INTELLIGENT EFFICIENT ON-CHIP MEMORY FOR MOBILE VIDEO STREAMING

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Dongliang Chen

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Electrical and Computer Engineering

June 2017

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Intelligent Efficient On-Chip Memory for Mobile Video Streaming

By

Dongliang Chen

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Na Gong
Chair

Jinhui Wang

Zhibin Lin

Approved:

06/23/2017
Date

Alan Kallmeyer
Department Chair

ABSTRACT

The growing popularity of powerful mobile devices such as smart phones and tablet devices has resulted in the exponential growth of demand for video applications. User experience and battery life are both crucial topics in the advancement of these devices. However, due to the high power consumption of mobile video decoders, especially the on-chip memories, short battery life represents one of the biggest contributors to user dissatisfaction.

Various mobile embedded memory techniques have been investigated to reduce power consumption and prolong battery life. Unfortunately, the existing hardware-level research suffers from high implementation complexity and large overhead. In this thesis, we focus on smart power-efficient memory design, considering both user's viewing experience and low power memory design. Our results shows up to 57.2% power saving is achieved in VCAS and 43.7% power saving is achieved in D-DASH with negligible area cost.

ACKNOWLEDGEMENTS

I am deeply grateful to Dr. Na Gong, my advisor for providing me the opportunity to work under her supervision and for her invaluable support and guidance. I would also like to thank Dr. Jinhui Wang and Dr. Zhibin Lin for having accepted to be my supervisory committee members. I have received the suggestions and assistance provided by them numerous times.

I would specially like to thank my colleagues and fellow students Xiaowei Chen, Seyed Alireza Pourbakhsh, Xin Wang, Peng Gao, Yifu Gong, Ruisi Ge, and Jonathon Edstrom for their contributions and support in the thesis.

This work was supported by the National Science Foundation, under Grant CCF-1514780.

DEDICATION

Dedicated to:

My father and mother, Lijun Chen and Yanqiu Zhao,

My sister, Yingnan Chen.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

DEDICATION v

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ABBREVIATIONS xii

1. INTRODUCTION 1

1.1. Trends in Designing Power Efficient Mobile On-chip Memories 1

1.2. Related Work 1

1.2.1. Related Work on Low-power General-purpose SRAM 1

1.2.2. Related Work on Video-specific SRAM Design 2

1.3. Contribution in the Thesis 3

2. PROPOSED VIEWING LUMINANCE CONTEXT AWARE SRAM 4

2.1. Context Influence on Mobile User Experience 4

2.1.1. H.264 Mobile Video Decoding 4

2.1.2. Impact of Viewing Context Influence on Mobile Videos 5

2.2. VCAS Technique 7

2.2.1. Enabling VCAS by Introducing Hardware Noise 7

2.2.2. VCAS Design Using Bit-truncation Technique 7

2.2.3. VCAS Circuit Implementation 9

2.3. Experimental Results 11

2.3.1. Experimental Methodology 11

2.3.2. Results 11

2.4. Conclusion of VCAS 17

3. PROPOSED DATA-DRIVEN POWER EFFICIENT ADAPTABLE SRAM HARDWARE 18
3.1. Mobile Video Data Pattern Analysis 18
3.1.1. Mobile Video Data Characteristics 18
3.1.2. Data Mining Assisted Video Analysis 19
3.1.3. Video Quality Metrics 22
3.2. D-DASH Technique 23
3.2.1. D-DASH-I 23
3.2.2. D-DASH-II 25
3.2.3. D-DASH-III 26
3.3. Simulation Results 27
3.3.1. Performance 28
3.3.2. Layout 28
3.3.3. Output Quality 30
3.3.4. Power Savings 30
3.3.5. Comparison with Prior Work 32
3.4. Conclusion of D-DASH 32
4. CONCLUSION 34
4.1. Summary 34
4.2. Suggestions for Future Work 34
4.2.1. Techniques Extension for Other Memories in Decoder 34
4.2.2. Hardware 34
REFERENCES 35
APPENDIX A. VERILOG CODE USED TO MODEL MEMORY FAILURE 41
APPENDIX B. VERILOG CODE USED FOR CALCULATING MSE 42
APPENDIX C. VERILOG CODE USED FOR BIT SWITCHING 49

APPENDIX D. VERILOG CODE USED FOR D-DASH POWER 56

APPENDIX E. PUBLICATIONS 64

LIST OF TABLES

Table Page

1. Video data analysis. 9

2. Switching probability of each bit. 16

3. Power savings of VCAS in different context. 16

4. Comparison with existing study. 17

5. PSNR and SSIM calculations. 26

6. SRAM Specifications 28

7. RBL power consumption for different types of SRAM. 31

8. Probability of each bit being 0 or 1 (F(i)). 31

9. Normal videos power savings of different schemes. 32

10. 4K HEVC videos power savings of different schemes. 32

11. YouTube-8M videos power savings of different schemes. 32

12. Comparison with prior art on low-power mobile video SRAM. 33

LIST OF FIGURES

Figure Page

1. Thesis focus. 3

2. H.264 video decoder and embedded memory. 5

3. Video quality distortion under three different contexts. (dark, overcast, and sunlight). (a). xxxx0000 in dark, (b). original in dark, (c). xxxx0000 in overcast, (d). original in overcast, (e). xxxx0000 in sunlight, (f). original in sunlight. 6

4. Video output with bit truncation and low Vdd techniques. (a) original video in dark, (b) bit truncation in dark, (c) reduce Vdd in dark, (d). original in sunlight, (e). bit truncation in sunlight, (f). reduce Vdd in sunlight. 8

5. Modelling the video in sunlight by adjusting the brightness and contrast. (a) modeling video in sunlight, (b) real video in sunlight. 9

6. SRAM circuit with VCAS control circuit. 10

7. Timing diagram of VCAS in sunlight (BIT2 = 1 and BIT1 = 0). 12

8. Proposed layout (in μm). (a) VCAS SRAM, (b) VCAS control circuit. 13

9. (a)-(c): video akiyo in different contexts (Proposed), (d)-(f): video akiyo in different contexts (Conventional). 14

10. (a)-(c): video foreman in different contexts (Proposed), (d)-(f): video foreman in different contexts (Conventional). 14

11. (a)-(c): video container in different contexts (Proposed), (d)-(f): video container in different contexts (Conventional). 15

12. Data mining assisted video data analysis and discovered rules. 21

13. Rules diagram for normal video. 21

14. Rules diagram for 4K HEVC video. 22

15. Data-aware D-DASH bitcells. (a) type-1 bitcell without discharging when reading 1, (b) type-0 bitcell without discharging when reading 0, (c) data pattern based wordline. 25

16. D-DASH with real-time adjustment between three schemes. 26

17. sign_irene output with bit-truncation technique. 27

18. SRAM write and read speed waveform. 29

19. Layout of D-DASH. 29

20. Video output 30

LIST OF ABBREVIATIONS

IoT.....Internet of Things

D-DASH .....Three Dimensional

VCAS.....Viewing Context Aware SRAM

CMOS .....Complementary Metal-Oxide-Semiconductor

QoE .....Quality of Experiences

IQ .....Inverse Quantization

IT.....Inverse Transformation

IC.....Integrated Circuit

MOSFET.....Metal-Oxide-Semiconductor Field-Effect Transistor

NAND.....Negative AND

NCSU.....North Carolina State University

NMOS .....n-type Metal-Oxide-Semiconductor Field-Effect Transistor

NOR .....Negative OR

PMOS.....p-type Metal-Oxide-Semiconductor Field-Effect Transistor

ECC.....Error Correcting Code

SOC.....System on Chip

SRAM .....Static Random Access Memory

VLSI.....Very Large Scale Integration

1. INTRODUCTION

1.1. Trends in Designing Power Efficient Mobile On-chip Memories

The rapid development of portable devices, it is becoming more and more popular for people to watch videos on mobile phones. According to market research, by 2020, the amount of data in mobile devices that is created, replicated, and consumed, will be as large as 40ZB (Zettabyte, or 1021B) [1]; and more than half of the data traffic will be video data [33]. Recent studies show that 53% of users often watch videos on mobile devices over 30 minutes [34], 69% of them watch videos at work, 80% of them while they perform outdoor activities, and 96% of them at home [36, 37]. As people enjoy watching videos anytime and anywhere, the experience and battery life are two major concerns [2]. Video processing has become the most important energy-intensive application used in mobile devices [3]. In particular, the major signal processing units in video decoders, such as motion estimation and compensation, require frequent embedded memory access. Embedded SRAM contributes to over 30% of the system power consumption of a mobile device [4, 5], and this situation is only expected to grow with the emerging high quality mobile video applications.

1.2. Related Work

Significant amount of researches that target low-power mobile video memory have been reported in the literatures. The techniques mainly fall into two aspects: general-purpose SRAM design and video-specific SRAM design. In this section, some of the existing work related to the proposed technique are briefly reviewed.

1.2.1. Related Work on Low-power General-purpose SRAM

Many solutions have been developed to lower the power consumption of memory utilizing assist schemes such as adjustment of cell voltage [37], boosted word-line voltage [38],

dual-rail supply schemes [39], and read-modify-write or write-back schemes [40]. Most existing solutions also adopt more than 6T to achieve low power operation, such as asymmetric 7T cell [41], column-decoupled 8T cells [30], read-disturb-free 9T [42] and 10T SRAM cells [28], and bit-interleaving 12T cells [43].

1.2.2. Related Work on Video-specific SRAM Design

Various low power SRAM designs have been developed for mobile video applications. In [28], due to FRAM provides higher reliability but lower energy efficiency compared to SRAM, a hybrid FRAM-SRAM is proposed to minimize the energy consumption by dynamically mapping programs' sections to FRAM and SRAM. In [10] and [23], two hybrid SRAM structures with 6T/8T and 8T/10T are developed to optimize the power efficiency for mobile video streaming because SRAM with more transistors provides higher reliability in low voltage. In that way, higher order bit's memory failure rate is reduced. Similar in [15], a heterogeneous sizing scheme was presented to reduce the failure probability of conventional 6T bitcells. In [6], a two-port SRAM with majority logic and data reordering is presented to minimize power consumption. However, all of those existing techniques suffer from large silicon area overhead. Also, the power-quality tradeoff is set during design time, which can no longer automatically guarantee maximum power efficiency for different video applications. Recently, Frustaci et al. [11] presents a voltage-scaled SRAM that can dynamically manage the trade-off between power and video quality, but the utilized write assist technique and Error Correcting Code (ECC) encoder and decoder circuits result in large penalties in computation complexity and silicon area.

However, the improvements in the memory power efficiency of those general-purpose design techniques are often achieved with significant design complexity, increased silicon area,

and power penalty for voltage regulator or boosting circuits needed to solve the memory failure issue.

1.3. Contribution in the Thesis

All of those existing techniques focus on hardware optimization, neither do they consider QoE of the users and therefore significant opportunities to minimize the power consumption are wasted, nor do they consider the data stored in the memory. In this thesis, a Viewing luminance Context Aware SRAM (VCAS) is proposed to maximize the power efficiency, while satisfying the users' expectation by introducing context-awareness into hardware design process. Also, a low-cost Data-Driven power efficient Adaptable SRAM Hardware (D-DASH) design with dynamic power-quality tradeoff for mobile video applications is proposed.

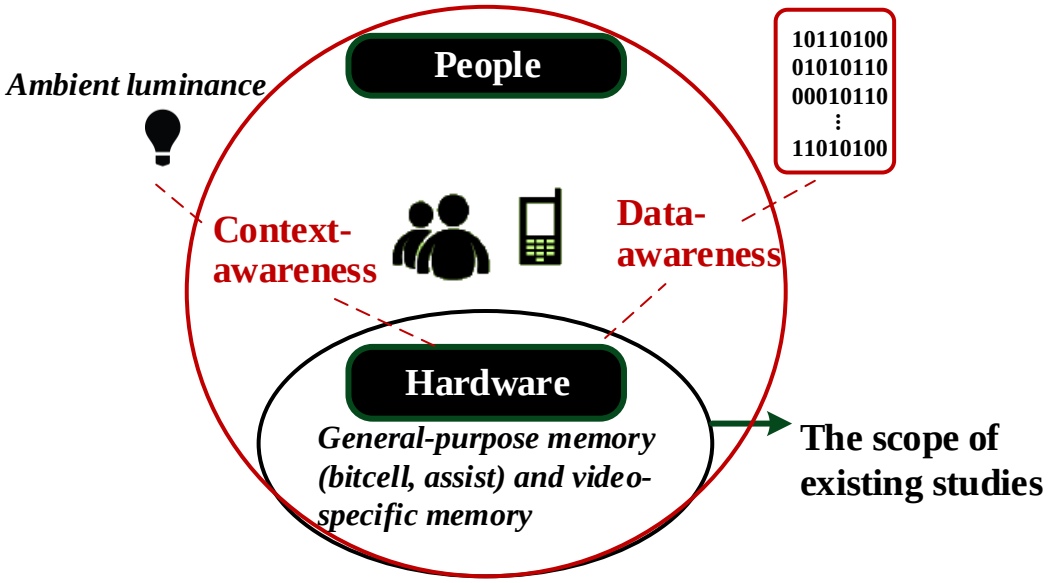


Figure 1. Thesis focus.

2. PROPOSED VIEWING LUMINANCE CONTEXT AWARE SRAM

In this thesis, VCAS is firstly introduced to maximize the memory power efficiency, while considering user's viewing context. Such interaction of both viewing context and hardware views enables a new dimension of power savings.

2.1. Context Influence on Mobile User Experience

2.1.1. H.264 Mobile Video Decoding

The proposed VCAS is applied to H.264, which is one of the most popular video codec standards in mobile multimedia communications. Note that, the proposed VCAS has potential to enable more power savings for emerging video format such as H.265 / High Efficiency Video Codec (HEVC) which has 2x-3x higher memory demand compared to that of the H.264/AVC [46].

Fig. 2 shows the general block diagram of the H.264 decoder. After decoding, inverse quantization (IQ) and inverse transformation (IT), the residual error of frames can be reconstructed based on the compressed video streams. The motion compensator uses the previous reconstructed frames stored in the reference frame buffer and the transmitted motion vectors to construct new frames.

In the video decoding process, the reference frame memory is accessed frequently. After each frame is decoded, a new frame will be written into the memory and then be read out as the next frame's reference. Fig. 2 also shows the typical frame data stored in embedded memory using 176 x 144 resolution YUV 4:2:0 video as an example. As shown, each pixel has 8-bit Luma data and 4-bit Chroma data. All frames from 1 to 300 are stored in embedded memory (not at the same time), and for each frame, the memory stores all the Luma (Y) data from the first line of pixels to the last line and followed by the Chroma data – Cb (U) and Cr (V). Accordingly, an

entire frame requires a 176 x144 (25344) x 8 bits SRAM to store Luma data and 12672 x 8 bits SRAM to store Chroma data. The frequent writing and reading operations in memory during decoding process consumes significant power consumption. Therefore, low power mobile video memory design is extremely important to enable power-efficient mobile video streaming systems.

