallel 8×1 binary search each cycle. For IBS with multiple iterations, the hardware accumulates SODs from each iteration and stores it in the pipelined buffers (PB0-PB7). The line search engine needs totally eight cycles to complete one 8×8 region search for each iteration, and

$(8 \times \phi)$ cycles for ϕ iterations. The pipelined buffers are designed to store the accumulated SODs. There are totally eight lines of 8×1 line search results needed for buffering in one 8×8 region search, so eight pipelined buffers are used.

Once the search engine completes one region search, the search locations are output from the vector generator (VG) to the decision engine for the final motion vectors selection using the information from both search locations and its associated SOD. The controller (CTRL) is the control center of the whole motion estimation design. It makes sure the decision engine meets the input timing of the SOD results from the pipelined buffers, controls the address generator (AG) to output the address for memory access, clears the accumulated SODs in the pipelined buffers at the beginning of each region search, sends the parameter of the target iterations (ϕ) to clock generator (CG) for appropriate working frequency, and controls the timing to enable the decision engine.



4.5.2 8×1 Line Search

The 8×1 line search engine realizes the 8×8 region search line by line. To avoid a huge amount of data access and hardware requirement for parallel processing, the 8×8 region search is divided into 8 lines of 8×1 line searches and is processed line by line for smooth data access. Using this architecture, we only need to design for the smallest search unit of one 8×1 line search. Parallel processing of 8 search locations takes only one cycle since the pattern matching criterion uses simple XOR operations for binary images. It takes eight cycles to complete one 8×8 region search.

The line search engine is designed to process in parallel the eight search locations for the 8×1 line search. As shown in Fig. 4.6, the search engine inputs 16×16 bits from reg-

ister array REG_CUR and 16×24 bits from REG_REF for each 8×1 searches. For each location in the 8×1 line search, the block matching criterion is 256-bit SOD operation using the same 16×16 bits from REG_CUR and different 16×16 bits from the 16×24 bits as the search window of the 8×1 line search. Each 256-bit SOD operation is further partitioned into 16 4×4 SOD computations since the smallest search block size in H.264/AVC is 4×4 . Then, the accumulated 16 4×4 SODs are stored in the pipelined buffers before outputting to the decision engine. The SOD results for 4×8 to 16×16 block size are computed by summing the 16 4×4 accumulated SODs in the decision engine to select final motion vectors.

4.5.3 Pipelined Buffers



The pipelined buffers are designed to store accumulated SODs for multiple iterations of IBS. We design eight sets of 1024-bit pipelined buffers to store the accumulated SODs from each 8×1 line search. For multiple iterations of searches, the pipelined buffers store 8 lines of SODs from each 8×1 line search to accumulate the SODs from each search. After all the iterations of searches are done, the pipelined buffers output the final SOD sums to the decision engine. Fig. 4.7 shows the data output timing from the 8×1 line search engine to the pipelined buffers for ϕ iterations of binary searches. If ϕ equals to one, the final SOD is generated from cycle 9 to cycle 16. If there are multiple iterations, the SODs for row 0 are accumulated with row 0 of next iteration at cycle 9. The final SODs for the eight lines of searches are generated at cycle $(8 \cdot \phi + 1)$ to cycle $(8 \cdot \phi + 8)$.

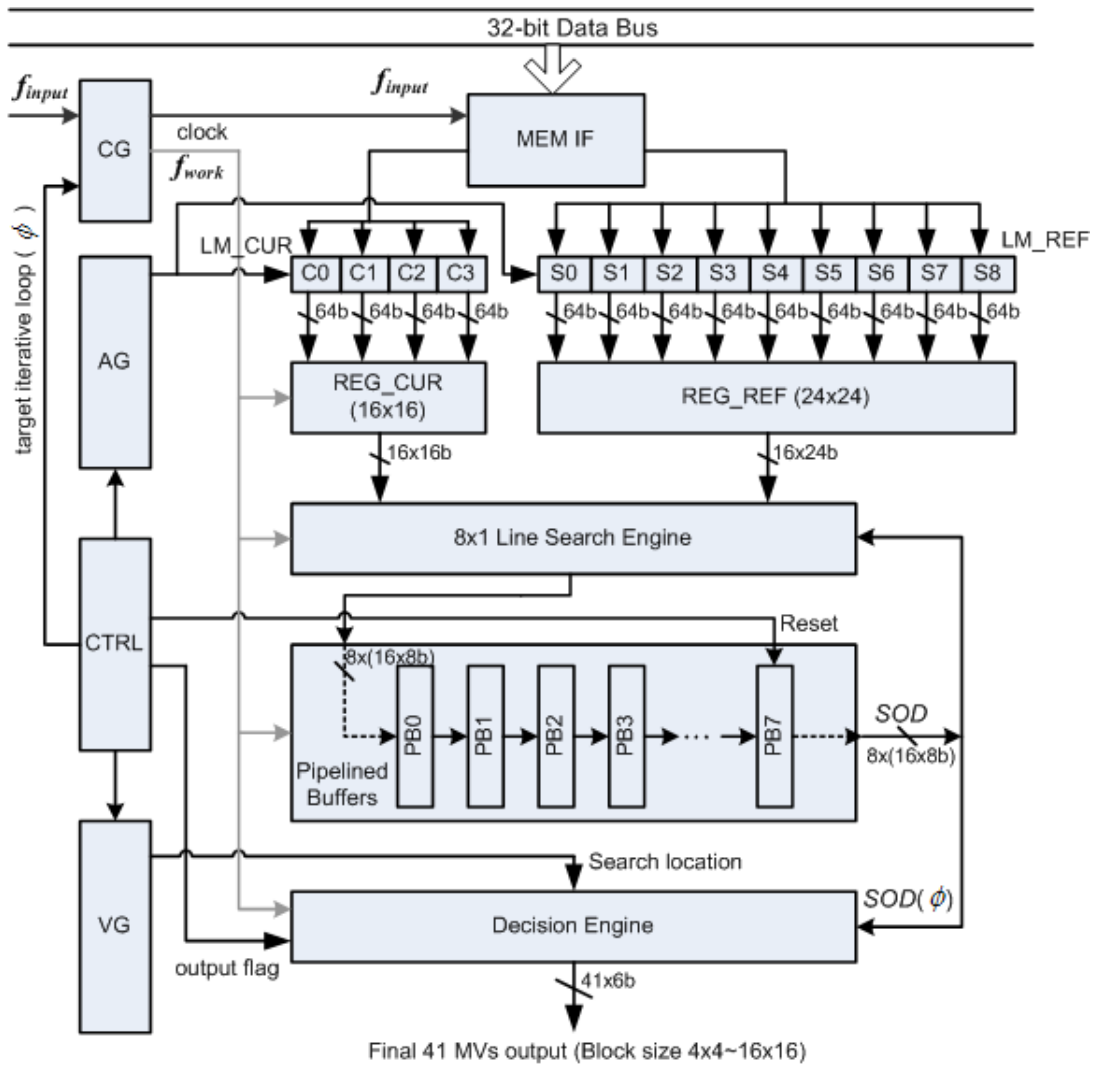


Figure 4.5: Block diagram of PA-IBS architecture.

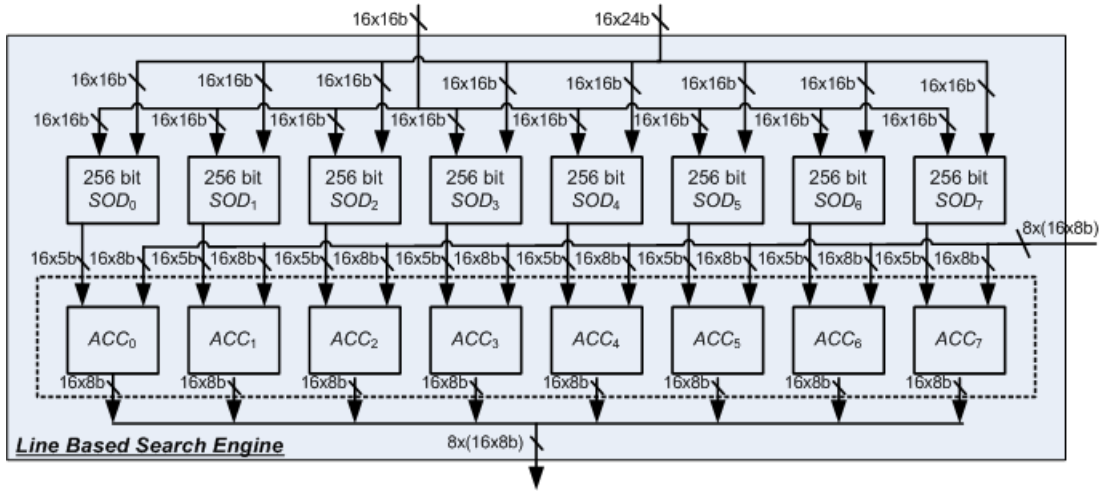


Figure 4.6: Architecture of line based search engine.

