Low Power Architectures for MPEG-4 AVC/H.264 Video Compression

Asral Bahari

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
July 24, 2008

Abstract

Multimedia communication will be an important application in future wireless communication. The second-generation mobile communication systems already support basic multimedia services such as voice, text-messaging services and still-imaging communication. However, next generation wireless communication technology combined with advances in integrated circuit design and process fabrication technology will allow more data to be processed and transmitted through wireless channels. This will lift the current barriers and enable more demanding multimedia applications such as video telephony, video conferencing and video streaming.

Video compression plays an important role in today's wireless communications. It allows raw video data to be compressed before it is sent through a wireless channel. However, video compression is compute-intensive and dissipates a significant amount of power. This is a major limitation in today's portable devices. Existing multimedia devices can only play video applications for a short time before the battery is depleted. This limits the user's entertainment experience and becomes a major bottleneck for the development of more attractive applications.

The focus of this thesis is to design a low power video compression system for wireless communication. In this thesis, we propose techniques to minimise the power consumption at the algorithmic and architectural level. The low power is achieved by minimising the switching power between interacting modules that contribute to major the power consumption in H.264 standard.

Motion estimation (ME) has been identified as the main bottleneck in MPEG video compression, including in the H.264 system where it takes up to 90% of the coding time. To reduce the power consumption in motion estimation hardware architecture, we have proposed a two-step algorithm that minimises the memory bandwidth and computational load of the ME. In this technique, the search is performed in low resolution mode at the first stage followed by high resolution mode in the second stage. This method reduces the total computation and memory access compared to the conventional method without significantly degrading the picture quality. The simulation results show that the proposed method gives good PSNR as compared to the conventional full search with PSNR drop $< 0.5dB$.

An energy efficient hardware for implementing the proposed two-step method is suggested. The architecture is able to perform both low resolution and high resolution searches without significantly increasing the area overhead. With a unique pixel arrangement, the proposed method is able to perform at both low resolution and high resolution while still being able to reduce the memory bandwidth. The results show that the proposed architecture is able to save up to 53% energy as compared to the conventional full search architecture.

In addition, video compression is a data intensive application where the data need to be read and stored between the main memory and the computational unit. In this thesis, we have proposed a method to reduce the bus switching activity on off-chip busses due to multiple search area requirements during motion prediction in H.264. An interframe bus encoding that exploits the high correlation between frames achieved significant power saving as compared to the conventional method. The proposed technique is able to reduce the off chip bus switching activity by 53%, which is equivalent to a 38% reduction of power consumption on a typical off-chip wire capacitance.

Finally, the proposed low power ME architecture and bus encoding technique were implemented on the H.264 system. The results show that the two-step algorithm is able to save 40% energy over the conventional system, while the interframe bus encoding technique saves 58% energy when transferring search area pixels on the 32-bit bus compared to the unencoded bus.

Publication

Refereed Papers

1. A. Bahari, T. Arslan, and A. T. Erdogan, "*Interframe bus encoding technique for low power video compression,*" 20th International Conference on VLSI Design, pp. 691 - 698, Jan 2007.

2. A. Bahari, T. Arslan, and A. T. Erdogan, "*Low power variable block size motion estimation using pixel truncation,*" IEEE International Symposium on Circuits and Systems, pp. 3663 – 3666, May 2007.

3. A. Bahari, T. Arslan, and A. T. Erdogan, "*Reduced computation and memory access for VBSME using pixel truncation,*" 20th IEEE International SOC Conference, Sep 2007.

4. A. Bahari, T. Arslan, and A. T. Erdogan, "*Low computation and memory access for variable block size motion estimation using pixel truncation,*" IEEE Workshop on Signal Processing Systems, pp.681 - 685, Oct 2007.

5. A. Bahari, T. Arslan, and A. T. Erdogan, "*Low power hardware architecture for VBSME using pixel truncation,*" 21st International Conference on VLSI Design, pp.389 - 395, Jan 2008.

Submitted Papers

1. A. Bahari, T. Arslan, and A. T. Erdogan, "*Interframe bus encoding technique for low power H.264 video compression,*" Submitted to 21st IEEE International SOC Conference.

2. A. Bahari, T. Arslan, and A. T. Erdogan, "*Low power H.264 video compression architectures for mobile communication,*" Submitted to IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

Acknowledgements

I would like to thank my supervisor, Professor Tughrul Arslan, for his excellent guidance and expertise throughout my research work.

I would also like to thank my second supervisor, Dr. Ahmet T. Erdogan, for his help and advice during different stages of my PhD.

Many thanks to those in the School of Electronics and Engineering, University of Edinburgh, especially in the System Level Integration Group (SLIG), for those memorable years.

My thanks go to my sponsor, the Government of Malaysia, for funding my studies.

Many thanks are also due to my parents for their guidance and support.

Special thanks to my wife for her love, care and encouragement during my studies. Thank you to my sons for making my life so wonderful.

Contents

List of figures

List of tables

Acronyms and abbreviations

1BT	One-Bit Transform
1D	One-Dimension
2BT	Two-Bit Transform
2D	Two-Dimension
3SS	Three Step Search
APBI	Adaptive Partial Bus Invert
ASIC	Application Specific Integrated Circuit
AVC	Advance Video Coding
BF	Base Frame
BI	Bus Invert Technique
BXOR	Binary XOR
CAVLC	Context Adaptive Variable Length Codes
CCD	Charge-coupled Device
CDF	Cumulative Density Function
CIF	Common Intermediate Format
CMB	Current Macroblock
CMOS	Complementary Metal Oxide Semiconductor
CPL	Complementary Pass Logic Circuits
CPU	Central Processing Unit
DC	Direct Current
DCT	Discrete Cosine Transform
DA	Distributed Arithmetic
DF	Dependence Frame
DFIR	Deblocking Filter
dbm	Difference Based Map
DMA	Direct Memory Access
DPC	Difference Pixel Count
DPM	Dynamic Power Management
DS	Diamond Search
DSP	Digial Signal Processor
DVD	Digital Video Disc
DVS	Dynamic Voltage Scaling
EC	Entropy Coder
FIFO	First In First Out
GIP	Giga Instruction Persecond
H.264	Recommendation of ITU Telecommunication Standardization Sector
HADA	Hadamard transform

HDL	Hardware Description Language
HDTV	High Definition TV
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IQ	Inverse Quantizer
LRQME	Low Resolution Quantize Motion Estimation
LSB	Low Significant Bits
MAD	Mean Absolute Different
MB	Macroblock
MC	Motion Compensation
ME	Motion Estimation
MF	Multiplication Factor
MIPS	Million Instruction Persecond
MMS	Multimedia Messaging Service
MPEG	Motion Picture Expert Group
MRMAD	Mean Reduce MAD
MSB	Most Significant Bits
MV	Motion Vector
NMOS	P-channel Metal Oxide Semiconductor Field-Effect Transistor
NTB	Number of Truncated Bits
NTSS	New Three Step Search
PBI	Partial Bus Invert
PDA	Personal digital assistant
PE	Processing Element
PMOS	P-channel Metal Oxide Semiconductor Field-Effect Transistor
PMVFAST	Prediction Motion Vector Field Adaptive Search Technique
PNR	Place and Route
PSNR	Peak Signal to Noise Ratio
Q	Quantiser
QCIF	Quarter Common Intermediate Format
QP	Quantiser Parameter
RAM	Random Access Memory
RC	Rate Control
RD	Rate-Distortion
RGB	Red/Green/Blue Colour Format
RISC	Reduced Instruction Set Computing
RTL	Register Transfer Level
SA	Search Area
SAD	Sum of Absolute Difference
SDF	Standard Delay Format
SDRAM	Synchronous Dynamic Random Access Memory
SF	Scaling Factor
SOC	System on Chip

SOI	Silicon on Insulator
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
UMC	United Microelectronics Corporation
vbm	Value Based Map
VBSME	Variable Block Size Motion Estimation
VCEG	Video Coding Expert Group
VLC	Variable Length Codes
VLSI	Very Large Scale Integrated Circuit
WMV	Windows Media Video
YUV	Luminance/Blue chrominance/Red chrominance Colour Format

Nomenclature

α	switching activity
$\bar{p}$	Current MB mean
π	Pi, the ratio of the circumference of a circle to its diameter (Value = 3.141592654)
B	Blue color signal
$C(k,l)$	Current macroblock pixel
C_i	Load Capacitance
C_L	Capacitance Load
dbm	Difference Based Map
f	Clock frequency
G	Green color signal
Gnd	Electrical Ground
H	Search area height
i	Horizontal coordinate in the image data sample
j	Vertical coordinate in the image data sample
J	Frame rate
M	Macroblock height
MSE	Mean squared error
mvx	Horizontal motion vector of a macroblock
mvx_A	Horizontal motion vector for top-left 8x8 partition of a macroblock
mvx_B	Horizontal motion vector for top-right 8x8 partition of a macroblock
mvx_C	Horizontal motion vector for bottom-left 8x8 partition of a macroblock
mvx_D	Horizontal motion vector for bottom-right 8x8 partition of a macroblock
mvx_{min}	Minimum horizontal motion vector for 8x8 partitions of a macroblock
mvx_{max}	Maximum horizontal motion vector for 8x8 partitions of a macroblock
mvy	Vertical motion vector of a macroblock
mvy_A	Vertical motion vector for top-left 8x8 partition of a macroblock
mvy_B	Vertical motion vector for top-right 8x8 partition of a macroblock
mvy_C	Vertical motion vector for bottom-left 8x8 partition of a macroblock
mvy_D	Vertical motion vector for bottom-right 8x8 partition of a macroblock
mvy_{min}	Minimum vertical motion vector for 8x8 partitions of a macroblock
mvy_{max}	Maximum vertical motion vector for 8x8 partitions of a macroblock
N	Macroblock width
p	Search range
$p(i,j)$	Pixel vlaue of the current macroblock
p_1	First search range
p_2	Second search range

P_{CL}	Power consume by the capacitance load
P_{Dec}	Bus decoder circuit power consumption
P_{Enc}	Bus encoder circuit power consumption
$P_{leakage}$	Leakage power
$P_{short-ciruit}$	Short-circuit power
$P_{switching}$	Switching power
P_T	Total power
$p_i(x_i)$	Transition probability at node x
P_{total}	Total power consumption
$PMAX$	The maximum possible pixel value of the image data
R	Red color signal
$R(i+k, j+l)$	Candidate macroblock pixel
$SAD(i, j)$	Sum of absolute difference
$t1$	First quantisation threshold for LRQME
$t2$	Second quantisation threshold for LRQME
$t3$	Third quantisation threshold for LRQME
U	Blue chrominance signal
V	Red chrominance signal
V_{dd}	Power supply
vbm	Value Based Map
W	Search area width
x	Horizontal Coordinate in the transform domain
X_{ij}	Image sample
y	Vertical Coordinate in the transform domain
Y	Luminance signal
Y_{xy}	Coefficient value

Chapter 1
Introduction

1.1 Motivation

Multimedia will be an important application in future wireless communication [1]. Second-generation mobile communication systems already support basic multimedia services such as voice, text-messaging services and still-imaging communication [2]. However, next generation wireless communication technology combined with advances in integrated circuit design and process fabrication technology will allow more data to be processed and transmitted through wireless channels. This will lift the current barriers and enable more demanding multimedia applications such as video telephony, video conferencing and video streaming [3].

Video compression plays an important role in today's wireless communications. It allows raw video data to be compressed before it is sent through a wireless channel. However, video compression is compute-intensive and dissipates a significant amount of power. This is a major limitation in today's portable devices. Existing multimedia devices can only play video applications for a short time before the battery is depleted. This limits the user's entertainment experience and becomes a major bottleneck for the development of more attractive applications.

The latest video compression standard, MPEG-4 AVC/H.264 [4], gives a 50% improvement in compression efficiency compared to the previous standard. However, the coding gain

comes at the expense of increased computational complexity at the encoder. The introduction of variable block size partitions and multiple reference frames in the standard resulted in large increases in computational power and memory bandwidth during motion prediction. This thesis proposes several new implementations that reduce the power consumption in video compression systems such as in H.264.

1.2 Thesis Contribution

This thesis investigates low power techniques for the MPEG-4 system at an algorithmic and architectural level. The low power is achieved by minimising the switching power between interacting modules which contribute to the major part of the power consumption.

Motion estimation (ME) has been identified as the main bottleneck in MPEG video compression, including in the H.264 system where it takes up to 90% of the coding time. To reduce the power consumption in the motion estimation hardware architecture, a two-step algorithm that minimises the memory bandwidth and computational load of the ME is proposed. In this technique, the search is performed in low resolution mode at the first stage followed by high resolution mode in the second stage. This method reduces the total computation and memory access, as compared to the conventional method, without significantly degrading the picture quality.

This work has led to the development of energy efficient hardware to perform the two-step algorithm. The proposed hardware requires minimal additional area on top of the conventional architecture, where it can perform both the conventional full search and the proposed

two-step method. With unique data arrangement, the proposed method is able to perform both low resolution and high resolution search while still being able to reduce the memory bandwidth. This allows the system to select the required ME computational load depending on the video data characteristics.

In addition, video compression is a data intensive application where the data need to be read and stored between the main memory and the computational unit. In this thesis, a method to reduce the bus switching activity on off-chip buses due to multiple search area requirements during motion prediction in H.264 is proposed. The interframe bus encoding exploits the video data characteristics to achieve significant power saving compared to the conventional method.

Finally, the proposed two-step method and the bus encoding technique are implemented on the MPEG-4 system. Due to the way in which data is stored and accessed to/from the external frame memory, the implementation of interframe bus encoding for H.264 system is also proposed. The power saving from these techniques is evaluated and compared with the conventional architecture. The results show that the proposed methods achieve significant energy savings compared to the conventional architecture.

1.3 Thesis Structure

The rest of the thesis is organised as follows:

Chapter 2 discusses the sources of power consumption in CMOS VLSI design. Since the switching power is the main contributor in the total power consumption, the existing tech-

niques to minimise the power at different levels of design abstraction are discussed in this chapter.

Chapter 3 describes the existing video coding algorithms. While various video coding standards have been released, the main functional blocks are still common in most standards. These essential blocks are highlighted in this chapter. The existing video coding standards are reviewed, including the new features of H.264.

Chapter 4 discusses the existing low power strategy for video compression. In this chapter, low power techniques for essential modules are reviewed. Low power techniques at the system level and memory communication are also considered, since these modules consume a significant amount of power.

Chapter 5 presents the proposed technique to reduce the power consumption in motion estimation and the memory unit. As a result, the two-step algorithm is proposed in this chapter. The justification of this algorithm for variable block size motion estimation is first analysed. Then, the proposed algorithm is described in detail. The simulation results which show the effectiveness of this technique are discussed in this chapter.

Chapter 6 presents the proposed energy efficient hardware that utilises the proposed two-step algorithm. The existing conventional hardware for motion estimation is discussed. The performance of the proposed hardware against the conventional architecture is also presented.

Chapter 7 presents our proposed technique to reduce bus switching activity for off-chip busses in the H.264 system. Existing off-chip bus encoding techniques are first introduced. A comparison of the switching activity reduction is shown against existing methods.

Chapter 8 discusses the integration of the proposed methods into the H.264 system. The power saving is compared against the existing conventional architecture. The effectiveness of these methods in minimising the overall H.264 power consumption is discussed.

Chapter 9 summarises and concludes this thesis. In addition, the contribution of this thesis is re-highlighted and further research based on the techniques developed in this thesis is suggested.

1.4 Summary

Multimedia will be important in future wireless communication. With the widespread use of this application, video compression is indispensable to allow the video data to be transmitted or stored in electronic devices. Since video compression requires high computational power, a low power solution is critical, especially for battery operated devices. This thesis proposes low power methods for minimising power in the H.264/MPEG-4 standard. This thesis focuses on the main power contributors in the H.264 system at the algorithmic and architectural levels. Particular attention is given to the interaction between modules such as motion prediction and frame memories, since they consume significant power.

Chapter 2
Low Power CMOS VLSI Design

2.1 Introduction

Low power design is becoming important in today's consumer electronics. With the increased popularity of portable devices, more electronic appliances are depending on energy supply from the battery. Furthermore, as more functions are embedded in the chip and more complex algorithms are introduced to improve the services, the chip power consumption increases dramatically. To overcome this problem, low power design is indispensable in today's electronics design. This makes today's design task more challenging since additional parameters have to be considered from the early stage of design.

This chapter discusses the source of power consumption in today's VLSI design. The evolution of technology toward CMOS technology is first reviewed to see how design trends affect power consumption in electronics devices. The main power source in today's CMOS circuit is reviewed, since this has become the main bottleneck in designing low power systems. Switching activity has been identified as the main contributor for power consumption. Existing techniques for minimising this power use are discussed in this chapter. In order to facilitate low power design, power estimation techniques are important in identifying the main power consumption in the design to be implemented. Thus, existing power estimation techniques are also discussed. Finally, the standard design flow that is used in this thesis is described.

This chapter is organised as follows. Section 2.2 reviews the background of low power electronics design. Section 2.3 discusses the main power consumption in CMOS technology. Existing low power techniques to minimise switching power consumption are discussed in Section 2.4. Power estimation techniques are discussed in Section 2.5. Section 2.6 discusses the design flow used in this thesis. Finally, section 2.7 concludes this chapter.

2.2 Low Power Electronics Design

The electronics device design has evolved since the mid-20th century. Starting from the electromechanically based machines, the technology shifted to vacuum tube, before becoming dominated by bipolar transistors. The invention of the integrated circuit and complementary metal-oxide semiconductor (CMOS) technology have greatly revolutionised the way today's devices are being designed in terms of area and power consumption.

Today, CMOS is a dominant technology widely used by the industry. Compared to bipolar technology, CMOS offers better device integration and lower power consumption. At the early introduction of CMOS technology, its speed was much slower compared to bipolar. As the technology improved, the speed has now become almost comparable to that of bipolar. Furthermore, CMOS offers easier circuit design that allows automation in most design stages. This helps designers to easily integrate more transistors on a single chip.

Since its introduction, CMOS technology has seen a dramatic improvement in terms of speed, area and power. Due to reduced feature size, effective chip area has increased. Thus, area is becoming less important than before and is traded-off to reduce power [5]. Figure 2.1

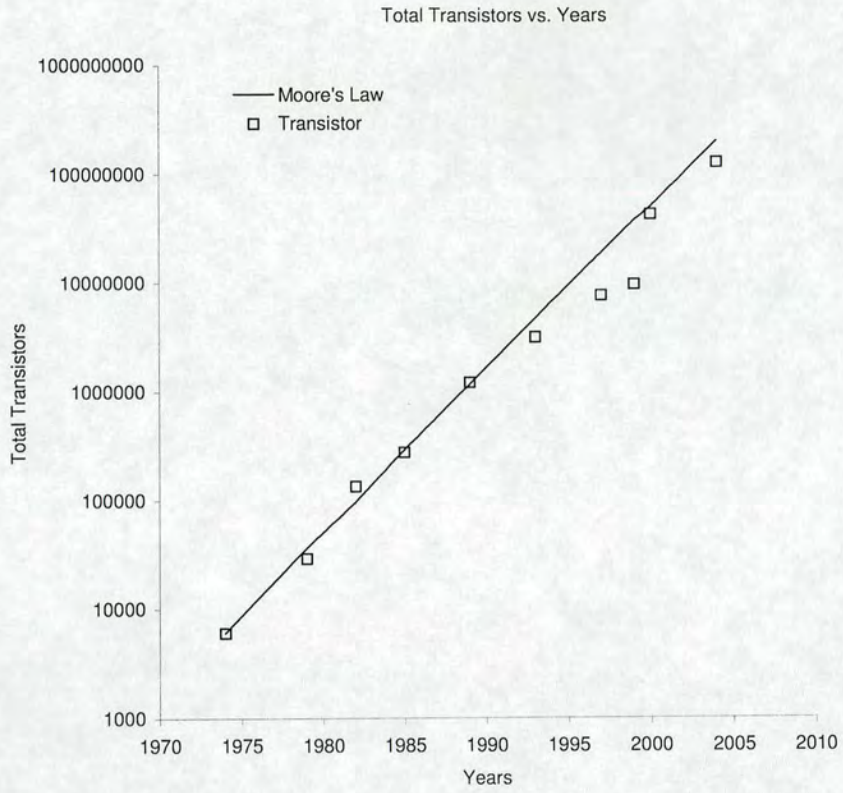


Figure 2.1: *Total transistors vs. time for Intel processors. [6]*

shows various processors developed by Intel Corporation [6]. As seen in the graph, the number of transistors are doubled every two years as predicted by Moore's law. Processors' operation becomes more complex and more functions are added onto the chip. This results in an increased number of transistors per chip and more power dissipated on a single chip. The power consumption is becoming more serious as the design is moving toward system-on-chip (SOC).

While CMOS has advantages in improved speed, area, and power consumption driven by aggressive technology scaling, this technology will reach its limit just as did its predecessors. Many experts agree that CMOS will reach the end of its progression in about 15 years - hitting physical, technological and economic limits [7]. Thus, many studies have been done to find

other alternatives to replace CMOS. However, up to now, no feasible solution has been found that can replace the CMOS technology completely [8]. Until a better solution can be found, the power dissipation in CMOS has to be dealt with in the early design stages.

In summary, there are several reasons why low power is important in today's VLSI design. One of the main factors is the increasing demand for portable devices since the early 1990s. Since these devices are operated using battery energy, longer battery lifetime is crucial. However, while the increase of computational power is doubled every two years, the improvement in battery capacity is slow, such as 6-7% per annum for Lithium Ion (Li-ion) [9]. If low power design is not adopted, the battery energy life time will be very short, or larger batteries will be required to cope with the computational demand.

Furthermore, as the fabrication technology improves, more devices can be integrated into a single chip. This results in increases in chip temperature. Thus, more expensive packing and cooling systems are required to cope with the problem. Another serious problem due to increase of temperature is device reliability. The increase of 10^0C results in double the device failure rate [10] and performance decreases by approximately 3% [11]. The high temperature causes problems such as electromigration, junction fatigue and dielectric breakdown.

In the following section, the main sources of power consumption in CMOS technology will be discussed. Then, the general techniques used to minimise the power consumption in CMOS circuits will be reviewed.

2.3 Sources of CMOS Power Consumption

There are three main sources of power dissipation in CMOS circuits. This can be summarised as follows [5]:

$$P_{total} = P_{leakage} + P_{short-circuit} + P_{switching} \tag{2.1}$$

where P_{total} is the total power dissipation of a CMOS circuit, $P_{leakage}$ is the leakage power, $P_{short-circuit}$ is the short circuit power and $P_{switching}$ is the switching power.

These uses of power can be categorized as dynamic or static. Dynamic power is used when the circuit is active, and largely depends on the behaviour of the switched signal, whereas static power is consumed even if the circuit is not active, i.e. in standby mode. In Equation 2.1, $P_{leakage}$ is considered as static power, while $P_{short-circuit}$ and $P_{switching}$ contribute toward dynamic power. Each of these components will be discussed in the following sections.

Leakage power occurs when there is unwanted current flow in the circuit due to the physical properties of the semiconductor devices. While the value is very small, it could be a problem for battery operated devices. Any battery operated device will run out of energy, even if it is idle most of the time.

Short circuit power is used when there is a direct path from V_{dd} to Gnd. A CMOS circuit operates based on a pull-up and pull-down concept, depending on the input voltage. The pull-up and pull-down circuit is built using PMOS and NMOS networks, respectively. For a CMOS circuit to be an ideal switch, only one network is turned on at any given time.

However, in practical use, the input voltage requires a finite amount of time to change its state from $0 \rightarrow 1$ or $1 \rightarrow 0$. During switching of the input, there is a time when both $NMOS$ and $PMOS$ networks are turned on simultaneously. This creates a path for current to flow directly between Vdd and Gnd. To reduce short-circuit power, it is important to keep the signal rise and fall times during the transition as small as possible, or at least keep both rise and fall times equal [12]. This can be done by properly sizing the transistors or by adjusting the capacitance load.

2.3.1 Switching Power

Switching current causes the highest power dissipation in CMOS circuits. In some designs, it consumes about 90% of the total power [13]. Switching power occurs when the capacitance load must be charged or discharged in response to the input changing.

Fig 2.2 shows typical CMOS inverter operation with the input constantly toggled. When the input is low, the PMOS transistor is turned on and the NMOS transistor is turned off. This causes current to flow from the V_{dd}, charging up the load C_L. When the input is switched to high, the PMOS is turned off and the NMOS is turned on. Since there is a potential difference between C_L and Gnd, current flows from C_L to Gnd. On average, the switching power consumed by the inverter is given by [5]:

$$P_{switching} = C_L V_{dd}^2 f \tag{2.2}$$

where C_L is the load capacitance, V_{dd} is the power supply and f is the rate of the toggled

input.

In synchronous digital circuits, the rate of input switching depends on the clock frequency. However, it may not toggle at the same rate as the clock. Thus, to take this into account, the input switching probability, α, is used to modify Equation 2.2:

$$P_{switching} = C_L V_{dd}^2 f \alpha \tag{2.3}$$

From Equation 2.3, since power is only consumed when the capacitive load is being charged or discharged, the term $C_L\alpha$ is known as effective capacitance. Capacitive load that is not charged or discharged will not consume any power. It is obvious that the switching power is proportional to V_{dd}^2, f and $C_L\alpha$. Thus the switching power can be minimised by reducing these parameters. The following section discusses how these parameters can be utilised to reduce total power and how this affects design.

2.4 General Low Power Design Techniques for VLSI Design

As discussed in the previous section, switching power consumes the majority of the power consumption in digital CMOS circuits. For CMOS technology with 45nm and below, leakage power will dominate the power consumption [14]. Various techniques at circuit and technology levels such as transistor stacking, multi threshold voltage CMOS, and transistor body biasing can be applied to reduce this power [15]. This thesis focuses on minimising

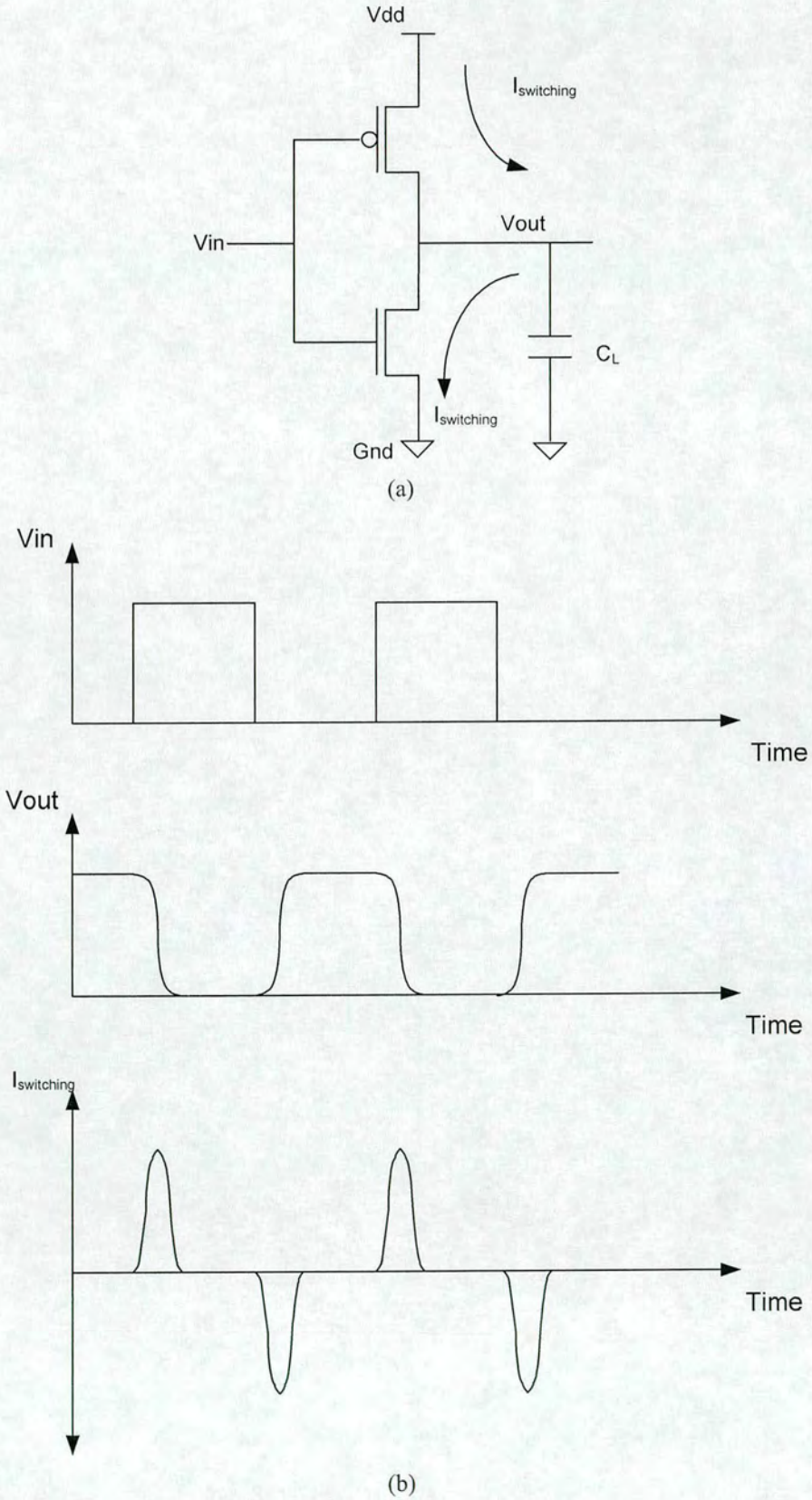


Figure 2.2: *Switching current of a CMOS inverter (a) circuit schematic (b) input and output voltage/current*

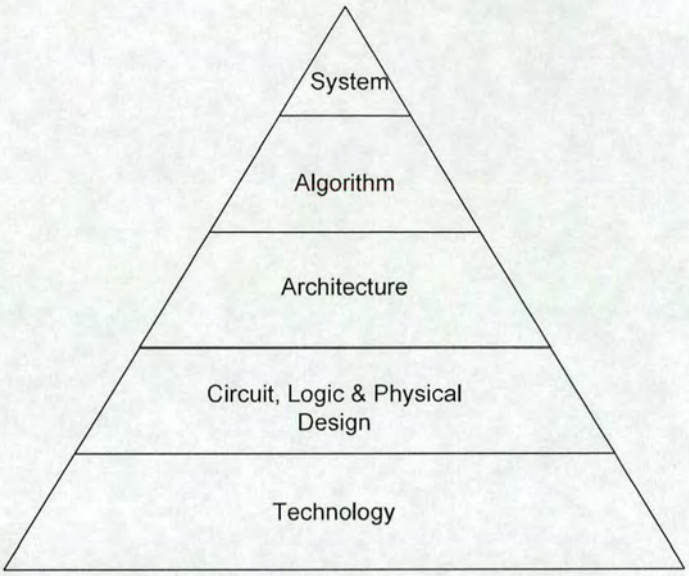


Figure 2.3: *Design abstraction level and the potential low power techniques [13]*

switching power that is applicable at algorithmic and architectural levels.

Many research studies have focused on reducing switching power, mainly by reducing the operating voltage, physical capacitance, and switching activity. In most cases, reduction of these parameters results in degradation of other performance criteria such as speed and area. For example, reducing the supply voltage results in quadratic power reduction, but increases the propagation delay in the circuit which results in reduced speed. Thus, a careful balance between power reduction and overall system performance is crucial.

Power consumption can be tackled at different levels of design abstraction as shown in Fig 2.3. The earlier the problem is addressed at the design stage, the greater the impact on power savings for the whole system. In this section, the general methods to reduce power consumption will be reviewed, starting from the system level and proceeding all the way to the technology level.

At the system level, the objective of power minimisation is to reduce the workload between

interacting modules. Since different modules require different workloads, these modules can be controlled to manage their operations. This is the basic reason for implementing dynamic power management (DPM), commonly used in communication systems and interactive devices. To minimise power consumption, the system operates in different states, such as active, idle and sleep mode. The transition between these states is controlled by a power manager such as timeout policy [16] and predictive policy [17].

While DPM selectively forces components into a low-power state, dynamic voltage scaling (DVS) changes processor speed and voltage at run time depending on needs. The voltage is reduced only when this will not result in noticeable degradation of performance. In DVS, the voltage and speed are controlled by a set of algorithms such as based on the processor utilisation within fixed intervals [18][19] or by analysing application requirements [20].

At the algorithmic level, low power design is achieved by transforming the computation method so that an optimum operation can be performed. This optimisation includes computation modules and sequence of operations. One of the transformations is operation reduction, where the operation is recomposed to reduce the number of operations needed which leads in reduction of switched capacitance [21]. Since certain operations require less energy than others, lower power operations can be substituted to minimise power consumption [21].

Another means to reduce power consumption is at the architectural level. This approach minimises the power consumption resulting from the structure of the architecture. Compared to the previous two hierarchies, where the power saving is achieved at a larger scale, this approach reduces power consumption at the specific module that is being implemented. Several techniques that are available at this level are parallelism [22], pipelining [23], low power data

representation [5] and clock gating [24, 25].

At the logic and circuit level, designers have greater control over final circuit performance. One of the important decisions is the type of logic implementation. Since the amount of capacitance depends on the total number of transistors, the decision on logic implementation greatly affects the total power consumption. In CMOS circuits, the inputs are connected to the transistor gates, which results in high input capacitance due to gate, drain and source capacitance. In contrast, complementary pass logic circuits (CPL) [26] have lower input capacitance since the input is connected through the drain or source and fewer transistors are used to implement important logic. The result in [27] shows that CPL can give 30% power reduction. Other methods such as logic optimisation [5], transistor sizing [28] and power aware placement and routing can be used to minimize power at the circuit level.

At the technology level, technology scaling has been shown to be very effective in reducing overall power during the last four decades. This technique scales the voltage and feature size according to the constant-field scaling law [29]. This approach results in improvements in power consumption, chip density and circuit performance. The technology reduces in scale by a factor of two every three years. However, this method is limited by how much lower threshold voltage can be scaled without causing serious leakage current [30]. Apart from technology scaling, other technology improvement, such as Silicon on Insulator (SOI), reduces the parasitic capacitance in the circuit by 30% compared to CMOS technology [31].