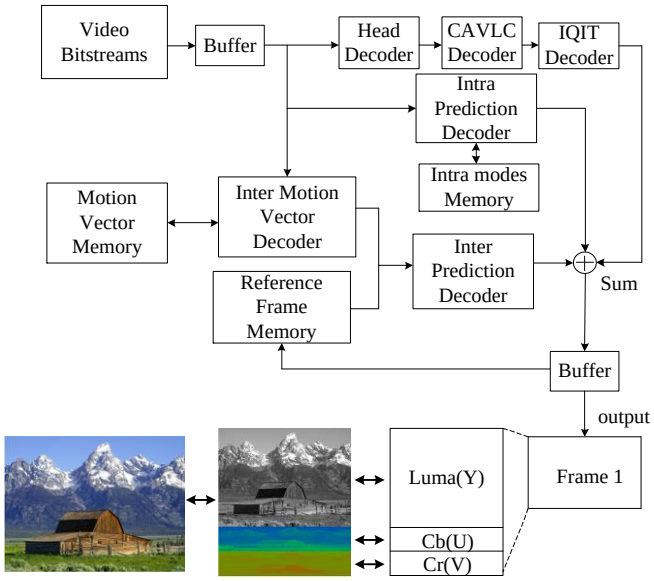


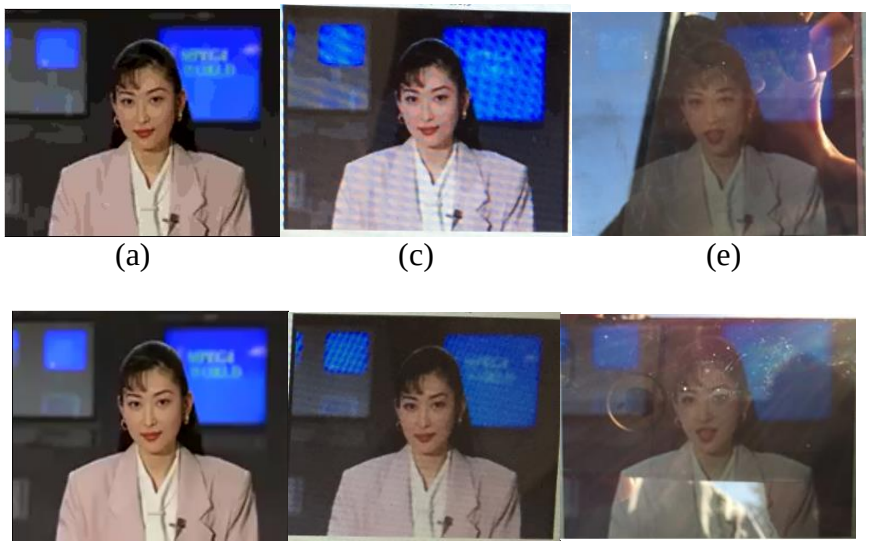
Figure 2. H.264 video decoder and embedded memory.

2.1.2. Impact of Viewing Context Influence on Mobile Videos

Psychophysical researchers have conducted research on impact of viewing surroundings on human experience [16-19]. It shows that the viewing conditions that influence mobile video watching experience falls into three major aspects: the display size and viewing distance between the user and mobile, the ambient luminance, and the viewer's body movements. In particular, the ambient luminance significantly influences the viewing experience. People experience videos quality with huge loss on mobile phone under the bright outdoor viewing condition. This is because, human eyes can function in both dark and bright environments, but their contrast resolution is limited. Thus, when eyes adapt to the brighter environment, they will lose contrast

resolution. Based on this, several recent works have presented low-power techniques from an algorithm perspective, such as adaptive transcoding techniques [20-22]. In this paper, for the first time, context awareness is introduced to hardware design process to reduce the power consumption.

We further conduct experiment on impact of ambient luminance on video output. Fig. 3 shows the results in three different contexts (dark, overcast, and sunlight). In the figure, x means using original video data, 0 means using 0 instead of original video data with bit truncation technique, which will be discussed in Section IV. Figs. 3 (a), (c), and (e) show the same image with four Least Significant Bit (LSB) of Luma data truncated to 0s and the PSNR is 31.67; Figs. 3 (b), (d), and (f) are the same original decoded image and the PSNR is 38.83. As shown, under those high luminance conditions users may perceive video quality to be unchanged, even though the actual quality has been degraded. Accordingly, we may utilize such noise-tolerance ability of human eyes to achieve a power-efficient mobile video storage system, thereby prolonging the battery life of mobile devices while maintaining the user experience.



(a) (c) (e)

(b) (d) (f)

Figure 3. Video quality distortion under three different contexts. (dark, overcast, and sunlight). (a). xxxx0000 in dark, (b). original in dark, (c). xxxx0000 in overcast, (d). original in overcast, (e). xxxx0000 in sunlight, (f). original in sunlight.

2.2. VCAS Technique

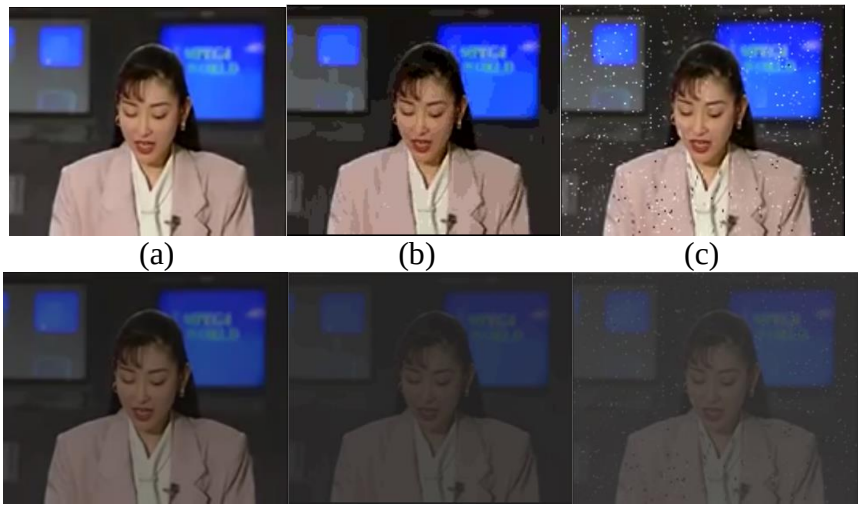
In this section, VCAS is presented, based on luminance sensors which are typically available in modern mobile phones to detect the ambient luminance. Here, the luminance definition in [19-22] is used: 0-1000 is dark, 1000-10000 is overcast, and 10000+ is sunlight.

2.2.1. Enabling VCAS by Introducing Hardware Noise

In embedded SRAM, there are two widely applied techniques to achieve power savings based on trade-off between power and quality. One approach is to reduce the V_{dd} (voltage scaling), which reduces the power consumption at low voltage. The other one is bit truncation, which will skip the storage of LSBs due to their low contribution to the output quality [3]. Fig. 4 compares the video quality with those two techniques. As observed, the bit truncation causes some blur in the video with the *PSNR* value as 31.67. Alternately, low V_{dd} technique causes many white or black noise points in the video with the *PSNR* value as 32.61. The two approaches have similar *PSNR* values, which indicates similar video quality, but the video quality loss using the bit truncation approach is much less obvious than that of the reducing V_{dd}. As expected, in the sunlight, the bit truncation image is even better than that of the low V_{dd}. In our implementation, bit truncation technique is used to tradeoff between power savings and quality in different viewing contexts.

2.2.2. VCAS Design Using Bit-truncation Technique

As mentioned before, the bit truncation technique will cause memory failure in high noise-tolerance viewing contexts. Specifically, the LSBs of the 8-bit data stored in SRAM are disabled. But, LSBs have less influence on the video quality, while the power consumption of reading and writing LSBs is the same as the MSBs. Therefore, determination of number of



(a) (b) (c)

(d) (e) (f)

Figure 4. Video output with bit truncation and low Vdd techniques. (a) original video in dark, (b) bit truncation in dark, (c) reduce Vdd in dark, (d). original in sunlight, (e). bit truncation in sunlight, (f). reduce Vdd in sunlight.

truncation bits in different viewing contexts is critical to implement VCAS. A luminance and contrast based model is developed. Based on the RGB Luminance expressed in (1), Luminance Contrast is defined by (2)

$$L_{RGB} = 0.3R + 0.59G + 0.11B \tag{1}$$

$$Contrast = \frac{Diff_{Luminance}}{Ave_{Luminance}} \tag{2}$$

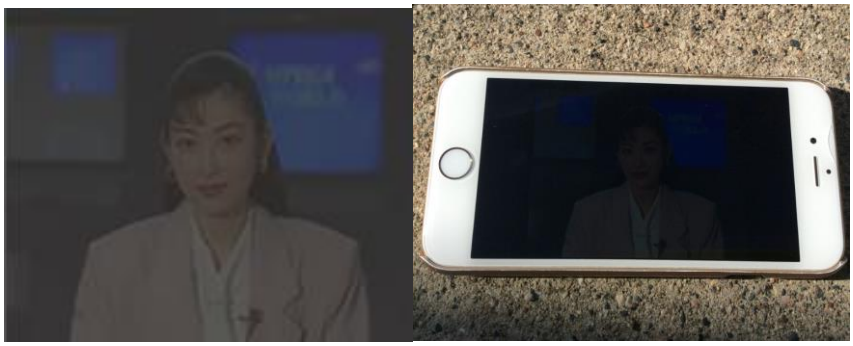
where LRGB is RGB luminance; $Ave_{Luminance}$ is the average luminance in a frame and $Diff_{Lumiance}$ is the sum of luminance difference between each pixel and the average luminance.

Fig. 5 compares the video frame using the developed model with the actual frame in the sunlight, overcast, and dark, which demonstrates the accuracy of the model. The results are listed in Table 1. It shows that, for both Akiyo and Foreman, as compared to the image in the dark, for 0, 3, 4, and 5 bit truncations, the average luminance decreases by approximately 20 in overcast and decreases by about 30 in sunlight. Table 1 also shows that the Luminance Contrast significantly decreases in sunlight and therefore human eyes can hardly find some video quality

distortion. However, as listed in Table 1, while with more bits truncated, the contrast increases in all of three viewing contexts, which means that the distortion is easier to be noticed.

Table 1. Video data analysis.

Video		***Akiyo***			***Foreman***		
Contexts	**# Bit truncated**	$Ave_{luminance}$	$Diff_{luminance}$	***contrast***	$Ave_{luminance}$	$Diff_{luminance}$	***contrast***
Dark	0	90	1207849	13420	170	1428896	8405
	3	87	1229762	14135	166	1432373	8628
	4	82	1249489	15237	161	1418935	8813
	5	73	1269937	17396	150	1462527	9750
Overcast	0	70	837859	11969	150	1003382	6689
	3	67	861422	12857	146	1005861	6889
	4	62	876261	14133	141	997266	7072
	5	53	888169	16757	130	1027670	7905
Sun	0	60	239310	3988	140	286762	2048
	3	57	243927	4279	136	288654	2122
	4	52	247075	4751	131	284979	2175
	5	43	251374	5845	120	292400	2436



(a) (b)

Figure 5. Modelling the video in sunlight by adjusting the brightness and contrast. (a) modeling video in sunlight, (b) real video in sunlight.

2.2.3. VCAS Circuit Implementation

In VCAS circuit design, SRAMs are used to store Luma and Chroma data, separately. Here, VCAS is taken as design case for storing Luma data. The ambient luminance level information can be captured from the light sensor embedded in mobile phones, and sent to the VCAS for adaption.

Fig. 6 shows the architecture of the proposed VCAS. The total array size is 32 kbit and there are four blocks with 256 wordlines × 32 bits. A hierarchical readout bitline scheme (local RBL and global RBL) is applied to reduce the access time. The VCAS control unit consists of a Write Enable (WE) control circuit and Read Enable (RE) control circuit. The detailed control circuits for WE and RE are shown in Fig. 6. Using the control unit, the memory disables 0, 3, or 4 LSBs' WE signal and RE signal according to the detected context information bits from the mobile sensor. For WE (or RE) control circuit, two-bit control signal (BIT2 and BIT1) will enable bit-truncation in the following ways:

{BIT2 BIT1} = 00: in dark, use 8 bit original data; {BIT2 BIT1} = 01: in overcast, set 3 LSB data WE (or RE) signal to 0; {BIT2 BIT1} = 10: in sunlight, set 4 LSB data WE (or RE) signal to 0.

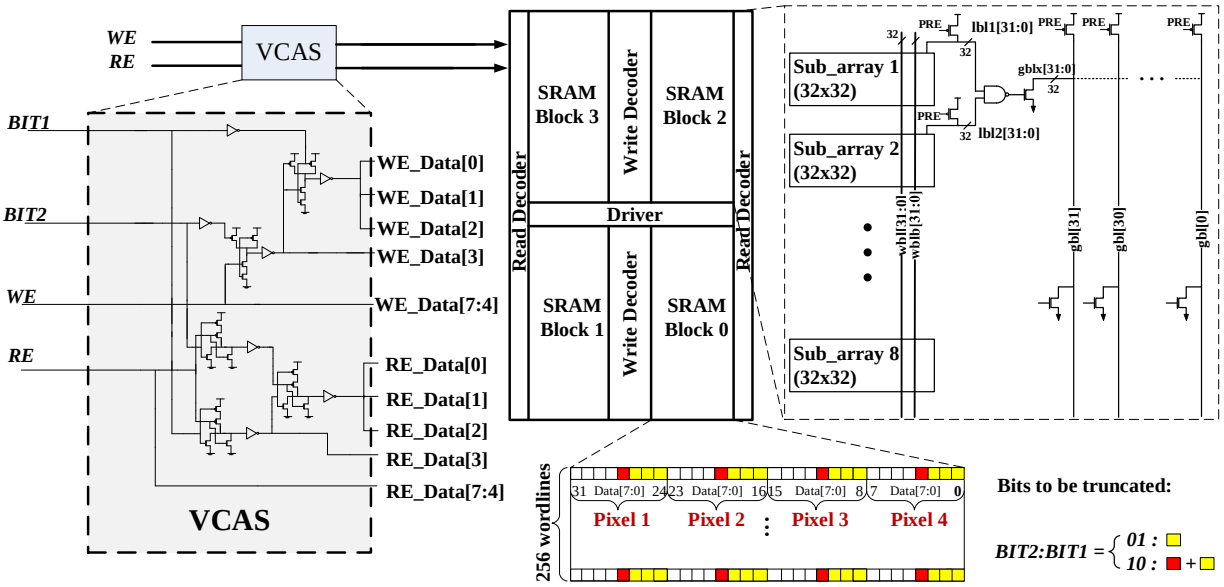


Figure 6. SRAM circuit with VCAS control circuit.

2.3. Experimental Results

2.3.1. Experimental Methodology

Experimental Setup: To verify the effectiveness of VCAS, a Verilog-based H.264 hardware simulator is implemented. As compared to software-based video coding simulator, such as JM simulator [44], the hardware simulator can specifically identify the memory modules and directly introducing hardware noise, achieving higher precision. The hardware implementation is based on a 45 nm technology [45]. Different test video sequences with diverse texture/motion features are used in our simulation [47]. The frame size is 176×144 pixels.

Power Consumption Model: The power consumption of mobile video memory is modelled as:

$$P = P_w + P_r \tag{3}$$

$$P_w = \sum_{k=0}^{7} \sum_{\substack{i=0,1 \\ j=0,1}} [F_k(i, j) \cdot P_{wk}(i, j)] \tag{4}$$

$$P_r = \sum_{k=0}^{7} \sum_{i=0,1} [F_k(i) \cdot P_{rk}(i)] \tag{5}$$

where k is the bit number; i and j are old and new values stored in an SRAM. F(i,j) indicates the bit change (switching) probability from i to j, which is extracted from the video frame in the decoding process.

2.3.2. Results

Speed: First, the performance of VCAS is evaluated to make sure it is fast enough to support the typical mobile videos. Due to space limitations, only the timing diagram of VCAS in the sunlight is presented, which is shown in Fig. 8. The same data (0x97, 0xf3, 0xc6, 0x0e) is written to the same address (0x0a, 0x1a, 0x25, 0x3b) in the writing and reading processes. If the mobile sensor detects the dark and BIT2 and BIT1 are both 0s, VCAS stores all data and the

original decoded video is displayed; if the user is viewing the device in overcast and BIT1 is 1, VCAS will truncate 3 LSBs; if it is in sunlight and BIT2 is 1, VCAS will truncate 4 LSBs (0x90, 0xf0, 0xc0, 0x00). The results also show that the proposed technique is fast enough to deliver the typical mobile video sequences (11MHz for CIF/QCIF and 72MHz for HD720 [3, 8]).

