



Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
<http://ce.et.tudelft.nl/>

CE-MS-2011-29

M.Sc. Thesis

Hardware Components for Real-Time Stereo Matching: Acceleration of 3D HD TV with FPGAs

Hsiu-Chi Yeh

Abstract

Stereo matching algorithm extracts the depth information by matching the corresponding positions on stereoscopic scenes and then computing the disparity maps. This thesis will focus on implementing a stereo matching computational flow on FPGA. In the proposed stereo matching computation flow, a dynamic programming algorithm is used to enforce the quality of the disparity map. In addition, a simple raw matching cost updating technique is used to strengthen the temporal consistency of disparity map sequences. In the stereo matching hardware design, a hardware-friendly dynamic programming processor unit is proposed, by taking advantage of the Potts model smoothness function, which penalizes disparity changes with a constant penalty. We also present a novel idea for disparity sequences compression by using run-length coding algorithm. The idea is further applied to the memory architecture design for the refinement stage of the stereo matching engine. Finally, the stereo matching engine design is integrated into IMEC 3D TV SoC for depth intensity adjustment. Several peripheral components, are developed in this project, include color space converters, video IO adaptors, and a dedicated memory hierarchy. They all support the stereo matching engine and the view synthesis engine. The proposed SoC is finally verified on EP3SL150 FPGA. The evaluation result shows that it can achieves 60 frames per second at a resolution of 1024 x 768 with an acceptable interpolated video quality.



Delft University of Technology Faculty of Electrical Engineering, Mathematics and Computer Science

Hardware Components for Real-Time Stereo
Matching: Acceleration of 3D HD TV with
FPGAs
Yeh Hsiu-Chi B.Sc.

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

Embedded Systems

by

Hsiu-Chi Yeh
born in Keelung, Taiwan

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
MICROELECTRONICS & COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled ” **Hardware Components for Real-Time Stereo Matching: Acceleration of 3D HD TV with FPGAs**” by **Hsiu-Chi Yeh** in partial fulfillment of the requirements for the degree of **Master of Science**.

Laboratory : Computer Engineering
Codenummer : CE-MS-2012-29

Committee Members :

Advisor: Dr.ir. Gauthier Lafruit, IMEC-Leuven

Advisor: Dr.ir. Georgi Kuzmanov, CE, TU Delft

Chairperson: Dr.ir. Koen Bertels, CE, TU Delft

Member: Dr.ir. Rene van Leuken, CAS, TU Delft

Acknowledgements

This master thesis research work was undertaken at the Department of NVision at IMEC-Leuven with the support from the Computer Engineering group of TU Delft. During the nine months of thesis research work, I received much support from many people. I would like to take this opportunity to express my gratitude to the people around me.

First, I must to thank my promoters, Dr. Gauthier Lafruit and Dr. Francesco Pessolano for providing me this opportunity to join the 3D TV project at IMEC. I must also thank my thesis supervisor, Dr. Georgi Kuzmanov, for his thesis work recommendation. During these nine months, the suggestions from Dr. Gauthier and Dr. Georgi Kuzmanov kept me on the right track of research decisions and provided me with all essential support.

During my nine months of research at IMEC, I was lucky to have a chance to work with a group of brilliant experts. I want to thank Ke Zhange for guiding me into the stereo matching world. Many ideas from Ke inspired me on algorithm implementation and optimization. I thank Geert Vanmeerbeeck and Eddy De Greef for all their support on building the developed environment and providing design suggestions. In this 3D TV project, we also collaborated with IMEC-Taiwans team members. I would like to thank Christine Lin and Josh Tu for all their technical support in building the SoC architecture and test works. I also appreciate the algorithm advice from CK Liao, PK Huang, and Eddy Wu. Further, the view of synthesis kernel design was supported by Professor Chang and NCTU. I would like to thank them all for their contributions. Last but not the least, I would like to especially thank Guanyu Yig. We worked day and night with each other to construct the complete system. He built the first version of the dynamic programming architecture, and my design was inspired and improved by his work. All in all, it has been my honor to work with these people.

Finally, I would like to express my gratitude to my family. Although I live abroad, they keep providing me full support. When I was struggling with the research work, they were always my best mental supporters. I would like to thank also all my friends who have shared all the joys and sorrows of my research life.

Thank You All

Hsiu-Chi Yeh

Delft, The Netherlands
January 2, 2012

Contents

Acknowledgements	iii
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Problem Definition	3
1.3 Solution and Contribution	4
1.4 Overview of Chapters	6
2 Background on Stereo Matching and Related Works	7
2.1 Background of Epipolar Geometry and Image Rectification	8
2.2 Background of Disparity Map Extraction Flow: Local and Global Approaches	8
2.2.1 Matching Cost Computation	9
2.2.2 Matching Cost Aggregation	9
2.2.3 Disparity Computation and Optimization with Dynamic Programming Algorithm	11
2.2.4 Disparity Map Refinement	17
2.3 Temporal Consistency for Disparity Sequence	20
3 Stereo Matching Algorithm Implementation and Optimization on Hardware	21
3.1 Hardware Efficient Dynamic Programming Processor	22
3.1.1 Dynamic Programming Algorithm and Architecture Co-design	22
3.1.2 Dynamic Programming - On-chip Memory Optimization - Backward Path Data Compression	23
3.1.3 Dynamic Programming - On-chip Memory Data Mapping	24
3.1.4 Dynamic Programming Processor - Hardware Architecture	25
3.2 Run-Length Coding Algorithm and Disparity Map Sequence	26
3.3 Temporal Consistency for Disparity Sequence	29
4 Evaluation of the proposed Stereo Matching Hardware	31
4.1 Global Stereo Matching with Dynamic Programming - Disparity Map Evaluation	31
4.1.1 Parameter Exploration for Dynamic Programming Processor	32
4.1.2 Comparison of Local and Global Stereo Matching Approaches	34
4.2 Temporal Quality Evaluation for Disparity Map Sequences	34

4.2.1	Parameter Exploration for Temporal Consistency	34
4.2.2	Evaluation of Temporal Consistency Function	34
4.3	Hardware Resource Estimation of Dynamic Programming Processor . . .	35
4.3.1	Hardware Resource Estimation	35
4.4	Evaluation of the Memory Architecture with Run-length Coding for Ver- tical Voting Processor	37
4.4.1	Parameter Exploration for the Memory Architecture with Run- length Coding	38
4.4.2	Hardware Resource Estimation and Comparison	41
4.5	Hardware Resource Estimation of Stereo Matching Engine on FPGA . . .	41
4.6	Performance Analysis of Stereo Matching Engine on FPGA	42
5	IMEC 3D Depth Intensity Adjustable System with Stereo Matching on FPGA	45
5.1	System Architecture Overview	45
5.1.1	Function Definition	45
5.1.2	Clock Domain Design	45
5.1.3	System On-chip Interconnection	47
5.2	Background of View Synthesis Engine	48
5.3	Video Adaptor Design and Implementation	49
5.4	Color Space Converter Design and Implementation	50
5.4.1	Background of Color Space Conversion	50
5.4.2	Background of Floating Point to Integer Mathematic Approaches .	52
5.4.3	Hardware Architecture Design and Implementations	53
5.5	Memory Hierarchy Design and Implementation	55
5.5.1	Memory Architecture Analysis for Stream Processing	56
5.5.2	Memory Architecture Design for Stream Processing	56
6	IMEC 3D TV SoC Evaluation and Experimental Result	67
6.1	Color Space Converter Design Evaluation	67
6.1.1	Quality Evaluation	67
6.1.2	Hardware Utilization Evalution	69
6.2	Quality Evaluation	70
6.3	Hardware Utilization Estimation	71
6.4	Evaluation of Real-Time Performance	73
7	Conclusion and Future Works	77
7.1	Conclusion	77
7.2	Summary of Chapters and Contributions	78
7.3	Future Work and Application Development	80
	Bibliography	86

List of Figures

1.1	Focus position	2
1.2	Depth perception according to visual comfort	2
1.3	Depth adjustment processing flow	3
1.4	Ghost effect and inaccurate disparity map	4
1.5	disparity map sequence without temporal consistency	4
2.1	Example of disparity maps	7
2.2	Stereo image pair rectification	8
2.3	Disparity map extraction flow	8
2.4	Census transform and hamming distance to generate raw matching cost .	10
2.5	Matching cost aggregation approaches	12
2.6	Example of matching corresponding pixel	12
2.7	Example of smoothness cost penlties by truncated linear model	14
2.8	Example of smoothness cost penlties by linear model	14
2.9	Example of disparity map by linear model	15
2.10	Example of disparity map by potts model	15
2.11	Example of smoothness cost penlty by Potts model	15
2.12	Example of forward pass function	16
2.13	Example of backward pass procedure	17
2.14	Example of occlusion problem	18
2.15	Simple occlusion handling method	19
2.16	2D disparity voting	19
2.17	Two-pass 1D disparity voting	19
3.1	System architecture of stereo matching engine	21
3.2	Forward pass with backward path encoding	24
3.3	Example of backward pass with decoded path data	25
3.4	DP operation sequence	25
3.5	Example of 2-port RAM access patterns	26
3.6	Proposed Forward Pass HW design	26
3.7	Proposed Backward Pass HW design	26
3.8	Disparity output from dynamic programming function	27
3.9	Circular memory architecture for Vertical Diaprity Voting processor . . .	28
3.10	Proposed memory architecture with run-length coding	28
3.11	Example of temporal consistency algorithm in dynamic programming . . .	30
4.1	Parameter exploration for Dynamic Programming	33
4.2	Evaluation results from Middleburry’s benchmark	33
4.3	Ground truth disparity maps and test disparity maps	33
4.4	Exploration of matching cost scaling parameter α with Book Arrival test set	35
4.5	Temporal consistency empirical evaluation	36

4.6	PSNR improvement of temporal consistency	37
4.7	RTL circuit gate count synthesis for Dynamic Programming Processor . .	37
4.8	On-chip memory utilization estimation for Dynamic Programming Processor	37
4.9	Exploration of truncated line buffer length and pixel error rate	38
4.10	Disparity results by using the proposed memory architecture with RLC .	39
4.11	Explore the range of run length counter (Outdoor)	40
4.12	Explore the range of run length counter (Cones)	40
5.1	Dual DVI receivers scenario	46
5.2	Avalon Interface	48
5.3	High level architecture of View Synthesis Engine	49
5.4	Standard VGA signal format	49
5.5	Example of RGB and YCbCr Format	50
5.6	Color space conversion flow	53
5.7	RGB to YCbCr Processor Unit	54
5.8	Constant multiplier implementation (77) with Shift and Add/Sub archi- tecture	54
5.9	Upper 4 bit nibble and lower 4 bit nibble multiplication	55
5.10	Example of Look Up Table	55
5.11	8x8 Constant multiplier with 4x8 LUTs	55
5.12	Proposed Memory Hierarchy	57
5.13	5x5 slide window operation	58
5.14	Example of data reuse	58
5.15	Proposed memory hierarchy for frame buffering	59
5.16	SG-DMA architecture for write function	60
5.17	Example of data concatenation	60
5.18	SG-DMA architecture for read function	61
5.19	The interface between SG-DMA and Arbiter	62
5.20	Handshaking protocol between SG-DMA and Arbiter	62
5.21	Avalon stream interface between processing unit and SG-DMA	63
5.22	Example of address generation pattern	65
6.1	Interpolated video evaluation structure for our system	71
6.2	Interpolated video evaluation structure for DERS+VSRS	72
6.3	Quality evaluation for Book Arrival	72
6.4	Anaglyph outputs for different depth intensity	73
6.5	Example of latencies during burst reading	74
6.6	Memory hierarchy evaluation environment	75

List of Tables

2.1	Definition of absolute difference and square difference	10
2.2	Definition of area-based matching cost functions	11
2.3	Common used smoothness function models	14
4.1	Hardware resource comparison of run length counter and compression rate	41
4.2	On-chip memory architecture resource utilization analysis	41
4.3	Hardware resource analysis of optimized stereo matching engine	42
4.4	comparison of state-of-art stereo matching implementations	43
5.1	Function definition in SoC	47
5.2	Common used YCbCr formats based on A:B:C notation	52
5.3	System memory breakdown	57
6.1	Test sets	68
6.2	PSNR evaluation for different color space conversion standards	68
6.3	scale resolutions of integer approximation from four bit (256) to ten bit (1024)	69
6.4	Rounding approach evaluation	69
6.5	Hardware resource comparison of Full LUT Size approach and 2 LUT Size + Adder approach	70
6.6	Hardware utilization analysis of RGB to YCbCr converter	70
6.7	Hardware resource utilization summary on EP3SL150 FPGA	72
6.8	Throughput estimation based on standard VGA video source	74
6.9	Burst length settings and critical latency analysis	75

Introduction

This thesis presents the researches about stereo matching algorithms and hardware implementation for a 3D TV System. The stereo matching algorithm refers to extracting depth information from two stereoscopic camera sources. The depth information, disparity value, is calculated from estimating the displacement of corresponding points on both images. The computed disparity maps can be used in the IMEC proposed depth adjustable 3D TV system for synthesizing intermediate points of views.