2.5 Power Estimation Techniques

Power estimation is important since it helps designers identify the main power bottlenecks so that the appropriate corrective measures can be taken in the early design stages. Power estimation can be performed at several stages of the design: algorithm, architecture, gate level and circuit level. The power estimation accuracy at higher design levels is less accurate than when the design abstraction is closer to the silicon implementation. Most power estimations trade off accuracy for improvements in run time and capacity.

At the behavioural level, the power estimation depends on the execution of the algorithm since it lacks hardware structure. In information-theoretic models, measures such as entropy are used to estimate power consumption [32]. Higher switching activity results from higher entropy. Complexity based model [33] estimates the power based on circuit complexity parameters, such as amount and type of arithmetic, number of states and Boolean expressions.

At the architectural level or register transistor level (RTL), the design has been divided into sub blocks and each functional unit has been identified. Thus estimating power consumption at this level is crucial since it can be used to gain insight into which modules consume the most power. The power estimation tool utilises lines of code to understand the design structure and estimate the power. Several power estimation methods are available at this level. In the gate count approach, the consumed power is estimated based on the equivalent gate count [34]. Another technique is the power factor approximation method [35] which is based on previously characterised functional blocks. Estimating tools such as WattWatcher claim to be within 20% to 25% of actual silicon power requirements at a good execution speed [36].

At the gate level, the design are represented by technology dependent logic gates. The probability method calculates power using the input probability where this will be propagated into other nodes. The average power is calculated using:

$$P_{switching} = \frac{1}{2} f V_{dd}^2 \sum_{i=1}^{n} C_i p_t(x_i)$$

where C_i represents the total capacitance load and $p_t(x_i)$ represents the transition probability at node x. This approach has the advantage of being test pattern independent, and is fast since it only requires a single analysis to be run. While this method is suitable for combinational circuits, it is difficult to apply it to sequential circuits [37]. Furthermore, this method depends on a simplified delay model. Better accuracy can be achieved, at the expense of computing time, by considering the correlation among the internal nodes.

In the event driven simulation method, the average power is calculated based on the number of transitions that occur and the amount of power consumed per transition [38]. The accuracy of this technique depends strongly on how accurately the technology library has been characterised for power consumption. Thus, the library must be precharacterised for static and dynamic power at various input slopes and output loads. Typically, the calculation is done using a circuit simulator such as SPICE. This provides a convenient method and can be used on any design and technology that has a precharacterised library for power. While this approach can give better accuracy compared to the probability method, the result depends on the given input test pattern. Thus, the input must be chosen to match the closest real world implementation. Furthermore, this method requires longer execution times especially for large circuits.

For highest power estimation accuracy, a circuit level simulator, such as SPICE, provides the best approach. In this method, the actual physical device model, such as transistor, capacitor or resistor, is used during the simulation. Since the actual physical model is used, this method can accurately calculate non-linear behaviour that is often missed by the higher level power approximation techniques. It also calculates the actual leakage, short circuit and dynamic power. But while this method provides the best accuracy and is easy to use for smaller circuits, it is very time consuming for larger circuits. Furthermore, the power dissipation result depends on input test pattern.

2.6 Design Flow

The design flow shown in Figure 2.4 is adopted throughout this thesis. In this work, the design starts from the algorithm specification and continues up to the place and route (PNR) stage. The algorithm is validated using Matlab since it provides a good environment for image and video processing. It is more convenient to explore various solutions at the algorithm stage since it can be modified and verified quickly.

Once the algorithm is validated, test benches are created from the algorithm simulation to assist during hardware verification. This test bench is sometimes called the golden test bench since it is used as a reference throughout the design flow. This is important to ensure that the design performs correctly at each design stage.

Transferring from algorithm to hardware requires careful partitions to ensure an efficient implementation. Division between data path, memory and control is made at this stage. The

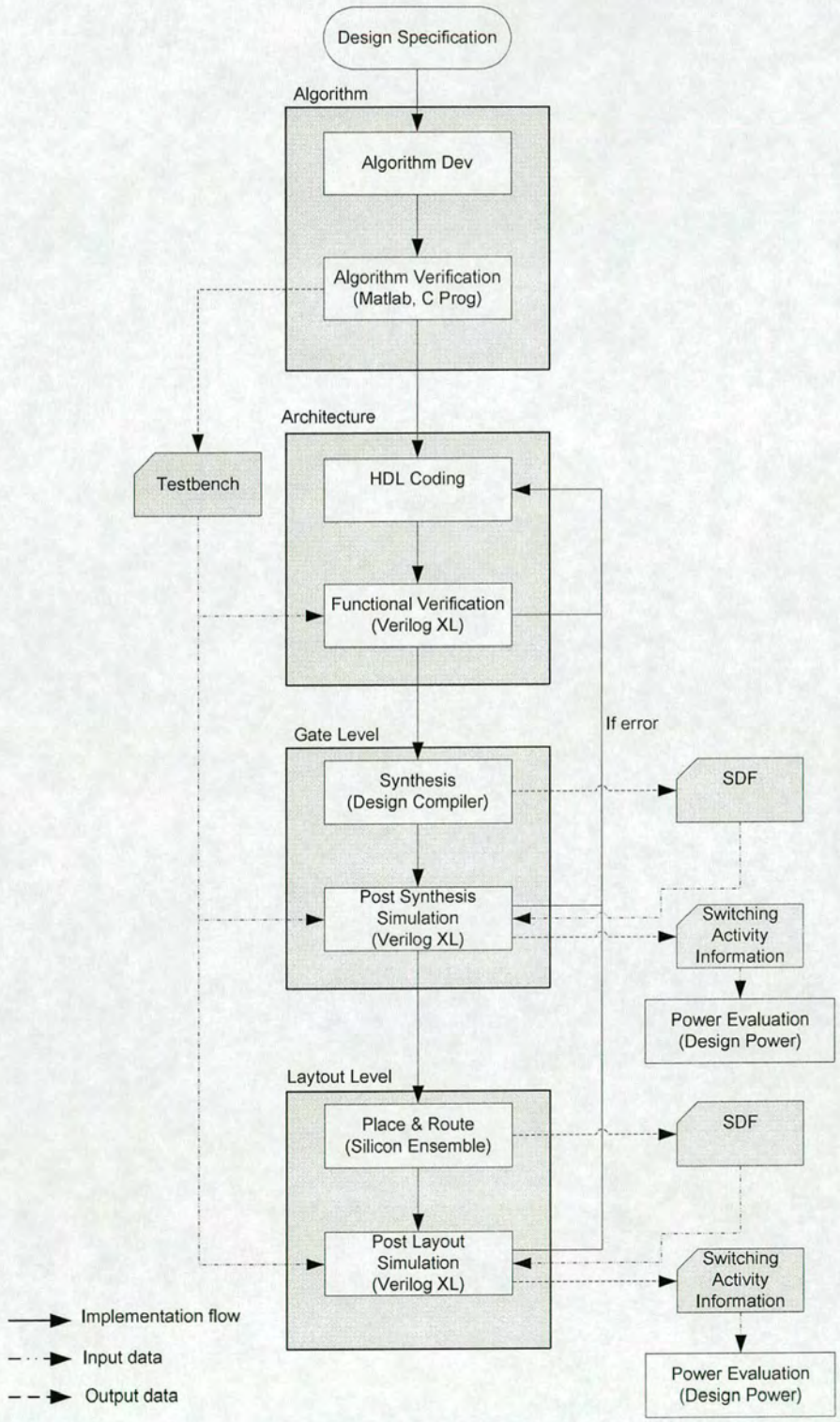


Figure 2.4: *Design flow for hardware implementation, verification and power evaluation.*

hardware is coded using the Verilog language and simulated using Verilog XL from Cadence Design System. At this stage, the golden test bench is used to verify the hardware.

Once the hardware coding is verified, it is mapped onto a technology dependent library using Design Compiler, a logic synthesis tool from Synopsys Inc. In this work, the UMC 0.13μm CMOS library, which consists of 557 cells and a memory compiler for 1-port, 2-port, and dual-port memories [39], is used. The gate level netlist is verified using the golden test bench and the switching activity is extracted. After the post synthesis simulation, the switching activity is used by the Power Compiler to calculate the power consumption.

While post-synthesis gate level power evaluation can give good power estimation, the interconnect capacitances are based on the wire load models provided by the technology library. For more accurate interconnect capacitance estimation, the post layout data are obtained. Silicon Ensemble, an automatic PNR tool, is used for routing the verified gate level netlist. The output is a netlist with more accurate interconnect capacitance extracted from the layout.

Throughout the design process, power consumption can be monitored at different design phases, such as after synthesis or after PNR. For fast comparison of different architectures, the post synthesis power estimates are used. For more accurate comparison in the final stage of the design, the power estimation is based on post-layout netlist.

As discussed in the previous section, power estimator provides a convenient way to estimate total power consumption of a design. However, for an event-driven power calculator such as the Power Compiler used in this work, the following factors affect its accuracy:

1. testbench dependency

2. cell library accuracy

3. assumption of constant operating environment such as voltage supply, temperature, fabrication process and electrically noiseless environment

While designers can tackle problem (1) by using a wide range of testbenches that reflect real world implementation, factors (2) and (3) are highly dependent on the accuracy of the technology library provided by silicon vendors, and the actual operating conditions when the chip is used. Despite these shortcomings, these tools provide very useful input for low power design. This is because the information provided by the power estimation tools allows designers to make power comparison between different architectures and offers good insight into how different architectures contribute to total power consumption. Furthermore, these tools give designers good estimation of the final power consumption with claimed accuracy of 5% to 17% of silicon results [40].

2.7 Summary

This chapter discussed the main sources of power dissipation in CMOS circuits and the existing solutions to minimise the problem. Throughout the history of electronics, the rapid advancement in human knowledge and creativity always pushes the existing technology to its maximum limit. This phenomenon has been happening to CMOS technology and power consumption has become the main problem in today's multi-million transistor designs. The main sources of power consumption in CMOS circuits are leakage, short circuit and switching power.

Many techniques have been proposed to reduce this power. This chapter reviewed some of the popular methods that have been shown to be effective in minimising CMOS switching power. The low power optimisation techniques can be performed at different levels of design abstraction such as algorithm, architecture, logic and technology. The sooner the problem is tackled during the design stages, the greater the power saving impact. Power estimation techniques were also reviewed in this chapter since they are crucial to determine the main bottlenecks in the design. In power estimation, accuracy is often traded off for faster run time. Finally, this chapter discussed the design flow that is used to evaluate power consumption throughout the thesis.

Chapter 3
Video Compression

3.1 Introduction

Digital multimedia applications have shown significant growth in the last few years. From personal computers, the application has become increasingly popular in mobile terminals such as mobile phones and personal digital assistance (PDA). One of the examples is mobile multimedia services (MMS) which have been widely adopted to send a short video message. This trend will advance further in the next generation wireless communication as video communication and wireless video distribution become widely available to the consumer market.

Video compression plays an important role in ensuring the success of today's video communication. It allows raw video data to be transmitted in current available transmission bandwidth or stored in minimal storage capacity. The success of video compression has allowed many other high quality video base applications to be widely available to the consumer such as video streaming, video broadcasting and video content distribution. It can be found in many electronics devices such as digital camcorders and DVD players.

This chapter presents an overview of video compression systems. While many video compression standards have been developed over the last two decades, the important principles and algorithms, which are similar among these standards, are highlighted in this chapter. The latest video coding standard, H.264, is reviewed and its impact on software and hardware

computational load is discussed to justify the requirement for a low power design. Since performance comparison between different algorithms is important to determine their efficiency, this chapter discusses benchmarking techniques and data that are widely used by the video coding community.

This chapter is organised as follows. Section 3.2 describes the typical raw video data and various formats used in video coding. The main building blocks for video compression are described in Section 3.3. Sections 3.4 and 3.5 describe new features in H.264 and how they affect the computational cost. Section 3.6 discusses benchmarking methods to compare video coding algorithms. Finally, Section 3.7 summarises the chapter.

3.2 Video Data

The basic element of video data is a series of still pictures that are captured at a specific time interval. Figure 3.1 shows a typical video camera system. When the picture is captured by a digital video camera, the source light falls onto a light sensor, such as a charge-coupled device (CCD) which is made up of tiny light-sensitive semiconductor diodes arranged in 2D. Each diode converts the light into an electrical signal recording the level of light intensity. This diode forms the basic unit of a picture which is called a picture element or pixel as shown in Figure 3.2 [41]. The quality of the picture depends on the number of sensors available. The denser the sensors are, the better the picture is, because more light can be captured per unit length. The pixel data is then stored into memory or displayed on screen.

The raw video data is differentiated by its frame's size and rate. The frame size is determined

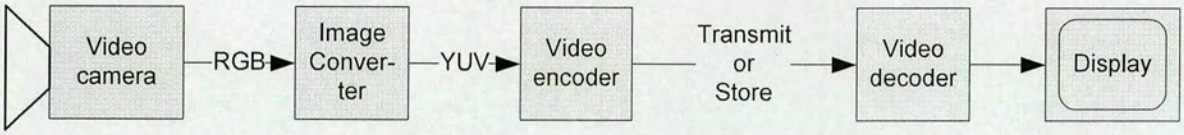


Figure 3.1: *Video camera system*

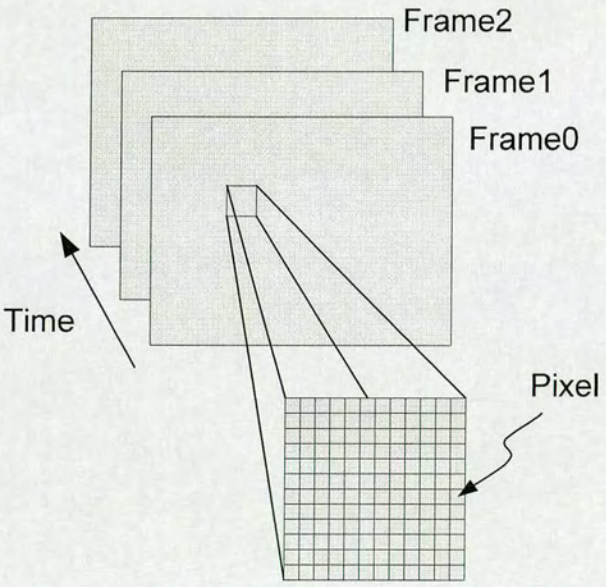


Figure 3.2: *Picture element (pixel)*

Typical frame size	QCIF	CIF	4CIF	VGA
	176×144	352×288	704×576	640×480
Typical frame rate (fps)	7.5,15, 30, 60			

Table 3.1: *Typical frame size and rates*

by total number of pixels, while the frame rate is the frequency at which the picture is captured (often expressed in frames per second, fps). The higher the frame rate, the smoother the object transition will be when perceived by the viewer. Some typical frame sizes and rates are shown in Table 3.1.

The original video colour is represented by red, blue and green (RGB). The combinations of these colours produce many other colours. In RGB format, all three colour components have equal importance. Thus, all three components need to be stored using the same resolution.

To reduce the amount of memory needed to store these colours, YUV format is introduced (also known as YCbCr). This format exploits the human visual system (HVS) which is more sensitive to brightness (luminance or luma) than to the colour component (chroma). It is more efficient to separate luminance from chroma, then represent the luminance using higher resolution than chroma. Thus, this format has the advantage of reducing the amount of stored data compared to the RGB format.

The overall brightness or luminance signal (Y) is achieved by adding the weighted R, G, and B colours using different factors. The colour components consist of two parts: U and V. The U signal is obtained by subtracting Y from B, followed by scaling, while the V is created by subtracting Y from R followed by scaling but using a different factor. Using the ITU-R recommendation BT.601, the conversion from RGB to YUV and YUV to RGB is given as follows [42]:

RGB to YUV:

$$Y = 0.299R + 0.587G + 0.114B \tag{3.1}$$

$$U = 0.564(B - Y) \tag{3.2}$$

$$V = 0.713(R - Y) \tag{3.3}$$

YUV to RGB:

$$R = Y + 1.402V \tag{3.4}$$

$$G = Y - 0.344U - 0.714V \tag{3.5}$$

$$B = Y + 1.772U \tag{3.6}$$

Several types of YUV formats are normally used depending on the resolution of the chroma per frame. Table 3.2 summarizes the existing YUV formats. The Hor and Ver columns show the ratio of chroma with respect to the luminance [43]. In this thesis, YUV 4:2:0 is used with

YUV	Hor(%)	Ver(%)
4:4:4	100	100
4:2:2	50	100
4:2:0	50	50

Table 3.2: *YUV Format*

8 bits per pixel since it is widely used in mobile video applications while delivering good picture quality. In this format, U and V each have half the horizontal and vertical resolution of Y. Thus this format requires half of the sample compared to RGB.

3.3 Video Compression

A five minutes video clip with appropriate size for a hand held mobile terminal (YUV420, CIF@15fps) requires about 648 Megabytes of data requiring a transfer rate of 18.2 MBits/s. This obviously shows that storing and transmitting raw video data is almost impractical using current devices. Thus, this data needs to be compressed before it can be transmitted through a wireless channel.

Video coding reduces the raw video data by exploiting the temporal and spatial relationship of video data. When the frame rate is sufficiently high, the data between neighbouring frames are highly correlated and have very small differences. This makes it feasible to compress video data by removing the redundancy between frames. The compressed video must be decodable and capable of handling any error during storage or transmission.

With the wide use of video applications, standardizing the video coding method is crucial to ensure interoperability between different devices and platforms [2]. Thus, video coding

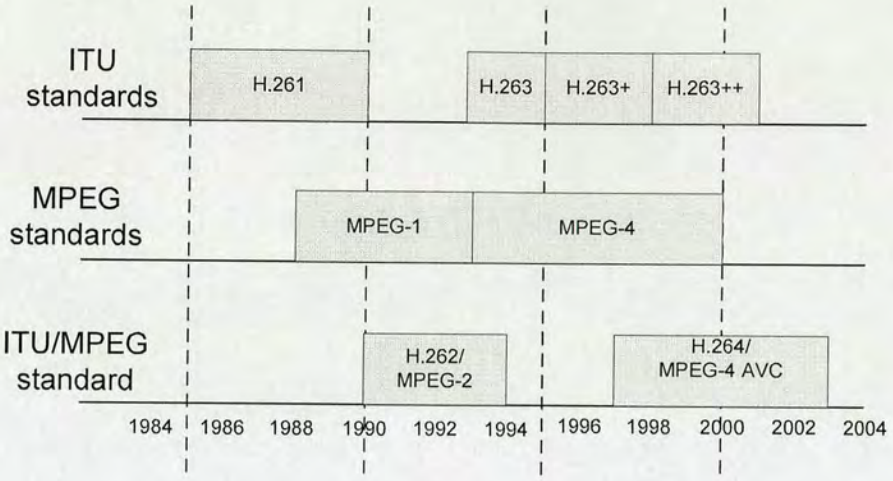


Figure 3.3: *Video coding standard history [43]*

standards have been introduced to the industry to allow a more efficient way to exchange video files between different parties. This permits various attractive applications and services to be offered to consumers. Furthermore, with wide acceptance of the standards, large scale production will follow which guarantees reduction in cost.

Video coding standards have seen a great improvement since its introduction in the last two decades. There are two main bodies actively involved in developing these standards: Motion Picture Expert Group (MPEG) and Video Coding Expert Group (VCEG). Figure 3.3 shows various video coding standards that have been introduced by these organisations.

MPEG is part of a joint technical committee known as ISO/IEC which is responsible to develop standards such as MPEG1, MPEG2, MPEG4, MPEG7 and MPEG21. The main objective of this group is to develop the video compression algorithm for commercialization and storage of digital video. On the other hand, VCEG is part of the International Telecommunication Union (ITU) which is actively involved in developing standards and recommendations related to video communication over telecommunication and computer networks. This group

has developed H.261, H.263, H.263+, and H.263++, and was also responsible for the initial development of the H.26L standard which started in 1999.

Due to their successful joint-venture during the development of MPEG2, the MPEG and VCEG combined their efforts to finalise H.26L. In 2003, the standard was officially accepted as an international standard which is also known as MPEG-4 Part 10 'Advance Video Coding' (AVC) by ISO/IEC and H.264 Recommendation by ITU.

In contrast to proprietary standards, such as Windows Media Video (WMV) by Microsoft and RealVideo by RealNetworks, the MPEG standard is an open standard which allows interested parties to study it and adapt their software/hardware to make it conform to the standard. The official standard document purposely describes only the decoding part which must be strictly followed. Since the encoding algorithm is not part of the standard, designers have greater flexibility to determine the best implementation for the encoder. This allows a large degree of freedom and encourages industrial players to compete in developing better encoder performance.

Typically, most popular standards adopt hybrid methods that combine motion prediction and transform coding to remove redundancy in frames. These approaches are fully exploited in major video coding standards such as those developed by MPEG and VCEG. Figure 3.4 shows typical hybrid coding functional blocks for the coder and decoder. The main components are motion estimation (ME), motion compensation (MC), discrete cosine transform (DCT), inverse DCT (IDCT), quantisation (Q), inverse quantisation (IQ), entropy coding (EC) and deblocking filter (DFIR). For ease of computation and data processing, each frame is divided into 16x16 pixel sections known as macroblocks (MB). Each macroblock is pro-

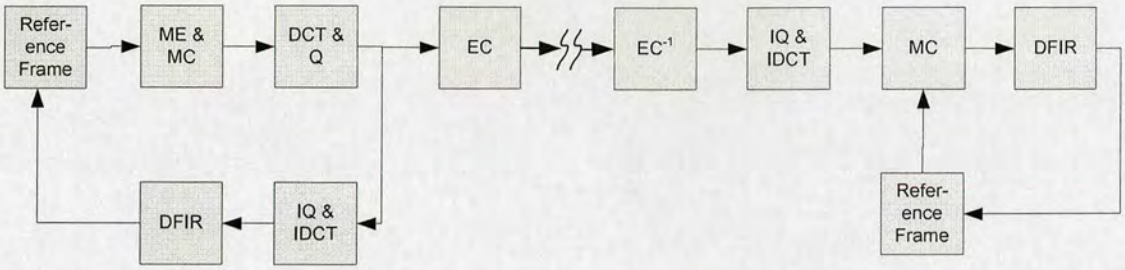


Figure 3.4: *Generic video encoder-decoder block diagram*

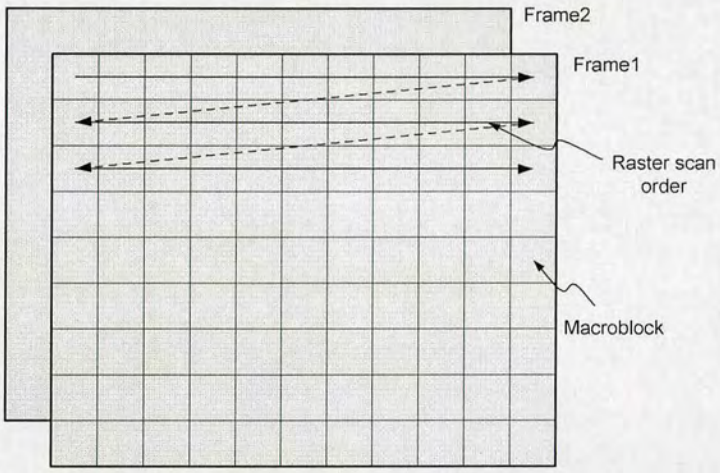


Figure 3.5: *MB raster scan*

cessed one at a time in raster scan order starting from the top left down to the bottom right as shown in Figure 3.5. The operation of each coding module is described in the following sections.

3.3.1 Motion Prediction

In a stream of frame sequences, each consecutive frame is almost identical and the difference between neighbouring frames is usually small. The main purpose of motion prediction is to remove temporal redundancy between frames. Compared to encoding the actual frame, encoding the difference between frames is more efficient since it contains less information than the original frame.

While directly subtracting the current frame and the neighbouring frames gives us the difference between these frames, the residues will be large if the motion in the current frame is high. To minimise this, the current frame is first predicted using the previously encoded frame as the reference. Then, the predicted frame is subtracted from the current frame to generate the residue.

One of the popular motion prediction schemes is block base motion estimation which has been widely adopted by the industry due to its simplicity and ease of implementation. In this method, each MB is predicted by finding the closest matching MB in the previously encoded frame. To limit searching time, this search is done within a defined search range of p pixels as shown in Figure 3.6. The location of the predicted MB with respect to the current MB is taken as the motion vector (MV). The best matched candidate is subtracted from the current MB to give the residue. These MVs and residues are the main output of the ME and can be used to reconstruct the predicted frame.

Finding the best match in motion estimation is an important problem since it will affect the overall performance of the video compression by influencing computation time, compression efficiency and power consumption. The prediction result is highly dependent on several factors, such as the matching criteria used during the search, the number of candidates involved in the comparison and the block size used during the search. One of the popular searching techniques is the full-search algorithm. In this technique, all possible candidates in the search area are evaluated. The most common matching criteria is sum of absolute difference (SAD),

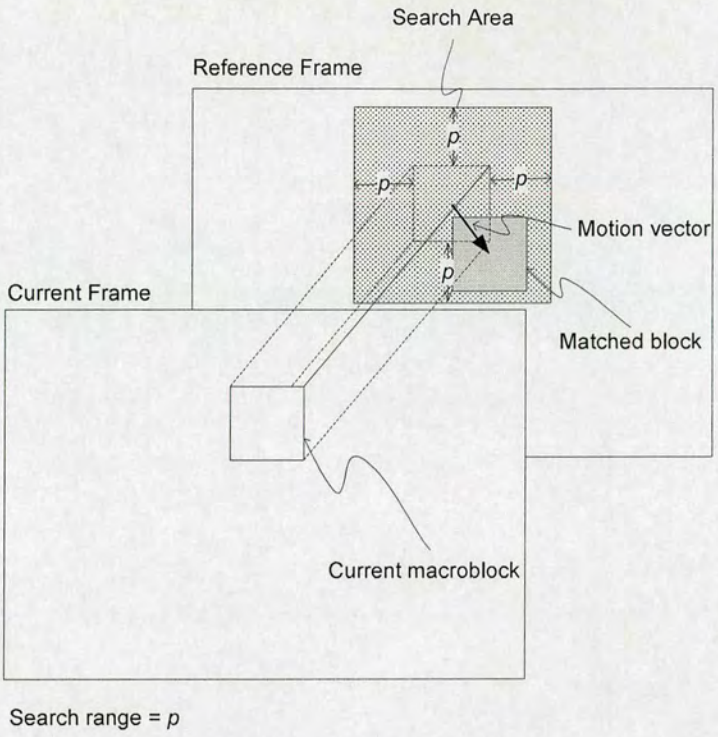


Figure 3.6: *Search area*

given as follows [44]:

$$SAD(i,j) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} |C(k,l) - R(i+k,,j+l)| \tag{3.7}$$

where $M \times N$ is the macroblck size, $C(k,l)$ is the current macroblock pixel at location (k,l) and $R(i+k,,j+l)$ is the candidate macroblock pixel located in the search window within the previously encoded frame (reference frame) with (i,j) is the motion displacement in pixels.

For a macroblock of size $M \times N$ and search range of p, the total number of search locations is $(2p+1)^2$; for frame size $W \times H$ and frame rate J fps, the total SAD computation is:

$$J \times \left(\frac{W}{M} \times \frac{H}{N}\right) (2p+1)^2 (MN) \tag{3.8}$$

This is equivalent to 3.3×10^9 SAD computations per second for $J = 30, W = 352, H = 288,$ $M = N = 16$ and $p = 16$. This can result in a very high computational load which can constitute a significant amount of the encoding process [45].

3.3.2 Transform Coding

While motion prediction exploits the correlation between neighbouring frames, transform coding exploits the correlation between neighbouring pixels. In this method, the data in the 2D spatial domain is transformed into the 2D frequency domain for more compact representation as shown in Figure 3.7. In this way, the image is represented using only a few byte of big values and the rest could be relatively smaller values. This translates to less energy when transmitted [46]. Since the high frequency domain is not visible to the human eye, this data can be removed while still maintaining overall picture quality.

One of the popular transform coding schemes that is widely used is discrete cosine transform (DCT) [47]. In this technique, the 2D data is transformed using the 2D-DCT as shown in Equation 3.9 where $N \times N$ is the block size, X is the 2-D input image sample where i,j are the spatial coordinates in the data sample and Y is the coefficient matrix where x,y are the coordinates in the transform domain. C_x and C_y are $\sqrt{\frac{1}{N}}$ when x,y equal 0 and $\sqrt{\frac{2}{N}}$ otherwise [42]. The resultant transform is in the 2D frequency spectrum, with low frequency and high frequency separated. The transform coefficient Y_{00} at the top-left corner of the

126	159	178	181
98	151	181	181
80	137	176	156
75	114	88	68

(a)

537	-76.0	-54.8	-7.8
-106.1	35.0	-12.7	-5.1
-42.7	46.5	10.3	-9.8
-20.2	12.9	39	-8.5

(b)

Figure 3.7: *Transform coding (a) image data (b) DCT coefficient values*

transform block is called DC coefficient because it contains the lowest frequency component. The magnitudes of low and high frequency components depend on the type of original spatial data. For a homogeneous plane, most of the frequency is concentrated at the lower frequency compared to objects with higher detail. The high frequency spectrum is normally quantised to reduce the number of generated bits. Equation 3.10 shows the equivalent inverse 2D-DCT.

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} cos \frac{(2j+1)\, y\pi}{2N} cos \frac{(2i+1)\, x\pi}{2N} \tag{3.9}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} cos \frac{(2j+1)\, y\pi}{2N} cos \frac{(2i+1)\, x\pi}{2N} \tag{3.10}$$

3.3.3 Entropy Coding

Unlike the previous two methods where images are treated as 2D data, entropy coding processes the data as 1D. One of the main inputs to the entropy coder is the set of quantised

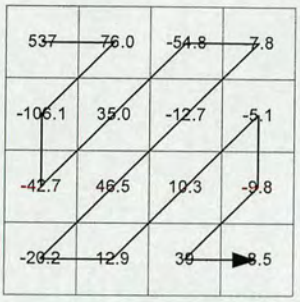


Figure 3.8: *Zig-zag scanning of coefficient data*

coefficients from the transform coder. To maximise the compression ratio, this data set is zigzag scanned from the low frequency toward the higher frequency as shown in Figure 3.8. This scanning ensures that there are more run zeros in the data string.

The scanned data is then mapped into a codeword using variable length coding such as Huffman coding or arithmetic coding [43]. This method utilises data probability; frequently occurring data is assigned with a shorter codeword, while less common data is assigned with a longer codeword. This permits the final data to be much smaller than the original. In addition to transform coefficients, other information, such as motion vector and encoder parameters, is also entropy coded.

3.3.4 Rate Control and Quantisation

During motion compensation, the amount of residue varies depending on the type of image and motion. For high motion frames, the generated residue is typically higher than the residue of the lower motion sequence, and vice versa. Thus, the generated bitstream may vary depending on the characteristics of the input sequences.

The variation in generated bitrates can cause problems during transmitting or storing the data.

To ensure the data rate produced by the encoder meets the specified bitrates, the generated bitstream is stored into a first-in first-out (FIFO) buffer from which it is emptied at the targeted bitrates. At the decoder, another FIFO is filled at constant bitrates, and its content is output at a variable bit rate.

The FIFO content at the encoder can over-flow or under-flow since the rate of the generated bitstream can vary. Thus, a rate-control mechanism is important to prevent this from happening. One popular way to do this is using the quantiser to regulate the data input into the FIFO.

During encoding, the FIFO content is monitored and the condition of the buffer will be fed back to the quantiser. In the case where the rate of data input into the FIFO is higher than the rate of the FIFO being emptied, the buffer can over-flow; to prevent this, the quantiser applies a higher quantisation value which causes more transform coefficients to be quantized into zeros. On the other hand, if the amount of data input into the FIFO is less than the rate of the FIFO being emptied, the buffer will underflow; to overcome this, a smaller quantiser value is used which causes less data to be removed.

3.3.5 Deblocking Filter

The data removed by the quantiser is non-recoverable, i.e. lossy. As a result, the image for a highly quantised MB tends to look blocky when reconstructed because the MB edges change abruptly compared to the original picture. Deblocking filters reduce the blocky effect by smoothing these edges using a filter. The filter smoothes the affected area by interpolating using the neighbouring pixels.

However, some MB edges exist originally due to object edges. Thus, the filter should be able to distinguish between blocking artefacts and real edges. To keep the original object edges from being filtered, an adaptive filter is used where the MB encoder parameter is used as a guide to differentiate between blocky artefacts and actual object edges [48].

3.4 H.264 Video Compression Standard

H.264 standard was officially releases in 2003 with a wide range of targeted video applications such as video conferencing, video streaming, video storing and video broadcasting. Since then, it has been getting wide attention from the industry because it is capable of achieving 50% compression improvement compared to its predecessors (60% compared to MPEG2 bitrate, 40% compared to MPEG4 ASP bitrate). There are two main improvements over the previous standard, prediction capability and coding efficiency [49]. The second release of H.264 was in 2005 which included support for increased pixel depth and higher color resolution. Detailed information, and the algorithm used during the development of the standard, are publicly available in [50] and [51].

H.264 defines several profiles such as Baseline, Main, Extended and High. In addition, several levels are defined for each profile. These parameters directly affect the computational load and hardware resources for the codec. The summary of each profile and level are given in Table 3.3 and Table 3.4, respectively. The comparison between H.264 with other standards is given in Table 3.5.