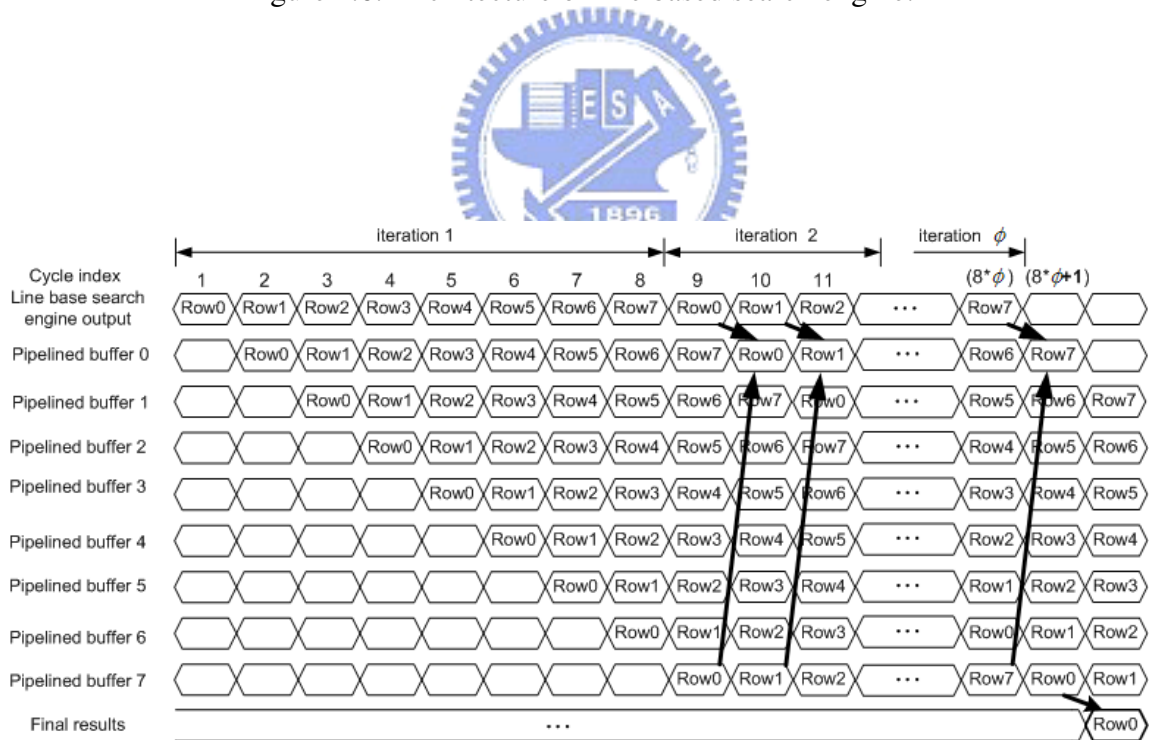


Figure 4.7: Output data timing from line based search engine to the pipelined buffers for ϕ iterations of binary searches.

4.6 Experimental Results and Analysis

4.6.1 Evaluation of Algorithmic Performance

The performance evaluation for PA-IBS without CAM demonstrates better PSNR performance as compared to prior power adaptive algorithms. The PA-IBS with CAM experiment demonstrates the further complexity reduction in software or power reduction in hardware for similar video quality and coding bit rate.

A. R-D Performance for PA-IBS without CAM

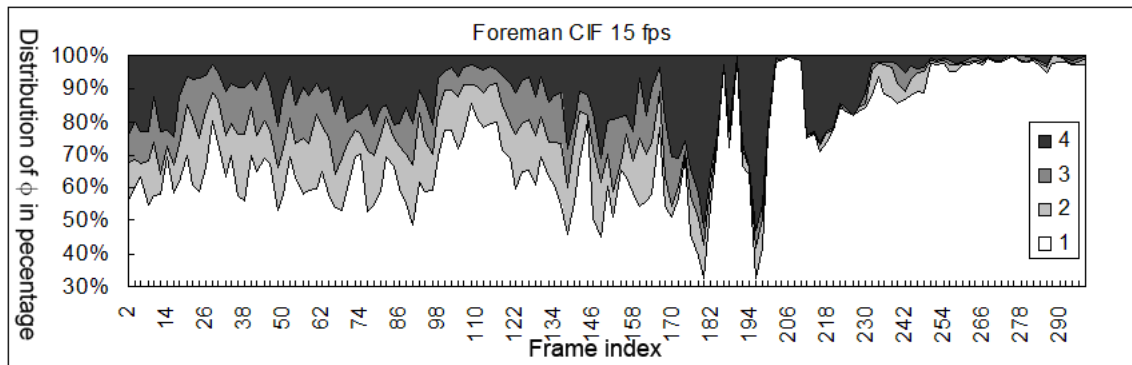
Table 4.6 summaries the rate-distortion (R-D) performance for IBS, Least Significant Bit (LSB) truncation [57, 58], and sub-sampling (SUB) algorithm [56] which are implemented on H.264/AVC reference software JM8.6. Six common MPEG test sequences including Foreman, Akiyo, Mobile, Flower, Tempete, and Football, are used for this test with the GOP structure of one I-frame and 149 P-frames. One frame skipping is applied in this test and the results are compared to the JM software with full search. From this Table, PSNR loss for IBS with $\phi = 1$ (or CAR=1/8) ranges from 0.15 to 0.45 dB. For $\phi = 2$ (or CAR=2/8), the PSNR loss ranges from 0.10 to 0.30 dB. When $\phi > 2$ is applied, at most 0.2 dB PSNR loss is observed.

To compare with LSB truncation and sub-sampling methods, the IBS at CAR equal to 1/8 has 0.20-0.45dB PSNR gain over LSB truncation, and has up to 0.15 dB PSNR gain over the SUB method. At CAR equal to 2/8, IBS is 0.1-0.2dB better than LSB truncation, and has similar performance as the SUB method. At CAR greater than 2/8, the difference between these algorithms becomes minor.

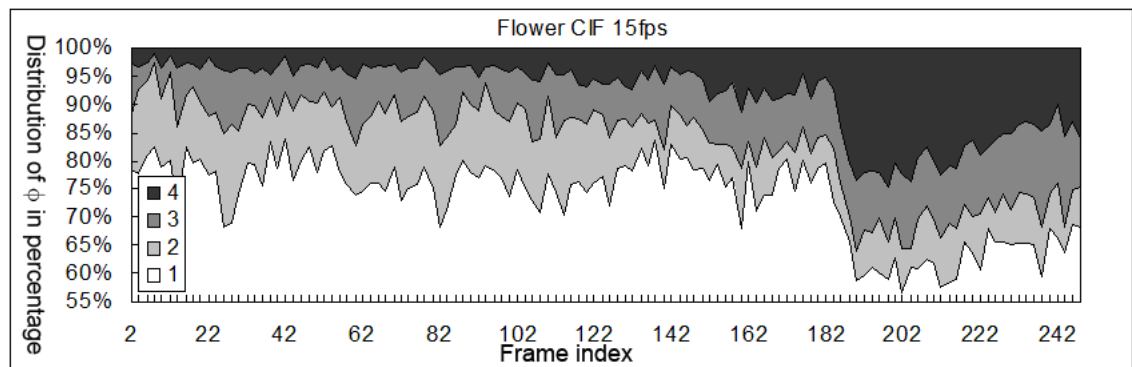
B. R-D Performance for PA-IBS with CAM

The CAM adapts the target iterations to activities in video contents for better power allocation. Table 4.6 shows the experimental results for PA-IBS with CAM. The first issue of the CAM is to decide the T_k values defined in Section 4.3.3. From Table 4.6, we can find the PSNR loss for ϕ greater than 4 is minor. So, only $\phi \leq 4$ are adopted in this test. With search range of ± 16 , we let $T_1 = 0$, $T_2 = 4$, $T_3 = 8$, $T_4 = 16$, $T_5 \sim T_9 = 16$. Table 4.7 lists the number of iterations for one MB search, bit rate and average PSNR for the six sequences. For Foreman sequence, the CAM method uses 1.57 iterations on the average for one MB search and shows 21.5% complexity reduction over the PA-IBS without CAM for similar encoding bit rate and PSNR performance. For the Akiyo sequence, the reduction ratio is more significant because it has very slow motion or no motion in the most area. So similar search results can be achieved using fewer iterations with CAM and a total of 49% in complexity reduction is observed. For the other sequences such as Mobile&Calendar, Flower&Garden, Tempete, and Football, there are 36.5%, 25%, 37%, and 18% complexity reduction on the average, respectively.

The CAM can dynamically adjust the target number of iterations ϕ on an MB basis according to the spatial or temporal video characteristics. Fig. 4.8 shows the temporal distribution of the target iterations (ϕ) for Foreman and Flower&Garden sequences. In Fig. 4.8(a), we use larger ϕ in the scene change area from frame 180 to 220 to gain the significant coding bits saving (up to 20 kilo bits) as shown in Fig. 4.9(b) with minor PSNR performance loss (less than 0.1dB) as shown in Fig. 4.9(a). In Fig. 4.8(b), most area in Flower&Garden sequence has uniform global motion and their motion activities in the neighboring MBs are similar. Thus, smaller ϕ is used to achieve similar PSNR quality



(a)



(b)

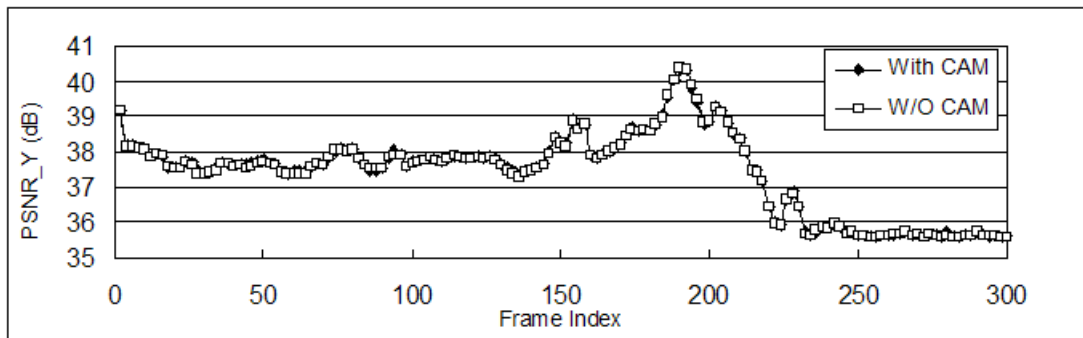
Figure 4.8: Temporal distribution of the target number of iterations (ϕ) for the PA-IBS algorithm. (a) Foreman with 300 frames (b) Flower&Garden with 250 frames.

with the significant complexity reduction. However, after frame 180 the motion activities become larger and non-uniform, so larger ϕ is used to achieve better search results.