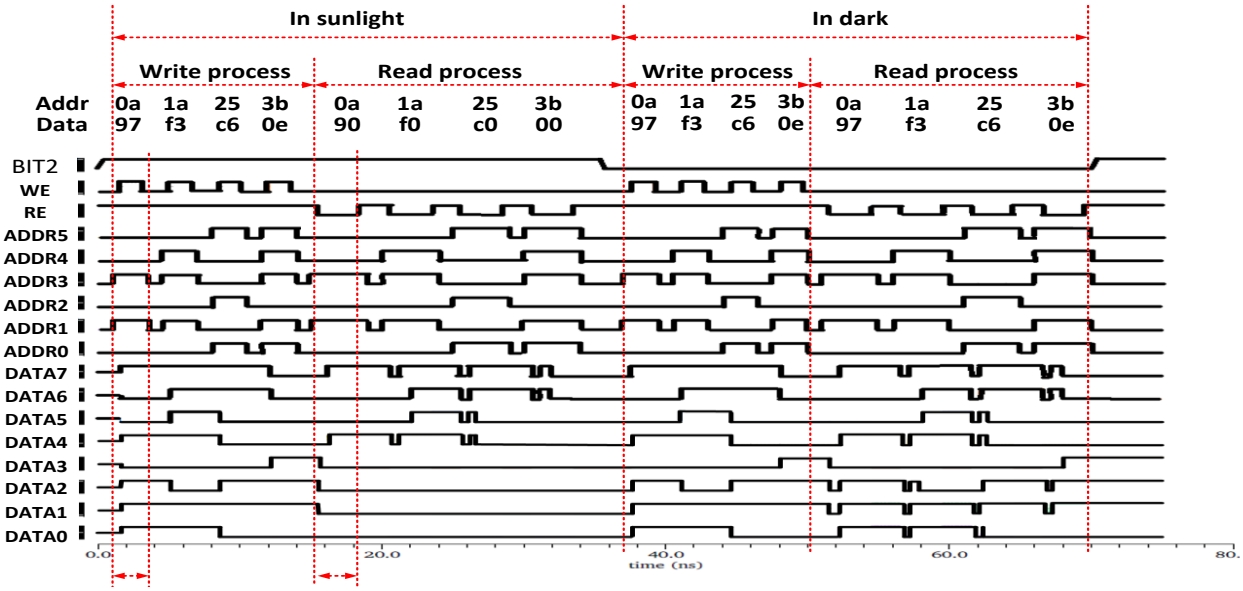


Figure 7. Timing diagram of VCAS in sunlight (BIT2 = 1 and BIT1 = 0).

Accordingly, if a user walks into a dark place from sunlight, VCAS can quickly adjust the video quality.

Area Overhead: The area overhead of the implemented VCAS is also evaluated. Fig. 9 shows the overall layout of the VCAS and control circuit. It can be seen that, since only several gates are added into the conventional 8T SRAM memory (see Fig. 6), the area overhead is negligible.

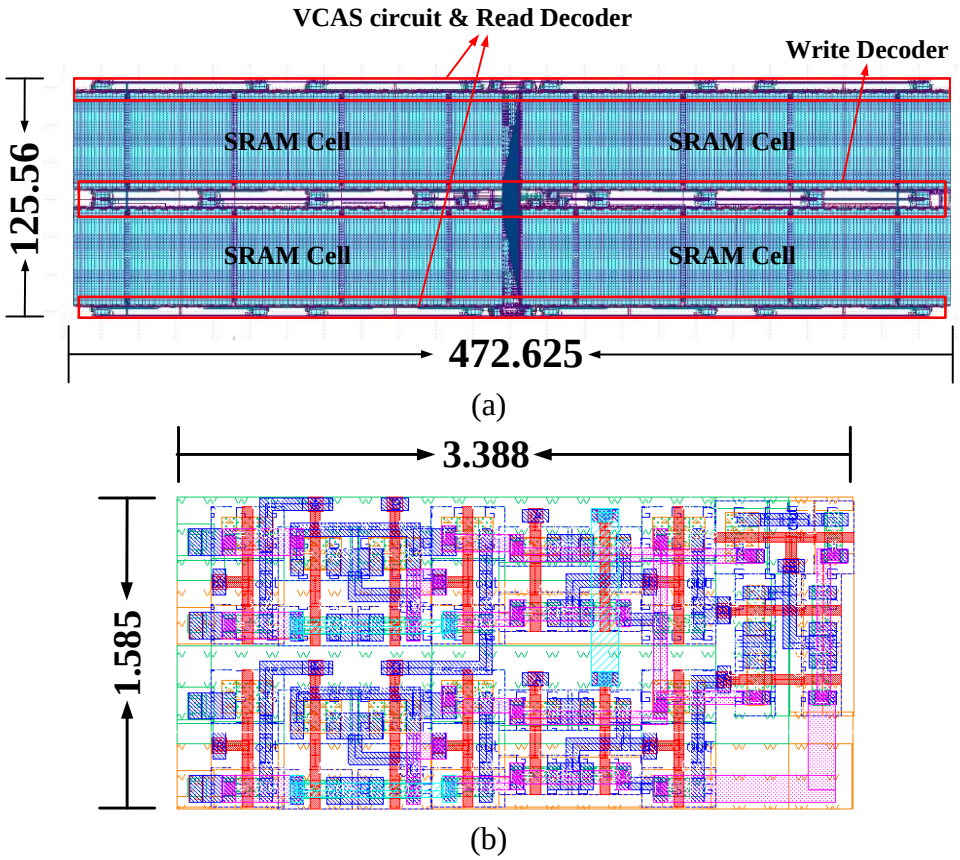
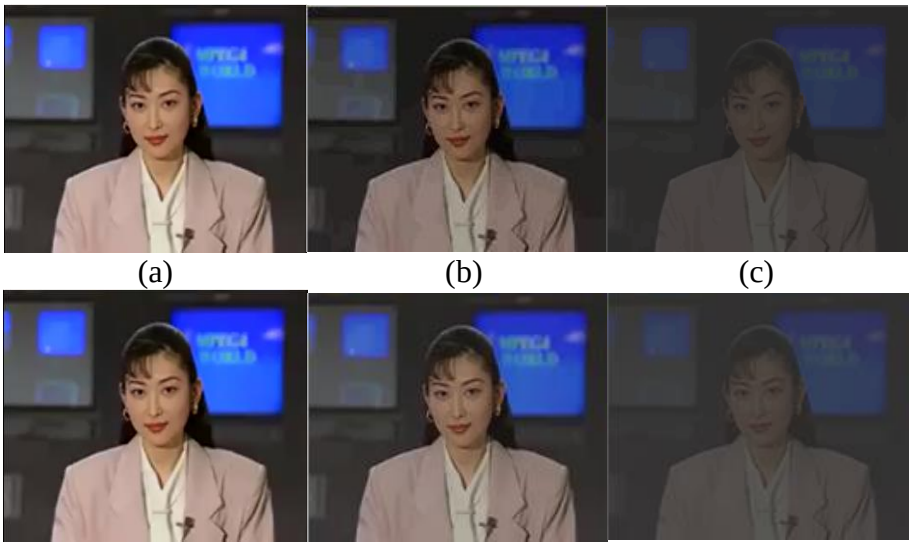


Figure 8. Proposed layout (in μm). (a) VCAS SRAM, (b) VCAS control circuit.

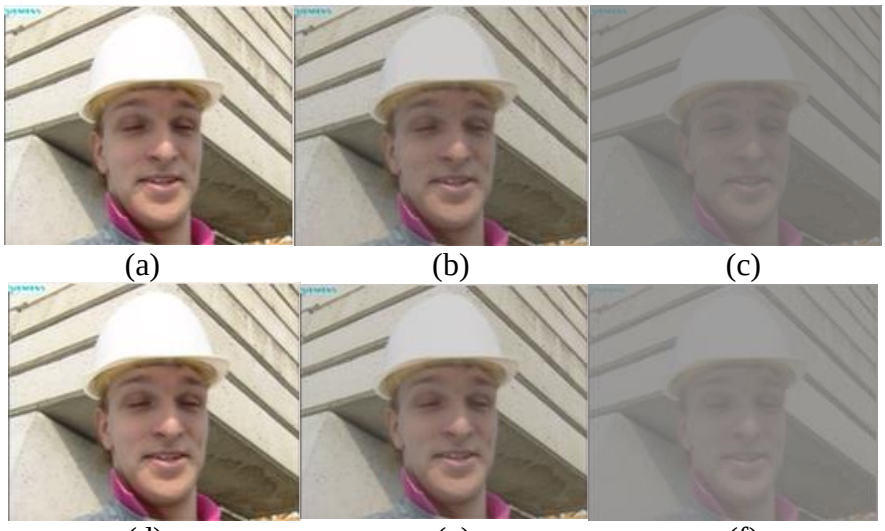
Output Quality: The output quality of the video with VCAS in different viewing contexts are shown based on three famous H.264 benchmark videos. As shown in Figs. (9)-(11), VCAS achieves similar perceivable quality as conventional memory.



(a) (b) (c)

(d) (e) (f)

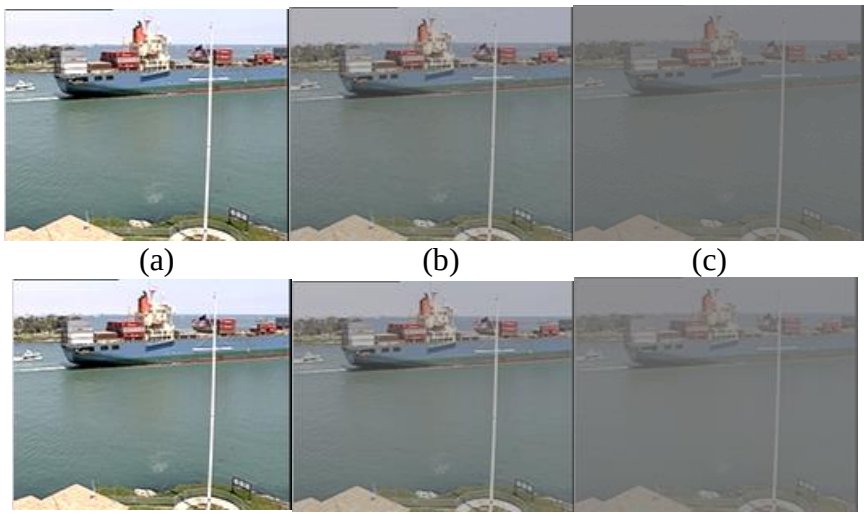
Figure 9. (a)-(c): video akiyo in different contexts (Proposed), (d)-(f): video akiyo in different contexts (Conventional).



(a) (b) (c)

(d) (e) (f)

Figure 10. (a)-(c): video foreman in different contexts (Proposed), (d)-(f): video foreman in different contexts (Conventional).



(a) (b) (c)

(d) (e) (f)

Figure 11. (a)-(c): video container in different contexts (Proposed), (d)-(f): video container in different contexts (Conventional).

Power Consumption: Finally, based on the power consumption model expressed in (3)-(5), the power efficiency of VCAS is evaluated. Table 2 lists the probability of bit changes in the popular benchmark videos, Akiyo, Foreman, and Container under three luminance levels. As shown in Table 2, by using the bit truncation technique, the switching probability of 3 LSBs (0th-2nd) in overcast or 4 LSBs (0th-3rd) in sunlight from 0 to 1 or from 1 to 0 is significantly reduced, thereby achieving power savings. Table 3 lists the power savings achieved by VCAS. As shown, 24.8% and 32.6% power savings can be achieved in the overcast and sunlight, respectively, with the proposed VCAS.

Table 2. Switching probability of each bit.

Context	Bit	1-1	1-0	0-1	0-0
Dark	7^{th}	0.660688	0.017224	0.016672	0.305416
	6^{th}	0.598384	0.024224	0.023592	0.353792
	5^{th}	0.337656	0.041608	0.041232	0.579504
	4^{th}	0.380208	0.0654	0.065304	0.489088
	3^{th}	0.458296	0.091464	0.092744	0.357504
	2^{nd}	0.36188	0.135296	0.133344	0.36948
	1^{st}	0.321144	0.185688	0.189848	0.303328
	0^{th}	0.271816	0.236584	0.236608	0.254992
Overcast	7^{th}	0.660688	0.017224	0.016672	0.305416
	6^{th}	0.598384	0.024224	0.023592	0.353792
	5^{th}	0.337656	0.041608	0.041232	0.579504
	4^{th}	0.380208	0.0654	0.065304	0.489088
	3^{th}	0.458296	0.091464	0.092744	0.357504
	2^{nd}	0	0	0	1
	1^{st}	0	0	0	1
	0^{th}	0	0	0	1
Sunlight	7^{th}	0.660688	0.017224	0.016672	0.305416
	6^{th}	0.598384	0.024224	0.023592	0.353792
	5^{th}	0.337656	0.041608	0.041232	0.579504
	4^{th}	0.380208	0.0654	0.065304	0.489088
	3^{th}	0	0	0	1
	2^{nd}	0	0	0	1
	1^{st}	0	0	0	1
	0^{th}	0	0	0	1

Table 3. Power savings of VCAS in different context.

context	***In dark***	***In overcast***	***In sunlight***
bit	xxxxxxxx	xxxxx000	xxxx0000
Write power	3.50E-07	1.10E-07	6.96E-08
Read power	1.11E-06	6.94E-07	5.55E-07
Power savings	0%	44.9%	57.2%

Comparison with Prior Work: Table 4 compares the VCAS performance with the state-of-the-art. VCAS exhibits the lowest implementation cost (area overhead) with dynamic power-quality tradeoff and the best video quality. VCAS demonstrates highest power efficiency-area overhead tradeoff.

Table 4. Comparison with existing study.

	TVLSI'08 [11]	*TCASVT'11 [9]*	*TCASII'12 [3]*	*ISVLSI'07 [25]*	This work		
					In dark	In overcast	In sunlight
video specific characteristics	correlation of MSB	contribution of MSB and LSB	different contribution of MSB and LSB	reconstructed image memory	ambient luminance awareness		
dynamic adaption	No	No	No	No	Yes		
low-power technique	data filpping	6T+8T bitcells	8T+10T bitcells	10T non-precharge	Bit truncation		
bitcell array modification	Yes	Yes	additional word line	No	No		
additional hardware needed	majority logic and data flipping block	single-ended 6T, peripheral circuitries	No	10T SRAM cell	VCAS circuit		
power penalty for extra bits	Yes	No	No	No	No	No	No
readout power[1]	-14%	-32%	-95%	-74%	0%	-44.9%	-57.2%
video quality	good	acceptable	acceptable	good	good	good	good
area	+14%	+11.64%	+52%	+14.4%	~0%		
technology	90nm	90nm	45nm	90nm	45nm		

2.4. Conclusion of VCAS

In this thesis, we have presented a new embedded memory to enhance power efficiency of mobile video streaming. The proposed memory can adapt user experience under environmental visual interference, thereby enabling better power-quality tradeoffs to prolong the battery life. Hardware schemes including system-level and circuit-level are developed to implement the proposed technique, based on the bit truncation technique. The results demonstrate up to 57.2% power reduction with negligible area overhead, while maintaining almost the same perception quality, as compared to the conventional memory.

3. PROPOSED DATA-DRIVEN POWER EFFICIENT ADAPTABLE SRAM HARDWARE

In this thesis, we propose a low-cost Data-Driven power efficient Adaptable SRAM Hardware (D-DASH) design with dynamic power-quality tradeoff for mobile video applications. By introducing advanced data mining techniques, we investigate meaningful data patterns hidden in video data and incorporate these key findings into our hardware design. D-DASH enables three levels of power-quality management (up to 43.7% power savings) with negligible area overhead (0.06%).

3.1. Mobile Video Data Pattern Analysis

3.1.1. Mobile Video Data Characteristics

Mobile video application characteristics imply that it is possible to incorporate application-level video data behaviors to the hardware-level design process. Although mobile videos are delivered over different networks and are visualized in various mobile terminals, there are three common characteristics that may potentially contribute to hardware-level memory design [59]: (1) inputs: the video data is noisy and redundant; (2) outputs: the videos on mobile devices are generated for humans and minor variations cannot be discerned by humans' eyes; and (3) computation patterns: statistical computations during the video decoding process potentially result in specific data patterns, which can contribute to low-power hardware design. However, it is difficult for hardware designers to observe the inherent video data behaviors directly from the large volume of video data. To achieve this, we introduce data-mining techniques to comprehensively explore mobile video storage data characteristics. In particular, we use an association rule mining technique to explore the relationship between different video data bits and obtain data patterns for efficient hardware design.