The H.264 standard still adopted a hybrid approach with differences in the detailed implemen-

Profile	Application
Baseline	Video telephony, Mobile video
Extended	Video streaming
Main	Video broadcasting, Entertainment
High	HDTV application

Table 3.3: *H.264 Profiles*

Level	Typical Frame Size	Typical Frame Rate	Maximum Bitrates (Mbs)	Maximum Reference Frame
1.0	QCIF	15	64	4
1.1	QCIF	30	192	9
	CIF	7.5		2
1.2	CIF	15	384	6
1.3	CIF	30	768	6
2	CIF	30	2000	6

Table 3.4: *H.264 Level and the corresponding picture type, frame rate and bitrates [52] [53]*

Modules	MPEG-2	MPEG-4 ASP	H.264 Baseline profile
Motion estimation			
- Block size	16x16	16x16, 8x8	16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4
- Quarter-pel precision	No	Yes	Yes
- Multiple reference frame	Up to 2	Up to 2	Yes (5 reference frame)
Intraprediction	DC	AC/DC	Yes (4x4 blk: 9 modes, 16x16 blk: 4 modes)
Transform	8x8 DCT	8x8 DCT	4x4 Integer transform
Entropy coding	VLC	VLC	VLC and CAVLC
In-loop deblocking filter	No	No	Yes

Table 3.5: *H.264 comparison with other standards [54]*

Module	Improvements
Motion Estimation	variable block-size motion compensation with small block sizes
	quarter-sample accurate motion compensation
	multiple reference picture motion compensation
Transform Coding	small block-size transform
	short word-length transform
	exact match inverse transform
Spatial prediction	directional spatial prediction
Deblocking filter	in-the-loop deblocking filter
Entropy coding	arithmetic entropy coding
	context-adaptive entropy coding

Table 3.6: *H.264 improvement on each module*

tation of each module. Apart from the main functional blocks, as discussed in the previous section, additional functional blocks, such as spatial prediction and deblocking filters, are included as part of the standard. Table 3.6 shows some of the improvement of H.264 for each functional module [55][52][53].

Figure 3.9 illustrates the main functional blocks of H.264. The encoder consists of motion prediction, spatial prediction, integer transform and quantiser, entropy coder, and deblocking filter. The main function of the encoder is to encode the current MB into a compact bitstream so that it can be decoded by the decoder.

In H.264, the encoder first predicts the current MB using both motion estimation and spatial prediction. The predicted MB that results in the lowest cost will be selected. The residue of the predicted MB will then be calculated by subtracting the current MB and the predicted MB, before it is transformed by the integer transform. The transform coefficient and motion vector are coded using the entropy coder to generate a compact bitstream. The encoder also reconstructs the current MB using information from transform coefficients, and filter these before storing the MB into the reference frame buffer for future frame prediction.

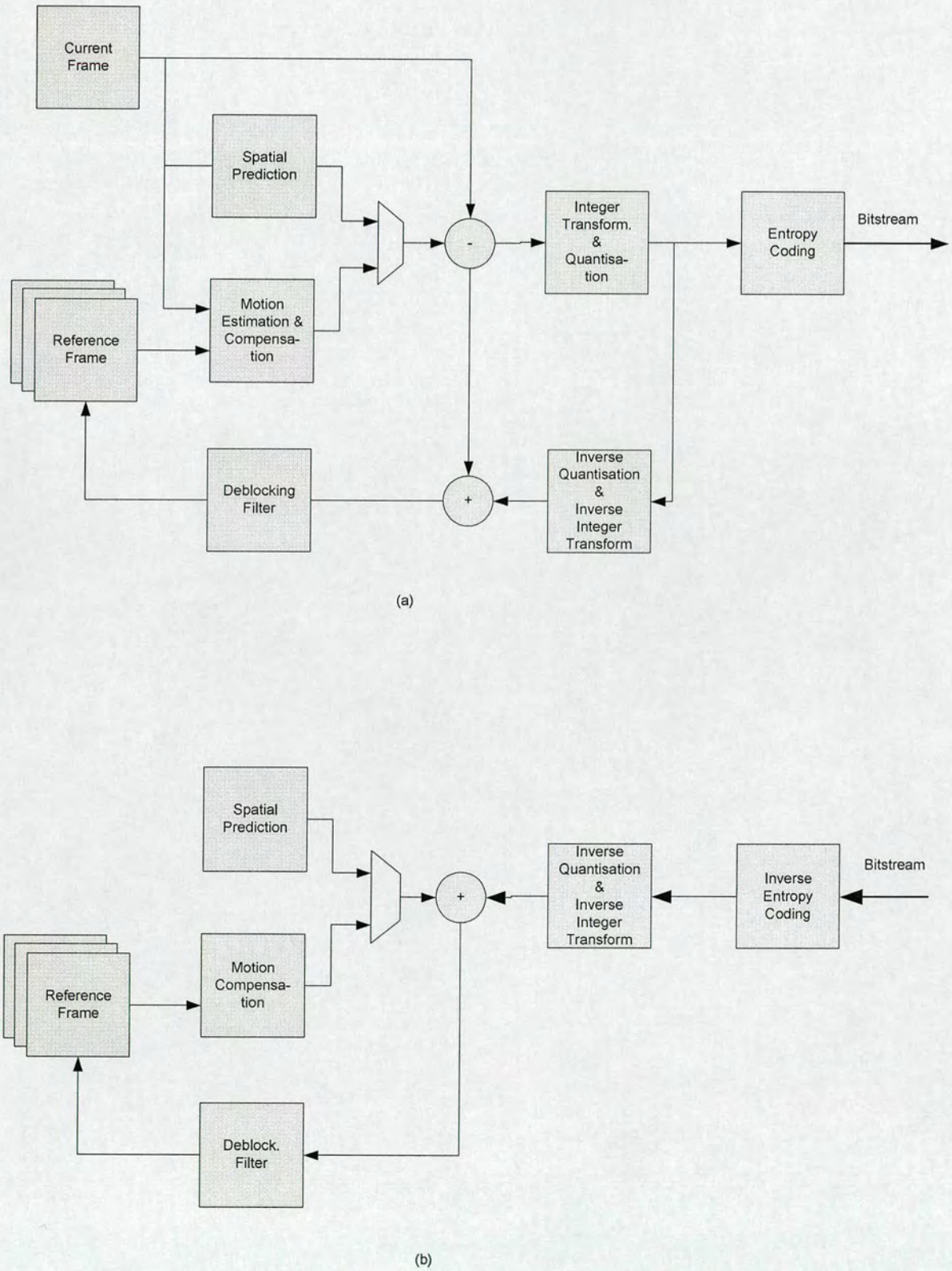


Figure 3.9: *H.264 block diagram (a) encoder (b) decoder*

The H.264 decoder functions as an inverse operator of the encoder. It first decodes the bit-stream which consist of, among others, MB information and transform coefficient. The coefficient is inversely transformed to recover the MB residue. Based on the decoded MB information, the decoder generates the predicted MB using either motion compensation or spatial prediction. The current MB is recovered by adding the predicted MB and the MB residue. As in the encoder, the current MB is filtered and stored into the reference frame buffer.

3.5 H.264 Computational Load

During the development of a video compression standard, the algorithm is developed with the main focus on the improvement of PSNR, visual appearance and bitrates [56]. When the standard is approved, the document that describes the decoding part is released together with the accompanying software model. The main objectives of the reference software are to verify the H.264 algorithm and to allow various parties to benchmark the standard capabilities with other video coding standards.

Since the target platform is unknown., the software model is not optimised for hardware implementation. Thus, complexity analysis of the algorithm is important during the early stages of hardware development for multimedia systems. The main goals are [56]:

- to assess the performance and implementation cost of the new multimedia algorithm
- to provide feedback for hardware system development
- to guide complexity reduction during early algorithmic development

As discussed in the previous section, H.264 improvement comes at the expense of an increase in computational complexity. The reference software for H.264 [51] shows ten times more simulation time compared to the previous MPEG-4 model [54]. Therefore, investigation of the computational requirement for each module is important to identify the main bottleneck in the algorithm. The most time consuming operations will be performed using hardware acceleration to increase the throughput.

Two approaches can be used to perform computational load analysis, namely the static and dynamic methods. The static approach measures computational load by manually counting the number of operations, memory accesses and address calculations needed for the algorithm [57]. In the dynamic method, the algorithm is simulated and the run time for the operation is monitored using software-only or hardware-assisted techniques [44].

In the software-only collection method, the run time of each function is monitored. The computational complexity is assumed to be proportional to the simulation time. The functions that consume the largest portion of simulation time are identified as the main bottleneck. This method normally utilises the program compiler utility, such as the GNU profiling tool (i.e. gprof), to monitor the runtime of each function. While this method provides the easiest way to examine algorithm complexity, the result can be affected by other factors such as the operating system, memory usage, etc. This method is used to profile the H.263+ algorithm in [58].

In the hardware-assisted method, the computational complexity information is obtained by running the software on a specific platform such as RISC processor [54]. The instruction-level information, such as arithmetic, control and data transfers, are profiled. For arithmetic,

Functions	Arithmetic		Controlling		Data transfer		
	MIPS	%	MIPS	%	MIPS	Mbytes/s	%
Integer-pel ME	95,491.9	78.31	21,915.1	55.37	116,830.8	365,380.7	77.53
Fractional-pel ME	21,396.6	17.55	14,093.2	35.61	30,084.9	85,045.7	18.04
Fractional-pel interpolation	558.0	0.46	586.6	1.48	729.7	1067.6	0.23
Lagrangian mode decision	674.6	0.55	431.4	1.09	880.7	2642.6	0.56
Intra prediction	538.0	0.44	288.2	0.73	585.8	2141.8	0.45
Variable length coding	35.4	0.03	36.8	0.09	44.2	154.9	0.03
Transform and quantization	3223.9	2.64	2178.6	5.50	4269.0	14,753.4	3.13
Deblocking	29.5	0.02	47.4	0.12	44.2	112.6	0.02
Total	121,948.1	100.00	39,577.3	100.00	153,469.3	471,299.3	100.0

Table 3.7: *Computational complexity for H.264 Baseline profile [54]*

specific operations such as multiplication, addition, subtraction, shift , or division are also monitored in detail [59].

Table 3.7 shows the instruction level analysis of H.264 on an Ultra Sparc II processor (Baseline profile, 30 CIF fps, 5 reference frames, ±16-Pel search range, and quantiser parameter (QP)= 20) [54]. From the table, the H.264 encoder required more than 300 Giga instructions per second (GIPS) and data transfer greater than 460 GBytes/s. The integer-pel motion estimation consumes 78% and 77% of the total arithmetic and data transfer operations, respectively, of the H.264 system. In order to accelerate this computation, dedicated hardware is crucial to perform this computation-intensive part and minimise the power consumption.

3.6 Video Quality Measures

In video compression, the quality of decoded video is an important parameter to be considered in addition to the compression ratio. While the compression ratio can be directly calculated from the output bitstream, video quality measures the difference between the re-

constructed video and the original video. This comparison can be quantified in two ways, namely subjective and objective measures.

Subjective quality utilises human observation to decide the quality of a video sequence. In this method, several videos are shown to a group of people (20-25 people) under certain settings, and scores are given by every individual in the group. At the end, the collective score is taken as the final result [2]. One such example is the single stimulus continuous quality scale technique (SSCQS) [60]. In this method, time varying picture quality of the processed video, without references, is evaluated by the subjects. The video is divided into 10 second segments and for each segment, five scores can be given: bad (1 - 20), poor (21 - 40), fair (41 - 60), good (61 - 80) and excellent (81 - 100). The mean opinion score is taken as the video quality.

In objective quality measures, the video quality is compared using mathematical formulae based on mean squared error (MSE). The peak signal to noise ratio (PSNR) between the original frames and reconstructed frames is calculated as:

$$PSNR = 10log_{10}\frac{PMAX^2}{MSE} \tag{3.11}$$

where $PMAX$ is the maximum possible pixel value of the image data. From the equation, the higher the PSNR, the better the video quality.

The subjective quality test can yield a good result since it directly uses human viewing and perception expectations which are close to the real world. However, there are several factors that affect subjective quality results [2]:

- the importance of distortion to the viewer (e.g. the face region is usually more important than background)
- the viewer's expectation/familiarity (the more familiar the user is with higher quality video, the higher sensitivity to the subjective quality)
- the viewer's expertise level (general audiences normally prefer blocky images whereas video coding expert audiences prefer smooth images)
- the frame size (distortion in larger frame sizes is more obvious than in smaller frame size)

Nevertheless, the subjective measure is more reliable than the objective measure. This is because the objective measure does not always correlate well with perceived video quality. However, subjective quality experimentation is time consuming and expensive. Objective quality testing is much simpler and can give faster results that make it more popular among video coding community. Furthermore, for higher PSNR, the subjective quality can be better, but not worse [2].

Since there is no standard way to measure the picture quality, PSNR value is used as a measure throughout this thesis. This helps in comparing the results in this thesis with other existing work in the literature. In cases where accuracy is important, subjective quality will be used as the final metric.

In order to create a meaningful comparison for different video coding algorithms, MPEG has established a set of test sequences to standardize benchmarking. By using the same test inputs, the performance of different coding can be compared more fairly. These test

Sequence class	Content complexity
A	Low spatial detail and low amount of movement
B	Medium spatial detail and low amount of movement or vice versa
C	High spatial detail and medium amount of movement or vice versa
D	Stereoscopic
E	Hybrid natural and synthetic

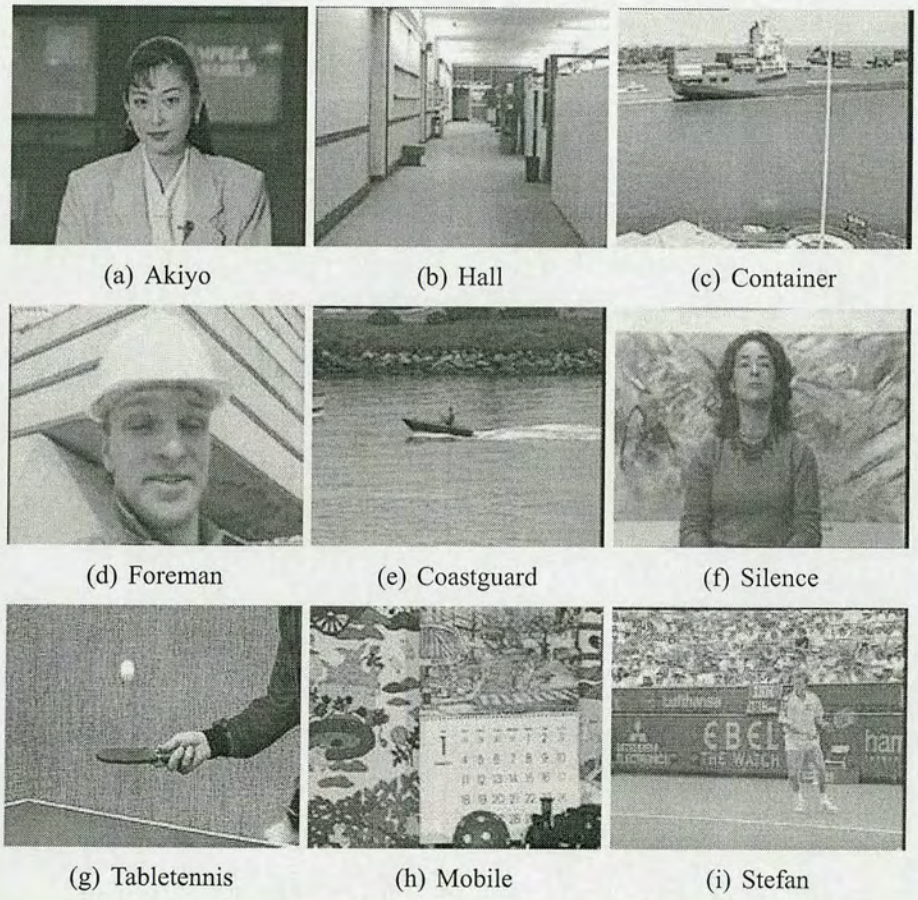
Table 3.8: *MPEG video sequence categories*

Name	Class	Total Frm	Description	Application
Akiyo	A	300	Head & Body	Live news broadcasting
Hall	A	300	Room	Security camera
Container	A	300	Outdoor (sea & sky)	Event recording
Foreman	B	300	Head & sholder + shaky camera	Videophone
Coastguard	B	300	Outdoor (sea + tree)	Recording
Silent	B	300	Head & Body + still camera	Video conferencing
Tabletennis	C	300	Zooming + change in camera	Sport
Mobile	C	300	In door + camera slow movement	Indoor recording
Stefan	C	300	outdoor + fast camera movement	Sport

Table 3.9: *MPEG video sequences description*

sequences contain various scenes which are typical in video communication applications. These video sequences are divided into 5 categories (A, B, C, D and E) as shown in Table 3.8 [61]. The classification is based on the level of spatial detail and the amount of movement in the video sequences.

In this thesis, video sequences from classes A, B, and C, which represent typical scenes in mobile communication, are used. These video sequence descriptions are summarised as shown in Table 3.9 and the snapshot of each video sequence is shown in Figure 3.10.



(a) Akiyo (b) Hall (c) Container

(d) Foreman (e) Coastguard (f) Silence

(g) Tabletennis (h) Mobile (i) Stefan

Figure 3.10: *MPEG video sequences snapshots*

3.7 Summary

This chapter has introduced video coding principles and associated algorithms. Raw video data that are used as the input for the compression engine were first presented. Standardization of coding is crucial for ensuring the wide spread use of this application. The history of video coding standards by the two main organisations, namely MPEG and VCEG, was reviewed. The standards have evolved over the last two decades and a lot of improvement has been made to increase the compression efficiency. These standards share common processing blocks which are fully utilised in today's video; these important blocks were reviewed in this chapter.

The latest video coding standard, H.264, offers superior performance compared to its predecessor but with an increase in computational load. The main feature that contributes toward this improvement was reviewed and the main bottleneck for low power design was discussed.

Comparing generated video between different coding algorithms is indispensable in video coding research. Two possible methods, the subjective and objective quality tests for performing fair comparison between different video outputs, were discussed. Finally, the standardised set of test sequences was discussed which is important to benchmarking different video algorithms.

Chapter 4
Low Power Works on Video Compression

4.1 Introduction

Current limitation in battery life poses a critical challenge to mobile devices. On average, current mobile devices can play a short video clip before the battery runs out. This seriously limits the user's experience and could be a major drawback in developing more attractive applications . Before real-time mobile video communication can be widely accepted, these problems must be addressed seriously by the industry [62][63].

Due to the higher level of computation, much work has been dedicated to reduce the computational power used by video codec. Typically, the encoder consumes more power compared to the decoder. In this chapter, the existing techniques to minimise power in video codecs will be reviewed. In general, the existing techniques exploit the unique features of video data and algorithms such as:

- High correlation between neighbouring pixels
- Asymmetric data switching
- Uniform and repetitive memory access
- Repetitive computation/looping

This chapter presents an overview of existing techniques for low-power MPEG-4 design. The low-power techniques used in motion estimation and transform coding are highlighted since these units contribute significant power to the MPEG-4 hardware. Tackling power consumption at a system level is also explained, since it can provide significant power reduction to the overall system. Video coding requires a high volume of computation and data transfer. Additionally, reducing power at the memory and datapath is also discussed.

This chapter is organised as follows: Sections 4.2 and 4.3 discuss the low power techniques for ME and transform coding, Section 4.4 focuses on minimizing MPEG-4 power at the system level, and Section 4.5 describes the memory power reduction techniques. Finally, Section 4.7 concludes this chapter.

4.2 Low Power Motion Estimation

Minimising power in a motion estimation block/unit involves reducing:

- the number of candidates
- the matching criteria complexity
- the pixel resolution

Reducing the number of candidates results in many fast algorithms such as three step search (3SS) [64], 2-D Log Search [65] and New Three Step Search (NTSS) [66]. These methods assume that the matching error increases monotonical as the search move away from the best location [65]. Since it only checks a few points, it has a tendency to find local minima.

In most video sequences, the majority of the MV lies around the search origin (0,0). This characteristic is utilised to reduce the search candidates used in Diamond Search (DS) [67] and Spiral Search [68]. In this technique, the search starts from the centre location and moves toward the minimum matching error position.

Researchers in [69] utilise a hierarchical search where the candidate is found at several frame resolutions with an increase in frame resolution after each step. The first search is done at the lowest frame resolution (coarse level) which requires less data compared to full frame resolution. The search is refined at a higher frame resolution centred around the result of the previous search [70][69].

Since the neighbouring MB normally moves in a similar direction, its motion can be predicted using these MBs. A motion adaptive search (MAS) [71] for fast motion estimation uses these spatially adjacent MBs (left, top and top-right neighbours) to predict the current block MV and adjust the search range and strategy. The prediction is then refined using DS with the predicted MV as the search centre. To improve the motion prediction, a prediction motion vector field adaptive search technique (PMVFAST) combines the general predictive selection, where MB in the previous frame is also used in MV prediction. The search is terminated once the calculated minimum distortion exceeds the threshold values. This technique works four times faster and 0.73dB higher than DS [72]. [73] proposes fast elimination of impossible motion vectors for full search algorithms. Three main properties are utilised in this technique: sub-block based matching, predetermined dithering order of matching scan, and complexity-based block matching. 30% computation reduction is achieved compared to conventional fast full-search algorithm.

4.3 Low Power Transform Coding

Since transform coding consumes a significant amount of computational resources, various methods have been introduced to lower the power consumption. [46] minimises the power consumption in distributed arithmetic-based DCT (DA) by dynamically reducing the coefficient precision to the minimum required level, depending on the quantisation value. This eliminates unnecessary high-precision computations and allows unused subsystems to be turned off. [74] exploits the MSB of locally correlated pixels in DA. Since these bits represent the DC offset value of the DCT coefficient, it does not contain significant information in computing a high frequency coefficient. Removing the common MSB shows up to 55% power reduction. Since most of the DCT coefficients will be quantised to zero, [75] uses this characteristic to shut down any circuit that results in a zero coefficient.

Reference [76] proposed reconfiguring asynchronous datapaths by adapting the structure to their computational requirements. Since the number of non-zeros is varied and normally consists of a small fraction of the coefficient, the unused datapath is disabled. [77] minimises the power consumption by trading off between accuracy and power consumption. It has been shown that removing the first 25% of the DCT coefficient does not significantly affect the output signal quality. 55% savings in energy is obtained by exploiting this characteristic.

4.4 Low Power MPEG-4 System

Rate control (RC) is important in video codecs, where it controls the generated bitstream and output picture quality. However, the software base RC using rate-distortion (RD), as in H.264

references software, is very expensive to be implemented in hardware. In this approach, the RD cost for all motion prediction errors are precalculated before the final decision is made. This involves re-accessing the same pixels to calculate different prediction modes. To overcome this problem, [78] introduced a model-base RC. This method predicts RD parameters without accessing the current input frame. This is accomplished by using the MAD based on calculated motion prediction errors of the last encoded frame. This prediction is valid as long as the neighbouring frame is closely correlated with the current frame.

The work in [79] exploits the context variation and human perceptual tolerance to reduce power consumption in video codec implemented on an adaptive SOC architecture. The hardware is configured at run-time through an adaptive interconnect that can dynamically reconfigure the hardware to select one of three ME cores: full, spiral, and three-step search. The power reduction is achieved by balancing the interconnect reconfiguration while the idle core is turned off. The author claims that the dynamic configuration requires less than 10% overhead from interconnect bandwidth and power consumption.

In contrast to low power design where video quality/bitrates are often traded off for power, power-aware design attempts to achieve maximum video coding gain under certain power dissipation constraints [80]. To facilitate this, the encoder is tuned on a scalable architecture to achieve the desired performance target [81] [82] [83][84]. Other general low power methods are widely being used to minimise power at the MPEG-4 system level, such as dynamic voltage/frequency scaling [85][86][87].

4.5 Low Power Memory for Video Compression

Video compression is a memory-intensive operation. In many applications, the frame buffer is located off-chip due to its large size to store the encoded frames. Accessing the off-chip memory during the motion prediction requires a large amount of power. This is because the capacitance for the off-chip buses are several orders of magnitude larger than the on-chip busses. In order to reduce the memory bandwidth for video compression, much research has focused on optimising the memory data transfer.

[88] investigates the trade-off in memory hierarchy to reduce data access and power from the main memory. However, this introduces extra memory transfer, increases the area and interconnects. The advantage of using the hierarchy memory is gained if the power saved by using smaller memory exceeds the power consumed by the additional memory transfer. [89] split the memory hierarchy into five categories by size: frame memory, buffer memory, search memory, MB buffer, and block size buffer. [90] divides the ME memory into four categories, depending on its reusability. The memory can be reused at the candidate and search-area levels by reusing its neighbouring strip of candidates or search areas. The data reuse reduces the bandwidth at the expense of additional memory area. [88] reduces the power consumption by using custom memory organisation that exploits the temporal location in memory access. In this approach, the frequently accessed data is placed into smaller memory in a short period of time. This reduces the burden of requiring much larger memory. [91] studies the possibility of optimising the data transfer and storage for data-dominated applications since these two parameters contribute significantly toward codec power consumption.

The asymmetric characteristic of the video data has been exploited to minimise power in the

memory circuit. In [92], Adaptive Bit Compression exploits the fixed pattern of the frame memory access characteristic. This method divides the pixel bits into MSBs and LSBs. Since the frame is accessed MB by MB, each neighbouring pixel in MB is highly correlated, and it is not necessary to activate the whole bit during each memory access. In this method, if the neighbouring pixel's MSBs are the same, it stores only the LSB. Otherwise, all bits will be stored. This method requires an additional bit to indicate whether or not the bit is compressed.

Another important component in video compression is the address generator. Since standard video processing employs hierarchy layer processing, an efficient addressing method for frame pixels is proposed by [93]. In this method, a bit-allocation addressing technique is examined. The result shows that this technique reduces the I/O complexity and shortens the access time.

In a general purpose DSP processor, almost half of all DSP cycles consumed by memory wait cycles are due to massive access to external memory. Since the access speed of the external memory is slower than the DSP, performance degradation in the processor will result. To overcome this problem, [94] introduces a software-based DMA queue. In this technique, the codec can request DMA transfer. As a result, the DSP can simultaneously execute video codec operations and data transfer between DSP internal and external memory. Since the data is ready before it is needed, it avoids the waiting cycle.

4.6 Further Discussion

As discussed in Chapter 3, the latest video coding standard, H.264, improves the compression efficiency and picture quality. However, these advantages come with an increase in computational complexity compared with previous video compression standards. In Section 3.5, motion estimation required significant processor computation time (>75%) compared to other operations, and this translates to higher power consumption. With the introduction of variable block sizes and multiple reference frames into the motion estimation, the computation and memory bandwidths for this module increase significantly. Thus, minimising the power consumption for this module is crucial.

Several low power techniques for motion estimation have been discussed in Chapter 4. However, the existing techniques focus on fixed block size motion estimation. Implementing these techniques directly into variable block size motion estimation will not result in optimal performance in terms of prediction quality [95]. Thus, a low power solution for variable block size motion estimation is important to ensure that power is minimised during motion prediction.

Furthermore, multiple reference frames introduced in H.264 result in high memory bandwidth on off-chip buses. This will cause high power consumption on these buses. Section 4.5 has described the existing techniques to minimise power due to high data transfer. Since these techniques are targeted for single frame transfer, these techniques do not exploit the correlation that exists between frames to reduce the power consumption.

To solve the problems arising from these two new features in H.264, the rest of the chapters

will investigate the effects of variable block size and multiple reference frame transfer on the power consumption by motion estimation. Then, a solution to minimise the power consumption due to these features will be proposed for the motion estimation hardware and off-chip bus communication that consume the largest amount of power in H.264 system.

4.7 Summary

This chapter has presented existing techniques for designing low power hardware for the MPEG-4 system. The chapter began with a description of techniques used to minimise power for the main MPEG module, such as motion estimation and transform coding. At the system level, techniques that show effective power reduction such as simplified rate control, adaptive hardware, and power-aware system were described. Memory was shown to be an important component in MPEG systems. The techniques to minimise power at the memory level were also highlighted in this chapter. Finally, the directions for the rest of the chapters are also discussed.

Chapter 5

Low Computation and Memory Access for Variable Block Size Motion Estimation using Pixel Truncation

5.1 Introduction

Motion estimation (ME) has been identified as the main source of power consumption in video encoders. As discussed in Section 3.5, motion estimation consumes more than 75% of arithmetic and data transfer operations in H.264 system. This due to the high computational load needed to predict the current frame. Thus, minimising the power consumption for this module is crucial. From Equation (3.7), the power consumption in motion estimation is affected by the number of candidates and the total computation to calculate the matching cost. Thus, the power can be reduced by minimising these parameters.

Pixel truncation can be used to reduce the computational load by allowing us to disable the hardware that processes the truncated bits. While previous studies focused on fixed-block-size motion estimation (16x16 pixels), very little work has been done to study the effect of pixel truncation for smaller block sizes. The latest MPEG-4 standard, MPEG-4 AVC/H.264, allows variable block size for motion estimation (VBSME). [4] defines 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 block sizes. At smaller block partitions, a better prediction is achieved for objects with complex motion.

In this chapter, a low-power method for motion estimation using pixel truncation for smaller block sizes is presented. The search is performed in two modes: (1) truncation mode; and (2) refinement mode. This method reduces the computational cost and memory access without significantly degrading the prediction accuracy.

The rest of this chapter is organised as follows. The existing techniques of low resolution ME are reviewed in Section 5.2. Section 5.3 investigates the effect of pixel truncation on variable block size motion estimation. Section 5.4 outlines the proposed two step search for VBSME. The experimental results are discussed in Section 5.5. Finally, Section 5.6 concludes this chapter.

5.2 Low Resolution Motion Estimation

In low resolution ME, the bit size and computational cost are normally tackled simultaneously. [96] introduced a one-bit transform (1BT) to reduce the computational cost. In this method, the original image is filtered using a band-pass filter. The output image is represented by one bit and the ME is carried out at this frame plane. To improve the frame prediction, [97] proposes a two-bit transform (2BT) where the original image is converted into two bits using the threshold value derived from the local image standard deviation. More than 0.2 dB improvement is achieved with this method as compared to 1BT. [98] proposes a low resolution quantised ME (LRQME) where the pixel is transformed to two bits using an adaptive quantiser. To produce the two bit image, three quantisation thresholds are calculated according to

the current MB pixel mean as:

$$t_3 = -t_1 = \frac{3}{2MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |p(i,j) - \bar{p}| \tag{5.1}$$

$$t_2 = 0 \tag{5.2}$$

where $p(i,,j)$ is the pixel value of the current MB, $M \times N$ is the MB size and $\bar{p}$ is the average value of current MB pixel calculated as follows:

$$\bar{p} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} p(i,j) \tag{5.3}$$

In motion estimation, pixel truncation has been used to reduce the computational load for the matching calculation unit [99]. [100] proposes adaptive pixel truncation during ME. The pixel's least significant bits (LSB) are adaptively truncated depending on the quantisation parameter (QP). This direct quantise provides a trade-off between PSNR and power. Truncating the pixel's most significant bits (MSB) was discussed in [101].

In low resolution ME, most methods focus on reducing the power by minimising the computational load. However, memory access is not taken into account. This is because the original pixel needs to be accessed before it is transformed into low resolution. In some cases, where the PSNR is dropped due to truncation error, a second search is done at full resolution. This increases the memory access and thus increases the power consumption. On the other hand, while direct pixel truncation has the potential to reduce memory access, it is often at the

expense of a decrease in PSNR in some motion types.

Truncating pixels at a 16x16 block size results in acceptable performance, as shown in the literature [100]. However, at smaller block sizes, the number of pixels involved during motion prediction is reduced. Due to the truncation error, there is a tendency for smaller blocks to yield matched candidates which could lead to the wrong motion vector. Thus, truncating pixels using smaller blocks results in poor prediction.

5.3 The Effect of Pixel Truncation for VBSME

For video applications, data is highly correlated, and the switching activity is distributed non-uniformly. Figure 5.1 shows the switching activity profile for typical input data for a motion estimation. The switching activity is calculated using Matlab with four different frame sequences which represent a sequence from low motion to high motion. The pixels are scanned in a block-based manner which is typical in video application. The switching activity is calculated using five frames from each sequence with 8-bit data size per pixel. As shown in Figure 5.1, since the less significant bits (LSB) of a data word experience a higher switching activity, significant power reduction can be achieved by truncating these bits.

Figure 5.2 shows the equivalent total switching activity for all bit positions as the number of truncated LSBs increases. In general, about 50% reduction in switching activity is obtained if up to 3 LSBs are truncated. Further reduction can be achieved if the number of truncated bits (NTB) is increased. For example, if NTB is set to 6, the switching activity could be reduced by 80-90%. This makes pixel truncation attractive to minimise power in motion estimation.

