Low-Power and Application-Specific SRAM Design for Energy-Efficient Motion Estimation

by

Mahmut Ersin Sinangil

B.S. in Electrical Engineering, Bogazici University, 2006
S.M. in Electrical Engineering, Massachusetts Institute of Technology, 2008

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2012

Certified by .....
Anantha P. Chandrakasan
Joseph F. and Nancy P. Keithley Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Low-Power and Application-Specific SRAM Design for Energy-Efficient Motion Estimation

by

Mahmut Ersin Sinangil

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Video content is expected to account for 70% of total mobile data traffic in 2015. High efficiency video coding, in this context, is crucial for lowering the transmission and storage costs for portable electronics. However, modern video coding standards impose a large hardware complexity. Hence, energy-efficiency of these hardware blocks is becoming more critical than ever before for mobile devices. SRAMs are critical components in almost all SoCs affecting the overall energy-efficiency. This thesis focuses on algorithm and architecture development as well as low-power and application-specific SRAM design targeting motion estimation.

First, a motion estimation design is considered for the next generation video standard, HEVC. Hardware cost and coding efficiency trade-offs are quantified and an optimum design choice between hardware complexity and coding efficiency is proposed. Hardware-efficient search algorithm, shared search range across CU engines and pixel pre-fetching algorithms provide 4.3× area, 56× on-chip bandwidth and 151× off-chip bandwidth reduction.

Second, a highly-parallel motion estimation design targeting ultra-low voltage operation and supporting AVC/H.264 and VC-1 standards are considered. Hardware reconfigurability along with frame and macro-block parallel processing are implemented for this engine to maximize hardware sharing between multiple standards and to meet throughput constraints.

Third, in the context of low-power SRAMs, a 6T and an 8T SRAM are designed in 28nm and 45nm CMOS technologies targeting low voltage operation. The 6T design achieves operation down to 0.6V and the 8T design achieves operation down to 0.5V providing $\sim 2.8\times$ and $\sim 4.8\times$ reduction in energy/access respectively.

Finally, an application-specific SRAM design targeted for motion estimation is developed. Utilizing the correlation of pixel data to reduce bit-line switching activity, this SRAM achieves up to 1.9× energy savings compared to a similar conventional 8T design. These savings demonstrate that application-specific SRAM design can introduce a new dimension and can be combined with voltage scaling to maximize energy-efficiency.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Joseph F. and Nancy P. Keithley Professor of Electrical Engineering

Acknowledgments

First, I would like to thank Professor Anantha Chandrakasan for his continuous support and guidance during the past six years. I am very grateful to Anantha for giving me this opportunity and making me a part of his group. He always expected more from me and challenged me to be more successful by creating exciting research projects and by motivating discussions. I have learned things much beyond circuits from him. His advices taught me to be an honest, critical, cooperative and respectful researcher. Thank you, Anantha, for your mentorship and support.

I would like to acknowledge Professor Jae Lim and Professor Li-Shiuan Peh for serving on my thesis committee and providing me insightful feedback on my research. Your valuable advices during our meetings have helped me tremendously to improve this thesis.

I would like to thank Texas Instruments for generously providing chip fabrication and funding for this research. It was a privilege to cooperate and interact with many engineers at TI during summer internships and many visits. I would like to especially thank Alice Wang, Dennis Buss, Minhua Zhou, Mike Polley and Hugh Mair for valuable discussions. I would also like to thank Vivienne Sze for not only the technical discussions and help on HEVC motion estimation project but also for her friendship.

I was fortunate to be a part of a large research group so that I got to know many people who were very helpful and cooperative and also had a great sense of humor. First, I would like to thank Margaret for her assistance and for her help with many problems I had over the years. I was fortunate to collaborate with many students for various projects. I would like to thank Naveen for helping me to ramp up on SRAMs and get used to graduate school life and research. His brilliance in circuit design has motivated me to push myself to work harder. During the video decoder project, I enjoyed working with Daniel and Vivienne a lot. They are both outstanding researchers and also great friends. I would like to thank Nathan for his collaboration not only during the Jupiter project but also sharing his expertise on many occasions.

I find myself fortunate to have worked with Chih-Chi, Rahul, Bonnie, Sabrina, Eric, Hank and Jason on different projects.

I started this journey at the same time with Patrick and Masood. Pat and his wife Chelce has been great friends. Yildiz and I very much enjoyed our movie nights, ski trips and Worms battles (especially the flying sheep) together and we will miss you. I would like to thank Masood for daily discussions covering a wide range of subjects and for sharing his interesting ideas with me. I think we have learned a lot from each other over these years.

I would also like to thank former and current students in *ananthagroup* especially Joyce, Yogesh, Marcus, Saurav, Arun, Phil, Mehul, Nachiket and Dina. I would also like to thank Cagatay and Selami who have been my best friends since high-school. I was fortunate to have them in the US during my studies.

Finally, I am very thankful to my family for continuously supporting and loving me. Although you were thousands of miles away, amazingly, you managed to continue to take care of me through phone and video calls. I would also like to thank Yildiz's parents for their support and love.

Lastly, I would like to thank my beautiful wife who has been with me throughout this journey. I am blessed to have met Yildiz, a successful and motivated electrical engineer and a loving wife to share my life with. Her continuous support and unconditional love have encouraged me to try harder.

Contents

List of Figures

List of Tables

Acronyms

SRAM Static Random Access Memories

IC integrated circuits

HEVC High-Efficiency Video Coding

JCT-VC joint-collaborative team on video coding

SAD sum-of-absolute-differences

AD absolute-differences

MV motion vector

TSS three-step-search

CS Cross-search

FSS four-step-search

PMRME parallel multi-resolution motion estimation

IME Integer motion estimation

FME Fractional motion estimation

SoC system-on-chip

DIBL drain induced barrier lowering

6T six-transistor

8T eight-transistor

MSB most-significant-bits

LSB least-significant-bits

PB-RBSA Prediction-based reduced bit-line switching activity

CCE cost and coding efficient

AMVP advanced motion vector prediction

MVP motion vector predictor

LCU largest coding unit

SCU smallest coding unit

CU coding unit

PU prediction unit

AMP asymmetric motion partitions

MB macro-block

M-SA main sense-amplifiers

E-SA Early decision sense-amplifier

SNM Static Noise Margin

Chapter 1

Introduction

Continuous scaling of process technologies driven by Moore's law [1] has resulted in integrating more transistors and more functionality on a single chip. This advancement has led to a wide range of new applications including mobile multimedia, biomedical monitoring and wireless sensor nodes. In these applications, the source of available energy is generally in the form of a battery and computationally demanding applications can cause the battery to be depleted quickly. However, very frequent charge cycles (i.e. amount of time between pre-charging of a battery) can cause these applications to be impractical. Hence, energy efficiency of circuits that are running from a limited energy supply is an important requirement. Moreover, energy harvesting is an emerging area that can lead to a very wide range of new applications and revolutionize the world. However, power harvested from the ambient is generally on the orders of micro-watts [2], requiring the circuits to be extremely energy efficient.

Video is becoming an indispensable feature for consumers on portable devices. With the integration of better optical lenses and more processing power, today's mobile devices (e.g. iPhone 4S) are capable of recording video at resolutions as high as 1920×1080 (full-HD) at a frame rate of 30fps [3]. In order to reduce storage memory requirements and the bandwidth to transfer video content, video encoding is performed during recording. Although recent video standards have provided significant improvement in compression efficiency, this comes at the expense of increased computational complexity and consequently increased power consumption resulting

in shorter battery life. Hence, portable devices can benefit significantly from energy-efficient implementations of video codecs.

Static Random Access Memories (SRAM) are the most common type of embedded memories and one of the most critical building blocks in modern system-on-chip (SoC) designs since a larger fraction of chip area is allocated for on-chip memories in modern integrated circuits (IC). SRAMs possess two important features that make them amenable to implementation in deep sub-micron processes [4]. First, SRAMs have a very high transistor density. This is due to the fact that the regular structure of memories allows photo-lithographic rules to be more aggressive. Secondly, despite the high transistor density, memories have a relatively low activity factor as only a small portion of a memory (e.g. a single row of data) is accessed every cycle. Hence, the power density of SRAMs is smaller than logic. As a result of these advantageous features, the work in [4] estimated that the amount of embedded memories will continuously increase over the years. This has actually been true over the course of the years as shown in Figure 1-1 [5] and in today's micro-processors, the total size of on-chip caches can be as high as 54MB on a single die [6]. Consequently, SRAMs account for a large fraction of the total power consumption of most SoCs.

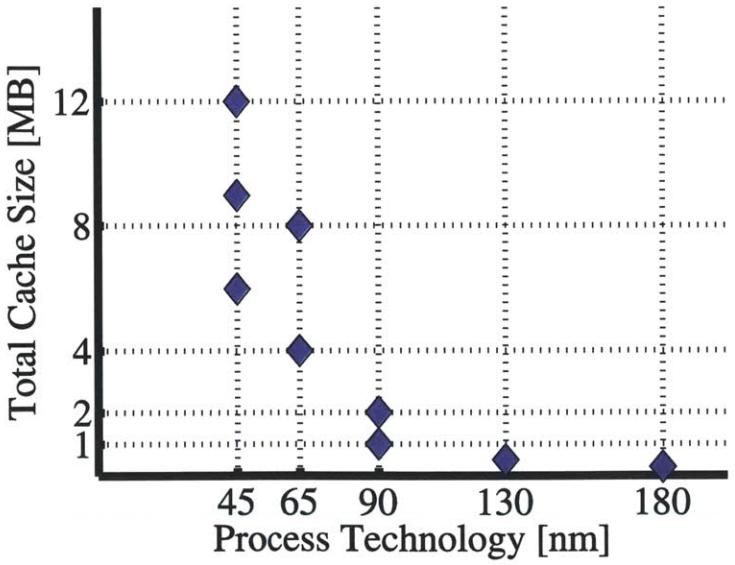


Figure 1-1: Total cache size in Intel processors for different process nodes.

One of the most important and power consuming tools in video encoding is motion

estimation. Through motion estimation, temporal redundancy in video data can be extracted and coding efficiency can be improved. Motion estimation requires a large number of computations and a large size of data to be stored on-chip for subsequent accesses. Hence, the motion estimation process results in a large activity on the chip and utilizes large blocks of SRAMs. These SRAMs are frequently accessed and accounts for a large portion of power consumption.

In this thesis, energy efficiency of a hardware block is considered at three different levels of the design: starting at algorithms, then at the architectural level and finally at transistor-level circuits. For motion estimation, this thesis shows that low-power algorithms and architectures that are suitable for hardware implementation can significantly reduce power consumption at the expense of a small loss in coding efficiency. Moreover, low-power SRAM designs employing circuit techniques for low voltage operation as well as application-specific SRAM design are considered in this thesis as means to improve energy-efficiency at the circuit level.

This chapter begins by briefly explaining video coding standards and motion estimation in Section 1.1. Then Section 1.2 talks about voltage scaling which is an effective method of reducing energy consumption of circuits. Finally, challenges associated with low voltage and low-power SRAM design and previous work in this area will be discussed in Section 1.3 and Section 1.4 respectively.

1.1 Recent Video Compression Standards and Motion Estimation

1.1.1 Recent Video Compression Standards

During the past decade, the amount of video content available on the Internet has grown significantly because of websites like YouTube [7] that allow users to share videos. With the introduction of 3G/4G mobile broadband technology, consumers now access the Internet through portable devices and mobile networks. Figure 1-2 plots monthly global mobile data traffic forecasts for the next three years [8]. First, the

mobile data traffic is increasing at a rapid rate and the global data traffic is expected to increase by 4× from 2012 to 2015. Secondly and more importantly, this increase is mainly attributed to the significant increase of the video content. Specifically, in 2015, video is expected to account for 70% of the total mobile data traffic. With the introduction of portable devices supporting video capture at higher resolutions and frame-rates, the weight of video can be expected to continue to increase in the next decade. Video coding, in this context, is crucial for lowering the transmission costs and data storage for portable electronics.

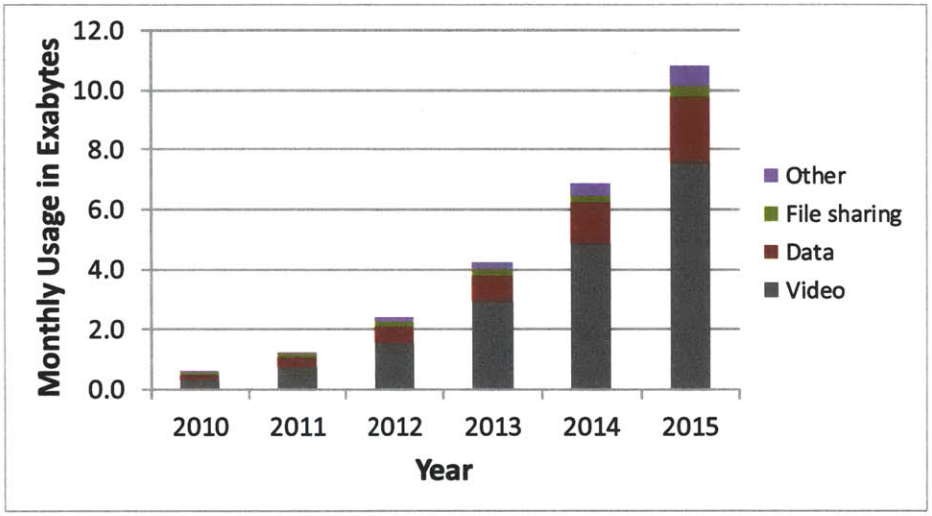


Figure 1-2: Monthly global mobile data traffic forecast. Traffic is given in exabytes (1 Exabyte=10^6TB).

Recent video standards such as AVC/H.264 and VC-1 provided significant coding efficiency gain over the previous standards (50% over MPEG-2 [9]) and is widely used in mobile devices [3]. However, the increase in efficiency is due to more complex algorithms which result in elevated complexity and power consumption in hardware implementation. For example, AVC/H.264 has 4× more hardware complexity with respect to MPEG-2 [9]. Figure 1-3 shows the change of relative complexity of a video processing core in a mobile processor with respect to a video core supporting MPEG4 standard with QCIF resolution and 30fps frame rate. Over the years, video core is adapted to meet increased demands [10]. This figure reflects the increased complexity due to

- more advanced video coding standards and

- the necessity to employ a more dedicated hardware for video coding to meet performance requirements.

By the year 2020, the complexity of a video core is expected to be 10× larger than today's demands. This is especially problematic for battery-operated portable devices. Consequently, the implementation of video cores in hardware is very critical for the overall power consumption of battery-operated applications.

Energy-constrained systems require careful and joint selection of solutions at the algorithm, architecture and circuit levels. In other words, specific algorithms should be implemented with suitable architectures and circuits for maximum energy savings [11]. For example, a low-power algorithm can be designed to reduce the number of computations and memory accesses. Then this algorithm can be implemented in a parallelizable architecture enabling voltage scaling. Furthermore, at the circuit level, specific topologies for logic and memories can be chosen to further reduce switching activity and energy/access numbers. Hence, algorithms, architectures and circuits should be considered jointly and trade-offs associated with hardware implementation should also be included in the algorithm development stage.

High-Efficiency Video Coding (HEVC) is a new video compression standard being standardized by the joint-collaborative team on video coding (JCT-VC) [12]. HEVC has a design goal of achieving a 50% coding efficiency gain over AVC/H.264 High Profile. For this purpose, several coding efficiency enhancement tools have been proposed for this new standard.

During the adoption process of the proposed tools, hardware implementation costs in terms of area, power consumption and data bandwidth are also considered. Hence, algorithms are evaluated not only based on their coding efficiency performance but also by taking their hardware implementability into account. This allows circuit designers to be involved in the definition process of the standard and help with the co-optimization of algorithms, architectures and circuits discussed above.

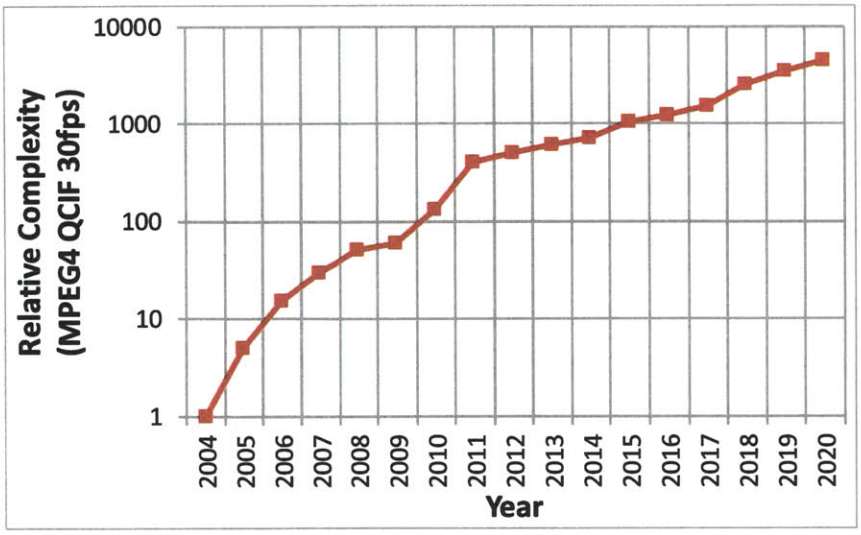


Figure 1-3: Relative complexity of video core over the years for a mobile applications processor. From 2012 to 2020, video core complexity is expected to increase by 10×.

1.1.2 Motion Estimation in Recent Video Compression Standards

Motion estimation is the process of determining the movement of objects within a sequence of image frames [13]. Through motion estimation , temporal redundancy between frames is extracted and a group of pixels is represented by its displacement with respect to a previously coded frame plus some possible residue. This results in significant reduction of bits to represent a group of pixels and hence provide significant coding efficiency improvement.

Finding the movement of objects involves a search performed on a pre-defined searching range as shown in Figure 1-4. In recent video standards, block-based motion estimation is used where a frame is divided into blocks of pixels and motion search is performed for these blocks. Motion search can be done exhaustively by checking every possible candidate or it can be done with a faster algorithm which goes through a smaller subset of possible candidates. In hardware implementations , sum-of-absolute-differences (SAD) of co-located pixels between the current block of pixels ($c_{i,j}$) and the candidate block's pixels ($r_{i,j}$) is used as the cost metric. For a block of $N \times N$ pixels, SAD is given by:

$$\mathrm{SAD} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c_{i,j} - r_{i,j}| \tag{1.1}$$

Based on this metric, a "best-match" is determined and the displacement of the best-matching block from the current block is coded as the motion vector (MV).

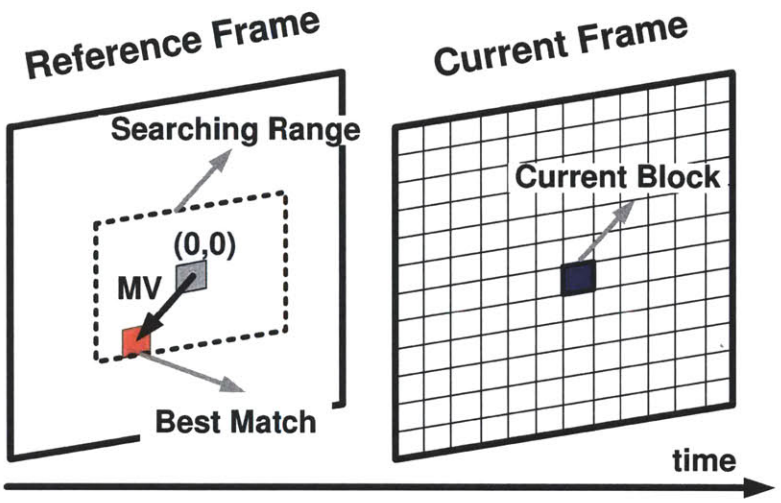


Figure 1-4: Block-based motion estimation. Best-matching block for the current block is searched within a searching range and the displacement of the "best-match" is given as the MV.

In hardware implementations, fast search algorithms are widely used. These algorithms are extremely critical for the complexity and the area of hardware, its power consumption and lastly its memory bandwidth requirement. Moreover, the search algorithm's performance also affects the coding efficiency depending on how accurately the algorithm finds the best possible MV. In recent publications, more elaborate search algorithms are also proposed. An overview of previous work on fast search algorithms will be provided in Section 1.1.3.

1.1.3 Previous Work on Fast Search Algorithms

Since the introduction of block-based motion estimation in video standards, fast search algorithms have been widely studied. Although exhaustive search is guaranteed to find a global minimum on a pre-defined set of points, its complexity increases quadratically with the size of the search window.

Fast Search Method	Number of Search Points
Exhaustive Search	$(2 \times SR + 1)^2$
[14]	$1 + 8 \times log_2 SR$
[15]	$9 + 8 \times log_2 SR$
[16]	$2 + 7 \times log_2 SR$
[17]	$1 + 8 \times log_2 SR$
[18]	$3 + 2 \times SR$
[22]	$5 + 4 \times log_2 SR$
[19]	$17 + 5 \times log_2 \frac{SR}{2}$

Table 1.1: Comparison of various fast search algorithms and number of search points used in them for a search window of $\pm SR$.

One of the earliest and most famous search algorithms is three-step-search (TSS) [14]. In TSS, 8 positions along the edges of a square surrounding the search center are checked and compared against the search center. In every step, side length of the square is halved and search center is moved to the best position from the previous step. For ± 7 search area, this algorithm finishes in three steps.

Although TSS is simple and effective, it lacks a bias for the initial starting point. "New" TSS algorithm [15] addresses this issue and provides a center bias of the initial search center. The work in [16] provides a logarithmic search pattern whereas works in [17] and [18] use directional methods to reduce the number of computations. Cross-search (CS) is another example which uses a cross-shaped search pattern in its steps. A four-step-search (FSS) is proposed in [19] where step sizes stay the same unless the search center is the best position. Otherwise, it is halved. Lastly, the works in [20] and [21] are also widely-used fast search algorithms using diamond and hexagonal search patterns.

Recent work focused on efficiency of hardware implementation of the search algorithm. Supporting higher video resolutions up to Quad-HD (3840×2160) resulted in the active power becoming one of the most pronounced problems in video encoders. Hence, computational complexity and parallelizable implementation for a throughput constraint are becoming important factors in the selection of fast search algorithms.

Moreover, variable block sizes supported by recent standards require calculation of MVs for sub-block partitions. So algorithm considerations on efficient implementability for variable block sizes are also important [23, 24].

Table 1.1 demonstrates the number of search points for different search algorithms [22]. In terms of coding efficiency, these algorithms provide similar results. Although the number of search points provided in this table is an important metric to quantify complexity, it cannot provide a good estimation alone. For example, there is a strong dependency between the outcome of each step and its next step in TSS. Although the number of search points is small, this algorithm cannot be easily parallelized because of the dependency between steps. Hence, when compared to a hardware-efficient search algorithm checking the same number of search points with the same throughput constraint, TSS requires

- a higher operating frequency for the same hardware area or
- a larger area for the same frequency of operation.

A recent example is the work in [25] which presents an algorithm and suitable architecture for hardware implementation of predict hexagon search and claims nearly $4\times$ reduction in the frequency of operations for the same throughput constraint. Another hardware-efficient example is from the work in [26] which uses parallel multi-resolution motion estimation (PMRME) algorithm suitable for AVC/H.264 encoder implementation. The work in [27] implements the PMRME algorithm in a 1080p AVC/H.264 video encoder chip.

1.1.4 Hardware Implementation of Motion Estimation

A simple hardware block to perform a motion search is considered in Figure 1-5. Current block's size is $N \times N$ and the search range is $\pm SR$ in each direction. M candidates are evaluated every cycle and compared to *bestCost*. At the end of the search, the candidate with the smallest cost (*bestCost*) is determined. On the search range, the number of possible candidates in each direction is $2 \times SR + 1$ and the total

number of candidates is $(2 \times SR + 1)^2$ as shown in Table 1.2. For each candidate, $N \times N$ pixels are input to the hardware block.

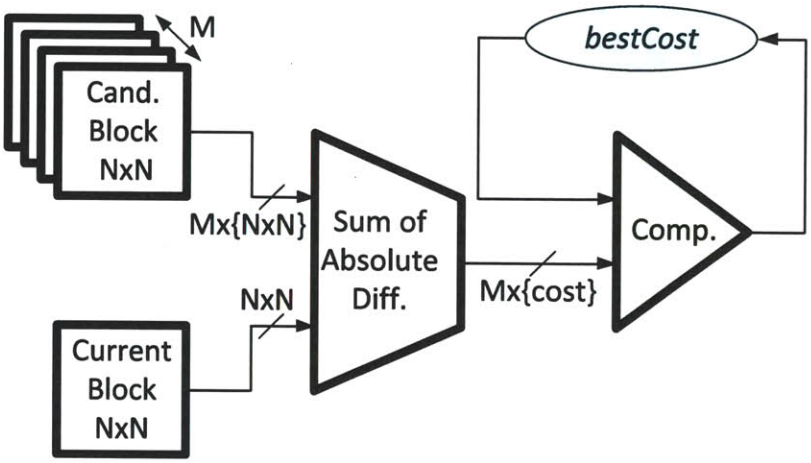


Figure 1-5: Block diagram of a simple hardware block performing motion search.

	Exhaustive Search	**Fast Search (TSS)**
Searching Range Size	$\pm SR$	$\pm SR$
Num. of Candidates	$(2 \times SR + 1)^2$	$1 + 8 \times log_2 SR$
Num. of Parallel Comp.	M	M
Num. of Cycles	$(2 \times SR + 1)^2 / M$	$25/M$
Total Cand. Pixels (No reuse)	$N^2 \times (2 \times SR + 1)^2$	$N^2 \times 25$

Table 1.2: Comparison of exhaustive search and a fast search algorithm in terms of cycle and data requirements.

For exhaustive search, each candidate is evaluated and it takes $(2 \times SR + 1)^2/M$ cycles to complete the search. Assuming no data is reused between candidates, total data requirement is $N^2 \times (2 \times SR + 1)^2$ pixels.

Table 1.2 also considers TSS. A total of $1 + 8 \times log_2 SR$ candidates are considered for TSS, reducing the data and cycle requirements significantly. Obviously, exhaustive search is guaranteed to find the global minimum on a searching range and fast search algorithms such as TSS can converge to a local minimum. This leads to a trade-off between the number of computations and coding efficiency. Specifically, with increasing number of computations, a search engine can find a better match and can

potentially improve coding efficiency. For many applications, coding efficiency is the main concern and more important than power consumption. For other applications, power consumption can be very critical. Hence, search algorithm design requires careful consideration of design trade-offs and is a critical part of a motion estimation design.

For simplicity, the example provided above does not consider some important concepts such as data reuse between candidates. In actual hardware implementations, algorithms are designed to allow data to be used across various candidates to reduce data requirements. In this context, algorithms with regular search patterns are more advantageous and they are easier to implement in hardware. For example for exhaustive search, a line of candidates can be evaluated at once. This can reduce the data requirement by a factor of roughly M. Fast search algorithms with very irregular search patterns can be more power consuming in hardware implementation due to increased bandwidth requirements. Hence, the number of candidates as well as bandwidth should be considered together for hardware suitable search algorithm development.

Recent standards also support fractional pixel accurate MVs. Fractional motion estimation (FME) is the process of interpolating the reference frame for sub-pixel locations and performing a motion search on the interpolated data. AVC/H.264, VC-1 and HEVC support refinement down to quarter-pixel accuracy. Quarter pixel accurate motion vectors can provide bit-rate improvements up to 30% except for very low bit-rate [28]. In FME, pixels from the original frame are first used to calculate half-pixel locations through an FIR filter. Then, original pixels and interpolated half-pixel data are used to calculate the quarter-pixel data. The length of the FIR filter generally determines how complex the FME process is.

Motion estimation can be done in a backward direction where the reference frame(s) is a past frame in the display order, or in a forward direction where the reference frame(s) is a future frame(s) in the display order. Frames allowing only backward search are called P-frames and frames allowing motion search in both directions are called B-frames. Supporting B-frames is critical for coding efficiency but this frame

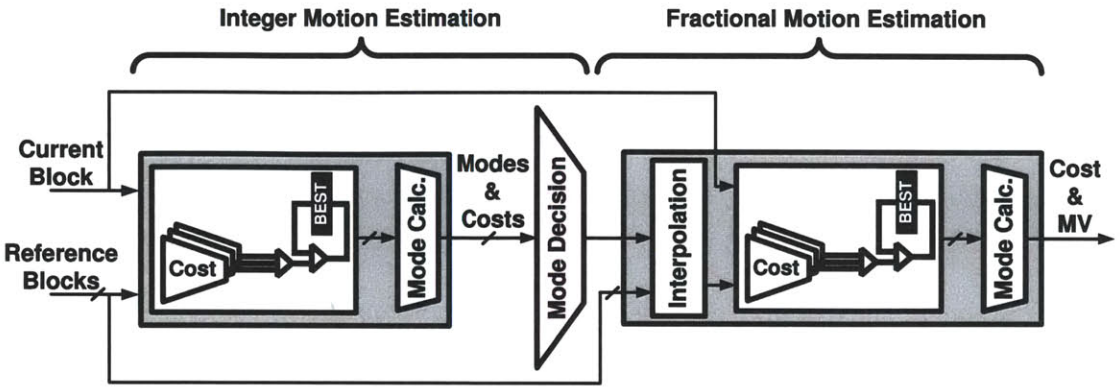


Figure 1-6: Block diagram of a motion estimation engine in hardware. IME and FME parts are shown.

type roughly doubles the hardware requirement for the same throughput constraint due to separate searches in forward and backward directions.

Figure 1-6 shows a block diagram of a motion estimation engine in hardware. Fractional motion estimation (FME) follows Integer motion estimation (IME) as shown in this figure. A motion search is performed in IME and then a few best candidates are selected and a fractional refinement is performed in the FME part. This refinement is done around the MV found in IME part. Reference blocks are generally stored in an on-chip reference pixel buffer which can be joint or separate for IME and FME.

Table 1.3 lists previous encoder chips in literature and compares their resolution, frame-rate, total area, motion estimation area, operating frequency, power and process technology. It should be noted that motion estimation accounts for at least half of the total chip area. Moreover, as the resolution increased over the years, operating frequencies also increased. Hence, despite the effect of process scaling, power consumption of encoder chips increased over the years. Thus, designing a low-power motion estimation module is critical for the overall performance of an encoder.

1.2 Voltage Scaling and Low Voltage Operation

Dynamic voltage scaling is an effective method of reducing energy consumption of circuits under a time varying performance constraint [32, 33]. Specifically, lowering

Work	Standard	Resolution	Rate	Area	ME Area	Freq.	Power	Process
			fps	mm^2	%	*MHz*	*mW*	μm
[29]	H.264/AVC	1280x720	30	31.7	80	108	785	0.18
[30]	MPEG4	640x480	30	7.7	55	28.5	18	0.18
[31]	H.264/AVC	1280x720	30	18.5	54	108	183	0.13
[27]	H.264/AVC	1920x1080	30	10.0	70	145	242	0.13

Table 1.3: Comparison of previously published encoder chips.

the supply voltage (V_{DD}) of the devices results in lower energy per operation at the expense of slower performance.

It has been shown that for most digital circuits, the minimum energy point lies in the sub-threshold region ($V_{DD} < V_T$) where devices are operated in weak inversion [34, 35]. Although transistor drive currents become weaker and circuits run at slower speeds at lower voltages, many applications have time-varying throughput constraints and are compatible with dynamic voltage scaling. Moreover, hardware parallelism can provide the necessary throughput while circuits are running at a lower voltage and at a lower speed [36].

The work in [37] is an earlier example of operation in the deep sub-V_T region. This work demonstrates an FFT processor operating down to 180mV. Other examples of sub-threshold work includes the works in [38], [39], [40] and [41].

However, the performance of the circuits in the sub-threshold region is on the orders of kHz which might not be suitable for various applications. Hence, recent work has focused on operating circuits at a voltage that is slightly above the V_T of the devices, at $\sim 500mV$ for most low-power processes, to balance the trade-off between energy efficiency and performance. An example is the work in [42] which demonstrates a low-power DSP in 28nm CMOS technology that can operate from 0.6V to 1.0V. Similarly, the work in [43] demonstrates a voltage scalable microprocessor SoC that can operate from 0.4V to 1.2V range. The work in [44] demonstrates the dynamic voltage and frequency scaling techniques on a commercial product targeted for mobile SoCs.

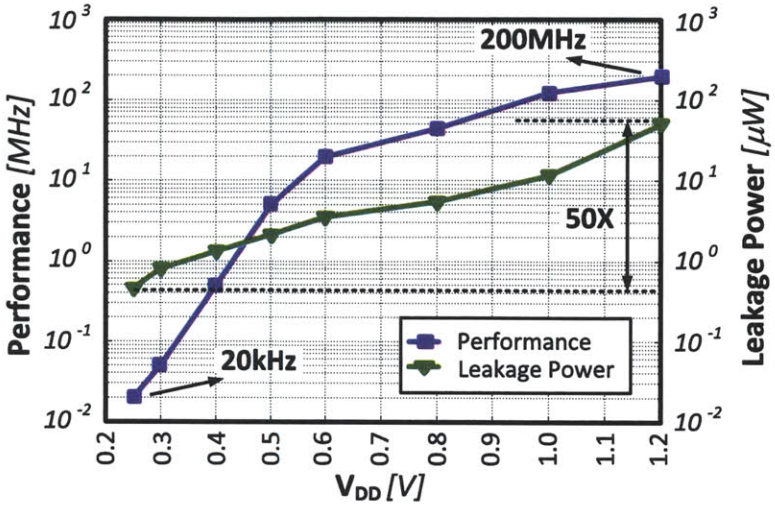
Voltage scaling has the additional benefit of reducing the leakage power consump-

tion of digital circuits. Due to the drain induced barrier lowering (DIBL) effect [45] which is highly pronounced in deeply-scaled CMOS technologies, voltage scaling provides exponential savings in leakage power. Figure 1-7 shows the effect of voltage scaling on performance, leakage power and energy/access for a 64Kbit SRAM macro in 65nm CMOS [46]. First, leakage power is reduced by more than 50X over the voltage range. This emphasizes the importance of voltage scaling in SRAM-rich applications where leakage power can be significant. Next, performance degrades gradually as V_{DD} scales down from strong inversion to moderate inversion. However, at 0.5V, transistors start to enter into the sub-threshold region where performance rolls off exponentially. Finally, energy/access improves monotonically as V_{DD} scales down until it reaches a minimum at 400mV which is know as the minimum energy point [47]. Beyond this point, performance degradation causes the leakage power to be integrated over very long access periods and makes leakage energy dominant. Leakage, performance and energy/access plots show that operating at around V_{DD} =0.5V can provide reasonable performance and significant energy savings.

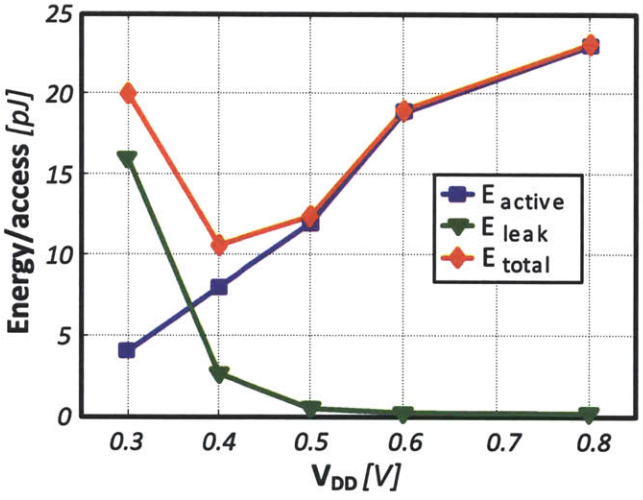
For real-time video applications, there is a throughput constraint. For video decoding, for example, a certain number of frames needs to be processed and displayed every second. Hence, frame-rate and resolution impose the number of pixels that need to be processed per second. As discussed above, hardware parallelism can provide high throughput while operating at a lower voltage. The work in [48] proposes a highly parallelized H.264 baseline decoder operating at 0.7V while decoding 720p frames at 30fps and reports 10X power savings compared to other H.264 baseline video decoders [49, 50].

1.3 Low-Power SRAM Design

SRAMs are the most common type of embedded memories in integrated circuits. As explained at the beginning of this chapter, SRAMs possess very high transistor density and SRAM design decisions are often driven by area efficiency. Hence, conventional SRAM implementations employ small but effective circuits targeting and optimized



(a)



(b)

Figure 1-7: (a) Performance and leakage power and (b) energy/access vs. V_{DD} . Leakage power scales more than 50X from 1.2V to 0.25V and energy/access reaches a minimum around 400mV while operating at 500kHz.

to full-V_{DD} operation.

Transistor variation occurs in two different forms: global and local variation [51]. Global variation affects all devices of the same type (e.g. NMOS or PMOS) the same way. Local variation, on the other hand, are the mismatches affecting each transistor on a die differently which can occur due to lithographic effects or non-uniform concen-

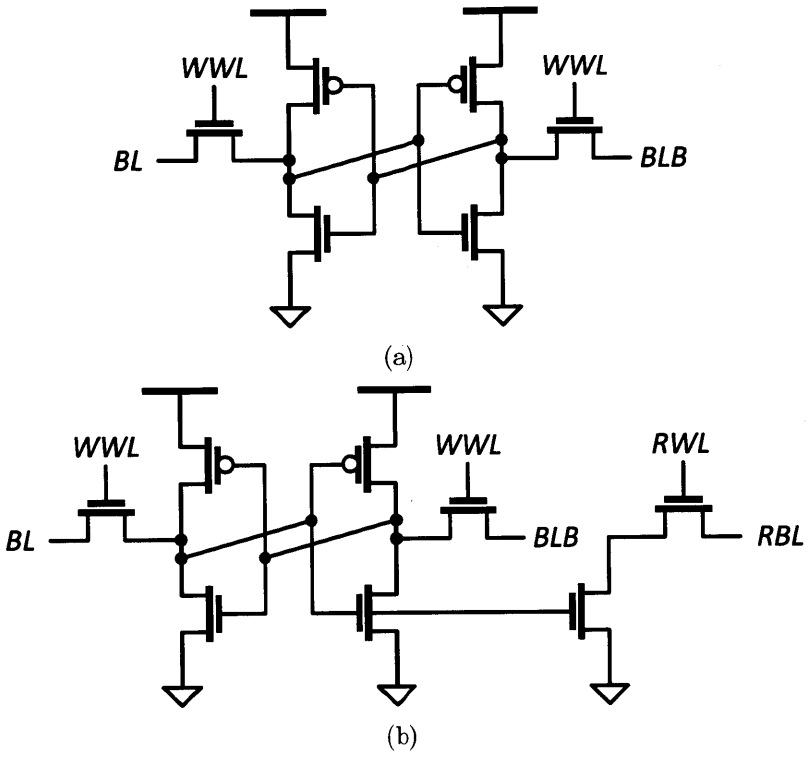


Figure 1-8: (a) Schematic of a six-transistor (6T) SRAM bit-cell and (b) schematic of an eight-transistor (8T) SRAM bit-cell. 8T bit-cell is proposed as a low voltage alternative to the conventional 6T cell.

tration of doping atoms in the channel area (random dopant fluctuation). Different sources of variation can be lumped into a simpler single parameter that implements a change in threshold voltage [52].

Conventional six-transistor (6T) SRAM bit-cell (Figure 1-8-a) is a ratioed circuit. In other words, correct operation of this topology depends on relative strength of its devices. SRAMs are often designed with minimum-size devices to maximize transistor density and consequently their operation is severely affected by process variation.

At low voltage levels in the sub-V_T region of operation, drive strength of a transistor is:

$$I_{sub} = I_o e^{\frac{V_{GS}-Vt+\eta V_{DS}}{nV_{th}}}(1 - e^{-\frac{V_{DS}}{V_{th}}})$$

where V_{GS} ($\leq V_{DD}$) is the gate-source voltage, V_t is the threshold voltage, I_o is from the leakage current model, V_{DS} (equal to V_{DD}) is the drain-source voltage, n is the

sub-threshold slope factor, η is the drain induced barrier lowering factor and V_{th} is the thermal voltage. From this equation, it can be seen that a change in the transistor's threshold voltage has an exponential effect on the transistor current. Hence, transistor mismatches are exacerbated at low voltages causing functional failures for digital circuits.

As the effect of transistor variation is exacerbated at low voltages, operating SRAMs at lower V_{DD} is a challenging design problem. Since a memory consists of a large number of bit-cells, sense-amplifiers and row/column drivers, it is essential to consider the worst-case process, voltage and temperature conditions on these circuits to ensure robust operation. Static Noise Margin (SNM) is a metric used to quantify the stability of a bit-cell under retention state and read/write conditions [53]. For large memories with millions of bits, it is not uncommon to consider $5-6\sigma$ tails of the SNM distributions to ensure robust operation.

Previous research has demonstrated different bit-cell topologies and peripheral assist circuits to enable low voltage operation in SRAMs. An eight-transistor (8T) bit-cell design (Figure 1-8-b) is proposed in [54] and has attracted significant attention from academia and industry.

Along with different bit-cell topologies, peripheral circuits play an important role in SRAM functionality, performance and power. Variations in the bit-cell can be opposed through assisting the bit-cell by various techniques such as voltage boosting or redundancy. Sense-amplifiers which are used to detect voltage changes on bit-lines, for instance, is a key component in SRAMs and should be carefully designed. This is because i) sense-amplifiers can limit the low voltage operation and ii) they have a significant effect on power consumption and speed of the memory.

Although voltage scaling is an effective method of reducing energy/access, application-specific features of a target application should also be considered to further reduce energy/access. For example, the work in [55] uses two different memories for most-significant-bits (MSB) and least-significant-bits (LSB) of pixels in a motion estimation engine. Specifically, MSBs of pixels are stored in an 8T SRAM that is suitable for low voltage operation. In contrast, LSBs of pixels are stored in a 6T design which is prone

to errors at low voltages. However, these errors in LSBs are much less important and barely affect the results of the motion search. Hence, instead of using 8T SRAMs for all bits of a pixel, 6T SRAMs can be used for a few LSB bits and area savings can be provided.

The idea discussed above is suitable for motion estimation as data accessed from the memory is not required to be exact and a few bit errors are tolerable. However, the same idea cannot be applicable for the instruction caches of a microprocessor where bit-errors are much more critical. Since different applications have different requirements, considering the target application at the design stage can provide a flexibility for the designer to explore different opportunities. Specifically, considering statistics of signals and data-dependency in SRAM design can provide significant energy/access savings.

1.4 Previous Work on Low-Power SRAM Design

6T bit-cell has been the workhorse in modern SRAMs because of its simple design and area-efficient layout implementation. The "thin cell" layout [56] of 6T bit-cell is a lithography-friendly implementation with transistor gates running in one direction in the cell.

However, at low voltages, 6T cell suffers from functional problems where a read operation can alter the state of the bit-cell or a write operation cannot overwrite the previous state of the bit-cell. Moreover, degradation of I_{ON}/I_{OFF} ratio of devices at low voltages introduces a challenge for the sensing network to distinguish a logic "0" from a logic "1". Therefore, conventional designs employing 6T bit-cells are not suitable for low voltage operation and recent work has focused on alternative bit-cell topologies, peripheral assist circuits and novel sensing techniques.

1.4.1 Alternative Bit-cell Topologies

One way of achieving low voltage operation is designing 6T bit-cell asymmetrically. The work in [57] uses a unit-β-ratio 6T cell to achieve operation down to 0.7V in

45nm CMOS. The work in [58] also uses an asymmetrical 6T bit-cell and supports it with assist circuits for operation down to 200mV in $0.13\mu m$ CMOS process.

The work in [59] proposes a 7T bit-cell. The extra NMOS transistor is used to break the positive feedback between the latch composed of cross-coupled inverters during a read operation. Consequently, this additional transistor makes the cell "read margin free". In order to maximize area utilization with an asymmetrical bit-cell topology, cell layout is designed to be L-shaped. Then space between the cells is used for the sense-amplifier. This creative layout implementation limits the area overhead to 11%.