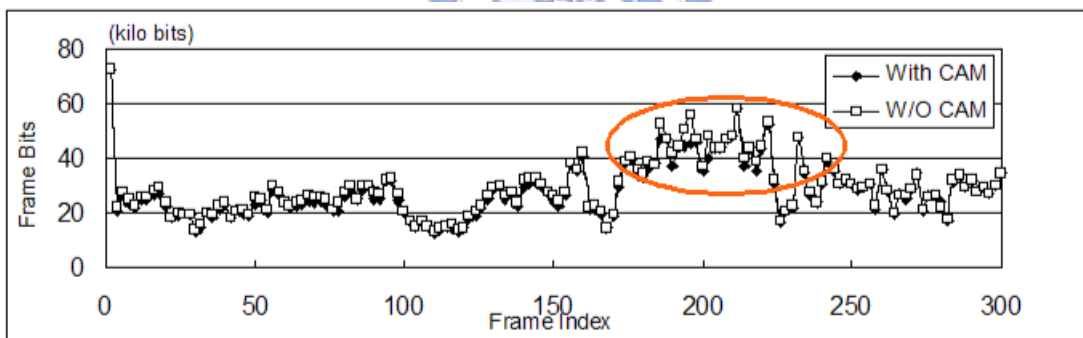
4.6.2 Evaluation of Hardware Performance

A. Chip Specification

Table 4.8 lists the chip specification for the PA-IBS hardware design and the chip layout is shown in Fig. 4.10. This design is synthesized with Artisan TSMC 0.18 μm cell library



(a)



(b)

Figure 4.9: Temporal distribution of PSNR and coding bits for the PA-IBS algorithm with and without the CAM on Foreman sequence.

Table 4.6: R-D performance for three power adaptive algorithms as compared to full search (LSB = LSB truncation, SUB = sub-sampling, IBS = iterative binary search).

Complexity Adaptation Ratio (CAR)		1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
Foreman	LSB	-0.80	-0.50	-0.35	-0.25	-0.15	-0.10	-0.05	0.00
	SUB	-0.40	-0.25	-0.15	-0.15	-0.15	-0.10	-0.05	0.00
	IBS	-0.45	-0.30	-0.20	-0.20	-0.15	-0.10	-0.10	-0.05
Akiyo	LSB	-0.35	-0.20	-0.05	-0.05	-0.05	-0.05	-0.05	0.00
	SUB	-0.15	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05	0.00
	IBS	-0.15	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05
Mobile	LSB	-0.50	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	SUB	-0.30	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05	0.00
	IBS	-0.25	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05
Flower	LSB	-0.90	-0.30	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	SUB	-0.60	-0.30	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	IBS	-0.40	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05
Tempete	LSB	-0.40	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	SUB	-0.30	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	IBS	-0.25	-0.15	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05
Football	LSB	-0.50	-0.30	-0.10	-0.05	-0.05	-0.05	-0.05	0.00
	SUB	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05	0.00
	IBS	-0.30	-0.20	-0.10	-0.05	-0.05	-0.05	-0.05	-0.05

Table 4.7: Complexity reduction for PA-IBS algorithm with CAM as compared to PA-IBS without CAM.

Method		PA-IBS with- out CAM	PA-IBS with CAM	Complexity Reduction
Foreman	Average No. of iterations	2	1.57	21.5%
	Bit rate (kbps)	419	418	-
	PSNR (dB)	37.46	37.42	-
Akiyo	Average No. of iterations	2	1.02	49.0%
	Bit rate (kbps)	76	77	-
	PSNR (dB)	40.76	40.76	-
Mobile	Average No. of iterations	2	1.27	36.5%
	Bit rate (kbps)	1237	1243	-
	PSNR (dB)	35.40	35.38	-
Flower	Average No. of iterations	2	1.50	25.0%
	Bit rate (kbps)	1237	1248	-
	PSNR (dB)	36.25	36.20	-
Tempete	Average No. of iterations	2	1.26	37.0%
	Bit rate (kbps)	961	970	-
	PSNR (dB)	36.24	36.19	-
Football	Average No. of iterations	2	1.50	25.0%
	Bit rate (kbps)	1094	1017	-
	PSNR (dB)	37.48	37.47	-

and Synopsys Design Compiler. The hardware cost is 217 kilo gates. Total 40 kilo bits on-chip memories are used to store current and reference search patterns. Total 2,101 input frequency cycles are needed for search range of ± 16 . The required frequency to run CIF 30 fps is 24.95 MHz. The power consumption measured by Agilent SOC93000 is 27.50, 23.04, 19.04, 16.29, 14.31, 11.45, 7.22 and 4.21 mW for $\phi = 8$ to $\phi = 1$, respectively.

B. Evaluation on Bus Bandwidth

Bus bandwidth is typically the bottleneck in modern VLSI design, especially for video applications. In the proposed design, the required data for bus access is proportional to the target iterations for IBS. For single iteration of binary search with range of ± 16 , it needs 16×16 bits data for the current search block and 48×48 bits data for the search window. If ϕ iterations of searches are applied, the bus access bandwidth will be $((16^2 + 48^2) \times \phi)$ bits for each MB search. In the prior power adaptive designs such as LSB truncation [57, 58] or SUB methods [56], it is necessary to retrieve all data for the search due to the data access strategy required by the algorithm. Our proposed method only needs $\phi/8$ and saves $8 - \phi/8$ of bus access. Fig. 4.11 shows the relationship between required bus bandwidth for different CAR cases. To make comparison among different architectures, the following assumptions are made when we claim the CAR is $h/8$ for our references. For the IBS the target number of iterations is h . For the LSB truncation method, only the h MSB bits are kept. For the SUB algorithm, the $8 : h$ sub-sampling is assumed. The maximal saving ratio of data transmission can achieve up to 87.5% when $\phi = 1$ or CAR=1/8.

C. Evaluation on Power Adaptation Performance

The power adaptation curve is shown in Fig. 4.12. As compared to the prior power adaptive designs, the PA-IBS has the best power adaptation performance defined as PAR/CAR. For $CAR = 1/8$, the PA-IBS can improve the power adaptation performance from 1.68 to 1.17 with around 50% overhead reduction as compared to the approaches in [57, 58]. For $CAR = 2/8$, the proposed design can improve the power adaptation overhead by 19%-125% as compared to the works in [56, 57, 58]. For $CAR = 4/8$, the improvements range from 8% to 50%. The improvement is more significant for small CAR, but is similar for large CAR. We summarize the comparison results with the prior power adaptive designs in terms of PSNR loss, bus access bandwidth, and power adaptation performance in Table 4.9.

D. Evaluation on Peak Power Consumption

Table 4.10 summarizes the design information for the prior H.264/AVC ME designs [65, 66, 40, 67]. The work in [65] implements a fast algorithm of three levels of multi-resolution search. The works in [66, 40, 67] are developed based on full search but with different search architecture, including parallel tree architecture, one-dimensional systolic array, and two-dimensional systolic array architectures. In Table 4.10, our proposed work can deliver minimal peak power consumption as compared to other designs.

For the comparison of power adaptive hardware designs, the one with minimal peak power consumption and better power adaptation performance delivers highest power efficiency. The PA-IBS has the minimal peak power consumption as shown in Table 4.10. The PA-IBS with better power adaptation performance as shown in Fig. 4.12 can also achieve better power efficiency as compared to other power adaptive designs.

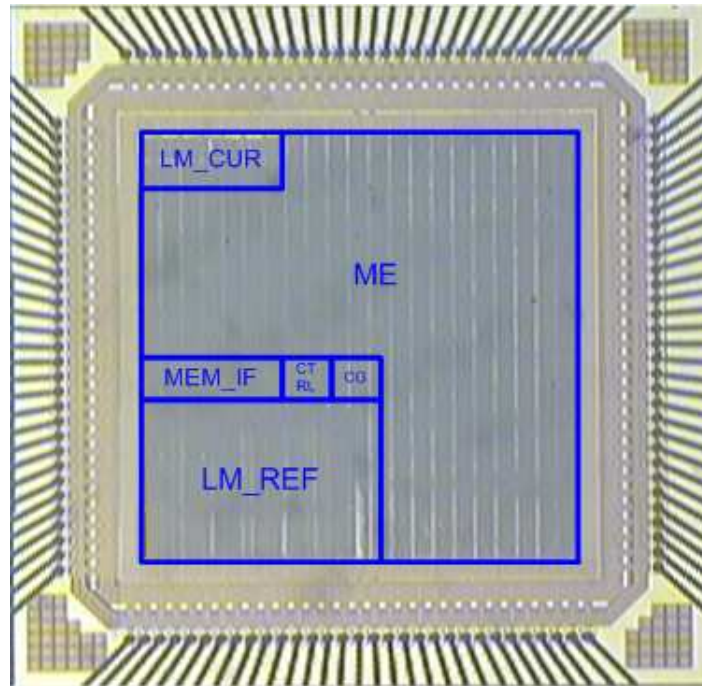


Figure 4.10: Chip photo.

Table 4.11 shows the design metrics evaluation for the H.264/AVC designs in Table 4.10. The hexagon plots are shown in Fig. 4.13 and Fig. 4.14. From this Table and these plots, we can find PA-IBS and Chen's work are the best two designs in this comparison from overall evaluations. However, PA-IBS is better in terms of quality (Q), bandwidth (B), throughput (T), a little worse in silicon area (A), and power (P), ties in utilization (U).