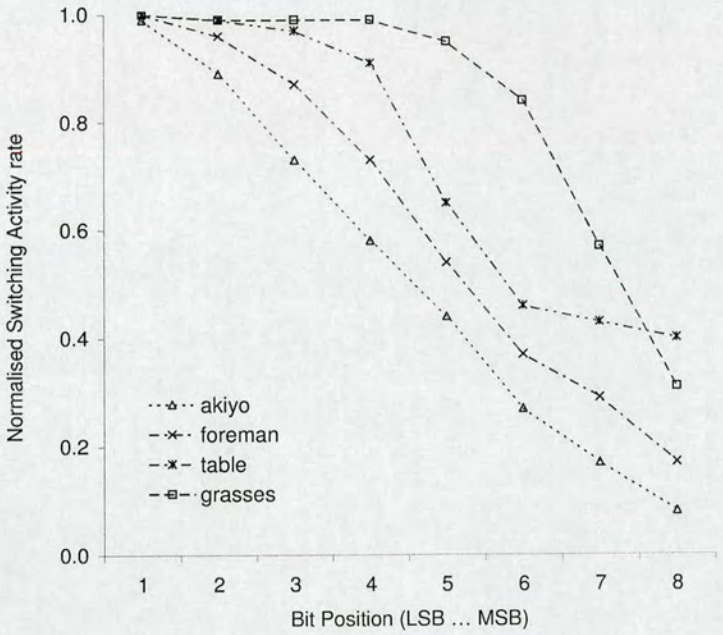


Figure 5.1: *Bit switching activity for video data.*

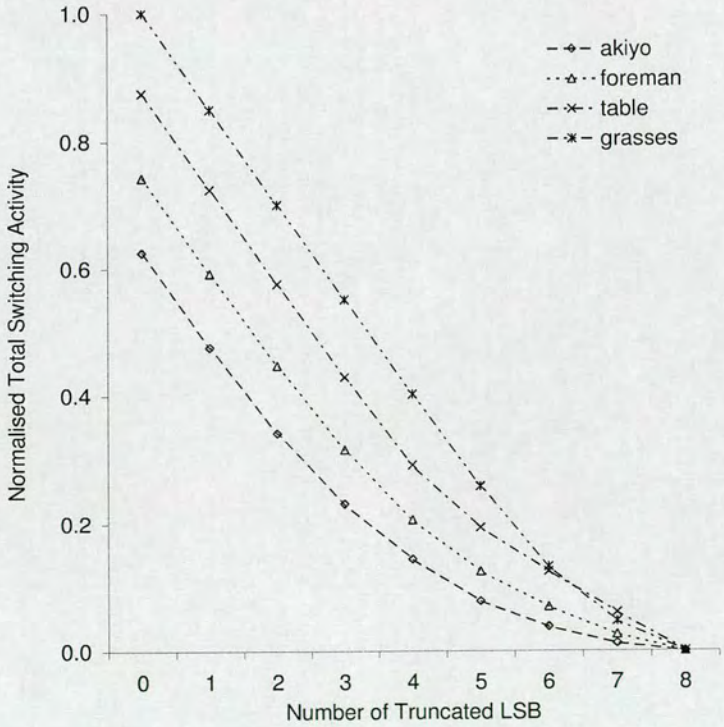


Figure 5.2: *Total switching activity vs. number of truncated LSBs.*

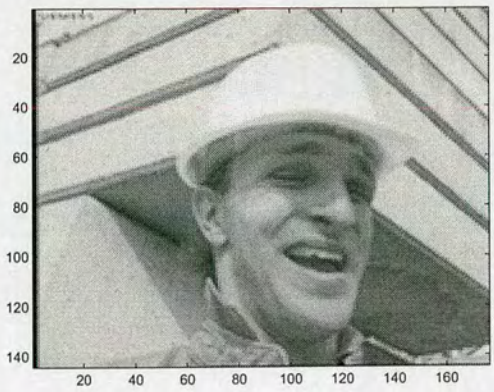
Block size	NTB	% Candidate with $SAD = 0$
16x16	0	0%
	4	0.2%
8x8	0	0%
	4	5%
4x4	0	0%
	4	12%

Table 5.1: *Percentage of matched candidates with $SAD = 0$ during motion estimation using Foreman sequence (search range, $p = \pm 8$).*

Table 5.1 shows the percentage of matched candidates with $SAD = 0$ during motion estimation using five Foreman sequences. For 16x16 block size with $NTB = 4$, the percentage of candidates with $SAD = 0$ is close to the untruncated bit ($NTB = 0$). This shows that for 16x16 block size, the truncated pixel is more likely to have the same matched candidate as in the untruncated pixel. However, for 4x4 block with $NTB = 4$, the percentage of candidates with $SAD = 0$ is 12% compared to 0% for $NTB = 0$. This shows that there are more matched candidates using truncated pixel for 4x4 block size which could lead to incorrect motion vectors.

To illustrate the effect of pixel truncation on variable block size motion estimation, the average peak signal-to-noise ratio (PSNR) is calculated for 50 predicted frames of a Foreman sequence (QCIF@30fps) as shown in Table 5.2. The frames are predicted using full search algorithm at different block sizes and NTB. Frames with three different NTB are shown in Figure 5.3. From Table 5.2, for full pixel resolution ($NTB = 0$), the prediction accuracy improves as the block size decreases. This is reflected by a higher PSNR for prediction using a 4x4 block compared to a 16x16 block.

For $NTB = 4$, a small PSNR drop is observed for a block size of 16x16 (0.08 dB) compared



(a) Number of truncated bits = 0

(b) Number of truncated bits = 4

(c) Number of truncated bits = 6

Figure 5.3: *Pixel truncation for Foreman sequences*

NTB	Block size					
	16x16	diff	8x8	diff	4x4	diff
0	33.11	0	34.89	0	36.82	0
2	33.12	0.01	34.85	-0.03	36.75	-0.07
4	33.03	-0.08	34.35	-0.54	34.66	-2.16
6	31.79	-1.33	30.29	-4.60	27.46	-9.36

Table 5.2: *Average full search PSNR for various NTB using SAD as matching criteria (search range, $p = \pm 8$)*

to untruncated pixels. The PSNR drop for prediction using smaller block sizes is higher with 0.54 dB and 2.16 dB drops for frames with block sizes 8x8 and 4x4, respectively.

As the NTB is increased to 6, the PSNR drop for the smaller blocks increases rapidly. The PSNR drop for the 16x16 block size is 1.33 dB. However, for 8x8 and 4x4 block sizes, the PSNR drop increases to 4.6 dB and 9.36 dB, respectively. This shows that pixel truncation is not suitable for smaller block sizes. In the H.264 standard, substantial improvement in motion prediction is gained by using smaller blocks. Therefore, it is important to improve the PSNR gain especially for smaller block partitions.

5.4 Two-Step Algorithm

In this chapter, a method of pixel truncation for variable block size motion estimation is proposed. This method is based on the following observations:

1. Truncating pixels for larger block sizes can result in better motion prediction compared to smaller block sizes,

2. At higher pixel resolutions, smaller block sizes can result in better prediction compared

// T=8'b1100_0000
// Truncating the search window pixel, Y.
$Y_t = BITAND(Y, T)$
// Truncating the current MB pixel, X.
$X_t = BITAND(X, T)$
// Initialise mv and min_cost
$mv_x = 0, mv_y = 0, cost_{min} = cost_{max}$
// Scanning the search windows and find the best match using block size N =8
For $i_1 = -p_1, p_1$
  For $j_1 = -p_1, p_1$
    $cost = \sum_{m=0_1}^{N-1} \sum_{n=0}^{N-1} [MATCH_1(X_t(m,n), Y_t(i_1+m, j_1+n))]$
    If$(cost < cost_{min})$
      $cost_{min} = cost, mv_x = i_1, mv_y = j_1$
  End of j
End of i
// Refining the search result using full pixel for variable block size
$cost_{min} = cost_{max}$
For $i_2 = -p_2, p_2$
  For $j_2 = -p_2, p_2$
    $cost = \sum_{m=0_1}^{N-1} \sum_{n=0}^{N-1} [MATCH_2(X_t(m,n), Y_t(i_2+m, j_2+n))]$
    If$(cost < cost_{min})$
      $cost_{min} = cost, mv_x = i_2, mv_y = j_2$
  End of j
End of i

Figure 5.4: *Pixel truncation algorithm using two step approach*

to the larger block sizes.

To avoid having large motion vector errors with smaller blocks, motion prediction is implemented in two steps. In the first search, the prediction is performed using pixels with $NTB = 6$ at 8x8 block size. Then, the result of the first search is refined using full pixel resolution (8 bit) in a smaller search area. The algorithm is summarised in Figure 5.4.

Figure 5.5 shows the simulation results using truncated pixels with several matching criteria. Matlab is used to model the full search motion estimation, and the simulation is done over 50 Foreman sequences. Two error-based matching criteria and two Boolean-based matching

criteria are compared against SAD, namely MinMax [102], mean removed MAD (MRMAD) [103], binary XOR (BXOR) [104] and difference pixel count (DPC) [98], respectively. From the figure, at high NTB, error-base matching criteria gives a poor result compared to the Boolean-based matching criteria. This is because, at high number of truncated bit, the pixel accuracy is reduced and leads to inaccurate calculation of the matching cost.

The combination of $NTB = 6$ and DPC gives a good trade-off between PSNR and the computational load. Simulation results using different frame sequences as shown in Figure 5.6, verify this. Boolean-based matching criteria are based on the similarity between pixels' bit positions. In 8 bit pixels, the MSB represents the main feature of the object, while the LSB carries the detailed information of the object. Thus, removing the LSB does not significantly affect the object's features. Thus, Boolean-based matching criteria can find a better match than error-based matching criteria at higher number of truncated bit. However, increasing the NTB up to 7 reduces the MSB information. This results in inadequate information during motion prediction. As a result, the prediction quality degrades significantly at $NTB = 7$ as shown in Figure 5.6.

At highly truncated bits, 16x16 block size is more error-prone since it has more data compared to the smaller block size. However, for complex motion, the motion vector for a smaller block size, especially a 4x4 block, is not necessarily close to that of a 16x16 block. Since the block with smaller size difference tends to move in a similar direction, the 8x8 block is used in the first search. This allows us to get better predictions for either the smaller block (8x4,4x8 and 4x4) or the larger block (16x8, 8x16, 16x16) from the 8x8 motion vector.

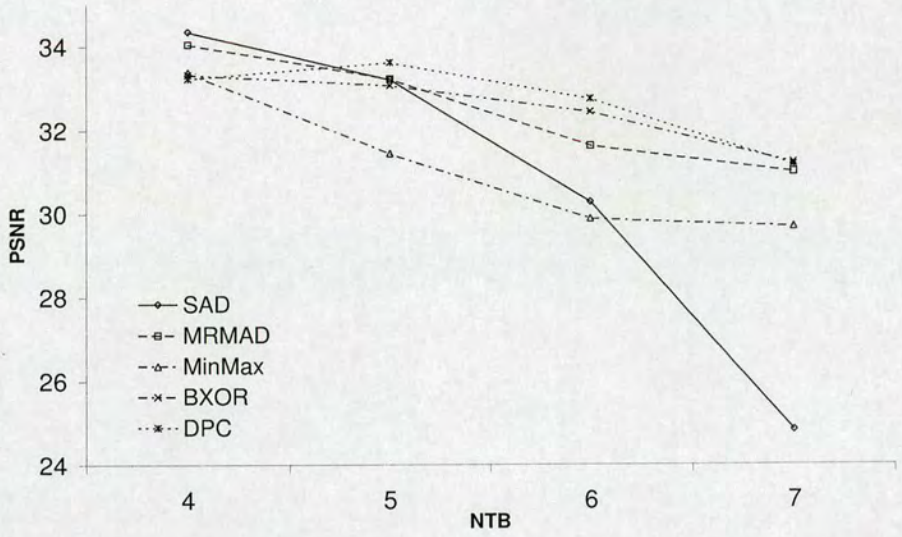


Figure 5.5: *Average PSNR vs. NTB for different block matching criteria using 8x8 block size (50 Foreman frame sequences, QCIF@30fps, search range, p=[-8,7]).*

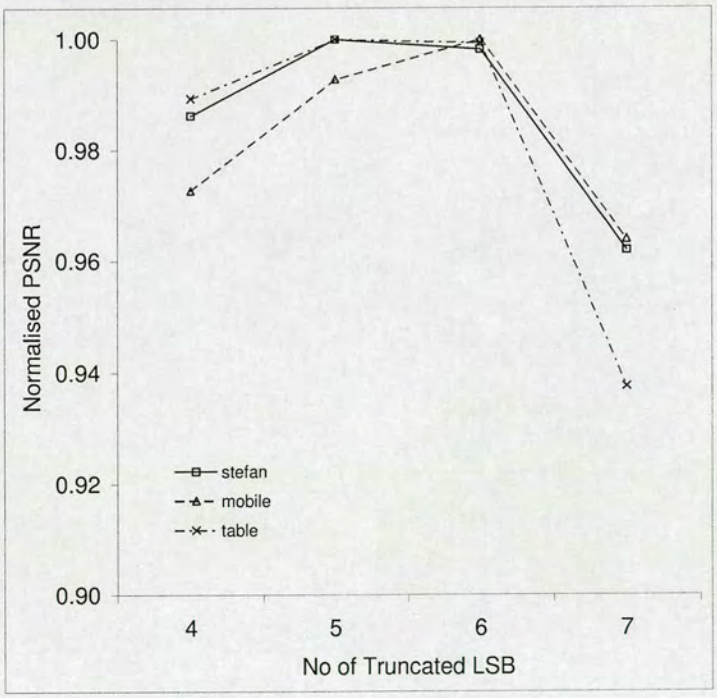


Figure 5.6: *Normalised PSNR vs. NTB using DPC as matching criteria on 8x8 block size (Stefan, Mobile and Table Tennis sequences, 50 frames each , QCIF@30fps, search range, p=[-8,7]).*

Total number of memory access

	Conv. Full Search	Proposed 2-Step
1st search	$(2p)^2 \times q \times 8bit$	$(2p)^2 \times q \times 2bit$
2nd search	-	$(2\frac{p}{2})^2 \times q \times 8bit$
Total	$32p^2q$	$16p^2q$

Total number of matching computation

	Full Search	Proposed 2-Step
1st search	$(2p)^2 \times r \times 1$	$(2p)^2 \times r \times 0.25$
2nd search	-	$(2\frac{p}{2})^2 \times r \times 1$
Total	$4p^2r$	$2p^2r$

Table 5.3: *Total number of memory access and matching computation for conventional full search and the proposed two-step search (p represents the search range; q and r are the numbers of memory access and matching computation per candidate, respectively)*

In the second step, full-pixel resolution is performed to refine the result obtained from the first search. To ensure that the overall computation cost does not exceed the conventional full search computation, the second search is done at a quarter of the size of the first search area. Increasing the refinement area will not only increase the total computation, but also increase the memory access, as shown in Table 5.3.

The second search range, p_2, is defined using the motion vector information from the first search, as follows:

1. Determine the minimum and maximum motion vector for all four 8x8 block partitions (horizontally and vertically) obtained in the first search

2. The centre of the rectangular area which is defined by (1) will be the centre for the second search

3. The second search area with range $p_2 = \frac{1}{2}p_1$ is defined around this centre

Figure 5.7 illustrates the method used to determine the second search area. Blocks A, B, C and D are the 8x8 partitions for a macroblock, as shown in Figure 5.7. After the first search, each partition A, B, C and D has its own motion vector, mv_A, mv_B, mv_C and mv_D respectively. Let mvx and mvy represent their horizontal and vertical motion vector components respectively. Thus, mvx_A and mvy_A represent the horizontal and vertical components, respectively, of the motion vector for block A; mvx_B and mvy_B represent the horizontal and vertical components, respectively, of the motion vector for block B, and so on. The minimum and maximum motion vector of each component is represented by:

$$mvx_{min} = min\{mvx_A, mvx_B, mvx_c, mvx_D\} \tag{5.4}$$

$$mvx_{max} = max\{mvx_A, mvx_B, mvx_c, mvx_D\} \tag{5.5}$$

$$mvy_{min} = min\{mvy_A, mvy_B, mvy_c, mvy_D\} \tag{5.6}$$

$$mvy_{max} = max\{mvy_A, mvy_B, mvy_c, mvy_D\} \tag{5.7}$$

The second search centre is defined as:

$$\left(\frac{mvx_{min+}mvx_{max}}{2}, \frac{mvy_{min+}mvy_{max}}{2}\right) \tag{5.8}$$

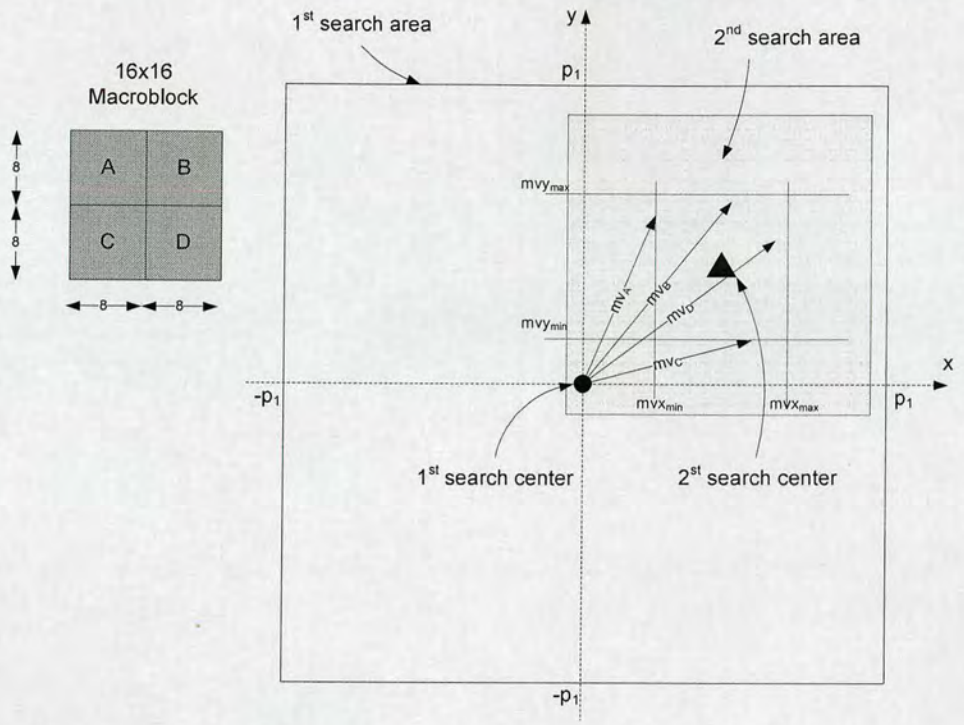


Figure 5.7: *Defining the second search area for the proposed two-step algorithm.*

with search range, $p_2 = \frac{1}{2}p_1$.

5.5 Results and Discussion

Tables 5.4 and 5.5 show the PSNR difference using the proposed method against the conventional full-search ME. The comparison is done for the frames predicted using 16x16, 8x8 and 4x4 partitions. Other block sizes are not included for simplicity. The difference is calculated based on the average PSNR of 85 frames. Different frame sequences that represent various types of motion from low to high are used in this experiment: Akiyo, Mobile, Foreman and Stefan. Both QCIF and CIF frame resolutions are considered, which represent the typical frame size for mobile devices. The search range, p_1=[-8,7] and p_1=[-16,15] is defined for QCIF and CIF, respectively.

	16x16			
	Akiyo	Mobile	Foreman	Stefan
fs_p4	0	0	0.12	0.22
2step16	0	0.01	0.08	0.03
2step8	0	0	0.07	0.03
	8x8			
	Akiyo	Mobile	Foreman	Stefan
fs_p4	0	0.02	0.3	0.41
2step16	0	0.03	0.23	0.13
2step8	0	0.02	0.19	0.11
	4x4			
	Akiyo	Mobile	Foreman	Stefan
fs_p4	0.06	0.16	0.54	0.58
2step16	0.06	0.17	0.47	0.31
2step8	0.05	0.14	0.44	0.27

Table 5.4: *PSNR drop (in dB)against conventional full search SAD QCIF@30fps,* p_1*=[-8,7]*

	16x16		
	Mobile	Foreman	Stefan
fs_p8	0.04	0.14	0.29
2step16	0.04	0.25	0.04
2step8	0.11	0.12	0.04
	8x8		
	Mobile	Foreman	Stefan
fs_p8	0.12	0.24	0.44
2step16	0.12	0.39	0.21
2step8	0.20	0.21	0.09
	4x4		
	Mobile	Foreman	Stefan
fs_p8	0.38	0.44	0.58
2step16	0.37	0.59	0.44
2step8	0.40	0.39	0.32

Table 5.5: *PSNR drop (in dB) against conventional full search using SAD for CIF@30fps,* p_1*=[-16,15]*

Total number of memory access

	fs_p4 or fs_p8	2step16
1st search	$\left(2\frac{p}{2}\right)^2 \times q \times 8bit$	$(2p)^2 \times q \times 2bit$
2nd search	-	$(2\frac{p}{2})^2 \times q \times 8bit$
Total	$8p^2q$	$16p^2q$

Total number of matching computation

	fs_p4 or fs_p8	2step16
1st search	$\left(2\frac{p}{2}\right)^2 \times r \times 1$	$(2p)^2 \times r \times 0.25$
2nd search	-	$(2\frac{p}{2})^2 \times r \times 1$
Total	p^2r	$2p^2r$

Table 5.6: *Total number of memory access and matching computation for fs_p4, fs_p8, and 2step16 (p represents the search range; q and r are the numbers of memory access and matching computation per candidate, respectively)*

2step8 represents the proposed 2-step search using the 8x8 block partition. For comparison, the results for the 2-step search are included where the first search is done using 16x16 partitions (2step16). The result of the first search is used as the centre for the second search. fs_p4 and fs_p8 represent the conventional full-search ME with a search range equivalent to $\frac{1}{2}p1$ for QCIF and CIF, respectively. The total number of memory access and computation for 2step16, fs_p4, and fs_p8 are shown in Figure 5.6.

From Tables 5.4 and 5.5, the proposed method is able to achieve a good prediction with a smaller PSNR drop compared to the other method. For a low-motion sequence such as Akiyo, the PSNR drop for QCIF is below 0.05dB. The PSNR drop increases slightly for a high-motion sequence such as Stefan. This is due to the prediction error and search range limitation during the first and second searches, respectively.

The smaller PSNR drop for 2Step8 compared to 2Step16 shows that the first search using 8x8 partition gives a good approximation compared to 16x16 block size. In the 8x8 partitions,

more information for the macroblock motion is obtained, and this additional information is important when determining the second search range for the high-motion sequence.

In addition, the PSNR drop varies depending on the level of detail of the frame. For a frame sequence with high detail, such as Mobile, the first search with $NTB = 6$ contains more information for the macroblock feature. Thus, it can obtain a better match, which is reflected by the lower PSNR drop. For a frame with less object detail, the PSNR drop is slightly higher. The same explanation is applicable for the higher PSNR drop for CIF compared to QCIF frame resolution. This is because the macroblock content for the CIF frame is more sparse compared to the QCIF macroblock.

The reduction in the search range in the refinement stage does affect the prediction for the 4x4 block partition. This is expected, since in 2step8, the full pixel resolution is done at $\frac{1}{4}$ of the conventional full-search area. Thus, the reduction in computation comes at the expense of decreasing the prediction accuracy. However, compared to the direct reduction of the search range, the proposed method gives a higher PSNR as opposed to the fs_p4 and fs_p8 for the QCIF and CIF frames, respectively.

In all frame sequences tested, the proposed method gives good PSNR compared to the conventional full search with PSNR drop $< 0.5dB$. Furthermore, the proposed method can yield a better uniform motion vector for the 4x4 partitions which is required to reduce the overall bitrates.

Table 5.7 shows the average PSNR drop for several existing motion estimation techniques discussed in Section 5.2. In 2BT, frames are converted from 8bit to 2bit using the threshold value derived from the local image standard deviation. The motion estimation is performed

	Conventional Full Search [44]	2BT [97]	LRQME [98]	Proposed 2-Step Algo.
Low resolution	No	Yes	Yes	Yes
High resolution	Yes	No	Yes	Yes
Block Size: 16x16	0.0	0.7	0.8	0.1
Block Size: 8x8	0.0	1.3	-	0.2
Block Size: 4x4	0.0	3.0	-	0.4

Table 5.7: *Average PSNR drop (dB) for several motion estimation techniques.*

based on the transformed two-bit image. LRQME transforms the eight-bit pixel to two bits using an adaptive quantiser. The result of the motion estimation obtained using the low-resolution image is refined at higher pixel resolution using 16x16 block size. As shown in Table 5.7, the proposed method shows superior performance compared to other techniques for 16x16 to 4x4 block sizes.

To evaluate the effectiveness of the proposed method within H.264 software toward generating the final bit rates, the existing H.264 reference software (version JM8.6 for baseline profile) is modified to include the proposed algorithm. The existing full search motion estimation algorithm is replaced with the two-step method. The simulations were done with rate control off, QP = {20, 25, 30, 35, 40}, 85 frames, QCIF and CIF frame format at 30fps. Both the conventional and the proposed method were compared.

Figure 5.8 and 5.9 show the PSNR versus bitrates graphs simulated using the modified H.264 reference software. Two video sequences (Foreman and Stefan), which represent medium and high motion sequence, are shown. The graphs show that the proposed method can achieve good performance, with PSNR drop less than 0.5 dB.

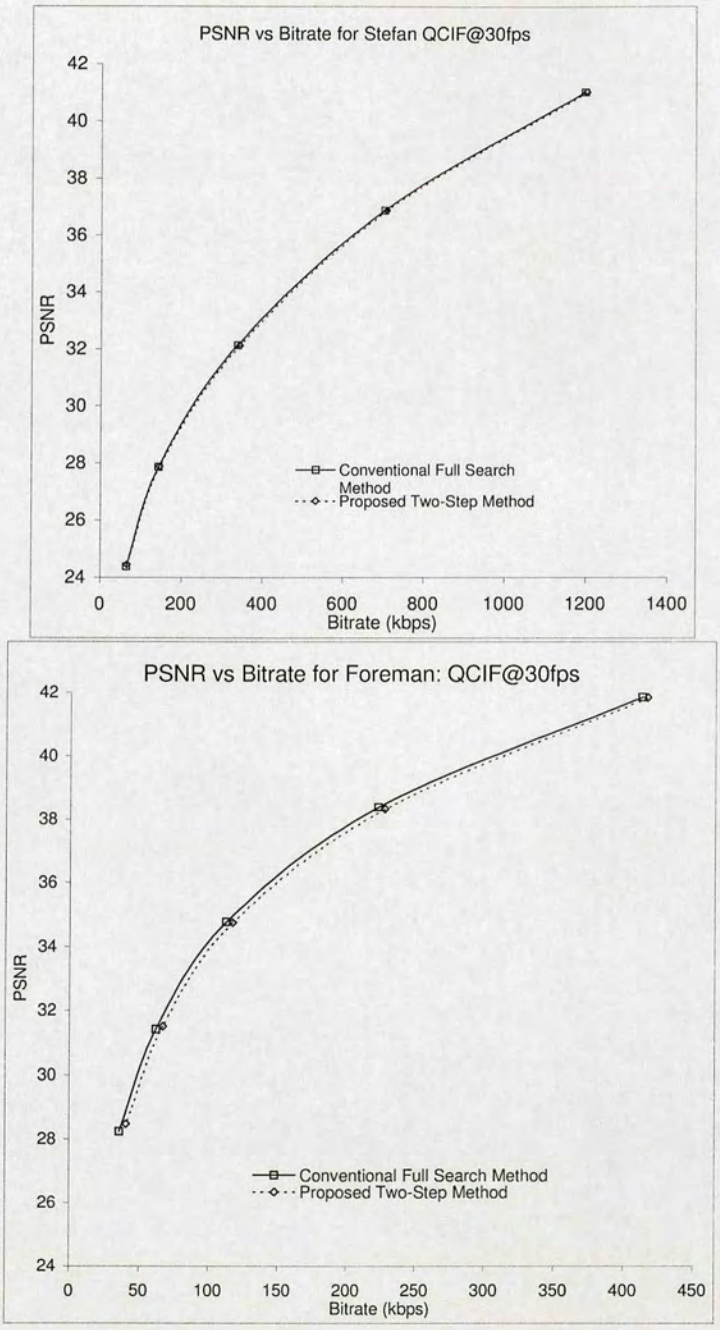


Figure 5.8: *PSNR vs. Bitrate using the proposed 2-Step method for QCIF@30fps.*

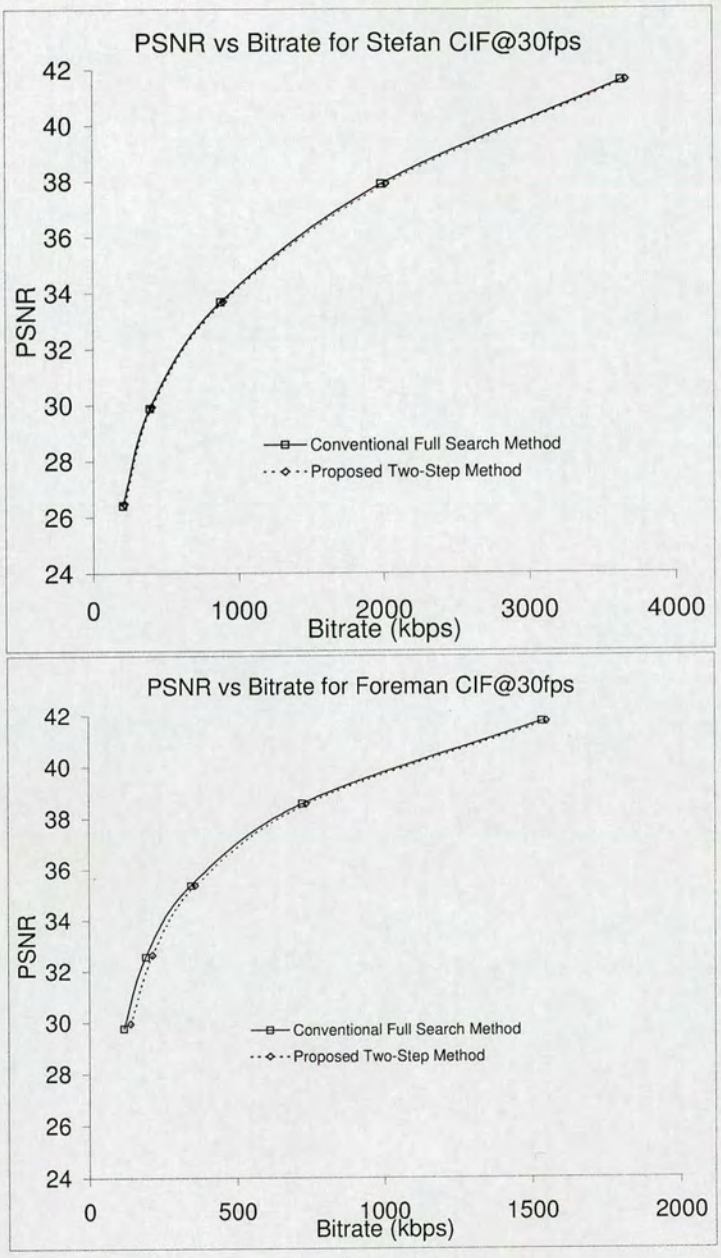


Figure 5.9: *PSNR vs. Bitrate using the proposed 2-Step method for CIF@30fps.*

5.6 Summary

This chapter has presented a method to reduce the computational cost and memory access for variable-block-size motion estimation using pixel truncation. Previous work has shown that pixel truncation provides an acceptable performance for motion prediction using a 16x16 block size. However, for motion prediction using smaller block sizes, pixel truncation reduces the motion prediction accuracy. In this chapter, a two-step search to improve the frame prediction using pixel truncation was proposed. The proposed method reduces the total computation and memory access by 50% compared with the conventional full search method with PSNR drop less than 0.5 dB.

Chapter 6
Energy Efficient Motion Estimation Architecture for H.264 VBSME using The Two-Step Algorithm

6.1 Introduction

Motion estimation plays an important role in video coding. As discussed in Section 3.5, ME takes a significant amount of computation in H.264 hardware (>75%). Thus, minimising the power in ME is important. In this standard, the variable block size and multiple reference frames are introduced, making the hardware more complex than before. While the conventional hardware supports a fixed block size, the hardware needs to be modified to support variable block size with multiple reference frames.

In real time implementation, the motion estimation computational load varies depending on the picture type. To optimise the power, the ME computational load should be matched with the computational demand. In addition, while minimising the average power is required, the actual total energy consumed will determine the actual efficiency of the hardware. This factor is important for battery operated devices and other consumer electronics.

This chapter presents a new design methodology for motion estimation using pixel truncation. Three energy saving architectures are also proposed based on this methodology for the implementation of the H.264 standard. Energy saving is achieved utilising the two-step algorithm proposed in Chapter 5.

The rest of the chapter is organised as follows. Section 6.2 reviews the existing motion estimation architectures. Section 6.3 analyses the proposed architecture. The results are discussed in Section 6.4. Finally, Section 6.5 concludes the paper.

6.2 Motion Estimation Implementation

Motion estimation can be realised in several ways, such as by using a general purpose processor, DSP processor or dedicated application specific integrated circuit (ASIC). The general purpose processor offers the highest flexibility since it is programmable to different ME algorithms. The DSP processor produces better performance compared to the general purpose processor, but has reduced flexibility because it requires a specific processor to execute the instruction. With dedicated ASIC, the highest performance is gained since the hardware is fully optimised for the specific ME algorithm. It can achieve the lowest power consumption and area compared to the other implementations.

In software based implementation, the instructions are operated in sequence. The ME complexity is determined by the number of required operations. Thus, many fast algorithms have been introduced to reduce the number of operations by minimising the total number of evaluated candidates such as in 3SS, 2D-Log and DS, as reviewed in Section 4.2. In this method, the search is done in a certain search pattern and several iterations are required before the final result is obtained. This method is widely adopted in software based implementation because it does not have any restrictions in terms of data access and control.

However, this approach is not efficient for dedicated hardware implementation since it causes

non-regular data access and requires complex control. Furthermore, hardware resources are limited and the operation is restricted by the allocated number of clock cycles. For high hardware utilisation, the data must be accessed before the operation is started. Thus, it is best to reuse the already accessed data [105][106].

With the H.264 standard, the multiple reference frame and variable block size increases the computational load. Therefore, dedicated hardware is commonly used to accelerate the ME computation. In contrast to the general purpose processor approach, dedicated hardware faces different challenges. The number of operations alone is not enough to use as a comparison for hardware efficiency. In general, the ME hardware design criteria should also consider the following factors [44]: silicon area, throughput, memory access, memory bitwidth, and latency.

6.2.1 General Motion Estimation Architecture

A motion estimation architecture consists of control units, address generation, pixel buffers and processing elements (PE). Figure 6.1 shows a typical ME architecture. Typically, the current MB and search area pixels are stored in buffers (SRAM). During motion prediction, the pixel is read from the current frame memory and the reference frame memory into the current MB and search area buffers respectively. During the matching process, pixels from the current MB and search area buffers are read and loaded into PE. The address generation unit calculates the address required to access data from the current MB and search area memories. This process continues until all candidates have been evaluated.