An 8T SRAM cell (Figure 1-8-b) is proposed in [54] and has been used in many SRAM designs [60, 61, 62, 63]. 8T cell uses two extra pull-down devices to replicate the pull-down path of the 6T cell. This extra part is called "read-buffer" since it is used for read operations.

A read operation is performed through *RBL* and *RWL* ports of this bit-cell whereas a write operation is performed through *BL/BLB* pair and *WWL*. During a read operation, internal storage nodes are decoupled from the read-buffer. Hence read stability problem does not exist in the 8T topology. However, it should be noted that read stability problem can be eliminated in 8T designs if column-interleaved is not used in the array. This problem and a proposed solution will be discussed in Section 4.3.

One major drawback of the 8T design is its area overhead as area efficiency is one of the most important considerations in SRAM design. Figure 1-9 shows previously published 6T and 8T bit-cell area between 2004-2010 on 90nm down to 32nm CMOS process. From Figure 1-9, it can be seen that 8T bit-cell has 35-40% larger cell area compared to the 6T bit-cell.

Other previous work proposes 10T SRAM cells operational in sub-V_T region ([64, 65, 66]). The works in [64, 65] use four extra transistors to address the sub-V_T functionality problems in the read-buffer. The area overhead of 10T bit-cells over 6T bit-cell is even larger due to additional transistors.

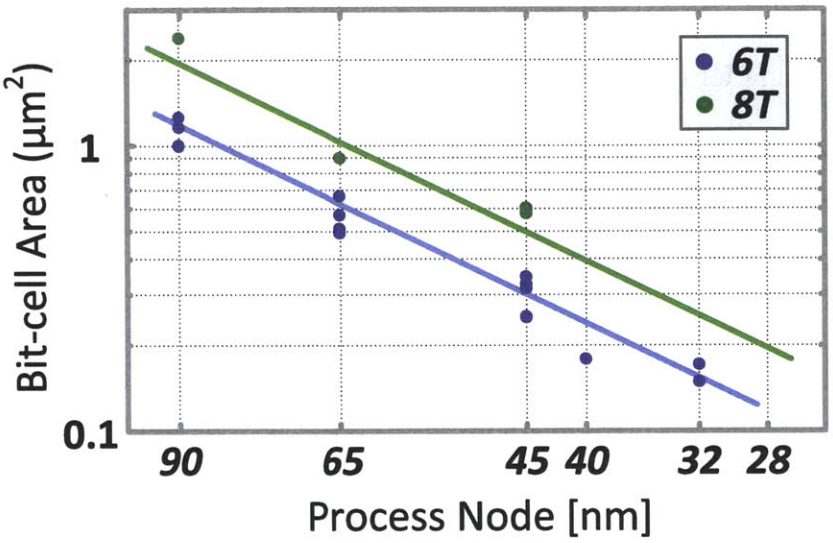


Figure 1-9: 6T and 8T bit-cell area from previously published designs from years 2004-2010. Designs are from various CMOS technologies from 90nm down to 32nm.

1.4.2 Row-wise and Column-wise Assist Circuits

Although different cell topologies enable low voltage operation, peripheral assist circuits are often needed to complement low voltage functionality of the bit-cell.

One of the most widely used methods to assist a write operation is boosting or under-driving specific voltages in the design to improve or degrade driving strength of transistors with respect to each other. Boosting the supply voltage results in a higher driving strength. Moreover, in or close to sub-V_T operating range, the dependency of transistor's driving strength on gate voltage is exponential. Hence, altering the voltage of transistors actively during write operation can provide significant improvement of margins.

Earlier examples of row-by-row power supply scheme are [67], [68] and [69]. The designs in [64] and [70] also use row-by-row V_{DD} , with [64] floating the virtual supply node and [70] actively pulling it down during a write operation.

Alternatively, the work in [71] uses a column-wise approach and modulates the supply voltage of active columns during a write operation to ensure successful overwriting of stored data in the bit-cell. The work in [72] also uses a column-wise scheme where the accessed columns' supply voltage is floated during write accesses.

The work in [63] uses a row-wise write assist circuit where word-lines are boosted to improve write-ability of bit-cells. The work in [73] uses a different method and proposes under-driving of the bit-lines below the ground level for better write-ability. Boosting and under-driving signals require an additional power supply or a voltage boosting circuit employing a capacitor. Both approaches result in area overhead but this overhead can be kept to a minimum by careful design. Lastly, [74] uses an on-chip monitoring circuit to set WL voltage that balances write-ability and read-disturbance for a 6T design to balance read and write failures.

1.4.3 Sensing Topologies

In addition to new bit-cell topologies and peripheral assist circuits, novel sensing circuitry designs further support low voltage operation.

One major difference between 6T and 8T designs is that the former uses differential sensing whereas the latter uses single-ended sensing. The work in [75] provides a comparison of differential and single-ended sensing schemes in SRAMs.

The work in [70] uses sense-amplifier redundancy. By employing a second sense-amplifier for every column and a calibration scheme that runs at the start-up, sense-amplifier with smaller offset can be selected and used in the SRAM operation. Although introducing a second sense-amplifier introduces an area overhead, the size of the sense-amplifiers can be chosen to be small as redundancy will improve probability of failure.

Another work in [76] proposes a regenerative sensing scheme where a small capacitor is used on the signal path to decouple the DC component of the bit-line signal. Amplification of the small signal voltage that is developing on the bit-lines is done through an inverter that is biased to its high gain point.

StrongARM latch type sense-amplifiers ([77]) are widely used in SRAMs. The work in [78] provides an analysis of speed and yield optimization of a similar sense-amplifier structure and claims that the common-mode of the input signals results in an improvement of the sense-amplifier offset. The work in [79] provides an offset and delay analysis for a similar topology structure and analyze the effect of slew-

rate of the enable signal on the speed of the sense-amplifier. It should be noted that previous work focused on design time optimizations on sense-amplifier offset in SRAMs but a run-time scheme measuring and compensating sense-amplifier offsets can be significantly beneficial.

1.5 Thesis Contributions

This thesis focuses on hardware-oriented algorithm and architecture development for motion estimation and provides a hardware implementation cost vs. coding efficiency trade-off analysis. It also explores low-power SRAM design techniques with two different approaches. First, low voltage SRAM design is considered and area-efficient circuit techniques are developed for 6T and 8T bit-cell based designs. Secondly, an application-specific SRAM is designed for motion estimation by considering the specific features of the input data and access patterns to the SRAMs. Statistics of storage data are incorporated into circuit design techniques to provide data-dependent transition probabilities on critical signals in the array to lower energy/access for SRAMs.

The main contributions in this thesis include:

1. **HEVC Motion Estimation Trade-Offs.**

   In Chapter 2, this thesis focuses on hardware cost vs. coding efficiency trade-offs for the next generation video coding standard, HEVC. A methodology is developed to quantify hardware area in terms of core area, on-chip memory area, on-chip memory bandwidth and off-chip memory bandwidth. A scatter plot technique is used to compare different motion estimation configurations based on their hardware cost and coding efficiency. Based on this analysis, architecture decisions are made to reduce hardware area with minimum coding efficiency loss. Moreover, hardware-oriented search algorithms are developed to further reduce implementation costs [80, 81]. A final design which is efficient in terms of cost and coding performance is implemented at the HDL level. Lastly, a hardware-oriented advanced motion vector prediction (AMVP) algorithm is

developed to reduce hardware implementation costs [82]. When compared to the original implementation, our proposed algorithm reduces area by 2× while providing the same coding efficiency. The proposed algorithm is adopted to the HEVC standard by the standards committee.

AMVP algorithm development was done in collaboration with engineers from Texas Instruments (Minhua Zhou and Vivienne Sze).

2. **Highly-Parallel Multi-Standard Motion Estimation Design.**

   Chapter 3 focuses on a highly-parallel motion estimation design targeting ultra-low voltage operation while supporting $4K \times 2K$ video resolution. Frame and macro-block parallel processing are implemented for this work to meet the high throughput constraint while running at a modest 25MHz frequency. Suitable algorithms and architectures are developed and implemented for this highly-parallel design. Moreover, hardware reconfigurability is used in the design to maximize hardware sharing between two supported standards (AVC/H.264 and VC-1).

   A post-doctoral researcher and two graduate students were involved in the design of the encoder and I was responsible for the motion estimation part.

3. **Low-Power and Low Voltage SRAM Design.**

   In Chapter 4, two voltage scalable SRAM designs are presented in deep-submicron CMOS process technologies. The first design demonstrated that using a high-density 6T bit-cell with area-efficient peripheral assist circuits can provide operation down to 0.6V in 28nm CMOS technology. Short local bit-lines, voltage boosting and low voltage oriented read path optimization ideas are developed while minimizing the area overhead associated with them. A prototype test chip fabricated in 28nm CMOS technology achieves functionality down to 0.6V as targeted [83]. These SRAMs are characterized and used in a low-power multimedia processor design as well [42]. The second design in Chapter 4 presents a different approach and uses an 8T bit-cell to achieve low-voltage operation.

SRAMs in this work implement a new array architecture to enable column-interleaving with 8T bit-cells. An on-chip reference selection loop is also demonstrated to reduce sense-amplifier offset. Test chips fabricated in 45nm CMOS technology achieve functionality down to 0.5V [84].

4. **Application-Specific SRAM Design.**

   In Chapter 5, the idea of using application-specific features to reduce energy/access is applied to SRAM design for motion estimation. A Prediction-based reduced bit-line switching activity (PB-RBSA) scheme is proposed to exploit the correlation of input pixel data to reduce switching activity on the bit-lines of the memory. To complement this idea, a hierarchical sensing network with statistical sense-amplifier gating is developed to take advantage of the biased transition probabilities on the bit-lines due to PB-RBSA. A prototype test chip implementing these ideas is fabricated in a 65nm CMOS technology and achieves up to 1.9× lower energy/access when compared to a conventional 8T design.

Chapter 2

Cost and Coding Efficient Motion Estimation Design for HEVC Standard

In this chapter, hardware cost of an HEVC motion estimation engine is going to be explained and the details of a cost and coding efficient (CCE) motion estimation implementation for HEVC is provided. As mentioned in Chapter 1, HEVC is the next generation video compression standard that is currently being standardized. Although there is not a reference hardware implementation that has been published yet, the complexity of the tools that are being adopted by HEVC is significantly higher than previous standards and this is a valid indicator to expect a higher hardware complexity as well. Since the hardware cost of a video codec is extremely important for portable multimedia devices, it is imperative to analyze the hardware implementation cost of various tools. Moreover, based on this analysis, design decisions can be made to provide a balanced trade-off between hardware cost and coding efficiency as these two concepts are almost always conflicting.

2.1 Overview of HEVC

HEVC has a design target of 50% coding efficiency gain over AVC/H.264 High Profile. To achieve this challenging goal, many new tools are being proposed to this standard. Table 2.1 provides a comparison between some of the tools used in AVC/H.264 and HEVC standards (based on HM-4.0).

Tool	AVC/H.264	HEVC
Coding Quad-Tree Structure	No	Yes
Largest Coding Unit (LCU) Size	16×16	64×64
Asymmetric Motion Partitions	No	Yes
Inter Merge Mode	No	Yes
Transform Size	4×4 and 8×8	Up to 32×32
Non-square Transform	No	Yes
Intra Prediction Angular Directions	8 Directions	33 Directions

Table 2.1: Comparison of some tools in AVC/H.264 High Profile and next generation video standard, HEVC. More complex HEVC tools require a more complex hardware.

2.1.1 Coding Quad-Tree Structure

One of the main differences of HEVC from its predecessor AVC/H.264 is the adoption of coding quad-tree structure to provide a modular coding structure. In HEVC a frame is divided into largest coding unit (LCU) and an LCU is further divided into four coding unit (CU) in a quad-tree structure. Currently, LCU size can be as large as 64x64 pixels and smallest coding unit (SCU) size can be as small as 8x8. This allows the selection of a different coding structure based on various factors such as input video resolutions and other properties of a video sequence.

Figure 2-1 shows three different coding tree structure selections for three different scenarios. For a high-resolution video with large and smooth backgrounds and a smaller level of details, LCU and SCU sizes can be selected to be 64×64 and 32×32 respectively. For the same resolution, but with a higher level of details in the video,

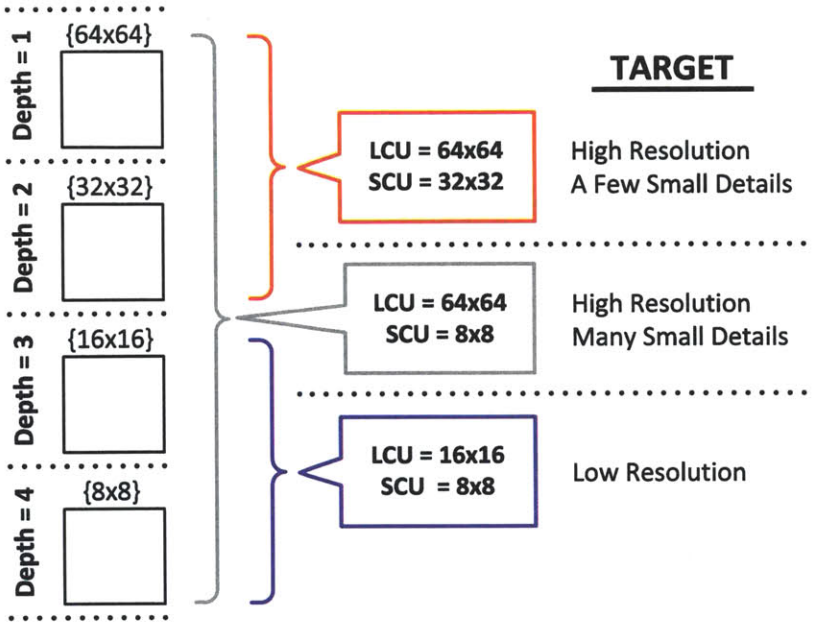


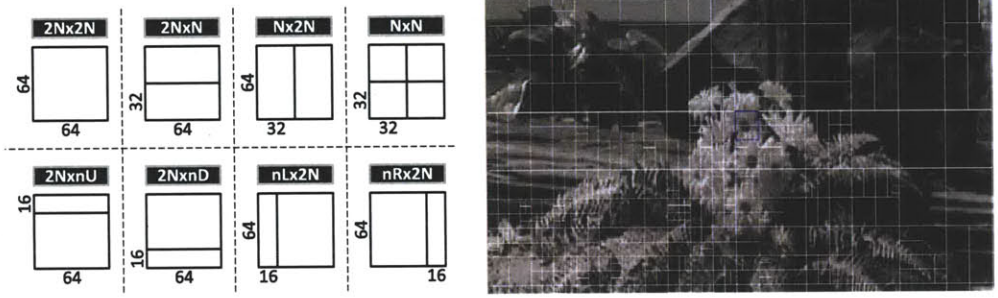
Figure 2-1: Coding quad-tree structure used in HEVC. Different LCU and SCU selections can be done based on the resolution and amount of details in the video frames.

SCU can be selected to be smaller (8×8) to capture and represent small details with a better coding efficiency. Lastly, for small resolutions, LCU size can be selected to be smaller (16×16).

If a CU is not divided into smaller CUs, it is predicted with one of several prediction unit (PU) types. PU types determine which prediction type will be used to code a particular CU. For inter-prediction (motion estimation), PU types can be $2N \times 2N$, $2N \times N$, $N \times 2N$ or $N \times N$ where $2N \times 2N$ corresponds to the size of the CU. If asymmetric motion partitions (AMP) are used, non-square PUs for inter-prediction also include $2N \times nU$, $2N \times nD$, $nL \times 2N$ and $nR \times 2N$. AMP partitions are not included in the hardware cost and coding efficiency analysis in this work but this analysis can be extended to cover these partition types as well. $N \times N$ is only used at the SCU level not to present a redundancy. This is because $N \times N$ PU of a CU can be represented with a $2N \times 2N$ PU at the next depth except for the SCU.

Figure 2-2-a shows all possible inter-prediction PU types and their corresponding sizes in a 64x64 LCU. Figure 2-2-b shows an example video frame with the CU and

PU partitions selected by an encoder [85]. It is important to note that the modular structure of the quad-tree improves coding efficiency by allowing smooth parts of a frame to be coded with a large CU and areas with a lot of details with smaller CUs. Object boundaries that fall into a CU, on the other hand, can be represented with different non-square PU types.



(a) (b)

Figure 2-2: (a) PU types for inter-prediction in a 64x64 CU and (b) an example showing PU types used in a video frame.

2.1.2 Reference Software Implementation

JCT-VC provides a reference software implementation, HM, of a codec to quantify coding efficiency gain [86]. Proposed changes are implemented on this software and a pre-determined set of sequences [12] are coded with the proposed changes to quantify results. Appendix A provides the list of this standard set of sequences, their resolution as well as total number of frames. In this thesis, all simulations for HEVC motion estimation used this set of sequences with HM-3.2 version of the reference software.

Test sequences cover a wide range of different videos. Indoor and outdoor videos, large/complex motion and slow/regular motion are a few examples of particular properties that are tested with these test sequences.

It is important to note that the reference software implementation is intended to see the maximum limits of the coding efficiency. Hence, at the encoder side,

HM goes through almost all possible CU sizes and PU types for motion estimation. Moreover, the motion search algorithm implemented in HEVC is highly complex and very costly to be realized in hardware. Although it is necessary to see the maximum achievable coding efficiency, new algorithms and architectures are necessary when targeting hardware implementation. The trade-offs associated with area, power and data bandwidth vs. coding efficiency should be taken into account when developing these algorithms.

2.2 Overview of Motion Estimation in HEVC

Motion estimation in HEVC is block-based where block sizes can be as large as 64×64 (LCU size) and as small as 4×4 ($N \times N$ PU in an 8×8 CU). A 64×64 LCU can be represented by a single $2N \times 2N$ PU or it can be divided into 8×8 CUs where each CU is represented with four 4×4 blocks ($N \times N$ PU type). In the former case, an LCU is represented with a single MV pair and in the latter case, with 256 MV pairs. For an LCU with many details, using smaller block sizes with separate MVs can provide better compression. In contrast, for large and smooth areas, using larger block sizes and fewer MVs can be more efficient. Hence, supporting all block sizes provide the highest flexibility and best compression efficiency but this also results in highest hardware implementation cost.

Motion estimation is on the encoder side but a video compression standard only defines the decoder side. Hence, encoder side decisions can be different from one design to the other as long as the output of the encoder is compliant with the standard. The decisions made on the encoder side, however, affects compression efficiency. In this thesis, the encoder implementation given in HM software is used as a reference point.

Since HM is implemented to achieve highest coding efficiency, in motion estimation, motion cost associated with every possible CU sizes and PU types are calculated to find the best combination that provides the smallest cost for the overall LCU and the overall frame. It should be emphasized that these searches for different CU sizes

and PU types are performed around different and independent centers. Hence, it is not possible to share the hardware and cycles between these searches in a hardware implementation. Moreover, the processing order is completely sequential in HM since there are some dependencies between the neighboring blocks. Figure 2-3 shows the processing order of a 8×8 CU for four different PU types ($2N \times 2N$, $2N \times N$, $N \times 2N$ and $N \times N$). Motion information of the top and left neighboring blocks are used in the cost calculations and MV coding and hence, coding order goes from left to right and from top to bottom.

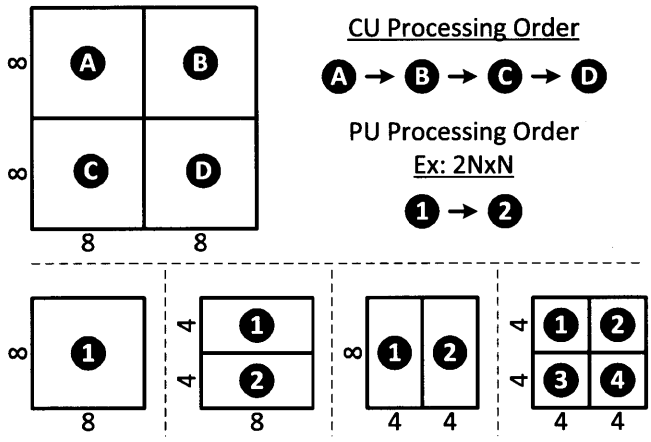


Figure 2-3: Processing order for 8×8 CUs in a 16×16 CU is from A to D. For each 8×8 CU, PU types are also processed sequentially from $2N \times 2N$ to $N \times N$. Finally inside a PU type, processing order is from 1 to 4.

This sequential processing is harder to implement in hardware. Hence, many work in the literature looked at breaking this dependency by using estimates of neighboring block's motion information [23, 24, 27]. However, these estimates cause a degradation in coding efficiency. So the main advantage of the sequential processing is that using the exact motion information results in the maximum achievable coding efficiency.

In the rest of this section, an overview of the important blocks in HEVC motion estimation will be provided.

2.2.1 Advanced Motion Vector Prediction (AMVP)

As discussed in Chapter 1, motion estimation extracts the temporal redundancy between frames and can represent a block of pixels by a pair of MVs (in horizontal and vertical directions) and some residue. If the motion is uniform on a large area, then MVs in that area can be highly correlated. Hence, the number of bits to represent a block can be further reduced if MV differences rather than absolute MVs are transmitted.

In AVC/H.264, this idea is used to calculate a motion vector predictor (MVP) and signaling the difference of a MV with the MVP. If the encoder and decoder use the same algorithm to calculate the MVP, then there is no reason for synchronization to be lost. MVP in AVC/H.264 used the neighboring blocks' motion information. Specifically, immediate neighbors from top, top-right and left that have already been coded are used to calculate the MVP since these blocks are the closest to the current block that is being coded.

In HEVC, AMVP, a more complex MVP calculation method, is being implemented. Spatial neighbors as well as temporal co-located blocks and their motion information are used to calculate MVP. Specifically, for the temporal blocks' motion information, temporal difference of frames in time is calculated and MVs are scaled accordingly. Moreover, a list rather than a single candidate of MVPs are calculated in HEVC and encoder decides and signals which member of the AMVP list is being used. It is important to note that the latter approach adopted in HEVC is far more complex compared to AVC/H.264 because of

- using a larger number of blocks to calculate AMVP list,
- scaling process which uses up to two multiplication operations and
- calculating a list of MVPs rather than a single MVP.

Consequently, the hardware implementation of the AMVP block can be significantly more costly in terms of area and power consumption. In this thesis we present a hardware-oriented algorithm developed for AMVP [82]. This algorithm is proposed to

the standard committee and is being adopted by HEVC. The details of the algorithm development and its hardware cost will be discussed in Section 2.5.

2.2.2 Integer Motion Estimation Search Algorithm in HM

Fast search algorithm in HM consists of four main stages:

1st Stage In the first stage, candidates from AMVP list as well as (0,0) are checked and the candidate with the smallest cost is selected as the search center.

2nd Stage In the second stage, a diamond search is performed on the search range. For a ± 64 search range in each direction, diamond search takes 7 steps to be completed. An early stopping scheme is implemented for this search. If the best candidate does not change in three consecutive steps, then diamond search is concluded.

3rd Stage Execution of the third stage depends on the result of the second stage. If the best candidate from second stage is not more than 5 pixels away from the search center, then third stage is skipped. Otherwise, in third stage, a raster scan is performed that goes over the entire search window by checking every 5th candidate in each direction.

4th Stage In fourth stage, another diamond search around the best candidate is performed. However, an early stopping scheme is not used in this case. Moreover, if a new best candidate is found, fourth stage is restarted with the new search center.

This search algorithm is very effective in finding the best MV in various different cases. For example, for regular motion on a large area, AMVP list candidates in the first stage can capture the correlation of MVs. For the case of more complex and irregular motion, third stage can span the entire search area. Finally fourth stage checks if a local minimum (rather than a global minimum) is found in the search area. However, total number of candidates that are checked can change significantly from one block to the next. In hardware implementation, this is not very desirable.

Designing for the worst-case can result in a large overhead whereas designing for the average load might cause the cycle budgets to be inadequate. Hence, more complex control circuitry is necessary to handle variable number of candidates.

Table 2.2 provides how many candidates are checked for the best and worst case conditions. Best case occurs if the second stage finishes with the early stopping scheme and fourth stage is performed only once. Worst case, in contrast, occurs when the second and third stages are performed and fourth stage is executed twice. It should be noted that the fourth stage can be executed more than twice but it is assumed to be twice for simplicity.

Stage	Best Case	Worst Case
1st Stage	4	4
2nd Stage	24	56
3rd Stage	0	676
4th Stage	56	112
Total	**84**	**848**

Table 2.2: Number of candidates checked for HM's fast algorithm. Worst and best case numbers are provided.

2.2.3 Fractional Motion Estimation

In HEVC, 8-tap filters are used to interpolate pixels for sub-pel locations (based on HM-3.2). Filter coefficients for half and quarter-pel positions are given in the standard as:

$$Half - Pel : [-1, 4, -11, 40, 40, -11, 4, -1]$$

$$Quarter - Pel : [-1, 4, -10, 57, 19, -7, 3, -1]$$

These filters can be implemented by using bit-wise shift operations followed by additions.

Fast search algorithm in HM performs a fractional pixel refinement after IME. FME algorithm is as follows:

1st Stage Set the integer MV as the search center and check eight neighboring half-pel locations surrounding the center. Update the best location to be the integer MV or one of the eight half-pel locations.

2nd Stage Set the MV from first stage as the search center and check eight neighboring quarter-pel locations surrounding the center. Update the best location to be the input from first stage or one of the eight quarter-pel locations.

This search algorithm is suitable for hardware implementation. Although there is a dependency between the two stages, number of candidates is fixed to 16 candidates which makes hardware implementation simpler.

2.3 HEVC Motion Estimation Design for Reference Software-Equivalent Coding Efficiency

As mentioned in Section 2.2, HM implementation is completely sequential on the processing of the blocks and consequently achieves highest coding efficiency. It is important to consider an architecture which is capable of implementing this sequential processing and quantify the hardware cost of realizing a motion estimation engine providing a coding efficiency that is equivalent to reference software. This section presents an architecture that is capable of processing blocks sequentially. Then, the hardware cost of HM's search algorithm is quantified with a methodology to estimate area, power and bandwidth. Finally, a trade-off analysis is done that compares different motion estimation configurations supporting only a subset of all block sizes in terms of area, bandwidth and compression efficiency.

2.3.1 HEVC Motion Estimation Architecture for Reference Software-Equivalent Coding Efficiency

In hardware, HM's sequential processing of blocks requires separate and independent engines performing motion search for different block sizes. Block sizes are determined by the corresponding CU sizes and PU types. Figure 2-4 shows an HEVC motion estimation engine architecture supporting all block sizes from 64x64 down to 4x4 except AMP partitions. This architecture can be generalized to cover AMP partitions as well.

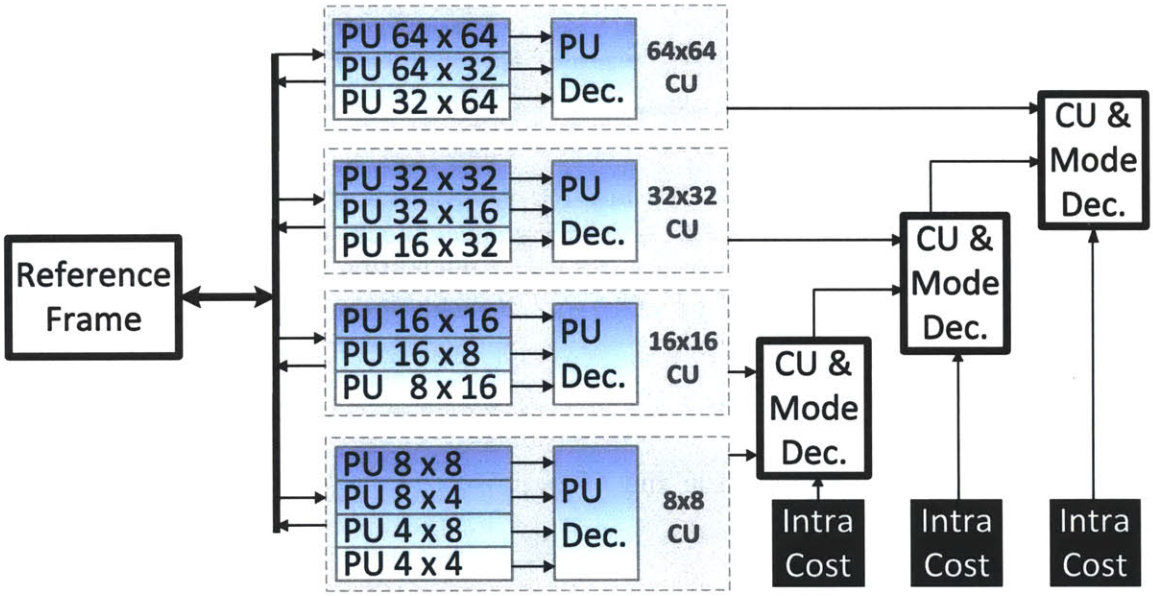


Figure 2-4: Architecture for an HEVC motion estimation engine supporting all block sizes from 64×64 to 4×4 (except AMP partitions). This architecture allows sequential processing of smaller blocks and can use exact motion information from neighboring blocks and provide a coding efficiency as good as the reference software implementation.

There are a total of 13 engines in the architecture in Figure 2-4: Three engines for each CU size (e.g. 32×32, 32×16 and 16×32 for the 32×32 CU) except for the 8×8 CU where there is a fourth engine to support $N \times N$ (4×4) partition. Each engine consists of blocks to perform

- AMVP list,

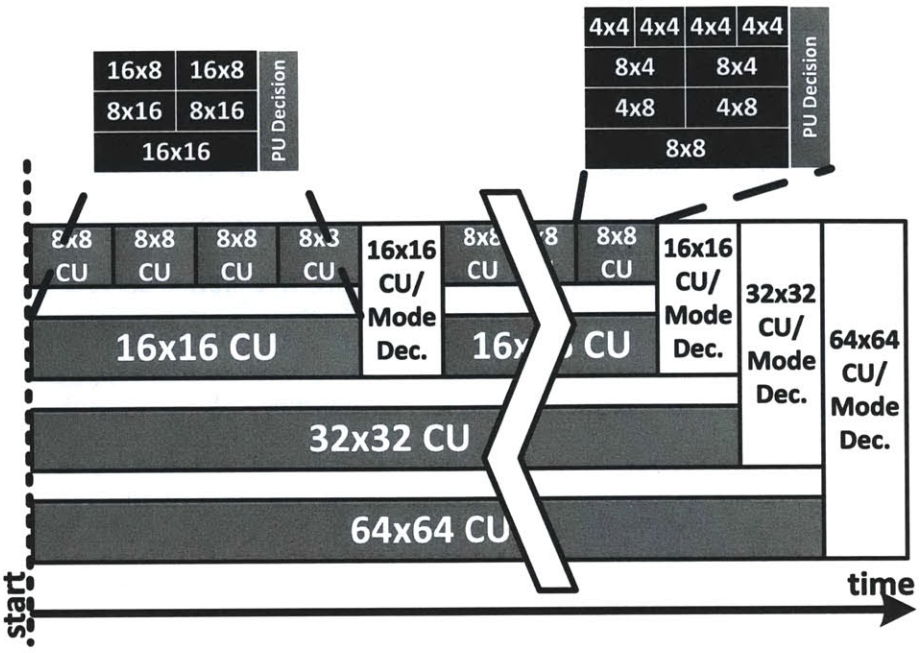


Figure 2-5: Processing order of CUs and PU types inside CUs for the architecture in Figure 2-4. For a 64 × 64 LCU, costs for smaller blocks are combined and then compared to larger block sizes to find the best combination of blocks providing the smallest cost for the entire 64 × 64 LCU.

- integer motion estimation,
- fractional motion estimation and
- a reference pixel buffer.

The processing order for one 64 × 64 LCU is shown in Figure 2-5. Motion searches are performed for four 4 × 4 blocks, two 8 × 4 and 4 × 8 blocks and one 8 × 8 block. Then a PU decision is done to decide what PU type provides the smallest cost for the first 8 × 8 CU. Similarly, three more 8 × 8 CUs are processed sequentially and their costs are output to *CU & Mode Decision* block. During this time, PU decision for the first 16 × 16 CU is also finished and a decision can be done for the first 16 × 16 CU. This continues until an entire 64 × 64 LCU is processed by all engines.

It should be noted that intra/inter decision is done at the CU level and hence costs associated with intra prediction are being provided as external inputs to make an intra/inter decision.

It is also important to note that, for a fixed throughput constraint, cycle budget to process a smaller block size is tighter. Hence, data bandwidth requirements can be significantly larger for smaller block sizes compared to larger block sizes. Consequently, smaller block sizes impose a larger hardware cost. Hardware cost analysis will be discussed with more details in the next section.

2.3.2 Hardware Cost Analysis Overview

This section will be talking about the hardware cost analysis of HEVC motion estimation module providing the reference software-equivalent coding efficiency, i.e. the coding efficiency provided by HM reference software. The top level architecture given in Section 2.3 will be used for this analysis and the algorithms used in HM implementation will be analyzed.

First, we need to decide on a set of hardware specifications to be able to quantify hardware cost. Table 2.3 shows the maximum resolution and frame-per-second that will be supported for an encoder and the frequency of operation for the hardware. From maximum resolution, frame-rate and hardware frequency numbers, available cycles/LCU can be calculated to be 3292 cycles. This number is the throughput requirement for the encoder to be able to achieve maximum resolution and frame-rate when running at 200MHz. It should be noted that because of the sequential processing of blocks, available cycles/block will be smaller for smaller block sizes. For example, for a 4×4 block, the number of available cycles is only

$$\frac{3292}{16 \times 16} \cong 13.$$

Since 13 cycles is very tight for the amount of candidates to be evaluated, we need to increase the amount of parallelism when comparing candidates to be able to meet the throughput requirement.

Specifications of an HEVC Encoder	
Maximum Resolution	3840 × 2160
Maximum Frame Rate	30
LCU Size	64
Frequency of Operation	200MHz
Number of LCUs/sec	60750
Available cycles/LCU	3292
Bi-direction Frames	Yes
# of Reference Frames	1 for each direction
Search Range	±64 in x- & y-dir

Table 2.3: Specifications for a possible HEVC encoder using the motion estimation design in this work. The number of available cycles/LCU is 3292.

2.3.3 Logic Area Estimation Method & Results for HM Implementation

Logic area estimation methodology is as follows:

1. Implement basic building blocks in hardware and use synthesis tools to get unit area and power numbers at the frequency of operation point.

2. Calculate the amount of parallelism required for the throughput constraint.

3. Estimate total area by using unit numbers and amount of parallelism.

Due to the sequential processing of blocks requirement for using exact motion information from neighboring blocks, in the implementation of the engine, it is assumed that there is no pipelining between consecutive stages. Specifically, AMVP, IME and FME blocks are not pipelined.

In the top level architecture given in Figure 2-4, there are a total of 13 parallel engines. Looking at the total number of candidates, total number of pixel calculations, number of available cycles and finally number of pixel calculations/cycle in Table 2.4, it can be seen that calculations/cycle number stays constant. Although the number of available cycles is getting larger from smaller blocks to larger blocks, number of

Block Size	# of Calc. for Search	# of Pixel Calculations	Available Cycles	Calculations per cycle
64x64	N	$64 \times 64 \times N$	T	$64 \times 64 \times N/T$
64x32	N	$64 \times 32 \times N$	$T/2$	$64 \times 64 \times N/T$
32x64	N	$32 \times 64 \times N$	$T/2$	$64 \times 64 \times N/T$
32x32	N	$32 \times 32 \times N$	$T/4$	$64 \times 64 \times N/T$
32x16	N	$32 \times 16 \times N$	$T/8$	$64 \times 64 \times N/T$
16x32	N	$16 \times 32 \times N$	$T/8$	$64 \times 64 \times N/T$
16x16	N	$16 \times 16 \times N$	$T/16$	$64 \times 64 \times N/T$
16x8	N	$16 \times 8 \times N$	$T/32$	$64 \times 64 \times N/T$
8x16	N	$8 \times 16 \times N$	$T/32$	$64 \times 64 \times N/T$
8x8	N	$8 \times 8 \times N$	$T/64$	$64 \times 64 \times N/T$
8x4	N	$8 \times 4 \times N$	$T/128$	$64 \times 64 \times N/T$
4x8	N	$4 \times 8 \times N$	$T/128$	$64 \times 64 \times N/T$
4x4	N	$4 \times 4 \times N$	$T/256$	$64 \times 64 \times N/T$

Table 2.4: Number of calculations per cycle is constant between engines supporting different block sizes.

computations/block is also getting larger with the same factor. Hence, the hardware required for different engines due to candidate evaluation is mostly constant.

Total area of one engine including IME, FME and AMVP blocks is estimated to be 305K gates.

It is important to note that the entire motion estimation module in Figure 2-4 consists of 13 engines, resulting in roughly 4M gates. Moreover, to support forward and backward motion estimation, this number needs to be scaled up by roughly a factor of two.

Block Size	On-Chip Mem. Size	Block Size	On-Chip Mem. Size
64x64	39KB	16x8	21KB
64x32	33KB	8x16	21KB
32x64	33KB	8x8	20KB
32x32	28KB	8x4	20KB
32x16	25KB	4x8	20KB
16x32	25KB	4x4	19KB
16x16	23KB		

Table 2.5: On-chip reference buffer size needed for each engine to support a search range of ±64 in each direction for a single reference frame.

2.3.4 Memory Size and Bandwidth Estimation Method & Results

On-Chip Reference Buffer Size

As explained in Section 2.2, each motion estimation engine in Figure 2-4 is performing independent searches and for each engine, a separate memory is necessary in each direction (forward and backward) and for each reference frame. Table 2.5 shows the size of on-chip memory needed to support ±64 search range. Four extra pixels are necessary on each side of the search window for pixel interpolation in FME and they are included in calculations.

A total of 0.65MB of on-chip memories are necessary to support the luma component of a single reference frame in forward and backward directions for the entire motion estimation module in Figure 2-4. As IME and FME are time-interleaved, they can share the same memory. This number heavily depends on the selected search range size and a smaller search range can result in smaller on-chip memory size and area. However, using a smaller search range size results in a loss in compression efficiency.

To quantify the effect of search range size on the compression efficiency, simulations in HM-3.0 are performed under the test conditions defined in [12] and results are provided in Table 2.6. From ±64 to ±16, bit-rate increase is 0.1%, 0.1% and 3.5%

in low delay, low delay with P and random access test conditions. It should be noted that HM searches through all possible combinations during encoding and also implements a highly-complex search algorithm. In a practical hardware implementation, the coding loss due to reduced search range size can be expected to be larger.

In high-definition sequences, a search range that is large enough to capture the movement of pixels is necessary. The loss in compression efficiency is more pronounced for high resolution sequences and a search range smaller than ± 64 is not feasible for an encoder supporting resolutions up to 3840×2160.

It should be noted that on-chip memory size for small block sizes is not significantly lower than the size for larger block sizes (39KB for 64x64 and 19KB for 4x4) since on-chip memory size does not scale down proportionally to the block size.

Additional on-chip storage (e.g. line buffers for motion information) can be necessary for *AMVP* but its size heavily depends on the specific implementation. Moreover, these buffers can be shared across parallel engines. For a $4K \times 2K$ video encoder, the amount of storage is estimated to be 0.03MB for forward and backward directions with a single reference frame.

On- and Off-Chip Bandwidth

The second consideration in this section is about on- and off-chip bandwidth since these numbers can be a limiting factor in practical implementations. On-chip bandwidth is determined by the size of reference buffer for each engine and how frequently it is accessed. For the search algorithm in HM, during IME, entire search range is

Search Range	Low Delay	Low Delay & P	Random Access
± 64	0%	0%	0%
± 32	0%	0%	0.8%
± 16	0.1%	0.1%	3.5%

Table 2.6: Effect of search range window size on coding efficiency. Increases in bitrate are given with respect to HM-3.0. Single reference frames in both directions are used.

accessed if the result of the initial search is not good enough (3rd stage). This occurs in the case of complex motion. To capture the worst-case upper limit, it can be assumed that the entire search range in the reference buffer is accessed for every block. On-chip bandwidth for FME is significantly smaller as only a refinement is done at this stage. Lastly, bandwidth for motion information of neighboring blocks that is necessary for *AMVP* candidate calculations is small compared to the on-chip bandwidth of the integer and fractional motion estimation so it can be ignored.

Block Size	**On-Chip BW**	**Off-Chip BW**	**Block Size**	**On-Chip BW**	**Off-Chip BW**
64x64	2.2	1.49	16x8	39.6	13.72
64x32	3.8	1.86	8x16	39.6	10.33
32x64	3.8	1.48	8x8	75.6	17.47
32x32	6.4	3.64	8x4	145.9	30.21
32x16	11.5	6.05	4x8	145.9	22.94
16x32	11.5	5.20	4x4	283.8	36.92
16x16	20.9	7.62			

Table 2.7: On- and off-chip bandwidth requirement for each engine in Figure 2-4 with a search range of ±64. All numbers are in GB/s.

Off-chip bandwidth considered here is the off-chip memory's read bandwidth to bring reference pixel data from off-chip to the on-chip buffers for motion estimation. Similarly, off-chip bandwidth is determined by the size of the reference buffer and how frequently reference buffers for each engine need to be updated. Because of the correlation of motion between neighboring blocks, in the ideal case, data re-use between consecutive blocks can be close to 100%. However, it should be noted that the processing order of blocks in an LCU does not allow 100% data re-use and hence causes the same part of the reference window to be read multiple times. Increasing size of the on-chip buffer can improve the data re-use at the expense of larger on-chip memory area. In this work, minimum buffer sizes given in the previous sub-section are assumed in the bandwidth calculations. Lastly, pixel data to evaluate *AMVP* candidates might require additional bandwidth if these candidates are spatially far away from each other and do not fall into the search range in reference buffer.

Table 2.7 shows on- and off-chip bandwidth requirement for each engine. It should be noted that small block sizes such as 4×4 require a very large on-chip and off-chip bandwidth compared to larger block sizes and impose a higher cost for hardware implementation.

2.3.5 Trade-off Analysis for Hardware Cost and Compression Efficiency

The analysis provided above is important to quantify the hardware cost of implementing a motion estimation hardware block that can provide a reference software-equivalent compression efficiency. In this section, we will analyze other motion estimation configurations where some block sizes are not supported and consequently we need less than 13 engines. However, the compression efficiency will be worse because of the exclusion of some block sizes. It is important to quantify the savings in hardware and loss in compression efficiency to be able to make an optimum decision between supported block sizes.

Figure 2-6 shows hardware area, power and bandwidth as well as coding efficiency results for 11 different motion estimation configurations. Each column corresponds to a different configuration supporting all or some of the available block sizes. Configuration #1 supports all block sizes and is the anchor configuration for this work. HM-3.2 simulations are performed to quantify coding loss for each configuration. The bit-rate increase in Figure 2-6 is given as the average of the numbers from all-intra, low-delay, low-delay P and random-access common test conditions defined by JCT-VC [12]. The common test conditions cover a wide range of sequences with resolutions as small as 416×240 and as large as 2560×1600 as given in Appendix A.