1.1 Motivation

Nowadays, 3D display technologies have become prevalent in many applications. For example, the number of movies with 3D content is growing dramatically. Besides, household entertainments such as 3D TV, 3D broadcast services and stereo cameras, etc. are gradually entering the consumer market. It is foreseeable that 3D display applications will become deeply embedded in our life in the future. Therefore, this thesis research especially interests in depth information extraction technology.

Binocular vision through our eyes is the basic way that our brain perceps object depth in real world. Because our eyes receive slightly different views, our brain can analyze the disparities and fuse the 2D information to a 3D perception. Based on the concept of binocular vision, stereoscopic video contents are generally used for 3D display.

Unfortunately, 3D contents are usually incompatible with different display platforms. For example, the 3D contents for movie theaters are not always available for mobile phones. Besides, it is frequently reported that audiences easily suffer from nausea, eye strain or headache after watching 3D contents. One of the reasons of discomfort is the focus mismatch of 3D objects [48]. Figure 1.1 is an example that shows different scenarios that focus on the cube. In Scenario (1) Accommodation, the focus point of the cube concentrates on a positive region because eye-sight is nearly parallel. When the focus of object locates on a positive region, it provides best visual comfort to the viewer. In Scenario (2) Convergence, the focus point of the cube is in front of a screen, which is too close to the viewer's face. Discomfort is felt because the viewer's eye muscles are in tension. In Scenario (3) Divergence, the focus point of the cube locates far behind the screen because the focus objects are too far apart that causes eyes diverging.

Figure 1.2 is an example that illustrates the personal depth comfort regions in a green zone. In general, the comfort zone depends on different viewers. For example, the eye distance of adult is around 6.25 cm. However, the 3D contents are usually not always applicable for children. Moreover, the comfort zone relates to screen size, view dis-

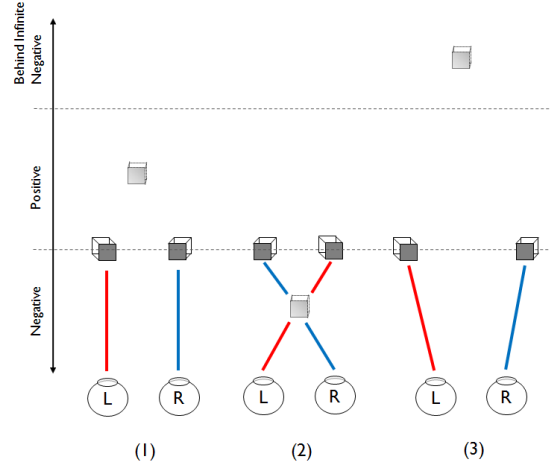


Figure 1.1: Focus position (1) Accommodation (2) Convergence (3) Divergence.

tance and display technology. It is definitely a great challenge for the 3D content vendors.

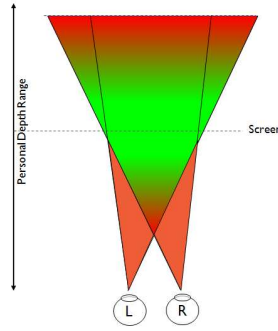


Figure 1.2: Depth perception according to visual comfort.

Addressing to the visual comfort, depth scaling [48] is a solution to the above-mentioned problem. If the viewer observes from the in-between virtual view, heshe will perceive less depth intensity of 3D content. Therefore, IMEC provides a depth scaling solution based on this concept. The processing flow Figure 1.3 calculates the standard stereoscopic video sources captured from left and right cameras and extracts the disparity maps by Stereo Matching Engine. With the disparity maps and input stereo images, View Synthesis Engine is able to generate the in-between virtual view. Therefore, a viewer can choose an arbitrary 3D depth intensity through IMEC 3D TV system.

We are motivated to develop the IMEC 3D TV system on SoC. The proposed system contains Video Adaptors, Color Space Converters, Stereo Matching Engine, memory hierarchy, and a Viewpoint Synthesis Engine. This thesis research and design works mainly concentrate on constructing a stereoscopic processing architecture to support

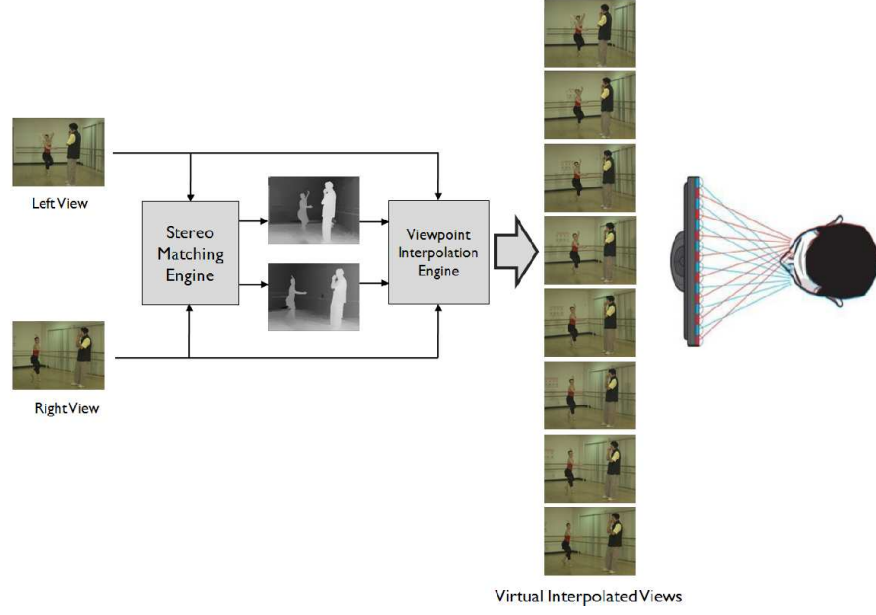


Figure 1.3: Depth adjustment processing flow with stereo matching and viewpoint synthesis engine.

depth extraction for view synthesis in real-time performance. Currently, the SoC has been verified on FPGA and readied for ASIC mapping.

1.2 Problem Definition

The stereo matching algorithms have been researched by the stereo matching community for many years. The most difficult handling regions are occlusion, texture less, and repetitive pattern. In the IMEC 3D TV system, generating quality disparity map is required because it affects the accuracy of the synthesized virtual view. Previous study has shown that humans are more sensitive to the edge area than the plain region. The incorrect disparity map easily generates the so called ghost effect in the synthesized view. Figure 1.4 demonstrates the synthesis error that is introduced from mismatching disparity map.

Generally, the disparity sequences suffer from a temporal inconsistency problem. Figure 1.5 demonstrates an example of inconsistent disparity map sequences. Because most stereo matching algorithms only take individual frames into the computation flow, there is a lack of links among the disparity map sequences. The disparity map sequences are easily influenced by things like camera noise, disparity map mismatching, occlusion region, luminance difference, etc. Unfortunately, a previous study [53] showed that humans are most sensitive to the inconsistent pattern (temporal noise) that flickers in 10 to 20 cycles per second. It also reports that temporal noise is even more obvious than spatial noise.

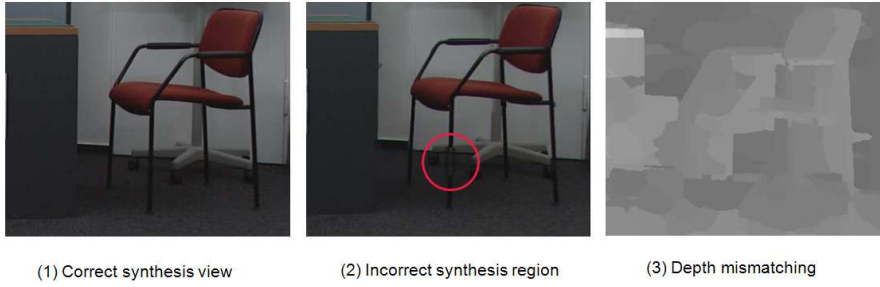


Figure 1.4: Ghost effect and inaccurate disparity map

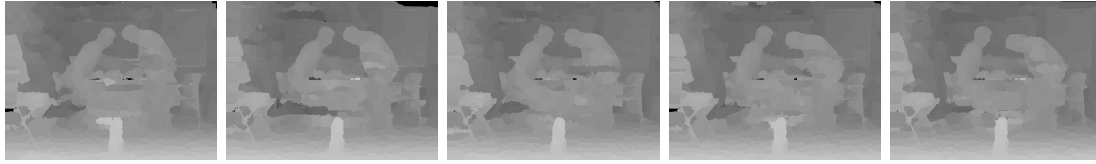


Figure 1.5: disparity map sequence without temporal consistency

Hardware overhead is a problem when implementing a dynamic programming algorithm on FPGA/ASIC. In this thesis, we chose dynamic programming algorithm to implement the global optimization function in stereo matching computation flow. Although the quality of the disparity map shows outstanding performance in Middlebury benchmark [43], tremendous hardware utilization is required for SoC implementation. If we apply scanline-based dynamic programming algorithm in stereo matching computation flow, it will require $O(W \cdot D_{max}^2)$ computational complexity and $(W \cdot D_{max} \cdot D_{bit})$ memory space for the computation. The term W represents the image width, and D_{max} represents the maximum disparity range. It is preferable to keep the memory on-chip because off-chip memory introduces extra throughput and cost problems. Therefore, algorithm selection for resource optimization is one of the goals in this thesis work.

1.3 Solution and Contribution

Two algorithms are applied on the design of the Stereo Matching Engine to improve the quality of the disparity map. In order to achieve a high quality in the disparity map, Dynamic programming algorithm [16] is introduced into stereo matching computation flow for global optimization. The smoothness assumption with the Potts model reserves the disparity discontinuity and also covers the texture-less region that can't be handled in a local algorithm. Then a left-and-right-check algorithm [16] and a cross-based algorithm [49] are used for disparity map refinement. The stereo matching flow shows that the above-average quality of the disparity map. According to Middlebury's benchmark [43], a 6.6% average pixel error rate is achieved based on the evaluation of four frequently used test sets. In this thesis, a couple of algorithm improvement proposals are further introduced into the stereo matching computation flow:

- A simple temporal consistency method is used to enhance the disparity map sequences by adjusting the matching raw cost based on the previous disparity map and image (luminance) information. After integrating with dynamic programming and refinement stage, the stability and quality of the disparity map sequence can be greatly improved.

Furthermore, a hardware efficient stereo matching architecture with dynamic programming algorithm is presented. In addition, a couple of memory optimization methods are further proposed to improve the hardware resource requirement:

- We choose Potts model as the smoothness function for the dynamic programming algorithm. By rewriting the energy minimize function, the computational complexity can be reduced from $O(W \cdot D_{max}^2)$ to $O(W \cdot D_{max})$, and the memory consumption is reduced from $(W \cdot D_{max} \cdot D_{bit})$ to $(W \cdot (D_{max} + D_{bit}))$. The term W represents the image width, and D_{max} represents the maximum disparity range. We also propose a hardware efficient memory architecture by using a single 2-Port BRAM with a sophisticated memory mapping mechanism instead of the conventional ping-pong BRAM architecture. All in all, the on-chip memory can be reduced 11 times without quality loss.
- Run-length coding algorithm is first proposed on disparity map data compression. In this thesis, it is applied on reducing the on-chip memory consumption of post processor. Our experiments shows that the compression rate can reach above 12 times with almost no loss of quality. In the proposed new memory architecture with run-length coding encoder/decoder, it achieves a 4.75 times of memory reduction rate. The high compression rate of run-length the disparity map by run-length coding shows a promising solution for disparity map compression.