Full search ME can be implemented using a systolic array architecture. This architecture

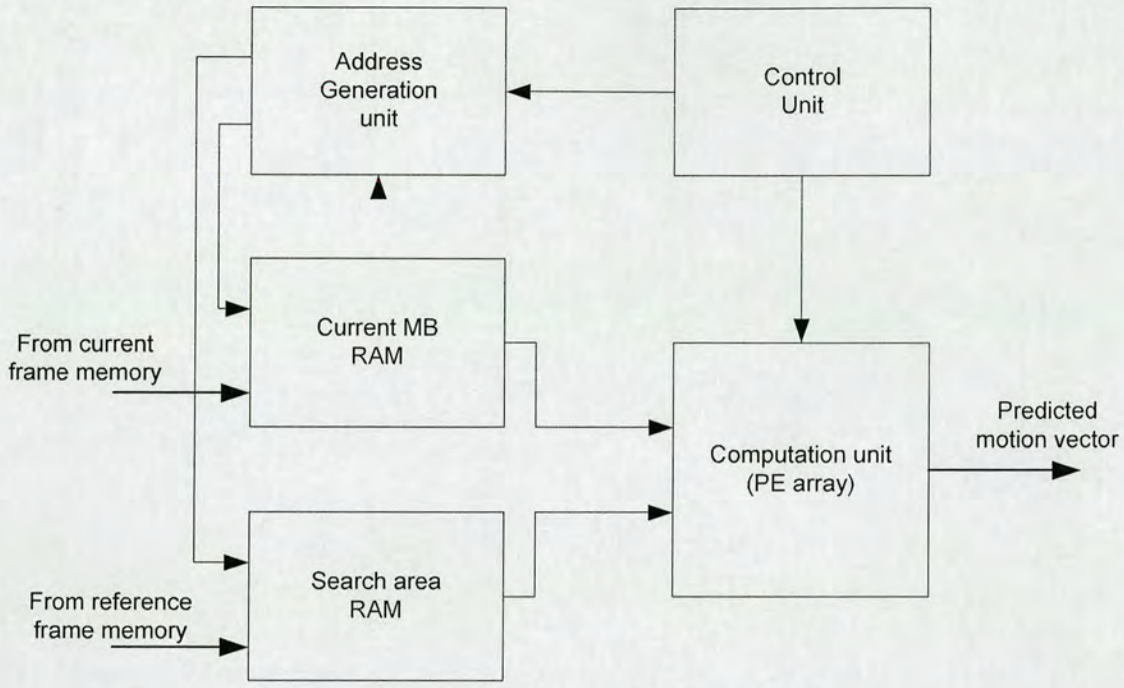


Figure 6.1: *General motion estimation architecture*

consists of a network of processing elements that compute and pass the data throughout the system [107]. The PE are an important part of ME, as it determines the overall performance of the architecture. Each PE calculates the matching cost of two pixels. The results per candidate will be accumulated before the final decision is made. In most designs, the number of PE is determined by either macroblock size or search range.

Since the PEs make up the largest block in the ME, many architectures have been introduced to reduce the area. There are two types of popular full search ME: one-dimensional (1D) and two-dimensional (2D). The fundamental architectures of 1D and 2D ME are discussed next, since most other architectures are derived from these two approaches.

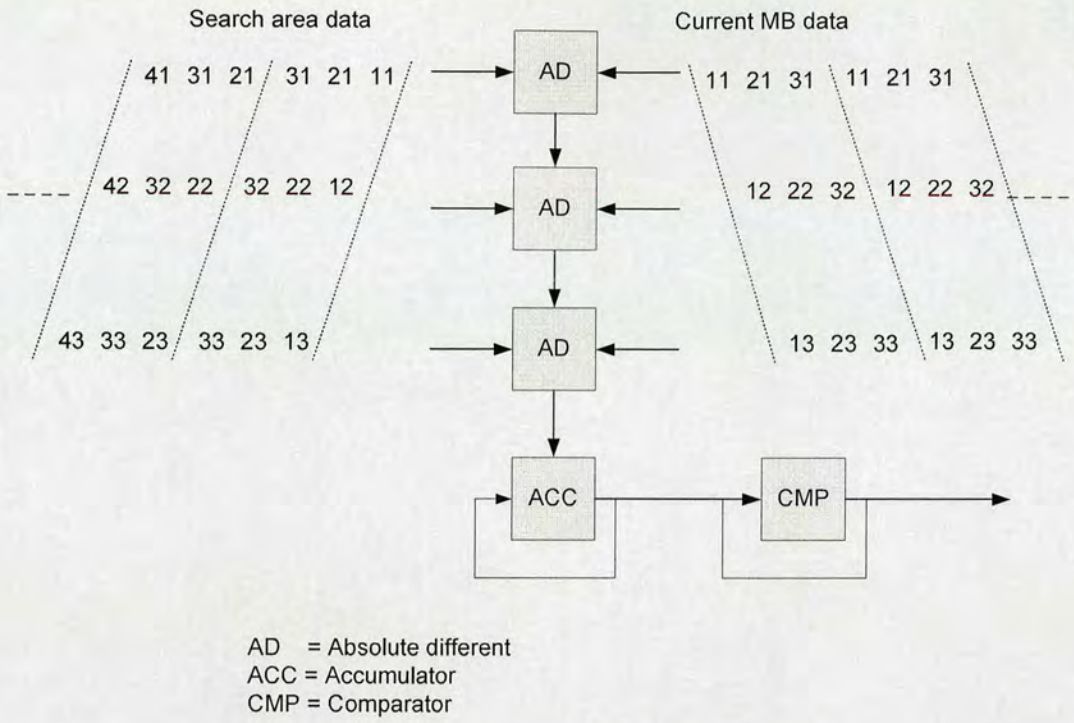


Figure 6.2: *One-dimensional motion estimation.*

6.2.2 One-dimensional Motion Estimation (1D ME)

In this architecture, N PEs are used to calculate the absolute difference, where N is the macroblock width. The current MB pixels and the search area pixels are input as shown in Figure 6.2 (for $N = 3$,$p = 2$) [108]. The absolute difference calculated in each PE is passed on and added to the absolute difference in the subsequent PE. The accumulator calculates the sum of absolute difference (SAD) until all pixels in the candidate block have been evaluated. Once the SAD calculation for each candidate macroblock is complete, the accumulator passes the value to the comparator. The comparator stores the minimum calculated SAD during the motion prediction. The value will be updated if the calculated candidate block results in smaller matching cost compared to the existing value in the comparator.

In this architecture, each PE compares a pair of pixels for one candidate at a time. The

number of PE is determined by the MB width, and each candidate requires an average of N clock cycles. The total clock cycle required to evaluate the whole candidate is $(2p)^2 \times N + N$, including the initialisation cycle. For $N = 16$ and $p = 8$, 4112 clock cycles are required to evaluate all candidates.

This 1D architecture has the advantage of a smaller silicon area as compared to 2D architecture. The total PE is independent of SA size. However, since the PE does not store any current or search area pixels, these pixels need to be accessed continuously, which results in high memory bandwidth.

6.2.3 Two-dimensional Motion Estimation (2D ME)

The 2D ME architecture has the advantage of high data reuse and lower memory bandwidth. In this architecture, the PE is arranged in $N \times N$, as shown in Figure 6.3 (for $N = 3$,$p = 2$) [108]. In contrast to 1D ME, the 2D ME stores the current MB pixel in the PE. The search area pixels are input N parallel and propagated to PEs horizontally. At each clock cycle, the absolute difference is calculated and accumulated before it is passed to the next PE. The accumulators calculate the partial SAD of the respective PE column. The calculated partial SAD will be propagated and added to the partial SAD value in the next accumulator.

N clock cycles are required to initialise the hardware. The next $(2p + N)$ clock cycles output the SAD for each candidate row. Thus, this architecture requires $2p \times (2p + N) + N$ clock cycles to evaluate all candidates [95]. For $N = 16$ and $p = 8$, this is equivalent to 528 clock cycles.

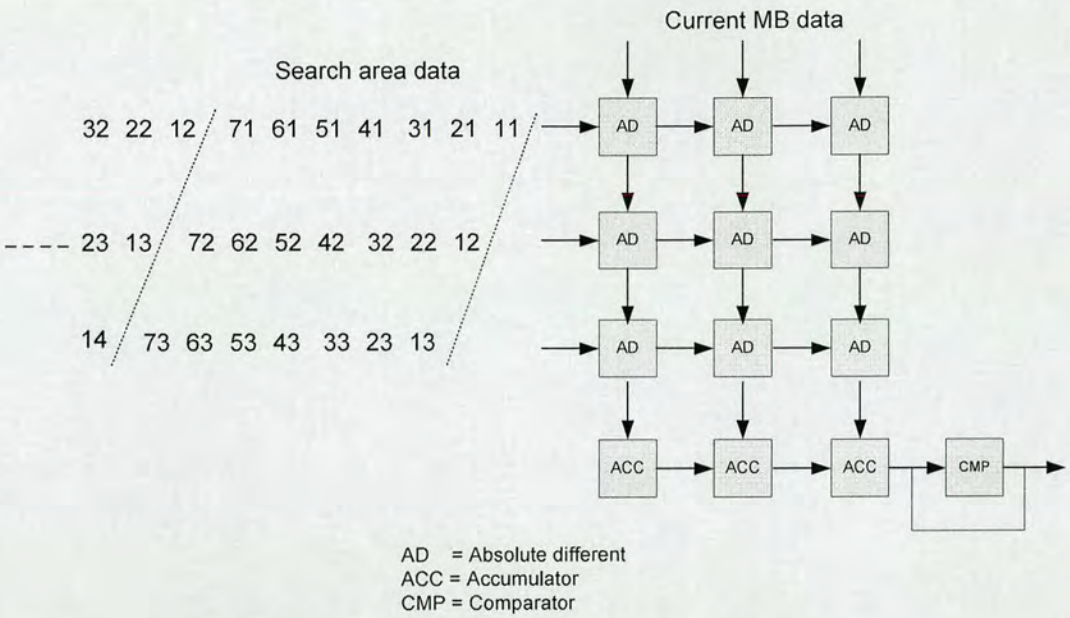


Figure 6.3: *Two-dimensional motion estimation.*

In contrast to 1D where each pixel is used once before it is discarded, high data reuse is achieved in 2D architecture because the accessed pixel is used to calculate multiple candidates. This architecture is optimum for search patterns where the successive candidates are next to each other, such as in full search.

6.3 Methodology and Hardware Design

In this section, the conventional ME architecture used in this thesis is first reviewed. Next, the architectures needed to support the two-step method as proposed in Section 5.4 are discussed. The area and power overhead for the computation and memory unit are also investigated. Based on these analyses, three low power ME architectures with different area and power efficiencies are proposed.

The ME architecture based on 2D ME as discussed in [95] is implemented. 2D ME is chosen

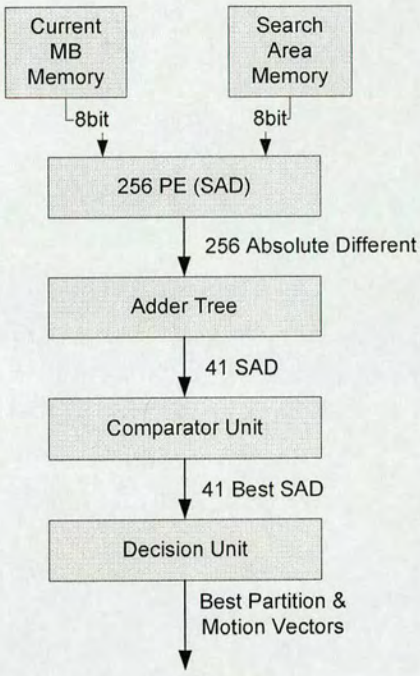
because it can cope with the high computational needs of the real-time requirements for H.264 by using a lower clock frequency than 1D architecture.The search range of $p = [-8, 7]$ is used throughout this chapter.

6.3.1 Computation Unit

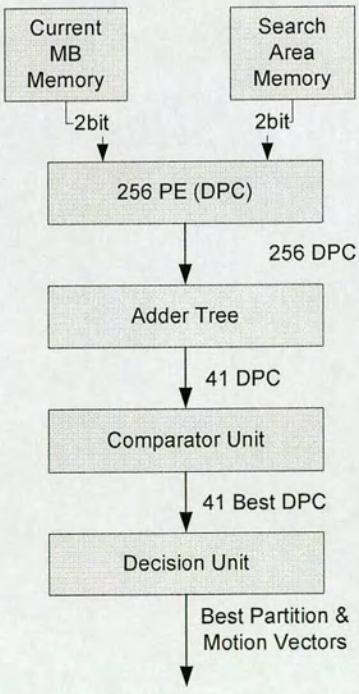
Figure 6.4 (a) shows the functional units in the conventional 2D ME (me_sad) [95]. The ME consists of search area (SA) memory, a processing array which contains 256 processing elements (PEs), an adder tree, a comparator and a decision unit. The search area memory consists of sixteen memory banks where each bank stores 8-bit pixels in a $H \times \frac{W}{N}$ total word, where H and W are the search area window's height and width respectively, and N is the macroblock's (MB) width. During motion prediction, 16 pixels are read from the 16 memory banks simultaneously. The data in the memory are stored in a ladder-like manner to avoid delay during the scanning [106].

At each initial search, the current and the first candidate MB are loaded into the processing array's registers. Then, the matching cost is calculated for one candidate per clock cycle. The 256 absolute-different from the PEs are summed by the adder tree, and outputs the sum of absolute-different (SAD) for 41 block partitions. The adder tree reuses the SAD for 4x4 blocks to calculate a larger block partition. In total, the adder tree calculates 41 partitions per clock cycle.

Throughout the scanning process, the comparator updates the minimum SAD and the respective candidate location for each 41-block partition. Once the scanning is complete, the decision unit outputs the best MB partition and its motion vectors. The ME requires 256



(a)



(b)

Figure 6.4: *Block diagrams for (a) me_sad (b) me_dpc*

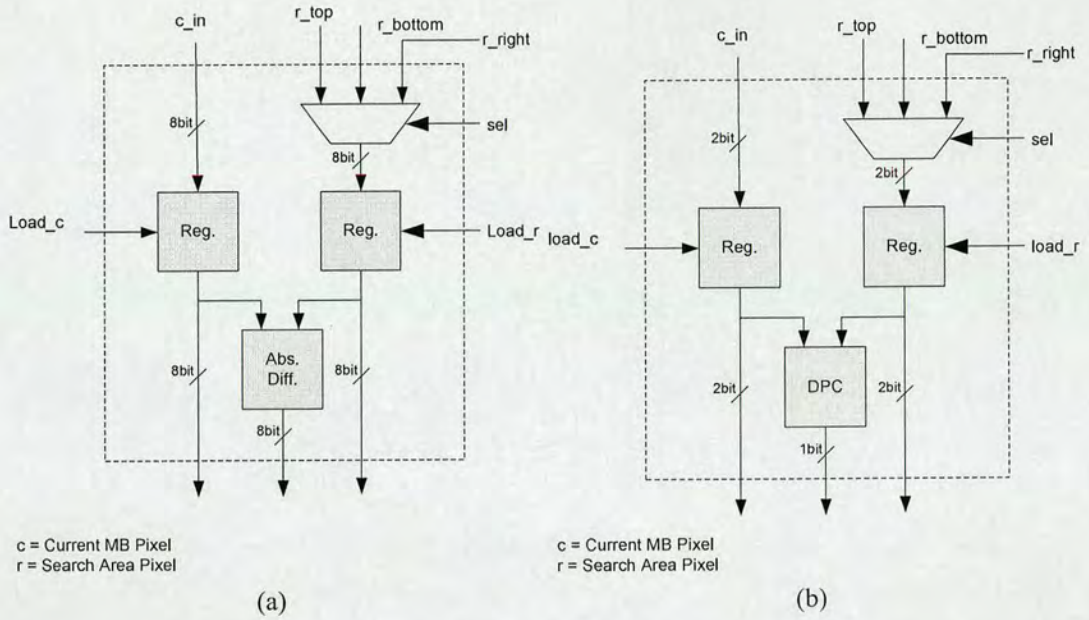


Figure 6.5: *Processing element (PE) to support (a) SAD and (b) DPC*

clock cycles to scan all candidates.

For me_sad, the input and output for each of the PE are 8 bits wide as shown in Figure 6.5 (a). The input for the adder tree is 8 bits wide, and the SAD output is 12 to 16 bits wide, depending on the partition size. These data are then input into the comparator, together with the current search location information.

Using a similar architecture to me_sad (Figure 6.4 (a)), DPC based ME (me_dpc) requires 2 bits for the current and reference pixel inputs as shown in Figure 6.4 (b). Furthermore, the matching cost is calculated using Boolean logic (XOR and OR) rather than arithmetic operations as in SAD based PE. These make the overall area for the 256 PEs in me_dpc much smaller than me_sad. The reduction in output bitwidth in DPC based PE also reduces the bitwidth required for adder tree and comparator unit. The input and output for the adder tree is 1 bit and 5 to 9 bitwidths, respectively. A similar bitwidth is applied to the comparator's

Modules	me_sad		me_dpc	
	Area	Power	Area	Power
256 PE	0.90	28.67	0.32	2.31
Adder_tree	0.13	5.53	0.04	0.99
Comparator Unit	0.11	1.25	0.09	0.86
Decision Unit	0.10	0.54	0.07	0.50
Total	1.24	36.00	0.52	4.66

Table 6.1: *me_sad and me_dpc area (mm^2) and power (mW)*

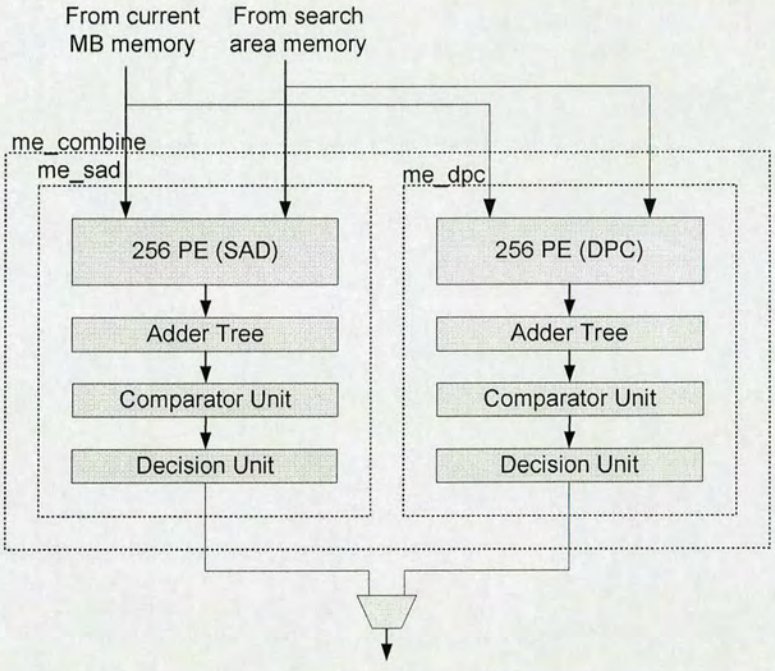
input.

Table 6.1 compares the area mm^2 and power consumption mW for me_sad and me_dpc computational units. The comparisons are based on synthesis results using 0.13μm CMOS UMC technology. The table shows that me_sad's area is dominated by the 256 PE (73%). Thus, with the significantly smaller area for 256 PE, the me_dpc will require less area than the me_sad. The overall me_dpc requires only 42% of the me_sad area.

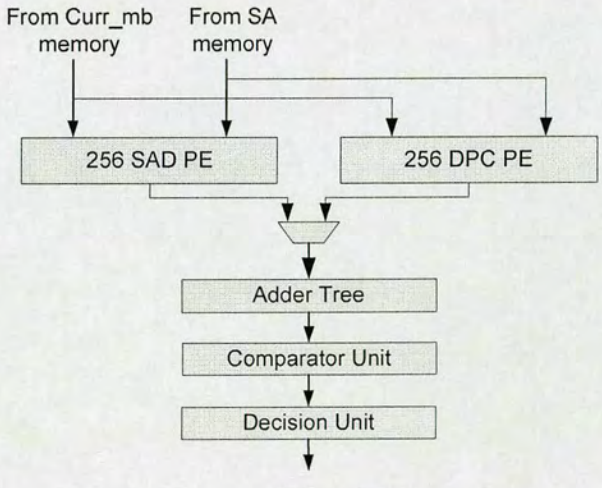
Based on the above analysis, two types of architectures for the ME computation unit that can perform both low resolution and full resolution searches are proposed. These are me_split and me_combine as shown in Figure 6.6.

The me_split implements both me_sad and me_dpc as two separate modules, as shown in Figure 6.6 (a). During low resolution search, me_sad is switched off while the me_dpc is used to perform the low resolution search. The second step uses the me_sad, while the me_dpc is switched off. This architecture allows only the necessary bit size to be used during different search modes. While potential power savings is possible, this architecture requires additional area for the adder tree, comparator and decision unit to support the low resolution search.

Due to the functions of the adder tree, the comparator and the decision units are similar for



(a)



(b)

Figure 6.6: *Computational unit: (a) me_split (b) me_combine*

both me_sad and me_dpc, me_combine shares these units during the low resolution search and full pixel resolution (Figure 6.6 (b)). This architecture results in a much smaller area compared to me_split. However, higher power consumption is expected during the low resolution search because the adder tree, comparator and decision unit operate at higher bit size than needed.

6.3.2 Memory Architecture

Conventional ME architecture implements the SA memory using single port static random access memory (SRAM) with a pixel (8 bits) per word. To implement the two-step search, the first two MSBs for each pixel during the first search and 8 bits in the second stage need to be accessed. Thus, the pixels need to be stored to allow two reading modes. For this, three types of memory architecture are proposed. These are (a) 8bit memory (mem8), (b) 2bit and 8bit memory (mem28), and (c) 8bit memory with prearranged data and transposed register (mem8pre) as shown in Fig 6.7.

The mem8 approach stores the data in the same way as in the conventional ME. Eight-bit data is accessed during both the low resolution and the refinement stages. However, during the low resolution search, the lower 6 bits are not used by the PE. Because the memory is accessed during both low resolution and the refinement stage, it results in higher memory bandwidth than the conventional ME architecture.

To overcome the problem in mem8, mem28 uses two types of memory: 2-bit and 8-bit. The 2-bit memory stores the first two MSBs of each pixel, and the 8-bit memory stores the complete full pixel bitwidth. During the low resolution search, the data from the 2-bit memory are

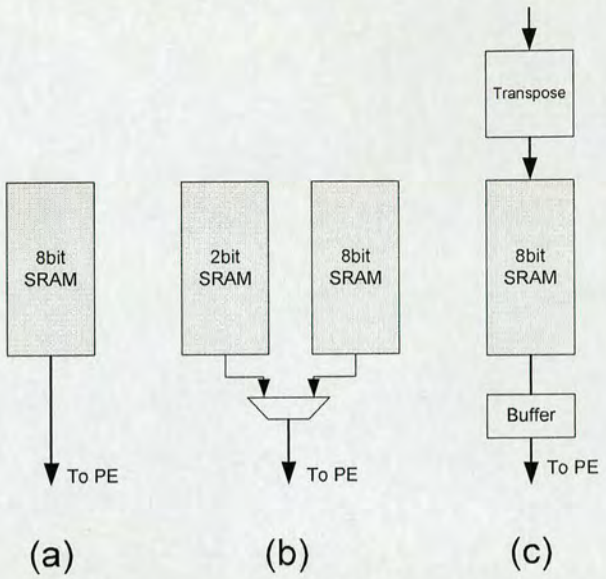


Figure 6.7: *SA memory arrangement (a) mem8 (b) mem28, (c) mem8pre*

accessed. This allows only the required bits to be accessed without wasting any power during low resolution. In the refinement stage, the 8-bit memory is read into the PEs. Although this architecture can potentially reduce memory bandwidth and power consumption, it needs an additional area for the 2-bit memory.

In mem8pre, the data is prearranged before storing them in 8-bit memory. Four pixels are grouped together, and then transposed according to their bit position, as shown in Figure 6.8. During the low resolution search, only the memory locations that store the first two MSBs of the original pixels is read. Thus, the total memory accessed during the low resolution is $\frac{1}{4}$ of the conventional full pixel access.

In full resolution search,four memory locations that contain the first up to eighth bits are read in four clock cycles. Delay buffers, as shown in Figure 6.7 (c), realigns these words to match the original 8-bit pixel. By prearranging the pixels this way, the same memory size as in the conventional full search can be used while retaining the ability to access the first two MSBs,

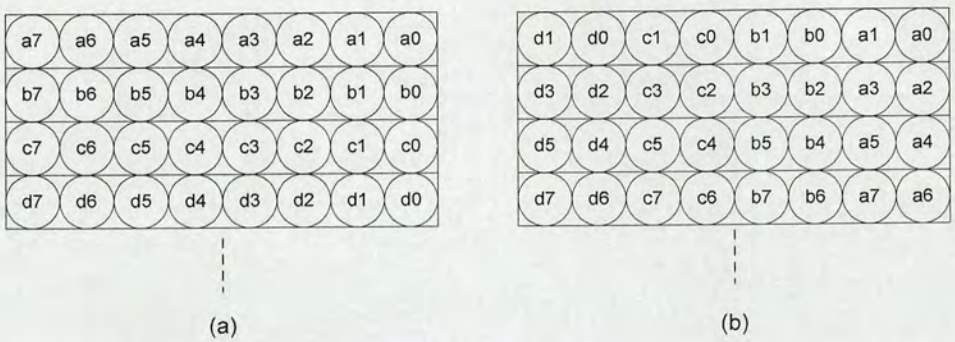


Figure 6.8: *Storing 8bit pixel in 8bit memory (a) conventional arrangement (b) mem8pre*

	Low Resolution	High Resolution
mem8	$NWH \times 8bit$	$N\frac{WH}{4} \times 8bit$
mem28	$NWH \times 2bit$	$N\frac{WH}{4} \times 8bit$
mem8pre	$NWH \times 2bit$	$N\frac{WH}{4} \times 8bit$

Table 6.2: *Memory bandwidth for different architectures*

as well as the full bit resolution. The drawback of this approach is it needs additional circuitry to transpose and re-align the pixels during the motion prediction.

The estimated bandwidth for the above three memory architectures are shown in Table 6.2. In summary, while all three architectures use the same memory bandwidth during high resolution search, , mem28 and mem8pre give $\frac{1}{4}$ memory bandwidth during low resolution search compared with mem8. However, the lower memory bandwidth comes at the cost of additional area overhead. The implementation of these memory configurations will be discussed in Section 6.4.

6.3.3 Overall Architecture

Based on the discussions in Section 6.3.1 and 6.3.2, three different ME architectures that can perform both low resolution and full resolution searches are proposed. By combining

different computation and memory units as shown in Figure 6.6 and 6.7, respectively, the following architectures are proposed:

1. me_split+mem28 (ms_m8)
2. me_combine+mem8 (mc_m8)
3. me_combine+mem8pre (mc_m8p)

In these architectures, both low resolution and full resolution search can be performed. With proper configuration, the conventional full search algorithm can be used during normal conditions to ensure a high quality picture at the output. In condition where energy consumption is the main concern, the two-step method is used. This allows us to reduce the energy consumption without significantly degrading the output picture quality.

6.4 Results

This section presents the synthesised results of the proposed architectures. First, the results for the computation unit are presented. Next, the area and power consumption for the proposed memory architectures are analysed. Finally, the results for the overall ME architectures that can provide efficient area and power consumption are presented.

The design was synthesised using the UMC 0.13μm CMOS library. Verilog-XL and Power Compiler were used to perform functional simulation and power analysis, respectively. Actual video data is used to verify the hardware and to obtain the estimated power consumption.

Modules	me_split			me_combine		
	Area	Power		Area	Power	
		Low Res.	High Res.		Low Res.	High Res.
256 PE	1.17	3.62	24.44	1.06	3.222	24.39
Adder_tree	0.18	1.54	6.63	0.13	1.871	6.48
Comparator Unit	0.20	0.80	1.26	0.11	1.034	1.25
Decision Unit	0.16	0.50	0.54	0.10	0.543	0.54
Others	0.04	0.01	1.29	0.07	0.17	1.36
Total	1.75	6.46	34.16	1.47	6.84	34.02

Table 6.3: *Computational unit: Area (mm^2) and power (mW) comparison*

Table 6.3 compares the synthesis results for the proposed computation units: me_split and me_combine. The power consumption during low resolution and full resolution searches are shown in detail. During the motion prediction, the search range $p_1 = [-8, 7]$ and $p_2 = [-4, 3]$ is used during low resolution and full pixel resolution search, respectively.

From the table, the me_split consumes 6% less power during the low resolution search than the me_combine. However, this comes at the cost of an additional 41% of area on top of the existing me_sad. On the other hand, because me_combine shares some modules, it requires only 19% additional area compared to me_sad. This shows that, combining the adder tree, comparator and decision unit as in me_combine is preferred over me_split.

Table 6.4 shows the area and power comparison for the proposed memory unit. The reported power includes the power consumed by the additional circuit needed by respective memory architectures. The SRAM model is generated using a UMC 0.13 memory compiler with estimated area and power provided by the datasheet.

From Table 6.4, it is clear that the mem8pre provided the lowest bandwidth and power compared to the other memory configurations during the low resolution search. This is expected

Modules	Area	Power	
		Low Res.	High Res.
mem8	0.23	4.7	4.7
mem28	0.39	3.3	4.7
mem8pre	0.42	2.2	7.5

Table 6.4: *Memory unit: area (mm^2) and power (mW) comparison*

since only $\frac{1}{4}$of the memory is accessed during the low resolution search. However, it requires extra area for the additional circuits needed to arrange the pixels.

On the other hand, mem8 gives the minimum area compared to the other configurations. Because the full pixel bitwidth is accessed in both low and full resolution, it requires higher memory bandwidth and uses more power than the others during the low resolution search.

Table 6.5 shows the total area and power consumption results based on extracted layout for the proposed overall ME architectures: ms_m28, mc_m8, mc_m8p. In order to make a fair comparison between these architectures, the total energy needed during each motion prediction is calculated. For the two-step method, this includes energy consumed during both low and full resolution search. The energy is calculated as the power consumption multiplied by the total time taken to complete the motion prediction. The normalised energy and area is shown in Table 6.6. From the table, the architecture mc_m8p consumes the least energy compared to the others, and saves 53% compared to conventional ME (me_sad). However, because it requires additional area for the DPC based PEs and buffer to arrange the pixels, 28% area overhead is required to implement this method.

Compared to other architectures, mc_m8 gives the best trade-off in terms of both area and energy efficiency. By using the two-step method, this architecture can save 48% of the energy compared to conventional ME with 16% additional area required to implement a low

Modules	Area	Power					
		Low Res.			Full Res.		
		Logic	Mem	Total	Logic	Mem	Total
me_sad	1.77	NA	NA	NA	45.00	4.7	49.70
ms_m28	2.56	8.32	3.3	11.62	44.02	4.7	48.72
mc_m8	2.05	8.79	4.7	13.49	43.72	4.7	48.42
mc_m8p	2.26	9.80	1.2	10.98	45.74	4.7	50.44

Table 6.5: *ME architecture area (mm^2) and power (mW) comparison*

	Area	Energy	Area×Energy
me_sad	1.00	1.00	1.00
ms_m28	1.45	0.48	0.70
mc_m8	1.16	0.52	0.60
mc_m8p	1.28	0.47	0.61

Table 6.6: *Normalised ME architecture area (mm^2) and power (mW) comparison*

resolution search.

6.5 Summary

This chapter has presented a methodology for motion estimation based on the two-step algorithm introduced in Chapter 5 for energy efficient design. Based on this methodology, three architectures were presented for variable block size motion estimation for H.264. These architectures can perform both low resolution search and full resolution search. The results show that the proposed architectures are able to save up to 53% energy compared to the conventional full search architecture. This makes such architectures attractive for H.264 application in future mobile devices.

Chapter 7

Interframe Bus Encoding Technique for Low Power Video Compression

7.1 Introduction

For MPEG-4 video compression, raw video data (pixels) dominate data transfer [109]. Fig 7.1 shows a typical portable multimedia system. In this system, the digital camera captures a sequence of pictures and stores the data into the memory unit or displays it on the display unit. Video and audio codec perform the compression on the video and audio data respectively. During this process, the data from the memory is accessed repetitively. The compressed bitstream is then transmitted or stored into a local storage unit. The central processing unit (CPU) manages the operation for the whole system.

During the compression of five minutes of video (CIF@15 fps), at least 4500 frames are transferred from the memory to the video compressor. For frame format YUV420, this is equivalent to the transfer of 684 million pixels. These values increase for higher frame rates and frame resolutions.

This high data transfer translates into high power dissipation on the memory-processor busses. This is severe for systems with off-chip memory where the bus load is several orders of magnitude higher than the on-chip bus with a typical value of around 15pF on each bus wire [110]. It has been reported that the off-chip bus consumes 10%-80% of overall power [111].

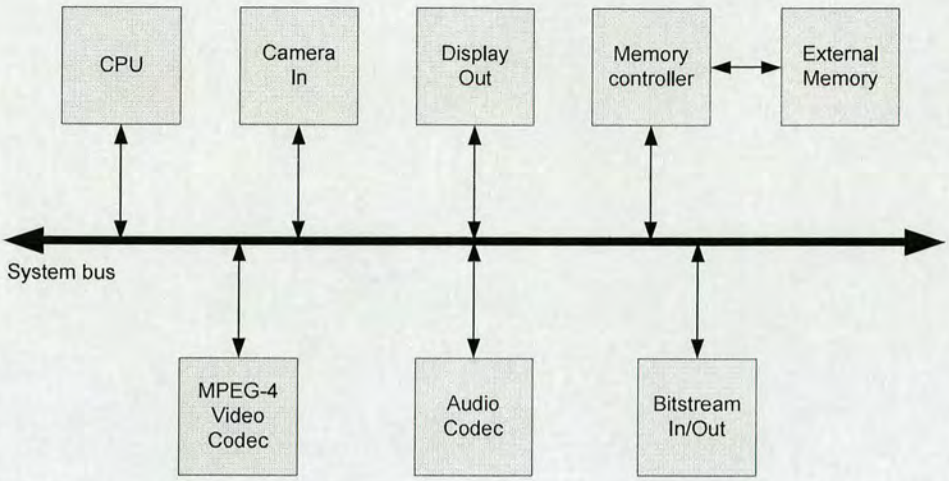


Figure 7.1: *Typical video communication system.*

For video communication where pixels are continuously being transferred to and from external memory, bus power consumption cannot be neglected.

This chapter focuses on minimising the power dissipation during pixel transfer from multiple frames on an off-chip system bus. The power reduction is achieved by utilizing bus encoding to reduce switching activity on the bus. The method focuses on transferring raw video data (pixels) between off-chip memory and on-chip memory, which is common in video compression applications. This method is based on entropy coding to minimize bus transition. The existing technique exploits the correlation between neighbouring pixels. In this technique, pixels' correlation between two consecutive frames are exploited.

This chapter is organised as follows. Section 7.2 reviews the existing intraframe techniques for bus encoding. Section 7.5 discusses the proposed approach to reduce the transition activity during memory data transfer. This is followed by the results and performance benchmarking of the proposed method in Section 7.6. Finally, Section 7.7 concludes the chapter.