3.1.2. Data Mining Assisted Video Analysis

Today's mobile video frames are typically stored and processed in YUV format. It includes one luma (Y) component, which contains the brightness information of the image and two chroma components which contain the blue-difference (Cb) and red-difference (Cr) color information. Fig. 12 shows a typical frame of video data stored in embedded memory using a 352 × 288 resolution YUV 4:2:0 video as an example. As shown, each pixel has 8-bit luma data and 8-bit subsampled chroma data. Since video data is stored in on-chip memory as binary bits, we utilize an association data mining technique to identify the bit-level data patterns.

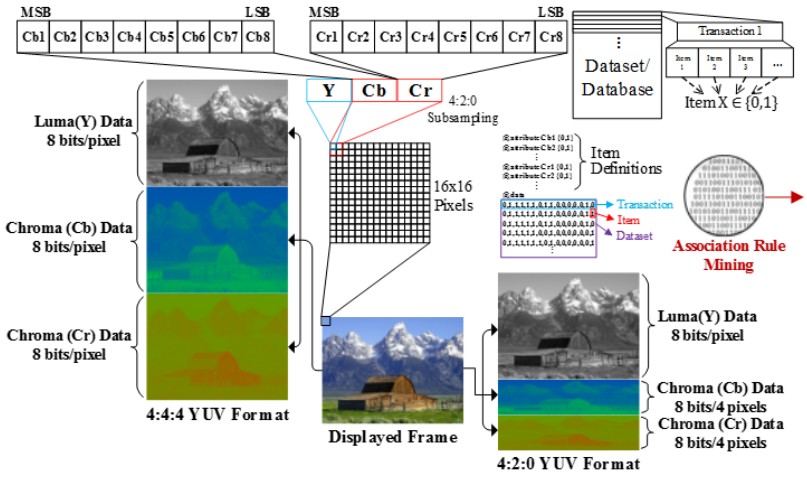
Association rule mining was introduced in 1993 to discover relationships between different variables, called items, in a dataset or database [55]. A complete dataset is made up of many transactions where each transaction contains a set of items. Each item can be associated with a binary attribute, 0 or 1, that is used to distinguish that item is present or not in its corresponding transaction. This type of data organization is illustrated in Fig. 12. Each resulting rule, generated from the association rule mining process, is an implication of the form $X \rightarrow Y$, where X and Y are disjoint sets of, or individual, items. Each rule is also accompanied by collected statistics from the dataset called support and confidence values. The support value for a set of items is the proportion of transactions in the dataset that contains such set of items. The confidence value for an association rule, $X \rightarrow Y$, is the proportion of transactions that contain X which also contain Y, or the conditional probability $P(Y \mid X)$.

To enable association data mining, we use different video benchmarks to build a dataset, including 12 videos from [47] and 4 videos from [50]. In total, the video data size is 415,600,000 bytes. 12 benchmark videos from [47] were combined into a single YUV file. This combined video file contains a total of 3470 frames with 352 × 288 resolution (Video source for 4K HEVC

videos). Each chroma bit is defined as an individual item and a sample of the .arff format with descriptions of its different parts can be seen in Fig.12. We used Weka [49] to perform the well-known association rule mining algorithm - Apriori on our large video dataset. To evaluate the impact of compression on the identified data patterns, the two formats including the non-sampled YUV 4:4:4 format and the subsampled YUV 4:2:0 format are investigated using data mining techniques. They both contain eight bits of luma (Y) data and eight bits of each chroma component (Cb and Cr) that are shared among 4 pixels.

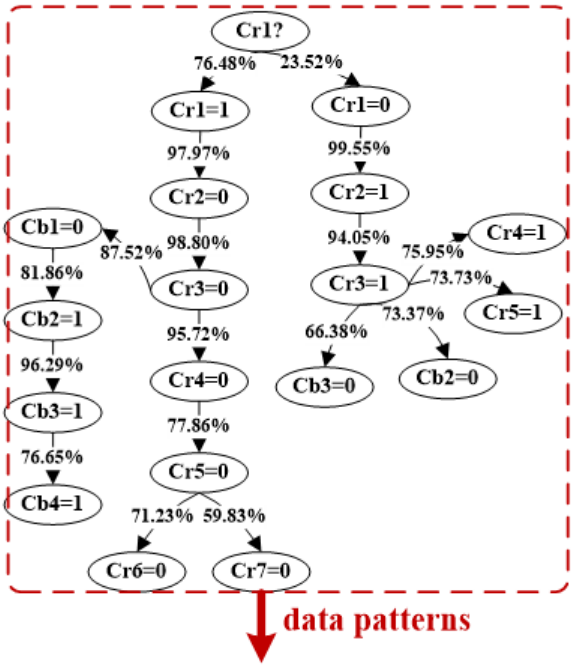
With the advancement of video processing technology, new, higher definition video resolutions have recently begun to gain popularity. Ultra-high-definition (UHD) resolutions such as 4K and 8K are soon to become the standard in terms of video technology. 4K video in particular consists of 3840 × 2160 pixels (8.3 megapixels) and is being implemented in modern television systems by various companies around the world. Association data mining was also used to discover rules from 12 UHD 4K videos from [75, 76]. The discovered rules for both normal videos and 4K HEVC videos can be seen in Fig. 13 and Fig. 14.

One interesting data pattern we obtained from the results is the strong association of chroma bits to Cr's most significant bit (MSB), Cr1. As shown in both Fig. 13 and Fig. 14, if the value of Cr1 is equal to 1 (0) the remaining Cr bits have a larger probability to be 0 (1). Other than Cb1, the majority of Cb bits have a larger probability to be 0 (1) when Cr1 equals 0 (1). Based on such identified data patterns, a flexible D-DASH with power-quality adaption is implemented which will be discussed in the following section.



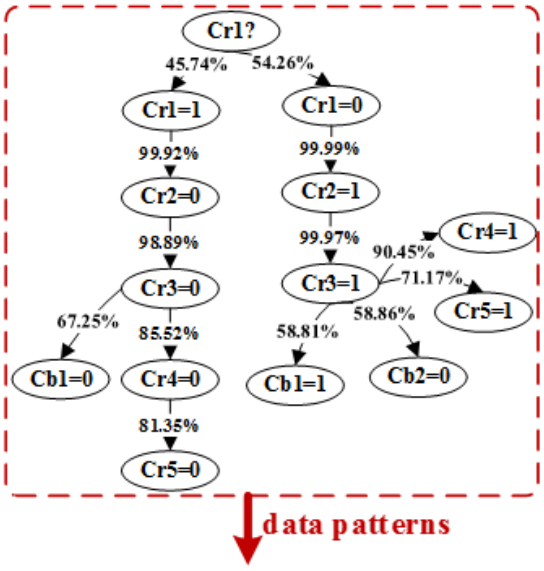
Association Rules	Support (%)	Confidence (%)
Cr1=0 Cr2=1 → Cr3=1	38.341	94.050
Cr1=0 Cr2=1 Cr3=1 → Cr4=1	29.120	75.950
Cr1=0 Cr2=1 Cr3=1 → Cr5=1	28.270	73.731
Cr1=0 Cr2=1 Cr3=1 → Cb2=0	28.132	73.373
Cr1=0 Cr2=1 Cr3=1 → Cb3=0	25.450	66.377
Cr1=1 Cr2=0 → Cr3=0	57.153	98.798
Cr1=1 Cr2=0 Cr3=0 → Cr4=0	54.707	95.720
Cr1=1 Cr2=0 Cr3=0 Cr4=0 → Cr5=0	42.593	77.857
Cr1=1 Cr2=0 Cr3=0 Cr4=0 Cr5=0 → Cr6=0	30.338	71.229
Cr1=1 Cr2=0 Cr3=0 Cr4=0 Cr5=0 → Cr7=0	25.485	59.833
Cr1=1 Cr2=0 Cr3=0 → Cb1=0	50.019	87.517
Cr1=1 Cr2=0 Cr3=0 Cb1=0 → Cb2=1	45.947	91.860

Figure 12. Data mining assisted video data analysis and discovered rules.



Values	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Cr1=0, CrX=1	99.9%	92.2%	89.7%	71.8%	71.4%	64.3%	56.3%
Cr1=1, CrX=0	99.2%	97.8%	83.3%	67.5%	59.8%	57.3%	53.2%
Cr1=0, CbX=0	60.5%	60.4%	64.0%	56.6%	52.8%	48.6%	50.5%
Cr1=1, CbX=1	86.6%	82.9%	55.3%	51.4%	54.8%	49.5%	50.4%

Figure 13. Rules diagram for normal video.



Values	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Cr1=0, CrX=1	99.9%	99.9%	90.5%	71.1%	64.9%	54.8%	52.2%
Cr1=1, CrX=0	99.9%	99.0%	85.4%	79.6%	64.3%	55.8%	52.7%
Cr1=0, CbX=0	58.8%	50.1%	53.5%	50.6%	46.4%	48.6%	49.9%
Cr1=1, CbX=1	67.8%	70.8%	53.2%	52.6%	51.8%	49.1%	49.9%

Figure 14. Rules diagram for 4K HEVC video.

3.1.3. Video Quality Metrics

The well-known peak signal-to-noise ratio (PSNR) metric is applied widely to evaluate video quality [60, 62, 21], which is defined as [60]:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{6}$$

where *MSE* is the mean square error between the original videos (*Org*) and the degraded videos (*Deg*), expressed as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[Org(i,j) - Deg(i,j)]^2 \tag{7}$$

However, recent research shows that the PSNR cannot describe the true human perception of videos since it only takes the amount of errors into account, not necessarily the effect that errors have on the user's perception of the image being displayed [48]. Accordingly, the structural similarity (SSIM) metric is developed to predict the perceived image quality and it

combines separate calculations for luminance, contrast, and structure changes all together, as expressed in (8) [48]:

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (8)$$

where the luminance comparison function *l(x,y)*, is a function of the mean intensities, μ_x and μ_y, the contrast comparison function *c(x,y)*, is a function of the standard deviations, σ_x and σ_y, and the structural comparison function *s(x,y)*, is a function of the correlation between *x* and *y*, σ_{xy}. Setting $\alpha = \beta = \gamma = 1$ in the original equation results in the second equation. C_1 (C_2) is a constant that is included to avoid instabilities when the sum of the means (standard deviations) squared is equal to values near zero.

In our analysis, we use both PSNR and SSIM to evaluate the video output quality. The quality reduction (*PSNR(SSIM)% Reduction*) can be calculated using (9):

$$PSNR(SSIM)\ \%\ Reduction = \left(1 - \left(\frac{Output\ PSNR\ (SSIM)}{Original\ PSNR\ (SSIM)}\right)\right) \times 100\% \quad (9)$$

3.2. D-DASH Technique

This section presents the identified data patterns enabled D-DASH. D-DASH is highly flexible with three design schemes (D-DASH-I to D-DASH-III), providing a run-time dynamic power-quality trade-off for mobile video streaming.

3.2.1. D-DASH-I

D-DASH-I enables zero-cost power-efficient storage, using *data-aware* low-power readout buffer connections based on the obtained data patterns in mobile video data, as shown in Fig. 14. In conventional SRAM, a readout buffer consisting of two NMOS transistors is used to access the stored value by connecting it to reversed storage node (QB) [29], as shown in Fig. 14 (a). During the reading process, the read bit-line (RBL) is precharged to supply voltage (Vdd)

before RWL is asserted. In the case a *1* is stored in the SRAM cell and RWL is asserted, RBL will stay at Vdd as the bottom NMOS is turned off and there is no switching activity in RBL, enabling low-power reading process. In the case that a *0* is stored in the SRAM cell, the RBL will be discharged to ground (GND), resulting in large readout power consumption. Readout power consumption dominates the total power consumption of mobile video memory due to more frequent read operations [3, 12, 60, 62]. In this thesis, data-aware low-power readout buffer connections are proposed. The traditional connection as shown in Fig. 14 (a) are referred to as type-*1* bitcell. Alternatively, a type-0 bitcell is presented to achieve low-power reading *0* process, by connecting readout buffers to Q, as shown in Fig. 14 (b). Note that, as compared to the conventional SRAM bitcell (type-1), the type-0 bitcell does not cause any silicon area overhead.

The obtained data patterns in before show that, in the chroma data for each pixel, there is over 70% probability that the Cr and Cb data will be 1000000001111111 in binary. Based on this, type-1 and type-0 bitcells are applied to store/load 1 and 0, respectively. Accordingly, for the majority of chroma data, the switching activity during the readout process is significantly reduced, achieving power savings without area overhead. Fig. 14 (c) shows the structure of one wordline in D-DASH-I. Based on the identified data pattern, it uses an optional combination of type-0 and type-1 bitcells to enable zero-overhead power efficient mobile video on-chip storage.

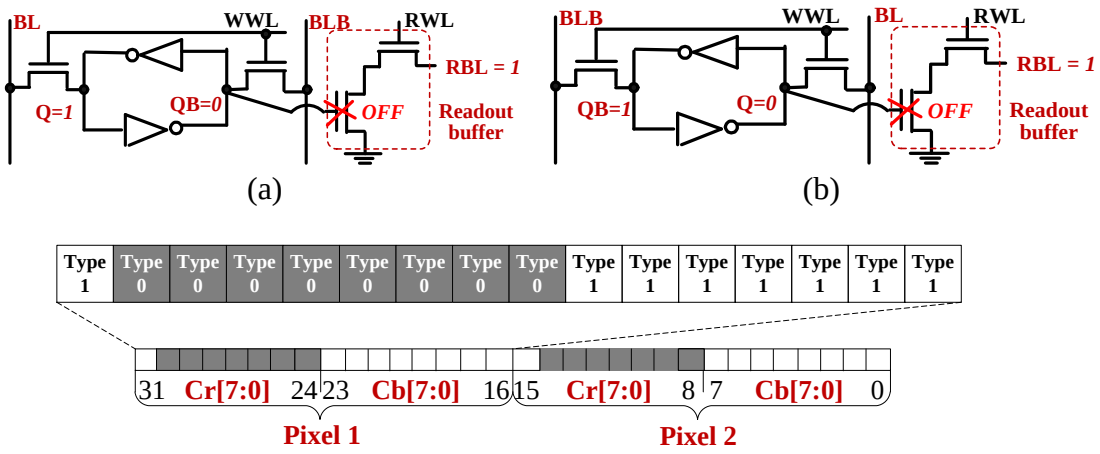


Figure 15. Data-aware D-DASH bitcells. (a) type-1 bitcell without discharging when reading 1, (b) type-0 bitcell without discharging when reading 0, (c) data pattern based wordline.

3.2.2. D-DASH-II

Based on D-DASH-I, we implement D-DASH-II for additional power savings by considering the rest of the data with the pattern of 1000000001111111. Since the most significant bit (MSB) of the Cr data determines the value of lower order bits (LSBs) as shown in the identified patterns, we implement a write circuit and a read circuit to the SRAM to maximize the read bitline power saving, as shown in Fig. 15. In the write circuit, we first detect the MSB of the Cr data and then determine whether to invert the input data or not, and use a similar flag-bit scheme as [29] to indicate if the data is flipped or not: 1 means the data is not inversed and 0 means data is inversed. The scheme implemented in [29] uses an additional bitcell to store the flag bit, causing 7% area overhead. To minimize the overhead, we utilize the least significant bit (LSB) of the chroma data to store the flag bit. Our results in Table 5 show that using the Cr LSB induces negligible video quality degradation (0.044% PSNR reduction and 0.058% SSIM reduction) and therefore we use the Cr LSB to store the flag bit, as shown in Fig. 15. Accordingly, by enabling D-DASH-II, the design can achieve power savings as compared to the implementation cost and video quality degradation of the original D-DASH design.