E. Evaluation on Power-Distortion Performance

Power-Distortion performance is another key parameter to evaluate the power adaptive designs. A effective power adaptive design is able to deliver best quality under the same power consumption or minimal power consumption under the same quality. This also means a effective power adaptive design should have the best Power-Distortion (P-D) curve

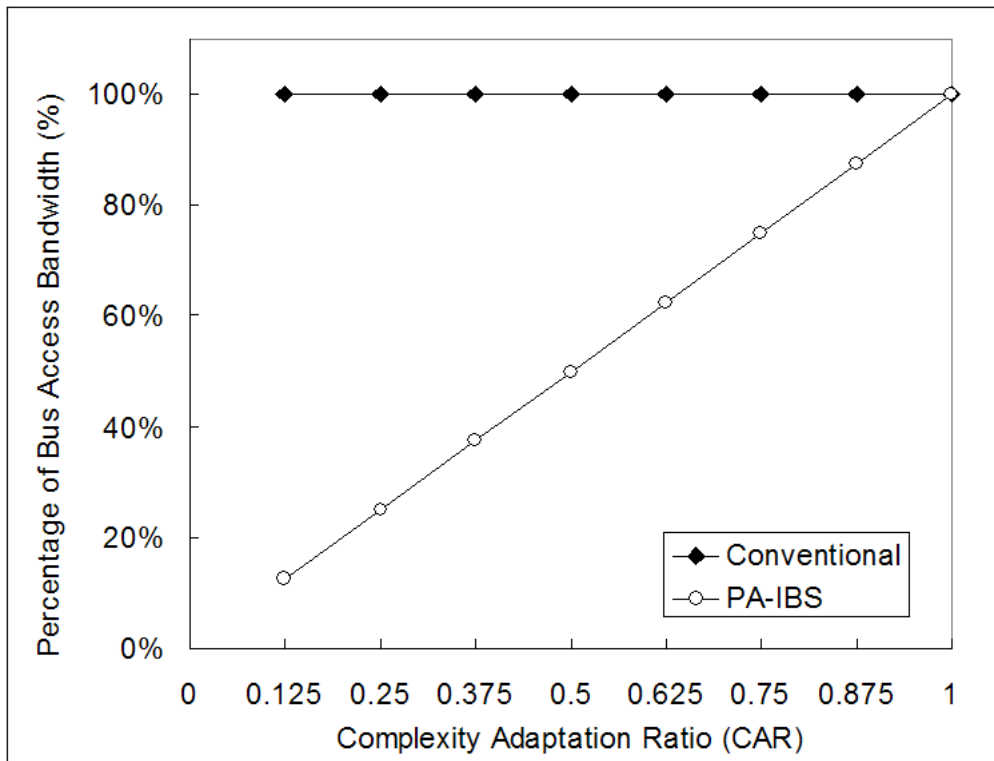


Figure 4.11: Comparison of bus bandwidth for the PA-IBS and conventional power adaptive designs with the hardware masking approach.

Table 4.8: Chip design specification of PA-IBS.

Process	TSMC 0.18 μ m 1P6M
Package	CQFP 128
Area	2766 \times 2766 μ m ²
Logic gate count	217 kilo gates
Chip frequency	CIF 30fps: 24.95 MHz for \pm 16
Memory	9 banks of 64x64 two-port register files 4 banks of 16x64 two-port register files Total 40 kilo bits
Power consumption	4.21 mW for 1 iteration (CIF 30fps) 27.50 mW for 8 iterations (CIF 30fps)
Max throughput	38.6 fps running at 32MHz

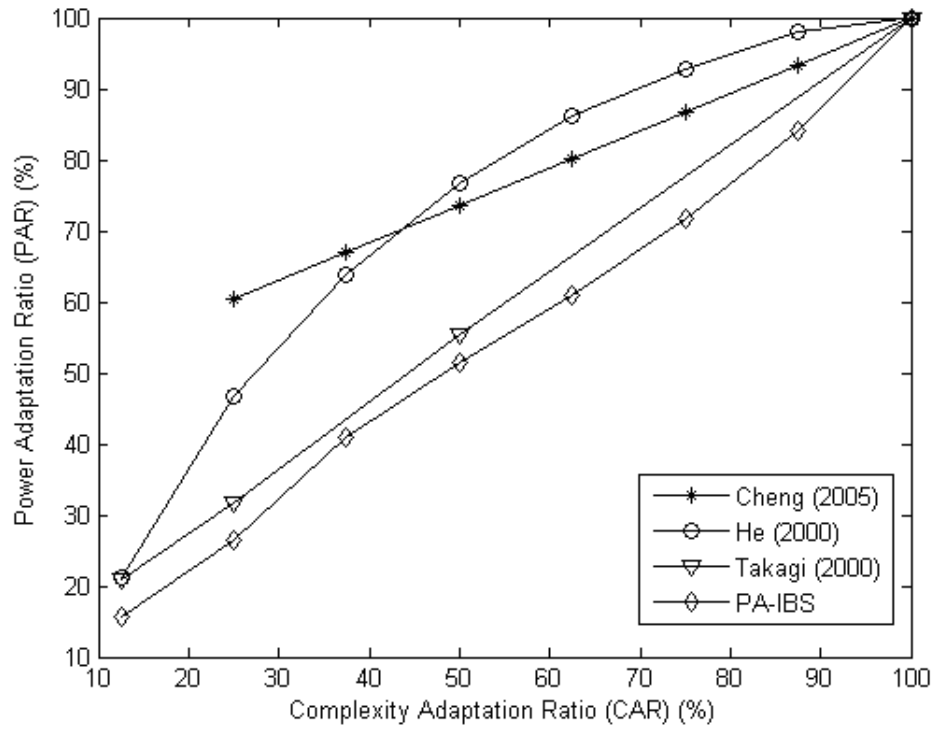


Figure 4.12: Comparison of the power adaptation performance for the PA-IBS and prior power adaptive designs.

Table 4.9: Comparison of the power adaptive designs.

Design	Cheng[56]	He [57]	Takagi [58]	This work
PSNR loss (dB)	0.05-0.60	0.05-0.90	0.05-0.90	0.05-0.45
Power adaptive technique	Hardware masking	Hardware masking	Hardware masking	Frequency scaling
Power adaptation performance (PAR/CAR)	1.00-2.42	1.00-1.86	1.00-1.68	1.00-1.17
Is hardware busy for all CARs?	No	No	No	Yes
Bus transmission bandwidth scalable	No	No	No	Yes

Table 4.10: Comparison of H.264/AVC ME hardware designs.

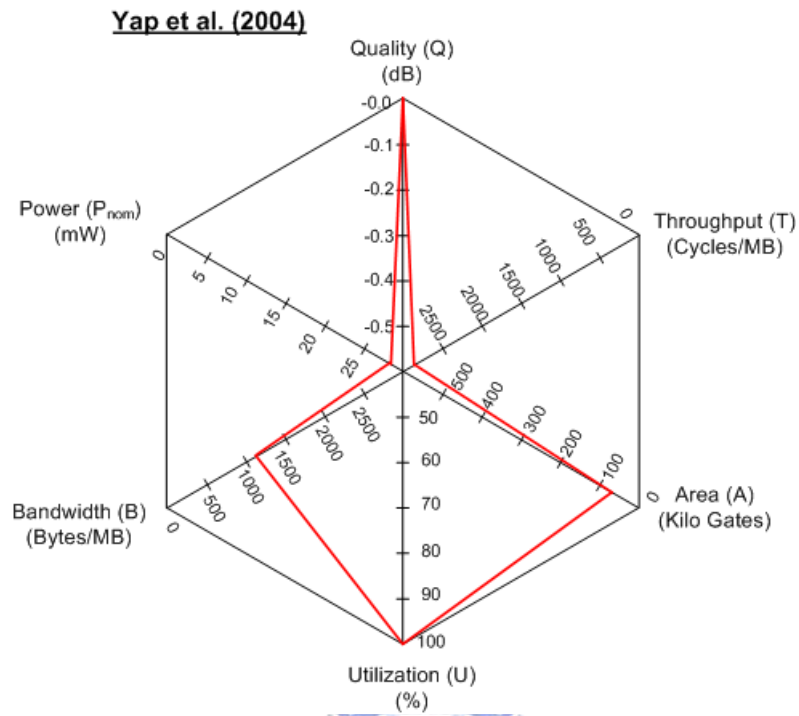
Architecture	multi-resolution search [65]	Parallel trees full search [66]	1-D systolic full search [40]	2-D systolic full search [67]	PA-IBS
Cycles/MB ¹	180	1024	4096	1024	1024
On-chip SRAM (kilo-bits)	N/A	208	N/A	N/A	104
Gate count (kilo)	N/A	330.2	61	597	217
Power (mW) ²	N/A	N/A	95.04	20.48 ³	22.54
Process (μm)	N/A	0.18	0.13	0.18	0.18
Search range	$\pm 64H \times \pm 32V$	$\pm 64H \times \pm 32V$	$\pm 16H \times \pm 16V$	$\pm 8H \times \pm 8V$	$\pm 16H \times \pm 16V$
Search precision	Integer-pel	Integer-pel	Integer-pel	Integer-pel	Integer-pel

¹Cycles for search range of $\pm 16H \times \pm 16V$ ²Power for CIF 30fps³Power for search range of $\pm 8H \times \pm 8V$