Total Reference Frame	QCIF	CIF	4CIF
1	0.3	1.2	4.8
2	0.6	2.4	9.6
3	0.9	3.6	14.4
4	1.2	4.8	19.2
5	1.5	5.9	24.0

Table 7.1: *QCIF, CIF and 4CIF frame buffer size YUV420 in MByte*

7.2 Bus Encoding

In an MPEG-4 encoder, the encoded frames are stored in the frame memory. This frame will be used for prediction for the next frame. Table 7.1 shows the frame buffer size for QCIF, CIF and 4CIF frames at different numbers of reference frames. As the frame size and number of reference frames increase, the reference buffer increases. With this size of memory, it is not practical for on-chip SRAM.

SDRAM is more popular for frame buffer because it requires a smaller footprint. Since the off-chip SDRAM comes with standard sizes, it can be mass produced, which reduces the cost. Furthermore, advanced video codecs tend to use more reference frames in their predictions. This necessitates even more storage for the frame buffer.

At the system design level, there are three main parts which significantly affect the power consumption: computational units, communication units and storage units [112]. Bus encoding falls into the second category, where it reduces the power in system communication using the bus system. Bus encoding minimises the amount of power by reducing the number of transitions on the bus. This is achieved by transforming the original data (source word) into another form (codeword) so that the consecutive transmitted data minimises the total transition.

Clk	Transmitted Data	Unencoded Bus (8 bit)	Bus Invert (1+8 bit)
1	0	0000 0000	0 0000 0000
2	95	0101 1111	1 1010 0000
3	228	1110 0100	0 1110 0100
4	252	1111 1100	0 1111 1100
5	19	0001 0011	1 1110 1100
Total Transition		21	10

Table 7.2: *Bus encoding example*

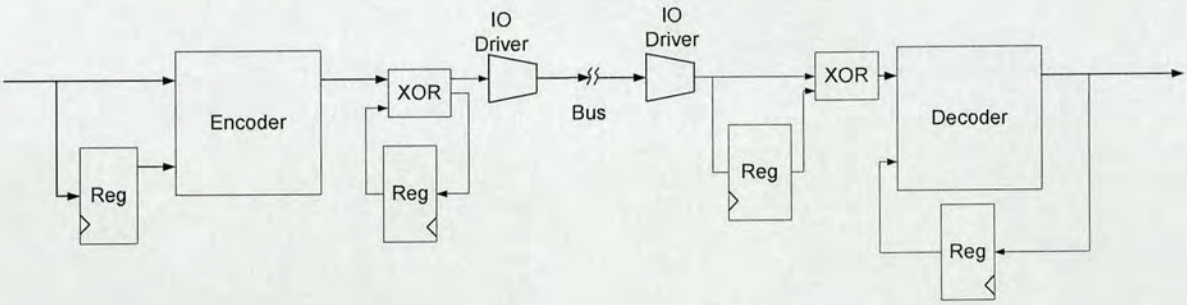


Figure 7.2: *Generic bus encoder*

Table 7.2 shows an example of bus encoding implemented on an 8bit bus. Transmitting the original data using the unencoded bus results in a total 21 transitions within the given 5 clock cycles. Using bus invert [113], an additional bus is introduced, resulting in 9 bits of data. This technique reduces the total bus switching to 10, which will translate to a significant total power reduction.

The generic bus encoding is shown in Fig 7.2. It consists of a data source and a receiver. The encoder is located close to the source, while the decoder is located close to the receiver. Typically, the encoder consists of a register, encoding unit and XOR circuit. Similarly, the decoder consists of a register, decoder unit and XOR circuit. The register stores the previous sourceword that will be used for the encoder/decoder unit. The XOR translates the binary value of the codeword such that '1' will result in transition, while '0' will result in non-transition. Thus, by using the XOR, the encoding function is simplified to minimise the

number of values of '1' in the codeword.

7.3 Previous Work

In general, the encoder function can be divided into three categories [114]. These are algebraic, permutation and probability. The algebraic function uses arithmetic or Boolean operations to generate the codeword. Permutation generates the codeword based on certain rules and conditions. In the probability method, the codeword is generated based on the statistical distribution of the transmitted data. The statistics may be obtained before designing the bus encoder or generated online during the transmission.

There are two types of buses: address and data. Address buses show correlated properties because high numbers of patterns in address buses are consecutive. This property has been widely exploited to reduce transitions on address buses. Reference [115] proposed sending the address in the form of gray code rather than binary form. This technique minimises the address bus transition by allowing one transition for every consecutive address. This method result up to 50% transition reduction on the address bus.

Reference [116] introduces zero-transition activity (T0) for address buses. The actual address is sent if the address is not consecutive. For consecutive addresses, a signal is sent to inform the decoder that the address is sequential. No actual addresses are sent at this stage. Further improvement to T0 is proposed by [117]. The method removes the need for redundant bits from the encoder by XOR-ing the T0 equation.

Compared with the address bus, the data bus shows more random characteristics. For this

data, the bus invert [113] provides a robust approach to reduce switching activity. It reduces the data transition by sending the inverted input if this will result in fewer bus transitions compared with sending the original value. This technique needs extra bits to inform the receiver of how to decode the input. For every input, the hamming distance between the current input and the bus value will be calculated. The input will be inverted if the hamming distance is larger than half of the bus size (including the redundant bit). This method ensures that all transferred data have transitions fewer than or equal to half of the total bus width. Reference [118] improved the bus invert method by introducing partial bus invert (PBI). This method selects a group of bits that will result in minimum transition.

Reference [119] implements a codebook method. Based on the input, this method finds a pattern that will result in minimum hamming distance at the output. This method updates the codebook according to the previous input. A generic bus encoding framework is discussed in [120]. The method utilises correlation between input data and reduces average transition by 36%. Reference [121] proposes an Exact-algorithm by exploiting the joint-probability of input data. This method achieves a transition reduction of more than 90%. However, this method requires a large circuit area for the mapping table. The mapping table grows exponentially as the bus width increases.

Most of the existing literature includes video data in its analysis. However, since these methods are geared more toward generic applications, bus encoding for MPEG-4 applications and its implementation in hardware has not been studied in depth. Few papers address bus encoding for MPEG-4 system such as in [109] where it discussed applying bus invert and gray coding for MPEG-4 data.

While many approaches can be used to design an encoder or decoder unit, there are certain constraints that limit the freedom in designing this circuit, mainly:

- Available bus width: The number of transitions can be reduced if the data can be transmitted in a wider bus, but this increases the pin count. Furthermore, the number of available data buses largely depends on the memory data width.
- Codeword decodability: The generated codeword must be able to be decoded correctly at the receiver
- Computational latency: Complex algorithms to encode and decode the data can result in better transition reduction. However, the performance may degrade the overall system throughput since more clock cycles are spent in generating the codeword.
- Encoder and decoder overhead: Additional circuitry is required to perform the encoding and decoding operation. The circuit area should be kept to a minimum to ensure that the power consumed by the circuit does not exceed the power saving achieved by encoding the bus.

In non-redundant methods, the decoder depends on received data to decode the codeword without any additional signals. The decoding information must either be embedded in the received data or predefined for each codeword. In redundant methods, an additional bit is added on top of the existing bus to signal how the data should be decoded. Figure 7.3 shows two cases where bus communication is used. In case (a), the encoded data is stored in the memory and shared by Modules A and B. In this case, the redundant bit has to be stored in the memory because it is required by both modules to decode the data. In case (b), the redundant

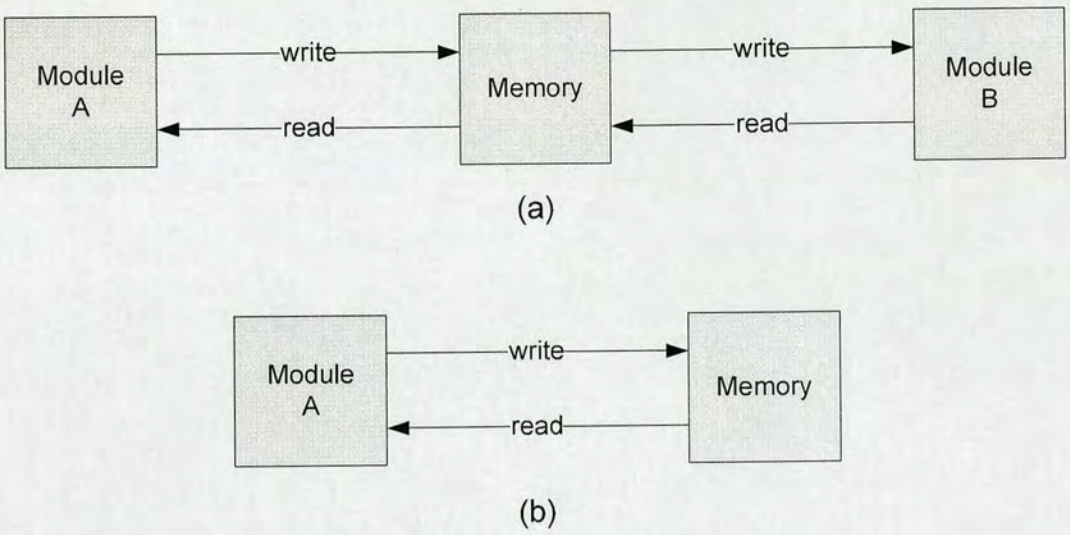


Figure 7.3: *Bus transmission scenario*

bit can be stored at Module A and only the codeword is stored in the memory, as discussed in [122]. Both pieces of information will be read to decode the data. For large amounts of data, as in image or video applications, storing the redundant bit as in case (b) will require additional large amounts of memory.

7.4 Intraframe Decorrelation

The technique discussed in this section is based on the combination of difference-base-mapped and value-base-mapped (dbm-vbm), as discussed in [120]. This method is adopted to exploit the pixel correlations widely available in video data.

In MPEG-4 memory-processor communication, the main data transfer consists of streams of pixels representing image sequences. Often, the pixels are scanned in a block-based manner to maintain high correlation between adjacent pixels. Figure 7.4 shows the probability distribution of two adjacent pixels' difference. Four different QCIF video sequences (Akiyo,

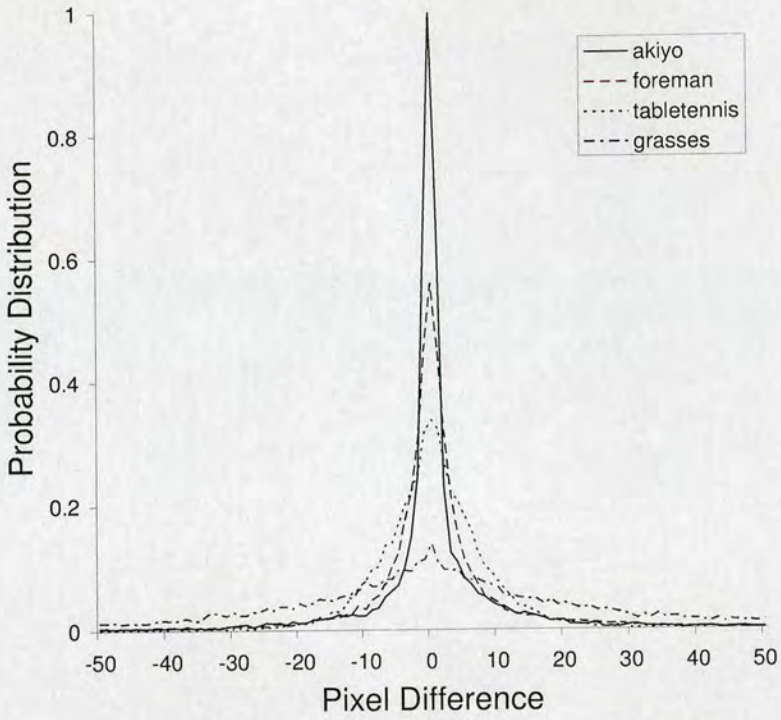


Figure 7.4: *Intraframe decorrelation using adjacent pixel.*

Foreman, Table Tennis and Grasses), which represent various motion types from low to high, are evaluated. Five frames from each sequence are evaluated, which consists of 190080 pixels. The graph shows that, for highly correlated data, the difference between two consecutive pixels with smaller magnitude has higher probability than the larger magnitude. Dbm-vbm utilises this characteristic to minimize the bus transition.

In addition, the bus switching activity for image data is non-uniformly and non-stationary as shown in Figure 5.1. The distribution depends on the type of video sequences. Frame with high texture tend to have more bits with high switching activity, and vice versa.

Figure 7.5 shows the block diagram describing the dbm-vbm operation. It consists of decorrelator (dbm) and entropy coder (vbm). The dbm-vbm technique is summarized as follows. First, two adjacent pixels (intraframe) are decorrelated using dbm. dbm calculates the rela-

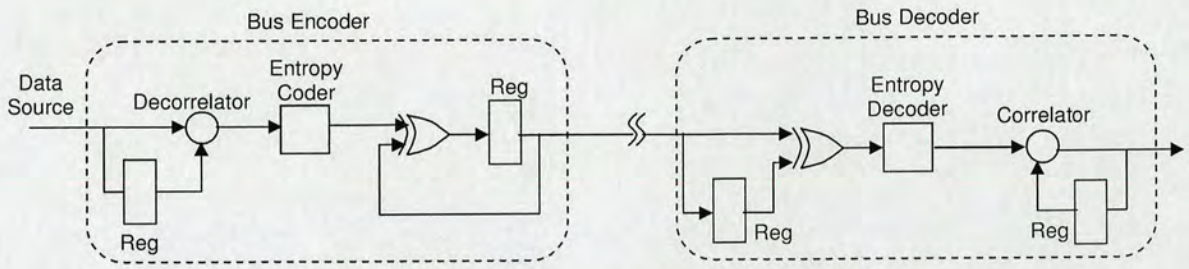


Figure 7.5: *dbm-vbm bus encoder and decoder.*

tive difference between the two pixels. vbm maps the values to patterns that have different weights (i.e., total number of 1s). To reduce the overall transition, it maps the low magnitude value to a pattern that has the fewest 1s, whereas higher magnitude values are mapped to patterns that have more 1s. At the output, the XOR translate 1s as transition and 0s as transition-less.

The average number of transitions for the dbm-vbm method depends on its source-word, i.e., the decorrelator output. The more the graph is skewed toward zero, the more patterns are assigned with less 1s. Thus, one way to improve the transition reduction is by improving the decorrelator.

7.5 Interframe Decorrelation

Video sequences consist of both spatial and temporal redundancy. The existing bus encoding techniques utilize spatial redundancy within frame. However, the temporal redundancy is not fully exploited to reduce bus transition.

In the proposed method, decorrelating the pixels using two consecutive frames (interframe) is proposed. This method is based on the observation that two consecutive frames are highly

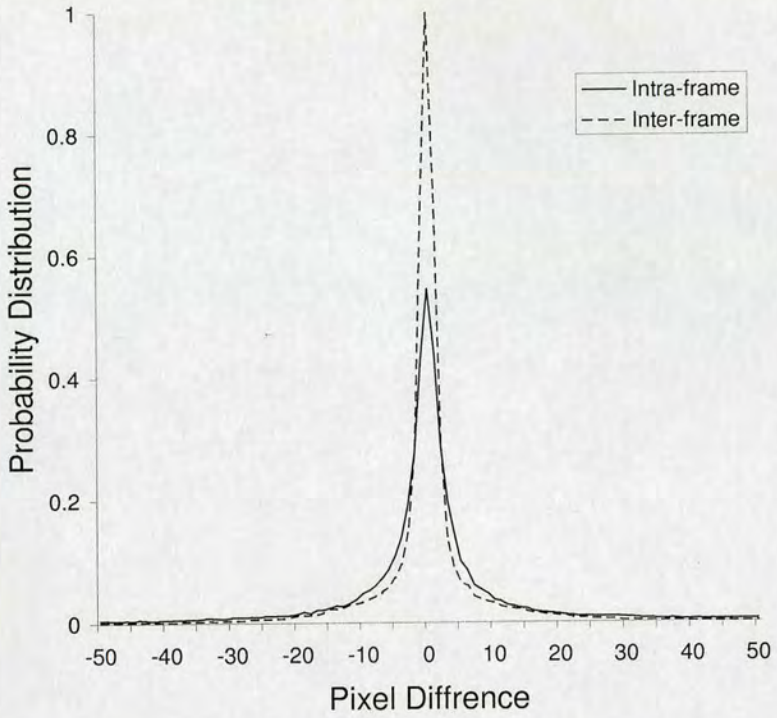


Figure 7.6: *Interframe vs. intraframe decorrelation for Foreman sequence.*

correlated. Often, the background of a scene is stationary. Furthermore, for a moving object, the differences between successive frames are very small. Figure 7.6

compares the pixel decorrelation using intraframe approach with that using interframe approach for the Foreman sequences. The figure shows that decorrelating the pixels using the interframe approach improves the graph skew-ness towards zero. This will translate to higher transition saving since more patterns will be assigned with less 1s.

In order to demonstrate the effectiveness of the proposed method, the setup shown in Figure 7.7 is used. Assuming 8 bits represent each pixel, two 8-bit buses are required to transfer two pixels in parallel as shown in Figure 7.7(a). Let frames A and B be the two consecutive frames, and BusA and BusB be the buses that transmit these frames, respectively.

To exploit the redundancy of these buses, one bus is decorrelated with respect to the other. Let

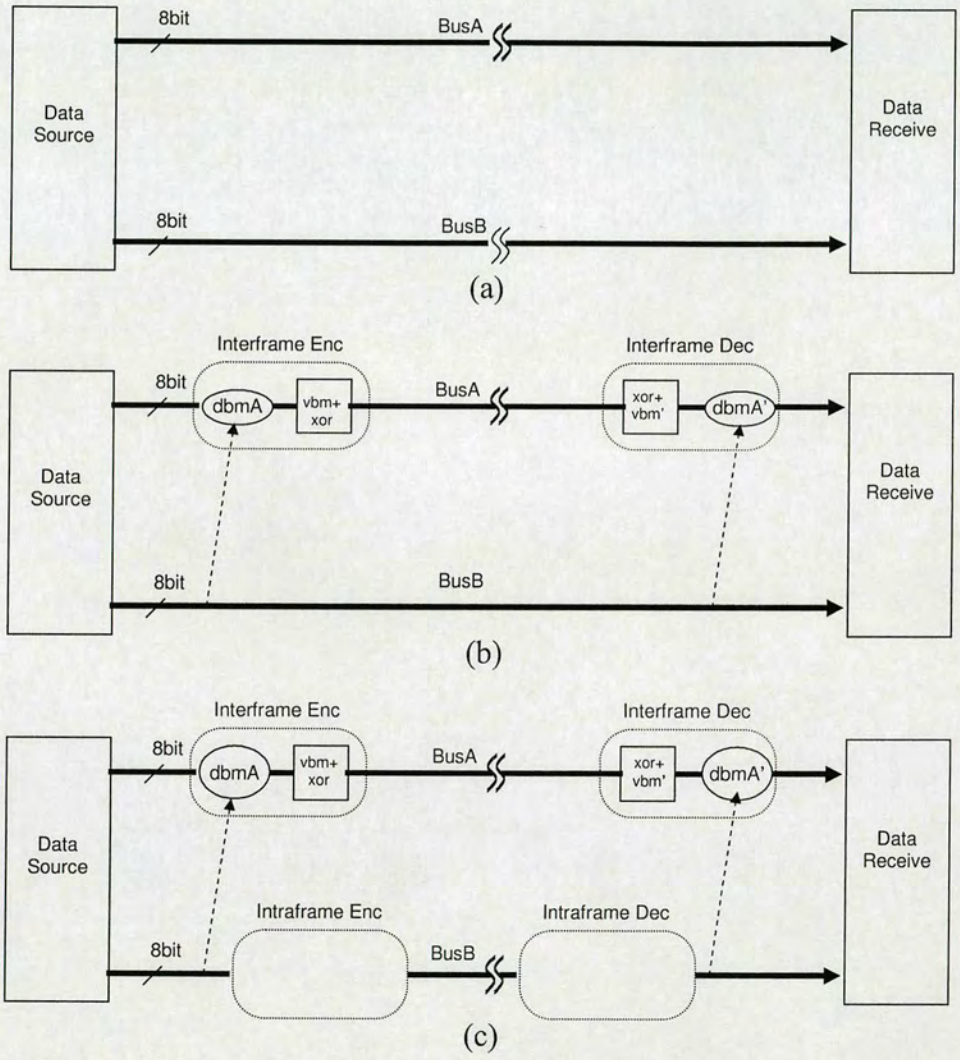


Figure 7.7: *Experiment setup: (a) Unencoded buses (b) Encoding BusA using interframe (c) Encoding BusA and BusB.*

BusB be the reference bus. BusA is then decorrelated with respect to BusB using dbmA. The dotted arrow in Figure 7.7(b) represents this. DbmA has a similar function to dbm, except that it calculates the relative difference between two current inputs. The output from dbmA is then mapped to the vbm table and XOR-ed as in intraframe. The decoder works as an inverse function of the encoder. To reduce the bus transition on the reference bus, the existing bus encoding techniques (intraframe) is applied to BusB. Figure 7.7(c) shows the complete setup for the proposed bus encoding technique.

7.6 Results and Discussion

In this section, the effectiveness of the interframe bus encoding technique as applied to BusA is first analysed. Next, the effect on interframe transition as the distance between frames increases is analysed. Finally, the total power reductions achieved utilising the proposed method are compared. Eight types of video sequences (150 frames each), which represent different sequence classes, were used throughout the analyses, as discussed in Section 3.6.

The proposed method was benchmarked against the intraframe method and bus invert method. As shown in Figure 5.1, the data switching activity is non-uniform. Thus, using normal bus invert method for the whole 8 bits will not result in an optimal solution. Furthermore, since the switching activity is non-stationary, partial bus invert [118] is not optimal for this type of data since the choice of bus lines involved is static. Instead, adaptive partial bus invert (APBI) as proposed in [123] and clustered bus invert (BI) are chosen as comparisons to the proposed technique. Simulation results show that grouping a 4-bit bus at the LSB and another 4-bit bus at the MSB gives better transition saving for a wide range of image data.

Bus	Unencoded		BI		APBI		Intraframe		Interframe	
	Trans	Sav	Trans	Sav	Trans	Sav	Trans	Sav	Trans	Sav
Akiyo	3539814	0	2734827	22	3075712	13	2029893	43	434764	88
Foreman	4252973	0	3282433	23	3693882	13	2476283	42	1993008	53
Container	4030081	0	2981150	26	3496356	13	2268541	44	1053195	74
Silent	4482196	0	3364351	25	3892186	13	2611724	42	1233714	72
Table Tennis	4529724	0	3251153	28	3951140	13	2491353	45	1197771	74
Hall	4004324	0	3086181	23	3519554	12	2395288	40	1440674	64
Coastguard	4548029	0	3564119	22	3925721	14	2659735	42	2471867	46
Mobile	5252726	0	4130726	21	4631436	12	3573444	32	2596130	51
Avg Sav	0		24		13		41		65	

Table 7.3: *Total number of transitions (Trans) for bus encoding applied to BusA, and percentage of transition reduction when compared with unencoded bus (Sav)*

Table 7.3 shows the percentage of transition reduction for applying bus encoding techniques to BusA as in Figure 7.7(c). The results show that the interframe method provides higher transition reduction compared to both bus invert and intraframe implementations. On average, the proposed method reduces 65% of the transition over unencoded buses. This is equivalent to a 1.6 and 2.7 times more transition saving over intraframe and clustered bus invert, respectively. APBI results in low transition saving compared to clustered bus invert. This is because the APBI requires some delay before it determines the right combination of bits to be inverted. This reduces the effectiveness of this technique compared to clustered bus invert.

The amount of switching reduction is dependent on the frame characteristics. For frames with a low amount of movement, the amount of transition reduction is large (88% as in Akiyo). This is because the correlation between successive frames is high. Alternatively, for frames with high amounts of movement, the correlation between frames is low. This results in less transition reduction, as shown in the Mobile sequences (51%). However, in both cases, the

Frame Distance	Akiyo		Foreman		Table		Mobile	
	Trans	Sav.	Trans	Sav	Trans	Sav	Trans	Sav
1	434764	88	1993008	53	1197771	74	2596130	51
2	508545	86	2508229	41	1310800	71	3269278	38
3	568612	84	2795122	34	1348278	70	3652606	30
4	605701	83	3024017	29	1445311	68	3896325	26
5	652209	82	3145602	26	1484711	67	4065273	23

Table 7.4: *Total number of transitions (Trans) for different frame distances applied to interframe bus encoding, and percentage of transition reduction when compared with unencoded bus (Sav)*

interframe shows superior performance compared to intraframe and bus invert techniques.

In addition, as the distance between two frames increases, the frame correlation decreases. This is reflected by the decrease in transition reduction for interframe, as shown in Table 7.4. For high motion sequences, the decrease is more rapid, as shown in the Mobile case. Increasing the frame distance from 1 to 2 degrades the performance by 25%. On the other hand, for low motion sequences (such as Akiyo), an increase in frame distance only degrades the transition reduction by about 2%. In general, using two consecutive frames improves transition reduction when the frame distance is kept as close as possible and the correlations between frames are high.

Table 7.5 compares the total transition saving on both BusA and BusB, as in Figure 7.7(c). Since interframe is only applicable on BusA, intraframe is implemented on BusB. The results show that the interframe-intraframe combination (inter-intra) shows superior performance compared to the intraframe-intraframe pair (intra-intra). The inter-intra combination gives a 53% transition reduction over unencoded bus. This is equivalent to a 29% improvement over the intra-intra pair. This improvement is because of the higher transition reduction due to the interframe technique on BusA.

Bus	Unencoded		BI-BI		Intra-Intra		Inter-Intra	
	Trans	Sav	Trans	Sav	Trans	Sav	Trans	Sav
Akiyo	7080214	0	5468705	23	4059929	43	2464800	65
Foreman	8509990	0	6566297	23	4954007	42	4470732	47
Container	8061030	0	5963733	26	4537337	44	3321991	59
Silent	8963679	0	6728665	25	5222477	42	3844467	57
Table	8541122	0	6339829	26	4888561	43	3933947	54
Hall	8534373	0	6337716	26	4887386	43	3593804	58
Coastguard	9096094	0	7128551	22	5319546	42	5131678	44
Mobile	10507221	0	8261831	21	7147641	32	6170327	41
Avg Sav	0		24		41		53	

Table 7.5: *Total number of transitions (Trans) for different bus codings implemented on BusA and BusB, and percentage of transition reduction when compared with unencoded bus (Sav)*

The bus encoder and decoder are synthesized using $0.13\mu m$ UMC technology library with 1.25 V supply voltage at 20 MHz clock frequency. The design was mapped onto logic gates using Synopsys Design Compiler, and the functional behaviour was verified using Verilog XL. Synopsys Power Compiler was used to perform gate level power evaluation.

Table 7.6 shows the total encoder and decoder circuit area (μm^2), power (μW) and delay (ns) overhead for implementing bus coding on BusA and BusB (refer to Figure 7.7(c)). Both inter-intra and intra-intra require more area compared to bus invert. This is because they require more logic gates to implement the decorrelator and vbm look-up table. Compared to intra-intra, inter-intra requires less area and power. This is because the intraframe decorrelator requires additional registers to store the previous input value whereas the interframe decorrelator uses current bus values as its input. This eliminates the need for extra register in the interframe decorrelator.

Figure 7.8 shows the total power consumption for bus encoding techniques implemented in Figure 7.7(c). The total power consumption, P_T, is calculated as $P_T = P_{Enc} + P_{Dec} +$

BusA	BusB	Area (mm^2)	Power (mW)	Delay (ns)
BI	BI	1936	84	2.6
Intra	Intra	29050	730	4.5
Inter	Intra	28691	635	4.3

Table 7.6: *Total encoder and decoder circuit overhead for different bus coding implemented on BusA and BusB*

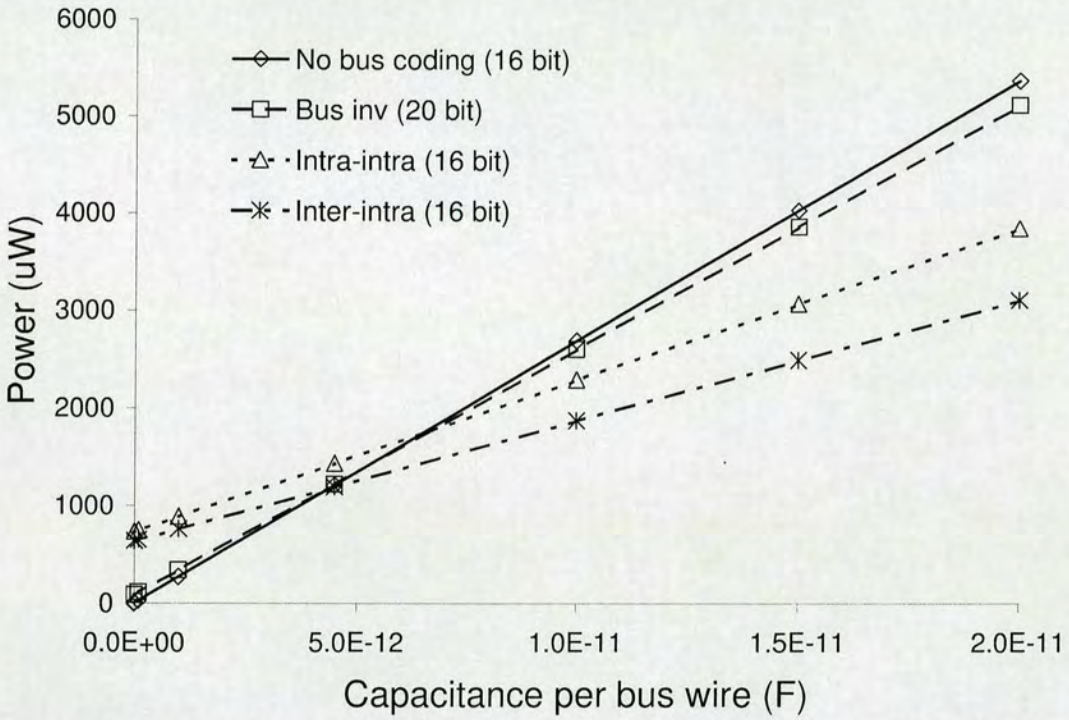


Figure 7.8: *Total bus power consumption vs. wire capacitance*

P_{CL}, where P_{Enc} and P_{Dec} represent the power consumption due to bus encoder and decoder circuits. P_{CL} is the total bus power consumption estimated by $P_{CL} = \frac{1}{2} C_L V_{dd}^2 f \alpha$. In general, the slope of the graphs in Figure 7.8 is proportional to α, which is dependent on the type of bus encoding used.

From the graph, for capacitances less than 5pF, the circuit power offsets the power saving on the buses. However, for wire capacitances greater than 5pF, the bus power reduction exceeds the circuit power overhead. This results in inter-intra dissipating much lower power

compared to the other techniques. For a typical off-chip wire capacitance of 15pF [110], the total bus power consumption is 2.5mW compared to 4mW for unencoded buses. This is equivalent to 38% power savings, compared to only 24% and 5% using intra-intra and bus invert, respectively. The greater power savings achieved in inter-intra is due to much lower transitions occurring on the buses. A smaller slope, as shown in the graph in Figure 7.8 reflects this.

7.7 Summary

We have presented an interframe bus encoding technique for MPEG-4 applications. The combination of interframe and intraframe results in a 53% transition reduction over unencoded bus. This is equivalent to 29% transition reduction improvement compared to existing techniques that use the intraframe approach. This method is suitable for applications where multiple frames are being transferred between memories, such as during motion prediction in MPEG-4 AVC/H.264 encoder. The implementation of this technique in H.264 system will be presented in the next chapter.

Chapter 8
Low power H.264 System

8.1 Introduction

For real time implementation, a hardware accelerator is important to overcome the increase of computational load in a video compressor such as the H.264. In addition, compared to general purpose processor implementation, dedicated hardware acceleration requires less power consumption. This makes this approach attractive for energy critical appliances such as portable devices.

In this chapter, a H.264 system is discussed. To minimize the power, the low power techniques proposed in the previous chapters are integrated into the system. The effects of these low power techniques on the overall system are investigated to measure the effectiveness of these methods.

This chapter is organized as follows. Section 8.2 provides an overview of the H.264 system and discusses the architecture of each of its components. Section 8.3 discusses the implementation of the two-step ME and interframe bus encoding into the mpeg system. The results of implementing the low power techniques are discussed in Section 8.4. Finally, Section 8.5 concludes this chapter.

8.2 H.264 Architecture

In the standard H.264, several profiles are defined to suit different applications. This work focuses on the baseline profile, since its computational load is suitable for mobile devices. The main modules in this profile are motion estimation, intra prediction, transform coding, quantiser, deblocking filter and entropy coding. For the entropy coder, the baseline profile uses CAVLC to code the MB coefficient. In order to provide real time communication, each MB must be processed at a specified period. The constraint on the total clock cycle to process one MB is largely determined by the frame rate and frame size.

In order to evaluate the low power techniques, an H.264 system was built, as shown in Figure 8.1, which will be referred to as the conventional system in this thesis. The architecture of each module in the system has been carefully selected based on existing literature to achieve real time implementation [95, 124–127].

This system consists of a motion estimator (ME), a motion compensator (MC), a transform coder, a deblocking filter (DFIR) and an entropy coder (EC). The transform coder is comprised of an integer transform (IT), an inverse integer transform (IIT), a quantiser (Q) and an inverse quantiser (IQ). Modules dealing with intra prediction and fractional pel motion estimation were not included due to time constraints. Furthermore, since the proposed low power techniques mainly affect the integer motion estimation and transform coding loop, intra prediction and fractional pel motion estimation are not directly affected by these techniques. The architecture aimed to process QCIF resolution (YUV420) at 30fps with a search range of $p = [-8, 7]$.