To construct IMEC 3D TV SoC, we design and implement extra peripheral components to support Stereo Matching and View Synthesis Engines. Those components are encapsulated in IMEC 3D TV SoC. More specifically, they are:

- A dedicated memory hierarchy is proposed and implemented to support frame buffering for temporal consistency function. The memory hierarchy successfully cooperates with stream processors by using pre-fetch and data burst techniques.
- Color space converters are designed to perform RGB-YCbCr and YCbCr-RGB conversions for the SoC design. Two hardware efficient designs are compared and evaluated from quality, hardware consumption, and flexibility aspects.
- Video signal adaptors are designed for input sequences synchronization, memory data extraction, and output display signal generation based on VGA standards.

Finally, the system integration work is prototyped and evaluated on EP3SL150 FPGA. We integrate the designs including video adaptors, color space converters, stereo matching engine, and memory hierarchy with a view synthesis engine provided by the

research group from NCTU (National Chiao Tung University). So far, the 3D TV prototype is capable of processing up to XGA format (1024x768@60FPS 65MHz Pixel Rate) of video streams in real-time performance. Users can adjust the depth intensity of 3D contents through on-board buttons and can watch anaglyph or synthesized stereoscopic video from 2D or 3D TV through IMEC 3D TV solution.

1.4 Overview of Chapters

The following chapters discuss this thesis works from algorithm selection to hardware implementation. The design works cover from individual components to integrated system.

In Chapter 2, the related background of stereo matching is introduced.

In Chapter 3, a depth extraction flow is presented. Based on the depth extraction flow, we evaluate the performance and hardware complexity. Then, several techniques and proposals are introduced to optimize the disparity map sequences quality and hardware usage.

The proposals in Chapter 3 are evaluated in Chapter 4. The designs are estimated based on three aspects: image/video quality, hardware utilization, and real-time performance.

In Chapter 5, we apply the Stereo Matching Engine design on IMEC 3D TV SoC. The overview of the system architecture will first be introduced. Then we will mention about the supporting peripherals, including memory hierarchy, color space converters, and video signal adaptors.

In Chapter 6, we evaluate the proposed IMEC 3D TV SoC design for EP3SL150 FPGA platform.

In the final chapter, the thesis work is summarized. In addition, several suggestions for future work and relative applications will be mentioned.

Background on Stereo Matching and Related Works

2

Stereo matching algorithm extracts the depth information from stereoscopic image pairs. The depth information refers to the displacement of corresponding points on the other image. In general, the displacement is quantified to the disparity value which is located in a so-called disparity range. Finally, the disparity map is constructed from the full-frame pixel number of disparity value. Figure 2.1 shows the disparity maps that are rendered from stereoscopic image pairs and displayed in gray scale. With the disparity map, it is possible to reconstruct a 3D scene by using triangulation method.

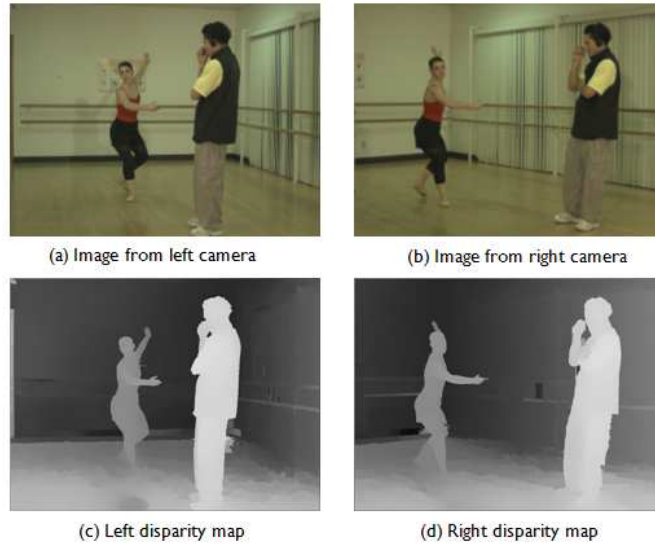


Figure 2.1: Example of disparity maps

There are two main classes of correspondence matching algorithms: feature-based and correlation-based. A feature-based approach refers to searching correspondence by matching sparse sets of image features. The image features are usually derived from feature-identified methods such as edge detection. The other approach, correlation-based, refers to searching for the best correlation pixel on the target image by matching the homologous pixel within disparity range. The matching image intensities are usually derived from a window of pixels. This thesis will only focus on a correlation-based approach because of its robustness and simplicity for real-time hardware implementation. In Section 2.1, the concept of Epipolar geometry is introduced as the background to a correlation-based approach. Section 2.2 sums up the correlation-based stereo matching algorithm for both local and global stereo matching approaches. In Section 2.3, we conclude with some related researches about the stereo matching for sequences.

2.1 Background of Epipolar Geometry and Image Rectification

Image rectification is an important step to simplify the stereo matching space from two dimensions to one dimension searching space. Because the stereo video is not always taken from the well horizontally-aligned cameras, as Figure 2.2 shows, the mapping work is based on the concept of Epipolar line geometry [15][2]. In the example, the upper two images are not taken from two perfectly aligned cameras. The yellow Epipolar lines of the left image are mapped to different positions on the right image. Therefore, the work of image rectification [42][32][6] uses the given intrinsic and extrinsic camera parameters which includes camera rotation, translation, and rotation information to align the Epipolar lines of two projected points on the new image planes.

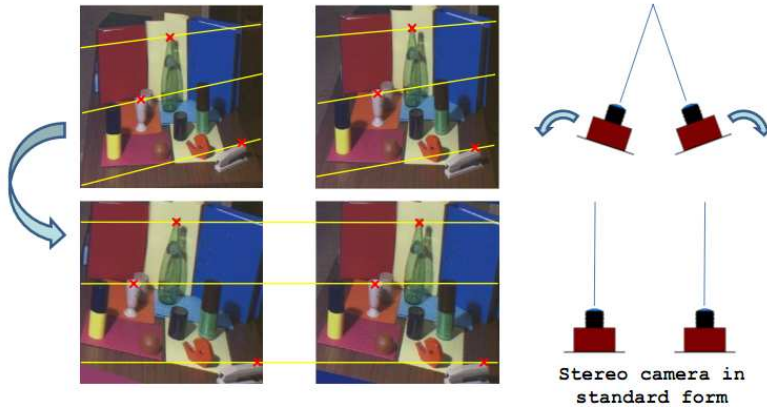


Figure 2.2: Stereo image pair rectification [29]

2.2 Background of Disparity Map Extraction Flow: Local and Global Approaches

This section introduces the disparity map extracting work flow in general correlation-based approaches. Referring to Scharstein and Szeliski's taxonomy [13], a stereo algorithm generally includes four steps that are briefly summarized in Figure 2.3.

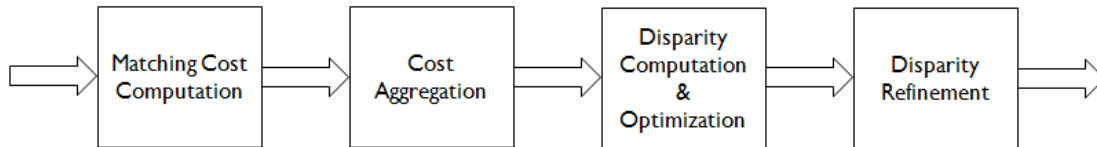


Figure 2.3: Disparity map extraction flow

PS. Local disparity computation algorithm generally performs all of steps but without global optimization in order to obtain accurate disparity map. Global disparity

optimization algorithm sometimes ignore step 2 (cost aggregation) because the algorithm itself covers enough neighborhood pixel information.

The local stereo matching algorithm performs the matching computation only within a finite disparity range. In the matching cost computation step, the matching cost can be calculated from window-based matching algorithms such as sum-of-squared-difference (SSD), sum-of-absolute-difference (SAD), or census-transform (CT) [33], since the matching cost reflects the accuracy of the disparity map. Cost aggregation, which sums a region of matching cost, includes more information around its neighbor pixels. It is generally used in local algorithms. After the matching costs are derived, the disparities are computed by the winner-take-all (WTA) strategy [13]. The WTA strategy selects the displacement position which possesses minimal matching cost value. The winner displacement is regarded as the disparity value. Finally, disparity refinement techniques such as consistency check and disparity voting are implemented to increase the accuracy of disparity map. All in all, the local algorithm normally relies on enforcing the matching cost computation and cost aggregation stages.

The global algorithm optimizes the WTA strategy with an energy function which introduces a smoothness assumption. With the smoothness assumption, global optimization can handle the texture-less regions that local algorithms can't handle. However, it requires more hardware resource than the local algorithm. The state-of-art global algorithms which are based on energy minimization include graph cuts [5], belief propagation [35], and dynamic programming [16]. In this thesis, we choose scanline-based dynamic programming approach to implement the global optimization function.

2.2.1 Matching Cost Computation

The simplest matching cost computation is pixel-based matching cost. The pixel-based matching techniques refer to the two most common methods: absolute difference (AD) and squared difference (SD). Table 2.1 lists the definition of AD and SD [40]. However, the raw matching cost generation only involves a single pixel of information on stereo image pair. To enhance the pixel-based matching cost approach, area-based matching cost approaches [23] are proposed to improve the problems that pixel-based approaches have.

2.2.2 Matching Cost Aggregation

The cost aggregation step aggregates raw matching cost within a certain shape of the neighborhood region in order to increase the matching accuracy. In area-based matching cost approaches, the most commonly used techniques are Sum of Absolute Intensity Difference (SAD), Sum of Square Difference (SSD), Normalized Cross Correlation (NCC), Rank, and Census Transform. The functions are summarized in Table 2.2. In recent years, the Census Transform approach has become popular in the stereo matching community because of its robust performance. The previous research work from Chang [7], Heiko [14], Bleyer [27] show Census Transform performs outstandingly

Table 2.1: Definition of absolute difference and square difference

<i>Matching Cost Alg.</i>	<i>Description</i>	<i>Definition</i>
AD	Absolute difference approach aggregates the color(luminance) difference of reference pixels and target candidate pixels.	$f(u, v, d) = I_{ref}(x, y) - I_{tar}(x + d, y) $
SD	Square difference approach squares and aggregates the difference of reference pixels and target candidate pixels.	$f(u, v, d) = (I_{ref}(x, y) - I_{tar}(x + d, y))^2$

against other approaches in both local and global stereo matching algorithms. The advantage of census-transform is that it only includes the information about the luminance relationship of the central pixel and neighboring pixels. Since no luminance difference value is involved in cost generation, it is robust to the luminance and gamma variations of stereo sources.

In the case of Census Transform matching cost function, the raw matching cost is generated from the hamming distance of the reference pixel and the target pixel of stereo image pairs. 2.4 is an example that shows the distance (raw cost) of two census series. The hamming distance of 01001001 and 00001111 is 3 because 3 bits are unmatched. The higher the raw matching cost represents the lower similarity between reference pixel and candidate matching pixel.