Figure 2-7-a and Figure 2-7-b plot core area savings vs. bit-rate increase and off-chip bandwidth savings vs. bit-rate increase for 10 configurations in Figure 2-6 with respect to the anchor, configuration #1. Each configuration is denoted by a dot on this figure except for the anchor configuration as the anchor would be at the origin of the plot. The slope of the lines connecting each configuration to the origin provides

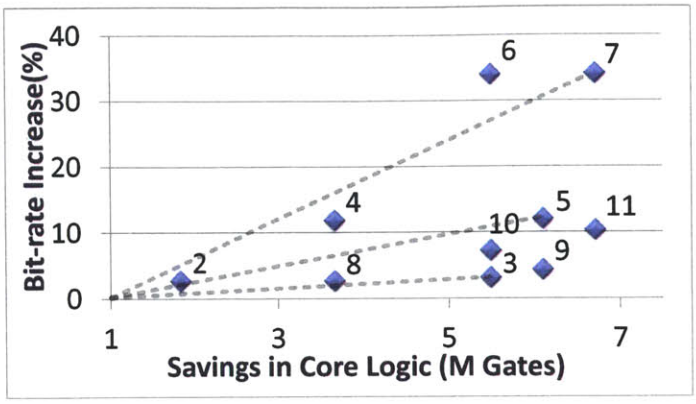
	Configuration #										
	1	2	3	4	5	6	7	8	9	10	11
64x64	Y	Y	Y	Y	Y	Y	Y	N	N	N	N
64x32	Y	Y	N	Y	N	Y	N	N	N	N	N
32x64	Y	Y	N	Y	N	Y	N	N	N	N	N
32x32	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
32x16	Y	Y	N	Y	N	N	N	Y	N	N	N
16x32	Y	Y	N	Y	N	N	N	Y	N	N	N
16x16	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
16x8	Y	Y	N	N	N	N	N	Y	N	Y	N
8x16	Y	Y	N	N	N	N	N	Y	N	Y	N
8x8	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
8x4	Y	N	N	N	N	N	N	N	N	N	N
4x8	Y	N	N	N	N	N	N	N	N	N	N
4x4	Y	N	N	N	N	N	N	N	N	N	N
Logic Area (M gates)	7.9	6.1	2.44	4.27	1.83	2.44	1.22	4.27	1.83	2.44	1.22
Normalized Core Power	1.00	0.76	0.31	0.54	0.23	0.31	0.15	0.54	0.23	0.31	0.15
Ref. Buffer Size (KB)	680	565	248	439	208	234	163	356	170	201	115
On-Chip BW (GB/s)	1581	429	209	121	59	32.5	17.3	409	205	351	192
Off-Chip BW (GB/s)	159	69	30.2	27.4	12.7	8.5	5.1	64	28.7	49.1	25.1
Bit-Rate Increase (%)	0	2	3	12	12	34	34	3	4	7	11

Figure 2-6: Hardware cost vs. coding efficiency comparison table for 11 different motion estimation configurations. "Y" and "N" represents if a block size is supported or not respectively.

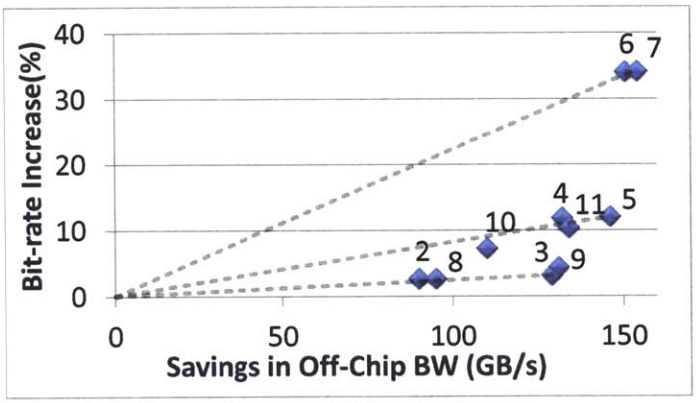
a graphical method to compare how efficient each configuration is. A smaller slope means that more savings can be achieved with smaller bit-rate increase (coding loss). Lines connecting configurations #3, #5 and #7 and the origin are given on Figure 2-7 as examples.

Observations and Conclusions

It can be observed from Figure 2-6 and Figure 2-7 that configurations supporting smaller block sizes such as 4×4 require largest area and bandwidth although the coding gain achieved through supporting them is relatively smaller. In other words, not supporting smaller partitions has a smaller effect on coding efficiency although these engines contribute significantly to bandwidth and area. For example, by re-



(a)



(b)

Figure 2-7: (a) Core area savings vs. bit-rate increase and (b) off-chip bandwidth savings vs. bit-rate increase scatter plots for all the configurations given in Figure 2-6.

moving 4x4, 4x8 and 8x4 block sizes in configuration #2, 17% memory area, 3.7× on-chip bandwidth and 2.3× off-chip bandwidth can be saved at the expense of only 2% coding loss.

Another observation from these figures is that not supporting $2N \times N$ and $N \times 2N$ does not result into significant coding efficiency loss. Fox example, from configuration #2 to #3, coding efficiency degrades by 1% and the degradation from configuration #4 to #5 and #6 to #7 are less than 1%.

On-chip reference buffer size mainly depends on the search range and block size. However, from smaller to larger block sizes, the increase in memory size is not very significant. In terms of memory bandwidth, small block sizes, especially smaller than

8x8, impose very high bandwidth requirements. If savings are necessary due to system level restrictions for bandwidth, small block sizes can be chosen not to be supported.

Lastly, final decision on supported block sizes depends on the area and bandwidth limitations as well as coding efficiency specifications of the target encoder. Since larger area and higher bandwidth often result in higher power consumption, battery-powered mobile applications might trade-off some of the coding efficiency for lower power consumption. If coding efficiency has the highest priority, all block sizes can be supported (configuration #1) although this might lead to a significantly large area and power consumption. If area as well as power are critical, configuration #5 and #7 are suitable solutions.

2.4 Cost and Coding Efficient (CCE) HEVC Motion Estimation Design

In Section 2.3, hardware cost of an HEVC motion estimation design providing a coding efficiency that is as high as the reference software implementation is discussed.

In this section we will talk about architecture and algorithm development for reducing the hardware cost even further with minimum impact on the coding efficiency. It should be noted that although the following algorithm and architecture developments are targeted for configuration #5, these algorithms and architectures are suitable for all configurations supporting square-shaped block sizes. At the end of this section, a comparison between five configurations (configurations #3, #5, #7, #9 and #11) will be provided.

2.4.1 Top Level Architecture

Top level architecture for CCE motion estimation module is given in Figure 2-8. Block sizes of 64×64, 32×32 and 16×16 are supported. Since there is only a single PU type ($2N \times 2N$) in each CU engine, an internal PU decision is not necessary.

It should be noted that this architecture is still capable of processing block sequen-

tially and consequently using exact motion information for the neighboring blocks.

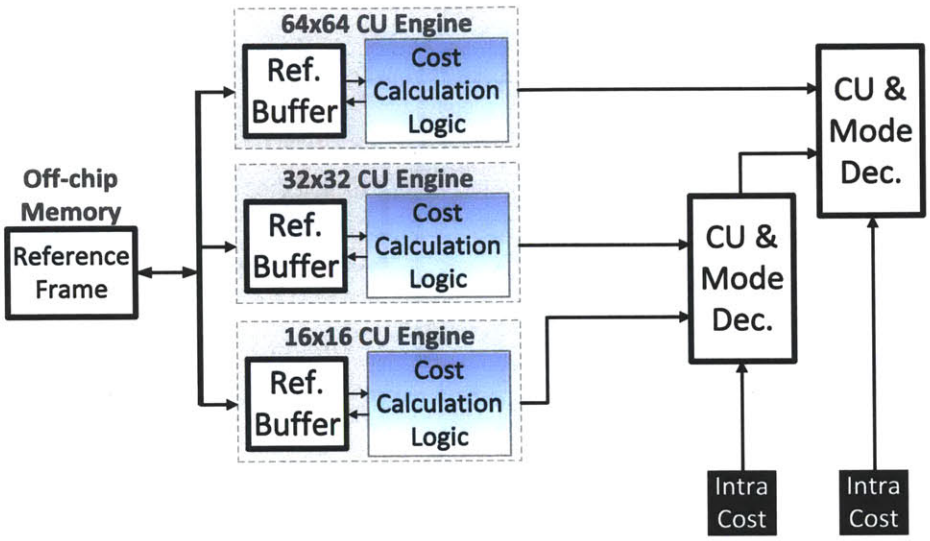


Figure 2-8: Top level architecture of the cost and coding efficiency motion estimation implementation. Block sizes of 64×64, 32×32 and 16×16 are supported.

2.4.2 Search Algorithm Development for CCE Motion Estimation

As discussed in Section 2.2, fast search strategy used in HM-3.0 starts the search around the best AMVP and consists of many inter-dependent stages. For example, the result of the initial diamond search determines if a sub-sampled raster search is performed or not. In hardware implementation, this dependency increases complexity and often results in extra cycles or extra hardware to account for the worst-case conditions.

Recent work focused on search algorithms that can be parallelized in hardware implementation [26, 31]. For CCE implementation, we implemented a similar, two-stage search strategy for IME where each stage can be independently performed in parallel. Figure 2-9 shows IME search patterns used in each of the stages. First, search center is decided by comparing AMVP list entries (up to three entries) and [0,0]. During this comparison, SAD (sum of absolute differences) cost is used. After search center is determined, two stage search is started.

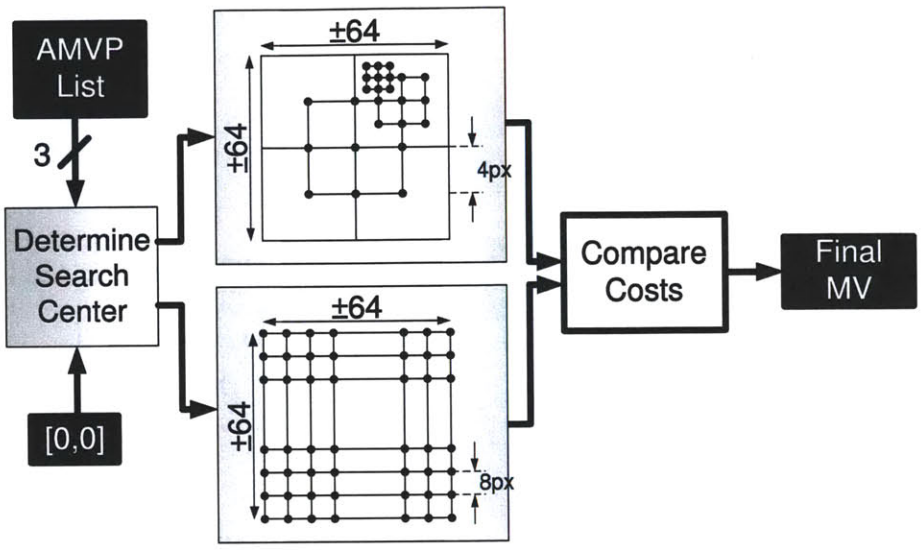


Figure 2-9: Two stage search approach used for CCE implementation. Stages are independent of each other and can be performed in parallel in hardware.

The first stage consists of a coarse search covering ±64 by checking every 8th candidate in each direction. This stage can capture a change in motion or irregular motion patterns that cannot be tracked by AMVP. The second stage performs a more localized three step search around the ±7 window of the search center. This stage can capture regular motion. It is important to note that the AMVP calculation for all blocks uses exact MVs of the neighbors and AMVP is accurate and hence can track motion well in most cases.

The proposed IME search strategy checks a total of 285 candidates for each block as opposed to up to 848 candidates that are checked in fast search strategy in HM-3.0. This results in roughly 2X hardware area reduction in IME for the same throughput constraint. Actual savings might be larger in implementation because of the additional complexity due to inter-dependent stages of HM-3.0 algorithm.

Lastly, for FME, search strategy of HM is used where refinement is performed around the best integer MV.

	LD	LDP	RA	Avg	Max	Min
HM-3.0 Anchor (Configuration #5)	0	0	0	0	0	0
Proposed Search Algorithm	0.6	0.8	1.6	1.0	3.1	0.1

Table 2.8: Simulation results for the coding efficiency change after the proposed search algorithm modifications with respect to HM-3.0 (configuration #5).

Effect on Coding Efficiency

Since the search algorithm for CCE motion estimation is more simple and checks a smaller number of candidates when compared to the reference software search algorithm, an increase in bit-rate is expected. Search algorithm for CCE motion estimation is implemented in the reference software and the effect on coding efficiency is quantified. Simulations are performed under the conditions defined in [12].

Table 2.8 shows coding efficiency change with respect to the HM-3.0 fast search algorithm in configuration #5 after the search algorithm for CCE modifications. Columns *LD*, *LDP* and *RA* stands for low-delay, low-delay with P and random-access test conditions as defined by JCT-VC [12]. *Avg* column is the average of *LD*, *LDP* and *RA*. Lastly, *Max* and *Min* columns are the maximum and minimum rate change for all tested sequences respectively.

The average rate increase due to the proposed changes is 1.0%. Random-access test conditions result in the largest coding efficiency degradation as the distance between the reference frame and the coded frame is longer in this test condition. The sequence with the highest coding efficiency loss is SteamLocomotive with 3.1%.

2.4.3 Sharing Reference Pixel Buffers for CCE Motion Estimation

As explained in Section Section 2.2, for maximum coding efficiency, each engine in Figure 2-8 is running independently and has separate reference buffers holding pixels for independent search windows. This approach is expensive in terms of area and external memory bandwidth. Sharing the on-chip reference buffer across parallel

	Separate Buf. (3 x 1R1W)	Shared Buf. (1 x 3R1W)
Memory Size	89KB	39KB
Est. Cell Area	$0.85\mu m^2$	$1.55\mu m^2$
Est. Array Area	$0.75mm^2$	$0.61mm^2$
Est. Perip. Area	$0.5mm^2$	$0.44mm^2$
Est. Total Area	$1.25mm^2$	$1.05mm^2$

Table 2.9: Area comparison of shared and separate reference buffers. Estimates are based on a 65nm CMOS technology.

engines can be significantly more efficient for practical implementations. However, restricting the search range of parallel engines to a shared window will result in coding efficiency loss. But, this loss can be minimized by determining the shared search window carefully.

Area Impact of Reference Buffers

In the case of separate reference buffers with ±64 search range for each engine, the implementation in Figure 2-8 requires three 1R1W (1 read, 1 write) port memories with 39KB, 27.5KB and 22.5KB sizes for 64 × 64, 32 × 32 and 16 × 16 engines respectively as given in Table 2.5. Total area consumed by these three memories can be estimated to be roughly $1.25mm^2$ in a 65nm CMOS technology [87] as shown in Table 2.9. It should be noted that this area is for storing the pixels on the chip for a single direction and single reference frame.

In the case of a shared reference buffer with ±64 search range, the size is determined by the largest block size. In this case, a single 39KB memory is needed with 3R1W ports. Although the bit-cell area and some peripheral components need to be expanded to support multiple read ports, the overall area can be smaller as shown in Table 2.9. Hence, shared search range across parallel engines results in 16% area savings for the implementation considered in Figure 2-8.

Block Size	Max. Off-Chip BW	Avg. Off-Chip BW
64x64	2.2GB/s	1.49GB/s
32x32	6.4GB/s	3.64GB/s
16x16	20.9GB/s	7.62GB/s
Total	29.5GB/s	12.39GB/s

Table 2.10: Maximum and average off-chip bandwidth requirement for different block sizes (search range is ± 64) for supporting $4K \times 2K$ at 30fps.

Data Bandwidth Impact of Reference Buffers

With independent motion searches, each engine might have different search centers and consequently access different parts of the reference frame as the search window. Table 2.10 shows maximum and average off-chip bandwidth for 64×64, 32×32 and 16×16 engines. The upper limit on the bandwidth is calculated by assuming that the entire on-chip reference buffer needs to be updated between consecutive blocks and hence no data re-use is possible. The total maximum off-chip bandwidth is 29.5 GB/s for supporting $4K \times 2K$ resolution at 30fps assuming a search range of ± 64. Average bandwidth number with close to 100% data re-use between consecutive LCUs is 12.39 GB/s. However, it should be noted that exactly 100% data re-use is not possible due to sequential processing of the blocks.

In the case of a shared reference window across engines, the maximum bandwidth is equal to the maximum bandwidth of the 64×64 block since the size of the shared search window is determined by the largest block size. Hence, sharing the search window provides 13.4X and 8.3X savings in terms of the maximum and average bandwidth requirements.

Strategy for Sharing Search Window

In order to minimize the coding efficiency impact of sharing search window across engines, a good representative should be selected for the motion of the all CUs within an LCU. AMVP of the LCU is observed to provide a good center point for the shared search window. Figure 2-10 shows the density map for the relative location of the

pixels from best matching blocks with respect to the AMVP of the LCU for two different sequences. Best matching blocks are calculated with the original HM-3.0 search algorithm and the search range is ±64 pixels in each direction. For both sequences, more than 99% of the best matching pixels lie in the ±64 vicinity of the AMVP of the LCU. This indicates that AMVP of the LCU can be used as the search window center without introducing significant coding efficiency loss.

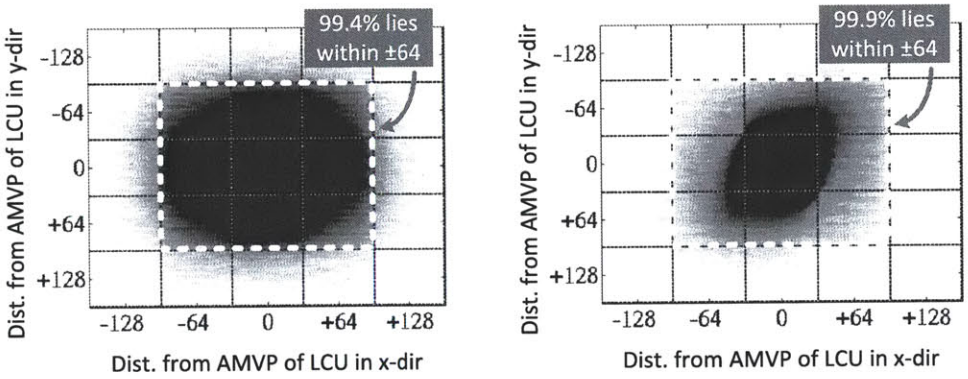


Figure 2-10: Density maps for the relative location of pixels from best-matching blocks with respect to the AMVP of the LCU for (a) PeopleOnStreet and (b) Traffic sequences. More than 99% of the pixels lie within ±64 of the AMVP of the LCU (2560x1600 sequences with QP=22 in random-access configuration).

For smaller blocks that have different AMVPs and consequently different search centers, the search window is modified to fit in the shared window. It is important to note that although the search window is modified, original AMVP of the block is used in cost calculations. Moreover, total number of candidates stay the same for all block sizes regardless of the search window being modified or not. This provides simplicity in hardware implementation.

Effect on Coding Efficiency

To see the effect of sharing search window across engines on the coding efficiency, changes are done in the reference software. Simulations are performed under the conditions defined in [12].

Table 2.11 shows coding efficiency change with respect to the HM-3.0 fast search

	LD	LDP	RA	Avg	Max	Min
HM-3.0 Anchor (Configuration #5)	0	0	0	0	0	0
Proposed Search Algo.	0.6	0.8	1.6	1.0	3.1	0.1
Proposed Search Algo. & Shared Search Window	0.6	1.0	2.9	1.5	7.4	0.2

Table 2.11: Simulation results for the coding efficiency after the search algorithm and shared search window modifications with respect to HM-3.0 (configuration #5).

algorithm after the search algorithm and shared search window modifications. Columns *LD*, *LDP* and *RA* stands for low-delay, low-delay with P and random-access test conditions as defined by JCT-VC [12]. *Avg* column is the average of *LD*, *LDP* and *RA*. Lastly, *Max* and *Min* columns are the maximum and minimum rate change for all tested sequences respectively.

The average rate increase due to sharing search window is an additional 0.5% on average. Similar to Table 2.8, random-access test conditions result in the largest coding efficiency degradation. Sequence with the highest coding efficiency loss is SteamLocomotive with 7.4%.

2.4.4 Reference Pixel Data Pre-fetching Strategy

For a practical hardware implementation, off-chip memories are used for large storage requirement of reference frames. It is necessary to request the data from off-chip memories in advance since the latency of these memories can be on the order of thousands of cycles. To address this, a pre-fetching strategy is implemented for CCE motion estimation in this work.

In order to share the cycles between writing to and reading from the reference buffer, a larger on-chip storage is necessary. This extra storage is used to start writing the data for the next LCU while motion estimation for current LCU is continuing. For this purpose, an extra storage that is 64 pixels wide (size of an LCU) is necessary as shown in Figure 2-11. Obviously, extra storage alone is not adequate if the search

center from current LCU to next LCU is changing. This issue can be addressed by allowing a larger storage for reference buffers and algorithm modifications. This will be discussed in Section 2.4.5.

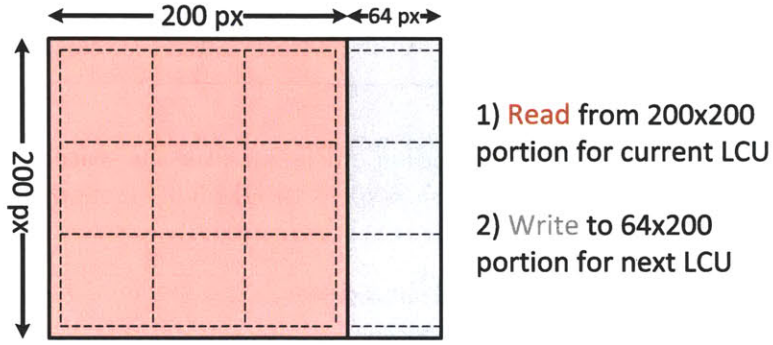


Figure 2-11: Extra storage is needed for on-chip buffers to share cycles for read and write accesses to the memories. 200×200 portion is used for current LCU and 64×200 portion is used for next LCU.

AMVP of the LCU is used to open the shared search range. However, AMVP calculation for current LCU depends on its left neighbor's motion data and cannot start until left neighbor's motion search is finalized. In this work, top (T), top-right (TR) and top-left (TL) neighbors of the current LCU are used to predict current LCU's AMVP and pre-fetch corresponding data from the off-chip memory.

The procedure to predict current LCU's AMVP is as follows:

- If none of the neighbors is available, data is pre-fetched from [0,0] location.
- If only one of the neighbors is available, AMVP of the available neighboring LCU is used to pre-fetch data.
- If two of the neighbors are available, AMVP of one of the available neighboring LCUs is used to pre-fetch data in the following precedence order:

$$T > TR > TL$$

- If all neighbors are available, median of the AMVPs of three neighbors is calculated and used to pre-fetch data.

	LD	LDP	RA	Avg	Max	Min
HM-3.0 Anchor (Configuration #5)	0	0	0	0	0	0
Proposed Search Algo.	0.6	0.8	1.6	1.0	3.1	0.1
Proposed Search Algo. & Shared Search Window	0.6	1.0	2.9	1.5	7.4	0.2
Proposed Search Algo. & Shared Search Window & Pre-fetch Strategy	0.9	1.0	2.9	1.6	7.3	0.2

Table 2.12: Simulation results for the coding efficiency change after the search algorithm, shared search window and pre-fetching modifications with respect to HM-3.0 (configuration #5).

With this strategy, data request can be sent to the off-chip memory as soon as the motion search of TR neighbor is completed.

Effect on Coding Efficiency

To see the effect of the pre-fetching strategy on the coding efficiency, changes are incorporated into the reference software. Simulations are performed under the conditions defined in [12].

Table 2.12 shows coding efficiency change with respect to the HM-3.0 fast search algorithm after the search algorithm, shared search window and pre-fetching strategy modifications. Columns *LD*, *LDP* and *RA* stands for low-delay, low-delay with P and random-access test conditions as defined by JCT-VC [12]. *Avg* column is the average of *LD*, *LDP* and *RA*. Lastly, *Max* and *Min* columns are the maximum and minimum rate change for all tested sequences respectively.

The average rate increase due to the pre-fetching strategy is an additional 0.1% on the average. Similar to Table 2.8 and Table 2.11, random-access test conditions result in the largest coding efficiency degradation. Sequence with the highest coding efficiency loss is SteamLocomotive with 7.3%.

2.4.5 Enlarging On-Chip Reference Buffers for Higher Data Reuse Rate

In the previous section, sharing the search window provided significant reduction in off-chip bandwidth since this approach enforces data to be used by the entire LCU. Further reduction in bandwidth can be achieved if data is reused between consecutive LCUs. In the ideal case where consecutive LCUs have the same AMVP, a 100% data reuse rate can be achieved where search window moves to the right by 64 pixels for every LCU. An illustration of 100% data reuse case is shown in Figure 2-12-a, where five LCUs and their corresponding search window are shown. However, this is highly unlikely and AMVP of consecutive LCUs can be very different from each other especially in frames with complex motion. Figure 2-12-b shows a case where data reuse between five LCUs is very poor.

In the discussion above, we always considered the case where on-chip reference buffer size is equal to the search window size and additional storage for the next LCU. However, if the on-chip memory size is increased to hold a larger window, data reuse rate can be improved as there is a higher chance of the data on the chip matching next LCU's search window. Although larger on-chip memories result in larger bandwidth per LCU, the improvement in data reuse rate can over-power this increase and results in a reduction in overall average bandwidth. It should be noted that although on-chip memories hold a larger window, search window is not increased and kept as ± 64 in each direction and consequently the total number of candidates in motion search is not affected from this modification.

The effect of increasing reference buffer size by N pixels on all four sides is analyzed in terms of bandwidth. Figure 2-13 plots total off-chip write bandwidth at the top plot and maximum data reuse rate and on-chip buffer size on the bottom plot for two different sequences with changing N. With increasing N, on-chip buffer size and the bandwidth due to updating a larger buffer for every LCU increase. However, also with increasing N, maximum data reuse rate increases. Because of these conflicting trends, write bandwidth makes a minimum around $N = 16$. This provides close to

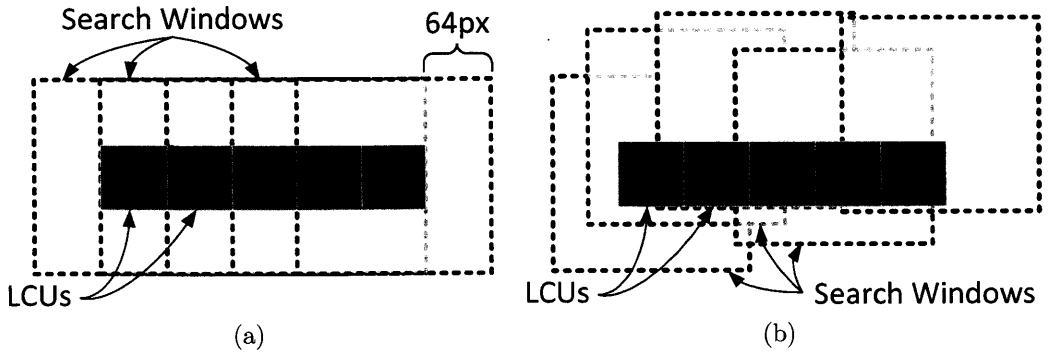


Figure 2-12: Search ranges of five consecutive LCUs with (a) uniform motion maximizing data reuse and (b) non-uniform motion causing lower data reuse rate.

1.8X savings in off-chip bandwidth at the expense of 35% area increase in reference pixel buffers.

To further improve data reuse rate and reduce off-chip bandwidth, pre-fetching algorithm is modified to limit the difference between two AMVPs (centers of search windows) to $\pm N$. Qualitatively, this translates to the search window being able to track changes in motion by at most N pixel step sizes.

In this case, amount of data that will be updated in reference buffers from current LCU to the next LCU is

$$64 \times (200 + 2 \times N)$$

in the best case and

$$64 \times (200 + 2 \times N) + N \times (200 + 2 \times N)$$

in the worst case.

For this work, N is chosen to be 16 to minimize its effect on the coding efficiency and to minimize total bandwidth.

Effect on Coding Efficiency

To see the effect on the coding efficiency of limiting the movement of search center to 16 in every direction between consecutive LCUs, changes are incorporated into the

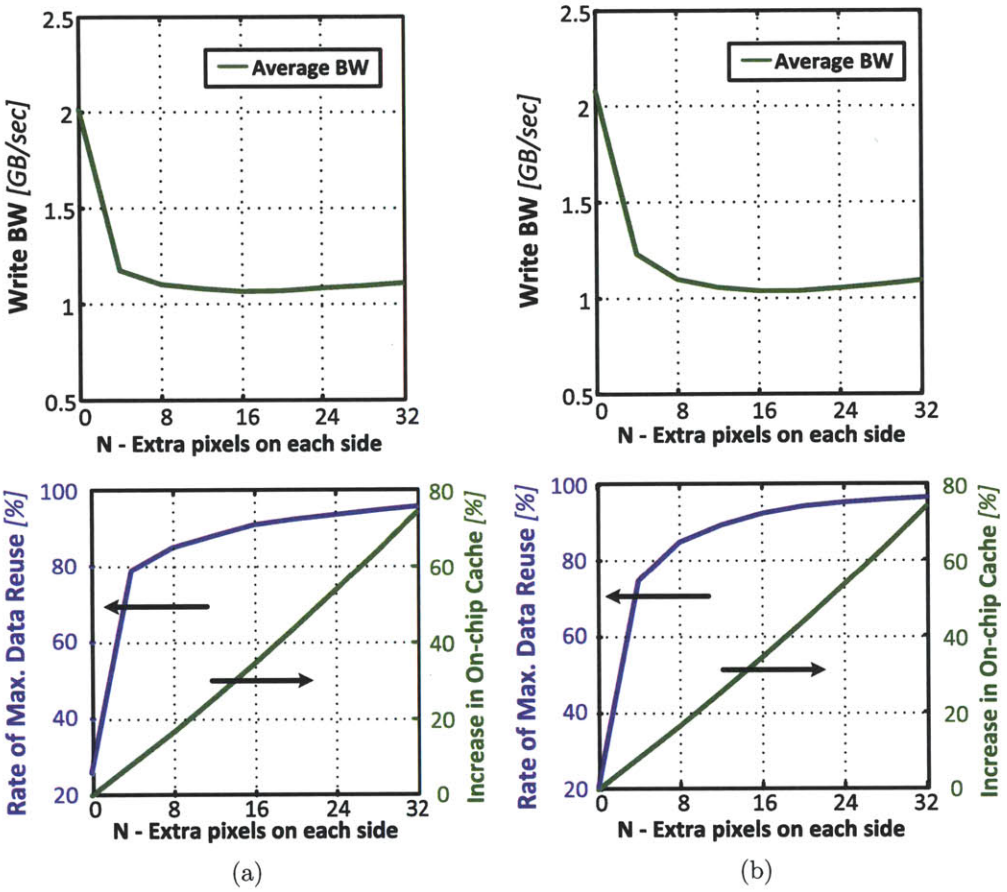


Figure 2-13: Total off-chip write bandwidth, maximum data reuse rate and on-chip buffer size for (a) BasketballDrive (1920 × 1080) and (b) Traffic (2560 × 1600) sequences. Simulations are performed in Random Access test condition with $QP = 22$.

reference software. Simulations are performed under the conditions defined in [12].

Table 2.13 shows coding efficiency change with respect to the HM-3.0 fast search algorithm after the search algorithm, shared search window, pre-fetching strategy modifications and limiting the movement of search range center by $N = 16$. Columns *LD*, *LDP* and *RA* stands for low-delay, low-delay with P and random-access test conditions as defined by JCT-VC [12]. *Avg* column is the average of *LD*, *LDP* and *RA*. Lastly, *Max* and *Min* columns are the maximum and minimum rate change for all tested sequences respectively.

The average rate is not increased due to final change of limiting the movement of search range center and this idea does not introduce additional coding efficiency

	LD	LDP	RA	Avg	Max	Min
HM-3.0 Anchor (Configuration #5)	0	0	0	0	0	0
Proposed Search Algo.	0.6	0.8	1.6	1.0	3.1	0.1
Proposed Search Algo. & Shared Search Window	0.6	1.0	2.9	1.5	7.4	0.2
Proposed Search Algo. & Shared Search Window & Pre-fetch Strategy	0.9	1.0	2.9	1.6	7.3	0.2
Proposed Search Algo. & Shared Search Window & Pre-fetch Strategy & Limited SR movement with $N = 16$	0.9	1.0	2.9	1.6	7.4	0.2

Table 2.13: Simulation results for the coding efficiency change after the search algorithm, shared search window, pre-fetching and limiting the movement of search range center by $N = 16$ with respect to HM-3.0 (configuration #5).

loss. However, the maximum coding efficiency loss increases from 7.3% to 7.4% for sequence SteamLocomotive.

2.5 AMVP Algorithm Development

AMVP algorithm used in HEVC is much more complicated compared to MVP defined for AVC/H.264. The complexity mainly comes from

- AMVP list generation procedure which does not allow hardware sharing and
- scaling used in AMVP candidate calculations which requires multiplication operations.

In this section, first, HM-3.0 algorithm and its drawbacks for hardware implementation will be discussed. Next, hardware-aware algorithm development for AMVP will be presented along with its effect on coding efficiency.

2.5.1 HM-3.0 Algorithm

Advanced Motion Vector Prediction (AMVP) list generation uses up to five spatial and up to two temporal neighboring blocks' motion information in AMVP calculations. Figure 2-14 shows all neighboring blocks. C_0 to C_4 are spatial neighbors at three corners of the current block. C and H are temporally co-located block and bottom-right neighboring block respectively from a previously coded frame.

AMVP calculation flow in HM-3.0 is explained below:

Step 1 Check left candidates (C_0 and C_1) for their availability. If both of them are available, C_0 has the precedence.

Step 2 Calculate an AMVP candidate ("scaled" or "not-scaled") from the available candidate of *Step 1* and store it as MVP_{LEFT}.

Step 3 Check each of the top neighbors (C_2, C_3, C_4) sequentially for their availability.

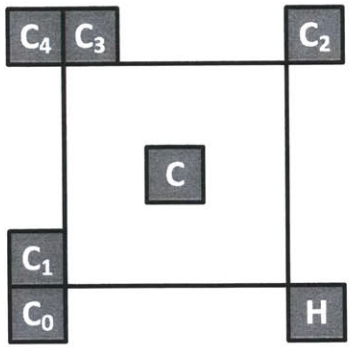


Figure 2-14: Five spatial and two temporal neighbors are used in AMVP calculation process.

Step 4 Calculate two predictors (one "scaled" and one "not-scaled") from each of the available neighbors, $MVP_T[i]$, $i = 0, 1, .., 5$.

Step 5 Check if $MVP_T[i]$ is equal to MVP_{LEFT} and store the first $MVP_T[i]$ as MVP_{TOP} that is not equal to MVP_{LEFT}.

Step 6 Check temporal neighbors (C and H) for their availability. If both of them are available, C has the precedence.

Step 7 Calculate an AMVP candidate ("scaled" or "not-scaled") from the available candidate of *Step 6* and store it as MVP_{TEMP}.

Step 8 "Uniquify" AMVP list by discarding candidates that are equal to other candidates.

From the AMVP calculation flow in HM-3.0 given above, it should be noted that there are many dependencies that make hardware implementation more complex. Moreover, these dependencies prevent hardware to be shared between candidates. As "scaling" is a complicated part accounting for a large fraction of total area, we can infer how many *Scale Hardware* is necessary in an implementation. It should be noted that the cycle constraint is very tight in a hardware implementation and hence AMVP calculation needs to be performed in one or two clock cycles which would refrain us from using sequential calculation of candidates.

From *Step 2* and *Step 7*, we can infer that one *Scale Hardware* per candidate (MVP_{LEFT} and MVP_{TEMP}) is adequate since one of the neighbors can be selected based on availability and the precedence order and input to the *Scale Hardware*.

However, for *Step 5*, it is not possible to select which candidate will be "scaled" as the "scaled" candidate is required to be different than MVP_{LEFT}. Consequently, three separate *Scale Hardware* is necessary for MVP_{TOP}.

A total of five *Scale Hardware* is necessary for this implementation as shown in Figure 2-15.

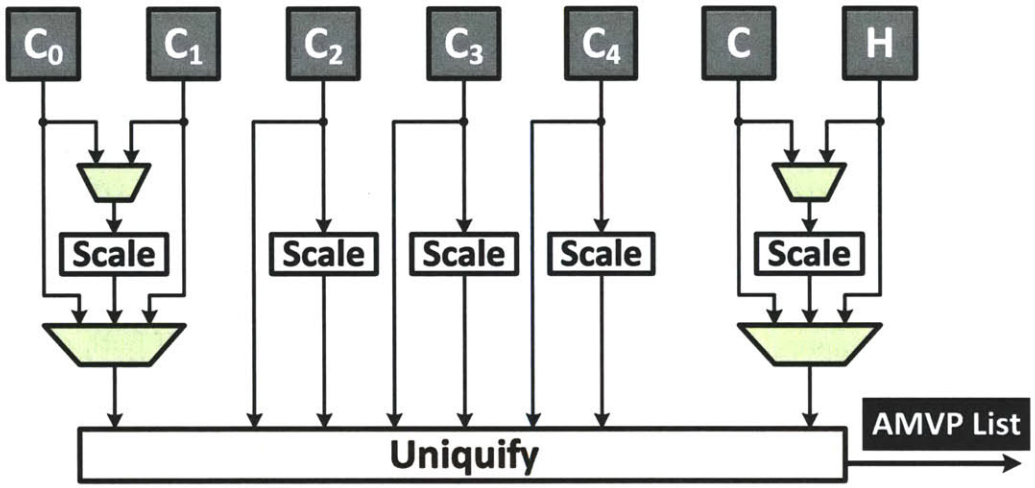


Figure 2-15: Block diagram of an hardware implementation of AMVP calculation algorithm given in HM-3.0.

2.5.2 Proposed Algorithm

The dependency explained above results in three separate *Scale Hardware* for only one AMVP candidate calculation. This analysis provides an insight about how the AMVP algorithm can be changed to reduce the number of *Scale Hardware* blocks and reduce overall area of AMVP.

AMVP calculation flow in the proposed algorithm is explained below:

Step 1 Check left candidates (C_0 and C_1) for their availability. If both of them are available, C_0 has the precedence.

Step 2 Calculate an AMVP candidate ("scaled" or "not-scaled") from the available candidate of *Step 1* and store it as MVP_{LEFT}. If a "scaled" candidate is used in this step, set $scale\_flag$ to "1".

Step 3 Check each of the top neighbors (C_2, C_3, C_4) sequentially for their availability.

Step 4 Calculate up to two predictors (one "scaled" if $scale\_flag$ is "0" and one "not-scaled") from each of the available neighbors, $MVP_T[i]$, $i = 0, 1, .., 5$. If a "scaled" candidate is used in this step, set $scale\_flag$ to "1".

Step 5 Check if $MVP_T[i]$ is equal to MVP_{LEFT} and store the first $MVP_T[i]$ as MVP_{TOP} that is not equal to MVP_{LEFT}.

Step 6 Check temporal neighbors (C and H) for their availability. If both of them are available, C has the precedence.

Step 7 Calculate an AMVP candidate ("scaled" or "not-scaled") from the available candidate of *Step 6* and store it as MVP_{TEMP}.

Step 8 "Uniquify" AMVP list by discarding candidates that are equal to other candidates.

By the introduction of the $scale\_flag$, this proposed algorithm ensures a single "scaling" for MVP_{LEFT} and MVP_{TOP}. MVP_{TEMP} does not check for $scale\_flag$ and hence requires an additional *Scale Hardware*. Figure 2-16 shows the block diagram of the proposed AMVP algorithm. Total number of *Scale Hardware* is two in this case, resulting in significant reduction of hardware area.

Hardware implementation details and the comparison of HM-3.0 and proposed algorithm results will be discussed in Section 2.6.3.

2.5.3 Effect on Coding Efficiency

To see the effect of the proposed AMVP algorithm on the coding efficiency, changes are incorporated into the reference software. Simulations are performed under the conditions defined in [12].

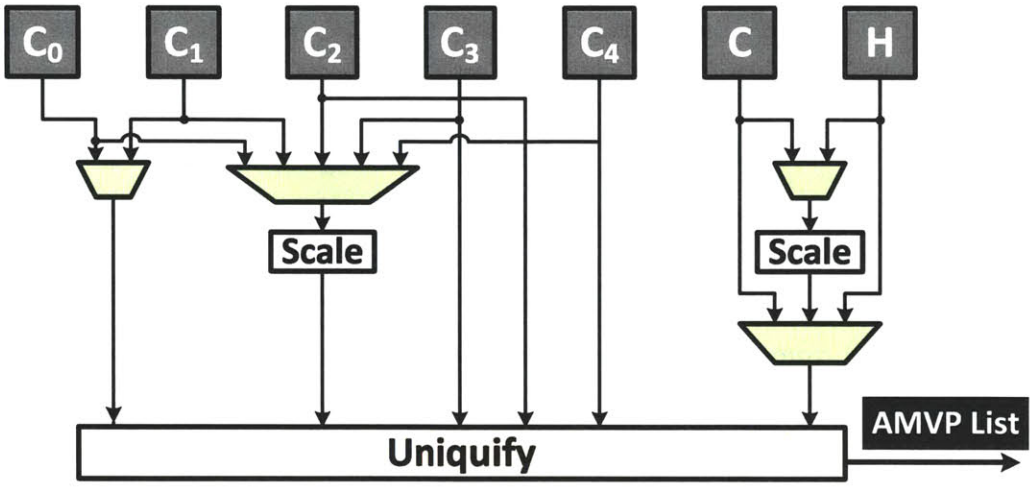


Figure 2-16: Block diagram of an hardware implementation of proposed AMVP calculation algorithm.

	LD	RA	Avg	Max	Min
HM-3.0 Anchor	0	0	0	0	0
Proposed AMVP Algo.	0	0	0	0.1	-0.2

Table 2.14: Simulation results of the coding efficiency change with the proposed changes in AMVP algorithm with respect to HM-3.0.

Table 2.14 shows coding efficiency change with respect to the HM-3.0 AMVP algorithm. Columns *LD* and *RA* stands for low-delay and random-access test conditions as defined by JCT-VC [12]. *Avg* column is the average of *LD* and *RA*. Lastly, *Max* and *Min* columns are the maximum and minimum rate change for all tested sequences respectively.

The effect on coding efficiency is negligible for the proposed algorithm as most sequences show no increase in bit-rate and the largest bit-rate increase is only 0.1% for the *BasketballPass* sequence.

2.6 Hardware Implementation Results for CCE HEVC Motion Estimation

Table 2.15 provides a comparison between different configurations in terms of various hardware costs: area with respect to configuration #1 given in Figure 2-6, on-chip memory area, on- and off-chip memory bandwidth and bit-rate increase. Bit-rate increase is due to (*i*) proposed hardware-efficient algorithms and (*ii*) unsupported block sizes with respect to configuration #1 in Figure 2-6.

Configuration	**#1**	**#3**	**#5**	**#7**	**#9**	**#11**
Supported Block Sizes	ALL	64×64 32×32 16×16 8×8	64×64 32×32 16×16	64×64 32×32	32×32 16×16 8×8	16×16 8×8
Area w.r.t. Conf. #1	1	0.31	0.23	0.15	0.23	0.31
On-chip Mem. Area (mm^2)	7.8	1.7	1.5	1.3	1.5	1.3
On-chip Bandwidth (GB/s)	1581	100.5	28.6	8.5	98.3	92.1
Off-chip Bandwidth (GB/s)	159	1.05	1.05	1.05	2.92	9.72
Bit-rate Increase w.r.t Conf. #1	0%	4.6%	13.6%	35.7%	4.7%	11.8%

Table 2.15: Comparison of configurations #3, #5, #7, #9 and #11 to the main anchor (configuration #1 in Figure 2-6) in terms of power, memory area, bandwidth and coding efficiency.

Configuration #7 provides the smallest area and bandwidth but the bit-rate increase is 34% which can be very high for many applications. Off-chip bandwidth requirement of configuration #11 is 9× larger due to the LCU size being smaller which requires more data to be transferred to on-chip buffers for every frame. Configuration #3 and #9 have 12× larger on-chip bandwidth due to supporting 8×8 block.

Based on the analysis explained above, CCE motion estimation with three engines (64×64, 32×32 and 16×16) is the optimum selection for the target area, bandwidth and coding efficiency results. This selection resulted in 4.3× core area and 5.2×

on-chip buffer area reduction. In terms of bandwidth, savings are 56× for on-chip bandwidth and 151× for off-chip bandwidth. Overall coding efficiency loss is 13.6% for our selection.