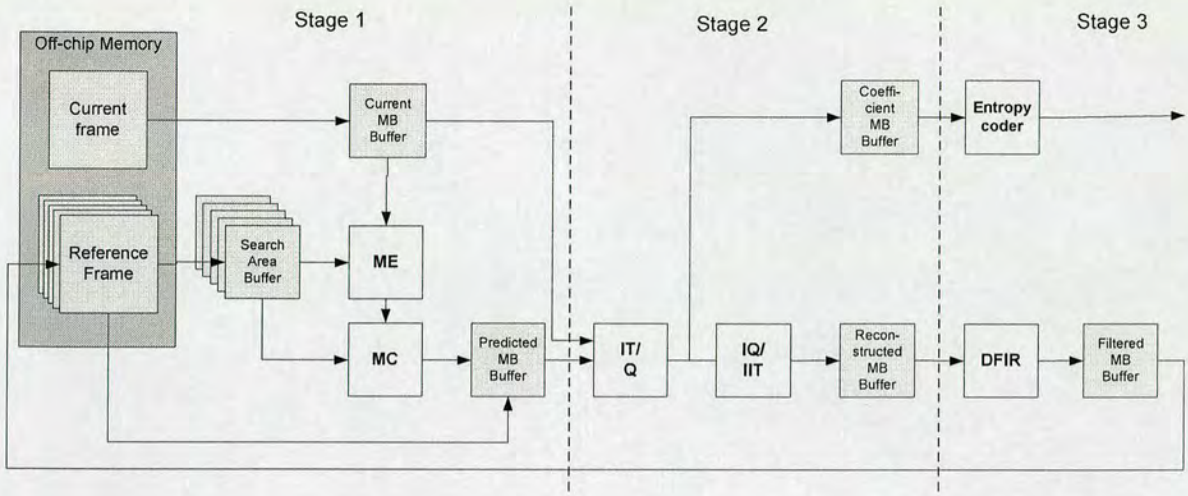


Figure 8.1: *MPEG system 3 stage pipeline*

The basic processing unit in a video encoder is a macroblock. To encode the macroblock in serial order, starting from predicting the current MB to bitstream generation would result in slow throughput and low hardware utilization. To overcome this problem, macroblock pipeline processing is typically adopted in MPEG hardware architecture [128].

In this work, the system is divided into three pipeline stages. The first stage performs motion prediction. This stage includes loading the search area from external memory into on-chip memory and performing motion estimation and motion compensation. The second stage performs transform coding including integer transform, quantiser, inverse quantiser and inverse integer transform. Since the entropy coder and deblocking filter are operated based on output from the transform coder, these operations are executed in the third pipeline stage. This arrangement allows the hardware of each stage to be ready for processing the next MB once the output is stored into the pipeline buffers.

Table 8.1 shows the amount of clock cycles necessary to operate each module. From the table it can be seen that the number of clock cycles required for each pipeline stage depends

Modules	Total Reference Frames				
	1	2	3	4	5
ME/MC	496	896	1296	1696	2096
IT/Q/IQ/IIT	178	178	178	178	178
DFIR	208	208	208	208	208
EC	1920	1920	1920	1920	1920
Maximum clock	1920	1920	1920	1920	>1920

Table 8.1: *Clock cycle requirements for each module*

on the number of reference frames used during motion estimation. For less than four reference frames, the entropy coder consumes the largest number of clock cycles. The motion estimation consumes the largest clock cycle compared to other modules for five or more reference frames. In this design, a maximum of four reference frames are used which allows the throughput for the pipeline to be set to 2000 clock cycle permacroblock. To achieve real-time operation, the clock has to operate at 6MHz for QCIF@30fps.

To synchronize the pipeline operation, the intermediate results of each stage are stored in pipeline buffers (SRAM). Figure 8.1 shows that five pipeline buffers are required to store information about current MB, predicted MB, reconstructed MB, coefficient MB and filtered MB. Each buffer stores four pixels per word for each Y, U and V component. Table 8.2 summarises the size, type, and the number of SRAM required for each buffer. In total, 28 Kbits of pipeline buffer are required to synchronize the operation.

The pipeline buffers operate as follows. The current MB buffer stores the pixels from the off-chip memory which consists of luma (Y) and chroma (UV) components. The buffer is read during the ME initialization and transform coding stage. The two main outputs from the transform coding, i.e. the transform coefficient and reconstructed MB, are stored into the coefficient buffer and reconstructed MB buffer, respectively. The reconstructed MB buffer

Buffer	Size	Type	Total SRAM	Data
Current MB	96x32bit	1port	2	YUV
Predicted MB	96x32bit	1port	1	YUV
Reconstructed MB	96x32bit	2port	1	YUV
Coefficient MB	96x64bit	1port	2	Transform coefficient
Filtered MB	104x32bit	1port	1	YUV
Total Size	27904 bits			

Table 8.2: *Pipeline buffer's size and type and the total number of SRAM required*

is read during MB filtering and the output is stored into the filtered MB. The entropy coder utilizes the data in the coefficient buffer and outputs the generated bitstream into the output buffer. The architecture of each module used in the H.264 system is discussed in the next section.

8.2.1 Motion Estimation

For search range $p = [-8, 7]$, where four pixels per word (32 bits) are transferred from the off-chip memory, the search area of the leftmost MB requires at least 256 clock cycles to store the pixel into the search area RAM. For the search area of other MB positions, since half of the neighbouring search area is overlapped and can be reused, only 128 clock cycles are required to update the one search area.

The search area memory contains 16 memory banks (SRAM) to store the search area for each reference frame. For a search area with size $W \times H$, the search area is divided into $\frac{W}{16}$ segments consisting of sixteen columns per segment. The first search area segment is stored in the first H address in the search area RAM. The second segment is stored in the second H address of the search area RAM, and so on. To support horizontal and vertical scans during the ME, the data is stored in a ladder-like style in the search area RAM [106].

Once the first candidate has been loaded into the PE, the ME requires 256 clock cycles to evaluate all candidates per reference frame. Details of the ME operation are described in Chapter 6. During motion compensation, the MC unit reads the luma components of the best prediction blocks from the search area RAM and stores the data into the predicted MB buffer. The chroma components are read from the external reference frame buffer. For MB other than the leftmost MB, the ME requires a total of 496 clock cycles to complete the operation. The number of clock cycles will increase for higher numbers of reference frames.

8.2.2 Transform and Quantiser Module

Compared to the conventional DCT, H.264 utilises an integer transform, an integer approximation of DCT [129]. The integer transform has the advantage of using only integer numbers, addition and shift during its operation. This ensures zero-mismatch when it is decoded [130].

Other than the integer transform, the H.264 performs 2×2 and 4×4 Hadamard transforms on the chroma DC and Intra16 luma DC, respectively. Figure 8.2 shows the architecture for the integer transform. The four luma pixel residue is first calculated by subtracting four current MB and four predicted MB pixels. The transformed coefficient is stored into the coefficient MB buffer. The reconstructed MB pixel is first output after the first 18 clock cycles. The next 64 clock cycles output all the luma components and stores them in the reconstructed MB buffer.

For the chroma component, the pixel is first transformed and quantised before the output from the inverse quantiser is stored into the temporary buffer. During the transform, the chroma DC values are stored temporarily into the 2×2 register before they are transformed by the 2×2

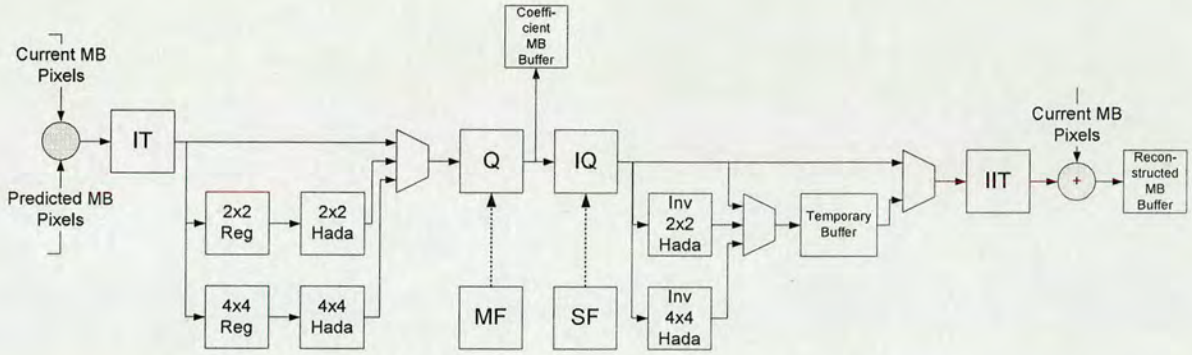


Figure 8.2: *Integer transform and quantiser architecture*

Hadamard transform (2×2 Hada). The output from the inverse 2×2 Hada of the DC chroma is then stored in the temporary buffer. When the entire chroma coefficient is available in the temporary buffer, the inverse integer transform will be performed on the chroma. The total clock cycles required to transform each chroma is 48 clock cycles. In total, the architecture requires 178 clock cycles to generate the reconstructed MB.

In the H.264, the quantiser is combined with the scaling factor that results from factorizing the DCT transform. By combining the quantiser and scaling factor, only one multiplication factor (MF) is needed for forward transform. The standard defines the MF for the first six QP values and its value is repeated for the rest of the QP values. The inverse quantiser scaling factor (SF) is also predefined by the standard.

8.2.3 Deblocking Filter

The deblocking filter consists of a filter circuit, transpose circuits, a delay buffer and an upper and left slice buffer as shown in Figure 8.3. Whilst the filtering operation is straightforward as defined by the standard [48], the horizontal and vertical filtering require complex data arrangement. Figure 8.4 shows the current MB and its neighbouring blocks. The left slice

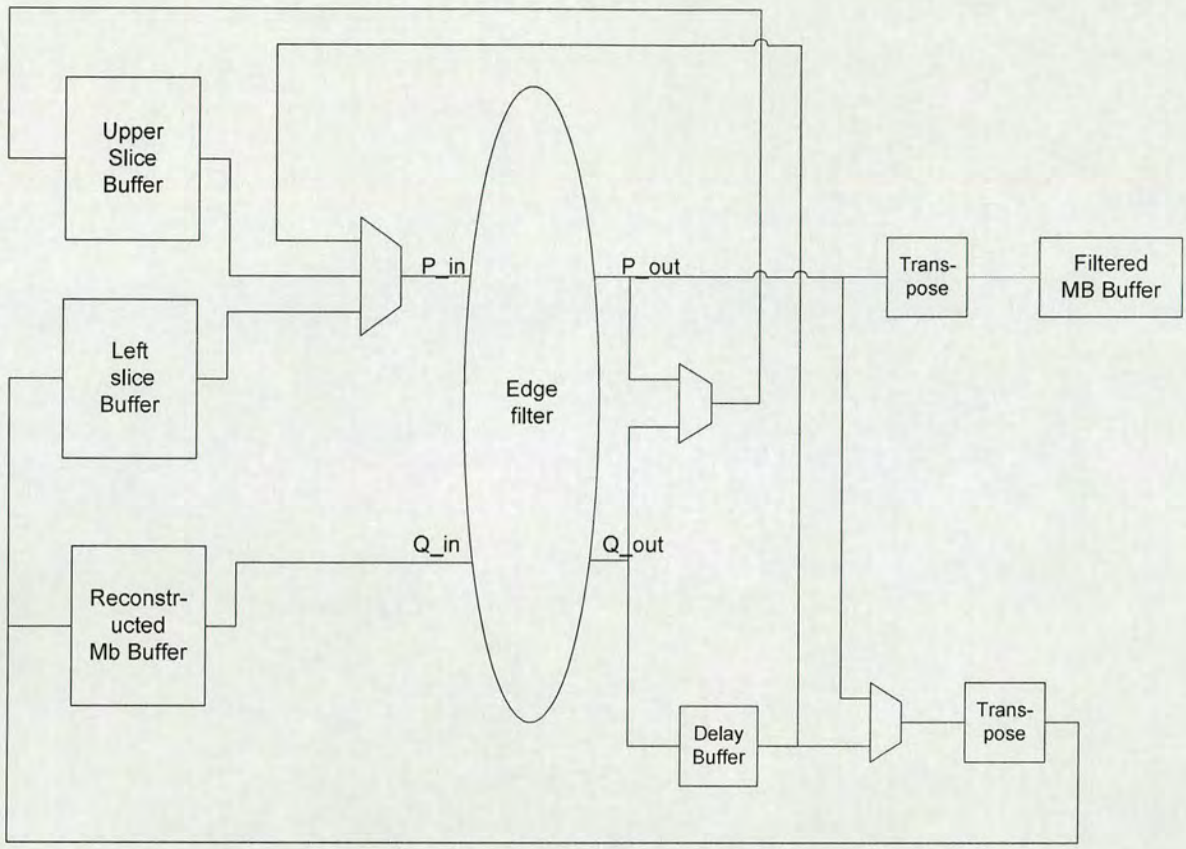


Figure 8.3: *Deblocking filter architecture*

buffer stores the block located on the left of the current MB for horizontal filtering. The upper slice RAM stores the neighbouring blocks that will be used during vertical filtering and its size depends on the frame width. The delay buffer temporarily stores the filtered pixel that will be used for filtering the next block edges. To rearrange the data for vertical filtering, the transpose unit transposes 4x4 pixels before storing them in the reconstructed MB buffer.

In this architecture, the horizontal filtering is performed first followed by vertical filtering. During the filtering, four pixels per clock cycle are accessed from the reconstructed MB buffer and another four pixels are accessed from the left slice, upper slice or delay buffer. Thus, the edge filter processes 8 pixels per clock cycle. In total, the luma component requires 128 clock cycles, whilst the chroma requires 32 clock cycles. An additional 8 clock cycles

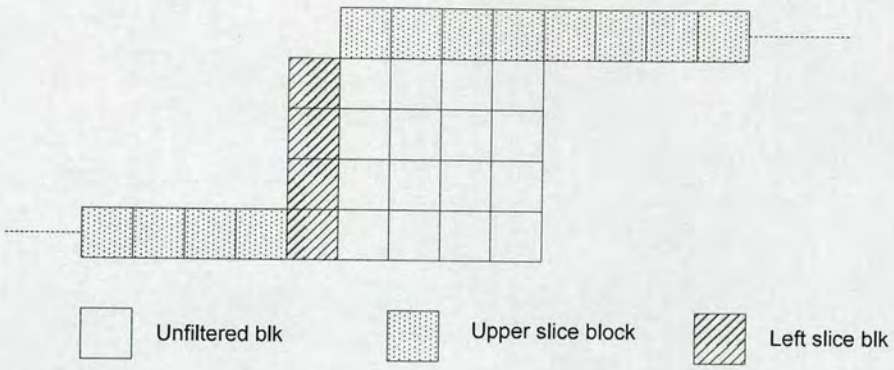


Figure 8.4: *Arrangement of blocks for deblocking filter: unfiltered blocks of current MB, upper slice blocks and left slice blocks.*

delay is required for each chroma in order to avoid conflict when storing the chroma pixels into the reconstructed MB during the change from horizontal to vertical filtering. In total, the deblocking filter requires 208 clock cycles to complete the operation and store the output into the filtered MB buffer.

8.2.4 Entropy Coding

The baseline profile uses Exp-Golomb and context adaptive variable length coding (CAVLC) to generate the compressed bitstream [131]. The Exp-Golomb is mainly used to code the picture, slice and MB information while the CAVLC is used to generate the codeword for the coefficient. The coefficient is first zigzag-scanned and the total sequence of zero, one and non-zero coefficients is coded. Since the generated codeword has a variable length depending on the coefficient information, the bitstream packer is used to group the generated bits every 32 bits.

During bitstream generation, the MB information is encoded first, followed by the coefficient information. The number of clock cycles required to encode one block depends on the co-

efficient data. For the best case, where all block coefficients are zeros, 20 clock cycles are required to code the block. The worst case occurs when all coefficients are non-zero. This requires 80 clock cycles to process one block. Thus, the maximum clock cycles required to generate bitstream for one MB is 1920 clock cycles.

8.3 Low power H.264 System

In order to reduce the power consumption of the existing system, the low power techniques proposed in the previous chapter are incorporated into the system. In the low power system, the ME unit is replaced by the proposed ME hardware that can perform a two-step search. This architecture is able to perform a full search and a low resolution search. During normal mode, the full search is selected. In the low power mode, the two-step search is utilized. The low power bus encoding is also incorporated to minimize the bus transition when the search area is loading from external memory to search area RAM.

8.3.1 Two-step Search

As discussed in Chapter 6, since the mc_m8p architecture (defined in Section 6.3.3) gives the largest energy saving compared to other architectures, this architecture is used to implement the two-step algorithm. The two-step search performs the low pixel resolution search followed by the full pixel resolution search. Thus, in total, the motion estimation requires 352 clock cycles per reference frame to initialize the ME and find the best candidate. This is equivalent to 30% more clock cycles than the conventional ME. Since the total pipeline MB

is set to 2000 clock cycles which is the maximum required by the entropy coder, a maximum of three reference frames are used during low power mode. This allows the system to be operated without increasing the clock frequency. Since the number of reference frames is reduced from four to three, a small drop in motion prediction quality is expected for this architecture.

8.3.2 Interframe Bus Encoding

In Chapter 7, the interframe bus encoding was found to be a significant improvement on the existing intraframe bus encoding. In this section, the proposed integration of the interframe bus encoding into the H.264 system is discussed. Due to the way in which pixels are stored and accessed going to/from the external frame memory, some modification is made on the interframe bus encoding circuit.

As shown in Figure 8.1, an interaction between the H.264 modules and the external reference frame buffer occurs on two occasions: (a) when the pixel is stored into the external reference frame buffer after filtering the MB through the deblocking filter; (b) during loading of the search area pixels from the external reference frame buffer into the search area RAM. Thus, the bus encoder and decoder should be implemented in (a) and (b), respectively.

Figure 8.5 shows the modified H.264 system to include the proposed interframe bus encoding method. To allow interframe decorrelation during bus encoding, reference frames stored in the external memory are arranged in a sequence as shown in Figure 8.6. Since the interframe technique requires other frames, the bus coding is alternated between intraframe and interframe.

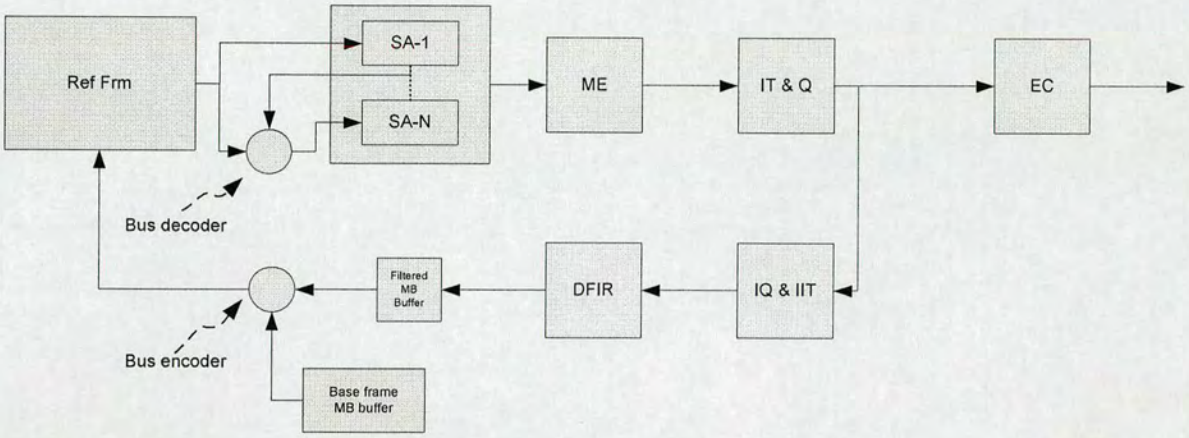


Figure 8.5: *H.264 system modified to include an interframe bus coder*

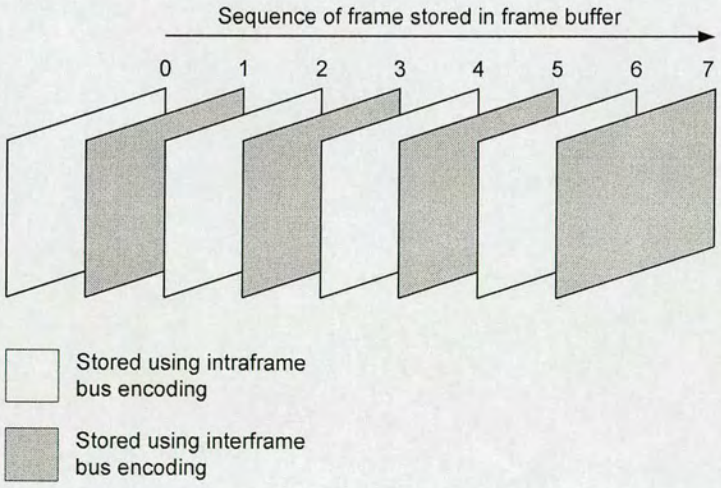


Figure 8.6: *Reference frame arrangement of external buffer and the type of bus encoding applied to it.*

Base frame (BF) refers to the frame that is stored as intraframe by the bus encoder, since it does not require any other frames. Dependence frame (DF) is a frame that is stored as interframe by the bus encoder, since it requires other frames to decode the pixel values. DF is always stored after its BF for ease of decoding later on.

In order to allow interframe decorrelation, pixels from the base frame have to exist for the bus encoder. The bus decoder has a similar requirement. For the bus decoder, since the search area is accessed from a multiple number of frames, the proposed solution is to load the base frame first, followed by the dependence frame. When the search area from the dependence frame is loaded, the search area pixel from the base frame is accessed as well to perform the interframe decorrelation.

In contrast to the bus decoder, since the fully filtered MB is only written into the external frame buffer, no base frame is available at the bus encoder. The solution proposed for this is to store the required base frame MB in a buffer. The base frame MB can be stored when loading the base frame search area into the search area memory. Since the deblocking filter operation is located in the third stage of the pipeline buffer, an additional MB buffer is required to store the last three MBs of the base frame. This buffer will be used during interframe decorrelation when writing the pixels into the reference frame buffers. Using this approach, it is possible to perform interframe decorrelation at the bus encoder.

The detailed hardware implementations for the bus encoder and decoder are shown in Figures 8.7 and 8.8. The hardware is able to perform either interframe or intraframe. When intraframe is selected, the bus encoder calculates the dbm decorrelation using the filtered pixel and the previous pixel stored in the register. Similar input is used to calculate the inverse of dbm at

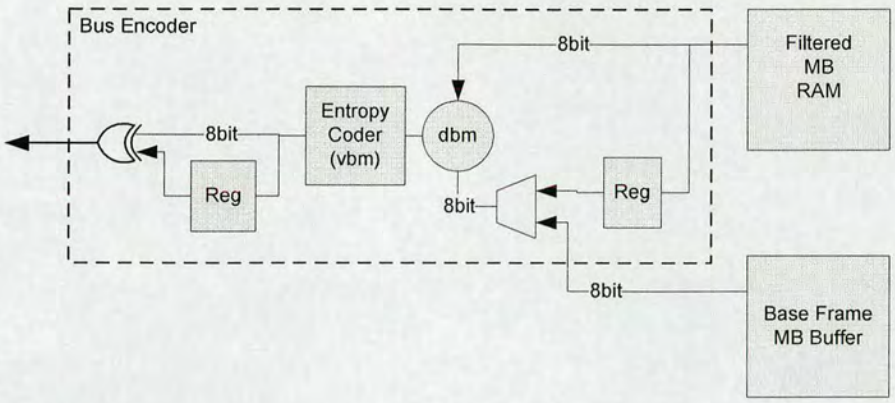


Figure 8.7: *Interframe-intraframe encoder for H.264 hardware*

the bus decoder.

During the interframe bus encoding, the encoder calculates the dbm using filtered pixels and pixels stored in the buffer MB. Since the buffer MB stores the corresponding base frame MB, this is equivalent to decorrelating a pixel from two frames. To decode the pixel, the search area from the base frame is first loaded into the search area buffer. When the search area from the dependence frame is loaded, the loaded search area from the base frame is read in parallel to correlate the search area pixel before storing it in the search area RAM.

8.4 Results

This section will first discuss the results of implementing the proposed two-step method into the H.264 system. Then, it will discuss the results of incorporating the proposed bus encoding into the system. In each of the sections, the effect on the overall area and power/energy is discussed using results obtained from the layout data.

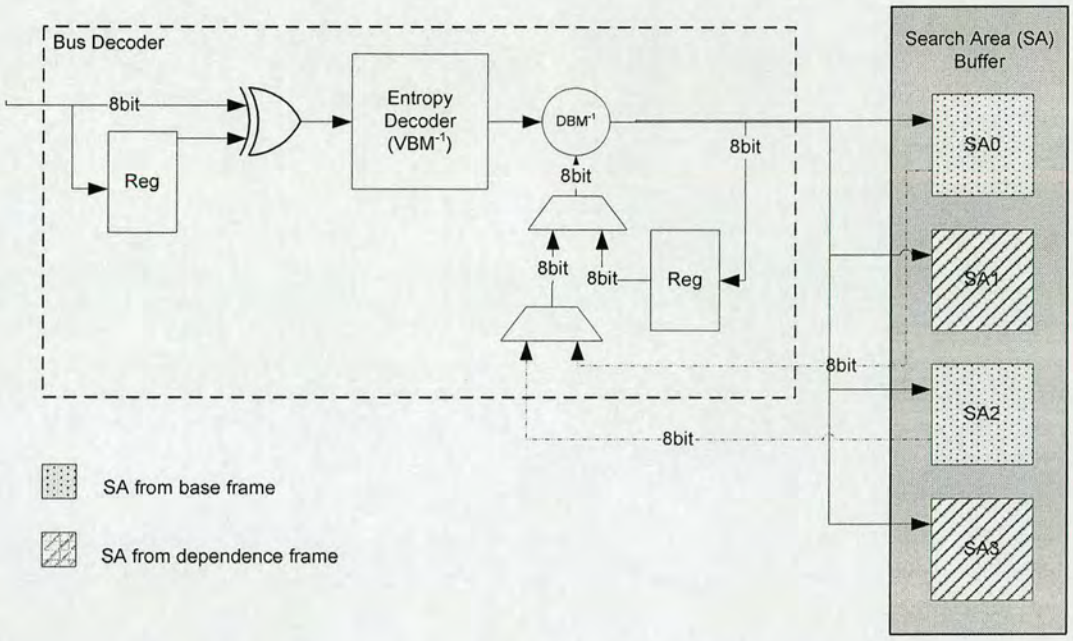


Figure 8.8: *Interframe-intraframe decoder for H.264 hardware*

8.4.1 Conventional H.264 Architecture

Table 8.3 shows the area used by logic and memory in the conventional H.264 system. From the table it can be seen that the logic takes up about 72% of the overall area whereas the memory unit takes up about 28%. The ME takes up the largest amount of area, about 53% compared to other modules. This is followed by the transform coder (IT/IIT/Q/IQ), entropy coder (EC) and the deblocking filter (DFIR), which take up 10%, 7% and 8% respectively. In total, the conventional architecture requires 4.85mm^2 of chip area for the chip core.

Table 8.4 shows the energy consumption for each logic module in the conventional architecture. Since different modules require different clock cycles to process one MB, the energy consumption is more accurate than the power values in representing the actual amount of work required to process MBs. In total, using one reference frame during motion prediction, the system consumes 695.43 nJ to process one MB within 2000 clock cycles (6MHz). The

Modules	Area (mm^2)			
	Logic	Memory	Total	%
ME	1.68	0.91	2.59	53
MC	0.04	-	0.04	1
IT/IIT, Q/IQ	0.44	0.03	0.47	10
DFIR	0.27	0.09	0.36	7
EC	0.39	-	0.39	8
Others	0.69	0.31	1.00	21
Total	3.51	1.34	4.85	100

Table 8.3: *Area for conventional H.264 architecture*

Modules	Energy (nJ)			
	Logic	Memory	Total	%
ME	460.76	75.51	536.27	77
MC	0.86	3.76	4.62	1
IT/IIT, Q/IQ	58.83	16.51	75.34	11
DFIR	36.61	17.41	54.02	8
EC	19.81	5.37	25.18	4
Total	576.87	118.56	695.43	100

Table 8.4: *Energy consumption (nJ) for the conventional H.264 architecture at 6MHz for one reference frame*

motion estimation dominates energy consumption, taking 77% of the total power. This is followed by the transform coder, deblocking filter and entropy coder at 11%, 8% and 4% respectively. For higher numbers of reference frames, the energy consumed by the motion estimation increases since the computation has to be repeated for each reference frame.

8.4.2 Two-Step Motion Estimation in H.264 System

By applying the proposed two step method, additional area is introduced to allow low resolution searches to be performed. As shown in Table 8.5, this increases the ME area by 0.49mm^2. At the system level, the additional area increases to 10% of the total area.

Modules	Area (mm^2)			
	Logic	Memory	Total	%
ME	2.17	0.91	3.08	57
MC	0.04	-	0.04	1
IT/IIT, Q/IQ	0.44	0.03	0.47	9
DFIR	0.27	0.09	0.36	7
EC	0.39	-	0.39	7
Others	0.69	0.31	1.00	19
Total	4.00	1.34	5.34	100

Table 8.5: *Area for the proposed H.264 low power architecture*

Modules	Energy (nJ)			
	Logic	Memory	Total	%
ME	218.81	46.93	265.74	62
MC	0.86	3.76	4.62	1
IT/IIT, Q/IQ	59.90	16.51	76.41	18
DFIR	36.61	17.41	54.02	13
EC	19.81	5.37	25.18	6
Total	335.99	89.98	425.97	100

Table 8.6: *Energy consumption (nJ) for the proposed H.264 low power architecture at 6MHz for one reference frame*

The introduction of the proposed two-step hardware results in reduced energy consumption in the ME as shown in Table 8.6. The total energy consumed by the ME using the proposed two-step method is 265.74nJ. The proposed architecture has reduced approximately 50% of ME power compared to the conventional ME architecture. The implementation of the low resolution which requires a smaller computational load has contributed to this saving.

Due to a slight decrease in the motion prediction accuracy during the two-step search, the residue generated from the two-step technique is slightly higher than the conventional ME. This results in a small increase in energy consumption (<2%) in the transform coder. Since most of the coefficient is quantised to zero, the small increase in residue is masked by the quantiser. Thus, the increase in the residue does not propagate to other modules, so there is

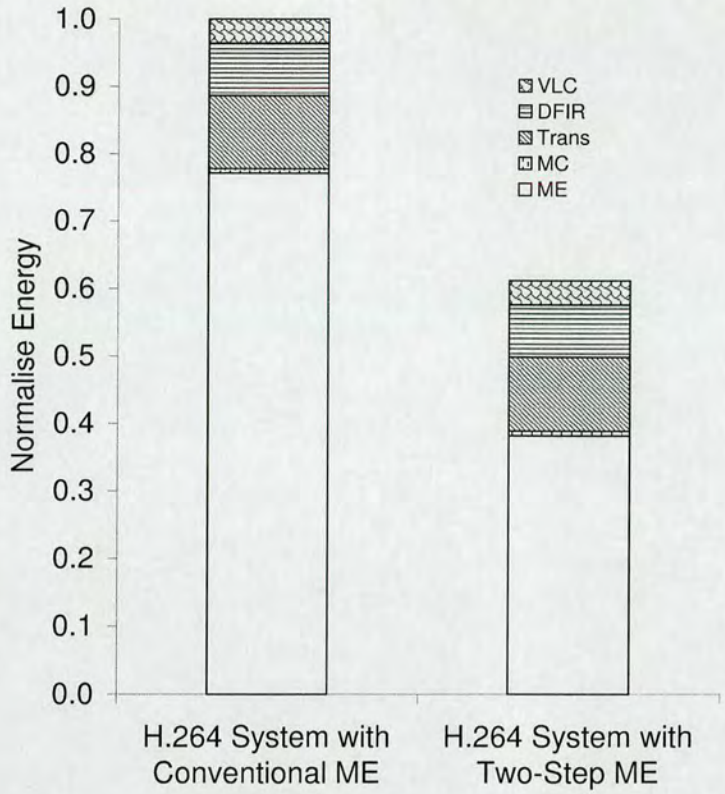


Figure 8.9: *Energy comparison of H.264 system using conventional and two-step ME using one reference frame.*

no change in energy consumption in the deblocking filter and entropy coder. In total, for one reference frame, the two-step method reduces the total energy consumption for the H.264 system by 40%, as shown in Figure 8.9. Since major power saving is achieved in the motion estimation, the total energy saving for the whole system increases as the number of reference frames increases.

8.4.3 Interframe Bus Encoding in H.264 System

Tables 8.7 and 8.8 illustrate the resources required to implement the proposed interframe bus encoder and decoder in the H.264 system. For this implementation, four pixels at a time are transferred to/from the external memory on a 32 bit bus. Thus, four encoders and decoders

Encoder Modules	Area (mm^2)	Power (mW)	
		Interframe mode	Intraframe mode
Logic	0.03	0.178	0.176
Memory	0.10	0.39	-
Total	0.13	0.568	0.176

Table 8.7: *The bus encoding area and power overhead required to implement the interframe-intraframe bus coding on a 32 bit bus at 6MHz*

Decoder Modules	Area (mm^2)	Power (mW)	
		Interframe mode	Intraframe mode
Logic	0.03	0.119	0.060
Memory	-	0.352	-
Total	0.03	0.470	0.060

Table 8.8: *The bus decoding area and power overhead required to implement the interframe-intraframe bus coding at 6MHz*

are used to encode and decode the bus. The tables show that the total area overhead required to implement the hardware is 0.16mm^2. This is equivalent to 3% of the total area in the conventional H.264.

From Table 8.7 and 8.8, it can be seen that the encoder and decoder circuits consume 0.568 and 0.470 mW of power during interframe bus coding mode, respectively, with memory dominating the circuit power overhead. Since the intraframe decorrelation does not require any memory access, the total circuit power during intraframe bus encoding and decoding mode is much lower at 0.176 and 0.060 mW, respectively.