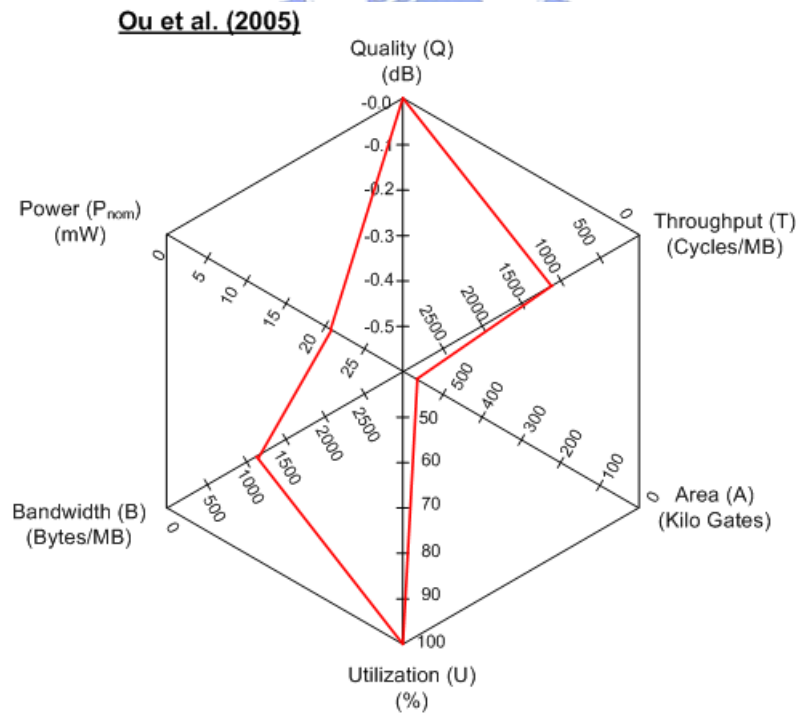
Table 4.11: Evaluation of design metrics for low power designs (Group 3).

Designs	Design Metrics	
PA-IBS	Q (dB)	-0.30 (CAR = 1/8)
	T (cycles/MB)	128 (CAR = 1/8)
	A (logic,memory)	217/104
	U (%)	100
	B (bytes/MB)	128 (CAR = 1/8)
	P (mW, μ m)	4.21 (CAR = 1/8)
	P _{nom} (mW)	4.21
Yap [40]	Q (dB)	-0.00
	T (cycles/MB)	4096
	A (logic,memory)	61/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	95.04/0.13
	P _{nom} (mW)	409.97
Ou [67]	Q (dB)	-0.00
	T (cycles/MB)	1024
	A (logic,memory)	597/n.a.
	U (%)	100
	B (bytes/MB)	n.a.
	P (mW, μ m)	20.48/0.18
	P _{nom} (mW)	20.48
Chen [26]	Q (dB)	-0.00/-0.07/-0.55 (mode 1-3)
	T (cycles/MB)	1136 (mode 3) ⁴
	A (logic,memory)	131.2/64
	U (%)	n.a.
	B (bytes/MB)	1024 (mode 1), 256 (mode 2,3)
	P (mW, μ m)	16.72,4.83,2.13/0.18 (mode 1-3)
	P _{nom} (mW)	4.08 ⁵

⁴CIF 30fps@13.5MHz.⁵Voltage=1.3V.

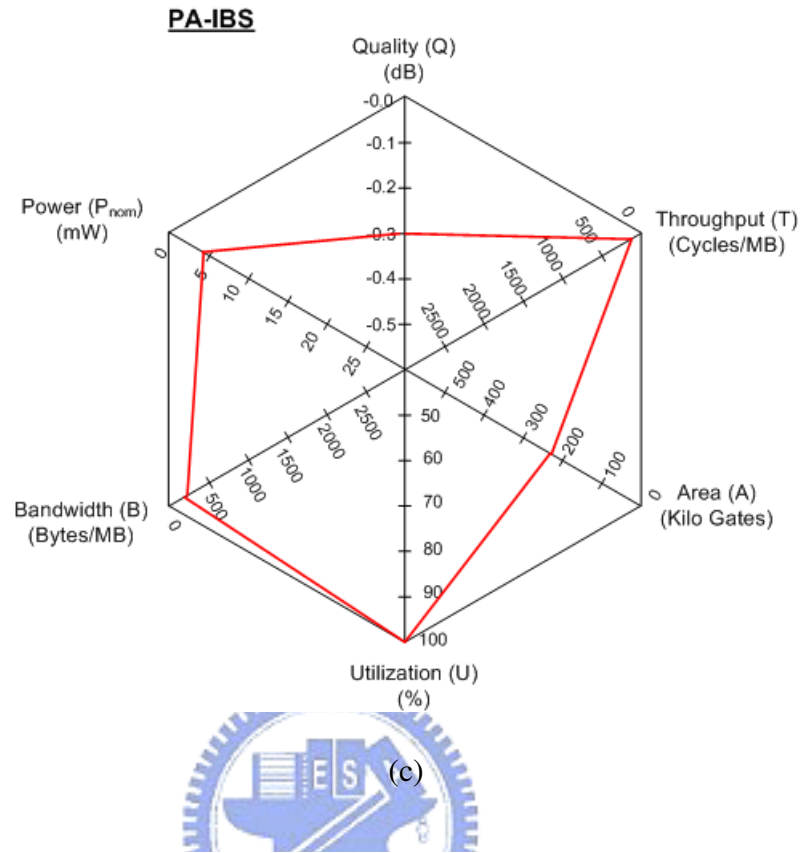


(c)

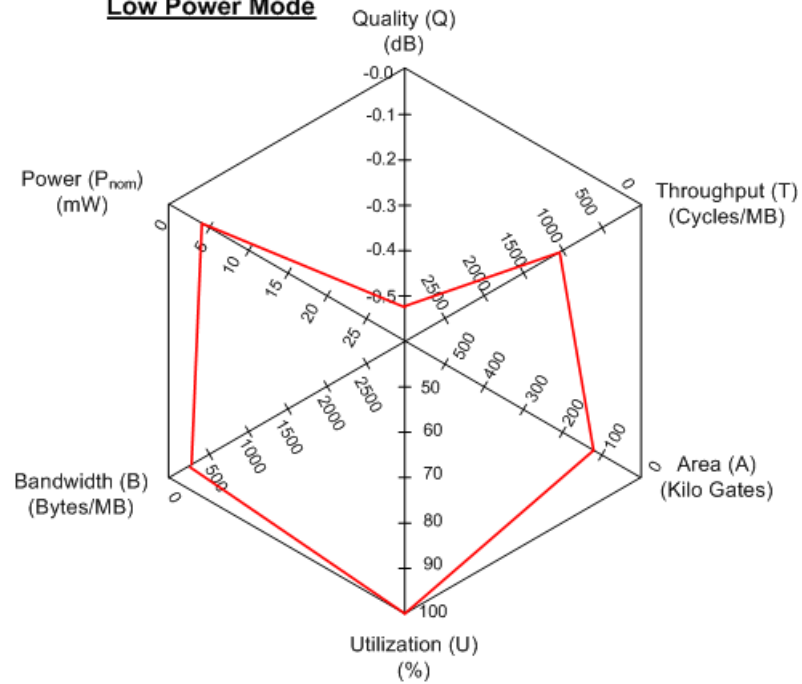


(d)

Figure 4.13: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) Yap's work [40] (b) Ou's work [67].



Chen et al. (2007) Ultra Low Power Mode



(d)

Figure 4.14: Hexagonal plot of 6 design metrics for different ME hardware architectures evaluation. (a) PA-IBS (CAR=1/8) (b) Chen’s work [26].

behavior as compared to the other power adaptive designs.

Table 4.6.2 summarizes the power in mW and distortion in PSNR loss for our PA-IBS, Chao *et al.*'s work [30], and Chen *et al.*'s work [26]. Chao *et al.*'s work is for MPEG-4. In his design, there are three power adaptive modes: (1) Successive Elimination Algorithm (SEA) which is one kind of fast full search, (2) SEA with early cut (at cycle 4208 to meet CIF 30fps at 50MHz), (3) The third one is DS. The third mode is for lowest power consumption, while the first mode is for highest quality. The chip operates at the three modes and switches according to the content adaptive detection. Chen *et al.*'s work is a H.264/AVC IME design with three operating modes: (1)high quality mode, (2)low power mode, and (3) ultra low power mode. The first mode is a full search with 2 reference frames, while the third mode is four-step search(4SS) with single reference frame. Running on the third mode, the power consumption is 2.13 mW for CIF 30 fps with 13.5 MHz clock frequency. For the power adaptive designs such as [56, 57, 58] we have mentioned in Section 4.6.2-B, they are not included in this Table due to the lack of exact power consumption data in mW or its manufacture process.

Fig. 4.15 shows the P-D performance for PA-IBS, Chao *et al.*'s work [30], and Chen *et al.*'s work [26]. PA-IBS has the best P-D performance at lower power stages, while Chen *et al.*'s work has best P-D performance at higher power stages. Chao *et al.*'s work is the worst in this comparison.