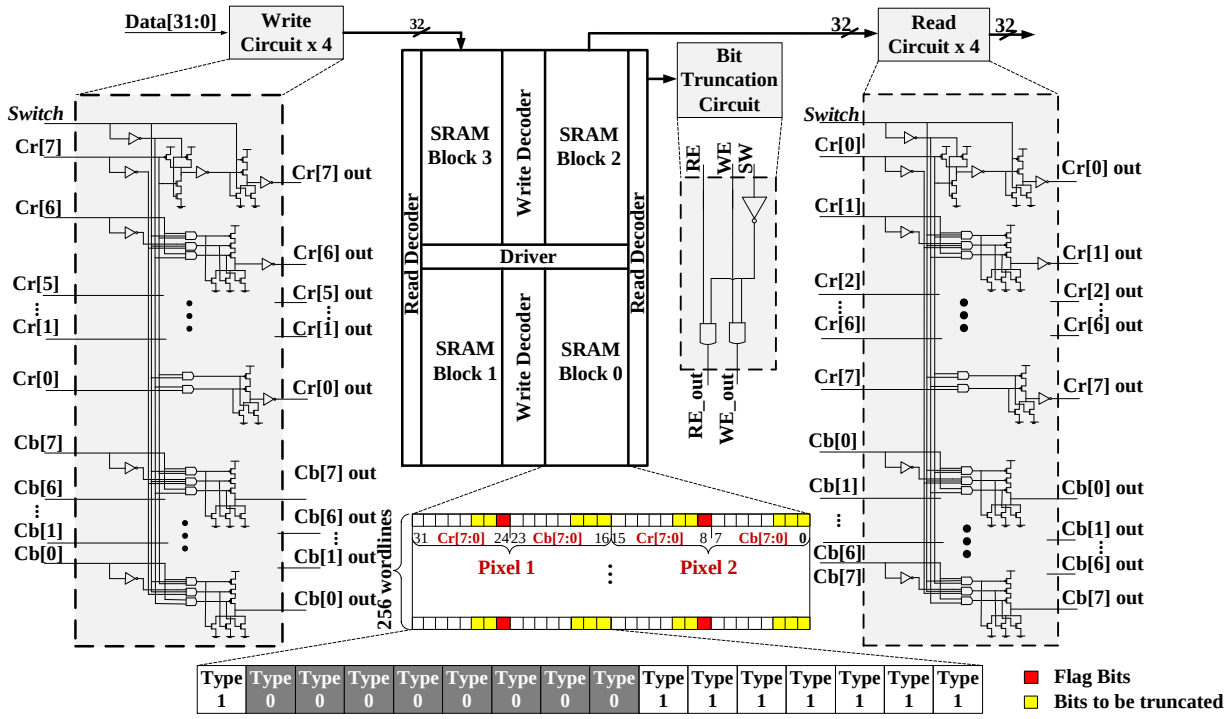


Figure 16. D-DASH with real-time adjustment between three schemes.

Table 5. PSNR and SSIM calculations.

# of drop bits	PSNR	SSIM	PSNR % Reduction	SSIM % Reduction
0 (Original)	36.868	0.934367	0.000%	0.000%
0 (Flag Bit)	36.852	0.933826	0.044%	0.058%
1	36.848	0.933213	0.054%	0.124%
3	36.641	0.929893	0.616%	0.479%
5	35.544	0.922561	3.592%	1.264%
7	32.593	0.910895	11.595%	2.512%
9	27.874	0.895294	24.395%	4.182%
11	21.506	0.862433	41.667%	7.699%
13	14.851	0.756258	59.718%	19.062%
15	10.782	0.623662	70.756%	33.253%

3.2.3. D-DASH-III

To meet the high power efficiency requirement of some video applications, we further design D-DASH-III to maximize the power savings based on bit-truncation (dropping) technique. Bit-truncation technique has been widely used in mobile video memory area [29]. To determine the number of bits for truncation, we evaluate the video quality with the bit-truncation technique and the results are listed in Table 1. As the number of truncated bits is larger than 7,

PSNR and SSIM reduction are significant (PSNR reduction ≥ 24.395% and SSIM reduction ≥ 2.512%), indicating large video quality degradation. Accordingly, the number of bits for truncation can be 5 or 7. We further evaluate the video output quality using the *sign_irene* video benchmark as shown in Fig. 16. The video *sign_irene* contains blue and red colors that would be directly affected by the corruption of chroma data [51]. It shows that, *truncating 5 LSB bits (2 LSBs of Cr and 3 LSBs of Cb) is an optimized trade-off between video quality and power saving*. To enable the bit-truncation technique, we implement the control circuit as shown in Fig. 15. The control bit SW is connected to Write Enable (WE) and Read Enable (RE) of a wordline. When SW is *0*, all of the 32 bitcells connected to the wordline work as traditional bitcells; when SW is *1*, D-DASH-III is enabled and the outputs (WE_out and RE_out) will disable the WE and RE of the truncated bitcells (yellow bitcells shown in Fig. 15) and therefore, bit truncation is enabled to achieve additional power savings.

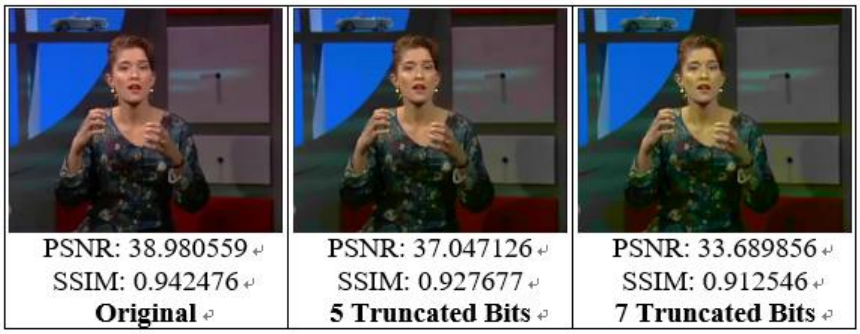


Figure 17. sign_irene output with bit-truncation technique.

3.3. Simulation Results

To evaluate the effectiveness of the proposed technique, a 32kb SRAM is implemented using a high-performance 45-nm FreePDK CMOS process to meet the multi-megahertz performance requirement of today's mobile video decoders.

3.3.1. Performance

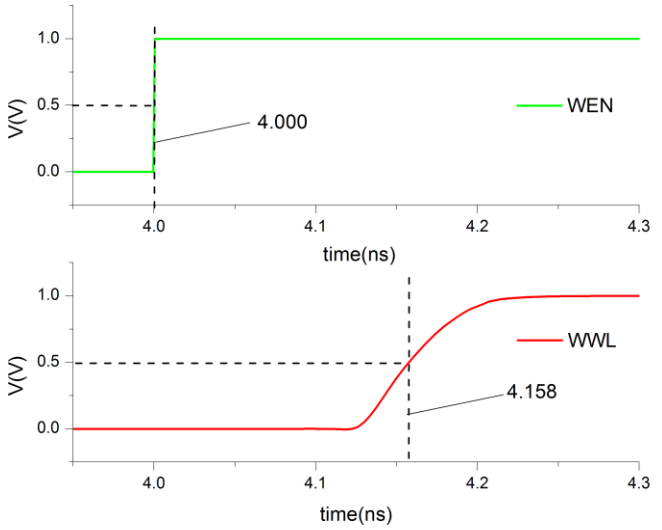
We first evaluate the performance of the proposed D-DASH. Table 2 lists the detailed performance parameters. Write delay is approximately 0.15 ns, and read delay is around 0.31 ns, which successfully delivers high-quality video format such as 8K Ultra HD applications [58]. Fig. 6 shows the waveform of write and read simulations. Write Enable(WEN) signal becomes valid at 4ns and the Write WordLine(WWL) becomes valid at 4.158ns. Read Enable(REN) becomes valid at 20ns, and after 0.314ns, read data become valid.

Table 6. SRAM Specifications.

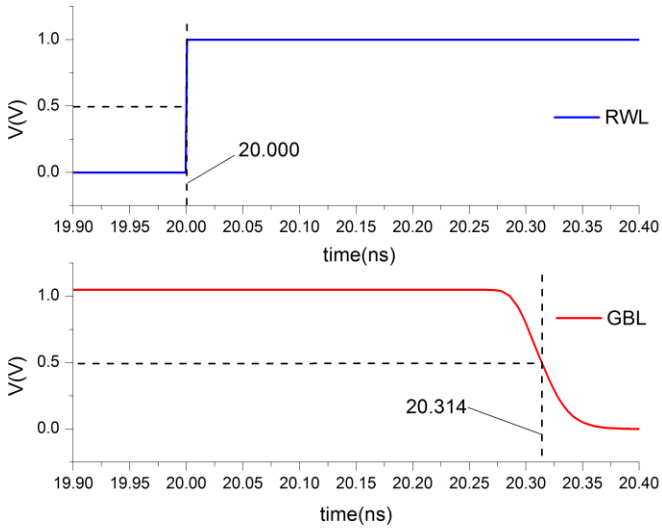
Technology	45nm CMOS
SRAM Storage	32Kb
Area	125.56 x 475.66 μm^2
Write delay	0.158 ns
Read delay	0.314 ns

3.3.2. Layout

We also evaluate the area overhead of the proposed design. Fig. 18 shows the layout of D-DASH. For scheme II, the area overhead is 0.64%; for scheme III, after careful layout design, we integrate the control circuit with read decoder, without additional area overhead.



(a)



(b)

Figure 18. SRAM write and read speed waveform.

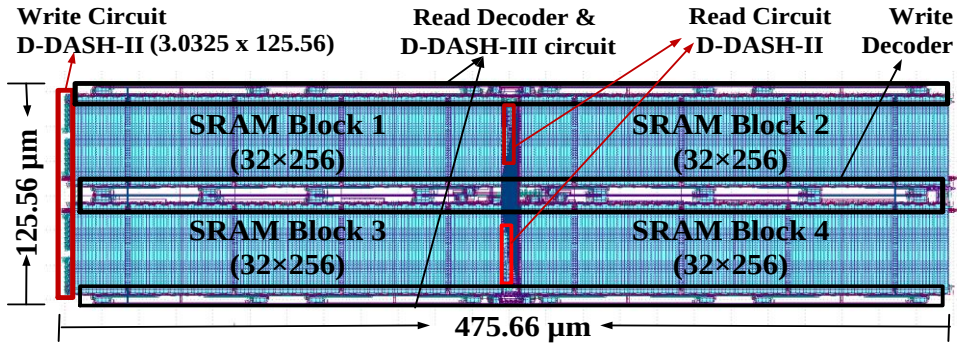


Figure 19. Layout of D-DASH.

3.3.3. Output Quality

We further expand the test video data to larger-scale and real video datasets to test our proposed technique. We randomly pick 50 videos in YouTube-8M [66], which is recently released by Google, and YouTube-8M is the largest video dataset to date. Specifically, 50 unique videos from different categories in YouTube-8M dataset, with 288 MB total data size, representing 2,500 individual frames, was tested using our data-mining methods. A script was written to download these 50 videos from the ~7 million available videos in Youtube-8M dataset. After each video was downloaded, we randomly select 50 contiguous frames from the video and convert from the MP4 format to the raw YUV format for data-mining analysis. Fig. 19 shows five video outputs as examples. Video Akiyo, Coastguard, and Foreman are videos used in analyzing the data patterns while video Concert and Game are videos we get from Youtube-8M database for testing. D-DASH-II can deliver good video output quality and D-DASH-III results in negligible video degradation to achieve optimal power efficiency.

D-DASH	*Akiyo*	*Coastguard*	*Foreman*	*Concert* *Youtube-8M*	*Game* *Youtube-8M*
D-DASH-II	 PSNR: 41.194586, SSIM: 0.961432	 PSNR: 35.638277, SSIM: 0.919196	 PSNR: 37.335870, SSIM: 0.923268	 PSNR: 37.958962, SSIM: 0.931152	 PSNR: 38.796625, SSIM: 0.951849
D-DASH-III	 PSNR: 37.971173, SSIM: 0.950497	 PSNR: 34.781284, SSIM: 0.914910	 PSNR: 35.829443, SSIM: 0.913802	 PSNR: 35.461625, SSIM: 0.920048	 PSNR: 37.668134, SSIM: 0.939547

Figure 20. Video output.

3.3.4. Power Savings

To evaluate the power efficiency of D-DASH, we model the read bitline (RBL) power consumption of mobile video memory as:

$$P_r = \sum_{k=0}^{16} \sum_{i=0,1} [F_k(i) \cdot P_{rkt}(i) \cdot Z(k)] \quad (10)$$

where Pr is the power consumption on read operation; k is the bit number; t is the SRAM type; i is the value stored in SRAM, F(i) indicates the probabilities of a bit to be 0 and 1, which is shown in Table 8; Z(i) indicates if the bit will be truncated (if truncated, Z(i) will be 0, if not truncated, Z(i) will be 1).

Table 7 lists the read power consumption for two types of D-DASH bitcells. We also extract the probability of each bit being 0 or 1 from all the benchmarks and the results are listed in Table 8. The bits marked in grey are truncated bits in D-DASH-III.

Table 9, Table 10, and Table 11 concludes the power savings of our proposed technique over a standard SRAM design. D-DASH enables power savings from 7.82% (D-DASH-I) to 45.82% (D-DASH-III).

Table 7. RBL power consumption for different types of SRAM.

SRAM type	Read 1	Read 0
Power of Type 1	3.22e-8	5.80e-8
Power of Type 0	5.88e-8	3.13e-8

Table 8. Probability of each bit being 0 or 1 (F(i)).

bitcell type	Bit	D-DASH-II		D-DASH-II (4K HEVC)		D-DASH-II Youtube-8M	
		0	1	0	1	0	1
1	Cr1	0	1	0	1	0	1
0	Cr2	0.9940	0.0060	0.9997	0.0003	0.9997	0.0003
0	Cr3	0.9649	0.0351	0.9950	0.0050	0.9817	0.0183
0	Cr4	0.8482	0.1518	0.8818	0.1182	0.7818	0.2182
0	Cr5	0.6853	0.3147	0.7501	0.2499	0.7874	0.2126
0	Cr6	0.6254	0.3746	0.6461	0.3539	0.5852	0.4148
0	Cr7	0.5892	0.4108	0.5529	0.4471	0.5515	0.4485
0	Cr8	0.5097	0.4903	0.5006	0.4994	0.5262	0.4738
0	Cb1	0.8331	0.1669	0.6288	0.3712	0.5578	0.4422
1	Cb2	0.1958	0.8042	0.3709	0.6291	0.5920	0.4080
1	Cb3	0.2240	0.7760	0.4042	0.5958	0.4943	0.5057
1	Cb4	0.4265	0.5735	0.4660	0.5340	0.4910	0.5090
1	Cb5	0.4735	0.5265	0.4851	0.5149	0.4725	0.5275
1	Cb6	0.4568	0.5432	0.5110	0.4890	0.4897	0.5103
1	Cb7	0.5069	0.4931	0.5118	0.4882	0.5021	0.4979
1	Cb8	0.4956	0.5044	0.5014	0.4986	0.4994	0.5006

Table 9. Normal videos power savings of different schemes.

SRAM Designs	Power (W)	Power saving
Conventional SRAM	7.29E-07	-
D-DASH-II	6.33E-07	13.17%
D-DASH-III	4.15E-07	43.07%

Table 10. 4K HEVC videos power savings of different schemes.

SRAM Designs	Power (W)	Power saving
Conventional SRAM	7.29E-07	-
D-DASH-II	6.48E-07	9.45%
D-DASH-III	4.28E-07	40.26%

Table 11. YouTube-8M videos power savings of different schemes.