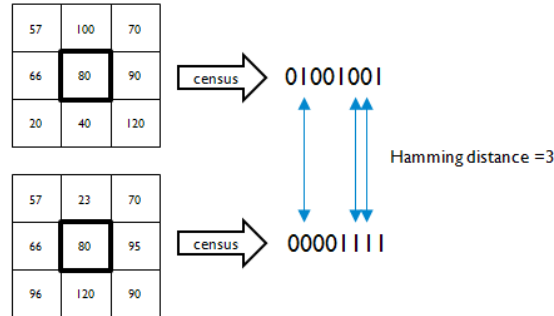


Figure 2.4: Census transform and hamming distance to generate raw matching cost

In advance, Figure 2.5 shows different cost aggregation region strategies, from coarse to fine [50]: fixed window, multiple window, adaptive shape, and adaptive weight. The fixed window approach is the simplest method. Although it possesses low computation complexity, it performs badly on boundary, slant surface, and repetitive pattern regions. The multiple windows approach [29] enhances the fixed window. A number of sub-windows are predefined to make the support region, which is not only constrained to rectangular shape. Another approach is the adaptive shape method. It partitions

Table 2.2: Definition of area-based matching cost functions

Matching Cost Alg	Description	Definition $f(u, v, d)$
SAD	Sum of Absolute Difference sums up the absolute differences in the corresponding region of pixels. (such as square window)	$\sum_{(x,y) \in B(u,v)} I_{ref}(x, y) - I_{tar}(x + d, y) $
SSD	Sum of Square Difference squares and aggregates the differences in the corresponding region of pixels. (such as square window).	$\sum_{(x,y) \in B(u,v)} (I_{ref}(x, y) - I_{tar}(x + d, y))^2$
NCC	Normalized Cross Correlation. The cross correlation is normalized by the mean value in the block. Higher NCC stands better match.	$\frac{\sum_{(x,y) \in B(u,v)} (I_{ref}(x, y) - \mu_{tar}(x + d, y)) \cdot (I_{tar}(x + d, y) - \mu_{tar}(x + d, y))}{\sqrt{\sum_{(x,y) \in B(u,v)} (I_{ref}(x, y) - \mu_{tar}(x + d, y))^2 \cdot \sum_{(x,y) \in B(u,v)} (I_{tar}(x + d, y) - \mu_{tar}(x + d, y))^2}}$
Rank	Rank transform calculates the number of neighbor pixels which have the value larger than the central pixel. The matching cost is calculated from the absolute difference of the two ranks.	$\sum_{(x,y) \in B(u,v)} Rank_{ref}(x, y) - Rank_{tar}(x + d, y) $ $Rank(u, v) = \sum_{(i,j) \in R(u,v)} L(i, j)$ $L(i, j) = \begin{cases} 0, & I(i, j) > I(u, v) \\ 1, & I(i, j) \leq I(u, v) \end{cases}$
Census	Census Transform encodes the comparison result of central pixel and neighbor pixels (window) into a bit string. The matching cost is calculated from the hamming distance of census bit string of corresponding matching candidate.	$f(u, v, d) = \sum_{(x,y) \in B(u,v)} Hamming(Census_{ref}(x, y) - Census_{tar}(x + d, y))$ $Census(u, v) = Bitstring_{(i,j) \in R(u,v)} (I(i, j) \geq I(u, v))$

the image in regions with similar color intensity and aggregates costs within the similar segmentation. Cross-based approach [49] is one example of adaptive shape methods that is proposed by Zhang. The last but the most accurate one is adaptive weight [22] [17]. It assumes that the nearest pixels with similar intensity to the central pixel share the same disparity value. Based on the assumption, the weight of the intensity difference to the central pixel is gradient based on distance. By using adaptive weight method, it helps achieve highest matching performance comparing to other methods. However, the computational complexity is obviously higher than others.

2.2.3 Disparity Computation and Optimization with Dynamic Programming Algorithm

There are two categories of disparity computation approaches: local stereo matching and global optimization method. Both approaches have pros and cons on computational

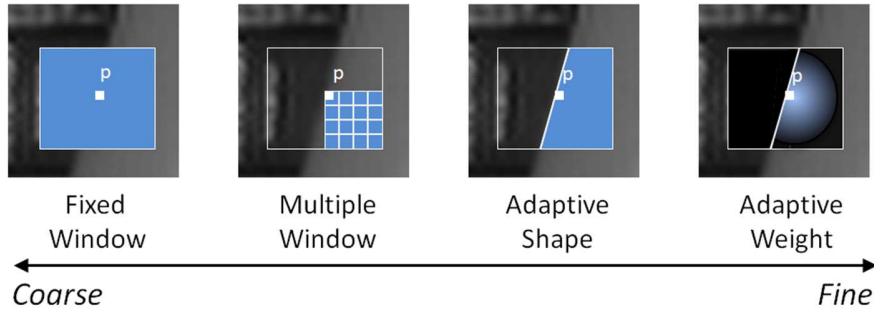


Figure 2.5: Categories matching cost aggregation approaches from coarse to fine.

resource and matching quality.

[Local Stereo Matching] The local stereo matching method computes the disparity value by selecting the disparity candidate which possesses the minimum raw cost value. This method is so-called local winner-take-all (WTA) strategy [13]. It selects the corresponding point (disparity) that has minimum raw cost. The raw costs are generated from the reference pixel and the matching target corresponding candidate pixels on the other image, using the matching cost generation/aggregation methods that were mentioned in the previous section. Then the disparity value is chosen from the disparity candidate that possesses the minimum raw cost. The distance between the reference pixel and the selected target pixel is regarded as a disparity value. Figure 2.6 is an example that illustrates matching one pixel from right image to the left image on an Epipolar line.

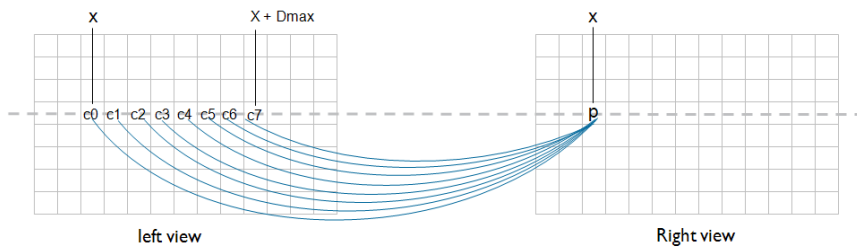


Figure 2.6: Example of matching corresponding pixel

[Global Optimization]

Energy Function

The global optimization algorithm computes the disparity with energy function which introduces the smoothness assumption for global optimization. An additional constraint is added to supports smoothness by penalizing changes of neighboring disparities. Therefore, the smoothness constraints can be treated as an energy minimization problem. The energy function is typically defined as 2.1.

$$E(d) = E_{data}(d) + E_{smooth}(d) \quad (2.1)$$

where $d \in [0, D_{max} - 1]$

$$E_{data}(d) = \sum_{j \in N} C(j, d) \quad (2.2)$$

where $C(\cdot)$ is matching cost function

$$E_{smooth}(d) = \sum_{j \in N, d' \in [0, D_{max}-1]} \lambda \cdot S(d, d') \quad (2.3)$$

where $S(\cdot)$ is smoothness function. λ is a scaling coefficient, which adapts to the luminance variation of adjacent pixels. It provides sharper depth discontinuity when encountering edge regions.

The first term represents the sum of matching costs for each disparity. The second term is a smoothness function, which generates a penalty value to smoothness assumption based on the disparity distance in adjacent pixels. λ is a scaling parameter which adapts to the luminance variation of adjacent pixels in order to provide sharper disparity discontinuity. The smoothness function models will be further introduced in the next paragraph. The energy function is widely used in global optimization approaches such as Dynamic Programming [16], Graph Cut [5], and Belief Propagation [35]. In this thesis, we will focus on scanline dynamic programming algorithm because of its hardware-friendly trait.

The second term, smoothness cost function, produces a penalty value based on the distance of d and d' pixels' disparities. d and d' are adjacent pixels. The idea is to impose cost penalty on disparity variation in order to increase the smoothness of disparity map. The higher penalty value reduces the chance of the disparity candidate to be chosen in winner take all(WTA) step, and vice versa. Thus this increases the chance that the disparity value of d pixel remains the same as d' pixel. In the following, we will conclude the most common used smoothness function models.

In the linear model, the higher distance between the disparities of pixel d and d' generate a higher smoothness penalty. Figure 2.8 illustrates the gradient penalty costs are added on different disparity positions. This leads the disparity to be changed gradually only within small steps. The linear model performs outstanding in slanted surfaces. However, it performs badly in the disparity continuity regions (edge). Figure 2.9 is an example shows the blurred edge in the disparity discontinuity region. To solve the weakness of the linear model, a truncated constant penalty value k is introduced to improve the linear model. The penalties of high distance between the disparities of d and d' are limited to k in order to preserve discontinuous disparity regions (Figure 2.7).

Potts model preserves the discontinuous edges in disparity map by only imposing cost penalty on disparity change. Figure 2.11 illustrates the cost penalties are added

Table 2.3: Common used smoothness function models

Model Name	Smoothness Function	Computational Complexity
Linear Model	$S(d, d') = d - d' $	$O(D^2)$
Truncated Linear Model	$S(d, d') = \min(d - d' , k)$ where k is user-defined truncation constant	$O(D^2 + D)$
Potts Model	$S(d, d') = \begin{cases} 0, & \text{if } d = d' \\ C, & \text{if } d \neq d' \end{cases}$ where C is a constant introduces smoothness cost penalties	$O(D)$
Modified Potts Model	$S(d, d') = \begin{cases} 0, & \text{if } d = d' \\ C_1, & \text{if } d - d' = 1 \\ C_2, & \text{if } \text{otherwise} \end{cases}$ where C_1 and C_2 introduce constant smoothness cost penalties, and C_1 is lower than C_2	$O(2D)$
Second(Higher) Order Model	$S(p, q, r) = d_p - 2d_q + d_r $ where q and r are p 's left and right neighborhood pixels	$O(D^3)$
Truncated Second Order Model	$S(p, q, r) = \min(d_p - 2d_q + d_r , k)$ where k is user-defined truncation constant	$O(D^3 + D)$

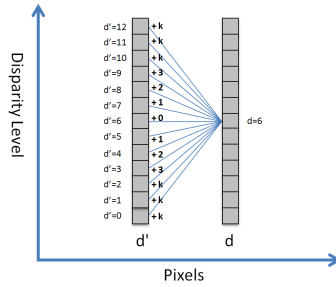


Figure 2.7: Example of smoothness cost penalties by truncated linear model

on the disparity dissimilar positions in energy function. Figure 2.10 is an example of

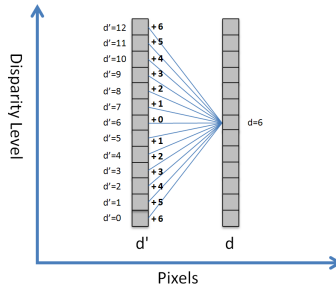


Figure 2.8: Example of smoothness cost penalties by linear model



Figure 2.9: Linear Model



Figure 2.10: Potts Model

a Potts model which performs superior in disparity discontinuous regions than linear model. However, the Potts model performs poorly in reconstructing slanted surfaces. To solve this weakness of Potts model, a modified Potts model introduces a lower cost penalty to trivial dissimilarity of disparities. The modification not only allows the Potts model to handle slight slanted surfaces but also permits disparity discontinuous regions.

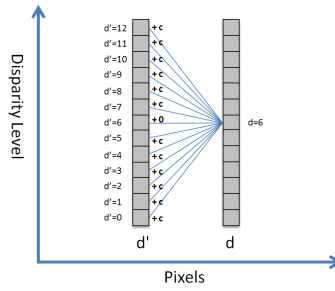


Figure 2.11: Example of smoothness cost penalty by Potts model

The Second (higher) order model is proposed by Woodford [46], which improves the disparity map in curvature surface regions. The second order model detects slanted planes and assigns lower cost penalty to them. Besides, the truncated concept can also be applied to this model. In recent researches, this possesses the state-of-the-art disparity quality among the above mentioned models. However, the computation complexity is relatively complex than the first order model.

Dynamic Programming

Dynamic programming (DP) solves the complex problem in a coarse-to-fine manner. Since stereo matching problems can be formulated as searching the optimal disparity values within 2D image or 1D (i.e. along image row) through minimizing the energy function $E(D)$, dynamic programming is proposed to solve the stereo matching problem. However, 2D image energy minimization problem is much more complicated than 1D energy minimization problem [41] [4]. Therefore, dynamic programming in stereo matching commonly refers to the 1D scanline energy minimization problem.

After the cost aggregation step, each pixel position has D_{max} (maximum disparity range) number of matching cost $C(j, d)$, where the term j represents the position on the image row (scanline) and d is defined within the disparity range. Then the DP algorithm can be computed in two steps, forward pass and backward pass, to find the optimal disparity solution for each scanline.