4.7 Summary

In this chapter, we proposed a new power adaptive ME IP core design called Power Adaptive Iterative Binary Search (PA-IBS) to improve power efficiency and longer battery

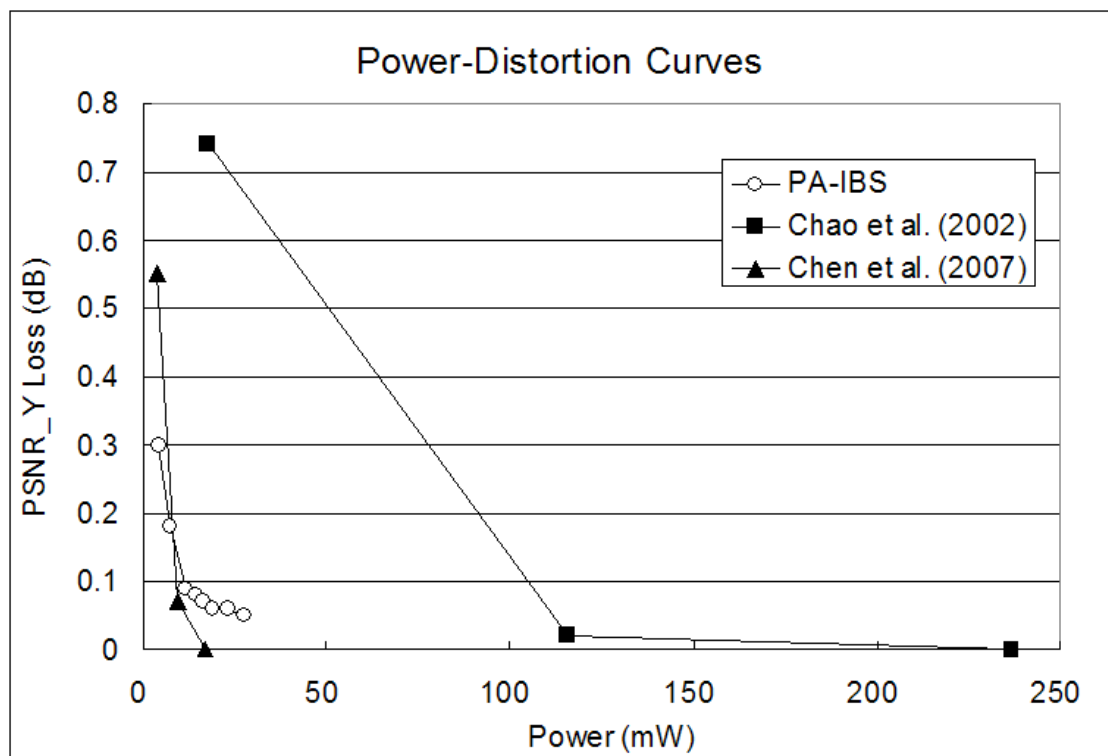


Figure 4.15: Power-Distortion curves for power adaptive designs.

Table 4.12: Power-Distortion performance for power adaptive designs.

Designs	Power ¹		Distortion
	Power Stage	Power (mW)	PSNR Loss (dB)
PA-IBS	1	4.21	0.3
	2	7.33	0.18
	3	11.45	0.09
	4	14.31	0.08
	5	16.29	0.07
	6	19.04	0.06
	7	23.04	0.06
	8	27.50	0.05
Chao [30]	1	17.6	0.74
	2	115.83	0.02
	3	236.85	0.00
Chen [26]	1	4.08	0.55
	2	9.26	0.07
	3	16.72	0.00

$$^1\text{Normalized Power} = \text{Power}(\text{original}) \times \left(\frac{0.18}{\text{Process}}\right)^2 \times \left(\frac{1.8}{\text{Voltage}}\right)^2$$

life. We first analyzed the design overheads in the prior power adaptive ME designs with the hardware masking approach. Then, the proposed power adaptive solution reduces those design overheads with a new ME algorithm called Iterative Binary Search (IBS) and hardware architecture called Power Adaptive IBS (PA-IBS). The IBS integrates a new frequency decomposed bit-plane design method to improve the rate-distortion curve and provide the flexibility for finer granularity of power adaptation. The IBS also executes the multiple bit-plane searches in an either individual or accumulated manner, thus redundant bus and on-chip memory access are eliminated. The PA-IBS hardware implements the frequency scaling technique to adapt the number of iterations for different power consumption levels. It reduces hardware idling and enhances hardware utilization.

Experiments show the PA-IBS can deliver lower peak power consumption, better power

adaptation performance and lower bus bandwidth requirement. As for the peak power consumption, the PA-IBS can deliver as low as 27.50 mW power consumption for CIF 30 fps and it has lower peak power consumption as compared to other H.264/AVC ME designs. As for the power adaptation performance, the PA-IBS has the best power adaptation performance from 1.00 to 1.17 as compared to the prior power adaptive designs in [56, 57, 58]. The PA-IBS can also improve the hardware idling problem and save up to 87.5% in bus bandwidth requirements.



Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have presented two algorithm and architecture designs of motion estimation for different power constrained video coding applications, and showed the better low power and power adaptive characteristics as compared to prior works.

The first power constrained application is for mobile terminals which have the demands of the low power consumption features. Hence, we proposed a new motion estimation hardware architecture to achieve low power and high bandwidth efficiency. The proposed design is developed from a very low complexity motion estimation algorithm called all binary motion estimation [11]. It integrates several important features including (1) MB based pre-processing (2) support of B-frame parallel search (3) parallel processing of 8x8 and 16x16 LV3 block searches (4) shared processing units to reduce the hardware cost (5) efficient LV2 search to reduce the latency. We also analyze how low power and high bandwidth efficiency can be achieved with the proposed design. Experiments show that

the power consumption can reach as low as $763\mu\text{W}$ for IPPPP CIF 30fps and $896\mu\text{W}$ for IPBPB CIF 30fps. The bus bandwidth saving can achieve up to 54.3% for P-frame only forward search and 67.1% for B-frame search.

The second power constrained application is for portable devices which have the demands of different video quality and power consumptions under different battery status. Hence, we proposed a new power adaptive ME IP core design called Power Adaptive Iterative Binary Search (PA-IBS) to improve power efficiency and longer battery life. We first analyzed the design overheads in the prior power adaptive ME designs with the hardware masking approach. Then, the proposed power adaptive solution reduces those design overheads with a new ME algorithm called Iterative Binary Search (IBS) and hardware architecture called Power Adaptive IBS (PA-IBS). The IBS adopts eight iterations of binary searches and can save unnecessary bus access and on-chip memory access. The PA-IBS hardware implements the frequency scaling technique to adapt the number of iterations for different power consumption levels. It reduces hardware idling and enhances the hardware utilization.

The experiments show the PA-IBS can deliver lower peak power consumption, better power adaptation performance and lower bus bandwidth requirement. As for the peak power consumption, the PA-IBS can deliver as low as 22.5 mW power consumption for CIF 30 fps and it has lower peak power consumption as compared to other H.264 ME designs. As for the power adaptation performance, the PA-IBS has the best power adaptation performance from 1.00 to 1.17 as compared to the prior power adaptive designs in [56, 57, 58]. The PA-IBS can also improve the hardware idling problem and save up to 87.5% in bus bandwidth requirements.

5.2 Future Work

The future work is to extend the proposed low power and power adaptive ME designs to Scalable Video Coding (SVC) standard [68]. The SVC standard is developed based on H.264/AVC standard[55], but has 2 major features of temporal and spatial scalability for video streaming applications.

The temporal scalability is to allow scalability in temporal domain to change the frame rates, and this feature is constructed based on Motion Compensated Temporal Filtering (MCTF) and Hierarchical B-Pictures approaches. To extend our work to support temporal scalability, the proposed binary motion search architecture can benefit from this scalability feature due to the data for block matching are in binary format. Hence, both the complexity and bus bandwidth are able to be reduced. However, the video quality may be an issue due to the long distance search using binary format may lead to imprecise prediction and poor motion compensated performance.

The spatial scalability is to allow scalability in spatial domain to support different video resolutions in single bitstream. The challenge is to improve the prediction performance from low to high or high to low resolutions with another exhaustive search for power savings. To extend our work to support spatial scalability, the proposed binary motion search architecture can benefit from this scalability feature due to the data for block matching are in binary format. Hence, both the complexity and bus bandwidth are able to be reduced. However, similar problems in the video quality may be an issue due to the downsampling process cause the aliasing for binary image to find less imprecise prediction as compared to the conventional 8-bit search.

Therefore, our next step is to study the relationship for motion vectors from temporal

and spatial scalability, and propos the new algorithm and architecture to support both the features.



Bibliography

- [1] N. Chaddha, and T. H. Y. Meng, "A low-power video decoder with power, memory, bandwidth and quality scalability," in *IEEE workshop on VLSI signal processing*, pp. 451-460, Oct. 1995.
- [2] J. Jung, and A. Bourge, "Power-scalable video encoder for mobile devices based on collocated motion estimation," *Proceedings of SPIE*, 2004.
- [3] L. Lu, and V. Sheinin, "Rate and decoding power constrained video coding scheme for mobile multimedia players," in *Proc. IEEE ICIP*, pp. 2861-2864, Oct. 2004.
- [4] Y. Liang, Z. He, and I. Ahmad, "Analysis and design of power constrained video encoder," in *Proc. IEEE 6th CAS Symp. on Emerging Technologies: Mobile and Wireless Comm.*, pp. 57-60, May 2004.
- [5] C.-J. Lian, S.-Y. Chien, P.-C. Tseng, L.-G. Chen, "Power aware multimedia: concepts and design perspectives," *IEEE Circuits and Systems Magazine*, pp. 26-34, April-June 2007.
- [6] M. Mizuno, *et al.*, "A 1.5-W Single-Chip MPEG-2 MP@ML Video Encoder with Low Power Motion Estimation and Clocking," *IEEE J. Solid State Circuits*, vol. 32, pp. 1807-1816, Nov. 1997.
- [7] S. Kumaki, *et al.*, "A 99-mm² 0.7-W Single-Chip MPEG-2 422P@ML Video, Audio, and System Encoder With a 64-Mb Embedded DRAM for Portable 422P@HL Encoder System," *IEEE J. Solid State Circuits*, vol. 37, pp. 450-454, March 2002.
- [8] Y.-W. Huang, *et al.*, "A 1.3 TOPS H.264/AVC single chip encoder for HDTV applications," *Proc. ISSCC*, pp. 128-129, 2005.
- [9] Fujitsu. <http://www.fujitsu.com/downloads/PR/2007/20070521-01a.pdf>
- [10] Zoran Coach 10. http://www.zoran.com/IMG/pdf/COACH_10.pdf
- [11] Sanyo. <http://www.sanyo.co.jp/koho/hypertext4-eng/0708/0830-1e.html>
- [12] TI. <http://focus.ti.com/docs/toolsw/folders/print/tmdh264e.html>