However, it should be noted that different designs can have different design budgets. For example, coding efficiency can be very critical in one design or power consumption can be the decisive factor in another design. Hence, final decision on supported block sizes can be different but the analysis provided here can be used to make this decision based on quantitative results.

2.6.1 Implementation of a CU engine

Figure 2-17 shows the architecture of one engine. Integer and fractional motion estimation parts are implemented together as these parts are not pipelined for maximum coding efficiency.

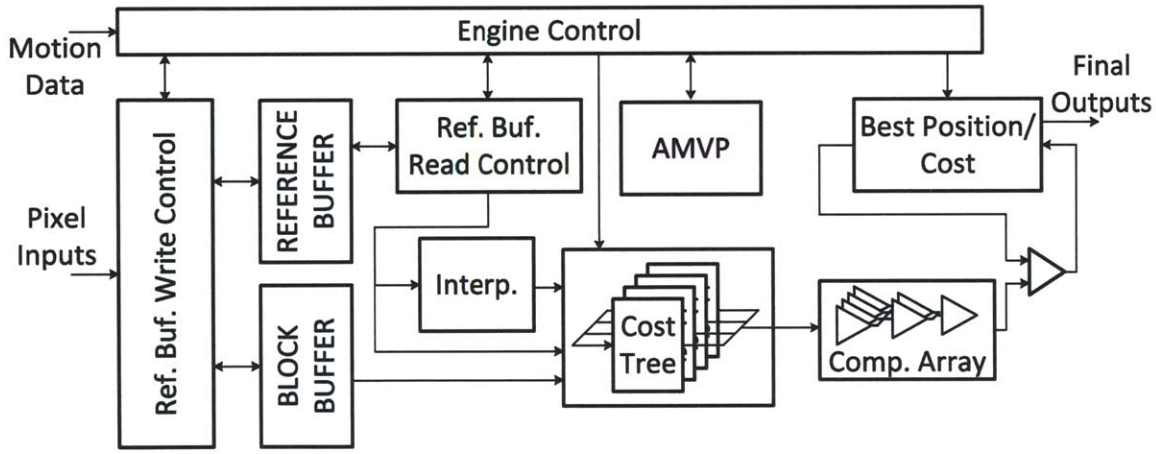


Figure 2-17: Architecture of one engine in CCE HEVC motion estimation implementation.

Reference buffer and block buffer hold reference and current block's data respectively. Reference buffer write control exerts write operations on the reference buffer for the next LCU whereas read control accesses the search range data. AMVP part calculates the motion vector predictor list. Cost tree and comparator array is capable of calculating the cost of 4 candidates/cycle for the 16 × 16 block for which the cycle budget is shortest. Best position and cost is stored in sequential elements and

compared against costs for newer candidates. Finally, engine control ensures the flow of data inside the engine as well as the communication of higher level control units.

2.6.2 Reference Buffer and Read/Write Control

To be able to support the 4 candidates/cycle output requirement, search range is partitioned into 88 blocks of SRAMs each holding 4 neighboring pixels on every word and holding roughly 200 words. Figure 2-18 shows the allocation of pixels on memory banks.

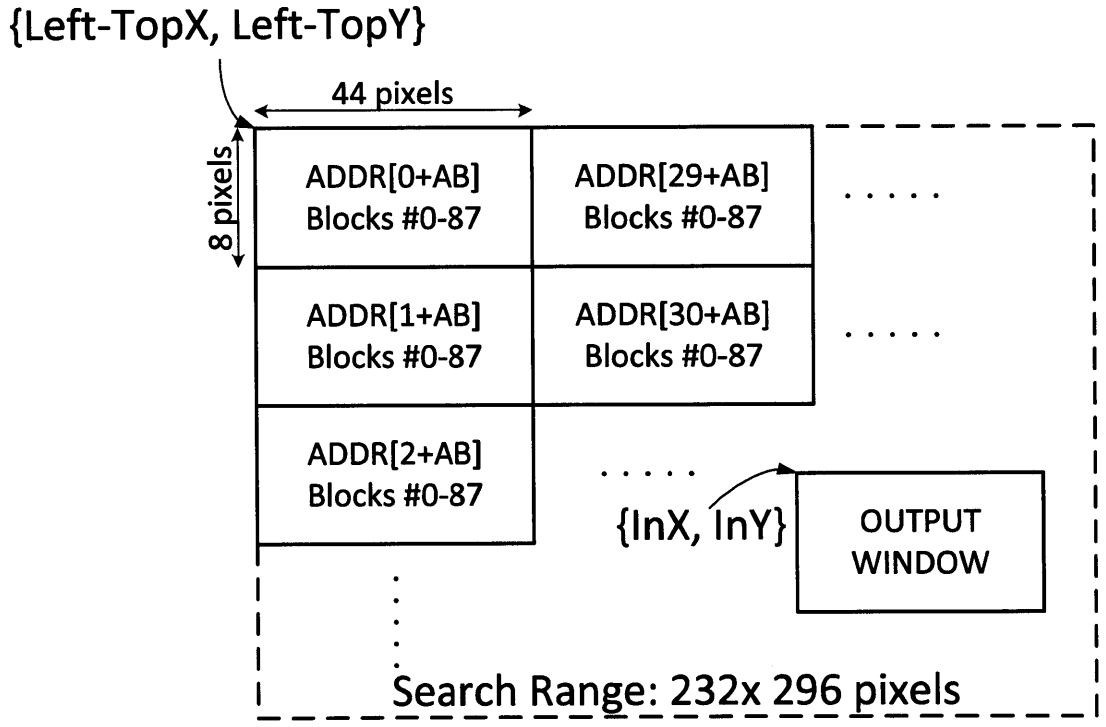


Figure 2-18: Search range partitioning and physical location of pixels in memory banks.

Going from one LCU to the next, since most of the data is reused, only pointers to the memory locations are changed. This is handled in the read control by holding the left-top coordinate (Left-TopX, Left-TopY) of the search range as well as an address bias (AB) which is incremented by 64 pixels for every LCU. Engine control requests a stripe (8×40) of reference pixels by providing the left-top coordinate (InX,InY) to read control. After data is read from SRAM blocks, 8×40 pixel block is output in

the next cycle. There is a large multiplexer array at the output of the read control to select appropriate outputs from SRAM blocks and put them in order.

New data overwrites the older data sequentially for every LCU. At the beginning of an LCU line in the frame, all memory locations need to be updated. For all other LCUs, a 64×232 block and possibly 16 pixel wide edges are updated since, at the algorithm level, the movement of the search center is limited to be less than 16 pixels between consecutive LCUs. Since the search range accessed by the read control and the pixels that are overwritten by write control are not overlapping, read and writes can be done in the same cycle.

Synthesis results for the reference buffer read and write show that a total of 52.6K gates is used. Read control takes up a larger area due to the large multiplexers to select the outputs from 88 SRAM blocks.

2.6.3 AMVP Implementation

Proposed AMVP algorithm is adopted by the HEVC standard so this proposed algorithm is implemented in the hardware engine. AMVP list generation consists of a left, a top and a temporal candidate generation. The scaling operation consisting of multiplications that account for a large fraction of the total area. Moreover, the delay through these multiplication units is relatively larger.

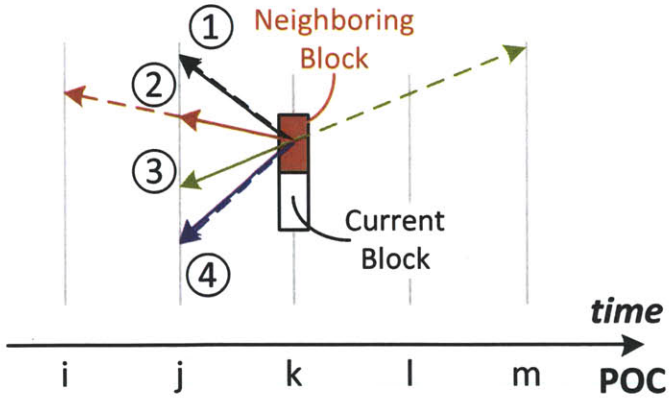


Figure 2-19: Illustration of MV scaling. MVs from different reference frames are used directly or scaled according to the POC order of the frames.

Figure 2-19 shows the scaling procedure for AMVPs [82]. POC stands for picture

order count and denotes the location of a frame in the display order. By using the difference of the POCs, MVs can be scaled up or down and their directions can be changed. This is very useful as this information can be used to generate various predictions to the MV and the best prediction can be selected eventually. In Figure 2-19, MVs from neighboring block denoted as '1' and '4' are not scaled but '2' and '3' are scaled.

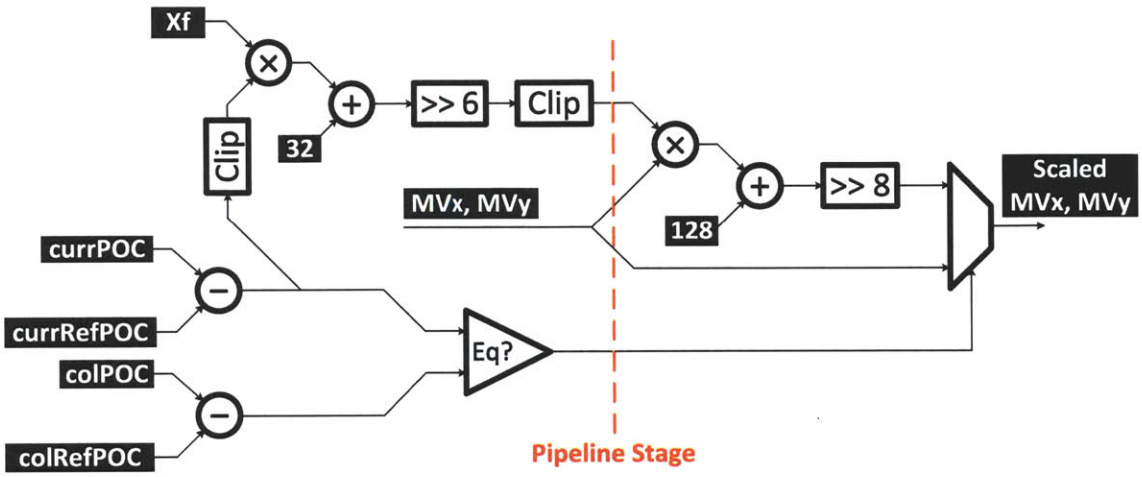


Figure 2-20: Implementation of the scaling unit in AMVP.

Figure 2-20 shows the implementation of the scaling unit in AMVP calculations. If the difference of POCs is equal to each other, then no scaling is needed and original MVs can be passed to the output. Otherwise, MVs are scaled. It should be noted that the scaling term (Xf) is signed and can alter the sign of the MVs. After each multiplication, a rounding and clipping is done. It should also be noted that Xf calculation requires a division operation which is implemented with a look-up-table but Xf calculation can be done for different POCs beforehand and can be used as a constant during AMVP calculations. Finally, two multiplication operations are pipelined in hardware as shown in Figure 2-20 as the critical path occurs across these two multiplier units.

The resulting area of the AMVP implementation is 26.1K gates. RTL code for the HM-3.0 algorithm that is replaced with our proposed algorithm is also written and area numbers are acquired. Table 2.16 shows area numbers for both implementations.

	Left Cand.	Top Cand.	Temp. Cand.	Uniquify	Total
HM-3.0	12.1K	26.8K	7K	5.3K	51.2K
Proposed	15.2K		7.4K	3.4K	26.1K

Table 2.16: Comparison of HM-3.0 and proposed AMVP algorithms in hardware implementation in number of gates.

As explained in Section 2.5, HM-3.0 algorithm requires three scale units in top candidate generation. This can be seen by the large area required for top candidate in Table 2.16. In the proposed algorithm, total area for left and top candidate generation is only 15.2K gates. Total area of the proposed algorithm is roughly half of the HM-3.0 implementation.

2.6.4 Cost Tree and Comparator Array

As shown in Figure 2-17, cost tree calculates SAD based costs and adds the MV cost to create total motion cost. Then, comparator array compares costs of candidates with the smallest cost and decides if the smallest cost needs to be updated or not. At the end of the search, smallest cost and its corresponding candidates are signaled as MVs.

Figure 2-21 shows the SAD tree and MV cost calculation. 1-bit partial absolute-differences (AD) are calculated and 1-bit 'msb' information is propagated to the output to make the critical path shorter. ADs and msb bits from multiple pixels are summed in parallel. MV cost calculation is implemented with a priority encoder as shown in Figure 2-21. The input to the priority encoder is the absolute difference of the MV and MVP. Lastly, cost tree and comparator array implementation results in 131K gates.

2.6.5 Interpolation Engine

Interpolation engine in HEVC requires a larger area due to the usage of longer FIR filters and larger filter coefficients. For example, for quarter-pel pixel generation,

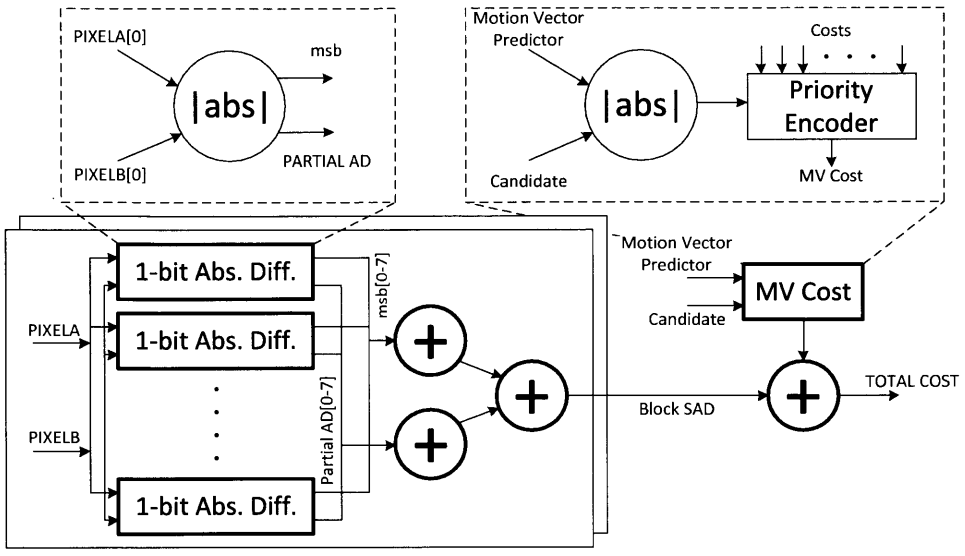


Figure 2-21: Cost tree implementation using 1-bit absolute difference (AD) and MV cost calculation.

AVC/H.264 used 2-tap filters. These filters are replaced with 8-tap filters in HEVC. Table 2.17 shows a comparison of filters used in AVC/H.264, VC-1 and HEVC to make a comparison in terms of area. HEVC's 8-tap quarter-pel filter is $> 20\times$ larger than AVC/H.264's 2-tap quarter-pel filter.

Filter	Coefficients	Area in K gates
AVC/H.264 Half-Pel	[1 -5 20 20 -5 1]	0.43
AVC/H.264 Quarter-Pel	[1 1]	0.05
VC-1 Half-Pel	[-1 9 9 -1]	0.41
VC-1 Quarter-Pel	[-4 53 18 -3]	0.30
HEVC Half-Pel	[-1 4 -11 40 40 -11 4 -1]	0.57
HEVC Quarter-Pel	[-1 4 -10 57 19 -7 3 -1]	1.28

Table 2.17: Comparison of interpolation filters used in AVC/H.264, VC-1 and HEVC standards. These filters are implemented with bit-wise shifts and additions.

It should be noted that single filter area is not the only measure of complexity as the the order of filtering and when clipping is done after filtering also determine overall complexity.

For our design, FME and IME are sequential so cost calculation tree and com-

parator array can be shared between the two parts of the design. FME design is done based on the work in [29]. Input data is provided as a row of pixels and shifted through the processing unit. The partial outputs are stored in a register array and cost calculation is performed when a candidate is generated. Implementation of the interpolator results in a total area of 137K gates.

2.7 Summary and Conclusions

HEVC is the next generation video compression standard that is currently being standardized. HEVC has a design target of achieving 50% coding efficiency gain over AVC/H.264 through various coding efficiency enhancement tools that often come at the expense of increased hardware complexity. Motion estimation, being the largest block in encoder designs, is one of the most critical blocks in HEVC and should be analyzed for its hardware implementation cost to provide a trade-off study in order to make critical design decisions.

This chapter provides a hardware cost vs. coding efficiency trade-offs analysis for

- a motion estimation implementation targeting highest coding efficiency that is equivalent to the reference software and
- a hardware-oriented cost and coding efficiency implementation.

Specifically, a motion estimation implementation providing a reference software-equivalent coding efficiency is considered for its area, on- and off-chip bandwidth and on-chip memory area. It has been shown that this implementation requires nearly 8M gates and 1.6TB/s and 159GB/s on- and off-chip bandwidth respectively. Clearly, this design is very costly in terms of hardware. To reduce hardware cost, first, a reduction in the number of coding engines (and consequently number of supported block sizes) is considered and quantitative analysis has been performed to find the configuration providing the best trade-off. It should be noted that the methodology used in this analysis can be generalized to compare various block sizes for different types of costs.

To further reduce hardware cost, hardware-oriented algorithms are developed that are suitable and targeted for the selected architecture. Overall, 56× on-chip bandwidth, 151× off-chip bandwidth, 4.3× core area and 5.3× on-chip memory area savings are achieved through these algorithm optimizations when compared to a hardware implementation of the HM design providing reference software-equivalent coding efficiency. For AMVP, a new algorithm is also proposed based on hardware implementation costs and proposed algorithm is shown to reduce area by 2× without introducing any coding efficiency loss.

It should be noted that this chapter is one of the first studies on HEVC motion estimation in the literature considering hardware implementation and the trade-off analysis. Hardware-oriented algorithm development and hardware implementation results can be used to make more optimized design decisions for an HEVC encoder.

Chapter 3

Highly Parallel Motion Estimation Design for Multi-Standard Video Encoder

This chapter will discuss a highly parallel motion estimation design that was part of a multi-standard video encoder supporting AVC/H.264 and VC-1 standards. The target of this design is achieving low-voltage and low-frequency operation while supporting a high throughput requirement and providing reconfigurability by supporting two different standards. To achieve these goals, a highly-parallelized and reconfigurable design is studied here and design decisions are made to achieve a high level of parallelism.

Due to the presence of various video coding standards, it can be necessary to support multiple standards for mobile multimedia device. On one hand, implementing different standards in software can be inefficient in terms of power consumption. On the other hand, having separate hardware blocks for different standards can be very costly in terms of area. To address these issues, this work targets a video encoder design supporting two standards (AVC/H.264 and VC-1) while using circuit reconfigurability to maximize hardware sharing across standards.

3.1 Overview of a Multi-Standard AVC/H.264 and VC-1 Encoder Design

3.1.1 Overview of AVC/H.264 and VC-1

AVC/H.264 and VC-1 are recent standards and successors of widely used MPEG-2 standard. These standards are developed to provide higher coding efficiency. Both standards perform block-based video coding and unit block size is 16×16 and named as a macro-block (MB). A MB can be predicted with inter- or intra-prediction.

For inter-prediction, a MB can be predicted with various modes. Figure 3-1 shows different modes supported in AVC/H.264 and VC-1 standards. AVC/H.264 supports sub-block sizes as small as 4×4 so a MB can be represented by as many as 16 different MVs. For VC-1, there are only two modes and a MB can be represented by either a single MV or by four MVs.

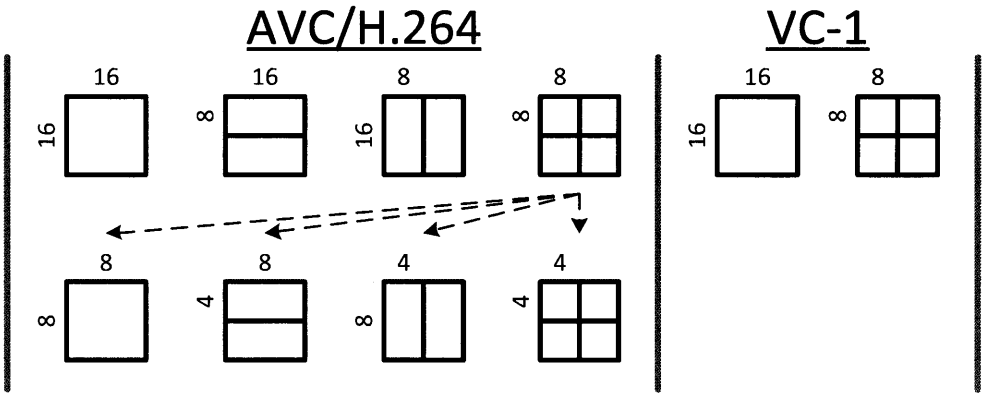


Figure 3-1: Inter modes supported by AVC/H.264 and VC-1 standards.

As mentioned above, AVC/H.264 and VC-1 standards provide better coding efficiency over MPEG-2 at the expense of higher complexity in hardware implementation. Table 3.1 shows some key features supported in AVC/H.264 and VC-1 standards that are related to motion estimation.

From Table 3.1, it can be seen that both standards have common features allowing sharing of hardware between standards for a reconfigurable motion estimation module. Specifically, calculation of cost metric and on-chip pixel buffers which constitutes the

Feature	H.264/AVC	VC-1
P Frames	Yes	Yes
B Frames	Yes	Yes
Number of Reference Frames	Up to 16 in each direction	Up to 2 in each direction
MB Sub-partitions	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4	16x16, 8x8
Fractional Motion Estimation	Quarter-pixel	Quarter-pixel
Luma Half-pixel Interpolation Filter	[1 -5 20 20 -5 1]/32	[-1 9 9 -1]/16
Luma Quarter-pixel Interpolation Filter	[1 1]/2	[-4 53 18 -3]/64
Chroma Interpolation Filter	[1 1]/2	[1 1]/2

Table 3.1: Comparison of H.264/AVC and VC-1 standards for supported features related to motion estimation

dominant portion of area can be shared. Some parts of mode selection and pixel interpolation can be implemented separately.

3.1.2 Specifications of the Multi-Standard Encoder

In a video encoder hardware implementation, resolution, frame-rate and other specific features of the standards set a throughput constraint on the encoder hardware. Table 3.2 summarizes the specifications for the multi-standard encoder design.

Both H.264 and VC-1 only support MB size of 16×16 pixels. Thus, the number of cycles available to process one MB can be found as

$$\text{Number of MBs per frame} = N = \frac{3840 \times 2160}{16 \times 16} = 32400 \tag{3.1}$$

$$\text{Number of MBs per second} = N \times \text{frame-rate} = 32400 \times 30 = 972000 \tag{3.2}$$

$$\text{Number of cycles per MB} = \frac{\text{Operation frequency}}{\text{Number of MBs per second}} = \frac{25 \times 10^6}{972000} = 25.7 \tag{3.3}$$

Feature	Target
Supported standards	H.264/AVC High Profile and VC-1 Main Profile
Resolution	Quad-HD (3840 × 2160 pixels)
Frame-Rate	30 frames/sec at quad-HD
GOP Structure	I B B P B B P
Number of Reference Frames	1 in each dir.
Operating Voltage	0.5V
Operating Frequency	25MHz
Bit-rate Increase w.r.t. Golden Encoder	25%

Table 3.2: Specifications for the multi-standard encoder project.

At 25MHz, processing a quad-HD video at 30fps requires every MB to be completed in about 25 cycles. By using pipelining, different processing parts such as intra-prediction, motion-estimation, transformation and quantization can be separated in time. Moreover, parallelism can be used to loosen this cycle constraint. For motion estimation, reference data needs to be loaded from an off-chip memory and then motion search should be done on this reference data. The 25 cycles constraint requires excessive replication of hardware modules. To address this problem, frame and MB level parallelism is used in this work. Six MBs are processed in parallel so roughly 150 cycles are available to process each MB in parallel.

In contrast to previous designs, this work proposes a highly-parallel implementation at the frame and MB level to provide necessary throughput at 25MHz for quad-HD video encoding at 30fps. Total bit-rate increase budget with respect to the golden encoder implementation (given in the reference software) is set as 25%.

3.2 Algorithm Development and Architecture Selection for Highly-Parallel Motion Estimation

3.2.1 Frame and MB Parallel Video Encoding Architecture

In order to be able to meet the throughput constraint discussed in the previous section, parallelism is implemented in frame and MB level. By doing frame-parallel and MB-parallel motion search in motion estimation

- off-chip bandwidth can be greatly reduced if all MBs are using the same search range from the same reference frame and
- available cycles to perform motion search can be greatly increased as parallel processing loosens the cycle constraint for each MBs.

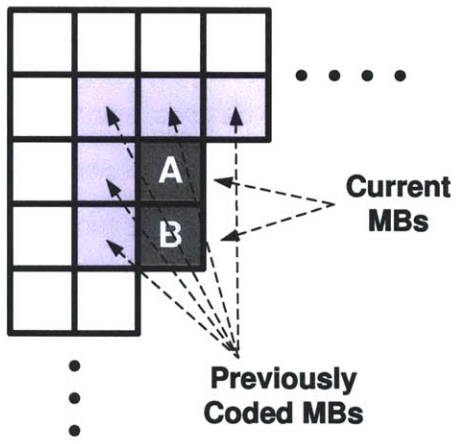


Figure 3-2: MB parallel implementation. Two MBs from each frame is processed in parallel.

Figure 3-2 shows the position of two MBs that are processed in parallel. Since both MBs are spatially located close to each other, they can share the same search-range and same reference pixel data can be used to calculate costs for each MB at the same time. This provides 2× parallelism. The number of parallel MBs from the same frame can be further increased for higher levels of parallelism. However, this would require storing more lines of data on- or off-chip as final bit-stream is created in the

Frame # in Sequence	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Frame ID	I_0	B_1	B_2	P_3	B_4	B_5	P_6	B_7	B_8	P_9	B_{10}	B_{11}	P_{12}

Table 3.3: GOP structure used in this work.

raster scan order and cannot be parallelized. Hence, selection of two MBs provides a good balance between the level of parallelism and the amount of additional storage requirement.

GOP structure used in this design is given in Table 3.3 and motion search order is given in Table 3.4.

Motion search cannot start until the first I frame (all-intra frame) is encoded and reconstructed. Then during t_1, frame I_0 is used as reference and motion search is performed on six MBs from P_3, B_1 and B_2. After this, frame P_3 can be encoded and reconstructed whereas frames B_1 and B_2 wait for time slot t_2. During t_2, P_3 is the reference frame and B_1, B_2 undergo forward search and P_6, B_4 and B_5 undergo backward search. It should be noted that B frames are not used as reference frames. After time slot t_2, a total of ten MBs (two MBs, each from five frames) share the same reference frame. This provides an additional 5× parallelism, resulting in 10× parallelism for motion estimation. As the search range is shared for ten MBs, off-chip bandwidth is reduced significantly.

Time Slot	**Reference Frame**	**Forward Search**	**Backward Search**
t_0	N/A	N/A	N/A
t_1	I_0	N/A	P_3, B_1, B_2
t_2	P_3	B_0, B_1	P_6, B_4, B_5
t_3	P_6	B_4, B_5	P_9, B_7, B_8
t_4	P_9	B_7, B_8	P_{12}, B_{10}, B_{11}

Table 3.4: Processing order of frames. After the initial I-frame and first P-frame, 10 MBs from five frames are motion searched in parallel.

3.2.2 Searching Algorithm Selection

Searching algorithm is a key part of the motion estimation as it determines the power consumption and on/off-chip bandwidth for the module. In this work, the cycle budget for motion estimation is very strict. To make a comparison, the work in [27] operates at 145MHz for 1920×1080 resolution whereas this work targets 25MHz operation for 3840×2160. Hence, more than $20\times$ reduction in cycle count is necessary. Frame and MB-parallel processing enables amortizing the cycle count over many MBs but an efficient searching algorithm is necessary to prevent quality loss.

Integer Motion Estimation (IME) Search Algorithm

In this work, a searching algorithm suitable for frame and MB parallel motion estimation is developed. First, since two MBs are processed simultaneously, they can share the same searching range as they are located on top of each other (Figure 3-2). So the cost of each candidate MV can be calculated for both current MBs at the same time. Second, for all sequences, a significant portion of MVs are centered around origin since in general many MBs in a frame do not move at all. So searching around [0,0] point can capture the MVs around origin. For moving objects, however, MVs can have a large magnitude. To capture these MVs, a prediction-based searching range can be utilized based on the work in [27]. Specifically, a searching range center can be calculated by using the MVs of neighboring blocks.

To be able to remain in the cycle count, the window around origin is searched by sub-sampling by four i.e. every fourth candidate is evaluated. This enables searching a larger window while keeping the cycle count fixed.

For an object with regular motion, MVs will have opposite directions and opposite signs for past and future frames. Thus, averaging MVs can result in loss of direction. So two separate search windows are opened for backward-predicted and forward-predicted frames. For calculation of search window center, MV average of neighboring five MBs as shown in Figure 3-2 are used. Specifically, for forward prediction, average of 10 MVs from two frames is used whereas for backward prediction, average of 15

MVs from three frames is used.

For the search window around the predictor, a full-search is done to have full-pixel resolution and capture the movement of objects accurately.

A summary of the searching strategy is given below:

- Open a search window around the origin and do a search with a sub-sampling ratio of four.
- Open two search windows around forward and backward MV averages and use one of the windows for each frame depending on frame being past or future compared to the reference frame.

Table 3.5 shows experimental results for the searching algorithm on 1920×1080 resolution frames. Experiments are performed on 30 frames for each sequence. In most sequences, bit-rate increase is small as compared to the golden results from a full-search with a ± 128 searching range. However, in sequences such as ParkJoy where motion is irregular this algorithm introduces about 53% bit-rate increase. On average, bit-rate increase for twelve sequences is 17.96%.

Fractional Motion Estimation (FME) Approach

In this work, fractional refinement is done for the best mode selected in IME part and its corresponding MVs. This approach is similar to the CCE motion estimation implementation explained in Chapter 2.

As shown in Table 3.1, VC-1 uses 4-tap filters for half- and quarter-pel positions whereas AVC/H.264 uses 6-tap and 2-tap filters. Since there is a larger number of possible quarter-pel positions, using 4-tap filters impose a higher complexity for VC-1.

3.2.3 Bypassing cost calculation based on similarity

MB parallel motion search provides the opportunity to avoid cost calculation if two current MBs are similar to each other. As an example, one can consider a case where two MBs are exactly the same. Instead of calculating the SAD for both MBs,

Sequence	BDRate Increase w.r.t. Golden Imp.
BlueSky (1920 × 1080)	11.01%
CrowdRun (1920 × 1080)	2.80%
DucksTakeOff (1920 × 1080)	4.32%
IntoTree (1920 × 1080)	7.59%
OldTownCross (1920 × 1080)	4.65%
ParkJoy (1920 × 1080)	53.74%
PedestrianArea (1920 × 1080)	40.59%
RiverBed (1920 × 1080)	8.70%
RushHour (1920 × 1080)	18.18%
Station2 (1920 × 1080)	22.94%
SunFlower (1920 × 1080)	22.69%
Tractor (1920 × 1080)	18.29%
Average	**17.96%**

Table 3.5: Search algorithm experiment results for twelve different 1920 × 1080 resolution sequences. For most sequences, proposed algorithm provides results that is withing the bit-rate increase budget of 25%.

alternatively, SAD can be only calculated for the first MB and then results from the first MB can be used for the second MB as well. Since pixels do match exactly, the outcome will be exactly the same. MBs do not match with their neighbors very often even in high-definition frames but if the two MBs are very similar to each other, the SAD results can still be very close. There is a dependency between how similar MBs are and how close SAD approximation is to the actual value. One way of defining similarity is the ratio of SAD between two MBs and the maximum possible SAD (all "1"s in one MB and all "0"s in the second):

$$\text{Similarity between MBs}(\%) = 100 - \frac{\text{SAD of two MBs}}{16 \times 16 \times 255} \times 100 \tag{3.4}$$

So if two MBs do match exactly, their SAD is zero and their similarity is 100%.

This idea can be used to create a scheme where SAD calculation for one MB can be bypassed based on similarity of the two current MBs (Figure 3-3). If the SAD

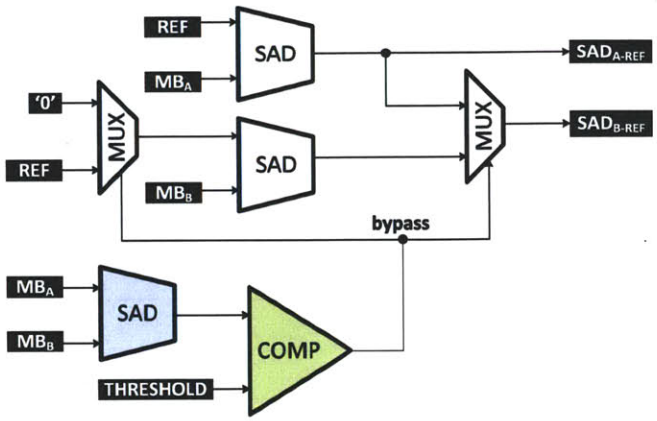


Figure 3-3: Block diagram of the SAD bypassing scheme. If the SAD between MB_A and MB_B is smaller than the threshold, SAD calculations for MB_B is bypassed providing energy savings.

difference between MB_A and MB_B is less than the threshold, *bypass* is" 1" and SAD between MB_A and reference data is calculated alone. Moreover, to eliminate switching activity on the SAD tree for MB_B, 0 is input through the input multiplexer in this case. It should be noted that during a motion search, the difference between MB_A and MB_B is calculated only at the beginning. However, if *bypass* is 1, switching activity in one SAD tree is avoided during the entire motion search.

Figure 3-4 shows the bit-rate increase vs. energy savings trade-off for bypassing SAD calculation for two MB parallel implementation. On the x-axis, threshold is the point below which SAD calculation of MB_B is bypassed. On the y-axis, the ratio of bypassed SADs to total number of SAD calculations and the bit-rate increase due to this approximation are shown. For a threshold value at 2%, 15% of SAD calculations can be saved with a bit-rate increase of around 1%. A higher threshold value can be chosen for lower power at the expense of increased bit-rate.

3.3 Hardware Implementation Results

This section will discuss the hardware implementation of the motion estimation block for multi-standard video encoder design.

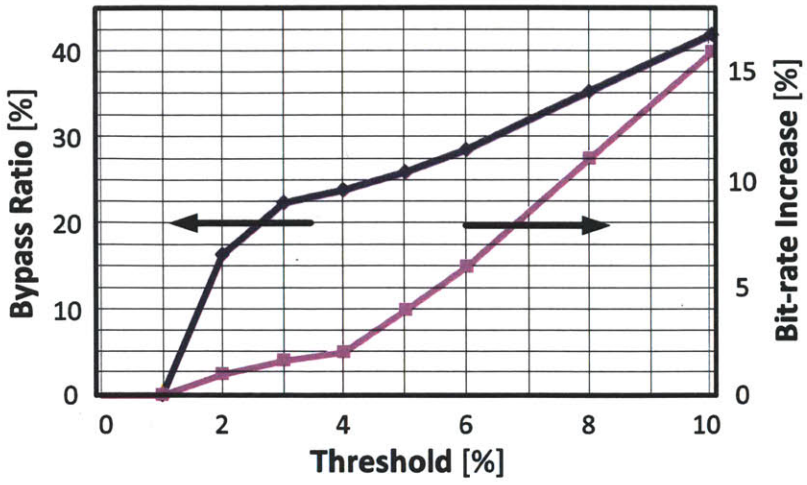


Figure 3-4: Plot showing energy savings vs. bit-rate increase trade-off for different threshold values. Sequence is *oldtowncross* at 1920 × 1080 resolution.

3.3.1 Top Level Block Diagram of Motion Estimation

Figure 3-5 shows the top level block diagram of the multi-standard motion estimation module. All data inputs and outputs are provided through standard FIFO interfaces. Since ten MBs are processed together, ten MBs worth of data is popped from the *MB pixels* FIFO and then stored in *MB Buffer*. Similarly, reference pixels are taken from *Reference Pixels* FIFO and stored in the *Reference Buffer*. *MB Buffer* and *Reference Buffer* are implemented with register-based memories to provide the necessary bandwidth. *Enable MB* input determines if motion estimation will be performed for any of the ten MBs as this decision can be done at the encoder level to lower power consumption if intra-prediction result is sufficiently good.

Top Control block is the main control for inputs from and outputs to the FIFOs as well as the data flow between internal blocks. *Last Line Buffer* holds the MV information of an entire line to use this information in MVP. It should be noted that from the two MBs that are processed in parallel from the same frame, the top one uses the information from this buffer whereas the second MB only has the left and left-top neighbors available for MVP. *Integer Motion Estimation* block performs integer motion search and determines the best inter-prediction mode and best integer MVs. *Fractional Motion Estimation* does not alter the mode decision from IME but only

makes a refinement for MVs. Finally, *Reconstruction* block provides the reconstructed block for each MB depending on the final MVs that are selected since this information can be used in the later stages of the encoder if inter-prediction is used for a MB.

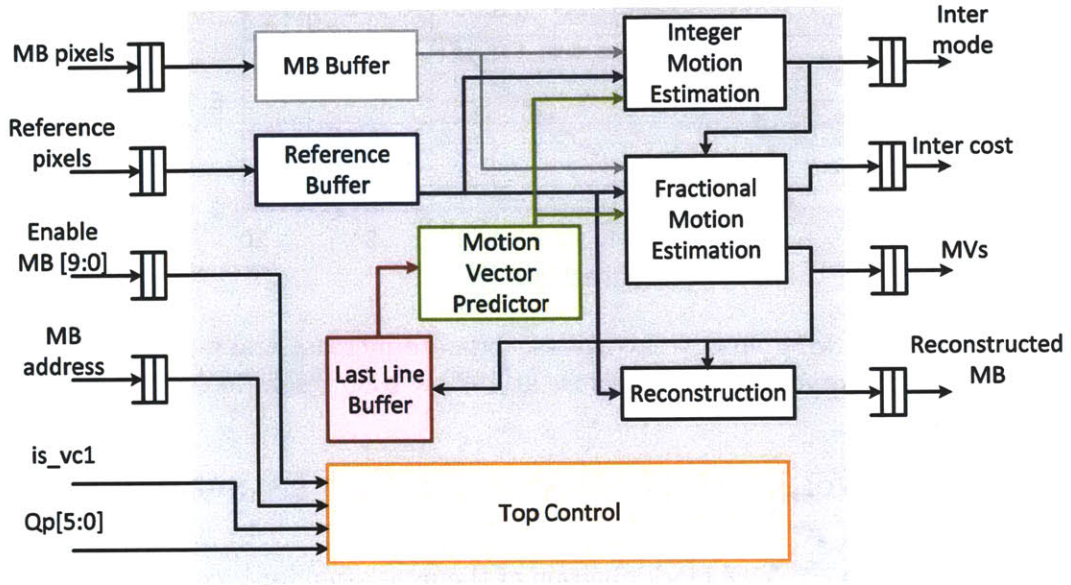


Figure 3-5: Top level block diagram of the multi-standard motion estimation module.

3.3.2 IME Implementation

Figure 3-6 shows hardware block diagram of the IME part. There are ten blocks each performing integer motion search for one MB. Pixels coming from *MB Buffer* and *Reference Buffer* are multiplexed and proper inputs are provided to each block. If some of the *IME_ONE_MB* blocks need to be turned-off, *IME Control* makes sure that data for those engines are not switching to prevent unnecessary switching activity. Outputs of each engine are given to the output to be used in FME.

Figure 3-7 shows the micro architecture of each *IME_ONE_MB* engine. Two candidates are evaluated every cycle. SAD calculations are done for 4×4 blocks since the smallest sub-block size is 4×4 for all modes from 16×16 to 4×4 in AVC/H.264. Then 4×4 costs are combined together to get the costs associated with other modes in *Combine Costs*. For example, for 16×16 mode, outputs of all 4×4 SAD blocks are summed together.

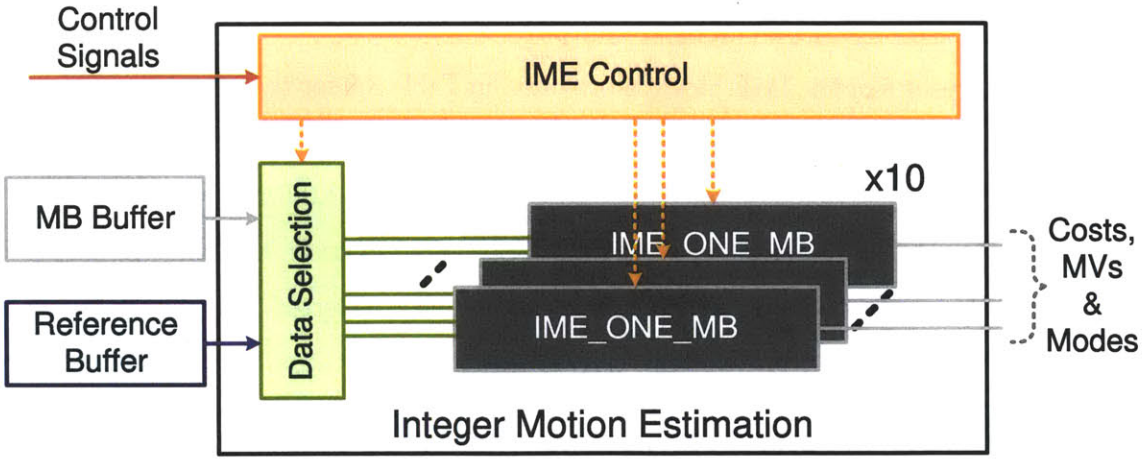


Figure 3-6: Block diagram of IME.

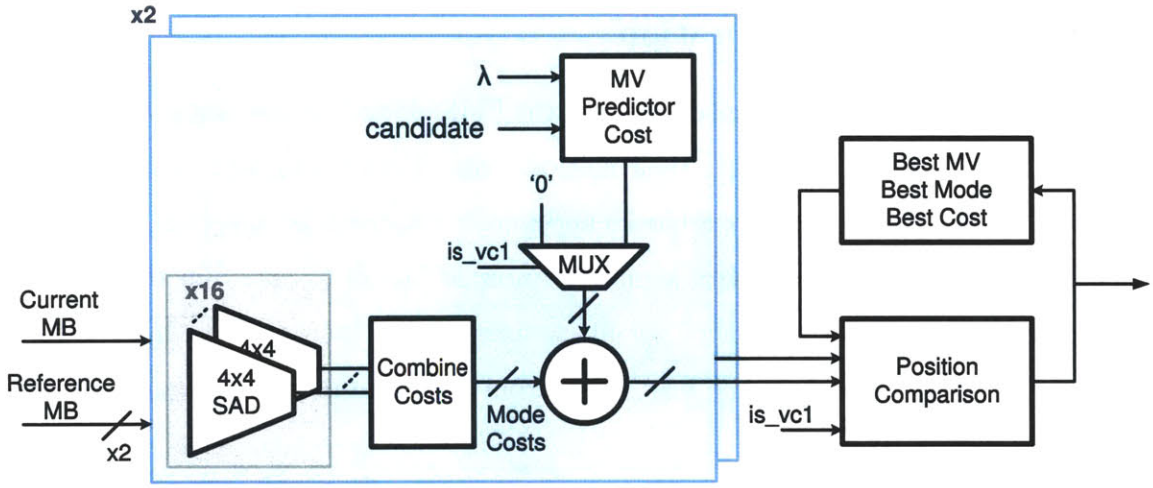


Figure 3-7: Block diagram of IME for one MB.

For AVC/H.264, motion cost is calculated as

$$\text{Total Cost} = \text{SAD} + \lambda \times (\text{MVP} - \text{MV}) \tag{3.5}$$

where MVP is the MVP defined in the standard. So total cost is a function of the MV's distance from the MVP. However, in VC-1, this additional term is not included in the cost calculations. To handle this, MVP cost is calculated and multiplexed with "0" cost.