Figure 8.10 and 8.11 show the total power consumption during interframe and intraframe bus coding compared to unencoded bus and bus coded using cluster bus invert. The total power consumption, P_T, is calculated as $P_T = P_{Circuit} + P_{CL}$, where $P_{Circuit}$ represents the power consumption due to bus encoder or decoder circuits. P_{CL} is the total bus power consumption estimated by $P_{CL} = \frac{1}{2} C_L V_{DD}^2 f \alpha$, where C_L is capacitance load, V_{DD} is operating voltage, f

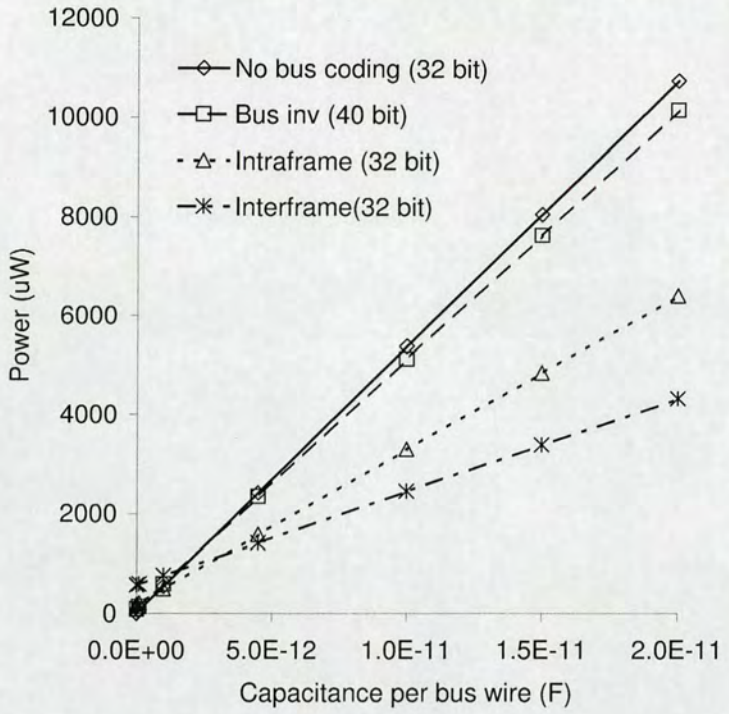


Figure 8.10: *Power consumption of 32 bit bus at 6MHz when transferring pixels into the external reference frame memory*

is operating frequency and α is switching activity. The slope of the graphs in Figure 8.10 and 8.11 is proportional to α, as discussed in Section 7.6.

For typical off-chip wire capacitance of 15pF [110], the total bus encoder power consumption during interframe mode is 3.39mW, while 4.8mW during intraframe mode. These are equivalent to 58% and 40% power savings compared to unencoded bus, respectively. In addition, the percentage of power saving is proportional to the off-chip wire capacitance. The greater power saving achieved by interframe is due to much lower transitions occurring on the busses. A smaller slope, as shown in the graphs in Figure 8.10 and 8.11 reflect this.

Table 8.9 and 8.10 show the total energy consumed by the 32bit bus when accessing the external reference frame buffer. Since loading the one search area transfers more data (512 pixels) than storing one MB data in the external reference frame buffer (256 pixels), more

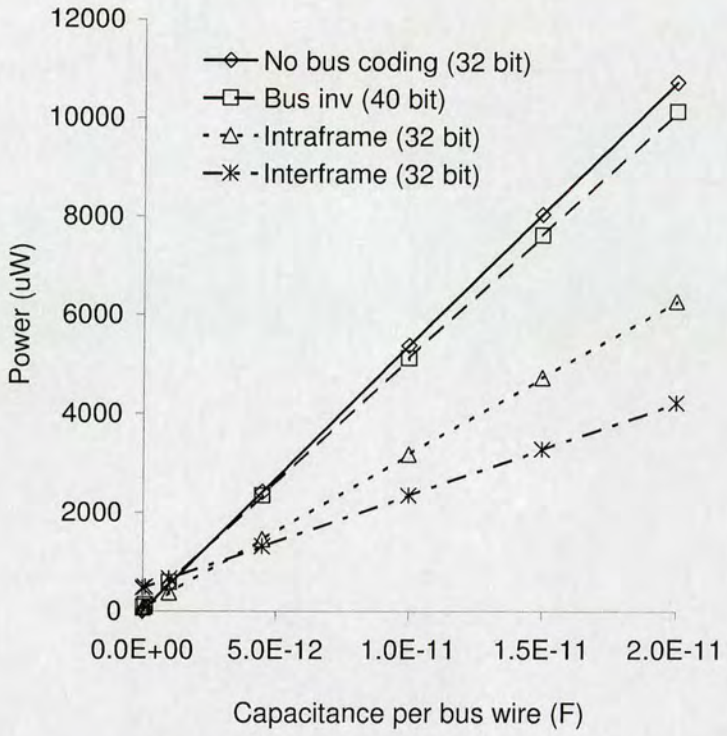


Figure 8.11: *Power consumption of 32 bit bus at 6MHz when loading pixels from the external reference frame memory*

energy is consumed during loading of the search area pixel. In addition, the data loaded from the external reference frame buffer is proportional to the number of reference frames used during motion estimation. From the tables, the interframe bus encoding saves 58% of total energy as compared to an unencoded bus.

Bus encoder	Total power (μW)	Total clock	Total energy (nJ)	Energy Saving (%)
Unencoded	8053.16	64	85.90	0
Bus Inv	7633.83	64	81.43	5
Intraframe	4846.83	64	51.70	40
Interframe	3386.60	64	36.13	58

Table 8.9: *Total energy consumption on 32bit bus at 6MHz with 15pF per bus wire when sending one filtered MB to external frame buffer*

Bus decoder	Total power (μW)	Total clock	Total energy (nJ)	Energy Saving (%)
Unencoded	8053.16	128	171.80	0
Bus Inv	7633.83	128	162.85	5
Intraframe	4730.83	128	100.92	40
Interframe	3288.60	128	70.16	58

Table 8.10: *Total energy consumption on 32-bit bus at 6MHz with 15pF per bus wire when receiving one search area from external frame buffer for one reference frame.*

	Conv. H.264 System	Low power H.264 system
Coding	695.43	425.97
Off-chip bus	257.7	106.29
Engergy	953.18	532.26

Table 8.11: *Overall energy consumption (nJ) when implement bolth the low power motion estimation and interframe bus encoding technique into the H.264 system.*

8.4.4 Overall Energy saving

Table 8.11 shows the total energy saving by combining the low power motion estimation and the interframe bus encoding technique in the system. In total, the proposed techniques save 44% of the energy compared with the conventional H.264 system. As shown in Figure 8.1, the conventional system is represented by integer motion estimation (ME), integer transform (IT, IIT, Q and IQ), deblocking filter (DFIR), and entropy coder (EC). From instruction level analysis shown in Table 3.7, these modules contribute 78% of the total H.264 computation. Thus, for a complete H.264 system, the proposed low power techniques can reduce approximately 37% of total power consumption.

Table 8.12 compares the power consumption of several H.264 systems published in the literature. In the table, specifications of baseline profile with a frame rate, 30 frames per second, are compared. The frame size determines the required macroblock to be processed per second. Thus, higher frame size requires higher clock cycle, which results in higher power

Name	[128]	[132]	[133]	Ours
Year	2006	2007	2008	2008
Specification	H.264	H.264	H.264	H.264
Profile	Baseline	Baseline	Baseline	Baseline
Frame size	D1(720×480),	CIF(352×288)	1024 x 768	QCIF(176×144)
Frame rate	30	30	30	30
Internal memory	35KB	13.3KB	NA	45KB
Operating freq	81 MHz	10MHz	100MHz	6MHz
Technology	0.18 um	0.13um	0.18 um	0.13
Power	581 mW	7mW (0.7 V)	95mW	1.28mW
$\frac{Power}{Frequency \times TotalMB}$	0.0053	0.0018	0.0003	0.0022

Table 8.12: *Comparison with other H.264 encoders*

dissipation. For a fair power comparison between different architectures, $\frac{Power}{Frequency \times TotalMB}$ is used to represent energy consumed per macroblock. As shown in the table, the proposed system shows comparable performance against other state-of-the-art implementations published in the literature. The implementation with the techniques proposed in this thesis is 60% more energy efficient per macroblock compared to the implementation in [128]. In addition, the proposed system closely matches the implementation in [132] where lower voltage is used to reduce the power consumption. The implementation in [133] consumes the least power compared to other implementations. In this implementation, it uses fast motion estimation algorithm during motion prediction where several candidates are skipped during motion prediction. However, this implementation results in degraded picture quality with reported PSNR drop by 0.8 dB.

8.5 Summary

In this chapter, the results of implementing the proposed low power techniques were presented. Based on the conventional H.264 system, the two-step ME and interframe bus encoding techniques are included to measure the effectiveness of these techniques. The results show that the two-step algorithm is able to save 40% energy over the conventional system, while the interframe bus encoding saves 58% energy when transferring search area pixels on the 32 bit bus compared to the unencoded bus.

Chapter 9
Conclusion

9.1 Introduction

This thesis presents a number of novel low-power techniques for an MPEG-4 video compression system. The power reduction is achieved by exploiting the algorithmic and architectural design methods. In this thesis, minimising the switching power between interacting modules is proposed, particularly in motion estimation and the external reference frame memory.

This chapter summarises the work presented in this thesis. The main objective of each chapter is reviewed and the goal of the research work is accessed and evaluated toward contributing to a low power MPEG-4 system. The main achievements are then highlighted and future developments that could follow on from this study are suggested.

This chapter is organised as follows. Section 9.2 discusses the conclusion of each chapter in the thesis. The main thesis contribution is highlighted in Section 9.3. Section 9.4 suggests future work and development based on the research work presented in this thesis. Finally, Section 9.5 concludes this chapter.

9.2 Thesis Conclusion and Discussion

Chapter 2 discusses the source of power consumption in the CMOS VLSI design. Since switching power consumes the greatest amount of power, various techniques have been pro-

posed to minimise this power. The problem can be tackled at different design stages such as system, algorithm, architecture and circuit. In order to assist in low power design, power estimation is crucial to calculate the power consumed by each module. This allows the main power bottleneck to be identified and low power techniques to be applied as the design progresses.

Chapter 3 presents the fundamental elements of the video coding algorithm. The standards which were developed by the two main bodies, namely MPEG-4 and VCEG, are reviewed in this section. As new standards are introduced, the complexity of the video coding algorithm increases. The essential building blocks of the MPEG-4 video coding are highlighted. Particular attention is given to the H.264 since it provides better compression compared to previous standards, but at the price of a significant increase in computational complexity. The computational load of the H.264 is discussed to justify the low-power technique for the H.264 hardware system.

Chapter 4 reviews the existing methods used to minimise the power consumed by video coding hardware. In most techniques, the video data characteristics are fully exploited to minimise the power consumption.

Chapter 5 presents an algorithm to reduce the computational load in hardware-based motion estimation. The proposed technique is performed in two steps: low-resolution and high-resolution. Compared to other low-resolution techniques, this method does not increase the memory bandwidth yet maintains good prediction accuracy for variable block size motion estimation. The simulation results shows that the proposed methods give good PSNR compared to the conventional full search with PSNR drop $< 0.5dB$.

Chapter 6 presents an energy efficient hardware for implementing the proposed two-step method. The proposed architecture is able to perform both low-resolution and high-resolution searches without significantly increasing the area overhead. With a unique pixel arrangement, the proposed method is able to perform at both low-resolution and high-resolution search, whilst still reducing the memory bandwidth. The results show that the proposed architecture is able to save up to 53% energy compared to the conventional full search.

Chapter 7 presents a low-power technique to reduce the power consumption due to high switching activity on the off-chip buses. This technique reduces the bus transition due to multiple accessing of search areas during ME. By utilising the high correlation between frames, the proposed technique is able to reduce the off-chip bus switching activity by 53%. This is equivalent to a 38% reduction of power consumption on a typical off-chip wire capacitance.

Chapter 8 presents the integration of the proposed methods into an H.264 system. The total power reduction is assessed and its interaction with other modules in the system is evaluated. The results show that the two-step algorithm is able to save 40% energy over the conventional system, while the interframe bus encoding saves 58% energy when transferring search area pixels on the 32-bit bus compared to the unencoded bus .

9.3 Summary of Achievements

This thesis has developed several algorithms and architectures that minimise the power consumption in video compression standard. The following highlights these achievements:

- A two-step algorithm for variable block size motion estimation that minimise compu-

tation and memory bandwidths during motion prediction was designed.

- Energy efficient architectures that can implement the proposed two-step motion estimation algorithm were implemented.
- An interframe bus encoding technique to minimise off-chip bus switching activity during multiple reference frame transfer was designed.
- A hardware architecture to implement the interframe bus encoding technique into H.264 system was developed.
- The proposed low power motion estimation techniques were integrated into H.264 system and the total power saving was evaluated.

9.4 Future Work

This thesis points to several areas that require further investigation:

1. An evaluation of the effectiveness of the two-step method on 1D ME architecture should be considered. In 1D ME, the memory bandwidth is higher than in 2D ME. The reduction of memory bandwidth can be achieved using the proposed method.
2. In this work, the first two MSB are used in a low resolution search. In order to improve the prediction accuracy, the selected two bits used during the ME can be varied depending on the video data characteristics.

3. Reducing the PE area in the proposed architecture should be considered, since it consumes the largest area. Techniques such as resource sharing can be explored for this purpose.

4. To further minimise the power consumption during ME, the two-step method can be combined with the existing low-power techniques, such as early termination and the successive elimination of impossible candidates.

5. To further optimise the power, an algorithm to select the best mode during the video encoding would be useful. The full search can be selected if high picture quality is required, whilst the proposed two-step algorithm can be used when lower energy is required.

6. Reducing the area required for the bus encoding implementation should be further investigated. This can be achieved by compressing the data stored in the memory.

9.5 Final Comment

Low-power is essential for today's wireless communications and consumer electronics. With the widespread use of multimedia applications and electronics devices; video compression is becoming very important. As new standards are introduced onto the market, video compression computational power tends to increase with the use of more sophisticated algorithms. This trend has been shown in previous MPEG-4 standards including H.264. While the compression efficiency is very encouraging, the high computational power will be a setback, especially for mobile devices. This thesis has proposed several solutions to minimise these

problems.

The next 10 years will see more sophisticated techniques and applications in consumer electronics. The demand from the consumer for more powerful devices and better features will force manufacturers to produce improved compression engines. This will certainly require more complex algorithms for video compression.

The works on more robust video coding are actively being researched including scalable video coding and 3D video compression. These will result in a tremendous increase in computational complexity. Thus, the pressure for low-power algorithms and architectures for video coding will be even greater in the near future.

References

[1] D. Markner and J. Nowotny, "Multimedia messaging service - the next killer application in mobile business?," *Journal of The Communications Network*, vol. 1, no. 2, pp. 236 –, 2002.

[2] M. Al-Mualla, C. N. Canagarajah, and D. R. Bull, *Video Coding for Mobile Communications: Efficiency, Complexity and Resilience*. Academic Press, 2002.

[3] G. Weinberger, "The new millennium: wireless technologies for a truly mobile society," *2000 IEEE International Solid-State Circuits Conference. Digest of TechnicalPapers*, pp. 20 – 4, 2000.

[4] "ITU-T Recommendation H.264 & ISO/IEC 14496-10 (MPEG-4) AVC "Advanced VideoCoding for Generic Audiovisual Services"," 2005.

[5] A. Chandrakasan and R. W. Brodersen, eds., *Low-Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[6] "http://www.intel.com/pressroom/kits/quickreffam.htm," May 2008.

[7] "http://www.sia-online.org/iss_technology.cfm," May 2008.

[8] C. Piguet, ed., *Low-Power Electronics Design*. CRC Press, 2005.

[9] D. Bradbury, "Seize the power," *The Guardian, Thursday January*, 5 Jan 2006.

[10] M. Mahalingam, "Thermal management in semiconductor device packaging," *Proceedings of the IEEE*, vol. 73, no. 9, pp. 1396 – 1404, 1985.

[11] W. W. Bachmann and S. A. Huss, "Efficient algorithms for multilevel power estimation of VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 2, pp. 238 – 253, 2005.

[12] H. J. M. Veendrick, "Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468 – 473, 1984.

[13] J. M. Rabaey, *Low Power Design Methodologies*. Kluwer Acaemic Publishers, 1996.

[14] S.-C. Lin and K. Banerjee, "Cool chips: Opportunities and implications for power and thermal management," *IEEE Transactions on Electron Devices*, vol. 55, no. 1, pp. 245 – 255, 2008.

[15] W. M. Elgharbawy and M. A. Bayoumi, "Leakage sources and possible solutions in nanometer cmos technologies," *IEEE Circuits and Systems Magazine*, vol. 5, no. 4, pp. 6 – 16, 2005.

[16] D. Ramanathan and R. Gupta, "System level online power management algorithms," *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000*, pp. 606 – 11, 2000.

[17] C.-H. Hwang and A. C.-H. Wu, "Predictive system shutdown method for energy saving of event-driven computation," *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, pp. 28 – 32, 1997.

[18] F. Yao, A. Demers, and S. Shenker, "Scheduling model for reduced cpu energy," *Annual Symposium on Foundations of Computer Science - Proceedings*, pp. 374 – 382, 1995.

[19] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power cpu," *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 13 – 25, 1995.

[20] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the iparm microprocessor system," *ISLPED'00: Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pp. 96 – 101, 2000.

[21] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen, "Optimizing power using transformations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12 – 31, 1995.

[22] H. Dongyan, Z. Ming, and Z. Wei, "Design methodology of CMOS low power," *IEEE International Conference on Industrial Technology, 2005 (ICIT 2005)*, 2005.

[23] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473 – 484, 1992.

[24] D. Garrett, M. Stan, and A. Dean, "Challenges in clockgating for a low power asic methodology," *Proceedings. 1999 International Symposium on Low Power Electronics and Design*, pp. 176 – 81, 1999.

[25] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 3, pp. 415 – 20, 2000.

[26] R. Zimmermann and W. Fichtner, "Low-power logic styles: Cmos versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1079 – 90, 1997.

[27] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns cmos 16×16-b multiplier using complementary pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 388 – 95, 1990.

[28] M. Borah, R. Owens, and M. Irwin, "Transistor sizing for low power cmos circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 665 – 71, 1996.

[29] F. Assad, Z. Ren, D. Vasileska, S. Datta, and M. Lundstrom, "On the performance limits for si mosfets: a theoretical study," *IEEE Transactions on Electron Devices*, vol. 47, no. 1, pp. 232 – 40, 2000.

[30] R. Isaac, "The future of cmos technology," *IBM Journal of Research and Development*, vol. 44, no. 3, pp. 369 – 78, 2000.

[31] M. E. I. Abdellatif Bellaouar, ed., *Low-powerower igital VLSI design*. Kluwer Academic Publishers, 1995.

[32] M. Nemani and F. Najm, "Towards a high-level power estimation capability [digital ics]," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 588 – 98, 1996.

[33] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 11, pp. 1061 – 79, 1998.

[34] D. Liu and C. Svensson, "Power consumption estimation in cmos VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663 – 670, 1994.

[35] S. R. Powell and P. M. Chau, "A model for estimating power dissipation in a class of dsp VLSI chips," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 6, pp. 646 – 650, 1991.

[36] J. Frenkil, "Tools and methodologies for low power design," *Proceedings 1997. Design Automation Conference, 34th DAC*, pp. 76 – 81, 1997.

[37] D. Singh, J. M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. J. Mozdzen, "Power conscious CAD tools and methodologies: a perspective," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 570 – 594, 1995.

[38] B. George, G. Yeap, M. Wloka, S. Tyler, and D. Gossain, "Power analysis for semi-custom design," *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, pp. 249 – 52, 1994.

[39] *0.13um High Density Standard Cell Library Databook (Process name: UMC130E HS-FSG, Part Number: UMCE13H210T3) from United Microelectronics Corporation*, Aug 2003.

[40] http://www.synopsys.com/news/pubs/compiler/, "Oki Techno Centre Design Team Achieves Lowest Power Consumption Using Power Compiler," 2008.

[41] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Prentice Hall, 2002.

[42] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. John Wiley & Sons, 2003.

[43] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. Institution of Engineering and Technology, 2003.

[44] P. M. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 MotionEstimation (1st edition)*. Springer, 1999.

[45] A. Tourapis, O. Au, and M. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 934 – 47, 2002.

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

[47] K. R. Rao and P. Yip, *Discrete Cosine Transform*. Academic Press, 1990.

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

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

[50] "http://ftp3.itu.ch/av-arch/jvt-site/," 2008.

[51] http://iphome.hhi.de/suehring/tml/, "H.264 Reference Software," 31 May 2008.

[52] A. Tamhankar and K. Rao, "An overview of H.264/MPEG-4 Part 10," *Proceedings EC-VIP-MC 2003. 4th EURASIP Conference focused on Video/ImageProcessing and Multimedia Communications*, vol. vol.1, pp. 1 – 51, 2003.

[53] G. J. Sullivan, P. Topiwala, and A. Luthra, "The h.264/avc advanced video coding standard: Overview and introductionto the fidelity range extensions," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5558, no. PART 1, pp. 454 – 474, 2004.

[54] S.-Y. Chien, Y.-W. Huang, C.-Y. Chen, H. H. Chen, and L.-G. Chen, "Hardware architecture design of video compression for multimedia communication systems," *IEEE Communications Magazine*, vol. 43, no. 8, pp. 123 – 131, 2005.

[55] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560 – 576, 2003.

[56] S. Saponara, K. Denolf, G. Lafruit, C. Blanch, and J. Bormans, "Performance and complexity co-evaluation of the advanced video coding standard for cost-effective multimedia communications," *Eurasip Journal on Applied Signal Processing*, vol. 2004, no. 2, pp. 220 – 235, 2004.

[57] H.-C. Chang, Y.-C. Chang, Y.-C. Wang, W.-M. Chao, and L.-G. Chen, "VLSI architecture design of MPEG-4 shape coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 9, pp. 741 – 51, 2002.

[58] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849 – 66, 1998.

[59] P. M. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3309, no. 1, pp. 498 – 509, 1998.

[60] "Recommendation ITU-R BT.500 (revised): 'Methodology for the subjective assessments of the quality of television pictures," tech. rep.

[61] F. Pereira and T. Alpert, "MPEG-4 video subjective test procedures and results," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 32 – 51, Feb. 1997.

[62] A. Katsaggelos, F. Zhai, Y. Eisenberg, and R. Berry, "Energy-efficient wireless video coding and delivery," *IEEE Wireless Communications*, vol. 12, pp. 24 – 30, 2005.

[63] Z. He, Y. Liang, and I. Ahmad, "Power-rate-distortion analysis for wireless video communication under energyconstraint," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5308, no. PART 1, pp. 57 – 68, 2004.

[64] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," *NTC '81. IEEE 1981 National Telecommunications Conference. Innovative Telecommunications - Key to the Future*, pp. 5 – 3, 1981.

[65] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. CM-29, no. 12, pp. 1799 – 1808, 1981.

[66] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438 – 42, 1994.

[67] J. Y. Tham, S. Ranganath, M. Ranganath, and A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369 – 77, 1998.

[68] T. Zahariadis and D. Kalivas, "A spiral search algorithm for fast estimation of block motion vectors," *Signal Processing VIII, Theories and Applications. Proceedings of EUSIPCO-96, Eighth European Signal Processing Conference*, vol. vol.2, pp. 1079 – 82, 1996.

[69] T.-H. Han and S. H. Hwang, "A novel hierarchical-search block matching algorithm and VLSI architecture considering the spatial complexity of the macroblock," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 2, pp. 337 – 42, 1998.

[70] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," *Signal Processing*, vol. 11, no. 4, pp. 387 – 404, 1986.

[71] P. Hosur, "Motion adaptive search for fast motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1330 – 40, 2003.

[72] A. Tourapis, O. Au, and M. Liou, "Predictive motion vector field adaptive search technique (PMVFAST)- enhancing block-based motion estimation," *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 4310, pp. 883 – 92, 2000.

[73] J.-N. Kim, S.-C. Byun, Y.-H. Kim, and B.-H. Ahn, "Fast full search motion estimation algorithm using early detection of impossible candidate vectors," *IEEE Transactions on Signal Processing*, vol. 50, no. 9, pp. 2355 – 65, 2002.

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

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

[76] K. Kim, P. A. Beerel, and Y. Hong, "Asynchronous matrix-vector multiplier for discrete cosine transform," *Proceedings of the International Symposium on Low Power Design*, pp. 256 – 261, 2000.

[77] T. Darwish and M. Bayoumi, "Energy aware distributed arithmetic dct architectures," *2003 IEEE Workshop on Signal Processing Systems*, pp. 351 – 6, 2003.

[78] C. De Vleeschouwer, "Model-based rate control implementation for low-power video communications systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 12, pp. 1187 – 1194, 2003.

[79] A. Laffely, J. Liang, P. Jain, W. Burleson, and R. Tessier, "Adaptive systems on a chip (aSoC) for low-power signal processing," *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. vol.2, pp. 1217 – 21, 2001.

[80] O. S. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1055 – 1069, 2003.

[81] Y. Liang and I. Ahmad, "Power and content aware video encoding for video communication over wireless networks," *2004 IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 269 – 74, 2004.

[82] W. Burleson, P. Jain, and S. Venkatraman, "Dynamically parameterized architectures for power-aware video coding: motion estimation and dct," *Proceedings Second International Workshop on Digital and Computational Video*, pp. 4 – 12, 2001.

[83] Y. Liang, Z. He, and I. Ahmad, "Analysis and design of power constrained video encoder," *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, pp. 57 – 60, 2004.

[84] T.-H. Lan and A. Tewfik, "Power optimized mode selection for h.263 video coding and wireless communications," *Proceedings 1998 International Conference on Image Processing. ICIP98*, vol. vol.2, pp. 113 – 17, 1998.

[85] T. Fujiyoshi, S. Shiratake, S. Nomura, T. Nishikawa, Y. Kitasho, H. Arakida, Y. Okuda, Y. Tsuboi, M. Hamada, H. Hara, T. Fujita, F. Hatori, T. Shimazawa, K. Yahagi, H. Takeda, M. Murakata, F. Minami, N. Kawabe, T. Kitahara, K. Seta, M. Takahashi, and Y. Oowaki, "An h.264/MPEG-4 audio/visual codec lsi with module-wise dynamic voltage/frequency scaling," *2005 IEEE International Solid-State Circuits Conference*, vol. Vol. 1, pp. 132 – 589, 2005.

[86] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," *Proceedings of the ACM International Multimedia Conference and Exhibition*, pp. 582 – 591, 2003.

[87] A. Portero, G. Talavera, M. Monton, B. Martinez, F. Cathoor, and J. Carabina, "Dynamic voltage scaling for power efficient MPEG4-sp implementation," *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, pp. 257 – 260, 2006.

[88] J. Diguet, S. Wuytack, F. Catthoor, and H. De Man, "Formalized methodology for data reuse exploration in hierarchical memorymappings," *Proceedings of the International*

Symposium on Low Power Electronics andDesign, Digest of Technical Papers, pp. 30 – 35, 1997.

[89] K. Denolf, C. De Vleeschouwer, R. Turney, G. Lafruit, and J. Bormans, "Memory centric design of an MPEG-4 video encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 609 – 19, 2005.

[90] 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 Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61 – 72, 2002.

[91] L. Nachtergaele, F. Catthoor, B. Kapoor, S. Janssens, and D. Moolenaar, "Low-power data transfer and storage exploration for h.263 video decoder system," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 1, pp. 120 – 129, 1998.

[92] V. Moshnyaga, K. Inoue, and M. Fukagawa, "Reducing energy consumption of video memory by bit-width compression," *ISLPED'02: Proceedings of the 2002 International Symposium on Lower PowerElectronics and Design*, pp. 142 – 7, 2002.

[93] S.-C. Hsia, "Efficient memory ip design for hdtv coding applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 465 – 71, 2003.

[94] A. Hatabu, T. Miyazaki, and I. Kuroda, "Qvga/cif resolution MPEG-4 video codec based on a low-power and general-purpose dsp," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 39, no. 1-2 SPECISS, pp. 7 – 14, 2005.

[95] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, Tung-Chienand Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimationfor h.264/avc," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 3, pp. 578 – 593, 2006.

[96] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 702 – 6, Aug 1997.

[97] A. Erturk and S. Erturk, "Two-bit transform for binary block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 7, pp. 938 – 46, July 2005.

[98] S. Lee, J.-M. Kim, and S.-I. Chae, "New motion estimation algorithm using adaptively quantized low bit-resolutionimage and its VLSI architecture for MPEG2 video encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 6, pp. 734 – 44, Oct 1998.

[99] Y. Chan and S. Kung, "Multi-level pixel difference classification methods," *IEEE International Conference on Image Processing*, vol. 3, pp. 252 – 255, 1995.

[100] 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 Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 669 – 678, 2000.

[101] V. G. Moshnyaga, "Msb truncation scheme for low-power video processors," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 291–294 –, 1999.

[102] M.-J. Chen, L.-G. Chen, T.-D. Chiueh, and Y.-P. Lee, "A new block-matching criterion for motion estimation and its implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 3, pp. 231 – 6, June 1995.

[103] S. Sharma, P. Mishra, S. Sawant, C. Mammen, and V. Gadre, "Pre-decision strategy for coded/non-coded mbs in MPEG4," *2004 International Conference on Signal Processing and Communications (SPCOM)*, pp. 501 – 5, 2004.

[104] H. Yeo and Y. H. Hu, "A novel architecture and processor-level design based on a new matchingcriterion for video compression," *VLSI Signal Processing, IX*, pp. 448 – 57, 1996.

[105] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 42, no. 3, pp. 297 – 320, 2006.

[106] T.-C. Chen, Y.-H. Chen, S.-F. Tsai, and S. Y. C. andLiang Gee Chen, "Fast algorithm and architecture design of low-power integer motion estimationfor h.264/avc," *IEEE Trans. Circuits Syst. Video Technol. (USA)*, vol. 17, no. 5, pp. 568 – 77, 2007.

[107] K. K. Parhi, *VLSI Design Signal Processing Systems: Design and Implementation*. John Wiley & Sons, Inc., 1999.

[108] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1301 – 1308, 1989.

[109] C.-H. Lin, C.-M. Chen, and C.-W. Jen, "Low power design for MPEG-2 video decoder," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 3, pp. 513 – 521, 1996.

[110] W.-C. Cheng and M. Pedram, "Chromatic encoding: a low power encoding technique for digital visual interface," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 320 – 8, Feb. 2004.

[111] T. Givargis and F. Vahid, "Interface exploration for reduced power in core-based systems," *Proceedings. 11th International Symposium on System Synthesis*, pp. 117 – 22, 1998.

[112] L. Benini and G. de Micheli, "System-level power optimization: techniques and tools," *Proceedings. 1999 International Symposium on Low Power Electronics and Design*, pp. 288 – 93, 1999.

[113] M. Stan and W. Burleson, "Bus-invert coding for low-power i/o," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 1, pp. 49 – 58, Mar. 1995.

[114] W.-C. Cheng and M. Pedram, "Memory bus encoding for low power: a tutorial," *Proceedings of the IEEE 2001. 2nd International Symposium on Quality Electronic Design*, pp. 199 – 204, 2001.

[115] H. Mehta, R. Owens, and M. Irwin, "Some issues in gray code addressing," *Proceedings. The Sixth Great Lakes Symposium on VLSI*, pp. 178 – 81, 1996.

[116] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-powermicroprocessor-based systems," *Proceedings Great Lakes Symposium on VLSI*, pp. 77 – 82, 1997.

[117] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano, "Power optimization of system-level address buses based on software profiling," *Proceedings of the Eighth International Workshop on Hardware/Software Codesign.CODES 2000*, pp. 29 – 33, 2000.

[118] Y. Shin, S.-I. Chae, and K. Choi, "Partial bus-invert coding for power optimization of system level bus," *Proceedings of the International Symposium on Low Power Design*, pp. 127 – 129, 1998.

[119] S. Komatsu, M. Ikeda, and K. Asada, "Low power chip interface based on bus data encoding with adaptive code-bookmethod," *Proceedings Ninth Great Lakes Symposium on VLSI*, pp. 368 – 71, 1999.

[120] S. Ramprasad, N. Shanbhag, and I. Hajj, "A coding framework for low-power address and data busses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 2, pp. 212 – 21, June 1999.

[121] L. Benini, A. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power-efficient bus interfaces," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 969 – 80, Sept. 2000.

[122] S. Osborne, A. Erdogan, T. Arslan, and D. Robinson, "Bus encoding architecture for low-power implementation of an amba-based soc platform," *IEE Proceedings-Computers and Digital Techniques*, vol. 149, no. 4, pp. 152 – 6, 2002.

[123] R. Siegmund, C. Kretzschmar, and D. Muller, "Adaptive partial businvert encoding for power efficient data transfer over wide system buses," (Manaus, Brazil), pp. 371 – 6, 2000.

[124] T.-C. Wang, Y.-W. Huang, H.-C. Fang, and L.-G. Chen, "Parallel 4*4 2d transform and inverse transform architecture for MPEG-4avc/h.264," *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems*, vol. vol.2, pp. 800 – 3, 2003.

[125] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C. Wang, Te-HaoChang, and L.-G. Chen, "Architecture design for deblocking filter in h.264/jvt/avc," *Proceedings 2003 International Conference on Multimedia and Expo*, vol. vol.1, pp. 693 – 6, 2003.

[126] S.-C. Chang, W.-H. Peng, S.-H. Wang, and T. Chiang, "A platform based bus-interleaved architecture for de-blocking filter inh.264/MPEG-4 avc," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 1, pp. 249 – 55, 2005.

[127] T.-C. Chen, Y.-W. Huang, C.-Y. Tsai, B.-Y. Hsieh, and L.-G. Chen, "Dual-block-pipelined vlsi architecture of entropy coding for h.264/avc baseline profile," vol. 2005, (Hsinchu, Taiwan), pp. 271 – 274, 2005. Context-based adaptive variable length coding (CAVLC);Coding engine;Dual-block-pipelined architecture;Network abstraction layers;.

[128] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, "Analysis and architecture design of an hdtv720p 30 frames/s h.264/avc encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 6, pp. 673 – 688, 2006.

[129] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in h.264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598 – 603, 2003.

[130] A. Puri, X. Chen, and A. Luthra, "Video coding using the h.264/mpeg-4 avc compression standard," *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 793 – 849, 2004.

[131] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620 – 36, 2003.

[132] H.-C. Chang, J.-W. Chen, C.-L. Su, Y.-C. Yang, Y. Li, C.-H. Chang, Z.-M. Chen, W.-S. Yang, C.-C. Lin, C.-W. Chen, J.-S. Wang, and J.-I. Guo, "A 7mw-to-183mw dynamic quality-scalable h.264 video encoder chip," (San Francisco, CA, United States), pp. 280 – 281, 2007.

[133] K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, and N. Vlassopoulos, "A real-time h.264/avc vlsi encoder architecture," *Journal of Real-Time Image Processing*, vol. 3, no. 1-2, pp. 43 – 59, 2008.