- [13] <http://shdesigns.org/batts/battcyc.html>
- [14] TI DaVinci. <http://focus.ti.com/docs/solution/folders/print/267.html>
- [15] Power management of TI DaVinci. <http://focus.ti.com/docs/pr/pressrelease.jhtml?preId=sc06194>
- [16] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full search block matching VLSI architecture," *IEEE Trans. Circuits and Systems for Video Technique*, vol. 12, pp. 61-72, Jan. 2002.
- [17] E. Brockmeyer, *et al.*, "Low power memory storage and transfer organization for the MPEG-4 full pel motion estimation on a multimedia processor," *IEEE Trans. on Multimedia*, vol. 1, pp. 202-216, June 1999.
- [18] M. Takahashi, *et al.*, "A 60-MHz 240-mW MPEG-4 Videophone LSI with 16-Mb Embedded DRAM," *IEEE J. Solid State Circuits*, vol. 35, pp. 1713-1721, Nov. 2000.
- [19] M. Mizuno, *et al.*, "A 1.5-W Single-Chip MPEG-2 MP@ML Video Encoder with Low Power Motion Estimation and Clocking," *IEEE J. Solid State Circuits*, vol. 32, pp. 1807-1816, Nov. 1997.
- [20] F. Momers, *et al.*, "Image: A low cost, low power video processor for high quality motion estimation in MPEG-2 encoding," *IEEE J. Solid State Circuits*, vol. 44, pp. 774-783, August 1998.
- [21] W. Badawy, and M. A. Bayoumi, "A low power VLSI architecture for mesh-based video motion tracking," *IEEE Trans. Circuits and Systems II-Analog and Digital Signal Processing*, vol. 49, pp. 488-504, July 2002.
- [22] Y. Hamamoto, *et al.*, "A low-power single-chip MPEG2 (Half-D1) video codec LSI for portable consumer product applications," *IEEE Trans. Consumer Electronics*, vol. 45, pp. 496-500, August 1999.
- [23] V. L. Do, and K. Y. Yun, "A low-power VLSI architecture for full-search block-matching motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, pp. 393-398, August 1998.
- [24] Y. Yatabe, *et al.*, "An MPEG2/4 dual codec with sharing motion estimation," *IEEE Trans. Consumer Electronics*, vol. 51, pp. 660-664, May 2005.
- [25] S. Saponara, and L. Fanucci, "Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems," *IEE Proc. Comput. Digit. Tech.*, vol. 151, Jan. 2004.
- [26] T.-C. Chen, *et al.*, "Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC" *IEEE Trans. Circuits and Systems for Video Technology*, vol. 17, pp. 568-577, May 2007.

- [27] S. Kawahito, *et al.*, "Low-power motion vector estimation using iterative search block-matching methods and a high-speed non-destructive CMOS image sensor," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, pp. 1084-1092, Dec. 2002.
- [28] M. Jiang *et al.*, "Low-power systolic array processor architecture for FSBM video motion estimation," *IEE Electronics Letters*, vol. 42, Sep. 2006.
- [29] R. Gao, D. Xu, and J. P. Bentley, "Reconfigurable hardware implementation of an improved parallel architecture for MPEG-4 motion estimation in mobile applications," *IEEE Trans. Consumer Electronics*, vol. 49, pp. 1383-1390, May 2003.
- [30] W.-M. Chao, *et al.*, "A novel hybrid motion estimator supporting diamond search and fast full search," in *Proc. IEEE ISCAS*, May 2002.
- [31] "ISO/IEC 14496-5:2001 Final Committee Draft, "MPEG01/N4025.
- [32] "Video coding for low bit rate communication, "ITU-T Rec. H.263, 1998.
- [33] P. Kuhn, "Complexity analysis and VLSI architectures for MPEG-4 motion estimation," Boston, MA, Kluwer, 1999.
- [34] M. M. Mizuki, U. Y. Desai, I. Masaki, and A. Chandrakasan, "A binary block matching architecture with reduced power consumption and silicon area requirement," in *Proc. IEEE ICASSP*, pp. 3248-3251, May 1996.
- [35] S.-H. Wang, *et al.*, "Platform based design of all binary motion estimation with bus interleaved architecture," in *Proc. IEEE Int. Symposium on VLSI-DAT*, pp. 241-244, April 2005.
- [36] M. Miyama, *et al.*, "A sub-mW MPEG-4 motion estimation processor core for mobile video application," *IEEE J. Solid State Circuits*, vol. 39, pp. 1562-1570, Dec. 2004.
- [37] Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, "Global elimination algorithm and architecture design for fast block matching motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, pp. 898-907, June 2004.
- [38] H.-M. Jong, L.-G. Chen, T.-D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, pp. 407-416. Aug. 1994.
- [39] J. -F. Shen, T.-C. Wang and L.-G. Chen, "A novel low power full search block matching motion estimation design for H.263+," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, pp. 890-897, July 2001.

- [40] S. Y. Yap and J. V. McCanny, "A VLSI architecture for variable block size video motion estimation," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 51, no. 7, pp. 384-349, July 2004.
- [41] J.-H. Luo, C.-N. Wang, and T. Chiang, "A novel all-binary motion estimation (ABME) with optimized hardware architectures," *IEEE Trans. Circuits and Systems for Video Technology*, vol.12, no. 8, pp. 700-712, Aug. 2002.
- [42] C. D. Vleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and architectural co-design of a motion-estimation engine for low power video devices," *IEEE Trans. Circuits and Systems for Video Technology*, vol.12, no. 12, pp. 1093-1105, Dec. 2002.
- [43] B.-C. Song, and K.-W. Chun, "Multi-resolution block matching algorithm and its VLSI architecture for fast motion estimation in an MPEG-2 video encoder," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, pp 1119-1137, Sep. 2004.
- [44] W. Burlison, P. Jain, and S. Venkatraman, "Dynamically parameterized algorithms and architectures to exploit signal variations for improved performance and reduced power," *IEEE International Conf. on Acoustics, Speech, and Signal Processing*, pp. 901-904, 2001.
- [45] S. Mietens, P. H. N. de With, and C. Hentschel, "Computational-complexity adaptive motion estimation for mobile MPEG encoding," *IEEE Trans. Consumer Electronics*, vol. 50, pp. 281-291, Feb. 2004.
- [46] S. -Y. Huang, C.-Y. Cho, and J.-S. Wang, "Adaptive fast block matching algorithm by switching patterns for sequences with wide range motion content," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 11, pp. 1373-1384, Nov. 2005.
- [47] S. -S. Lin, P. C. Tseng, C. P. Lin, and L. G. Chen, "Multi-mode content-adaptive motion estimation algorithm for power-adaptive video coding systems," *Proc. IEEE Workshop on Signal Processing Systems*, pp. 239-244, 2004.
- [48] T. Li, S. Li and C. Shen, "A novel configurable motion estimation architecture for high efficiency MPEG-4/H.264 encoding," *Proc. IEEE ASP-DAC*, pp. 1364-1367, 2005.
- [49] Y. Wang, Y. Wang, and H. Kuroda, "A globally adaptive pixel decimation algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 6, pp 1006-1011, Sep. 2000.
- [50] V. G. Moshnyaga, "A new computational adaptive formulation of block matching motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 1, pp 118-124, Jan. 2001.

- [51] S.-H. Wang, C. N. Wang, and T. Chiang, "A complexity adaptive variable-bit-depth motion estimation," *Proc. IEEE International Conf. on Consumer Electronics*, pp. 233-234, 2005.
- [52] A. Takagi, K. Nishikawa, and H. Kiya, "Low-bit motion estimation with edge enhanced images for low power MPEG encoder," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 505-508, 2001.
- [53] A. M. Tourapis, O. C. Ou, M. L. Liou, C. W. Bay, and H. K. Kowloon, "Predictive motion vector field adaptive search technique (PMVFAST) - enhancing block matching motion estimation," *Proc. International Conf. on Visual Communication and Image Processing*, pp. 883-892, 2001.
- [54] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," *Proc. International Conf. on Visual Communication and Image Processing*, pp. 1069-1079, 2002.
- [55] "Draft of version 4 of ISO/IEC 14496-10," ISO/IEC JTC1/SC29/WG11, MPEG05/N7081, April 2005.
- [56] H.-W. Cheng and L.-R. Dung, "A content based methodology for power adaptive motion estimation architecture," *IEEE Trans. Circuits and Systems II-Express Briefs*, vol. 52, pp. 631-635, Oct. 2005.
- [57] Z.-L. He, C.-Y. Tsui, K.-K. Chan, and M.-L. Liou, "Low-power VLSI design for motion estimation using adaptive pixel truncation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, pp 669-678, Aug. 2000.
- [58] A. Takagi, S. Muramatsu, and H. Kiya, "Motion estimation with power adaptation and its VHDL model," *Proc. IEEE ICIP*, pp. 118-121, 2000.
- [59] M. Pedram and J.M. Rabaey, "Power aware design methodologies," Kluwer Academic Publishers, 2002.
- [60] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 4, pp 702-706, Aug. 1997.
- [61] P. H. W. Hong and O. C. Au, "Modified one-bit transform for motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, pp.1020-1024, Oct. 1999.
- [62] X. Lee and Y.-Q. Zhang, "A fast hierarchical motion-compensation scheme for video coding using block feature matching," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, pp. 627-635, Dec. 1996.