After *Combine Costs* block, MVP cost is added and this cost is compared with the best cost in *Position Comparison* block. At the end of motion estimation process,

best mode, best cost and best MVs are output.

Synthesis results of the IME block is provided in Table 3.6 for the target frequency of the encoder. Area break-down of different parts of the design shows that SAD trees account for 40% of the total area. Total gate count is 1.08M gates. Hardware reconfigurability allowed > 85% of hardware to be shared between the two standards. The area and timing overhead of the additional hardware inserted into the design to support two standards are very small. IME engine implementation is common across the standards and simple multiplexers that are inserted at various places in the design are not in the critical path and introduce very small area overhead.

3.3.3 FME Implementation

Figure 3-8 shows hardware block diagram of the FME. There are two engines to interpolate 8×8 blocks and calculate their motion costs. Then *Combine Costs/Decision* block combines the costs of 8×8 blocks and makes a decision on the best fractional MV. *FME Control* sends control signals to internal blocks to maintain the flow of data. Since FME is not supported for block sizes smaller than 8×8, *FME Control* turns off *FME_8x8* blocks if the best mode from IME contains blocks smaller than 8×8.

Figure 3-9 shows the micro architecture of a *FME_8x8* block. AVC/H.264 interpolator and VC-1 interpolators are implemented separately and their outputs are multiplexed with *is_vc1* signal. For the unselected interpolator, inputs are driven to

Part	**Area(%)**
SAD Trees	40%
MV Predictor Cost	31%
Position Comparison	7.5%
Combine Costs	2.5%
Other Parts and Sequential Elements	19%
Total	100%

Table 3.6: Area breakdown of IME part.

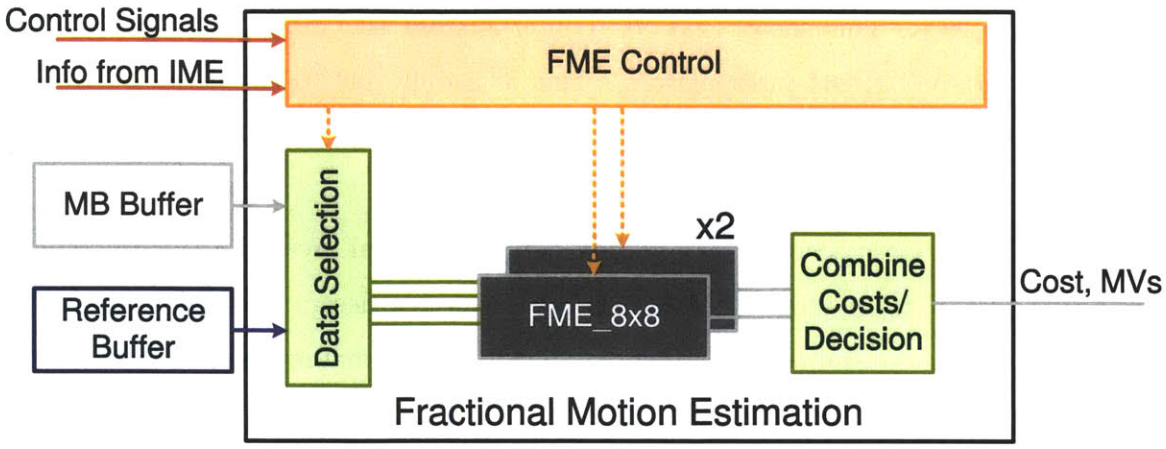


Figure 3-8: Block diagram of fractional motion estimation.

0 such that there is no switching activity inside these parts. In contrast to IME, SAD calculation is done for 8×8 blocks in FME. Similar to IME, MVP cost is added to SAD cost for AVC/H.264 to account for this additional cost. Finally, 8×8 block costs are output to the higher level where these costs are combined and a final decision on fractional MV position is done.

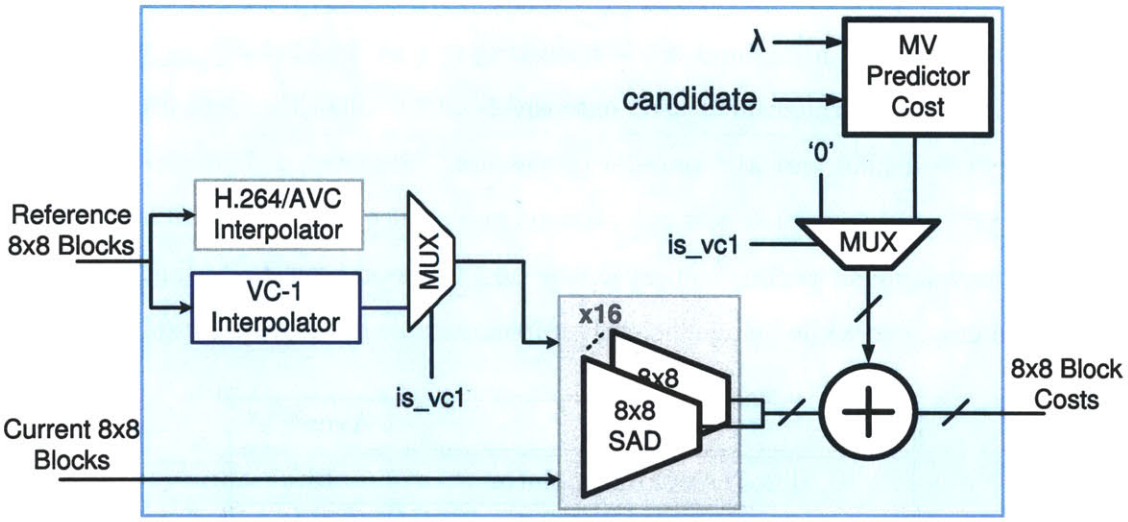


Figure 3-9: Block diagram of fractional motion estimation for an 8×8 block.

Synthesis results of the FME block is provided in Table 3.7 for the target frequency of the encoder. Area break-down of different parts of the design is also provided in this table. VC-1 interpolators account for 47% of the total area whereas AVC/H.264

accounts for 29%. This shows that VC-1 interpolators take 50% larger area when compared to AVC/H.264 interpolators. This is mainly due to the usage of 4-tap filters for quarter-pel positions in contrast to 2-tap filters of AVC/H.264 for quarter-pel.

Lastly, SAD trees account for another 29% of the total area. Total gate count is 1.76M gates. It should be noted that reconfigurable design resulted in 40% of the FME area to be shared across two standards. Additional multiplexers to select between two interpolators account for 6% of the total FME area.

3.4 Summary and Conclusions

Motion estimation is the largest and most power consuming block in video encoders due to large bandwidth requirement and large switching activity. Moreover, driving the inter-prediction portion, motion estimation directly affects coding efficiency of encoders.

This chapter presents the design of a highly-parallel motion estimation engine targeting a very high throughput while operating at a modest 25MHz to enable low voltage operation. Parallelism is used extensively at the algorithm and architecture level to achieve frame and MB parallel processing. Moreover, a "similarity-based" scheme is proposed to bypass cost calculations and reduce switching activity. Search algorithm development is done to provide the best trade-off between coding efficiency and data bandwidth while being efficiently implementable in the frame and MB paral-

Part	Area(%)
H.264/AVC Interpolator	13%
VC-1 Interpolator	47%
SAD Trees	29%
Other Parts and Sequential Elements	11%
Total	100%

Table 3.7: Area breakdown of FME part.

lel schemes. This design has the potential to support up to ultra-high definition level (e.g. $16K \times 8K$ pixels @ 30fps) by boosting the voltage to full-V_{DD} and increasing the frequency to 400MHz.

Encoder design is done as a group consisting of a post-doctoral researcher and three graduate students including myself. I was responsible for the design of the motion estimation engine.

Chapter 4

Low-Power SRAM Design for Multimedia Applications

Low-power SRAM design has been an important research area for many years as the amount of SRAM on a single die has been continuously increasing over the years. SRAM design is generally driven by three metrics: area, performance and power. In most cases, area is the main driver in design decisions. Array efficiency which is defined as

$$Array\ Efficiency = \frac{Area\ of\ Memory\ Cell\ Array}{Total\ Memory\ Area}$$

is expected to be over 70% to be able to achieve high transistor density. Power and performance are also important factors depending on the target application. For example, for a modern micro-processor design, SRAMs are expected to run at the same clock frequency with the processor core (3-4GHz).

In a dynamic voltage scalable system, running all individual blocks from the same supply voltage provides simplicity in the design and avoids additional hardware such as level converters. However, conventional SRAM's minimum operating voltage, V_{min}, is generally higher than logic and consequently, SRAMs limit low voltage operation in a system.

In this chapter, two SRAM designs will be discussed targeting voltage scalability. The design decisions starting from bit-cell topology to peripheral assist circuits are

driven by area considerations as well as target V_{min}. This chapter begins with an example target application for the low-power SRAMs discussed in this chapter. Then, two separate SRAM designs are discussed: a 28nm 6T SRAM design achieving 0.6V operation and a 45nm 8T design achieving 0.5V operation.

4.1 A Low-Power DSP for Multimedia Applications

Processors for next generation mobile devices will target operation on a wide supply voltage range. At high voltage levels, high performance requirements can be satisfied and high energy-efficiency goals can be achieved through scaling the supply voltage down. For this purpose, the work in [42] presented a DSP system-on-chip (SoC) that is designed to operate from 1.0V down to 0.6V in 28nm CMOS technology.

SRAMs for this SoC are designed by myself and will be presented in Section 4.2. SoC design was done in collaboration with two other students (Nathan Ickes and Rahul Rithe) and engineers from Texas Instruments and will be briefly discussed here.

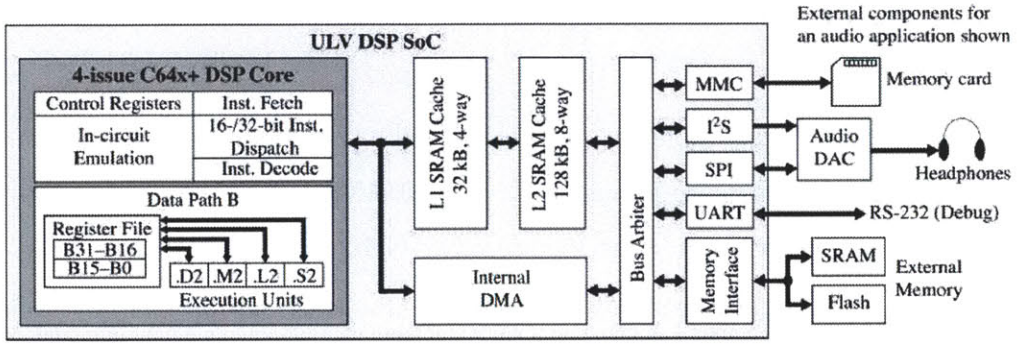


Figure 4-1: Block diagram of the multimedia applications SoC. The design employs 600K gates and 1.6Mbits of SRAM in its L1 and L2 caches.

Figure 4-1 shows the block diagram of the SoC. The DSP core is a 32 bit, 4-issue VLIW processor based on Texas Instruments C64x+ core. SoC employs L1 and L2 caches of sizes 32KB and 128KB respectively, an internal DMA and peripherals to

the outside world. The performance goals for this design were 400MHz at 1.0V where various multimedia applications can be run and 20MHz at 0.6V where barebones MP3 decoding can be performed. The design employs 600K gates and 1.6Mbits of SRAM.

SRAMs being an important part of this design should also be operational on the same voltage range to achieve energy scalability at the full system level. Hence, special low voltage designs are necessary that are targeted for the 0.6V-1.0V range.

4.2 6T Low Voltage SRAM with Peripheral Assist Circuits in 28nm

In this work, a standard high-density $0.12\mu m^2$ 6T bit-cell array is designed to work at low voltages (down to 0.6 V) using area-efficient peripheral assist circuits. An SEM image of the bit-cell is shown in Figure 4-2 [83].

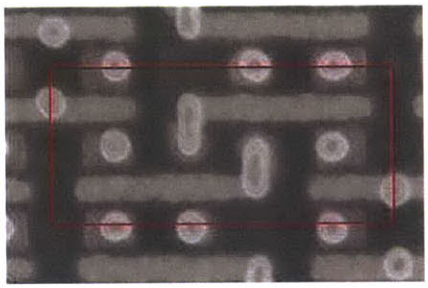


Figure 4-2: SEM image of the 28nm high-density ($0.12\mu m^2$) 6T bit-cell.

The principle mechanisms used to enable low voltage operation of this SRAM are

1. short local BLs, which minimize read disturbances on the bit-cell,
2. word-line (WL) voltage boosting to ensure write-ability at low voltage levels and
3. large-signal sensing of the local BLs, to maximize area efficiency.

It should be noted that using larger transistors can aid with low voltage operation as larger transistors are less susceptible to process variation. Specifically, Pelgrom in

[88] showed that the standard deviation of the threshold voltage is inversely proportional to the channel area. However, increasing transistor sizes would result in an increase in the bit-cell area and consequently degrade area efficiency. The $0.12\mu m^2$ bit-cell used in this work is an industry-standard high-density bit-cell and low voltage operation is achieved through the assist circuits which will be discussed next.

4.2.1 SRAM Array Architecture

Figure 4-3 illustrates the array architecture.

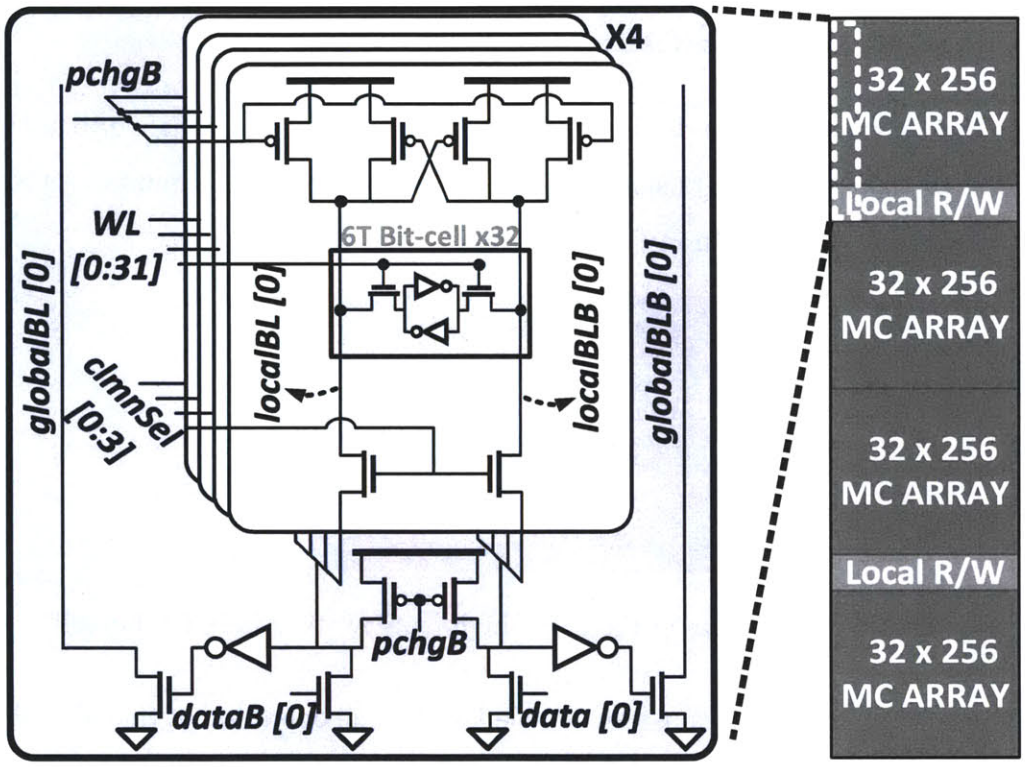


Figure 4-3: Array architecture for the 28nm low-power SRAM design.

Stripes of local read/write (R/W) circuitry are inserted inside the memory cell array. Each local R/W stripe is shared across two sub-arrays of 32 rows and 256 columns. These R/W circuits are coupled to the local bit-lines (BLs) of the sub-arrays. A detailed view of four columns of memory cell array and the implementation of the local R/W circuit is also shown in Figure 4-3. During a read operation, signal development on the local BLs is sensed through a pair of inverters and pull-down

devices connected to the globalBLs. Similarly, for the globalBLs, large signal sensing is used through cross-coupled static NAND gates. During a write operation, the local BLs are driven by one of the two NMOS devices in the R/W circuitry and the cross-coupled PMOS devices connected to the local BLs. The polarity of data/dataB inputs determines which data is written to the selected bit-cells. Lastly, data/dataB pair is driven to "0" during a read operation.

4.2.2 Short Local Bit-lines Reduce Read Disturbance

Traditionally, read margin is characterized through SNM analysis which does not consider any dynamic effects [53]. However, read disturbance of a bit-cell involves dynamic transitions of the BLs and recent work focuses on the dynamic nature of read margin [89, 90, 91]. To capture the effect of BL transitions on the bit-cell stability, read margin must be modeled and analyzed through transient simulations.

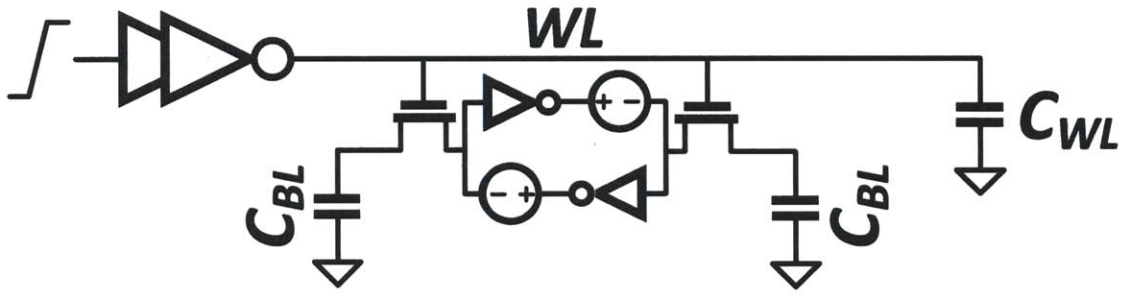


Figure 4-4: Simulation setup for dynamic read margin characterization.

Figure 4-4 shows the dynamic read margin simulation setup. Two DC voltage sources are placed between the cross-coupled inverters. Layout extraction is used to capture the BL and WL capacitance values. Then the values of DC sources are swept in consecutive transient simulations up to the point where a read operation alters the state of the bit-cell. To reduce simulation time, a coarse-to-fine three-step search approach with 100, 10 and 1mV step sizes is used.

Results of the dynamic read margin and static simulations are shown in Figure 4-5. A smaller number of cells on a bit-line results in a faster bit-line transition and exposes the bit-cell to read disturbance for a shorter period of time. This significantly

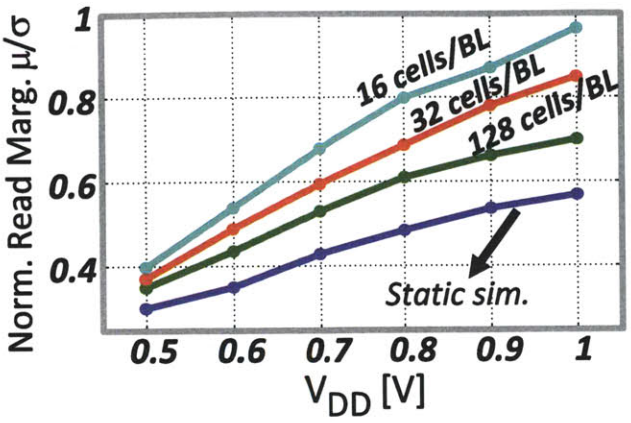


Figure 4-5: Dynamic read margin simulation results on 0.5V-1.0V voltage range.

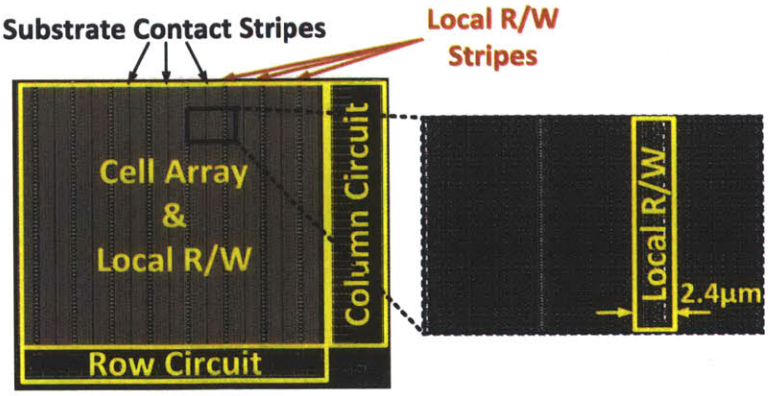
improves the read margin and enables correct operation at a lower supply voltages. In this work, a limit of 32 cells/BL is necessary to ensure operation down to 0.6 V.

Although using a smaller number of cells on BLs improves read margin, the resulting repetition of the local R/W circuit reduces area efficiency. To minimize area overhead, the local R/W circuit is designed by using minimum number of transistors and by sizing each transistor carefully. In layout, R/W circuit's height is limited to only $2.4\mu m$, as shown in Figure 4-6-a. Figure 4-6-b shows the area overhead of the local BL architecture compared to a conventional implementation with 512 cells/BL. Area overhead increases rapidly with smaller number of cells per BL. In this work, selection of 32 cells/BL provides significant improvement of read margin at the expense of 15% area overhead.

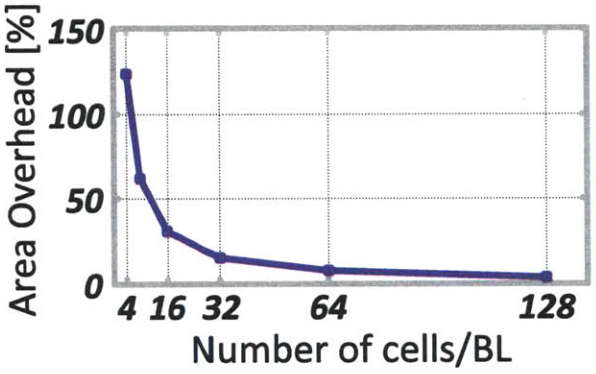
4.2.3 Voltage Boosting Increases Write-Ability

At low voltage levels, transistor variation causes write-ability problems and in this work, WL voltage boosting is used to overcome this issue.

The voltage boosting circuit used to generate the WL voltage is shown in Figure 4-7 [70]. The VDD_{BST} node is used to power up the last level of buffers in WL driver circuits. The operation of the WL voltage boosting circuit is as follows: During a write operation, in the first half of the clock cycle, the boost signal is kept low and the WL is first asserted to VDD_{ARRAY}. Then, after the negative edge of the clock,



(a)



(b)

Figure 4-6: (a) Layout snapshot of an SRAM macro with a zoomed-in view of the local R/W circuitry and (b) area overhead for different cells/BL with respect to a conventional implementation with 512 cells/BL.

voltage boosting takes place. Triggered by the boost signal, charge stored across the capacitor is used to elevate the voltage level on node VDD_{BST} above the array voltage, VDD_{ARRAY}.

The amount of necessary voltage boosting is decided by the write margin simulations. Figure 4-8 plots write margin improvement with 100mV of voltage boost on WL node. An overdrive of 100mV is adequate for correct operation down to 0.6 V. The size of the capacitor is selected carefully with respect to the capacitive loading of the WL driver to ensure correct amount of voltage boosting.

Since only one row of a sub-array is active at any given time, the boosting circuit and capacitor can be shared across all 32 rows of a sub-array. To further reduce the area impact, large capacitors are placed underneath the metal wiring of the address

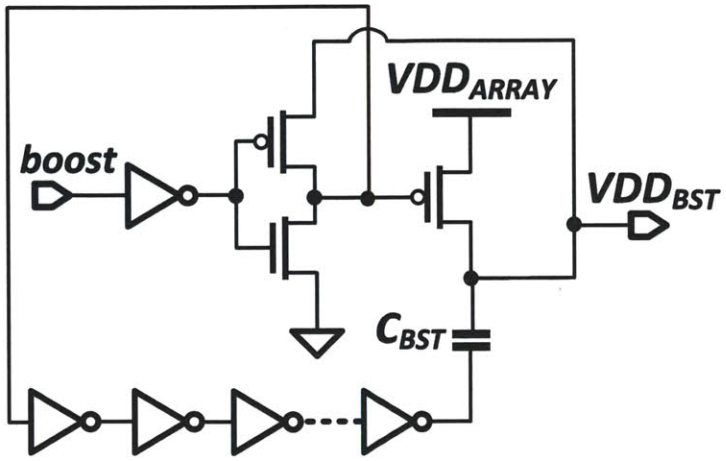


Figure 4-7: WL voltage boosting circuits used in the design.

decoder as shown in Figure 4-9 with a layout sketch. Similar shared boosting circuits are used in the column-select (CS) and data-line signal generation to achieve robust low voltage operation. For data boosting circuit, the load capacitance is smaller and boosting capacitors can be correspondingly smaller. The area overhead due to voltage boosting circuits in this design is less than 4%.

Timing of the boost signal is critical, as turning on the boost circuit too early will result in loss of charge stored on the capacitor and turning on the boost circuit too late can degrade the performance of the SRAM. For this design, we used the negative edge of the clock to trigger voltage boosting using inverter delay lines in the timing circuit. The pulse width of boost signal is chosen through transient Monte-Carlo

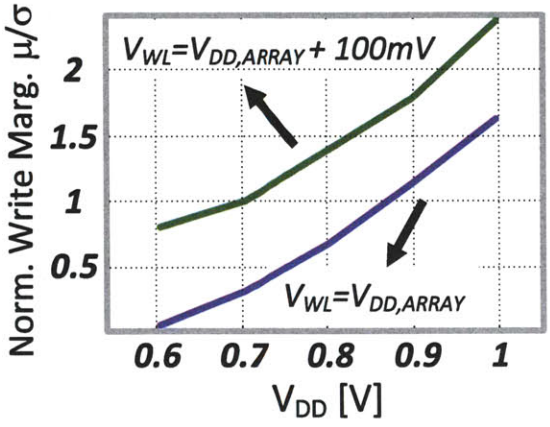


Figure 4-8: Write margin improvement with 100mV of WL overdrive.

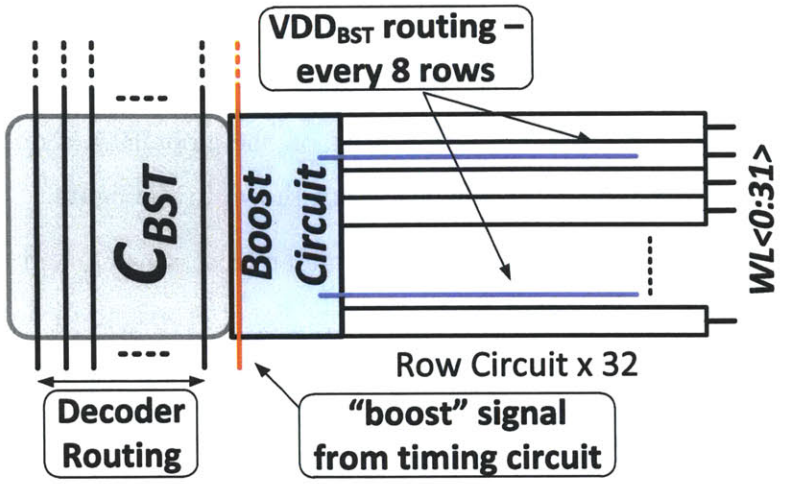


Figure 4-9: Sketch of the WL voltage boosting circuit's layout placement with respect to other blocks.

simulations on the bit cell to allow enough time for write operation to be performed. For the voltage boosting circuit, the boost signal is sent from the timing circuit to trigger WL boosting operation. Similar boost signals are used for CS and data boost circuits. Charging/discharging time of the boosted signals can be overlapped with BL pre-charging time and does not require additional margin for functionality.

Finally, in layout, the power lines associated with VDD_{BST} need to be routed such that the metal parasitic capacitances on them are minimized, as these parasitics will add to the load capacitance and affect the amount of voltage boosting. However, the lines also need to be dense enough to prevent IR drop when the current drawn from this node spikes. We chose to route a power strap every 8 rows as a balance as shown in Figure 4-9.

4.2.4 Improving Read Access Time

The differential read path from the bit-cell to globalBLs is shown in Figure 4-10.

To improve read access time, first, NMOS switches are selected for column-multiplexing since a large-signal development is necessary on the local BLs. Secondly, sensing inverters are designed to favor a low-to-high transition to speed up the signal propagation from local BLs to globalBLs. This is done by designing the PMOS

devices larger than the NMOS devices of the sensing inverters. Finally, differential globalBLs are used to read data from the sub-arrays to maintain signal integrity even at low voltage levels. The last level of sensing on the globalBLs is done through a pair of cross-coupled NAND gates which are not shown in the figure.

Although the cell variation affects both hierarchical sensing and conventional small-signal schemes, in the small-signal scheme, signal propagation on the long BL (512 cells/BL) would dominate the delay, and read path delay is severely sensitive to cell variation especially at low voltages. However, in the hierarchical sensing approach, local BL driven by the bit-cell is much shorter (32 cells/BL) and the propagation continues through the local sense and NMOS pull-down device in the peripheral circuitry. These devices can be designed to be slightly larger with minimal area overhead for a target performance improvement and this reduces their susceptibility to variation. Thus, hierarchical sensing approach provides better worst-case delay at low voltage levels.

Simulated read access time distributions are shown in Figure 4-11. As a reference design, a conventional implementation is also considered which employs small-signal

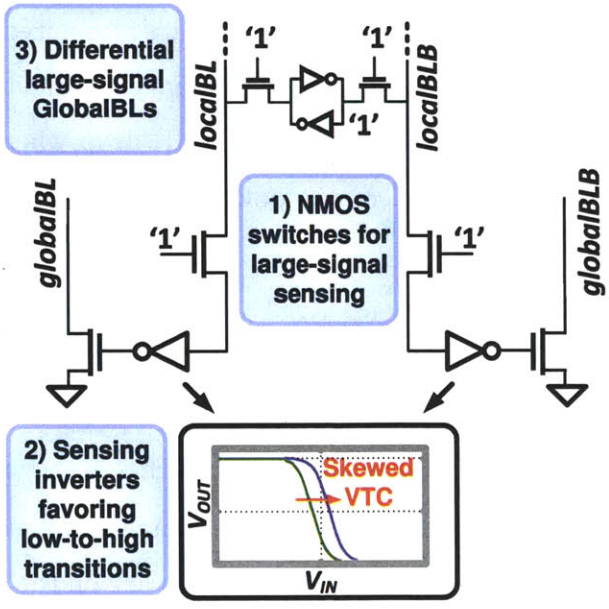


Figure 4-10: Differential read path from bit-cell to globalBLs and read access time improvements done in this design.

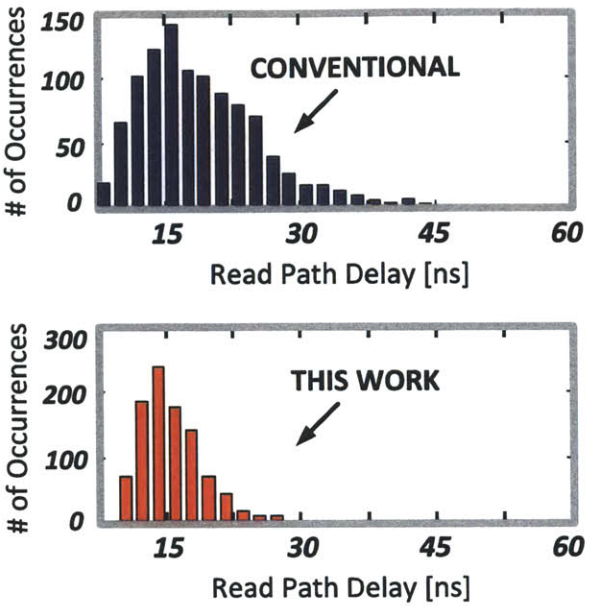


Figure 4-11: Read access time distributions for a conventional design with 512 cells/BL and 50mV sense-amplifier input offset and for this work.

sensing with 512 cells/BL and a sense-amplifier input offset of 50mV. At 0.6 V, distributions show that the improvements explained in this subsection result in 1.8× reduction in worst-case read access time.

4.2.5 Waveforms of Critical Signals in SRAM array

Figure 4-12 shows the waveforms of critical signals during write and read operations. Inverter delay lines are used to create different edges from the clock.

A write operation starts with the rising edge of the clock. WL is asserted after a short delay. During the first half of the clock cycle, WL is kept asserted to V_{DD} and half-selected bit-cells undergo a read operation. Voltage boosting takes place after the negative edge of the clock. clmnSel and data signals are boosted first to provide enough time for local BLs to settle to their appropriate values before WL boosting takes place.

Read operation starts with the rising edge of the clock as well and similarly, WL is pulled high after a short delay. The clmnSel assertion is delayed with respect to WL not to expose the bit-cell to the additional capacitive loading of local R/W

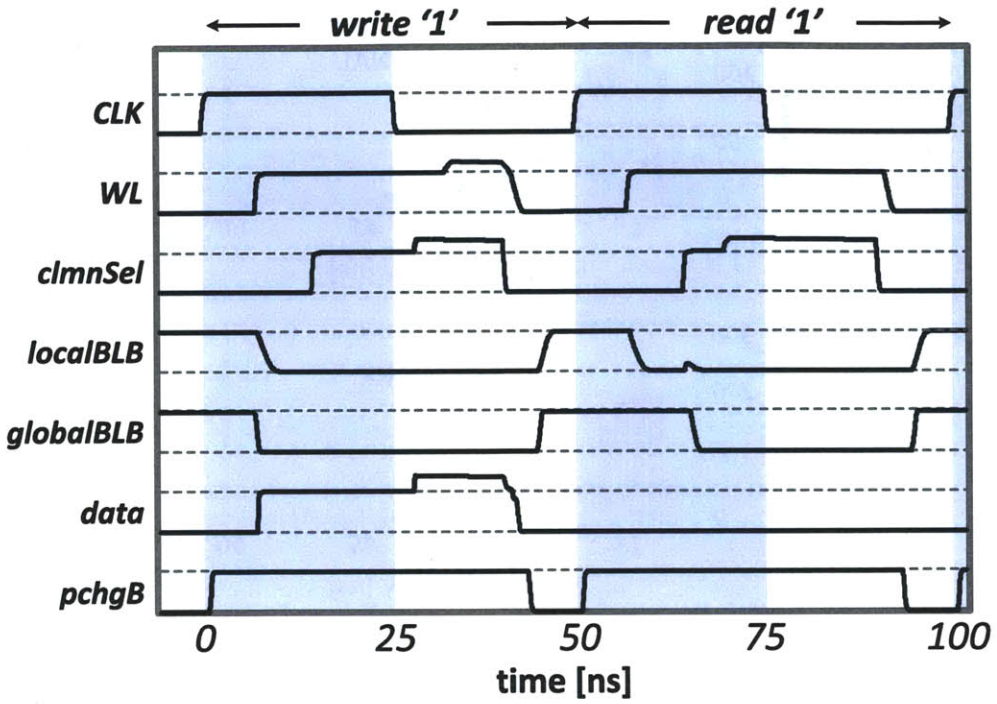


Figure 4-12: Waveforms of critical signals during a write and read operations.

circuit. The clmnSel signal is boosted during a read operation as well to improve the performance of slower read paths.

4.2.6 28nm Test Chip Architecture

Figure 4-13 shows the test chip architecture. Four SRAM blocks with different sizes are placed on the test chip.

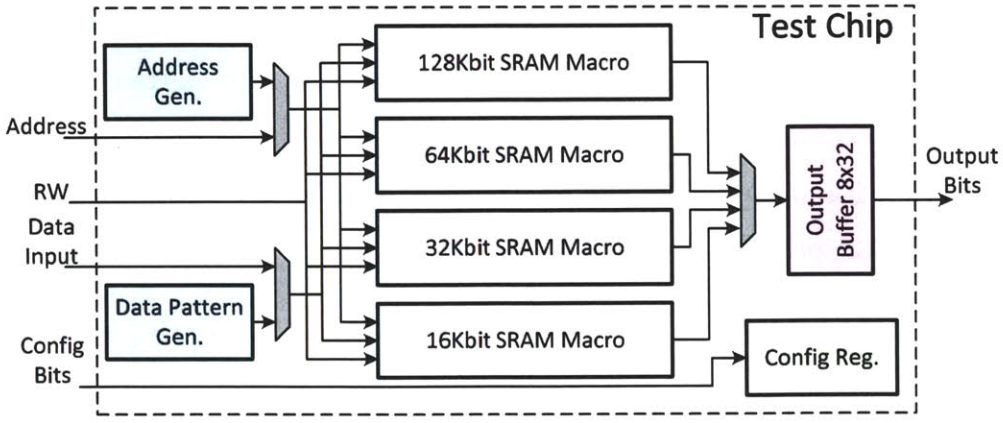


Figure 4-13: Top level architecture of the 28nm SRAM test chip.

Address and data can be generated from on-chip circuits or can be input from outside the chip. On-chip address and data generation make the testing and characterization easier. Address generator consists of an up-counter and some additional logic to go through different addresses. Data pattern generator consists of shift registers combined with multiplexers to create different patters. An example pattern is checker-board pattern consisting of alternating zeros and ones. By writing different combinations of these patterns, worst-case as well as best-case conditions on SRAM leakage and active power and BL leakage can be easily generated.

Because of the pad limitations, output bits are multiplexed to the output. A 32-bit wide 8 words deep output array is used to hold the data read from SRAMs. Multiplexers embedded into this output array enable outputting different parts of the SRAM rows to the output.

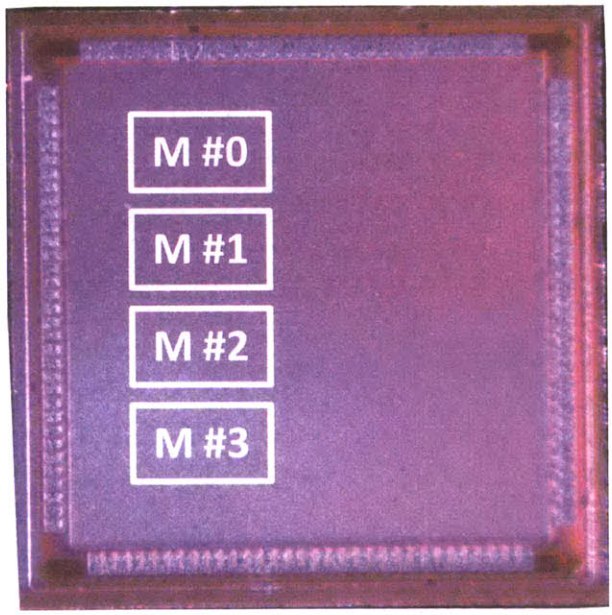


Figure 4-14: Die photograph of the 28nm SRAM chip.

Lastly, a configuration register is implemented as a shift register to set various configuration bits in the design.

4.2.7 28nm Test Chip Measurement Results

Ideas presented in this section are implemented in a 28nm low-power CMOS process. A separate chip is fabricated for the low-power DSP which employing SRAMs from this work as well. A die photograph of the test chip is shown in Figure 4-14 and Table 4.1 summarizes the features of this test chip. Size of the die is $2.3mm \times 2.3mm$ and total pin count is 132. Four SRAM macros are placed on a die with total size of 240Kbits on a die.

SRAMs designed in this work are used in a separate low-power DSP chip implementation as explained in Section 4.1 [42]. Die photograph of the 28nm test chip is shown in Figure 4-15 and 1.6Mbits of SRAM blocks are highlighted on the figure.

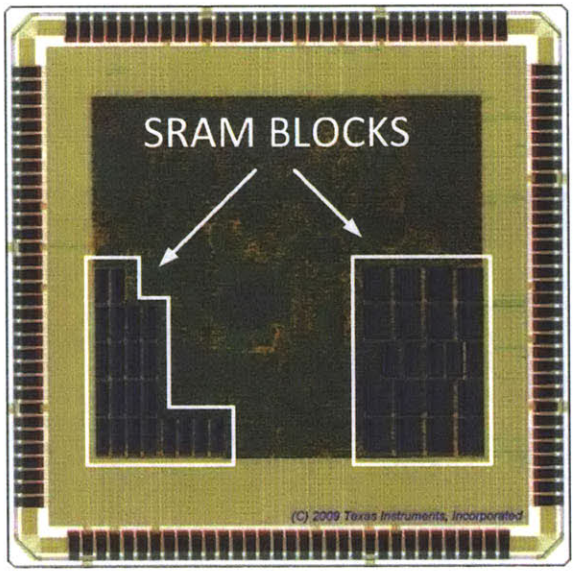


Figure 4-15: Die photograph of the 28nm low-power DSP chip featuring 1.6Mbits of the 28nm SRAMs described in this section.

For testing purposes of the 28nm SRAM test chip, fabricated die are packaged into a 144-pin QFP ceramic package and a 4-layer test PCB board is designed.

Figure 4-16 shows measured performance vs. V_{DD} shmoo plot. SRAMs operate from 1.0V down to 0.6V. On this voltage range, SRAM performance scales from 400MHz down to 20MHz. This meets the initial target of the design. Although some bit errors occur above 0.6V, they are below the repairable limit which is two rows and two columns of redundancy. It should be noted that redundancy is not implemented

Technology	28nm Low-Power CMOS%
Die Size	2.3mm × 2.3mm
Number of Pads	132
Total SRAM on die	240Kbits
SRAM Macro Organization	1K words × 128bits/words
Column-Interleaving Ratio	4-to-1
Bit-cell Size	6T HD ($0.12\mu m^2$)
Voltage Range	0.6V - 1.0V

Table 4.1: Summary of the 28nm test chip.

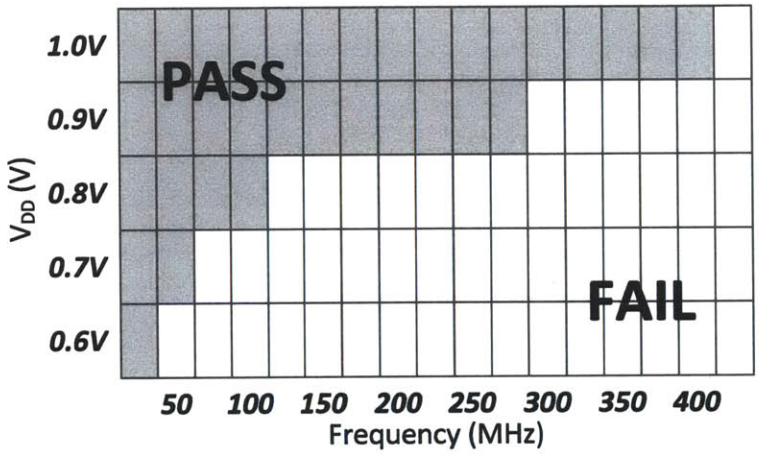


Figure 4-16: Measured performance vs. V_{DD} shmoo plot.

in this work but the design is compatible with redundancy implementation as well.

Figure 4-17 shows the scope snapshot of one of the SRAMs when operating at 0.6V. Clock input and one of the output bits is shown on the figure.