1. Forward Pass

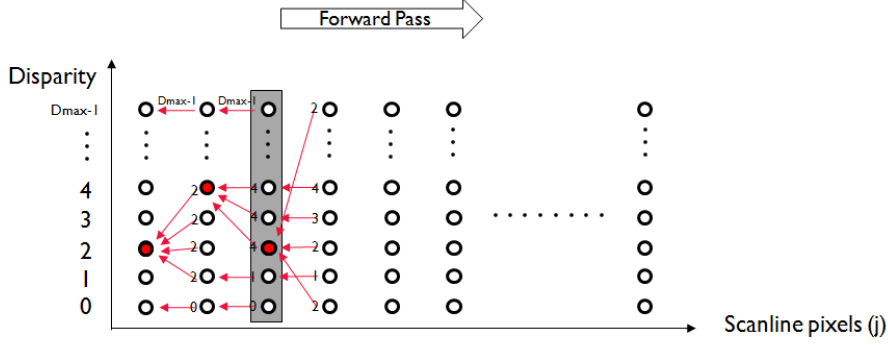


Figure 2.12: Example of forward pass function

The forward pass procedure seeks optimal backward disparity entries (path) along a scanline by selecting the minimum matching cost energy with smoothness assumption. Figure 2.12 illustrates the forward pass procedure as an example. As we mentioned before, most global optimization methods can be regarded as energy minimization problems. In the formula of forward pass Equation 2.4, the second term can be represented as an energy function which includes aggregation, matching cost and smoothness penalty terms. Therefore, the second term searches for the minimum sum of aggregation, matching cost and smoothness penalty for each disparity candidate of pixel along the scanline. In each iteration, the winner of minimum aggregation cost energy (second term) will be summed with matching cost (first term) as an aggregation cost again for the next iteration of the calculation. When executing a forward pass step, the dynamic programming tree path information is stored in *Backward_Path* array Equation 2.5 for a later backward pass. It is worthy of note that the minimum selection path is the information that the backward pass step is searching for but not the aggregation cost.

$$C_{agg}(j, d) = C_{raw}(j, d) + \min_{j \in \text{scanline}} \min_{d' \in [0, D_{max}-1]} \{C_{agg}(j-1, d') + S(d, d')\} \quad (2.4)$$

where $d \in [0, D_{max} - 1]$

$C_{agg}(j, d)$ is a matching cost aggregation array

$C_{raw}(j, d)$ is raw cost array

$S(\cdot)$ is smoothness cost function

$$Backward_Path(j, d) = \arg \min_{d' \in [0, D_{max}-1]} (C_{agg}(j-1, d') + S(d, d')) \quad (2.5)$$

where $d \in [0, D_{max} - 1]$

2. Backward Pass

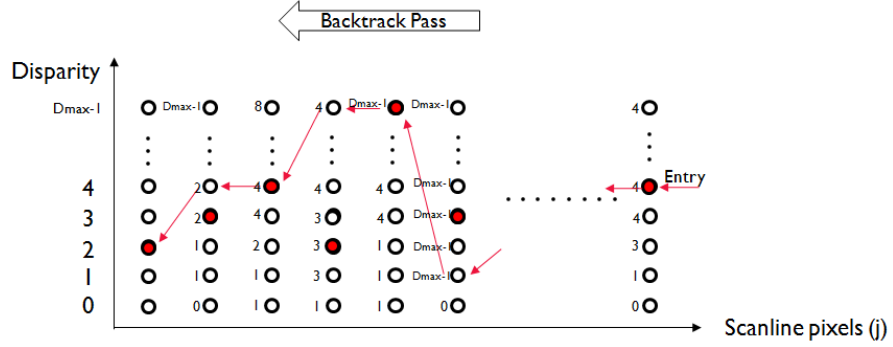


Figure 2.13: Example of backward pass procedure

The backward pass step tracks the backward pass paths in order to obtain an optimal solution over the scanline. In the beginning of the backward pass step, the initial disparity entry of the last point is decided by the minimum winner of disparity matching cost candidates as Equation 2.6. Afterward, the backward path searching Equation 2.7 follows the backward paths from the last pixel of the scanline back to start-up pixel. Finally, the final backward pass path is regarded as the optimal disparity solution over the scanline. Figure 2.13 shows the backward pass procedure as an example.

Initially, the entry point at the end of the line $(W - 1, d(W - 1))$ is computed from:

$$d(W - 1) = \arg \min_{d \in [0, D]} C_{agg}(W - 1, d) \quad (2.6)$$

where $W-1$ represents the last pixel of image scanline

Then we traverse backwards from $j = W - 1$ to $j = 0$ along the paths that were built in the forward pass stage iteratively:

$$d(j - 1) = \text{Backward_Path}(j, d(j)) \quad (2.7)$$

2.2.4 Disparity Map Refinement

This section surveys three common used disparity map refinement algorithms [49] including Left-Right Consistency Check, Cross-based Disparity Voting, and median filter for design references.

1. **Occlusion handling - Left-Right Consistency Check** Occlusion issue is introduced from the lack of correspondence matching position in stereo image pairs. Figure 2.14 illustrates that the marked regions are unmatched on the

stereo image pairs. There is no corresponding position on the other scene because the cover of foreground object. The same scenario happens on both scenes. In general, occlusion regions usually adhere to object boundaries.

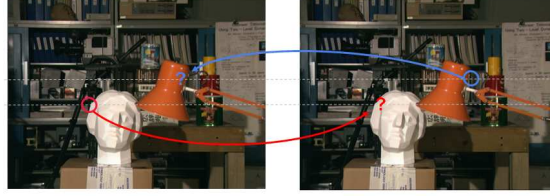


Figure 2.14: Example of occlusion problem

Left-Right consistency check is a commonly used technique to detect the occlusion regions of disparity map. Equation 2.8 checks whether the disparity value on left disparity map shares the same disparity value with the corresponding pixel position of the right disparity map. The same scenario happens on checking the right disparity map, Equation 2.9 checks whether the disparity value on right disparity map shares the same disparity value with the corresponding pixel position on the left disparity map. If their difference is larger than a threshold, we can regard the disparity pixel is inconsistent.

Consistency check of left disparity map

$$\begin{cases} \text{Good Disparity} , & \text{if } |d_{right'}(x - d_{left}(x, y), y) - d_{left}(x, y)| \leq \text{Threshold} \\ \text{Occlusion} , & \text{if } |d_{right'}(x - d_{left}(x, y), y) - d_{left}(x, y)| > \text{Threshold} \end{cases} \quad (2.8)$$

Consistency check of right disparity map

$$\begin{cases} \text{Good Disparity} , & \text{if } |d_{left'}(x + d_{right}(x, y), y) - d_{right}(x, y)| \leq \text{Threshold} \\ \text{Occlusion} , & \text{if } |d_{left'}(x + d_{right}(x, y), y) - d_{right}(x, y)| > \text{Threshold} \end{cases} \quad (2.9)$$

Since the occlusion regions can be detected by L-R consistency check method, the simplest solution to fix them is to replace the occlusion regions by the nearest good disparity pixels. Taking the left occlusion map as an example, the occlusion part can be replaced by the nearest good disparity value from the left side. On the contrary, the right occlusion map can be replaced by the nearest good disparity from the right side. Figure 2.15 demonstrates that the occlusion regions can be replaced by the nearest good disparity values from a different direction in left and right scenes. Another solution is to replace the occlusion region by the good disparity value from the same segment. In the next item, disparity voting technique will be introduced, which are based on the idea of image segmentation.

2. **Disparity Voting** A disparity voting technique is applied based on the segmentation region method. It is assumed that the segmentation with similar



Figure 2.15: Simple occlusion handling method

color intensity shares the same disparity value. Therefore, the disparity voting stage counts the disparity values within the support region to the central pixel into a histogram and then chose the winner as a disparity value. Figure 2.16 demonstrates an example of 2D support region voting for disparity map refinement.

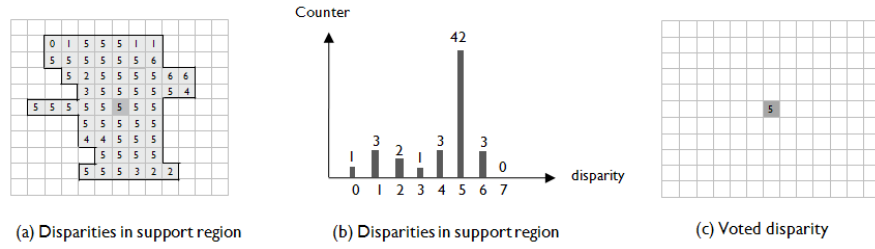


Figure 2.16: 2D disparity voting

In this thesis, we apply the cross-based algorithm that was proposed by Zhang [49] to construct the support segment region. The cross-arms information is used again to include the voting members by masking. The number of disparity values within the support region are counted into a histogram. The disparity value has maximum population within the support region is regarded as the voting result. In Lu's proposed design [51], he simplified the 2D voting into two-pass 1D voting in order to reduce the computational complexity Figure 2.17. The maximum computational complexity can be reduced from $O(N^2)$ to $O(N)$, where N is the maximum double lengths of the cross-arm.

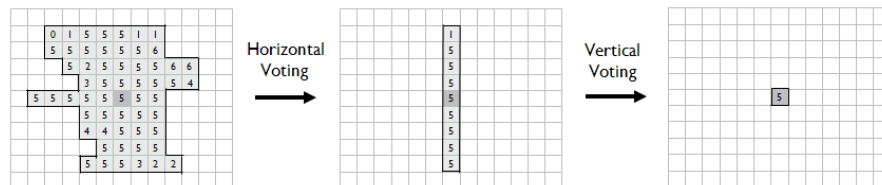


Figure 2.17: Two-pass 1D disparity voting

3. **Median Filter** Median filter is widely used in image processing applications to eliminate the so-called pepper and salt noises. It is implemented as a $N \times N$ window. Median filter selects the median value within the window pixels to form the final result of central pixel. It helps alleviate speckle and impulsive noises.

2.3 Temporal Consistency for Disparity Sequence

Previous researches [10][26][8] have proposed using energy minimization approach to enhance the temporal consistency of disparity map sequences. The general idea is to update the matching cost based on the information of motion detection. A disparity map generation reference software, DERS (Depth Map Reference Software), that comes from MPEG community also achieves temporal consistency by updating the cost function for Graph Cut stereo matching algorithm. The DERS defines the motion regions with pixel blocks by using mean absolute difference (MAD) method with threshold value. In the static regions, the data cost corresponding to the disparity value of the previous frame is scaled down. The result shows that the stability of disparity map sequences is enforced.

Stereo Matching Algorithm Implementation and Optimization on Hardware

3

This thesis proposes a SoC architecture for stereo matching computation in Figure 3.1. The architecture includes three parts: pre-processor, dynamic programming, and post-processor. In the design of pre-processor, the matching costs are generated from census transform and vertical aggregation functions. In the disparity matching processor, dynamic programming algorithm is chosen for disparity map computation and optimization step. Finally, the post-processor refines the disparity maps by L-R consistency check function, cross-based disparity voting, and median filter. Beyond the stereo matching engine, an external memory hierarchy is designed to support frame buffering function for enhancing the temporal consistency of disparity map sequences. In this thesis, we will focus on the gray regions include Dynamic Programming and Vertical Voting functions. The peripheral components include video adaptors, color space converters, and memory hierarchy will be proposed in Chapter 5 to support the stereo matching engine.

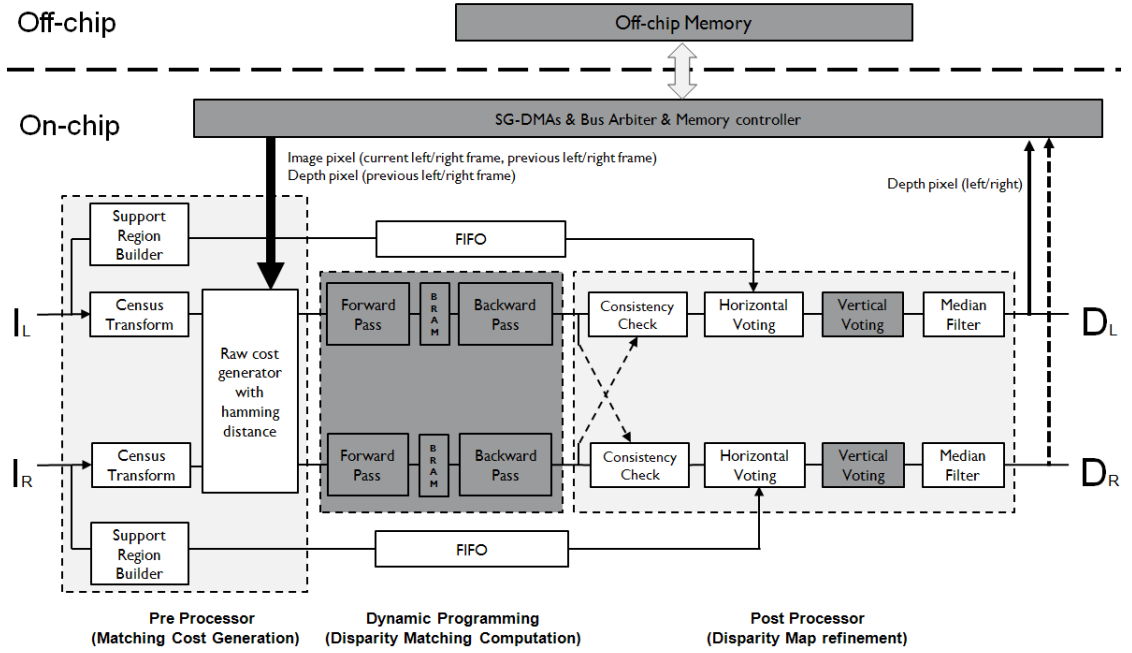


Figure 3.1: System architecture of stereo matching engine

Two proposals are implemented on Dynamic Programming and Vertical Voting functions to improve the hardware utilization. Because dynamic programming algorithm

requires tremendous memory resources for storing the backward path information, this thesis will propose a hardware efficient architecture in Section 3.1. In addition, an unconventional on-chip memory architecture with run-length encoder/decoder will be proposed in Section 3.2 to reduce the on-chip memory consumption of the Vertical Voting Processor.