SRAM Designs	Power (W)	Power saving
Conventional SRAM	7.29E-07	-
D-DASH-II	6.61E-07	9.32%
D-DASH-III	3.95E-07	45.82%

3.3.5. Comparison with Prior Work

Table 12 compares the D-DASH's performance with the state-of-the-art. D-DASH exhibits the lowest implementation cost (0.06%) with dynamic power-quality tradeoff. D-DASH-II and D-DASH-III exhibit the best video quality, except for reference [11], which, however, is realized with large area overhead (~14%). D-DASH-III demonstrates highest power efficiency, except for [3], but the hybrid 8T/10T structure in [3] requires bitcell array modification, resulting in as high as 52% silicon area overhead.

3.4. Conclusion of D-DASH

This thesis presents a data-driven flexible mobile video SRAM with negligible silicon area overhead (0.06%) that can be adjusted between three different levels of power-quality tradeoff. Based on the data patterns obtained by data-mining techniques, a low-power data-aware readout buffer connection technique is applied to enhance the power efficiency without

Table 12. Comparison with prior art on low-power mobile video SRAM.

	TVLSI'08 [11]	TCASVT'11 [56]	TCASII'12 [3]	JSCC'15 [57]	This work	
					D-DASH-II	D-DASH-III
video specific characteristics	correlation of MSB	contribution of MSB and LSB	different contribution of MSB and LSB	contribution of MSB and LSB	data-mining assisted video data patterns exploration	
dynamic adaption	No	No	No	Yes	Yes	
low-power technique	data filpping	6T+8T bitcells	8T+10T bitcells	ECC	data- aware bitline connection and data flipping	data-aware bitline connection, data flipping and bit truncation
bitcell array modification	Yes	Yes	additional word line	No	No	No
additional hardware needed	majority logic and data flipping block	single-ended 6T, peripheral circuitries	No	ECC encoder and decoder, write assisted circuit	control circuit consists simple gates	control circuit consists simple gates
power penalty for extra bits	Yes	No	No	No	No	No
readout power1	-14%	-32%	-95%	-28%	-13.17%	-43.07%
video quality	good	acceptable	acceptable	acceptable	good	acceptable
area	+14%	+11.64%	+52%	+1.5%	+0.06%	+0.06%
technology	90nm	90nm	45nm	28nm	45nm	45nm

implementation overhead; based on this, a bit-flipping technique is used to further improve the power efficiency by introducing 0.06% area overhead; finally, a bit-truncation technique is utilized to maximize the power efficiency with negligible video quality degradation. The developed D-DASH provides three flexible low-cost schemes, each with different power-quality tradeoff, to meet the requirements of various applications.

4. CONCLUSION

In this thesis, we proposed two low-power techniques in the application of mobile on-chip SRAM. We introduced both context awareness and data patterns into the hardware design, achieving significant power savings with negligible penalty.

4.1. Summary

Low power memories are becoming a more common technique used in mobile on-chip SRAMs.

In this thesis, two techniques are introduced to reduce the power consumption of on-chip SRAMs for the video decoding process. By introducing context information while users watching mobile videos, bit truncation technique is used to achieve different levels of power savings with negligible area overhead. Also, by analyzing data patterns in the memory, a low-power data-pattern based memory scheme is introduced to achieve 3 levels power savings with good video quality and small penalty.

The power saving for VCAS is up to 57.2% and for D-DASH is up to 43.07%. Both techniques are self-adjustable and delivering good viewing experience for users.

4.2. Suggestions for Future Work

4.2.1. Techniques Extension for Other Memories in Decoder

Since video decoders have many memories in the whole process, more possible power savings may be achieved by introducing different techniques into the memories.

4.2.2. Hardware

A FPGA based hardware video decoder platform is needed to verify the proposed techniques,.

REFERENCES

[1] IDC (2012). "The Digital Universe in 2020: Big Data, bigger Digital Shadows, and Biggest Growth in the Far East. December 2012". [Online]. Available: https://www.emc.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf

[2] N. Rastogi, "You Charged Me All Night Long," [Online]. Available: http://www.slate.com/articles/health_and_science/the_green_lantern/2009/10/you_charged_me_all_night_long.html.

[3] N. Gong, S. Jiang, A. Challapalli, S. Fernandes, and R. Sridhar, "Ultra-Low Voltage Split-Data-Aware Embedded SRAM for Mobile Video Applications," *IEEE Transactions on Circuits and Systems II*, vol. 59, no. 12, pp. 883-887, Dec. 2012.

[4] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Energy Efficient Multimedia Streaming to Mobile Devices – A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 579-597, First Quarter, 2014.

[5] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz, "Energy Consumption Modeling of H.264/AVC Video Decoding for GPP and DSP," in *Proc. 16th Euromicro Conference on Digital System Design*, pp. 890-896, 2013.

[6] M. E. Sinangil and A. P. Chandrakasan, "Application-Specific SRAM Design Using Output Prediction to Reduce Bit-Line Switching Activity and Statistically Gated Sense Amplifiers for Up to 1.9 Lower Energy/Access," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 107-117, Jan. 2014.

[7] N. Gong, S. Jiang, A. Challapalli, M. Panesar, and R. Sridhar, "Variation-and-Aging Aware Low Power embedded SRAM for Multimedia Applications," in *Proc. 25th IEEE International SoC Conference (SoCC'12)*, pp. 21-26, 2012.

[8] J. S. Wang, P. Y. Chang, T. S. Tang, J. W. Chen, and J. I. Guo, "Design of subthreshold SRAMs for energy-efficient quality-scalable video applications," *IEEE Trans. Emerging Sel. Topics Circuits Syst.*, vol. 1, no. 2, pp. 183-192, Jun. 2011.

[9] I. Chang, D. Mohapatra, and K. Roy, "A priority-based 6T/8T hybrid SRAM architecture for aggressive voltage scaling in video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 101-112, Feb. 2011.

[10] J. Kwon, I. Lee, and J. Park, "Heterogeneous SRAM Cell Sizing for Low Power H.264 Applications," *IEEE Transactions on Circuits and Systems I*, vol. 99, no. 2, pp. 1-10, Feb. 2012.

[11] H. Fujiwara, et al., “A Two-Port SRAM for Real-Time Video Processor Saving 53% of Bitline Power with Majority Logic and Data-Bit Reordering,” *ACM/IEEE Int. Symp. on Low Power Electronics and Design*, pp. 61-66, 2006.

[12] H. Fujiwara et al., “Novel video memory reduces 45% of bitline power using majority logic and data-bit reordering,” *IEEE Trans. Very LargeScale Integr. (VLSI) Syst.*, vol. 16, no. 6, pp. 620–627, Jun. 2008.

[13] W. Liu, L. Liu, S. Yin, and S. Wei, “A high parallel motion compensation implementation on a coarse-grained reconfigurable processor supporting H.264 high profile decoding,”, *IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1-3, Oct. 2014.

[14] C. Liu and M. D. Fairchild, “Measuring the relationship between perceived image contrast and surround illumination,” in *Proc. IS and TSID 12th Color Imaging Conf.*, pp. 282–288, 2004.

[15] S. Y. Choi,M. R. Luo, andM. R. Pointer, “The influence of the relative luminance of the surround on the perceived quality of an image on a large display,” in *Proc. 15th Color Imaging Conf.*, pp. 157–162, 2007.

[16] Z. Wei and K. N. Ngan, “A temporal just-noticeable distortion profile for video in DCT domain,” in *Proc. 15th IEEE International Conference on Image Processing*, pp.1336-1339, 12-15, Oct. 2008.

[17] J. Xue and C. W. Chen, “Towards viewing quality optimized video adaptation,” in *2011 IEEE In-ternational Conference on Multimedia and Expo (ICME)*, pp.1-6, July 2011.

[18] J. Xue and C. W. Chen, “Mobile JND: Environment adapted perceptual model and mobile video quality enhancement,” in *Proc. 3rd Multimedia Syst. Conf., 2012, ser. MMSys ’12*, pp. 173–183, New York, NY, USA: ACM.

[19] J. Xue and C. W. Chen, “A study on perception of mobile video with surrounding contextual influences,” in *Proc. 4th Int. Workshop Quality of Multimedia Experience (QoMEX)*, Jul. 2012, pp. 248–253.

[20] J. Xue and C. W. Chen, “Mobile video perception: new insights and adaptation strategies,” *IEEE journal of selected topics in signal processing*, vol. 8, no. 3, June 2014.

[21] N. Gong, S. Jiang, J. Wang, B. Aravamudhan, K. Sekar, and R. Sridhar, “Hybrid-Cell Register Files Design for Improving NBTI Reliability,” *Microelectronics Reliability*, vol. 52, no. 9, pp. 1865-1869, 2012.

[22] J. Wang, N. Gong, L. Hou, X. Peng, R. Sridhar, and W. Wu, “Leakage Current, active power, and Delay Analysis of Dynamic Dual Vt CMOS Circuits under P-V-T

Fluctuations," *Microelectronics Reliability,* 2011, vol. 51, no. 9, pp. 1498-1502, Sep. 2011.

[23] J. Wang, N. Gong, L. Hou, X. Peng, S. Geng, and W. Wu, "Low Power and High Performance Dynamic CMOS XOR/XNOR Gate Design," *Microelectronic Engineering*, vol. 88, no. 8, pp. 2781-2784, 2011.

[24] N. Gong, B. Guo, J. Lou, and J. Wang, "Analysis and Optimization of Leakage Current Characteristics in Sub-65nm Dual Vt Footed Domino Circuits," *Microelectronics Journal*, vol. 39, no. 9, pp. 1149-1155, Sep. 2008.

[25] H. Noguchi et al., "A 10T non-precharge two-port SRAM for 74% power reduction in video processing," in *Proc. IEEE Computer Society Annual Symp. VLSI Circuits*, pp. 107-112, March 2007.

[26] 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 *Proc. IEEE Int. Solid-State Circuits Conf.(ISSCC)*, pp. 330-606, Feb. 2007.

[27] J. P. Kulkarni, K. Kim, and K. Roy, "A 160 mV robust schmitt trigger based subthreshold SRAM," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2303-2313, Oct. 2007.

[28] F. Abouzeid, A. Bienfait, K. C. Akyel, A. Feki, S. Clerc, L. Ciampolini, F. Giner, R. Wilson, and P. Roche, "Scalable 0.35 V to 1.2 V SRAM Bitcell Design From 65 nm CMOS to 28 nm FDSOI," *IEEE J. Solid-State Circuits*, vol. 49, no. 7, pp. 1499-1505, Jul. 2014.

[29] D. Chen, X. Wang, J. Wang, and N. Gong, "VCAS: Viewing Context Aware Power-Efficient Mobile Video Embedded Memory," in *Proc. IEEE International SoC Conference (SoCC'15)*, pp. 333-338, Sep. 2015.

[30] R. V. Joshi, R. Kanj, and V. Ramadurai, "A novel column-decoupled 8T cell for low-power differential and domino-based SRAM design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 5, pp. 869–882, May 2011.

[31] M. Sharifkhani and M. Sachdev, "Segmented virtual ground architecture for low-power embedded SRAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 2, pp. 196–205, Feb. 2007.

[32] C. Lo, S. Huang, "P-P-N Based 10T SRAM Cell for Low-Leakage and Resilient Subthreshold Operation," *IEEE Journal of Solid-State Circuits*, vol.46, no.3, pp.695-704, 2011.

[33] Cisco Systems, Inc., “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017,” Feb. 2013. [Online]. Available: http://broadcastengineering.com/automation/http_abr_streaming.

[34] A. Truong, “Mobile Video Viewing Increased by 700% between 2011 and 2013,” [Online]. Available: http://www.fastcompany.com/3028446/fast-feed/mobile-tv-viewing-increased-by-700-between-2011-and-2013.

[35] M. Scott and S. Sale, “Consumer Smartphone Usage 2014: OTT Communication Services,” [Online]. Available: http://www.analysysmason.com/About-Us/News/Insight/consumers-smartphone-usage-May2014- RDMV0/.

[36] “Where Do You Use Your Smartphone?” [Online]. Available: http://zeendo.com/info/where-do-people-use-their-smartphones/.

[37] K. Nii, M. Yabuuchi, Y. Tsukamoto, S. Ohbayashi, S. Imaoka, H. Makino,Y.Yamagami, S. Ishikura,T.Terano, T. Oashi, K. Hashimoto, A. Sebe, G. Okazaki, K. Satomi, H. Akamatsu, and H. Shinohara, “A 45-nm bulk CMOS embedded SRAM with improved immunity against process and temperature variations,” *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 180-191, Jan. 2008.

[38] O. Hirabayashi, A. Kawasumi, A. Suzuki, Y. Takeyama, K. Kushida, T. Sasaki, A. Katayama, G. Fukano, Y. Fujimura, T. Nakazato, Y. Shizuki, N. Kushiyama, and T. Yabe, “A process-variation-tolerant dual-power-supply SRAMwith 0.179 cell in 40 nm CMOS using level-programmable wordline driver,” in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, pp. 458-459, Feb. 2009.

[39] F. Tachibana, O. Hirabayashi, Y. Takeyama, M. Shizuno, A. Kawasumi, K. Kushida, A. Suzuki, Y. Niki, S. Sasaki, T. Yabe, and Y. Unekawa, “A 27% Active and 85% Standby Power Reduction in Dual-Power-Supply SRAM using BL Power Calculator and Digitally Controllable Retention Circuit,” *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 118-126, Jan. 2014.

[40] K. Kushida, K. Kushida, A. Suzuki, G. Fukano, A. Kawasumi, O. Hirabayashi, Y. Takeyama, T. Sasaki, A. Katayama, Y. Fujimura, and T. Yabe, “A 0.7 V single-supply SRAM with 0.495 cell in 65 nm Technology utilizing self-write-back sense amplifier and cascaded bit line scheme,” *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1192-1198, Apr. 2009.

[41] K. Takeda et al., “A read-static-noise-margin-free SRAM cell for low-VDD and high-speed applications,” *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 113-121, Jan. 2006.

[42] S. A. Verkila, S. K. Bondada, and B. S. Amrutur, “A 100 MHz to 1 GHz, 0.35V to 1.5V supply 256 64 SRAM block using symmetrized 9T SRAM cell with controlled read,” in *Proc. Conf. VLSI Design*, pp. 560-565, Jan. 2008.

[43] Y.-W. Chiu, Y.-H. Hu, M.-H. Tu, J.-K. Zhao, Y.-H. Chu, S.-J. Jou, and C.-T. Chuang, “40 nm Bit-Interleaving 12T Subthreshold SRAM With Data-Aware Write-Assist,” *IEEE Trans. Circuits Syst. I*, vol. 61, no. 9, pp. 2578-2585, Sep. 2014.

[44] H.264/AVC JM Simulator [Online]. Available: http://iphome.hhi.de/suehring/tml/

[45] FreePDK45. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

[46] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, “Energy-efficient Architecture for Advanced Video Memory,” in *Proc. 2014 IEEE/ACM International Conference on Computer-Aided Design*, pp. 132-139, Nov. 2014.

[47] “YUV Video Sequences,” [Online]. Available: http://trace.eas.asu.edu/yuv

[48] Z. Wang, A.C. Bovik, H.R.Sheikh, and E.P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

[49] Weka 3, http://www.cs.waikato.ac.nz/ml/weka/.

[50] Xiph.org Video Test Media, https://media.xiph.org/video/derf/.

[51] W. Yueh, M. Cho, and S. Mukhopadhyay, “Perceptual Quality Preserving SRAM Architecture for Color Motion Pictures,” *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. Grenoble, France, pp. 103-108, 2013.

[52] V. Vijayakumar, and R. Nedunchezhian, “Recent Trends and Research Issues in Video Association Mining,” *The International Journal of Multimedia & Its Applications (IJMA)*, vol. 3, no. 4, pp. 49-61, 2011.

[53] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, “13.8 A 32kb SRAM for error-free and error-tolerant applications with dynamic energy-quality management in 28nm CMOS,” *Proc. Int. Solid-State Circuits Conf.*. San Francisco, CA, USA, pp. 244-246, 2014.

[54] R. Agrawal, R. and Srikant, “Fast Algorithms for Mining Association Rules,” *IBM Almaden Research Center*, 650 Harry Road, San Jose, CA 95120.