4.3 8T Column Interleaved SRAM with Sense-Amplifier Reference Selection in 45nm

This section presents the second SRAM design for this chapter. Employing an 8T bit-cell and fabricated in 45nm technology, this design proposes a new array architecture to allow column-interleaving for the 8T bit-cell and also utilizes an on-chip reference

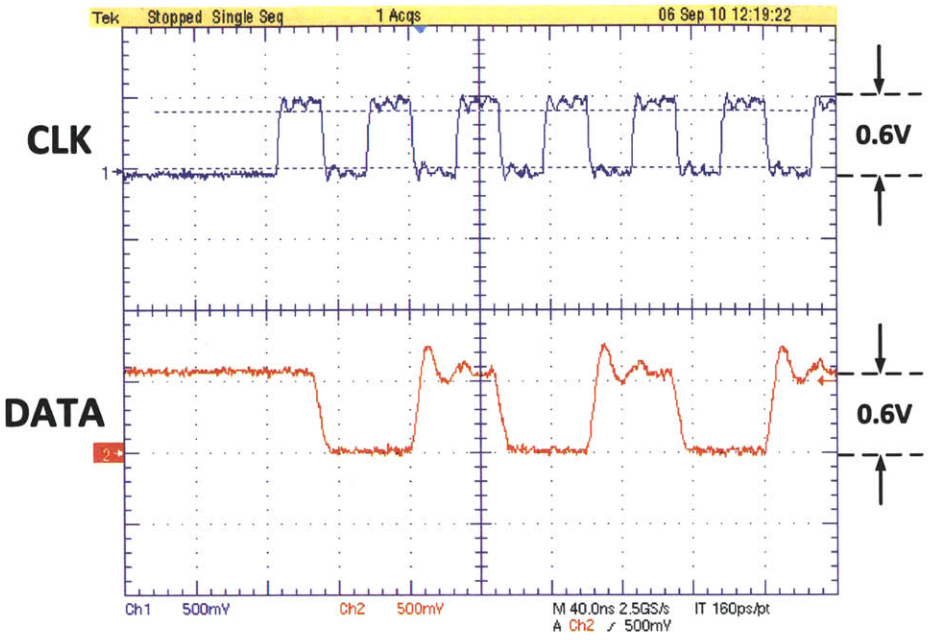


Figure 4-17: Scope snapshot of clock input and one of the data output bits when operating at 0.6V.

selection loop to perform sense-amplifier offset reduction.

4.3.1 Column-Interleaving with 8T Designs

As mentioned in Section 1.4, 8T bit-cells hold great promise for overcoming device variability in deeply scaled SRAMs and enabling aggressive voltage scaling for ultra-low-power. However, if used with column-interleaving, 8T bit-cell suffers from "half-select" problem. During a write operation, un-accessed bits on the accessed row experience a condition that is equivalent to a read disturbance on a 6T cell. Hence, 8T designs (e.g. [62],[70]) often prefer "one-word-per-row" architectures to avoid the half-select conditions. The work in [63] addresses this problem by applying a "read-then-write-back" scheme whereas the work in [66] proposes a 10T bit-cell that can be column-interleaved by vertical and horizontal word-lines (WL). Column-interleaved architectures (Figure 4-18) are almost always preferred due to

1. increased soft-error immunity and

2. better sense-amplifier area utilization.

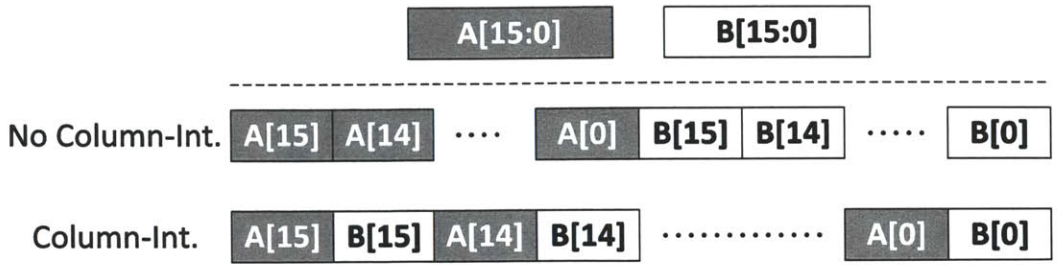


Figure 4-18: Physical allocation of two words in column-interleaved and non-column-interleaved architecture.

First, in a column-interleaved architecture, bits of a word are spatially separated in the row and simple single-bit error correction coding (ECC) can be used to address soft-errors. Non-interleaved architectures, however, require more complex multi-bit ECC schemes which can be costly in terms of area and delay. Second, only one of many columns are active in a column-interleaved architecture which enables sense-amplifier sharing across multiple columns. In contrast, non-interleaved SRAMs require a sense-amplifier for each column to read all bits of a word at the same time.

Maximum attainable SRAM performance is limited by the weakest bit-cell and its ability to create a voltage differential on the bit-lines (BL) that is larger than the input offset of the worst sense-amplifier. In deep-sub-micron technologies, transistor mismatches are becoming more prominent. Hence, to keep its input offset at an acceptable level, sense-amplifier area cannot scale down as rapidly as the bit-cell area. This introduces a significant area problem with regards to the sense-amplifier and the problem is exacerbated in 8T designs without column-interleaving where sense-amplifier area cannot be shared across multiple columns.

Figure 4-19 demonstrates the trade-off between array efficiency (AE) and sense-amplifier offset with and without column-interleaving for an array size of 64 rows by 64 columns (64 rows of 64-bit words).

It can be seen from Figure 4-19 that a higher array-efficiency demands a smaller sense-amplifier area and consequently results into a larger input offset voltage. Increasing column-interleaving (clmn-int) ratio, however, provides better array efficiency for the same offset voltage by amortizing sense-amplifier area over multiple columns.

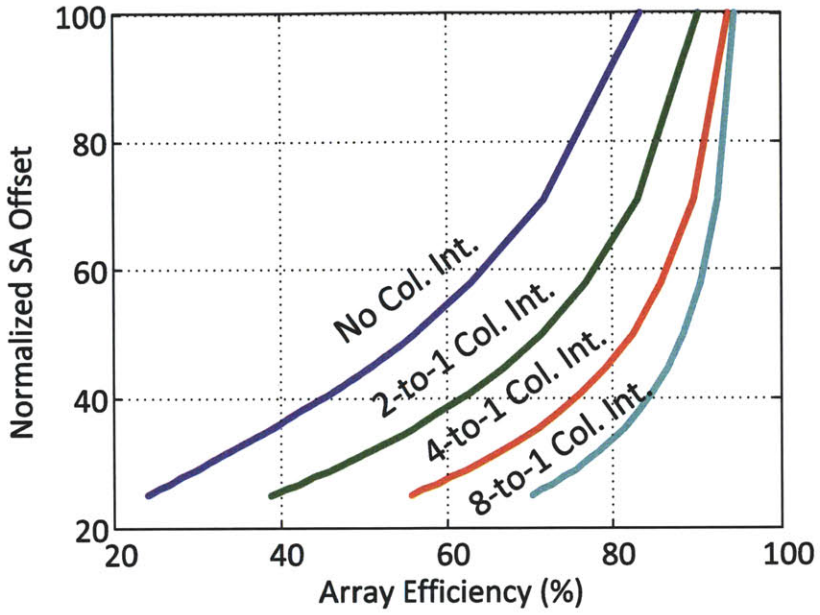


Figure 4-19: Sense-amplifier offset vs. array efficiency with different column-interleaving ratios.

In this section, we will present an array architecture and circuits with minimal area overhead to allow column-interleaving while eliminating the half-select problem for 8T bit-cell. This enables sense-amplifier sharing and soft-error immunity. To further reduce sense-amplifier offset, a reference selection loop is also designed and implemented in the column circuitry and will be discussed in this section. By choosing one of the two reference voltages for each sense-amplifier in a pseudo-differential scheme, selection loop effectively reduces input offset.

4.3.2 Column-Interleaved Array Architecture for 8T Bit-cell

Conventional bit-cell layout has an aspect ratio close to three i.e. its width is three times longer than its height. Consequently, efficient placement of additional transistors into bit-cell pitch in the horizontal direction can be more area efficient when compared to the vertical placement.

Figure 4-20 shows a simple schematic illustration of a unit portion of the proposed array architecture. Four new transistors are inserted between four bit-cells. BL-ports of adjacent cells are shared horizontally (intBL/intBLB) and column-lines

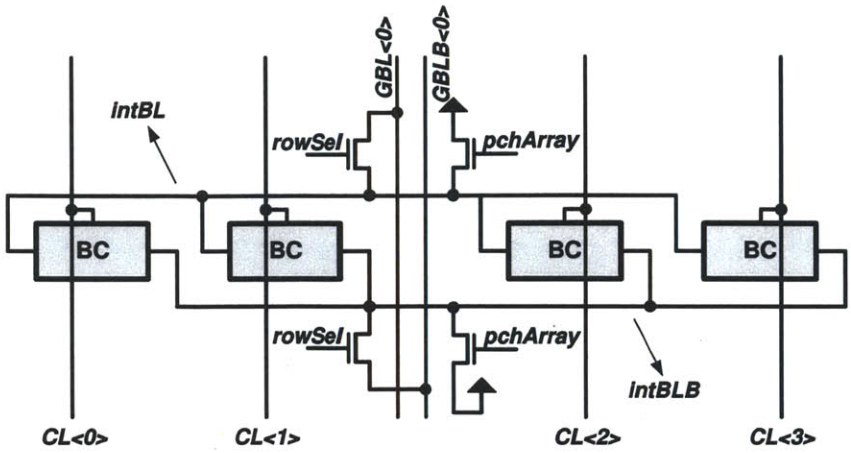


Figure 4-20: Schematic illustration of the proposed architecture suitable for column-interleaving. *BL/BLB*-ports of four bit-cells are shared in horizontal direction and Column-Line (CL)s are routed in vertical direction. rowSel selects the active row.

(CL) are routed vertically and connected to the access devices of the 8T bit-cells. Only the CLs of the active columns are asserted during a write access. The rowSel signal selects active row and drives internal-bit-lines (intBL/intBLB) to global-bit-lines (GBL/GBLB) through NMOS pass-gates. Although these pass-gates cause a degradation on logic levels, bit-cells are sized to ensure write-ability under this condition.

For un-selected rows (i.e. rowSel="0"), intBL/intBLBs are isolated from GBL/GBLBs. Therefore, an un-selected bit-cell needs to drive a small intBL/intBLB capacitance. As explained in Section 4.2, small BL capacitance significantly improves bit-cell stability and hence resulting half-select-disturbance on this cell is small. At the end of every write-cycle, The pchArray signal is asserted for a short period of time to pre-charge intBL/intBLB to a known state. This is necessary because the previous state of intBL/intBLB can cause an elevated level of disturbance on the half-selected bit-cells. Finally, a read-access is done through the read-buffer of the 8T cell in the conventional way.

Layout realization of the schematic illustration in Figure 4-20 is shown in Figure 4-21.

The following steps are taken for efficient layout implementation.

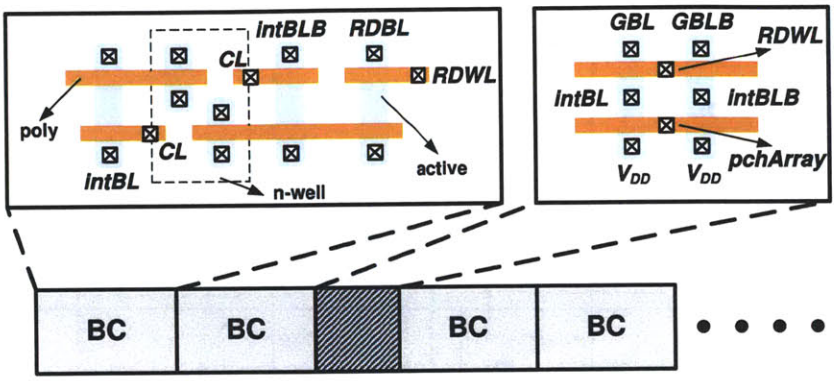


Figure 4-21: In layout, additional NMOS transistors fit in the bit-cell pitch providing area efficient implementation. RDWL is used instead of rowSel which allows shorting of poly-layer between bit-cells and row-select NMOSs.

1. Devices connected to pchArray are chosen to be NMOSs to be able to stack them with row-select devices.

2. RDWL for each row is used for rowSel to prevent extra metal routing. During a write-access, RDWL is also asserted to select the active row.

3. intBL and intBLB are shared between adjacent rows from above or from below. Hence, effectively eight bit-cells share the same local-bit-line. To prevent two bit-cells driving same internal-bit-line, two separate CL signals are routed for each column and they are connected to bit-cells on alternating rows ($CL<0>$ and $CL<1>$) in Figure 4-22-a.

Area overhead of this layout implementation compared to a conventional 8T array with non-interleaved architecture is 12%. This is due to i) four additional NMOS devices and ii) non-overlapping CL contacts between adjacent columns of bit-cells. In layout, minimum sizes allowed by core design rules are used.

Complete schematic of three rows and four columns of the proposed architecture and waveforms of critical signals are shown in Figure 4-22.

Since RDBLs (not shown in Figure 4-22 for simplicity) are used for selecting the active column during write accesses, RDBLs are kept low and only pre-charged to V_{DD} at the beginning of a read cycle to prevent unnecessary RDBL discharge during write accesses. pchArray is pulled to high at the end of a write cycle and kept high

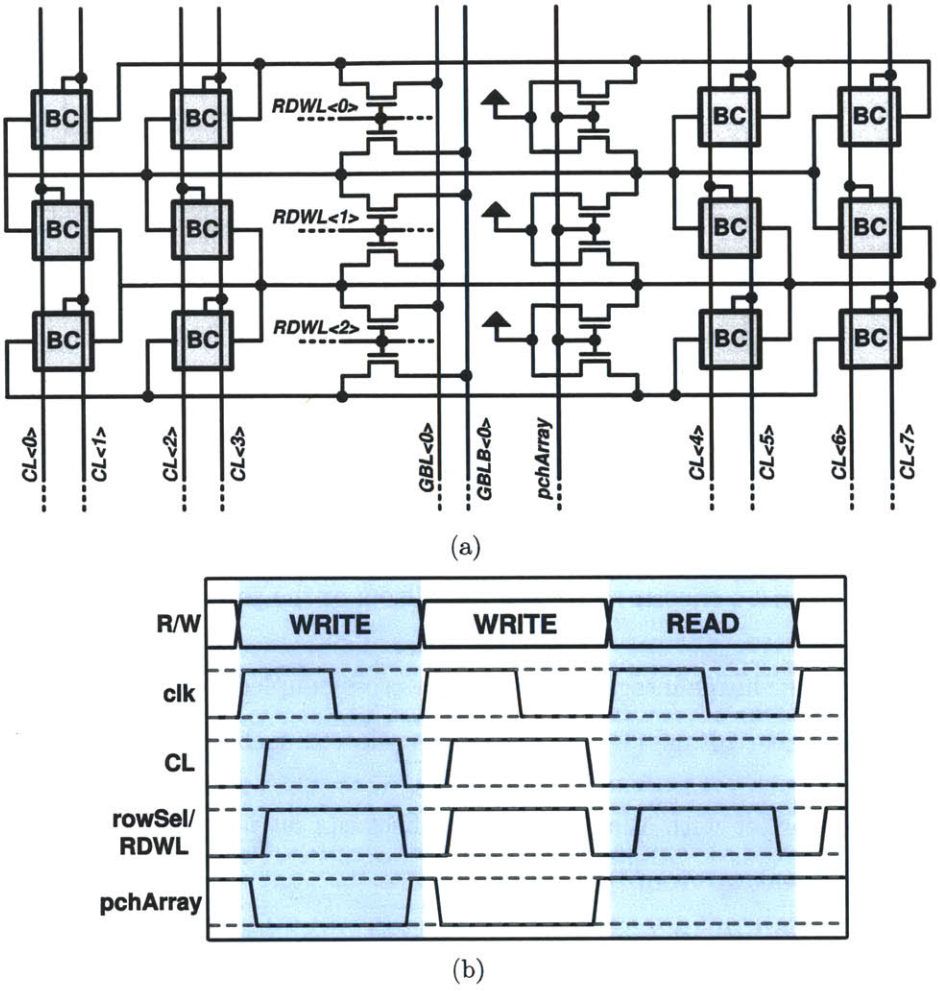


Figure 4-22: (a) Schematic of three rows and four columns of bit-cells in proposed architecture and (b) waveforms for important signals during read and write accesses. RDWL is used for row-select during write operation and pchArray signal is asserted at the end of each write cycle.

during a read access.

4.3.3 Reference Selection Loop for Sense-Amplifier

Sense-amplifier input offset is a critical metric directly impacting the performance of an SRAM. Because of the single-ended RDBL, 8T designs often use pseudo-differential sense-amplifiers where one of the inputs is connected to RDBL while the other one is connected to a reference voltage.

Figure 4-23 shows input offset of a widely-used sense-amplifier [77]. Four PMOS

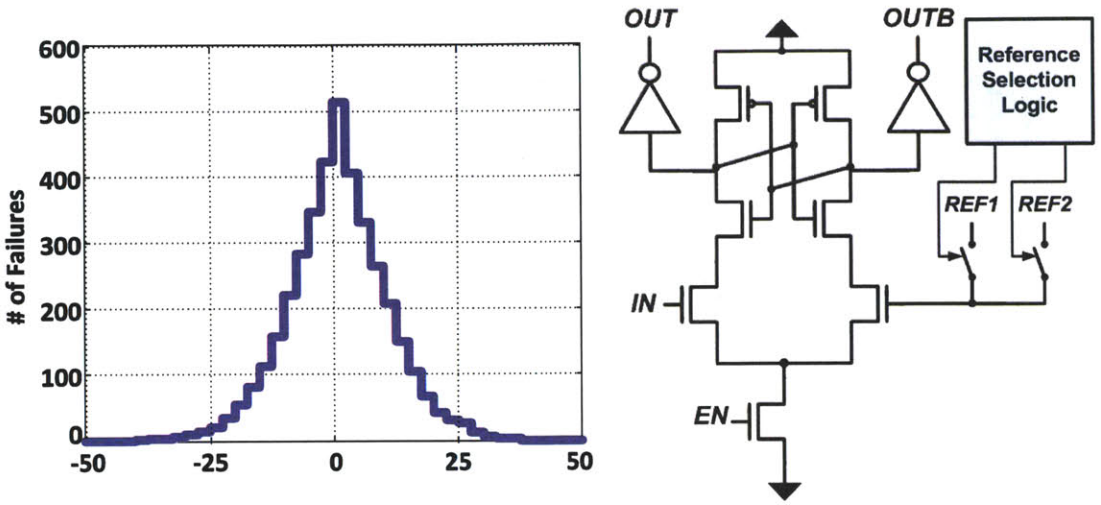


Figure 4-23: Sense-amplifier offset distribution and two-reference voltage scheme. Reference levels can be chosen to reduce the offsets of the sense-amplifiers.

transistors that pre-charge internal nodes of the cross-coupled inverters are not shown for clarity. Reference voltage (V_{REF}) should be selected such that:

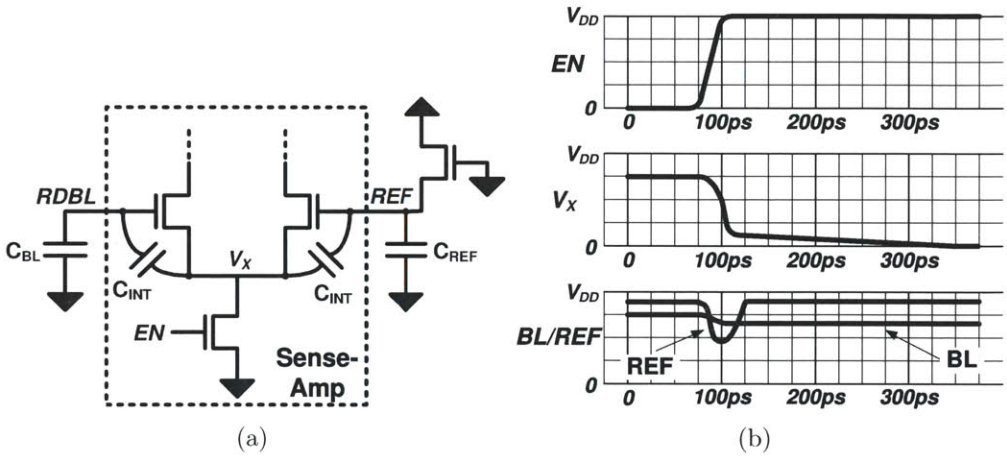
1. A sense-amplifier with a large negative offset can output '1' when RDBL=V_{DD} (neglecting leakage from RDBL) and

2. A sense-amplifier with a large positive offset can output '0' if RDBL is sufficiently low after discharged by the bit-cell.

For a worst-case input offset of $\pm 50mV$ as shown in Figure 4-23, V_{REF} can be placed $50mV$ below V_{DD} . As a result, to sense a logic low, RDBLs should be discharged by at least V_{DISCH}=$100mV$. Alternatively, if two reference voltages are available (V_{REF1}, V_{REF2}), these voltage levels can be selected to compensate for the offset of each sense-amplifier. Specifically, a sense-amplifier with a negative offset can be assigned a higher reference and a sense-amplifier with a positive offset can be assigned a lower reference voltage. This lowers V_{DISCH} and hence improves SRAM performance significantly at low voltages where the ratio of worst-case cell current to nominal cell current is very large due to variation. The work in [92] proposes a similar approach with 16 reference voltage levels for a specific SRAM architecture where a hierarchical sensing scheme is used and total number of sense-amplifiers are

much lower.

In this work, column-circuit is designed with a simple reference selection logic and two reference voltages. To minimize area overhead, only a latch and a few logic gates are used. At the start-up, selection loop is triggered by a series of off-chip signals and each sense-amplifier is tested with RDBL=V_{DD} and V_{REF}=V_{REFH}. If the output of the sense-amplifier is correct ('1'), then V_{REFH} is selected. Otherwise, V_{REFL} is assigned to the sense-amplifier. The result of the selection loop is stored in a latch in column circuit.



(a) (b)

Figure 4-24: Effect of coupling to RDBL and REF nodes with different capacitive divider ratios. Different level of coupling to input transistors alters sense-amplifier inputs and negate the effect of offset compensation.

Robust operation of the sense-amplifier and the offset reduction through reference voltage selection relies on stable, low-noise reference levels. In a single-reference scheme, large decoupling capacitors for REF can be placed and their area can be amortized over the entire array since every sense-amplifier is using the same reference voltage. However, for our scheme, each sense-amplifier is connected to its assigned reference voltage through a PMOS switch. Figure 4-24 plots the effect of signal coupling on sense-amplifier inputs. After the assertion of EN, node V_x is rapidly pulled down which is coupled to both inputs through capacitor dividers. If divider ratios are significantly different, RDBL and REF voltages are altered at the beginning of sensing. Although REF is actively driven by a PMOS, the response of the PMOS

to this coupling might not be adequate especially at the early stage of sensing. To address this problem, in this work, GBLB for every column is connected to REF during read-accesses. GBLB is designed to have a capacitance very close to C_{BL} so the amount of coupling to both inputs is almost the same.

4.3.4 45nm Test Chip Architecture

Test chip consists of four macros each having 256 rows and 128 columns. A column-interleaving ratio of 4-to-1 is implemented for the standard 8T cell. Address and data generation circuits similar to the ones explained in Section 4.2 are placed on the chip to ease testing and characterization.

4.3.5 45nm Test Chip Measurement Results

A column-interleaved architecture suitable for 8T bit-cell and reference selection loop for sense-amplifier ideas are implemented in a 128Kbit 45nm SRAM test chip. Die photo is shown in Figure 4-25 and Table 4.2 provides a summary of the test chip. Die size is $0.9mm \times 1.5mm$ and total pin count is 46. Die are packaged into a 64-pin QFP ceramic package for testing.

Technology	45nm Low-Power CMOS
Die Size	0.9mm × 1.5mm
Number of Pads	46
Total SRAM on die	128Kbits
SRAM Macro Organization	256 words × 128bits/words × 4 blocks
Column-Interleaving Ratio	4-to-1
Voltage Range	0.5V - 1.1V

Table 4.2: Summary of the 45nm test chip.

Test chips achieve functionality down to 0.6V with no bit-errors and down to 0.5V with 2×10^{-4} bit-error-rate.

Figure 4-26 shows measured performance of the test array. When operating at

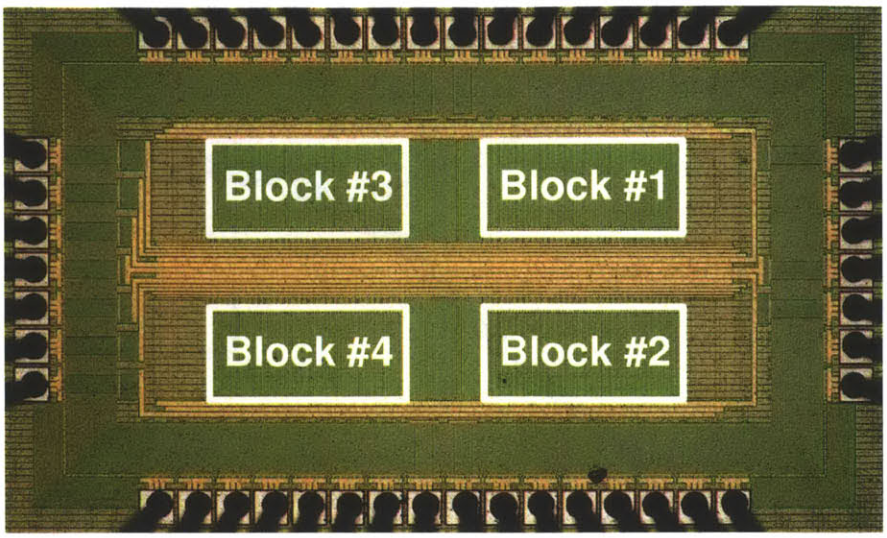


Figure 4-25: Die photo of the 128Kbit SRAM test chip fabricated in 45nm CMOS process.

1.1V, this design achieves 450MHz read and write functionality. Reference selection loop improves performance by around 10%. Leakage power scales down from 179μW to 10.7μW on 1.1-0.5V range. Below 0.5V, write-ability problems begin to emerge and increase bit-error-rate to 10^{-3} at 0.4V.

Figure 4-27 plots measured energy vs. V_{DD} for the 45nm SRAM test chip. Active and leakage components of the total energy are plotted on the figure. Leakage energy increases rapidly because of the rapid performance drop as V_{DD} approaches sub-V_T region. Active energy scales down quadratically with V_{DD} as expected. Total energy makes a minimum around 0.5V. As mentioned above, bit failures occur at 0.4V but the error rate is very low to have an effect on the measured energy.

4.4 Summary and Conclusions

Due to the increasing amount of SRAMs in SoC, SRAM energy is becoming a determining factor in the overall energy-efficiency of a system. Voltage scaling is an effective way of reducing energy/operation for digital circuits and applying voltage scalability to SRAMs is a critical problem because conventional SRAMs suffer from functional failures as the supply voltage is scaled down.

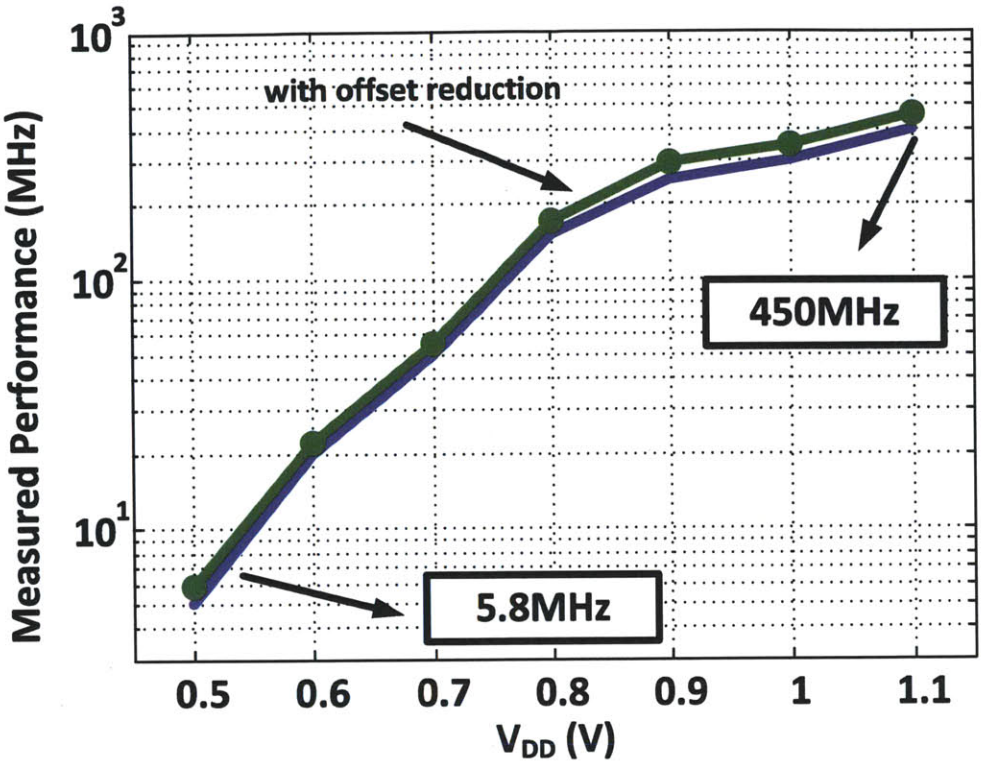


Figure 4-26: Measured performance vs. V_{DD} with and without offset reduction. SRAM performance scales from 450MHz down to 5.8MHz over the voltage range.

Since SRAMs are often designed with minimum-size devices to maximize transistor density, their operation is severely affected by process variation. At low voltages, the exacerbated effects of transistor variation limit SRAM operation due to functional failures related to the bit-cell or related to peripheral circuits such as sense-amplifiers. Since an SRAM consists of a large number of bit-cells and peripheral elements, it is essential to consider the worst-case process, voltage and temperature conditions on these circuits to ensure robust operation at low voltages.

Different bit-cell topologies and various peripheral assist circuits are considered in this chapter for achieving low voltage operation.

First, a 6T SRAM is considered. Trade-offs associated with achieving 0.6V operation by supporting an industry-standard 6T bit-cell with peripheral assist circuits in 28nm CMOS is analyzed. With this approach, no area overhead is introduced at the bit-cell level and overhead associated with peripheral circuits is amortized across the

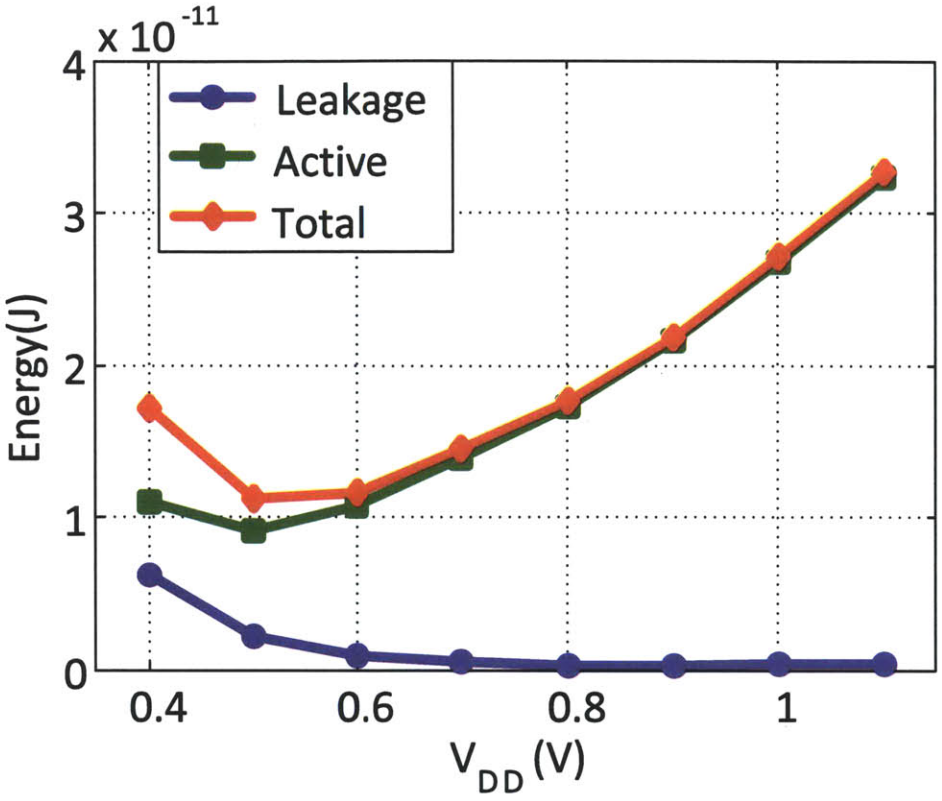


Figure 4-27: Measured energy vs. V_{DD} .

entire macro. Short local bit-lines, voltage boosting and low voltage oriented read path optimization ideas are developed and implemented in a test chip. Fabricated chips show functionality down to 0.6V while SRAM macros achieve an array efficiency of ~70% with <21% area overhead. In comparison, a conventional array employing the same bit-cell would start to fail below 0.9V and an 8T design achieving 0.6V would introduce 35-40% area overhead in the bit-cell.

8T bit-cell holds great promise for overcoming device variability in deeply scaled SRAMs and enabling aggressive voltage scaling. However, one major limitation of this topology is the half-select problem which occurs if an 8T bit-cell is used in a column-interleaved architecture. This led many 8T designs to be implemented without column-interleaving at the expense of potential area savings.

Second SRAM design presented in this chapter addresses this problem by introducing a new array architecture to allow column-interleaving for the 8T bit-cell. The

12% of area overhead introduced at the array can be justified by >20% area savings by 4-to-1 column-interleaving and shared sense-amplifiers (for the same sense-amplifier offset in both cases). Moreover, an on-chip reference selection loop is implemented to boost performance by ~10%. The 45nm design achieves functionality down to 0.5V with nearly 80× scaling in performance which makes this design very suitable for applications with large variations in throughput requirements.

It should be noted that the target V_{min} is extremely important and critical on the design decisions such as bit-cell topology and assist circuits. For designs with extremely low V_{min}, 8T bit-cell is the suitable option. However, although 8T bit-cell is regarded as the main low voltage option, this work demonstrated that a high-density bit-cell supported with assist circuits can achieve low voltage operation and be more area efficient than an 8T design. Hence, tailoring the SRAM design to the needs of an application provides the best trade-off in terms of area, power and performance.

Chapter 5

Motion Estimation Specific SRAM Design for Low-Power Operation

Application-specific designs can improve energy efficiency when compared to general-purpose designs as the former has the opportunity to optimize for the specific needs of a single scenario whereas the latter has to support a wider range of possible scenarios. Supporting a wider range of scenarios often results in a larger hardware and larger energy consumption.

The concept of hardware reconfigurability (e.g. reconfigurable SRAM in [46]) provides a balance between the two ends of the spectrum. By introducing some programmability to the hardware, a design can support more than one scenario while introducing minimal penalty in area and energy-efficiency.

Voltage scaling is an effective method to achieve energy savings at the expense of lower performance. An application-specific design, on the other hand, can provide the next level of energy consumption savings on top of the savings from voltage scaling. This can be very critical for applications running from a limited energy source. In this chapter, we will present the design of an SRAM that has been tailored at the transistor level to fit the features and necessities of its target application. It should be noted that although the SRAM design presented here is targeted for motion estimation, it can be used for other applications possessing similar features like motion estimation as well.

5.1 Motion Estimation Specific Features

In motion estimation, reference pixel buffers constitute the dominant portion of on-chip storage. In this section, we will make some observations and demonstrate analysis results of some specific features of SRAMs in motion estimation. These features will be critical in designing a circuit level low-power SRAM solution specific to this application.

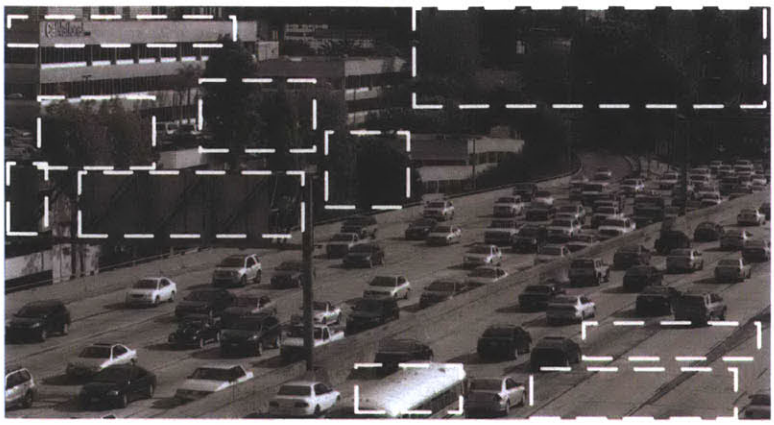
5.1.1 Correlation of Pixel Data in Reference Buffers

In hardware implementation of video encoders, motion estimation is performed on the luminance (Y) component of the frame data to reduce hardware complexity by reducing the number of computations and on-chip bandwidth. For the chrominance (U, V) components, MVs calculated for the corresponding Y component are used. In other words, cost calculations are performed on the Y component of the pixels.

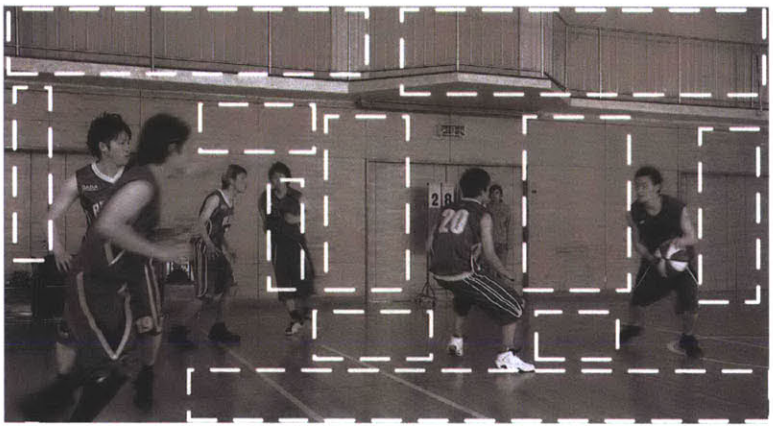
It should be noted that the contents of the reference buffer are portions of a previously coded frame. For example, for a block size of 64×64 and a search range of ± 64 in both directions, a block of 192×192 pixels of the reference frame is filled into the reference buffers. Since these pixels are coming from the same part of the frame, there is a good chance that these pixels belong to the same object or same background. Moreover, since only Y components are used in motion estimation, the luminance of a part of the frame can be highly correlated i.e. pixel values are similar.

Figure 5-1-a and Figure 5-1-b show two video frames and mark areas with high correlation of pixels. Traffic sequence in Figure 5-1-a consists of many small objects moving but still has large areas in the background where pixel luminance is correlated. Another example is BasketballDrive sequence in Figure 5-1-b. Although it is a sequence with fast motion, large background areas are present in the frames.

To quantify the correlation of pixels, a block average can be calculated for every 16×16 region and then the difference of each pixel value from the block average can be plotted. Figure 5-2 shows the distribution of these differences for the video frames in Figure 5-1.



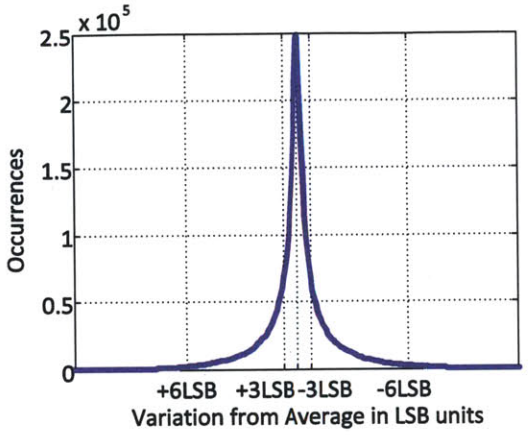
(a)



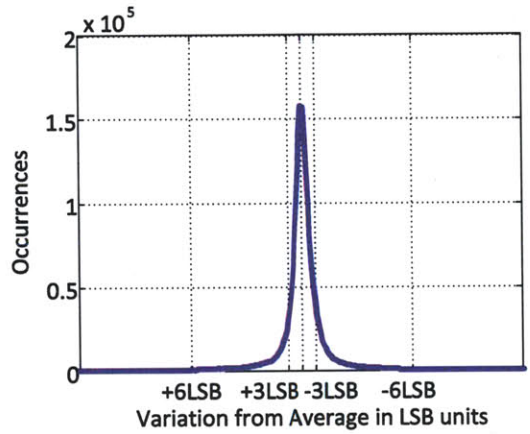
(b)

Figure 5-1: One frame from (a) 2560 × 1600 Traffic sequence and (b) 1920 × 1080 BasketballDrive sequence. Areas with high correlation of pixels are marked with white rectangles.

The distributions of differences from block average show that 58% and 76% of the pixels lie within $\pm 3LSB$ of the block average. In other words, out of 8 bits of a pixel, more than half of the bits can be the same with the block average. Of course, the binary representation of the pixels will result in the MSBs to switch at certain values. For example, from the binary representation of 127 ('01111111') to 128 ('10000000'), all bits do change. However, it should be noted that this is a corner case. Moreover, a simple mapping can be done by shifting every pixel's binary representation by a certain value [93] to reduce these effects.



(a)



(b)

Figure 5-2: Distribution of pixel values from 16×16 block average for (a) 2560×1600 Traffic sequence and (b) 1920×1080 BasketballDrive sequence.

5.1.2 Access Patterns for Reference Buffers

As explained in Chapter 2, data reuse between consecutive blocks is necessary for reducing off-chip bandwidth. Reusing the same data for more than one block results in the same data to be read multiple times from the reference buffer before it is overwritten. Moreover, depending on the implementation of the search algorithms, same reference data can be accessed multiple times even for the same block. An example is the fast search algorithm in HM-3.0 where multiple passes of refinement are done. This results in the number of read accesses to be higher than the number of write accesses for reference buffers in motion estimation. Consequently, energy

consumed during read accesses is more dominant in total energy consumption of the reference buffers.

For the CCE motion estimation algorithms presented in Chapter 2, read-to-write ratio in reference buffers is found to be slightly larger than three.

5.2 A 65nm Application-Specific SRAM Design Targeted for Motion Estimation

As explained in Section 5.1, SRAMs in motion estimation possess two important features. First, the data stored in the reference buffers is correlated. Secondly, the number of read accesses is higher than the number of write accesses so addressing read energy consumption is more critical for reference buffers. So a design taking these features into consideration can provide significant benefits in terms of energy-efficiency.

5.2.1 Contribution of Bit-line Switching to SRAM Power Consumption

From [94], during a read operation, SRAM power consumption can be given as

$$P_{ACC,RD} = C_{WL}V_{DD}^2 f + nC_{SA}V_{DD}^2 f + nC_{BL}V_{DD}\Delta V_{min} f$$

where f is the frequency of operation, C_{WL}, C_{SA} and C_{BL} are effective word-line, sense-amplifier and bit-line capacitances that are switching and n is the number of columns or word-length of the SRAM array. One important observation from this equation is as follows: for a large array with a large word-length (e.g. 64 or 128), the multiplicative factor n results in bit-line and sense-amplifier energy consumption to dominate total read energy consumption of SRAMs.

Analyzing the power consumption due to bit-line switching in SRAMs more closely gives power consumed by bit-line switching as

$$P_{consumed} = \alpha_{0\rightarrow1} \times C_{BL} \times V_{DD} \times \Delta V_{min} \times f$$

where $\alpha_{0\rightarrow1}$ is the activity factor and f is the frequency of operation.