Temporal inconsistency is an issue in disparity map sequences. The flickering noises are caused from the lack of links between disparity maps in the stereo matching algorithm. Any reasons such as camera noise, depth mismatching, occlusion problem, etc. could cause disparity map sequences be inconsistent. Therefore, Section 3.3 introduces temporal consistency algorithm into the stereo matching computation flow in order to increase the video stability.

3.1 Hardware Efficient Dynamic Programming Processor

This section proposes a hardware-friendly dynamic programming architecture. As in the synthesis results that have been mentioned in the beginning of this chapter, dynamic programming approach requires tremendous a memory space to store the scan line length of backward path information. Therefore, this section will implement a path data simplification idea that is inspired by Zhang Ke by taking the advantage of Potts model smoothness function. Finally, a dynamic programming architecture with Potts model smoothness function design will be presented in the last subsection.

3.1.1 Dynamic Programming Algorithm and Architecture Co-design

First, the parallelism of dynamic programming algorithm is explored for hardware implementation. Equation 3.1 shows the conventional minimum energy searching method of dynamic programming with Potts model smoothness function. The computational complexity is $O(W \cdot D_{max}^2)$ and the memory consumption is $(W \cdot D_{max} \cdot D_{bit})$. The term W represents the image width, and D_{max} represents the maximum disparity range. To make the hardware design be more efficient, we simply rewrite the minimum energy searching functions in Equation 3.2.

$$\begin{aligned} C_{agg}(j, d) &= C_{raw}(j, d) + \min_{d' \in [0, D_{max}-1]} \{C_{agg}(j-1, d') + S(d, d')\} \\ Backward_Path(j, d) &= \arg \min_{d' \in [0, D-1]} \{C_{agg}(j-1, d') + S(d, d')\} \end{aligned} \quad (3.1)$$

where the smoothness function $S(d, d') = \begin{cases} 0, & \text{if } d = d' \\ C, & \text{if otherwise} \end{cases}$ is potts model
 d and d' are adjacent disparity arrays $\in [0, D_{max} - 1]$
 j represents the pixel position of a scanline
 C is constant for smoothness cost penalty

$$\begin{aligned}
C_{minAssum} &= \min_{d' \in [0, D_{max}-1]} \{C_{agg}(j-1, d')\} + C \\
C_{agg}(j, d) &= C_{raw}(j, d) + \min\{C_{minAssum}, C_{agg}(j-1, d')\} \\
Backward_Path(j, d) &= \arg \min\{C_{minAssum}, C_{agg}(j-1, d')\}
\end{aligned} \tag{3.2}$$

The minimum aggregated cost is computed firstly and summed up with a smoothness cost penalty to form a minimum aggregate cost assumption $C_{minAssum}$. Then the term, $C_{minAssum}$, will be compared with the original aggregation cost array $C_{agg}(j-1, d')$. The compared results represent the encoded backward path information which will be stored into on-chip Block RAM. It is noteworthy that only the path information, the selection information of minimum aggregation cost, is needed for the backward step but not the minimum cost itself. If the aggregate cost is less than $C_{minAssum}$, the backward path will point to the same disparity position. If $C_{minAssum}$ is less than the aggregate cost value C_{agg} , the backward path will point to the corresponding disparity value of $C_{minAssum}$. After rewriting the forward pass equation, the computational complexity can be reduced from $O(W \cdot D_{max}^2)$ to $O(W \cdot D_{max})$. The term W represents the width of image, and D_{max} represents the maximum disparity range.

3.1.2 Dynamic Programming - On-chip Memory Optimization - Backward Path Data Compression

The backward path information can be further simplified in order to reduce the on-chip memory requirement. In Equation 3.2, the memory requirement for storing the backward path can be formulated in Equation 3.3.

$$BRAM \text{ for backward path}(bit) = (W - 1) \times D_{max} \times D_{bit} \tag{3.3}$$

where W represents scanline length
 D_{max} is maximum disparity range
 D_{bit} is the bit numbers of backward pass path information

To reduce the memory consumption, the backward path can be represented in Equation 3.4. Thanks to the characteristic of the Potts model, the backward path only has two decisions in the dynamic programming tree: to retain the same disparity or jump to the path which has the minimum sum of aggregation cost $C_{minAssum}$. After applying the proposed backward path reduction idea, the memory requirement for storing backward path is formulated in Equation 3.5. The memory stores the decision of the backward path in 1 bit and the path with minimum aggregation cost assumption. It uses 1 bit to store the backward path decisions, jump (1) or not jump (0) to the disparity assumption with minimum aggregation cost, instead of using full D_{bit} physical path. Equation 3.5 shows the memory requirement after the simplification. The D_{bit} term contains the maximum disparity range number of 1 bit encoded paths, and D bit is the path with the minimum aggregation cost assumption. Finally, the memory consumption complexity is reduced from $(W \cdot D_{max} \cdot D_{bit})$ to $(W \cdot (D_{max} + D_{bit}))$. The term W represents the image width, and D_{max} represents the maximum disparity range.

$$Backward_Path(j, d) = \begin{cases} 1, & \text{if } C_{minAssum} \leq C_{agg}(j-1, d) \\ 0, & \text{if } C_{minAssum} > C_{agg}(j-1, d) \end{cases} \quad (3.4)$$

$$Backward_MinC_Path(j) = \arg \min_{d \in [0, D-1]} \{C_{agg}(j-1, d)\}$$

where $j \in [0, \text{Image Width} - 1]$

D represents maximum disparity range

$$\text{BRAM for backward path (bit)} = (W - 1) \times (D_{max} + D_{bit}) \quad (3.5)$$

where W represents scanline width

D_{max} is maximum disparity range

D_{bit} is the bit number of backward path information

Figure 3.2 is an example shows each backward path information is encoded in one bit. The red dot represents the path with minimum aggregation cost assumption.

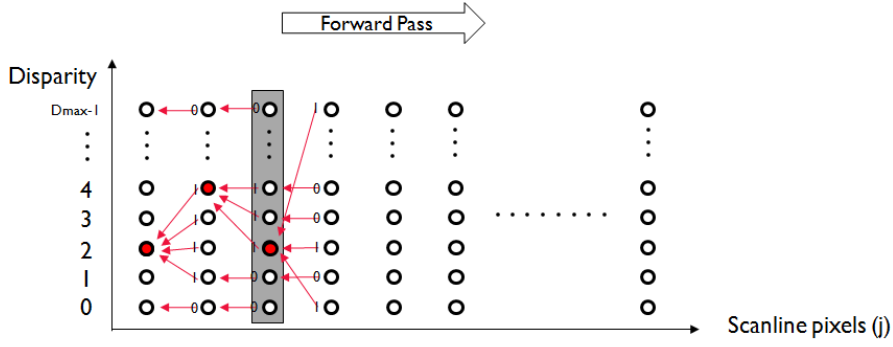


Figure 3.2: Forward pass with backward path encoding

In the backward pass step, the backward path information will be decoded as the Equation 3.6. When the path decision is 0, the backward entry retains the same disparity. When the path decision is 1, the backward entry will point to the path with the minimum aggregation cost assumption. Finally, Figure 3.3 is an example that shows how to decode the backward path data and traverse the procedure work.

$$d(j-1) = \begin{cases} Backward_MinC_Path(j), & \text{if } Backward_Path(j, d) = 1 \\ d(j), & \text{if } Backward_Path(j, d) = 0 \end{cases} \quad (3.6)$$

where j represents image scanline pixels

3.1.3 Dynamic Programming - On-chip Memory Data Mapping

This thesis proposes using only one scanline length of 2-Port Block RAM (BRAM) to store the backward path data for the Dynamic Programming architecture. 2-Port RAM

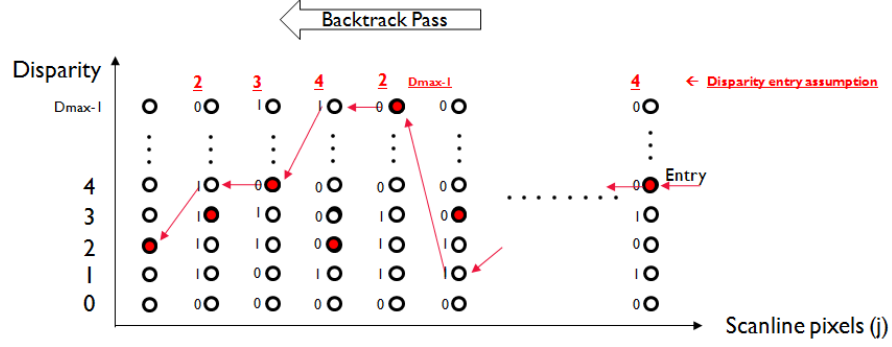


Figure 3.3: Example of backward pass with decoded path data

allows that writing and reading commands can operate concurrently. As Figure 3.4 shows, the forward pass loop writes the new incoming backward path data into BRAM; meanwhile, the backward pass loop reads the backward path data that was generated for the previous scanline from BRAM. So the Dynamic Programming processor can keep processing the matching cost inputs and generating disparity output simultaneously in pipeline architecture. In order to avoid data conflict problem, a relatively sophisticated address generation mechanism is proposed here. The memory reading address (backward pass) is generated in a back and forth order, and the memory writing address (forward pass) will follow the reading address in 1 cycle of delay. Figure 3.5 is an example of memory address access pattern for 1024 data length of line buffer. By applying the 2-Port-RAM and a sophisticated address generator, the Dynamic Programming processor is able to achieve pipeline processing and utilize the on-chip memory efficiently.

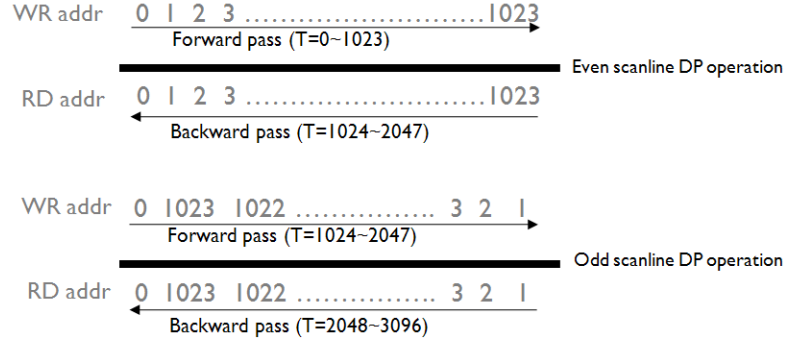


Figure 3.4: DP operation sequence

3.1.4 Dynamic Programming Processor - Hardware Architecture

Finally, a hardware efficient Dynamic Programming architecture is proposed in Figure 3.6 and Figure 3.7. In the proposed hardware architecture, Equation 3.4 is applied to construct the forward pass function, which generates encoded backward path information. This design expands the maximum parallelism. Both the forward pass and backward

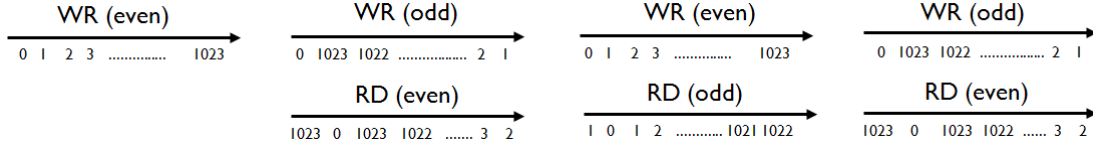


Figure 3.5: Example of 2-port RAM access patterns

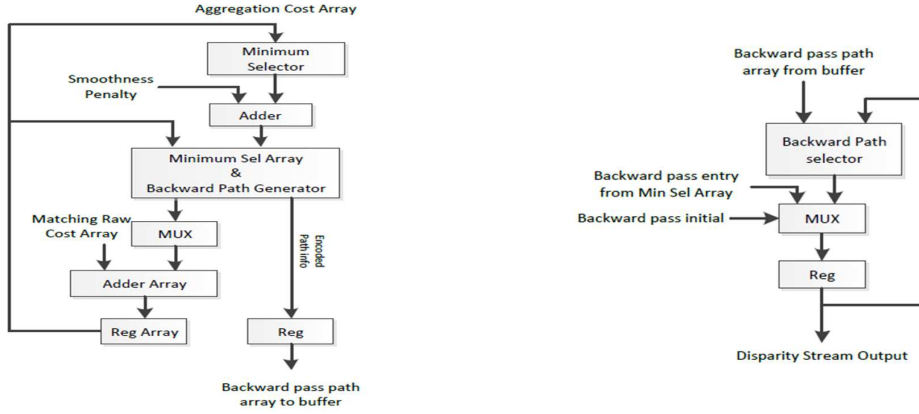


Figure 3.6: Forward Pass HW design

Figure 3.7: Backward Pass HW design

pass computations should complete in 1 clock cycle. The forward pass function processes one set of matching cost array for one pixel at one time. In the meanwhile, the backward pass function back tracks the path information that was stored in 2-port RAM and generates disparity output for one pixel. concurrently

3.2 Run-Length Coding Algorithm and Disparity Map Sequence

In this section, we first propose using a run-length coding algorithm to compress disparity map data. Then we apply this idea to reducing the on-chip memory utilization of Vertical Voting processor in our stereo matching engine.