[55] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *IBM Almaden Research Center*, 650 Harry Road, San Jose, CA 95120.

[56] I. J. Chang, D. Mohapatra, and K. Roy, "A priority-based 6 T/8 T hybrid SRAM architecture for aggressive voltage scaling in video applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 101-112, 2011.

[57] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, "SRAM for error-tolerant applications with dynamic energy-quality management in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 5, pp. 1310-1323, 2015.

[58] D. Zhou, S. Wang, H. Sun, J. Zhou, J. Zhu, Y. Zhao, J. Zhou, S. Zhang, S. Kimura, T. Yoshimura, and S. Goto, "A 4Gpixel/s 8/10b H.265/HEVC Video Decoder Chip for 8K Ultra HD Applications," Proc. Int. Solid-State Circuits Conf. pp. 266-267, 2016.

[59] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate Computing and the Quest for Computing Efficiency," *Proc. DAC'15*, pp. 120-126, 2015.

[60] A. Hore, and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," *Pattern Recognition (ICPR), 2010 20th International Conference on. Istanbul*, pp. 2366-2369, 2010.

[61] Xiph.org Video Test Media, https://media.xiph.org/video/derf.

[62] Ultra Video Group, ultravideo.cs.tut.fi/#testsequences.

[63] D. Chen, J. Edstrom, X. Chen, W. Jin, J. Wang, and N. Gong, "Data-Driven Low-Cost On-Chip Memory with Adaptive Power-Quality Trade-off for Mobile Video Streaming," *Proc. International Symposium on Low Power Electronics and Design (ISLPED'16)*, San Francisco, CA, 2016.

[64] "Internet of Things: Science Fiction or Business Fact?". *Harvard Business Review, 2014*, retrieved 2016.

[65] H. Jayakumar, A. Raha, and V. Raghunathan, "Energy-Aware Memory Mapping for Hybrid FRAM-SRAM MCUs in IoT Edge Devices," *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 264-269, 2016.

[66] YouTube-8M videos. https://research.google.com/youtube8m/

APPENDIX A. VERILOG CODE USED TO MODEL MEMORY FAILURE

```
//----------------------------fault injection-----------------------------
reg             [data_width-1:0]       fault_active_bit;
wire            [31:0]                 failure_rate_bit [31:0] ;
integer                                          i ;

assign failure_rate_bit[0]  = failure_rate_bit00;
assign failure_rate_bit[1]  = failure_rate_bit01;
assign failure_rate_bit[2]  = failure_rate_bit02;
assign failure_rate_bit[3]  = failure_rate_bit03;
assign failure_rate_bit[4]  = failure_rate_bit04;
assign failure_rate_bit[5]  = failure_rate_bit05;
assign failure_rate_bit[6]  = failure_rate_bit06;
assign failure_rate_bit[7]  = failure_rate_bit07;
assign failure_rate_bit[8]  = failure_rate_bit08;
assign failure_rate_bit[9]  = failure_rate_bit09;
assign failure_rate_bit[10] = failure_rate_bit10;
assign failure_rate_bit[11] = failure_rate_bit11;
assign failure_rate_bit[12] = failure_rate_bit12;
assign failure_rate_bit[13] = failure_rate_bit13;
assign failure_rate_bit[14] = failure_rate_bit14;
assign failure_rate_bit[15] = failure_rate_bit15;
assign failure_rate_bit[16] = failure_rate_bit16;
assign failure_rate_bit[17] = failure_rate_bit17;
assign failure_rate_bit[18] = failure_rate_bit18;
assign failure_rate_bit[19] = failure_rate_bit19;
assign failure_rate_bit[20] = failure_rate_bit20;
assign failure_rate_bit[21] = failure_rate_bit21;
assign failure_rate_bit[22] = failure_rate_bit22;
assign failure_rate_bit[23] = failure_rate_bit23;
assign failure_rate_bit[24] = failure_rate_bit24;
assign failure_rate_bit[25] = failure_rate_bit25;
assign failure_rate_bit[26] = failure_rate_bit26;
assign failure_rate_bit[27] = failure_rate_bit27;
assign failure_rate_bit[28] = failure_rate_bit28;
assign failure_rate_bit[29] = failure_rate_bit29;
assign failure_rate_bit[30] = failure_rate_bit30;
assign failure_rate_bit[31] = failure_rate_bit31;

always @(rd_addr)begin
        for ( i =0 ; i < data_width ; i=i+1)begin
                if(({$random} % 10000) < failure_rate_bit[i])
                        fault_active_bit[i]    =       1'b1;
                else
                        fault_active_bit[i]    =       1'b0;

//              fault_injection_data_out[i] = data_out[i] + fault_active_bit[i];
        end
end

assign  fault_injection_data_out = data_out + fault_active_bit;
//----------------------------fault injection-----------------------------
```

APPENDIX B. VERILOG CODE USED FOR CALCULATING MSE

```
`timescale     1ns/1ns
module MSE_luma_calculator;
reg            clk;
reg            rst;
reg            luma_calculator_valid;
reg            cb_calculator_valid;
reg            cr_calculator_valid;

reg    [31:0]  video_ram0[1:2841696];
reg    [31:0]  video_ram1[1:2841696];
reg    [31:0]  video0_data;
reg    [31:0]  video1_data;
reg    [31:0]  video0_data_1d;
reg    [31:0]  video1_data_1d;
reg    [31:0]  video_ram_addr;
wire   [7:0]   luma_difference_value0;
wire   [7:0]   luma_difference_value1;
wire   [7:0]   luma_difference_value2;
wire   [7:0]   luma_difference_value3;
wire   [7:0]   cb_difference_value0;
wire   [7:0]   cb_difference_value1;
wire   [7:0]   cb_difference_value2;
wire   [7:0]   cb_difference_value3;
wire   [7:0]   cr_difference_value0;
wire   [7:0]   cr_difference_value1;
wire   [7:0]   cr_difference_value2;
wire   [7:0]   cr_difference_value3;
reg    [63:0]  sum_luma;
reg    [63:0]  sum_cb;
reg    [63:0]  sum_cr;
reg    [31:0]  luma_cnt;
reg    [31:0]  cb_cnt;
reg    [31:0]  cr_cnt;
```

```
initial begin
        clk = 1'b1;
        rst = 1'b0;
        #1100 rst = 1'b1;
        #1000 rst = 1'b0;
end

always begin
        #340 clk = ~clk;
end

initial begin
        $readmemh("C:/projects/xilinx/mse_cal/akiyo_original.txt",video_ram0);
        $readmemh("C:/projects/xilinx/mse_cal/akiyo_no_fault.txt",video_ram1);
//      $readmemh("C:/projects/xilinx/mse_cal/test_ori.txt",video_ram0);
//      $readmemh("C:/projects/xilinx/mse_cal/test_inj.txt",video_ram1);
end

always @(posedge clk)begin
        if(rst)
                video_ram_addr <=      32'd1;
        else if(video_ram_addr == 32'd2841696)
                video_ram_addr <=      video_ram_addr;
        else
                video_ram_addr <=      video_ram_addr + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                luma_cnt       <=      32'd0;
        else if(video_ram_addr == 32'd2841696)
                luma_cnt       <=      32'd0;
        else if(luma_cnt == 32'd6336)
                luma_cnt       <=      32'd0;
        else if(cr_cnt == 1584)
                luma_cnt       <=      luma_cnt + 1'b1;
```

```
        else if((luma_cnt > 0) && (luma_cnt < 32'd6336))
                luma_cnt       <=     luma_cnt + 1'b1;
        else if((luma_cnt == 0) && (cb_cnt  == 0) && (cr_cnt  == 0))
                luma_cnt       <=     luma_cnt + 1'b1;
end



always @(posedge clk)begin
        if(rst)
                cb_cnt  <=     32'd0;
        else if(cb_cnt == 32'd1584)
                cb_cnt  <=     32'd0;
        else if(luma_cnt == 32'd6336)
                cb_cnt  <=     cb_cnt + 1'b1;
        else if((cb_cnt > 0) && (cb_cnt < 32'd1584))
                cb_cnt  <=     cb_cnt + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                cr_cnt  <=     32'd0;
        else if(cr_cnt == 32'd1584)
                cr_cnt  <=     32'd0;
        else if(cb_cnt == 1584)
                cr_cnt  <=     cr_cnt + 1'b1;
        else if((cr_cnt > 0) && (cr_cnt < 32'd1584))
                cr_cnt  <=     cr_cnt + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                luma_calculator_valid  <=     1'b0;
        else if(luma_cnt == 0)
                luma_calculator_valid  <=     1'b0;
        else
                luma_calculator_valid  <=     1'b1;
```

```
end

always @(posedge clk)begin
        if(rst)
                cb_calculator_valid    <=      1'b0;
        else if(cb_cnt == 0)
                cb_calculator_valid    <=      1'b0;
        else
                cb_calculator_valid    <=      1'b1;
end

always @(posedge clk)begin
        if(rst)
                cr_calculator_valid    <=      1'b0;
        else if(cr_cnt == 0)
                cr_calculator_valid    <=      1'b0;
        else
                cr_calculator_valid    <=      1'b1;
end

always @(posedge clk)begin
        video0_data <= video_ram0[video_ram_addr];
        video1_data <= video_ram1[video_ram_addr];
        video0_data_1d <= video0_data;
        video1_data_1d <= video1_data;
end
//data_difference calculate

assign  luma_difference_value0        = (      video0_data_1d[7:0] > video1_data_1d[31:24])?
                                                                        (video0_data_1d[7:0] -
video1_data_1d[31:24]) :
                                                                        (video1_data_1d[31:24] -
video0_data_1d[7:0]);
assign  cb_difference_value0   = (      video0_data_1d[7:0] > video1_data_1d[31:24])?
                                                                        (video0_data_1d[7:0] -
video1_data_1d[31:24]) :
```

```
                                                                (video1_data_1d[31:24] -
video0_data_1d[7:0]);
assign  cr_difference_value0   = (     video0_data_1d[7:0] > video1_data_1d[31:24])?
                                                                (video0_data_1d[7:0] -
video1_data_1d[31:24]) :
                                                                (video1_data_1d[31:24] -
video0_data_1d[7:0]);

assign  luma_difference_value1        = (     video0_data_1d[15:8] > video1_data_1d[23:16])?
                                                                (video0_data_1d[15:8] -
video1_data_1d[23:16]) :
                                                                (video1_data_1d[23:16] -
video0_data_1d[15:8]);
assign  cb_difference_value1   = (     video0_data_1d[15:8] > video1_data_1d[23:16])?
                                                                (video0_data_1d[15:8] -
video1_data_1d[23:16]) :
                                                                (video1_data_1d[23:16] -
video0_data_1d[15:8]);
assign  cr_difference_value1   = (     video0_data_1d[15:8] > video1_data_1d[23:16])?
                                                                (video0_data_1d[15:8] -
video1_data_1d[23:16]) :
                                                                (video1_data_1d[23:16] -
video0_data_1d[15:8]);

assign  luma_difference_value2        = (     video0_data_1d[23:16] > video1_data_1d[15:8])?
                                                                (video0_data_1d[23:16] -
video1_data_1d[15:8]) :
                                                                (video1_data_1d[15:8] -
video0_data_1d[23:16]);
assign  cb_difference_value2   = (     video0_data_1d[23:16] > video1_data_1d[15:8])?
                                                                (video0_data_1d[23:16] -
video1_data_1d[15:8]) :
                                                                (video1_data_1d[15:8] -
video0_data_1d[23:16]);
assign  cr_difference_value2   = (     video0_data_1d[23:16] > video1_data_1d[15:8])?
```

```
                                                                (video0_data_1d[23:16] -
video1_data_1d[15:8]) :
                                                                (video1_data_1d[15:8] -
video0_data_1d[23:16]);

assign  luma_difference_value3        = (     video0_data_1d[31:24] > video1_data_1d[7:0])?
                                                                (video0_data_1d[31:24] -
video1_data_1d[7:0]) :
                                                                (video1_data_1d[7:0] -
video0_data_1d[31:24]);
assign  cb_difference_value3   = (     video0_data_1d[31:24] > video1_data_1d[7:0])?
                                                                (video0_data_1d[31:24] -
video1_data_1d[7:0]) :
                                                                (video1_data_1d[7:0] -
video0_data_1d[31:24]);
assign  cr_difference_value3   = (     video0_data_1d[31:24] > video1_data_1d[7:0])?
                                                                (video0_data_1d[31:24] -
video1_data_1d[7:0]) :
                                                                (video1_data_1d[7:0] -
video0_data_1d[31:24]);

//luma mse calculate
always @(posedge clk)begin
        if(rst)
                sum_luma <= 64'd0;
        else if(luma_calculator_valid)
                sum_luma <= (luma_difference_value0 * luma_difference_value0) +
                                        (luma_difference_value1 * luma_difference_value1) +
                                        (luma_difference_value2 * luma_difference_value2) +
                                        (luma_difference_value3 * luma_difference_value3) +
sum_luma;
end
//cb mse calculate
always @(posedge clk)begin
        if(rst)
                sum_cb <= 64'd0;
```

```
        else if(cb_calculator_valid)
                sum_cb <=       (cb_difference_value0 * cb_difference_value0) +
                                        (cb_difference_value1 * cb_difference_value1) +
                                        (cb_difference_value2 * cb_difference_value2) +
                                        (cb_difference_value3 * cb_difference_value3) + sum_cb;
end
//cr mse calculate
always @(posedge clk)begin
        if(rst)
                sum_cr <= 64'd0;
        else if(cr_calculator_valid)
                sum_cr <=       (cr_difference_value0 * cr_difference_value0) +
                                        (cr_difference_value1 * cr_difference_value1) +
                                        (cr_difference_value2 * cr_difference_value2) +
                                        (cr_difference_value3 * cr_difference_value3) + sum_cr;
end



endmodule
```