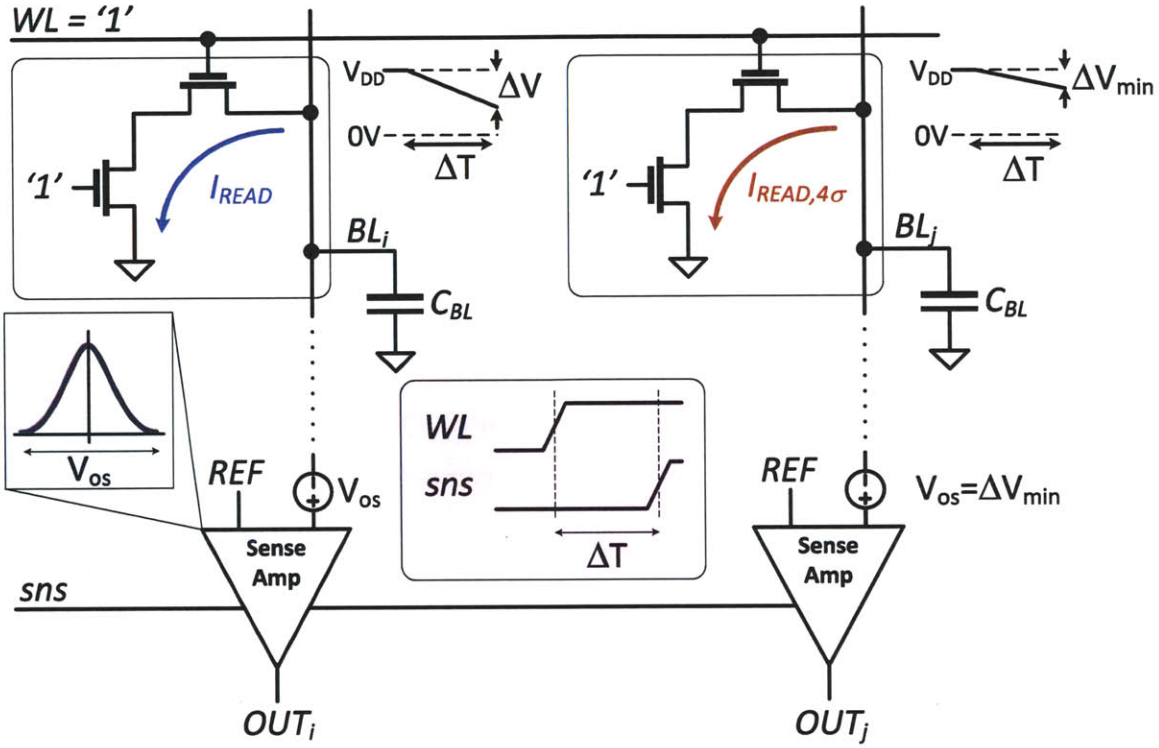


Figure 5-3: In an SRAM array, sense-amplifier activation time is set by the slowest bit-cell's read current to create a ΔV_{min} on the bit-lines. During this time, all other cells in the array discharge their bit-lines at a faster rate.

In high density arrays, bit-lines are shared across 256 or 512 bit-cells. The parasitic capacitance due to the devices connected to this signal as well as the metal parasitic capacitance due to routing of this signal contribute to the total capacitance of a bit-line. For a constant array size, bit-line capacitance can be assumed to be constant between different topologies.

Since bit-lines are used to read data from memory cells, during a read access, all bit-lines go through pre-charging and signal development phases. For 6T bit-cells, differential bit-lines result in one of the bit-lines to be actively pulled down during a read access. Hence, activity factor of one of the bit-lines in a 6T SRAM cell is 1. For an 8T bit-cell, discharging of RBLs depends on the data that is being read from

the bit-cell. Hence, the activity factor of bit-lines in an 8T SRAM is data dependent and can range from 0 to 1. It can be seen from the equation above that reducing the activity factor results in linear savings in power consumption due to bit-lines.

Finally, ΔV_{min} is set by the offset of the sense-amplifiers in the memory. For high-density arrays, sense-amplifier input referred offset can range from 50-150mV. Since there is an inverse relation between the area of transistors and transistor mismatches [88], input referred offset can be reduced by up-sizing the sense-amplifiers but this will result in an increase in the area of the SRAM. Moreover, up-sizing sense-amplifiers will result in their energy consumption to be larger as well.

It is important to note that ΔV_{min} voltage differential has to occur for all the bit-lines in the design to be able to perform a read operation without errors. Consequently, the slowest bit-cell in the array has to discharge its bit-line by ΔV_{min}, before sense-amplifiers are activated as shown in Figure 5-3. However, while waiting for the slowest cell in the array, all other cells discharge their bit-lines at a faster rate. Hence, effective ΔV for the array is larger than the minimum ΔV_{min} set by sense-amplifier input referred offset.

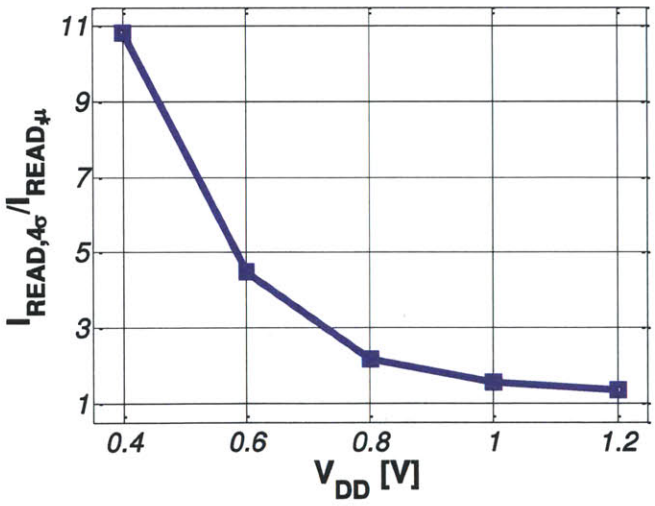


Figure 5-4: Ratio of the 4σ cell current to average cell current at different supply voltages.

This effect is even more prominent at low voltage levels where the difference

between the slowest cell and an average cell is larger. This is because the drive strength of transistors gets closer to an exponential relationship as V_{DD} approaches sub-V_T region and a shift in the threshold voltage causes a larger change in the transistor current at low voltages. Figure 5-4 plots the ratio of 4σ read current to nominal read current at different voltages. The ratio is very close to 1 at V_{DD} =1.2V but it becomes as large as 11 at 0.4V. For large arrays, even higher σs should be considered as explained in Chapter 1. Hence at low voltage levels, the large difference between slowest cell's current and average current causes almost all bit-lines to be discharged to 0V while waiting for the slowest cell to develop ΔV_{min} on its bit-line.

Based on the discussion above, reducing the switching activity on the bit-lines is the most effective way of reducing SRAM energy consumption, especially at low voltages where V_{DD} is close to the threshold voltage of transistors.

5.2.2 Prediction-Based Reduced Bit-line Switching Activity (PB-RBSA) Scheme

Since reducing bit-line activity is an important factor in reducing energy consumption of SRAMs and motion estimation data are correlated, a new bit-cell topology is proposed to reduce bit-line switching activity based on a prediction on the output bit.

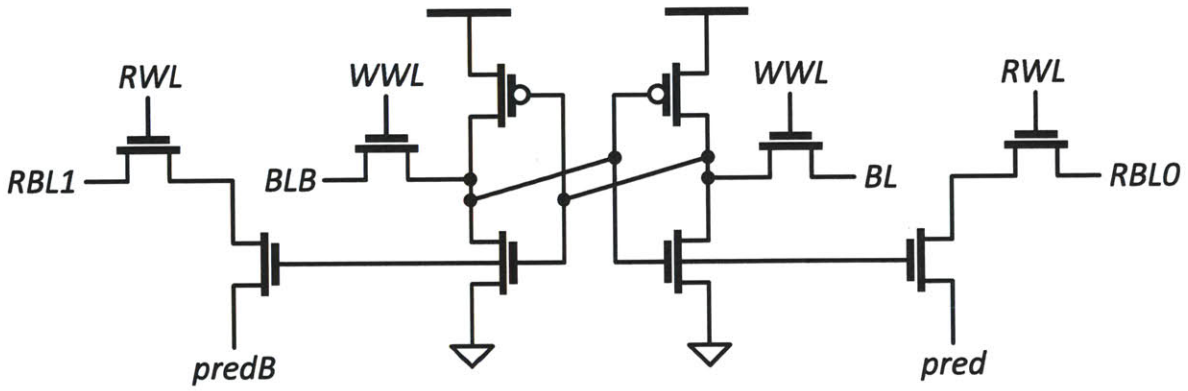


Figure 5-5: PB-RBSA bit-cell topology.

Figure 5-5 shows the PB-RBSA bit-cell topology. It consists of a cross-coupled

inverter pair, two NMOS access devices connected to the storage nodes and two read-buffers. The footer node of the read-buffers are not connected to ground, but connected to a predictor, pred and its complement predB. In other words, pred is a prediction of what the data stored in the bit-cell are.

Figure 5-6 shows the array architecture for the PB-RBSA SRAMs. pred/predB pair is routed in the column-wise direction and they are shared by the entire column of PB-RBSA bit-cells. Compared to an SRAM constructed with 8T bit-cells, an additional row-wise signal (RWL1) and three additional column-wise signals (pred, predB and RBL1) are introduced in this scheme.

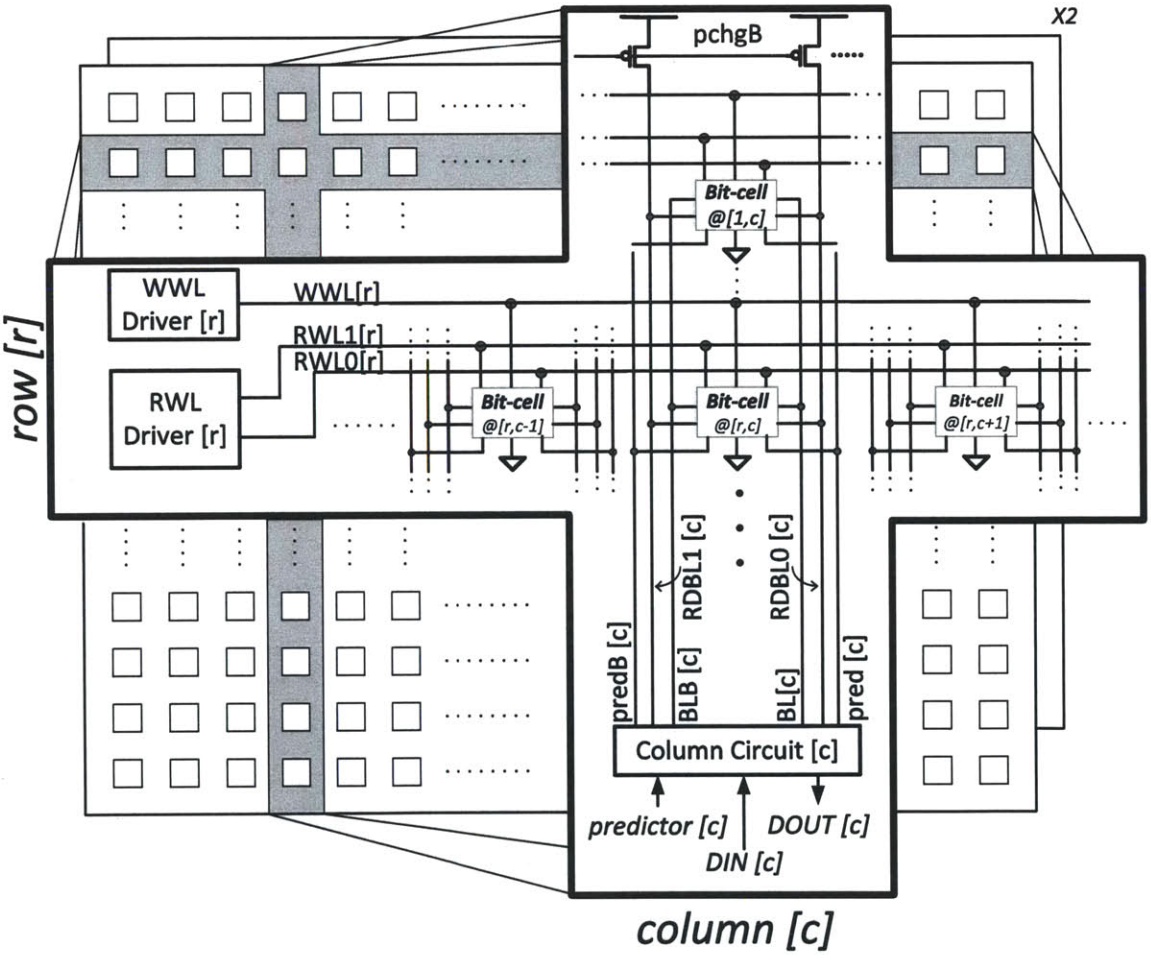


Figure 5-6: Array architecture for the PB-RBSA SRAM. WWL, RWL0 and RWL1 are routed in horizontal direction and BL/BLB, RBL0/RBL1 and pred/predB pairs are routed in vertical direction.

Data Retention

In data retention mode for PB-RBSA bit-cell, WWL and both RWLs are '0' and cross-coupled inverters hold the state of the bit-cell.

Write Operation

BL/BLB pair and WWL are used to exert a write operation on PB-RBSA bit-cell similar to a 6T or an 8T bit-cell. After BL and BLB are driven to correct voltages corresponding to the input data, WWL is asserted.

Read Operation

Read operation is performed through the read-buffers. At the beginning of a read cycle, RBL0 and RBL1 are pre-charged to V_{DD} and then RWL is asserted.

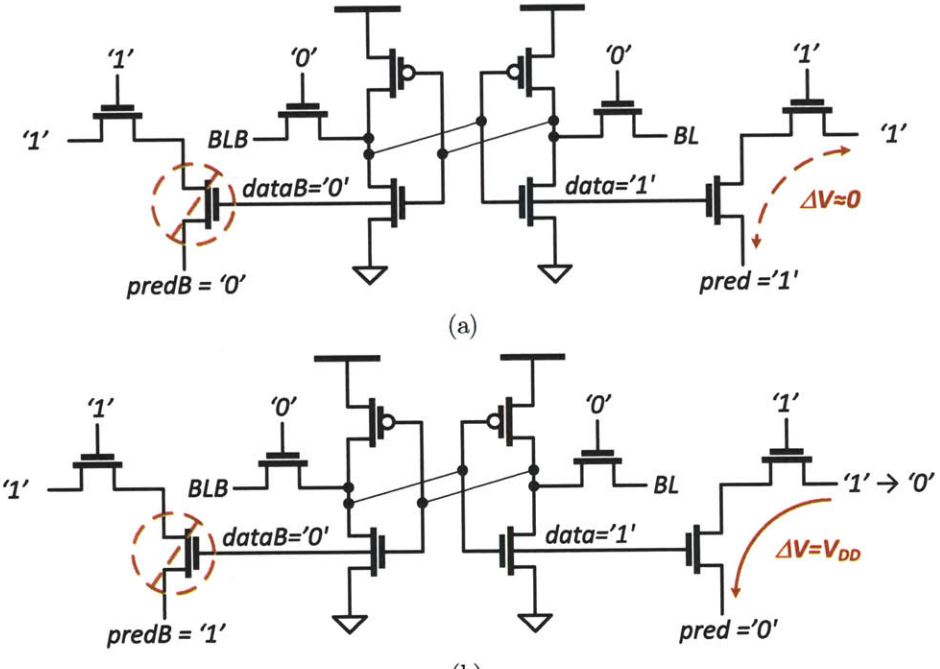


Figure 5-7: The cases when (a) pred is correct and matches the data in the bit-cell and (b) pred is incorrect.

If the prediction is correct (Figure 5-7-a), there is no voltage drop across one of the read-buffer (on the right in Figure 5-7-a) and one of the transistors is 'OFF' for

the second read-buffer (on the left in Figure 5-7-a). Consequently, both RBL0 and RBL1 stay at V_{DD} (ignoring the effect of leakage). In contrast, if the prediction is incorrect (Figure 5-7-b), one of the read-buffers drives its RBL to ground (on the right in Figure 5-7-b) and the other RBL stays at V_{DD} as one of the transistors in the stack is 'OFF' (on the left in Figure 5-7-b). Consequently, one of the RBL is discharged to ground in this case.

With PB-RBSA scheme, if the predictor is correct, both bit-lines stay at V_{DD} and energy consumption associated with charging these nodes back to V_{DD} can be prevented. For motion estimation where pixel values are highly correlated, a pixel-predictor can be calculated and used in this scheme.

In the column circuit, it can be determined if the predictor is correct or not through the evaluation of RBL0 and RBL1 being discharged to ground or not so data in the bit-cell can be inferred based on bit-line transitions and the value of pred.

5.2.3 Introducing Throughput Scalability to PB-RBSA Bit-cell

PB-RBSA bit-cell presented above has two read-buffers and these read-buffers can potentially be accessed independently by:

1. routing separate RWL signals to read-buffers, namely RWL0 and RWL1 and
2. forcing both pred and predB signals to '0' during a read access.

This change will enable a throughput-scalable SRAM with two modes:

Low-Power (LP) Mode In LP mode, PB-RBSA bit-cell is used with a predictor. RWL0 and RWL1 are asserted together for the same row and pred and predB signals are driven to the predictor and its complement. In this mode, SRAM can output one word/access but energy savings can be achieved through PB-RBSA scheme.

High-Throughput (HT) Mode In HT mode, PB-RBSA bit-cell is used without a predictor. RWL0 and RWL1 are independently asserted for different rows. pred

and predB are driven to '0'. In this mode, SRAM can output two words/access, doubling its read throughput. It should be noted that read-buffer connected to RBL1 and RWL1 is connected to the complement of the data in the bit-cell. Hence, RBL1 and REF should be connected to opposite terminals of the corresponding sense-amplifier.

Lastly, In layout implementation, PB-RBSA bit-cell requires 20% larger area compared to an 8T bit-cell due to the additional read-buffer. RDWL0 and RDWL1 are routed in the fifth metal layer and WWL is routed in third metal layer. Since the width of the cell is longer, BL, BLB, RBL0, RBL1, pred and predB are all routed in second metal layer.

5.2.4 Hierarchical Sensing and Statistical Sense-Amplifier Gating

Since PB-RBSA scheme needs two separate read-buffers and RBLs, two separate sense-amplifiers are necessary. A differential sense-amplifier cannot work as both RBLs can stay at '1' if the prediction is correct in LP mode. Having two sense-amplifiers results in extra energy consumption in the column circuit so a hierarchical sensing and sense-amplifier gating idea is implemented in this work.

Figure 5-8 shows a straightforward way of implementing read path from PB-RBSA bit-cell to the sensing network. Pseudo-differential sense-amplifiers are used with global reference voltage (REF). Two main sense-amplifiers (M-SA) are directly connected to RBLs and activated by snsB signal.

In LP mode, RBL connected to the read-buffer with its footer (pred or predB) driven to '0' has a chance of discharging to ground. To sense this voltage development, one of the M-SAs is turned on. In HT mode, both M-SAs are activated as signal development on both RBLs need to be resolved.

Schematic of M-SAs are given in Figure 5-8 and their offset distribution is given in Figure 5-9. M-SAs are designed to have a $\pm 3\sigma$ tail-to-tail offset distribution of 100mV. Hence, REF voltage can be set to V_{DD} -50mV and ΔV_{min} can be 100mV.

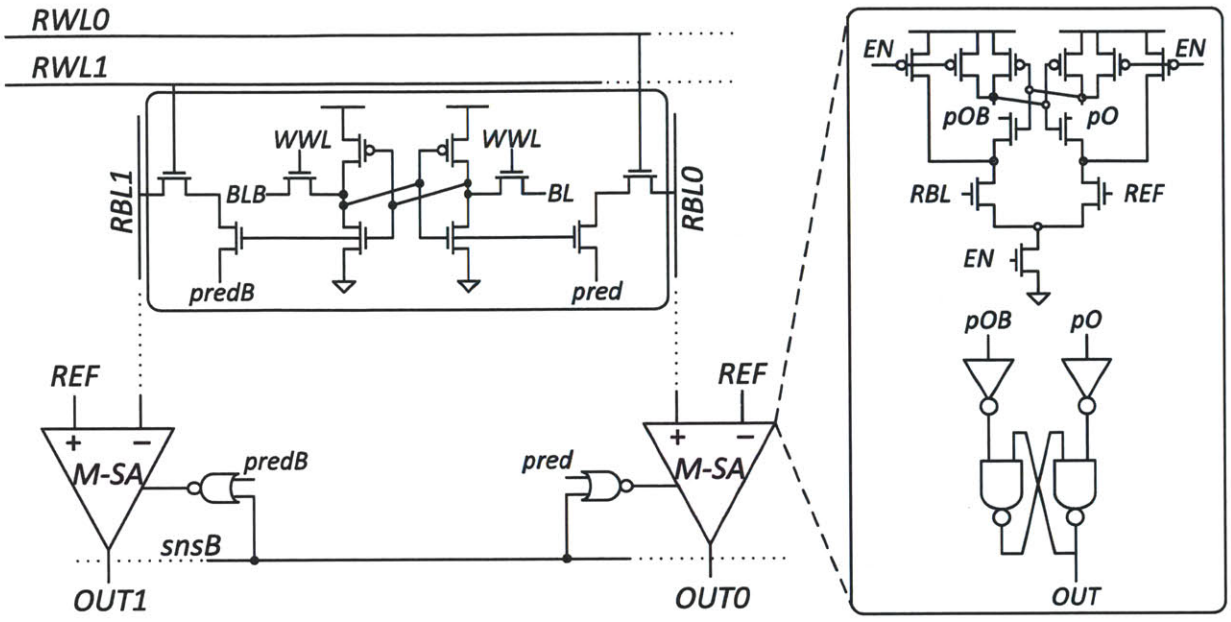


Figure 5-8: A straightforward implementation of the sensing network employing two sense-amplifiers with a global reference voltage, REF.

With the LP mode in PB-RBSA scheme, RBLs will have a higher probability of staying at V_{DD} provided that a good predictor can be generated for the data in the SRAM. This motivates for designing a hierarchical-sensing technique and gating M-SA with the output of a smaller sense-amplifier that is designed to favor an output of '1' over '0' in its offset distribution.

Figure 5-10 shows the hierarchical sensing network implemented in this work. Early decision sense-amplifier (E-SA) have the same topology as M-SA but they are sized to be smaller than M-SAs so that their energy consumption is correspondingly smaller than M-SA. However, this results into a larger offset distribution for E-SA and can limit the performance of the SRAM if E-SA starts limiting ΔV_{min}.

If E-SAs are used in the sensing network alone to resolve RBLs with ΔV_{min} of 100mV, some sense-amplifiers would give erroneous results. However, in this work, we use E-SAs to decide if M-SA need to be activated or not. The ambiguity in the output of the E-SA can be avoided by skewing its input referred offset and enabling M-SA every time E-SA's output is ambiguous.

E-SAs are sized to be 3× smaller than M-SA. Moreover, its offset distribution

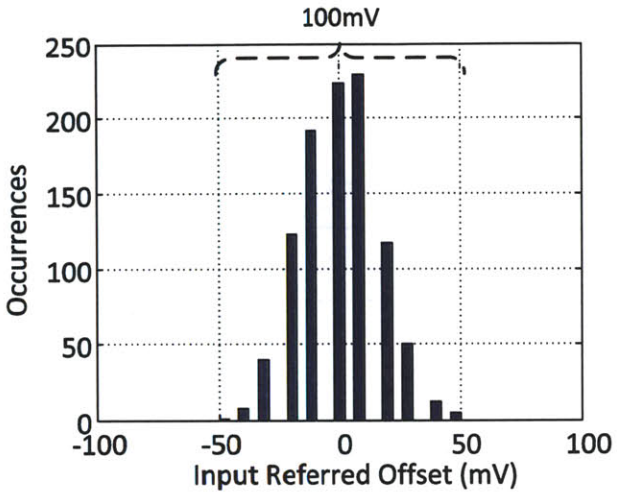


Figure 5-9: Offset distribution of M-SA used in this design. $\pm 3\sigma$ tail-to-tail offset distribution is designed to be 100mV.

is skewed by making one of its input transistors stronger than the other one. The resulting offset distribution is shown in Figure 5-11. $\pm 3\sigma$ tail-to-tail offset distribution is 150mV for E-SA and it is skewed towards negative offsets.

For E-SAs not to limit ΔV_{min} of the SRAM, REFEARLY voltage should be set to a higher voltage, V_{DD} . In this case, a 100mV of ΔV can be sensed by E-SA and M-SA can be turned on to ensure that RBL is actually discharged. Because of the larger offset distribution, sense-amplifiers that have a positive offset in Figure 5-11 will output a '1' erroneously and result in M-SA to be activated unnecessarily. The percentage of sense-amplifiers with a positive offset from Figure 5-11 is only 5%. Hence, 5% of the time, M-SA will be activated to resolve that RBL has actually not been discharged.

Figure 5-12 summarizes the energy consumed in different cases when operating in LP mode. Three main cases are considered and energy consumed by the sensing network is given in each case.

Predictor Correct, E-SA with Negative Offset When predictor is correct, both RBLs stay at V_{DD} . For E-SAs with a negative input referred offset ($REFEARLY = V_{DD}$), E-SA can resolve correctly that RBL stayed high and does not activate M-SA. In this case, only E-SA is activated and energy consumed is due to E-SA

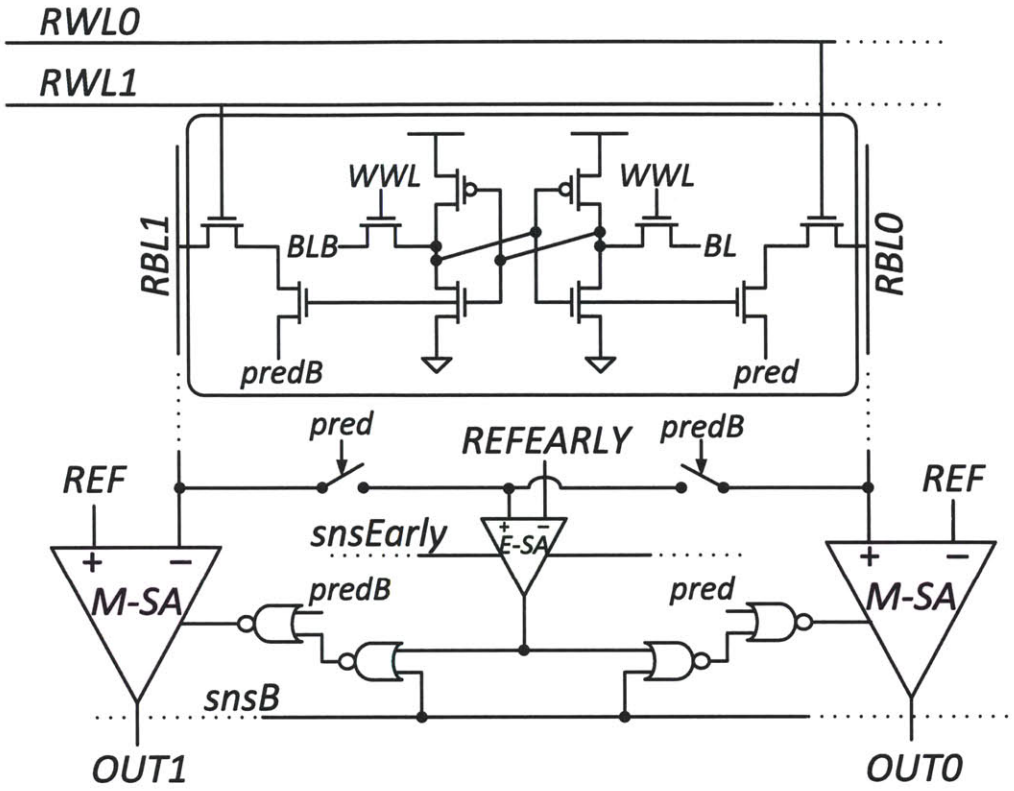


Figure 5-10: The hierarchical sensing network design implemented in this work. E-SAs are sized to be 3× smaller compared to M-SAs.

only.

Predictor Correct, E-SA with Positive Offset Again, when predictor is correct, both RBLs stay at V_{DD} . For E-SAs with a positive input referred offset ($REFEARLY = V_{DD}$), E-SA resolves this situation incorrectly and activates M-SA. In this case, both E-SA and M-SA are activated.

Predictor Incorrect When predictor is incorrect, one of the RBLs is discharged by ΔV_{min}. E-SAs can resolve this correctly and activate M-SA. In this case, both E-SA and M-SA are activated and energy consumed is the sum of both sense-amplifiers.

Although it looks like energy consumed is larger than the straightforward implementation (where one M-SA is activated every cycle), signal statistics play an important role when computing the actual energy consumed.

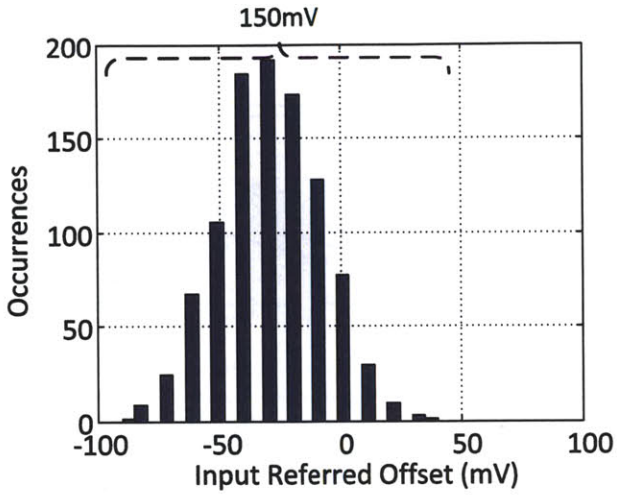


Figure 5-11: Offset distribution of E-SA used in this design. $\pm 3\sigma$ tail-to-tail offset distribution is designed to be 150mV and distribution is skewed towards negative offsets.

		Low Power Mode	
		pred - Correct	*pred - Incorrect*
E-SA w/ (-) offset	*M-SA gated?*	Yes	No
	Energy Consumed	$C_{E\text{-}SA}V_{DD}^2$	$C_{E\text{-}SA}V_{DD}^2 + C_{M\text{-}SA}V_{DD}^2$
E-SA w/ (+) offset	*M-SA gated?*	No	No
	Energy Consumed	$C_{E\text{-}SA}V_{DD}^2 + C_{M\text{-}SA}V_{DD}^2$	$C_{E\text{-}SA}V_{DD}^2 + C_{M\text{-}SA}V_{DD}^2$

Figure 5-12: Energy consumed in the sensing network when operating in LP mode with different cases considered.

First, because of the PB-RBSA scheme and pixel data being correlated, the cases where predictor is correct should out-number the opposite cases. Secondly, in the array, there are 19× more sense-amplifiers with negative offset than with positive offset. So, when predictor is correct, 95% of the time, E-SAs will resolve the RBL correctly and will not activate M-SA unnecessarily.

If we formulate the energy consumed in the sensing network:

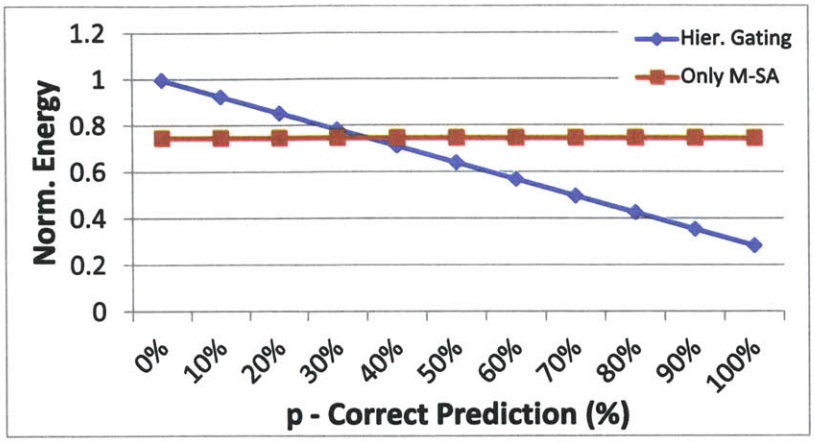


Figure 5-13: Normalized energy consumed in the sensing-network with proposed hierarchical sensing scheme and with the straightforward M-SA only scheme.

$$E_{sensing} = p \times (C_{E-SA}V_{DD}^2 \times 0.95 + (C_{E-SA}V_{DD}^2 + C_{M-SA}V_{DD}^2) \times 0.05) + (1-p) \times (C_{E-SA}V_{DD}^2 + C_{M-SA}V_{DD}^2) \quad (5.1)$$

where p is the correct prediction percentage. Figure 5-13 plots the energy consumption in the sensing-network with respect to p. For the straightforward method explained in Figure 5-8 to out-perform our proposed solution, p should be less than 35%.

It should be noted that this scheme can be easily turned-off if the predictor bits are correct for $< 35\%$ by measuring the change in predictor. Moreover, by setting the predictor as '00000000' or '11111111', one can assume to have roughly 50% matching if the bits in the data is totally random.

The delay increase due to sense-amplifier gating is small ($< 8\%$) when compared to the time that is allocated for bit-line signal development.

5.2.5 Predictor Generation

To maximize savings with PB-RBSA scheme, a good predictor is needed to be calculated and input to the SRAMs. However, it should be noted that pred and predB

signals are column-wise signals that are connected to the source/drain region of the read-buffer devices from all rows. Effectively, pred and predB signals have a very similar capacitance to RBL0 and RBL1. Consequently, switching activity of pred and predB signals can be a dominant factor of energy consumed in PB-RBSA if their activity factor is large.

Since pixel data are correlated on large blocks in a video frame, pred and predB can be updated much less frequently. Hence, energy consumed in switching of pred and predB can be amortized across many cycles.

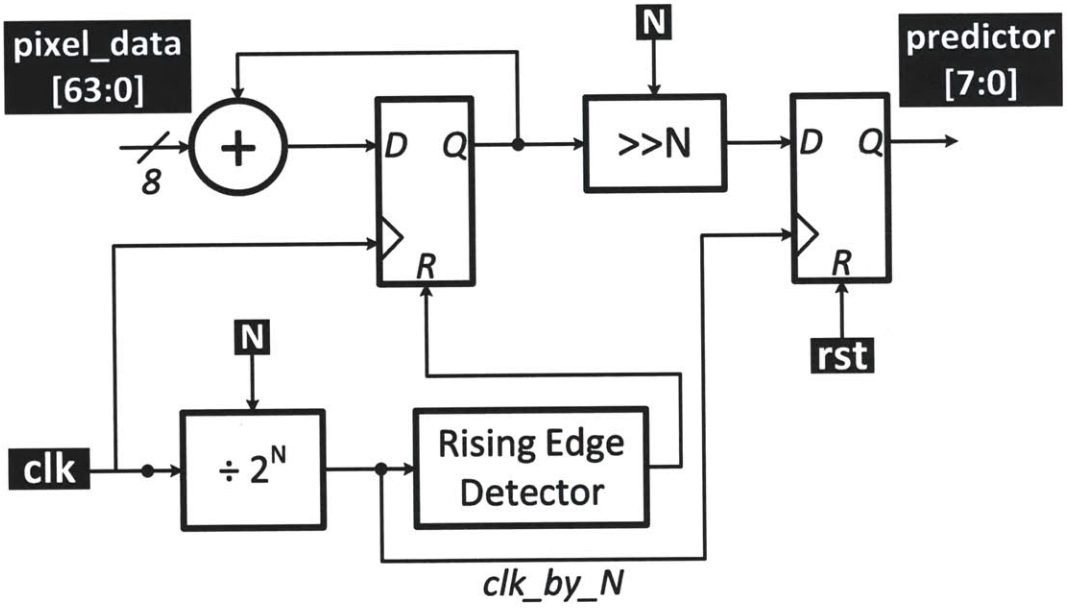


Figure 5-14: Predictor generation circuit used in this design.

In this work, we considered an arithmetic average of the output pixels from the SRAMs as the predictor. Figure 5-14 shows the implementation of this circuit. Although the sum of pixels is accumulated every cycle, it is updated every 2^N cycles and hence activity factors of bits of pred and predB are bounded by

$$\alpha_{0\to1} < \frac{1}{2^N}$$

There is an interesting trade-off between the selection of N and the lag of the average calculation. If N is selected to be a large number (e.g. 8), then predictor will be updated less frequently (every 2^N cycles) and cannot track recent changes in

the pixel data. This can cause PB-RBSA scheme to have a predictor that is often incorrect. Choosing N to be a smaller value (e.g. 2), can enable the predictor to track changes more frequently but this will increase the activity factor of pred/predB in the SRAM and cause extra energy consumption due to averaging a small number of pixels.

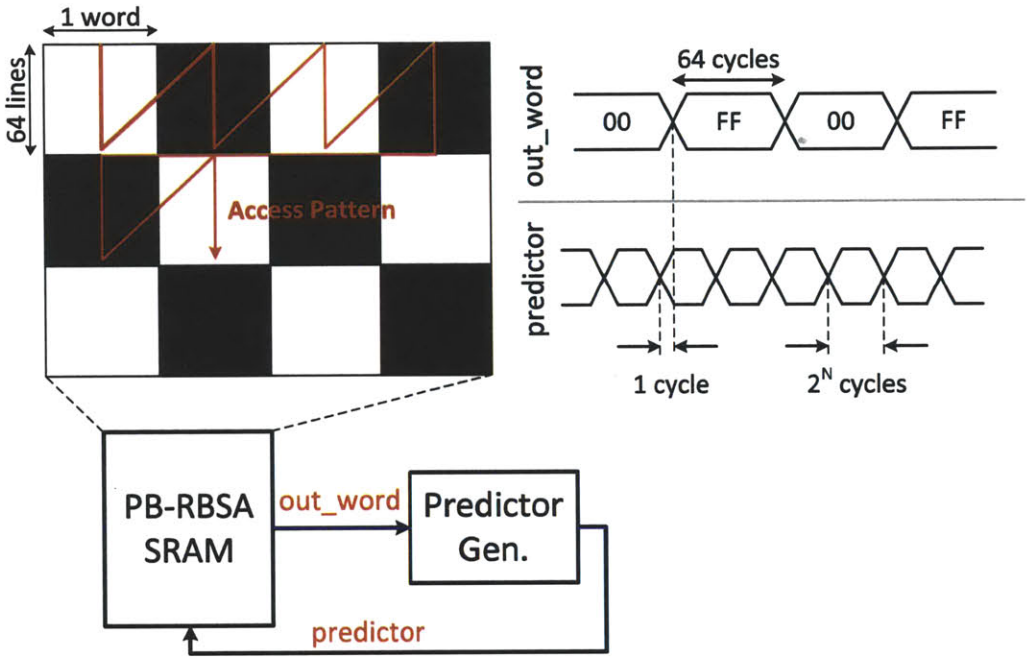


Figure 5-15: A scenario where SRAM is filled with blocks of white and black pixels.

Figure 5-15 shows an example corner case where a frame consists of 64×64 blocks of alternating black and white regions. SRAMs are accessed to read 64 rows of white regions and then 64 rows of black regions and for simplicity energy consumption of write accesses is not included. By using the pixel average generator updated every 2^N cycles, a prediction is calculated and input to the SRAMs in this work. Moreover, let's assume the output of the predictor generator changes one cycle before the access of a white or black block is finished.

To capture the effect of N, measured power consumption is plotted in Figure 5-16. Because of the conflicting trade-offs of the selection of N explained above, power makes a minimum around $N = 3$ ($2^N = 8$). Optimum selection of N provides 1.6× savings in energy/access when compared to the case where N is selected to be 6 (i.e.

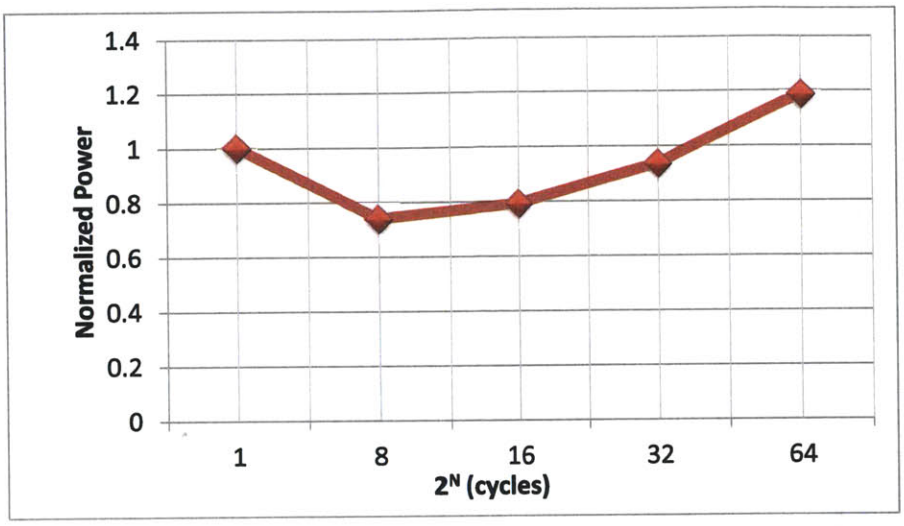


Figure 5-16: Normalized measured power consumption of the PB-RBSA SRAMs when operating under the scenario described in Figure 5-15.

$2^N = 64$).

Finally, it should be emphasized that predictor generation is an important factor determining the energy savings in this scheme. Arithmetic average is chosen to be a simple yet efficient solution to predictor generation. However, more complex predictor generation schemes are possible at the expense of a larger hardware area. An example is a weighted running average calculator which favors more recent outputs over past outputs.

Arithmetic average calculation block can be shared by two blocks in this work and introduces an overhead of 3% in layout . This overhead can be smaller if there are more than two blocks and predictor generation circuit is shared across many blocks.

5.2.6 Test Chip Architecture

Ideas presented in this section are implemented in a test chip in 65nm CMOS low-power process. Top level test chip architecture is shown in Figure 5-17. Two blocks of motion estimation specific SRAMs as well as two blocks of conventional 8T SRAMs are placed in the test chip with the same inputs so that the outputs can be compared against each other and their power consumption can be measured and compared as well. Blocks are organized as 256 rows and 64 columns.

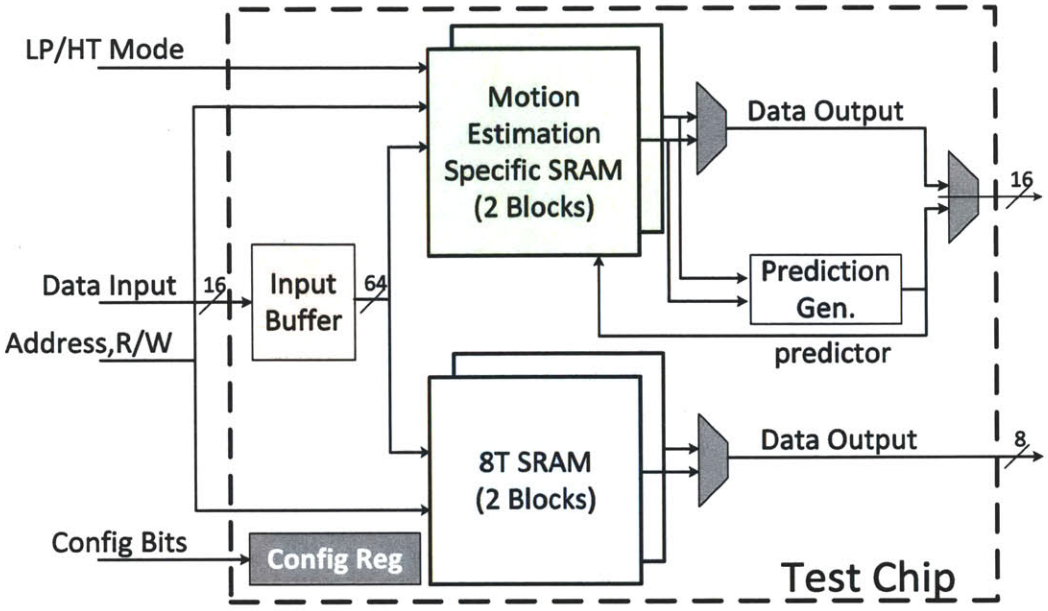


Figure 5-17: High level architecture of the test chip fabricated in 65nm low-power CMOS process.

Data inputs are provided from off-chip pins. Since only 16 pins can be allocated for this, a shift register structure is used to buffer input bits for four cycles before writing them to the SRAMs. For outputs, multiplexer trees are used to select two 8-bit outputs from motion estimation specific SRAMs and one 8-bit output from 8T SRAMs. Output of the on-chip predictor generator can also be output to off-chip.

Finally a configuration register is also placed on the chip to set various configuration bits in the design to ease testing and debugging.