- [63] X. Song, T. Chiang, X. Lee, and Y.-Q. Zhang, "New fast binary pyramid motion estimation for MPEG2 and HDTV encoding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, pp. 1015-1028, Oct. 2000.
- [64] G. Magklis et al., "Dynamic frequency and voltage scaling for a multi-clock-domain microprocessor," *IEEE Micro*, vol. 23, pp. 62-68, Nov. 2003.
- [65] J. H. Lee, and N. S. Lee, "Variable block size motion estimation algorithm and its hardware architecture for H.264/AVC," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 741-744, 2004.
- [66] C. Y. Chen, *et al.*, "Analysis and architecture design of variable block size motion estimation for H.264/AVC," *IEEE Trans. Circuits and Systems I - Regular papers*, vol. 53, pp. 578-593, March 2006.
- [67] C.-M. Ou, C.-F. Lee, and W.-J. Hwang, "An efficient VLSI architecture for H.264 variable block size motion estimation," *IEEE Trans. Consumer Electronics*, vol. 51, pp. 1291-1299, Nov. 2005.
- [68] J.-R. Ohm, "Advances in scalable video coding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 42-56, Jan. 2006.
- [69] M. Bhardwaj, R. Min, and A. Chandrakasan, "Power-aware systems," *Proc. IEEE 34th asilomar conference on signals, systems and computers*, pp. 1695-1701, Oct. 2000.
- [70] L. Mazzoni, "Power-aware design for embedded systems," *IEE electronics systems and software*, pp. 12-17, Oct. 2003.
- [71] Z. He, *et al.*, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, no. 5, pp 645-658, May 2005.

Shih-Hao Wang

CONTACT INFORMATION

Engineering Building IV - 529R *Voice:* 03-5712121-54228
Institute of Electronics *Fax:* 03-5731791
National Chiao Tung University *E-mail:* shwang.ee90g@nctu.edu.tw
Hsinchu, Taiwan 30010 R.O.C. *WWW:* cwww.ee.nctu.edu.tw/~vdo90843

RESEARCH INTERESTS

Video Compression, Video Signal Processing, VLSI Design on Multimedia

EDUCATION

National Chiao Tung University, Hsinchu, Taiwan ROC

Ph.D., Institute of Electronics, September 2007

- Dissertation: "Algorithm and Architecture Design of Motion Estimation for Power Constrained Video Coding Systems"
- Advisor: Tihao Chiang

M.S., Electrical and Control Engineering, June 2001

- Thesis: "Wavelet Tree Based Watermarking for Copyright Protection"
- Advisor: Yuan-Pei Lin

National Tsing Hua University, Hsinchu, Taiwan ROC

B.S., Power Mechanical Engineering, June, 1999

- Ranked 2nd in class

Tainan First Senior High School, Tainan, Taiwan ROC

Ranked 1st in class & 5th in that grade

HONORS AND AWARDS

Nomination in Marquis Who's Who in Asia, 2007.

Outstanding Ph.D. student scholarship, 2004. (awarded by EE, NCTU)

Academic achievement award and scholarship, 1996.

ACADEMIC EXPERIENCE

National Chiao Tung University, Hsinchu, Taiwan ROC

Ph.D. candidate

Sep. 2002 - Sep. 2007

Algorithm and architecture design of motion estimation

- Low power bi-directional binary motion estimation architecture.
- Power aware motion estimation design using configurable iterative binary searches.

Video transcoding

- A FGS multi-layers to single layer transcoder.
- A unified MPEG-4 FGS to MPEG-1/2/4 single layer transcoder.

H.264 decoder system

- A FPGA prototyping solution (ARM based platform).
- A novel software-hardware co-design architecture (QCIF 10fps real-time decoding).

MPEG-4 codec system

- TI DSP based video codec solution.
- A codec system with on-line (not real-time) MPEG-4 encoder/decoder system from video capturing to TV display.

Digital watermarking

- A robust and blind digital watermarking technique based on wavelet characteristics for secure information embedding.
- Journal publication in IEEE Trans. on Image Processing.

Teaching Assistant

Sep. 1999 - Sep. 2003

Introduction to Digital Signal Processing (1999 Fall)

Digital Signal Processing (2000 Fall)

Digital Compression (2001 Spring)

Multimedia Communication (2003 Spring)

PROFESSIONAL
EXPERIENCE

Ambarella Taiwan Ltd., Hsinchu, Taiwan.

Member of Technical Staff

June, 2004 - Present

H.264 algorithm development & Firmware programming.

Part-time Engineer

Oct., 2003 - June, 2004

H.264 algorithm development.

COMPUTER SKILLS

- IC CAD Tools: Verilog, Design Compiler, SOC Encounter, Prime POver, etc.
- FPGA CAD Tools: Xilinx ISE, Quartus, Synplify, etc.
- Languages: C/C++, Matlab, etc.



著作目錄

● 期刊論文

1. S.-H. Wang, W.-H. Peng, Y.-W. 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 decoder with block level pipelining," *Journal of VLSI Signal Processing Systems*, vol. 41, no. 1, pp. 93-110, Jan. 2005.
2. S.-H. Wang, W.-L. Chen, and Tihao Chiang, "An efficient FGS to MPEG-1/2/4 single layer transcoder with R-D optimized multi-layer streaming technique for video quality improvement," *Journal of the Chinese Institute of Engineers*, vol. 30, no.6, pp. 1059-1070, 2007.
3. S.-C. Chang, W.-H. Peng, S.-H. Wang, and T. Chiang, "A Platform based Bus-interleaved Architecture for Deblocking Filter in H.264/MPEG-4 AVC," *IEEE Trans. Consumer Electronics*, vol. 51, no. 1, pp. 249-255, Feb. 2005.
4. S.-H. Wang, and Y.-P. Lin, "Wavelet tree quantization for copyright protection watermarking," *IEEE Trans. Image Processing*, vol. 13, no. 2, pp. 154-165, Feb. 2004.

● 審查中期刊論文

1. S.-H. Wang, and T. Chiang, "A power adaptive motion estimation IP core design using iterative binary search," *IEEE Trans. Circuit and System for Video Technology*, 2006.
2. S.-H. Wang, S. -H. Tai, and T. Chiang, "A low power and bandwidth efficient motion estimation IP core design using binary search," *IEEE Trans. Circuit and System for Video Technology*, 2006.

● 國際會議論文

1. S.-H. Wang, W.-H. Peng, Y. He, G.-Y. Lin, C.-Y. Lin, S.-C. Chang, C.-N. Wang, and T. Chiang, "A Platform Based MPEG-4 Advanced Video Coding Decoder with Block Level Pipelining," *Proc. IEEE ICICS-PCM*, Singapore, Nov. 2003.
2. S.-H. Wang, W.-L. Tao, W.-H. Peng, C.-N. Wang, and T. Chiang, "Platform based design of all binary motion estimation (ABME) with bus interleaved architecture," *Proc. IEEE International Symposium on VLSI Technology, System and Applications*, Hsinchu, April 2005.
3. S.-H. Wang, C.-N. Wang, and T. Chiang, "A complexity aware variable-bit-depth motion estimation," *Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, Jan. 2005.
4. S.-C. Chang, W.-H. Peng, S.-H. Wang, and T. Chiang, "A platform-based de-blocking filter design with bus interleaved architecture for H.264," *Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, Jan. 2005.

5. S.-H. Wang, and Y.-P. Lin, "Blind watermarking using wavelet tree quantization," *Proc. IEEE International Conference on Multimedia and Expo*, Lausanne, August, 2002.

- MPEG 視訊標準會議文件

1. S.-H. Wang, C.-N. Wang, Yi-Shin Tung, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M9951: AHG report on editorial convergence of MPEG-4 reference software," Oct. 2003.
2. S.-H. Wang, C.-N. Wang, Y.-S. Tung, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M9632: AHG report on editorial convergence of MPEG-4 reference software," July 2003.
3. S.-H. Wang, C.-N. Wang, Y.-S. Tung, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M9355: AHG report on editorial convergence of MPEG-4 reference software," March 2003.
4. S.-H. Wang, C.-N. Wang, G.-Y. Lin, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M9073: AHG report on editorial convergence of MPEG-4 reference software," Dec. 2002.
5. S.-H. Wang, C.-N. Wang, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M8886: Proposed text of proposed draft technical reports of ISO/IEC PDTR 14496-7 for optimized simple profile reference software," Oct. 2002.
6. S.-H. Wang, C.-N. Wang, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M8884: AHG report on editorial convergence of MPEG-4 reference software," Oct. 2002.
7. S.-H. Wang, C.-N. Wang, Tihao Chiang, and H.F. Sun, "ISO/IEC JTC1/SC 29/WG 11 M8603: AHG report on editorial convergence of MPEG-4 reference software," July 2002.
8. S.-H. Wang, Y.-C. Lin, C.-N. Wang, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M8408: AHG report on editorial convergence of MPEG-4 reference software," May 2002.
9. S.-H. Wang, C.-N. Wang, T. Chiang, and H. Sun, "ISO/IEC JTC1/SC 29/WG 11 M8041: AHG report on editorial convergence of MPEG-4 reference software," March 2002.

- 專利

1. M.-Y. Huang, T.-L. Su, S.-H. Wang, C.-N. Wang and T. Chiang, "MPEG-4 streaming system with adaptive error concealment," 美國專利，專利號 20060104366.

- 審查中專利

1. S.-H. Wang, L. Kohn, and T. Chiang, "Mode decision using approximate 1/2 pel interpolation," 美國專利. (Filed on Nov. 23, 2005).

2. S.-C. Chang, W.-H. Peng, S.-H. Wang, and T. Chiang, “A Platform Based Bus-interleaved Architecture for Deblocking Filter in H.264/MPEG-4 AVC,” 美國專利. (Filed on March 24, 2005)