APPENDIX C. VERILOG CODE USED FOR BIT SWITCHING

```
`timescale     1ns/1ns

module power_consumption;

`define addr_length 32'd6336

reg                    clk;
reg                    rst;

reg                    valid;
reg                    valid_1d;
reg                    valid_2d;
reg                    valid_3d;

//reg   [31:0]  video_ram0[1:9504];
//reg   [31:0]  video_ram1[1:9504];

reg     [31:0]  video_ram0[1:`addr_length];
reg     [31:0]  video_ram1[1:`addr_length];

reg     [31:0]  video0_data;
reg     [31:0]  video1_data;
reg     [31:0]  video_ram_addr;

initial begin
        clk = 1'b1;
        rst = 1'b0;

        #1100 rst = 1'b1;
        #1000 rst = 1'b0;
end

always begin
        #340 clk = ~clk;
```

```
end

//0-0 1-1 1-0 0-1
initial begin
        $readmemh("C:/projects/xilinx/mse_cal11041443/frame1.txt",video_ram0);
        $readmemh("C:/projects/xilinx/mse_cal11041443/frame2.txt",video_ram1);
end

always @(posedge clk)begin
        if(rst)
                video_ram_addr <=      32'd0;
//      else if(video_ram_addr == 32'd9054)
        else if(video_ram_addr == `addr_length)
                video_ram_addr <=      video_ram_addr;
        else
                video_ram_addr <=      video_ram_addr + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                valid   <=      1'b0;
//      else if(video_ram_addr == 32'd9054)
        else if(video_ram_addr == `addr_length)
                valid   <=      1'b0;
        else if(video_ram_addr > 32'd0)
                valid   <=      1'b1;
end

always @(posedge clk)begin
        valid_1d        <=      valid;
        valid_2d        <=      valid_1d;
        valid_3d        <=      valid_2d;
end

always @(posedge clk)begin
        video0_data <= video_ram0[video_ram_addr];
```

```
        video1_data <= video_ram1[video_ram_addr];
end
//data_difference calculate
//************************************20150109*****************************************
//change 0 to 0
reg             [31:0]  bit_0_0 [31:0];
reg             [31:0]  sum_0_0 [7:0];
wire    [31:0]  sum7_0_0,sum6_0_0,sum5_0_0,sum4_0_0,sum3_0_0,sum2_0_0,sum1_0_0,sum0_0_0;
integer                 i ;

always @(posedge clk)begin
        for ( i =0 ; i < 32 ; i=i+1)begin
                if(rst)
                        bit_0_0[i] <= 1'b0;
                else if((video_ram_addr > 32'd0) & (!video0_data[i]) & (!video1_data[i]))
                        bit_0_0[i] <= 1'b1;
                else
                        bit_0_0[i] <= 1'b0;
        end
end

always @(posedge clk)begin
        for ( i =0 ; i < 8 ; i=i+1)begin
                if(rst)
                        sum_0_0[i] <= 32'd0;
                else if(valid | valid_1d | valid_2d)
                        sum_0_0[i] <= bit_0_0[i] + bit_0_0[i+8] + bit_0_0[i+16] + bit_0_0[i+24] +
sum_0_0[i];
                else
                        sum_0_0[i] <= sum_0_0[i];
        end
end

assign          sum7_0_0        =       sum_0_0[7];
assign          sum6_0_0        =       sum_0_0[6];
assign          sum5_0_0        =       sum_0_0[5];
```

```
assign          sum4_0_0        =       sum_0_0[4];
assign          sum3_0_0        =       sum_0_0[3];
assign          sum2_0_0        =       sum_0_0[2];
assign          sum1_0_0        =       sum_0_0[1];
assign          sum0_0_0        =       sum_0_0[0];

//change 0 to 1
reg             [31:0]  bit_0_1 [31:0];
reg             [31:0]  sum_0_1 [7:0];
wire    [31:0]  sum7_0_1,sum6_0_1,sum5_0_1,sum4_0_1,sum3_0_1,sum2_0_1,sum1_0_1,sum0_0_1;

always @(posedge clk)begin
        for ( i =0 ; i < 32 ; i=i+1)begin
                if(rst)
                        bit_0_1[i] <= 1'b0;
                else if((video_ram_addr > 32'd0) & (!video0_data[i]) & video1_data[i])
                        bit_0_1[i] <= 1'b1;
                else
                        bit_0_1[i] <= 1'b0;
        end
end

always @(posedge clk)begin
        for ( i =0 ; i < 8 ; i=i+1)begin
                if(rst)
                        sum_0_1[i] <= 32'd0;
                else if(valid | valid_1d | valid_2d)
                        sum_0_1[i] <= bit_0_1[i] + bit_0_1[i+8] + bit_0_1[i+16] + bit_0_1[i+24] +
sum_0_1[i];
                else
                        sum_0_1[i] <= sum_0_1[i];
        end
end

assign          sum7_0_1        =       sum_0_1[7];
assign          sum6_0_1        =       sum_0_1[6];
```

```
assign          sum5_0_1        =       sum_0_1[5];
assign          sum4_0_1        =       sum_0_1[4];
assign          sum3_0_1        =       sum_0_1[3];
assign          sum2_0_1        =       sum_0_1[2];
assign          sum1_0_1        =       sum_0_1[1];
assign          sum0_0_1        =       sum_0_1[0];

//change 1 to 1
reg             [31:0]  bit_1_1 [31:0];
reg             [31:0]  sum_1_1 [7:0];
wire    [31:0]  sum7_1_1,sum6_1_1,sum5_1_1,sum4_1_1,sum3_1_1,sum2_1_1,sum1_1_1,sum0_1_1;

always @(posedge clk)begin
        for ( i =0 ; i < 32 ; i=i+1)begin
                if(rst)
                        bit_1_1[i] <= 1'b0;
                else if((video_ram_addr > 32'd0) & (video0_data[i]) & (video1_data[i]))
                        bit_1_1[i] <= 1'b1;
                else
                        bit_1_1[i] <= 1'b0;
        end
end

always @(posedge clk)begin
        for ( i =0 ; i < 8 ; i=i+1)begin
                if(rst)
                        sum_1_1[i] <= 32'd0;
                else if(valid | valid_1d | valid_2d)
                        sum_1_1[i] <= bit_1_1[i] + bit_1_1[i+8] + bit_1_1[i+16] + bit_1_1[i+24] +
sum_1_1[i];
                else
                        sum_1_1[i] <= sum_1_1[i];
        end
end

assign          sum7_1_1        =       sum_1_1[7];
```

```
assign          sum6_1_1        =       sum_1_1[6];
assign          sum5_1_1        =       sum_1_1[5];
assign          sum4_1_1        =       sum_1_1[4];
assign          sum3_1_1        =       sum_1_1[3];
assign          sum2_1_1        =       sum_1_1[2];
assign          sum1_1_1        =       sum_1_1[1];
assign          sum0_1_1        =       sum_1_1[0];

//change 1 to 0
reg             [31:0]  bit_1_0 [31:0];
reg             [31:0]  sum_1_0 [7:0];
wire    [31:0]  sum7_1_0,sum6_1_0,sum5_1_0,sum4_1_0,sum3_1_0,sum2_1_0,sum1_1_0,sum0_1_0;

always @(posedge clk)begin
        for ( i =0 ; i < 32 ; i=i+1)begin
                if(rst)
                        bit_1_0[i] <= 1'b0;
                else if((video_ram_addr > 32'd0) & (video0_data[i]) & (!video1_data[i]))
                        bit_1_0[i] <= 1'b1;
                else
                        bit_1_0[i] <= 1'b0;
        end
end

always @(posedge clk)begin
        for ( i =0 ; i < 8 ; i=i+1)begin
                if(rst)
                        sum_1_0[i] <= 32'd0;
                else if(valid | valid_1d | valid_2d)
                        sum_1_0[i] <= bit_1_0[i] + bit_1_0[i+8] + bit_1_0[i+16] + bit_1_0[i+24] +
sum_1_0[i];
                else
                        sum_1_0[i] <= sum_1_0[i];
        end
end
```

```
assign          sum7_1_0        =       sum_1_0[7];
assign          sum6_1_0        =       sum_1_0[6];
assign          sum5_1_0        =       sum_1_0[5];
assign          sum4_1_0        =       sum_1_0[4];
assign          sum3_1_0        =       sum_1_0[3];
assign          sum2_1_0        =       sum_1_0[2];
assign          sum1_1_0        =       sum_1_0[1];
assign          sum0_1_0        =       sum_1_0[0];

wire    [31:0]                  sum_all;
assign          sum_all = sum7_0_0 + sum6_0_0 + sum5_0_0 + sum4_0_0 + sum3_0_0 + sum2_0_0 +
sum1_0_0 + sum0_0_0
                                                + sum7_0_1 + sum6_0_1 + sum5_0_1 + sum4_0_1 +
sum3_0_1 + sum2_0_1 + sum1_0_1 + sum0_0_1
                                                + sum7_1_0 + sum6_1_0 + sum5_1_0 + sum4_1_0 +
sum3_1_0 + sum2_1_0 + sum1_1_0 + sum0_1_0
                                                + sum7_1_1 + sum6_1_1 + sum5_1_1 + sum4_1_1 +
sum3_1_1 + sum2_1_1 + sum1_1_1 + sum0_1_1;

reg                             err;
reg             [31:0]  sum_all_1d;

always @(posedge clk)begin
        sum_all_1d <= sum_all;
end

always @(posedge clk)begin
        if(rst)
                err <= 1'b0;
        else if((video_ram_addr > 32'd3) & ((sum_all - sum_all_1d) !== 32'd32))
                err <= 1'b1;
end

//******************************************************************************

endmodule
```

APPENDIX D. VERILOG CODE USED FOR D-DASH POWER

```
`timescale     1ns/1ns


module prob;


reg                    clk;
reg                    rst;


reg                    valid;
reg     [7:0]   video_ram2[1:19200];
reg     [7:0]   video_ram3[1:19200];
reg     [7:0]   video2_data;
reg     [7:0]   video3_data;
reg     [7:0]   video2_data_1d;
reg     [7:0]   video3_data_1d;
reg     [31:0]  video_ram_addr;
reg     [31:0]  luma_cnt;


initial begin
        clk = 1'b1;
        rst = 1'b0;


        #1100 rst = 1'b1;
        #1000 rst = 1'b0;
end


always begin
        #340 clk = ~clk;
end


initial begin
//      $readmemh("C:/projects/xilinx/islped_test/video_concert1.txt",video_ram2);
        $readmemh("C:/projects/xilinx/islped_test/cr.txt",video_ram2);
        $readmemh("C:/projects/xilinx/islped_test/cb.txt",video_ram3);
end
```

```
always @(posedge clk)begin
        video2_data <= video_ram2[video_ram_addr];
        video2_data_1d <= video2_data;
        video3_data <= video_ram3[video_ram_addr];
        video3_data_1d <= video3_data;
end

always @(posedge clk)begin
        if(rst)
                video_ram_addr <=      32'd1;
        else if(video_ram_addr == 32'd19200)
                video_ram_addr <=      video_ram_addr;
        else
                video_ram_addr <=      video_ram_addr + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                luma_cnt       <=      32'd0;
        else if(luma_cnt == 32'd19200)
                luma_cnt       <=      32'd0;
        else if((luma_cnt > 0) && (luma_cnt < 32'd19200))
                luma_cnt       <=      luma_cnt + 1'b1;
        else if(luma_cnt == 0)
                luma_cnt       <=      luma_cnt + 1'b1;
end

always @(posedge clk)begin
        if(rst)
                valid   <=      1'b0;
        else if(luma_cnt == 0)
                valid   <=      1'b0;
        else
                valid   <=      1'b1;
end
```

```
reg             [31:0]  cr1_cnt;
reg             [31:0]  cr2_cnt;
reg             [31:0]  cr3_cnt;
reg             [31:0]  cr4_cnt;
reg             [31:0]  cr5_cnt;
reg             [31:0]  cr6_cnt;
reg             [31:0]  cr7_cnt;
reg             [31:0]  cr8_cnt;


always @(posedge clk)begin
        if(rst)
                cr1_cnt <= 0;
        else if(valid)
                cr1_cnt <= cr1_cnt + 1;
end

always @(posedge clk)begin
        if(rst)
                cr2_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[6])
                        cr2_cnt <= cr2_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[6]))
                        cr2_cnt <= cr2_cnt + 1;
        end
end

always @(posedge clk)begin
        if(rst)
                cr3_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[5])
                        cr3_cnt <= cr3_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[5]))
```

```
                        cr3_cnt <= cr3_cnt + 1;
        end
end



always @(posedge clk)begin
        if(rst)
                cr4_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[4])
                        cr4_cnt <= cr4_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[4]))
                        cr4_cnt <= cr4_cnt + 1;
        end
end


always @(posedge clk)begin
        if(rst)
                cr5_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[3])
                        cr5_cnt <= cr5_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[3]))
                        cr5_cnt <= cr5_cnt + 1;
        end
end



always @(posedge clk)begin
        if(rst)
                cr6_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[2])
                        cr6_cnt <= cr6_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[2]))
                        cr6_cnt <= cr6_cnt + 1;
```

```
        end
end


always @(posedge clk)begin
        if(rst)
                cr7_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[1])
                        cr7_cnt <= cr7_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[1]))
                        cr7_cnt <= cr7_cnt + 1;
        end
end


always @(posedge clk)begin
        if(rst)
                cr8_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video2_data_1d[0])
                        cr8_cnt <= cr8_cnt + 1;
                else if((~video2_data_1d[7]) & (~video2_data_1d[0]))
                        cr8_cnt <= cr8_cnt + 1;
        end
end


//////////////////////////////
reg             [31:0]  cb1_cnt;
reg             [31:0]  cb2_cnt;
reg             [31:0]  cb3_cnt;
reg             [31:0]  cb4_cnt;
reg             [31:0]  cb5_cnt;
reg             [31:0]  cb6_cnt;
reg             [31:0]  cb7_cnt;
```

```
reg             [31:0]  cb8_cnt;



always @(posedge clk)begin
        if(rst)
                cb1_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[7])
                        cb1_cnt <= cb1_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[7]))
                        cb1_cnt <= cb1_cnt + 1;
        end
end


always @(posedge clk)begin
        if(rst)
                cb2_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[6])
                        cb2_cnt <= cb2_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[6]))
                        cb2_cnt <= cb2_cnt + 1;
        end
end


always @(posedge clk)begin
        if(rst)
                cb3_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[5])
                        cb3_cnt <= cb3_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[5]))
                        cb3_cnt <= cb3_cnt + 1;
        end
end
```

```
always @(posedge clk)begin
        if(rst)
                cb4_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[4])
                        cb4_cnt <= cb4_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[4]))
                        cb4_cnt <= cb4_cnt + 1;
        end
end

always @(posedge clk)begin
        if(rst)
                cb5_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[3])
                        cb5_cnt <= cb5_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[3]))
                        cb5_cnt <= cb5_cnt + 1;
        end
end

always @(posedge clk)begin
        if(rst)
                cb6_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[2])
                        cb6_cnt <= cb6_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[2]))
                        cb6_cnt <= cb6_cnt + 1;
        end
end

always @(posedge clk)begin
        if(rst)
                cb7_cnt <= 0;
```

```
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[1])
                        cb7_cnt <= cb7_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[1]))
                        cb7_cnt <= cb7_cnt + 1;
        end
end

always @(posedge clk)begin
        if(rst)
                cb8_cnt <= 0;
        else if(valid)begin
                if(video2_data_1d[7] & video3_data_1d[0])
                        cb8_cnt <= cb8_cnt + 1;
                else if((~video2_data_1d[7]) & (~video3_data_1d[0]))
                        cb8_cnt <= cb8_cnt + 1;
        end
end

endmodule
```

APPENDIX E. PUBLICATIONS

These works of research resulted in to multiple publications listed below:

1- **Dongliang Chen**, Jonathon Edstrom, Xiaowei Chen, Wei Jin, Jinhui Wang, Na Gong, "*Data-Driven Low-Cost On-Chip Memory with Adaptive Power-Quality Trade-off for Mobile Video Streaming*", International Symposium on Low Power Electronics and Design (ISLPED'16), 2016, ***Nominated for Best Paper Award***.

2- Jonathon Edstrom, **Dongliang Chen**, Jinhui Wang, Mark E. McCourt, Na Gong, "*Luminance Adaptive Smart Video Storage System*", IEEE International Symposium on Circuits and Systems (ISCAS'16), 2016.

3- **Dongliang Chen**, Xin Wang, Jinhui Wang, Na Gong, "*VCAS: Viewing Context Aware Power-Efficient Mobile Video Embedded Memory*", in Proc. IEEE International SoC Conference (SoCC'15), Sep. 2015.

4- **Dongliang Chen**, Jinhui Wang, Na Gong, "*Viewing Context Adaptive On-chip Video Memory*", in Proc. 48th International Symposium on Microelectronics, 2015.

5- N. Gong, Jonathon Edstrom, **Dongliang Chen**, Jinhui Wang, "*Data-Pattern enabled Self-Recovery multimedia storage system for near-threshold computing*", IEEE 34th International Conference on Computer Design (ICCD), 2016.

6- Jonathon Edstrom, **Dongliang Chen**, Yifu Gong, Jinhui Wang, Na Gong, "*Bringing Offline Mining to Online Learning System: Low-Cost and Efficient Self-Healing Synaptic Storage for Deep Learning*", IEEE International Symposium on Circuits & Systems (ISCAS), 2017.

7- Seyed Alireza Pourbakhsh, Xiaowei Chen, **Dongliang Chen**, Xin Wang, Na Gong, Jinhui Wang, "*SPIDER: Sizing-Priority Based Application-Driven Memory for Mobile Video Applications*", In IEEE Transactions on Very Large Scale Integration (TVLSI) Systems, 2017.

8- Seyed Alireza Pourbakhsh, Xiaowei Chen, **Dongliang Chen**, Xin Wang, Na Gong, Jinhui Wang, "*Sizing-Priority Based Low-Power Embedded Memory for Mobile Video Applications*", 17th International Symposium on Quality Electronic Design, (ISQED) 2016, ***Nominated for Best Paper Award***.