From our observation, the output sequence of disparity map often shows a long repetitive patterns. Figure 3.8 is an example of disparity sequence that is captured from the output of Dynamic Programming processor. Therefore, it inspired us to encode the disparity sequence to format **[repeat count, disparity value]**. This encoding technique is so called run-length coding (RLC)[30]. The pseudo-code of run-length encoder and decoder are presented in Equation 1 and Equation 2 separately.

In this thesis, we creatively apply run-length coding algorithm to the Vertical Disparity Voting processor [51] architecture to reduce the on-chip memory resource.

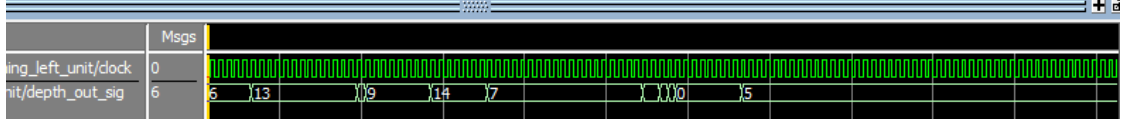


Figure 3.8: Disparity output from dynamic programming function.

The depth/disparity value can be represented in run-length coding format [Repeat count, Disparity value]: [12, 13], [0, 5], [7, 9], [6, 14], [18, 7], [1, 2], [0, 3], [0, 2], [7, 0], [31, 5], [15, 5].

Algorithm 1 Run-Length Coding: encoder

```

while (1) do
  if ( $disp \neq next\_input\_disp$ ) or ( $count > max\_run\_length$ ) then
     $output\_disp \leftarrow disp$ 
     $output\_count \leftarrow count$ 
     $disp \leftarrow next\_input\_disp$ 
     $count \leftarrow 0$ 
  else
     $count \leftarrow count + 1$ 
  end if
end while

```

Figure 3.9 shows the original memory architecture of the Vertical Disparity Voting processor. It takes 30 scanline lengths of 2-Port RAM (line buffer) to buffer the output sequence of disparity map in a circular memory architecture. This memory architecture is commonly used in stream processors to include both horizontal and vertical dimensions of pixel data by using data-reuse technique. However, the 30 line buffers will consume tremendous on-chip memory resources. Hence, we propose a new memory architecture with run-length coding mechanism in Figure 3.10.

Different from the circular memory architecture, the proposed memory architecture only writes the encoded run-length disparity sequence into one line buffer at a time; meanwhile, the encoded run-length data are fetched out in parallel in order to access

Algorithm 2 Run-length coding: decoder

```

while (1) do
  if ( $count \neq 0$ ) then
     $output\_disp \leftarrow disp$ 
     $count \leftarrow count - 1$ 
  else
     $output\_disp \leftarrow next\_input\_disp$ 
     $disp \leftarrow next\_input\_disp$ 
     $count \leftarrow next\_input\_count$ 
  end if
end while

```

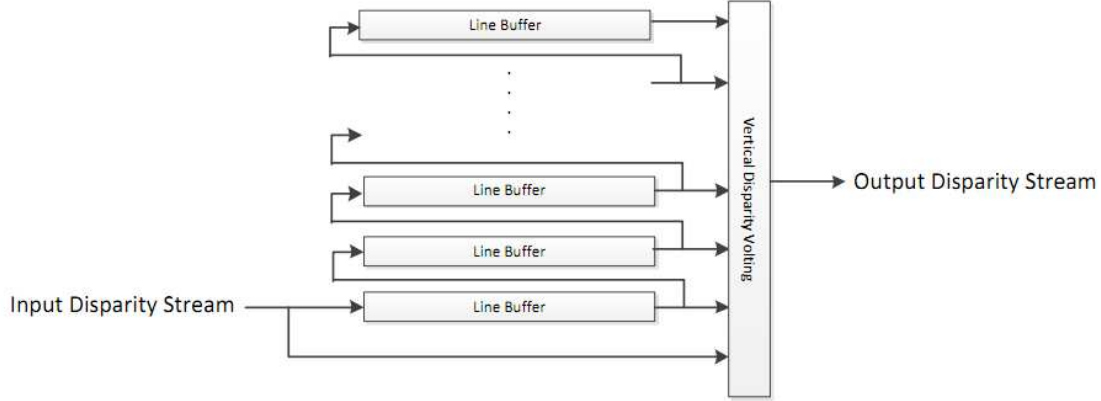


Figure 3.9: Circular memory architecture for Vertical Disparity Voting processor

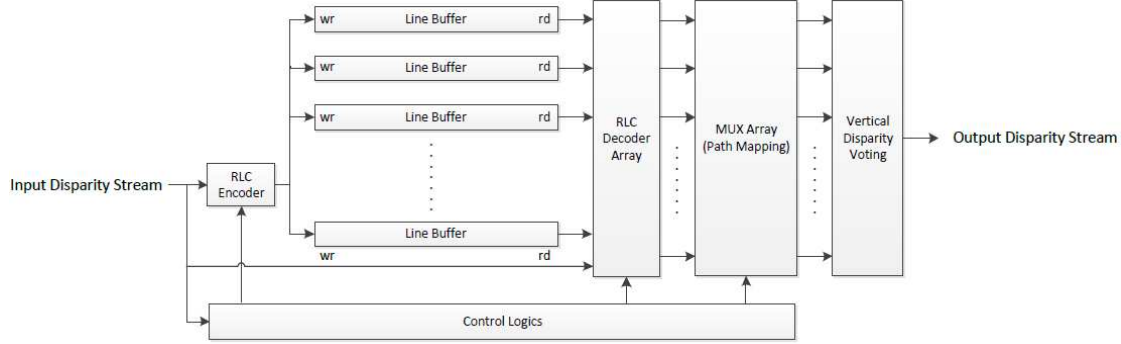


Figure 3.10: Proposed memory reduction architecture with run-length coding

vertical disparity pixels for computation. The MUX array is used to map the disparity pixels to a corresponding vertical positions. Although the proposed memory architecture is more sophisticated than the original circular memory architecture, the length of the line buffers can be reduced dramatically in several magnitudes.

Unfortunately, the run-length coding algorithm doesn't generate fixed-length outputs. The length of encoded sequence changes depend on the complexity of the disparity map. In the other words, the length of line buffers should be long enough to store the compressed disparity sequence. If the length of the compressed disparity sequence exceeds the length of line buffer, the run-length encoder will discard them. Besides, the decoder is designed to repeat the last valid disparity pixel when encountering an incomplete run-length sequence. It shows a tradeoff between the length of line buffers (on-chip memory resource) and pixel error rate. In the next Chapter, we will further evaluate the optimal settings based on different complexities of disparity map sequences.

3.3 Temporal Consistency for Disparity Sequence

To solve the inconsistent disparity video problem that was mentioned in Chapter 1, this section applied a simple cost adjustment method based on Absolute Difference (AD) algorithm to enforce the temporal consistency of disparity sequence. The temporal consistency method encourages the static background to select the same disparity value again; contrarily, it encourages motion regions to explore new disparity result. In our proposed algorithm, the matching cost is adjusted based on the luminance variation of the pixel in 2 frames. If the pixel luminance in the previous frame shares a similar luminance value in current frame, the disparity value in the previous frame is expected to be selected again. Equation 3.7 shows the matching raw cost adjustment function. The corresponding raw matching cost to the disparity value will be scaled down when the absolute difference of pixel luminances between previous and current frames is less than a defined threshold value.

$$C_{raw} = \begin{cases} C_{raw}(j, d) \times \alpha, & \text{when } (|Y_{(x,y,t)} - Y_{(x,y,t-1)}| < Threshold) \text{ and } (d = d(x, y, t)) \\ C_{raw}(j, d), & \text{otherwise} \end{cases} \quad (3.7)$$

where $j \in [1, Scanline \ width]$

α represents a scaling down coefficient, $Y_{(x,y,t)}$ is the luminance value of the pixel in current frame; $Y_{(x,y,t-1)}$ is the luminance value of the pixel in previous frame.

Figure 3.11 demonstrates that the corresponding matching cost is modified depending on the luminance variation of pixels in 2 frames.

Because the temporal consistency function requires the pixel luminance and disparity information from current and previous frames, the system architecture of the stereo matching engine has to construct a memory hierarchy for buffering history disparity map and luminance image. The design of the memory hierarchy will be introduced in the next chapter.

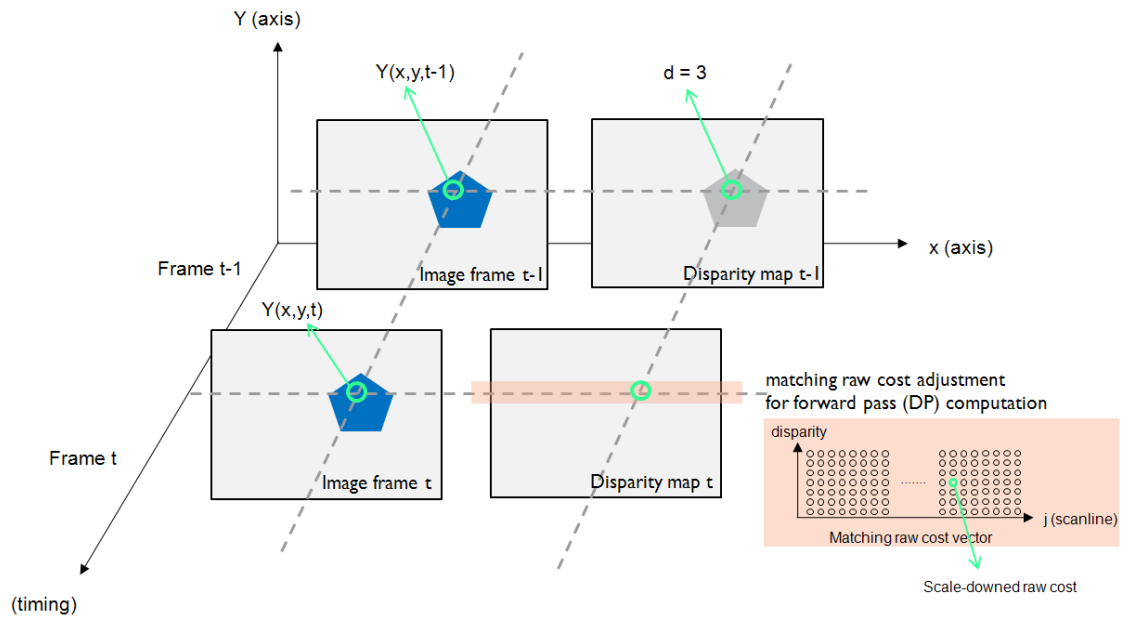


Figure 3.11: Example of temporal consistency algorithm in dynamic programming

Evaluation of the proposed Stereo Matching Hardware

4

This chapter summarizes the experimental results of the stereo matching architecture for last chapter. The proposed algorithms and hardware architectures are co-evaluated from two points of view:

- Quality of disparity map and disparity sequences
- Hardware utilization and scalability
- Computational performance

In the beginning of the last chapter, we proposed a stereo matching computational flow. In Section 4.1, we measure the quality of disparity maps that are generated from the Stereo Matching Engine with dynamic programming algorithm. In Section 4.2, we measure the temporal quality of disparity map sequences that are generated from the Stereo Matching Engine with temporal consistency function. Then, in Section 4.3, we estimate the hardware usage of the proposed Dynamic Programming Processor design. Afterward, in Section 4.4, the trade-off between compression rate (length of line buffers) and pixel error rate are explored based on the proposed on-chip memory architecture with run-length coding for Vertical Voting Processor. Section 4.5 evaluates the hardware usage of the entire stereo matching design. Finally, the critical path of the Stereo Matching Engine design is evaluated in Section 4.6.