5.2.7 Measurement Results

A die photograph of the test chip is shown in Figure 5-18. The die size is $2.3mm \times 2.3mm$ and total number of pads is 100. Table 5.1 provides a summary of the test chip specifications.

Figure 5-19 plots measured energy/access numbers for the test chip at 0.6V for varying correct prediction percentage. Energy/access values are normalized to energy/access at 100% correct prediction.

Measurements are done to achieve no errors in the memory by sweeping the word-

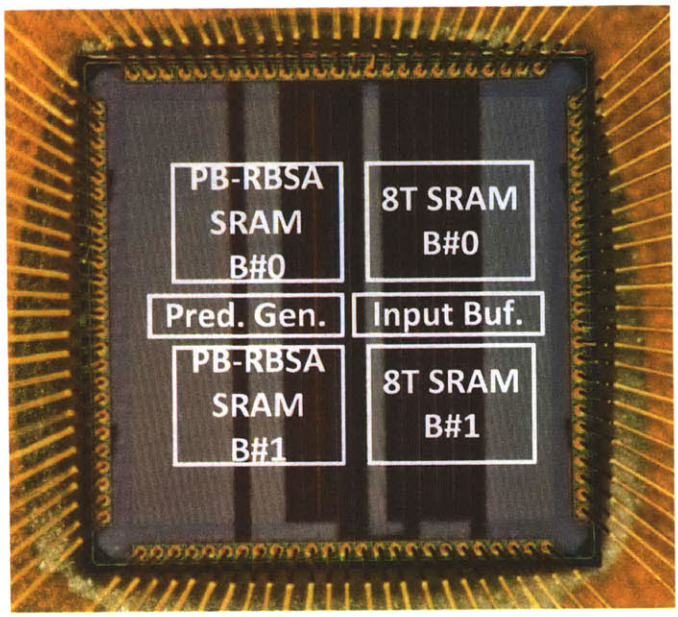


Figure 5-18: Die photograph of the 65nm test chip.

Technology	65nm Low-Power CMOS
Die Size	2.3mm × 2.3mm
Number of Pads	100
Total SRAM on die	32Kbits of PB-RBSA & 8T SRAMs
SRAM Macro Organization	512 words × 64bits/words
Voltage Range	0.52V - 1.2V

Table 5.1: Summary of the 65nm PB-RBSA test chip.

line activation time to the point where the worst-case cell is capable of discharging its RBLs by ΔV_{min}. Correct prediction percentage values from 50% to 100% are shown. With increasing correct prediction percentage, energy/access is reduced as this corresponds to reduced RBL transitions.

Also plotted in Figure 5-19 is the energy/access when SRAMs are operated in high-throughput mode. It should be noted that in this mode, read bandwidth of the memories is two times larger than the read bandwidth in low-power mode. It is important to emphasize that this feature introduces almost no area overhead as HT mode is fully compatible with the original PB-RBSA scheme idea and is beneficial

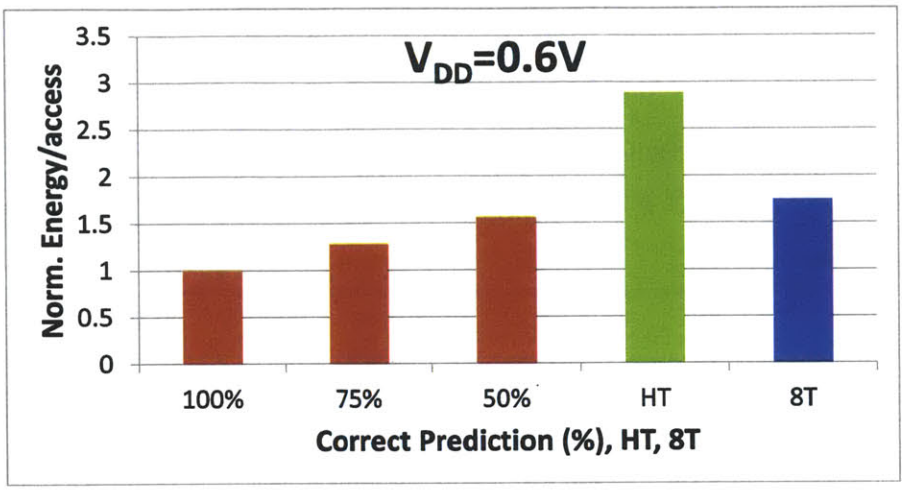


Figure 5-19: Measured energy/access with respect to correct prediction percentage at V_{DD} = 0.6V. Energy/access numbers are normalized to energy/access with 100% correct prediction. HT denotes measured energy/access in high-throughput mode and 8T denotes measured energy/access of 8T SRAMs.

in applications where slight changes in read bandwidth can be adjusted by switching the operating mode rather than the supply voltage.

To provide a comparison between PB-RBSA SRAMs and a conventional 8T SRAM, last bar graph provides energy/access numbers at V_{DD} = 0.6V for the 8T SRAMs in the prototype chip. It should be noted that 8T energy/access is data dependent and bit-line switching activity depends on the data stored in the bit-cell being a '0' or a '1'. Here an equal distribution of '0's and '1's are used in the array to achieve the $\alpha_{0\rightarrow1,8T} = 0.5$. With this activity factor, 8T SRAM energy/access is 1.75× larger than PB-RBSA SRAMs with 100% correct prediction.

It should be noted that when accessing reference frames, neither the activity factor of the 8T will be exactly 0.5 nor PB-RBSA scheme will achieve 100% correct prediction. Hence, energy savings with PB-RBSA SRAMs will be different from one frame to the other.

To capture this, 1100 frames with resolutions ranging from 1280 × 720 to 2560 × 1600 are provided to the prototype test chip to measure energy/access for the PB-RBSA SRAMs and for the 8T SRAMs. Figure 5-20 shows the distribution of the PB-RBSA SRAM's energy savings with respect to the 8T SRAM. The savings achieved

over the 8T SRAM can be up to 1.9×. Moreover, for sequences with higher resolution, energy savings are found to be higher. This is due to the fact that objects are represented with a larger number of pixels in higher resolution sequences and consequently correlation of pixels is higher.

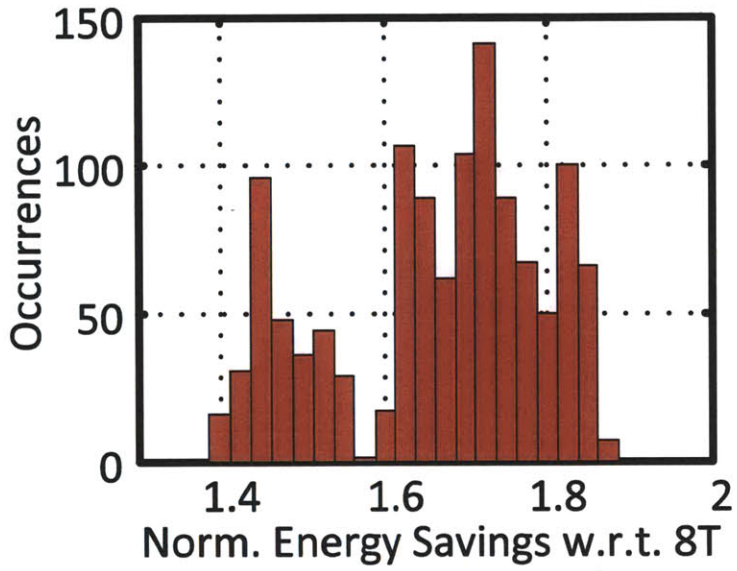
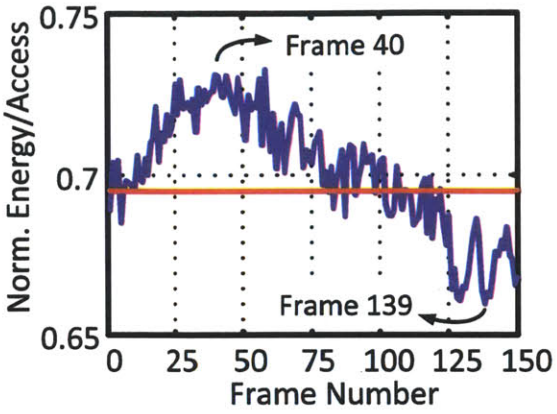


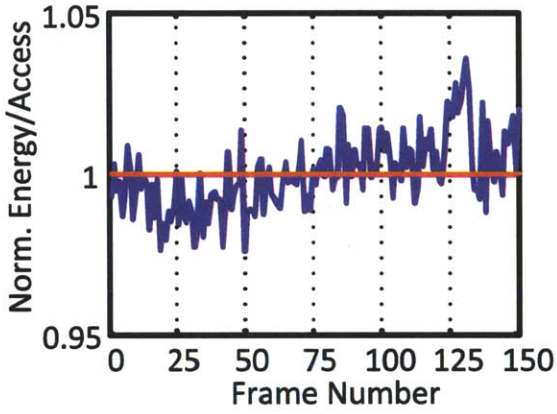
Figure 5-20: Distribution of PB-RBSA SRAM's energy savings with respect to the 8T SRAM for 1100 different video frames with resolutions ranging from 1280 × 720 to 2560 × 1600.

Figure 5-21-a and Figure 5-21-b shows measured energy/access numbers for PB-RBSA SRAM and 8T SRAM respectively both normalized to the average energy/access of the 8T SRAMs. Energy/access numbers are shown for a 416 × 240 sequence for 150 frames. Figure 5-21-c and Figure 5-21-d shows the contents of the 40th image frame and 139th image frame respectively. Energy/access of the PB-RBSA SRAM changes as the contents of the image frames change. When the image frame consists of smoother objects predictor for PB-RBSA scheme works better and energy/access goes down. It should be noted that for the 8T SRAMs, energy/access depends on the number of '0's and '1's in the pixels and it does not change significantly from one frame to the next.

Figure 5-22 shows the effect of the predictor accuracy on energy/access for the PB-RBSA SRAM. Specifically, energy/access results for N=4 and N=6 are provided where



(a)



(b)

(c)

(d)

Figure 5-21: Measured energy/access numbers for (a) PB-RBSA SRAM and (b) 8T SRAM normalized to the average energy/access of the 8T SRAMs for a 416×240 video sequence for 150 frames. Red lines show the average for each SRAM implementation. (c) 40th and (d) 139th frames of the sequence are also shown.

predictor is updated every 2^N cycles with the predictor generation circuitry given in Figure 5-14. For the same sequence in Figure 5-21, when the predictor is updated less frequently the prediction accuracy is degraded and consequently energy/access goes up.

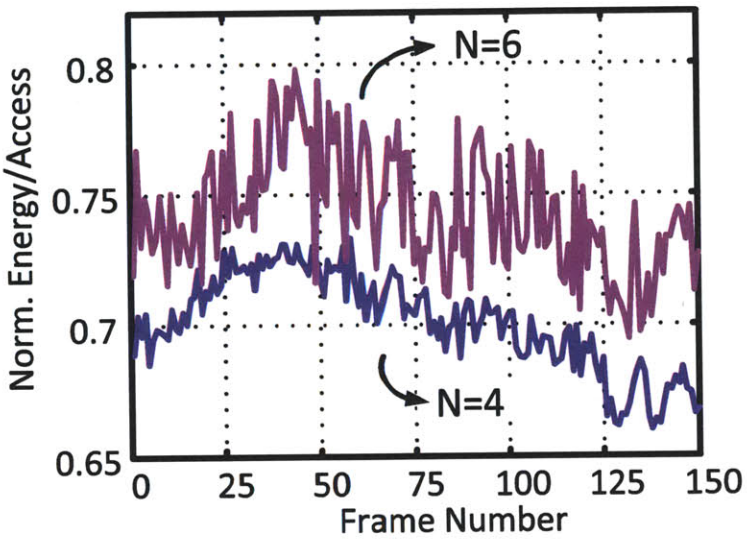


Figure 5-22: Effect of predictor accuracy on energy/access for the PB-RBSA SRAM.

Figure 5-23 plots measured energy/access numbers, again at V_{DD} = 0.6V with and without sense-amplifier gating. Similar to Figure 5-19, values are normalized to energy/access at 100% correct prediction with sense-amplifier gating.

SA gating provides ~20% energy savings for the overall SRAM when correct prediction rate is 100%. With smaller rate of correct prediction, sense-amplifier gating begins to be more costly. The break-even point for correct prediction percentage happens at nearly 40%.

Figure 5-24 plots measured frequency of operation vs. V_{DD} when operating in LP mode. SRAMs achieve functionality down to 0.52V where timing violations and retention problems begin to occur. SRAM performance at 1.2V is 145 MHz and it scales down to 0.11MHz at 0.52V. It should be noted that SRAM performance is independent of the operation mode as the same timing generation circuit is used in both modes. In LP mode, an additional signal for early sensing is generated and sent

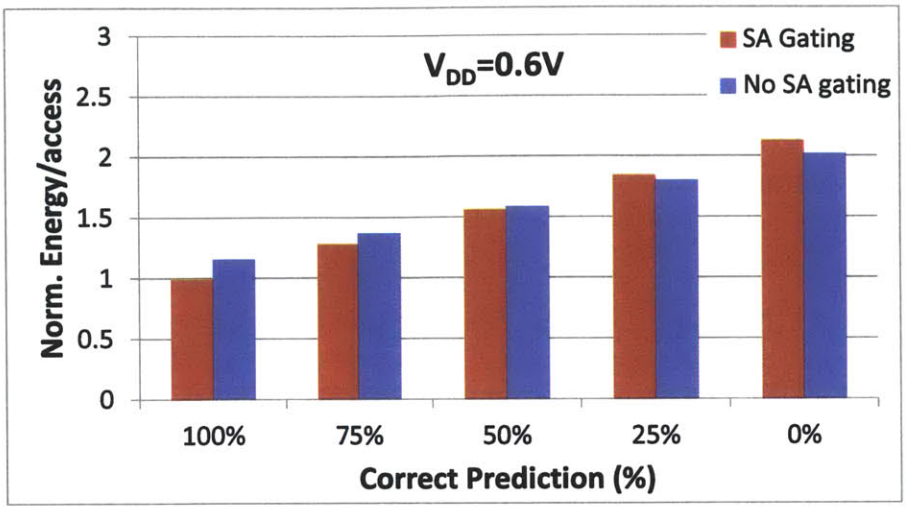


Figure 5-23: Measured energy/access with and without sense-amplifier gating at V_{DD} = 0.6V. Energy/access numbers are normalized to energy/access with 100% correct prediction.

to the column circuits and the effect of this additional path on total timing is $< 8\%$ at 1.2V.

5.3 Summary and Conclusions

Designing circuits to be application specific can provide significant improvements in terms of power and performance by

- optimizing the design for a specific target and
- exploiting the specific features of the application.

For example, supporting processors with hardware accelerators for specific tasks results in achieving orders of magnitude savings in power [95]. This idea of application specific hardware, however, should not limited to algorithm and architecture level and can be implemented at the circuits level as well. In this chapter, application specific SRAM design is investigated for motion estimation application and a power reduction technique is developed for applications where input data are correlated.

First, prediction-based reduced bit-line switching activity (PB-RBSA) scheme is proposed in this chapter to exploit the correlation of input data to the memories.

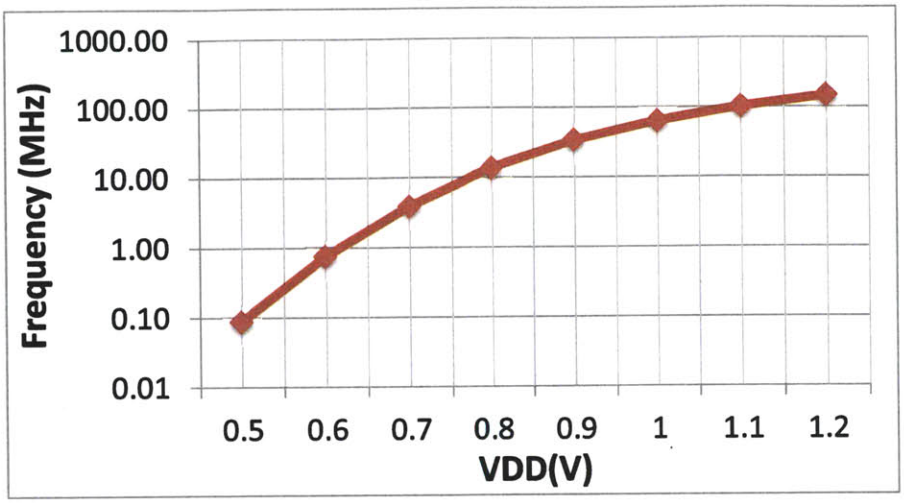


Figure 5-24: Measured SRAM performance for the 65nm test chip.

Specifically, PB-RBSA scheme introduces a predictor for the output of the SRAM and bit-line transitions are avoided when the predictor is correct. To complement this idea, a hierarchical sensing network with sense-amplifier gating is developed to take advantage of the biased transition probability on the bit-lines due to PB-RBSA. A small sense-amplifier that is intentionally designed with a non-symmetric input offset distribution is used to do a pre-evaluation of the bit-line and to gate M-SAs.

Proposed techniques are implemented in a 65nm prototype which is tested for functionality down to 0.52V. PB-RBSA scheme with sense-amplifier gating provides up to 1.9× energy reduction with respect to a similar 8T design that is also implemented on the same test chip. To achieve a similar energy reduction through only voltage scaling would require an additional 1.4× scaling of V_{DD} from $C \times V_{DD}^2$ formula. However, this would result in slower performance and potential assist circuits requiring additional silicon area.

It should be noted that energy savings achieved through application-specific SRAM design is the next level of savings on top of the savings from voltage scaling. In other words, PB-RBSA scheme does not prevent SRAMs to do voltage scaling but this scheme enables a completely new dimension for savings that can be achieved by using application-specific features.

Chapter 6

Conclusions and Future Work

This thesis presented design optimizations for cost and coding efficient implementation of a motion estimation block at three different levels of the design: at algorithms, architectures and transistor-level circuits. First, a methodology is presented to characterize hardware cost vs. coding efficiency. This analysis is used to quantify design trade-offs between algorithm and architecture decisions for their hardware complexity and coding impact. Depending on the priorities of different designs in terms of silicon area, power consumption and coding efficiency, optimum decisions can be made. Second, hardware-oriented algorithms are developed to reduce hardware area and data bandwidth. Cost and coding efficient (CCE) implementation provided orders of magnitude reduction in hardware complexity. Considering the hardware implementation cost of algorithms at the design stage can provide valuable insight to where the bottlenecks occur and how these problems can be addressed.

At the circuit level, SRAM design considerations are optimized for voltage scalability and performance with minimum impact on array efficiency. Design decisions in terms of circuit topologies and low voltage assist techniques are based on target operating voltage and frequency range. Lastly, application-specific SRAM design for motion estimation is presented. This approach using the correlation of storage data in SRAMs to predict future outputs is not limited to motion estimation and can be generalized to other applications as well. Application-specific SRAMs can provide next level of savings in energy/access on top of the savings from voltage scaling and

maximize overall energy efficiency at the circuit level. Incorporating signal statistics into transistor-level circuit design can provide a new dimension for circuit designers to explore.

6.1 Summary of Contributions

In this section, we will summarize the key ideas presented in this thesis.

This thesis focuses on hardware-oriented algorithm and architecture development for motion estimation. At circuit level, low-power SRAM design techniques are explored.

6.1.1 Hardware-Oriented Algorithms and Architectures for Motion Estimation

Motion estimation hardware cost vs. coding efficiency analysis for HEVC

- Development of a methodology to quantify hardware cost (in terms of core/memory area and data bandwidth) vs. coding efficiency for next generation video coding standard, HEVC.
- Trade-off analysis for the effect of supported block sizes in HEVC motion estimation on coding performance and hardware complexity.

Cost and Coding Efficient (CCE) Motion Estimation for HEVC

- CCE motion estimation design with hardware-oriented search algorithms to address hardware area and memory bandwidth. With respect to the anchor configuration, 4.3× core area and 5.3× on-chip memory area, 56× on-chip bandwidth and 151× off-chip bandwidth savings are achieved.
- Hardware-oriented advanced motion vector prediction (AMVP) algorithm. 2× reduction in hardware area is achieved.
- Adoption of proposed AMVP algorithm to HEVC.

Highly-Parallel Multi-Standard Motion Estimation Design

- Development of hardware-oriented search algorithms suitable for frame and MB parallel motion estimation.

- Design of a reconfigurable motion estimation engine supporting two standards (AVC/H.264 and VC-1) to maximize hardware sharing between standards. 85% and 40% of hardware is shared across standards in integer and fractional motion estimation parts respectively.

6.1.2 Low-Power SRAM Design

- Development of area-efficient assist circuits for a high-density 6T SRAM and demonstration of these ideas in a 28nm test chip. This work presented one of the first published SRAMs in 28nm to work down to 0.6V.

- Development of an array architecture enabling column-interleaving for 8T designs and an on-chip reference selection loop to reduce sense-amplifier offset.

- Implementation of a 45nm test chip featuring an 8T design using the techniques above. 0.5V-1.1V operating voltage range is demonstrated for this work.

6.1.3 Application-Specific SRAM Design

- Proposal of a prediction-based reduced bit-line switching activity (PB-RBSA) scheme and a bit-cell design to implement it.

- Proposal of hierarchical sensing and sense-amplifier gating techniques to complement PB-RBSA scheme.

- Implementation of above ideas on a test chip in 65nm CMOS technology. Up to 1.9× energy/access reduction is achieved over an 8T design.

6.2 Future Work

There are many interesting challenges that lie ahead for next generation video coding and application-specific SRAM design.

- *Application-Specific SRAM in Other Areas.* Application-specific SRAM has the potential to be applied to various areas as SRAMs are integral parts of almost all SoC designs. Features of input data and access patterns to the SRAM can change greatly from one application to the other. Taking these features into account at the design stage can provide a new dimension to address energy-efficiency and can provide significant savings. An example is using adaptive cache sizing in the design of a processor where special SRAMs can be employed to flush caches quickly between adaptations. Although these designs are mainly targeted for a specific application, the level of energy savings or performance improvements can be substantial.

- *Extension of Hardware Cost vs. Coding Efficiency Analysis to HEVC Encoder.* The analysis provided in Chapter 2 for motion estimation can be extended to cover the rest of the encoder. This can provide valuable insight on the relative complexity and hardware cost of various tools to identify key areas to do algorithm and architecture level optimizations. For example, Skip/Merge estimation and transform engines are highly complex in HEVC. Quantifying this complexity and decision of supported transform sizes and skip/merge modes can provide significant savings.

- *Integration of PB-RBSA SRAMs with CCE Motion Estimation for HEVC.* PB-RBSA SRAMs can be used in the hardware implementation of a CCE motion estimation for HEVC to maximize energy savings at algorithm, architecture and circuit level.

- *Motion Estimation Algorithms for Multiview and Scalable Video Coding.* The extensions of HEVC standard in the multiview and scalable video coding areas will introduce interesting challenges for motion estimation. Data bandwidth will

be an important issue especially for multiview coding. Hardware-efficient motion estimation algorithms can be very beneficial for hardware implementation of these.

- *3D Chip Stacking and Cache Architectures.* Motion estimation requires a large off-chip bandwidth to access reference picture frames. Stacking a video encoder with a DRAM chip can provide significant savings in terms of power consumption as well as a higher bandwidth. This can provide a new space for hardware efficient search algorithms if it is supported with new cache architectures to take advantage of the 3D implementation.

Appendix A

Test Sequences Used in HEVC Standardization

Standard set of sequences for HEVC standardization are given in Table A.1.

Name	Resolution	fps	Number of Frames
BQMall	832x480	60	600
BQSquare	416x240	60	600
BQTerrace	1920x1080	60	600
BasketballDrill	832x480	50	500
BasketballDrive	1920x1080	50	500
BasketballPass	416x240	50	500
BlowingBubbles	416x240	50	500
Cactus	1920x1080	50	500
Kimono1	1920x1080	24	240
NebutaFestival	2560x1600	60	300
ParkScene	1920x108	24	240
PartyScene	832x480	50	500
PeopleOnStreet	2560x1600	30	150
RaceHorsesD	416x240	30	300
RaceHorsesC	832x480	30	300
SteamLocomative	2560x1600	60	300
Traffic	2560x1600	30	150
vidyo1	1280x720	60	600
vidyo3	1280x720	60	600
vidyo4	1280x720	60	600

Table A.1: Standard set of sequences used in HEVC.

Bibliography

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.

[2] A. P. Chandrakasan, D. Daly, J. Kwong, and Y. K. Ramadass, "Next Generation Micro-power Systems," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2008, pp. 2–5.

[3] "iPhone Specifications." [Online]. Available: http://www.apple.com/iphone/specs.html.

[4] S. Borkar, "Obeying Moore's Law beyond 0.18 micron [microprocessor design]," in *IEEE International ASIC/SOC Conference*, Sep. 2000, pp. 26–31.

[5] "AnandTech." [Online]. Available: http://www.anandtech.com/show/2671.

[6] R. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, and T. Grutkowski, "A 32nm 3.1 billion transistor 12-wide-issue Itanium processor for mission-critical servers," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb. 2011, pp. 84 –86.

[7] "YouTube." [Online]. Available: http://www.youtube.com.

[8] "Cisco Virtual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016" [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

[9] J. Ostermann, P. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, Performance and Complexity," *IEEE Circuits and Systems Magazine*, vol. 4, pp. 7–28, 2004.

[10] G. Delagi, "Harnessing technology to advance the next-generation mobile user-experience," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb. 2010, pp. 18 –24.

[11] D. Markovic, B. Nikolic, and R. Brodersen, "Power and Area Minimization for Multidimensional Signal Processing," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 4, pp. 922 –934, Apr. 2007.

[12] "Joint Call for Proposals on Video Compression Technology," ITU-T SG16/Q6, 39th VCEG Meeting: Kyoto, 17-22 Jan. 2010, Doc. VCEG-AM91.

[13] J. S. Lim, *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1989.

[14] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. NTC81, New Orleans, LA,*, Nov. 1981, pp. C9.6.1–9.6.5.

[15] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, no. 4, pp. 438 –442, Aug. 1994.

[16] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *Communications, IEEE Transactions on*, vol. 29, no. 12, pp. 1799 – 1808, Dec. 1981.

[17] R. Srinivasan and K. Rao, "Predictive Coding Based on Efficient Motion Estimation," *Communications, IEEE Transactions on*, vol. 33, no. 8, pp. 888 – 896, Aug. 1985.

[18] S. Kappagantula and K. Rao, "Motion Compensated Interframe Image Prediction," *Communications, IEEE Transactions on*, vol. 33, no. 9, pp. 1011 – 1015, Sep. 1985.

[19] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 3, pp. 313 –317, Jun. 1996.

[20] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*, vol. 1, Sep. 1997, pp. 292 –296 vol.1.

[21] T.-H. Tsai and Y.-N. Pan, "A novel predict hexagon search algorithm for fast block motion estimation on H.264 video coding," in *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*, vol. 1, Dec. 2004, pp. 609 – 612 vol.1.

[22] M. Ghanbari, "The cross-search algorithm for motion estimation [image coding]," *Communications, IEEE Transactions on*, vol. 38, no. 7, pp. 950 –953, Jul. 1990.

[23] Y.-W. Huang, T.-C. Wang, B.-Y. Hsieh, and L.-G. Chen, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG-4 AVC/JVT/ITU-T H.264," in *Int. Symp. on Circuits and Systems (ISCAS) Dig. Tech. Papers*, vol. 2, May 2003, pp. II–796 – II–799.

[24] S. Y. Yap and J. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation," *Circuits and Systems II: Express Briefs, IEEE Tran. on*, vol. 51, no. 7, pp. 384 – 389, Jul. 2004.

[25] T.-H. Tsai and Y.-N. Pan, "High Efficiency Architecture Design of Real-Time QFHD for H.264/AVC Fast Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 11, pp. 1646 –1658, Nov. 2011.

[26] C.-C. L. et al., "PMRME: A Parallel Multi-Resolution Motion Estimation Algorithm and Architecture for HDTV Sized H.264 Video Coding," in *IEEE Int. Conf. on Acoustic, Speed, and Signal Processing (ICASSP) Dig. Tech. Papers*, vol. 2, Apr. 2007, pp. II–385 –II–388.

[27] Y.-K. Lin, D.-W. Li, C.-C. Lin, T.-Y. Kuo, S.-J. Wu, W.-C. Tai, W.-C. Chang, and T.-S. Chang, "A 242mW $10mm^2$ 1080p H.264/AVC High-Profile Encoder Chip," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 314–315.

[28] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity," *Circuits and Systems Magazine, IEEE*, vol. 4, no. 1, pp. 7 – 28, quarter 2004.

[29] Y.-W. Huang, T.-C. Chen, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, C.-S. Chen, C.-F. Shen, S.-Y. Ma, T.-C. Wang, B.-Y. Hsieh, H.-C. Fang, and L.-G. Chen, "A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2005, pp. 128–129.

[30] C.-P. Lin, P.-C. Tseng, Y.-T. Chiu, S.-S. Lin, C.-C. Cheng, H.-C. Fang, W.-M. Chao, and L.-G. Chen, "A 5mW MPEG4 SP encoder with 2D bandwidth-sharing motion estimation for mobile applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006, pp. 1626–1627.

[31] 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. Quo, "A 7mW-to-183mW Dynamic Quality-Scalable H.264 Video Encoder Chip," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 280–281.

[32] T. Burd and R. Brodersen, "Design issues for Dynamic Voltage Scaling," in *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, 2000, pp. 9 – 14.

[33] V. Gutnik and A. Chandrakasan, "Embedded power supply for low-power DSP," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 425 –435, Dec. 1997.

[34] B. H. Calhoun and A. Chandrakasan, "Characterizing and Modeling Minimum Energy Operation for Subthreshold Circuits," in *Int. Symp. on Low-Power Elec. and Design (ISLPED) Dig. Tech. Papers*, 2004, pp. 90–95.

[35] A. Wang, A. Chandrakasan, and S. Kosonocky, "Optimal Supply and Threshold Scaling for Sub-threshold CMOS Circuits," in *IEEE Computer Society Annual Symposium on VLSI*, Apr. 2002, pp. 7–11.

[36] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[37] A. Wang and A. Chandrakasan, "A 180mV FFT Processor Using Sub-threshold Circuit Techniques," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2004, pp. 292–293.

[38] J. Kwong, Y. Ramadass, N. Verma, and A. Chandrakasan, "A 65 nm Sub-V_t Microcontroller With Integrated SRAM and Switched Capacitor DC-DC Converter," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 115 –126, Jan. 2009.

[39] B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin, "A 2.60pJ/Inst Subthreshold Sensor Processor for Optimal Energy Efficiency," in *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*, 2006, pp. 154 –155.

[40] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw, "Performance and Variability Optimization Strategies in a Sub-200mV, 3.5pJ/inst, 11nW Subthreshold Processor," in *VLSI Circuits, 2007 IEEE Symposium on*, Jun. 2007, pp. 152 –153.

[41] V. Sze, R. Blázquez, M. Bhardwaj, and A. Chandrakasan, "An Energy Efficient Sub-Threshold Baseband Processor Architecture For Pulsed Ultra-Wideband Communications," in *IEEE Int. Conf. on Acoustic, Speed, and Signal Processing (ICASSP) Dig. Tech. Papers*, May 2006, pp. 908–911.

[42] G. Gammie, N. Ickes, M. Sinangil, R. Rithe, J. Gu, A. Wang, H. Mair, S. Datla, B. Rong, S. Honnavara-Prasad, L. Ho, G. Baldwin, D. Buss, A. Chandrakasan, and U. Ko, "A 28nm 0.6V Low-Power DSP for Mobile Applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2011, pp. 132–133.

[43] N. Ickes, Y. Sinangil, F. Pappalardo, E. Guidetti, and A. Chandrakasan, "A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip," in *ESSCIRC (ESSCIRC), 2011 Proceedings of the*, Sept. 2011, pp. 159 –162.

[44] G. Gammie, A. Wang, M. Chau, S. Gururajarao, R. Pitts, F. Jumel, S. Engel, P. Royannez, R. Lagerquist, H. Mair, J. Vaccani, G. Baldwin, K. Heragu, R. Mandal, M. Clinton, D. Arden, and U. Ko, "A 45 nm 3.5G Baseband-and-Multimedia Applications Processor using Adaptive Body-Bias and Ultra-Low-Power Techniques," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 258–259.

[45] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.

[46] M. Sinangil, N. Verma, and A. Chandrakasan, "A Reconfigurable 8T Ultra-Dynamic Voltage Scalable U-DVS SRAM in 65 nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 11, pp. 3163 –3173, Nov. 2009.

[47] J. Burr, "Cryogenic Ultra Low Power CMOS," in *Int. Symp. on Low-Power Elec. and Design (ISLPED) Dig. Tech. Papers*, 1995, pp. 82–83.

[48] D. F. Finchelstein, V. Sze, , M. E. Sinangil, Y. Koken, and A. P. Chandrakasan, "A low-power 0.7V H.264 720 video decoder," in *IEEE Asian Solid State Circuits Conference*, Nov. 2008, pp. 173–176.

[49] C.-D. Chien, C.-C. Lin, Y.-H. Shih, H.-C. Chen, C.-J. Huang, C.-Y. Yu, C.-L. Chen, C.-H. Cheng, and J.-I. Guo, "A 252kgate/71mW multi-standard multi-channel video decoder for high definition video applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 282–283.

[50] T.-M. Liu, T.-A. Lin, S.-Z. Wang, W.-P. Lee, K.-C. Hou, J.-Y. Yang, and C.-Y. Lee, "A 125μW, fully scalable MPEG-2 and H.264/AVC video decoder for mobile applications," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 161–169, Jan. 2007.

[51] D. Boning and S. Nassif, *Models of Process Variations in Device and Interconnect.* IEEE Press, 2001, pp. 98–115.

[52] "BSIM4v4.7 Mosfet Model User's Manual." [Online]. Available: http://www-device.eecs.berkeley.edu/bsim/files/bsim4/bsim470/bsim470_manual.pdf.

[53] E. Seevinck, F. List, and J. Lohstroh, "Static Noise Margin Analysis of MOS SRAM Cells," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 5, pp. 748–754, Oct. 1987.

[54] L. Chang and et al., "Stable SRAM Cell Design for the 32nm Node and Beyond," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2005, pp. 128–129.

[55] I. J. Chang, D. Mohapatra, and K. Roy, "A Priority-Based 6T/8T Hybrid SRAM Architecture for Aggressive Voltage Scaling in Video Applications," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 2, pp. 101 –112, Feb. 2011.

[56] M. Khare, S. Ku, R. Donaton, S. Greco, C. Brodsky, X. Chen, A. Chou, R. DellaGuardia, S. Deshpande, B. Doris, S. Fung, A. Gabor, M. Gribelyuk, S. Holmes, F. Jamin, W. Lai, W. Lee, Y. Li, P. McFarland, R. Mo, S. Mittl, S. Narasimha, D. Nielsen, R. Purtell, W. Rausch, S. Sankaran, J. Snare, L. Tsou, A. Vayshenker, T. Wagner, D. Wehella-Gamage, E. Wu, S. Wu, W. Yan, E. Barth, R. Ferguson, P. Gilbert, D. Schepis, A. Sekiguchi, R. Goldblatt, J. Welser, K. Muller,

and P. Agnello, "A high performance 90nm SOI technology with $0.992\mu m^2$ 6T-SRAM cell," in *Electron Devices Meeting, 2002. IEDM '02. International*, Dec. 2002, pp. 407 –410.

[57] A. Kawasumi and et al., "A Single-Power-Supply 0.7V 1GHz 45nm SRAM with An Asymmetrical Unit-β-ratio Memory Cell," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 382–383.

[58] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson, "A Sub-200mV 6T SRAM in $0.13\mu m$ CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 332–333.

[59] K. Takeda, Y. Hagihara, Y. Aimoto, M. Nomura, Y. Nakazawa, T. Ishii, and H. Kobatake, "A Read-Static-Noise-Margin-Free SRAM Cell for Low-Vdd and High-Speed Applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2005, pp. 478–479.

[60] L. Chang, Y. Nakamura, R. K. Montoye, J. Sawada, A. K. Martin, K. Kinoshita, F. Gebara, K. Agarwal, D. Acharyya, W. Haensch, K. Hosokawa, and D. Jamsek, "A 5.3GHz 8T-SRAM with Operation Down to 0.41V in 65nm CMOS," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2007, pp. 252–253.

[61] S. Ishikura, M. Kurumada, T. Terano, Y. Yamagami, N. Kotani, K. Satomi, K. Nii, M. Yabuuchi, Y. Tsukamoto, S. Ohbayashi, T. Oashi, H. Makino, H. Shinohara, and H. Akamatsu, "A 45nm 2port 8T-SRAM using hierarchical replica bitline technique with immunity from simultaneous R/W access issues," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2007, pp. 254–255.

[62] R. Joshi and et al., "6.6+ GHz Low Vmin, read and half select disturb-free 1.2Mb SRAM," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2007, pp. 250–251.

[63] Y. Morita and et al., "An Area-Conscious Low-Voltage-Oriented 8T-SRAM Design under DVS Environment," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2007, pp. 256–257.

[64] B. Calhoun and A. Chandrakasan, "A 256-kbit Sub-threshold SRAM in 65nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006, pp. 628–629.

[65] T.-H. Kim, J. Liu, J. Keane, and C. H. Kim, "A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtual Ground Replica Scheme," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 330–331.

[66] I. J. Chang, J. Kim, S. P. Park, and K. Roy, "A 32kb 10T Subthreshold SRAM Array with Bit-interleaving and Differential Read Scheme in 90nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 388–389.

[67] K. Itoh, A. Fridi, A. Bellaouar, and M. Elmasry, "A Deep Sub-V, Single Power-Supply SRAM Cell with Multi-V_T, Boosted Storage Node and Dynamic Load," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 1996, pp. 132–133.

[68] K. Kanda, T. Miyazaki, M. K. Sik, H. Kawaguchi, and T. Sakurai, "Two Orders of Magnitude Leakage Power Reduction of Low Voltage SRAM's by Row-by-Row Dynamic V_{DD} Control (RRDV) Scheme," in *IEEE International ASIC/SOC Conference*, Sep. 2002, pp. 381–385.

[69] M. Yamaoka, K. Osada, and K. Ishibashi, "0.4-V Logic Library Friendly SRAM Array Using Rectangular-Diffusion Cell and Delta-Boosted-Array-Voltage Scheme," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, 2002, pp. 170–173.

[70] N. Verma and A. Chandrakasan, "A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006, pp. 328–329.

[71] K. Zhang, U. Bhattacharya, Z. Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Y. Wang, B. Zheng, and M. Bohr, "A 3-GHz 70Mb SRAM in 65nm CMOS Technology with Integrated Column-Based Dynamic Power Supply," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2005, pp. 474–475.

[72] M. Yamaoka, N. Maeda, Y. Shinozaki, Y. Shimazaki, K. Nii, S. Shimada, K. Yanagisawa, and T. Kawahara, "Low-Power Embedded SRAM Modules with Expanded Margins for Writing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2005, pp. 480–481.

[73] Y. Fujimura, O. Hirabayashi, T. Sasaki, A. Suzuki, A. Kawasumi, Y. Takeyama, K. Kushida, G. Fukano, A. Katayama, Y. Niki, and T. Yabe, "A configurable SRAM with constant-negative-level write buffer for low-voltage operation with $0.149\mu m^2$ cell in 32nm high-k metal-gate CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 348–349.

[74] K. Takeda, H. Ikeda, Y. Hagihara, M. Nomura, and H. Kobatake, "Redefinition of Write Margin for Next-Generation SRAM and Write-Margin Monitoring Circuit," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006.

[75] Y. Ye, M. Khellah, D. Somasekhar, and V. De, "Evaluation of Differential vs. Single-Ended Sensing and Asymmetric Cells in 90nm Logic Technology for On-Chip Caches," in *Int. Symp. on Circuits and Systems (ISCAS) Dig. Tech. Papers*, 2006, pp. 963–966.

[76] N. Verma and A. Chandrakasan, "A High-Density 45nm SRAM Using Small-Signal Non-Strobed Regenerative Sensing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 380–381.

[77] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto, "A Current-Controlled Latch Sense Amplifier and a Static Power-Saving Input Buffer for Low-Power Architecture," *IEEE J. Solid-State Circuits*, vol. 28, no. 4, pp. 523–527, Apr. 1993.

[78] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel, "Yield and Speed Optimization of a Latch-Type Voltage Sense Amplifier," *IEEE J. Solid-State Circuits*, vol. 39, no. 7, pp. 1148–1158, Jul. 2004.

[79] R. Singh and N. Bhat, "An Offset Compensation Technique for Latch Type Sense Amplifiers in High-speed Low-power SRAMs," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol. 12, no. 6, pp. 652–657, Jun. 2004.

[80] M. E. Sinangil, V. Sze, M. Zhou, and A. P. Chandrakasan, "Hardware-Aware Motion Estimation Search Algorithm Development for High-Efficiency Video Coding (HEVC) Standard," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept. 2012, [Accepted].

[81] ——, "Memory Cost vs. Coding Efficiency Trade-Offs for HEVC Motion Estimation Engine," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept. 2012, [Accepted].

[82] M. Zhou, M. Sinangil, V. Sze, S. Park, J. Park, and B. Jeon, "JCTVC-F088: CE9: Simplified AMVP Design," in *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11*, Jul. 2011.

[83] M. Sinangil, H. Mair, and A. Chandrakasan, "A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb. 2011, pp. 260 –262.

[84] M. Sinangil, N. Verma, and A. Chandrakasan, "A 45nm 0.5V 8T column-interleaved SRAM with on-chip reference selection loop for sense-amplifier," in *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, Nov. 2009, pp. 225 –228.

[85] T. Lu, X. Lu, Q. Xu, Y. Zheng, J. Sole, and P. Yin, "A video coding analyzer for next-generation compression standards," in *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, Jan. 2011, pp. 707 –708.

[86] "JCT-VC Reference Software HM-3.0," ISO/IEO MPEG and ITU-T.

[87] "Taiwan Semiconductor Manufacturing Company Limited." [Online]. Available: http://www.tsmc.com/english/dedicatedfoundry/technology/65nm.htm.

[88] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct. 1989.

[89] S. O. Toh, Z. Guo, T.-J. Liu, and B. Nikolic, "Characterization of Dynamic SRAM Stability in 45 nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 11, pp. 2702 –2712, Nov. 2011.

[90] M. Yamaoka, K. Osada, and T. Kawahara, "A cell-activation-time controlled SRAM for low-voltage operation in DVFS SoCs using dynamic stability analysis," in *IEEE European Solid-State Circuits Conf. (ESSCIRC) Dig. Tech. Papers*, Sep. 2008, pp. 286–289.

[91] K. Kushida and et al., "A 0.7V single-supply SRAM with $0.495\mu m^2$ cell in 65nm technology utilizing self-write-back sense amplifier and cascaded bit line scheme," in *Symp. on VLSI Circuits (VLSI) Dig. Tech. Papers*, Jun. 2008, pp. 46–47.

[92] S. Cosemans, W. Dehaene, and F. Catthoor, "A 3.6pJ/access 480MHz, 128Kbit on-chip SRAM with 850MHz boost mode in 90nm CMOS with tunable sense amplifiers to cope with variability," in *IEEE European Solid-State Circuits Conf. (ESSCIRC) Dig. Tech. Papers*, Sep. 2008, pp. 278–281.

[93] R. Rithe, C.-C. Cheng, and A. Chandrakasan, "Quad Full-HD transform engine for dual-standard low-power video coding," in *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian*, Nov. 2011, pp. 401 –404.

[94] N. Verma, "Ultra-Low-Power SRAM Design In High Variability Advanced CMOS," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2009.

[95] J. Kwong and A. P. Chandrakasan, "An Energy-Efficient Biomedical Signal Processing Platform," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 7, pp. 1742 –1753, Jul. 2011.