4.1 Global Stereo Matching with Dynamic Programming - Disparity Map Evaluation

The evaluation disparity maps are generated from the hardware model of the stereo matching algorithm. Since the corresponding hardware RTL designs are implemented in VHDL, we can also generate test disparity maps or video sequences by using RTL simulation tools such as Modelsim and Candence Simulator.

Two evaluation models [13], Root-Mean-Squared Error(RMS) Equation 4.1 and Bad Matching Pixel(B) Equation 4.2, are usually chosen to compute the matching quality of disparity maps. The generated disparity maps are compared with the ground truth disparity maps. The ground truth disparity maps are regarded as precise disparity maps, which are produced from structured light system. In this thesis, we use an academic evaluation benchmark tool, Middlebury Stereo Evaluation Benchmark [43], which is provided by Middlebury University. It applies Bad Matching Pixel evaluation model on disparity map quality estimation. In the Middlebury Stereo Evaluation website, three error metrics are supported: unconcluded, complete image and disparity discontinuity

regions. Therefore, it is possible to compute the percentage of bad matching pixels regardless of non-occlusion regions or border regions. This provides a designer with more flexibility to focus on improving specific region.

1. Root-Mean-Squared Error

$$RMS = (\frac{1}{N} \sum_{(x,y)} |d_C(x,y) - d_T(x,y)|^2)^{\frac{1}{2}} \quad (4.1)$$

where N is the total number of pixels.

$d_C(x,y)$ is computed disparity map and $d_T(x,y)$ is ground truth map.

2. Percentage of Bad Matching Pixels

$$B = \frac{1}{N} \sum_{(x,y)} |d_C(x,y) - d_T(x,y)| > \lambda_d \quad (4.2)$$

where λ_d is the disparity error threshold. Normally, it is set to 1.

Four commonly used academic stereoscopic images (including Tsukuba, Venus, Teddy, and Cones [44]) are evaluated with ground truth disparity maps. The Middlebury's stereo evaluation benchmark compares the average disparity map quality of the four test sets with others' stereo matching algorithms.

4.1.1 Parameter Exploration for Dynamic Programming Processor

There are two parameters have great effects on the performance of Dynamic Programming Processor. One is the threshold for the absolute difference of adjacent pixel luminances. If the absolute difference is larger than the threshold, the continuous boundary is assumed on the disparity map. The other parameter is the smoothness cost penalty from Potts model smoothness function. If the adjacent pixel is regarded as a discontinuous boundary, the smoothness cost penalty will be scaled down in order to provide the minimum energy selection function with more flexibility to select other backward path. By contrast, the smoothness penalty will be kept high in order to preserve the smoothness (select the same backward path again). In the following, we will explore the optimal settings of the threshold (th_c) and the smoothness cost penalty C .

The evaluation starts from exploring the threshold. Then use the optimal fixed threshold to explore the smoothness penalty. We do it iteratively until we get the optimal result. Finally, the evaluation results Figure 4.1 show $th_c = 15$ and $C = 5$ are the optimal combination based on the average error rate of four test sets. The optimal error rate achieves a 6.6% pixel error rate in average on Middlebury's testbench Figure 4.2. The disparity map results are showed in Figure 4.3.

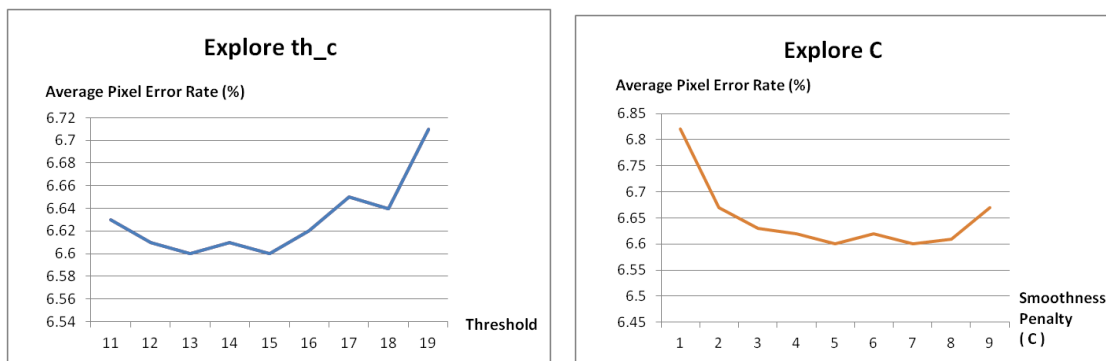


Figure 4.1: Parameter exploration for Dynamic Programming

Error Threshold = 1		Sort by nonocc						Sort by all						Sort by disc						Average Percent Bad Pixels						
Error Threshold...																										
Algorithm	Avg.	Tsukuba <small>ground truth</small>			Venus <small>ground truth</small>			Teddy <small>ground truth</small>			Cones <small>ground truth</small>															
	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc										
RegionTreeDP [18]	41.4	1.39	37	1.64	17	6.85	38	0.22	24	0.57	25	1.93	14	7.42	45	11.9	34	16.8	39	6.31	57	11.9	59	11.8	54	6.56
VSW [108]	41.5	1.62	45	1.88	35	6.98	41	0.47	45	0.81	35	3.40	40	8.67	70	13.3	50	18.0	53	3.37	25	8.85	25	8.12	20	6.29
YOUR METHOD	41.8	2.42	58	2.78	57	10.9	73	0.20	19	0.47	21	2.07	15	6.38	24	11.7	25	16.1	25	4.32	58	10.6	54	11.3	55	6.60
EnhancedBP [24]	43.8	0.94	7	1.74	23	5.05	5	0.35	37	0.86	44	4.34	45	8.11	62	13.3	47	18.5	50	5.09	73	11.1	51	11.0	55	6.69
PUTv3 [63]	44.2	1.77	52	3.86	71	9.42	51	0.42	41	0.95	50	5.72	59	7.02	41	14.2	52	18.3	57	2.40	2	9.11	32	6.56	2	6.64

Figure 4.2: Evaluation results from Middlebury's benchmark

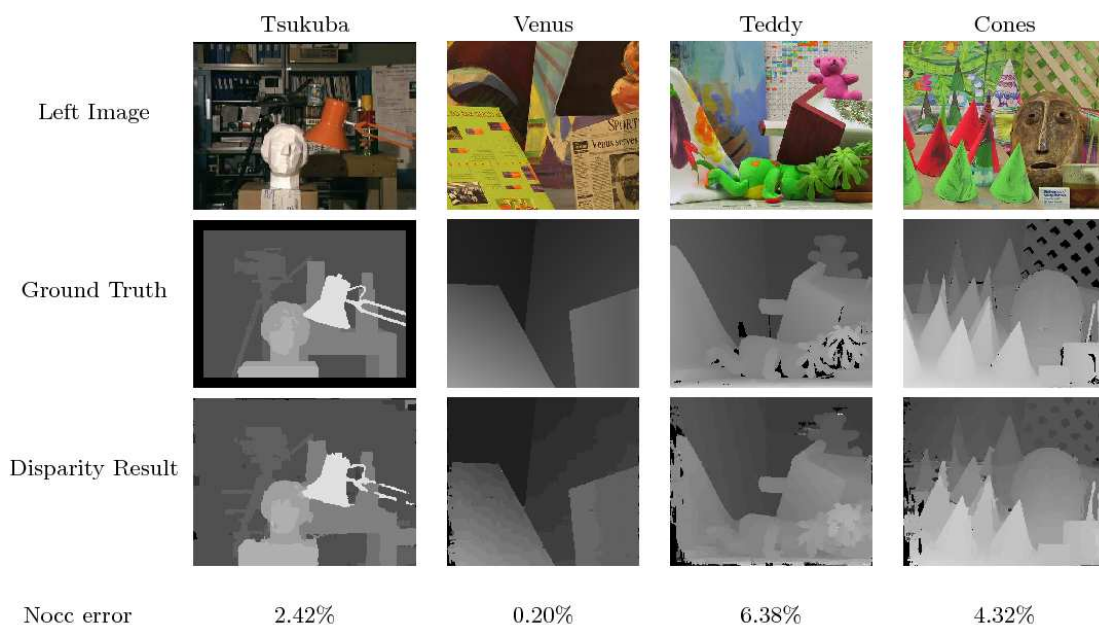


Figure 4.3: Ground truth disparity maps and test disparity maps

4.1.2 Comparison of Local and Global Stereo Matching Approaches

In this subsection, we estimate the quality improvement of the disparity maps with the help of dynamic programming algorithm. Without global optimization stage, the stereo algorithm can be regarded as a local approach because it only uses WTA strategy for matching computation.

Referring to Zhang’s work [52], we improve the winner-take-all stereo matching method with Dynamic Programming algorithm. Zhang’s work achieves a 8.2% average error rate on Middlebury’s Benchmark. After applying Dynamic Programming algorithm in the stereo matching computational flow, the average error rate decreases to 6.6%.

4.2 Temporal Quality Evaluation for Disparity Map Sequences

In this section, the temporal quality of disparity map sequences is assessed through two methods: empirical and VSRS. The main problem of temporal quality evaluation for disparity map sequences is the lack of ground truth disparity video source. Therefore, we first adapt empirical observation because it is the most straightforward way to evaluate the temporal quality of a disparity map sequences. The other evaluation method measures the synthesis quality through View Synthesis Reference Software (VSRS) [36], which is proposed by MPEG-FTV Group. VSRS requires both left and right disparity maps that are extracted from multiple or stereoscopic cameras to generate the virtual central view. The virtual central video sequences are then compared with the true central video sequences in terms of PSNR. MPEG-FTV Group also provides a model, Temporal PSPNR [53], to measure the temporal quality of the synthesized view. PSPNR model classifies an image into two regions: static and motion regions. It converts the noise perception sensitivity that is influenced by motion into a probability model. In the motion regions, three types of noise are differentiated: plain, edge, and texture. Therefore, we will use VSRS and PSPNR measurement tool 2.0 [53] that are supported by MPEG-FTV community to measure the temporal consistency performance of our stereo matching algorithm.

4.2.1 Parameter Exploration for Temporal Consistency

We explore the parameter α , which scales down the matching cost. The scaling parameter not only influences the consistency effects of disparity map sequences but also affects the disparity map quality because global algorithm is used. Therefore, we try to explore the optimal configuration for α . Figure 4.4 illustrates that $\alpha = 0.9$ achieves the optimal PSNR and TPSPNR in the 100 frames in Book Arrival test sequences.

4.2.2 Evaluation of Temporal Consistency Function

The following captured disparity map sequences Figure 4.5 demonstrate the stability of disparity sequence after temporal consistency function is introduced into the stereo

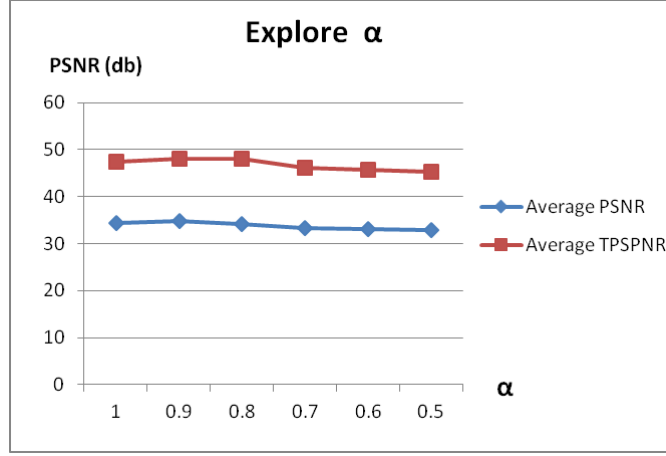


Figure 4.4: Exploration of matching cost scaling parameter α with Book Arrival test set

matching algorithm. The flickering and mismatching disparity regions are reduced obviously.

Since there is no standard ground truth video source that can be used to measure the consistency of disparity map sequences, one solution is measuring the quality of synthesized virtual sequences. To measure the influence of temporal consistency function to the stereo matching algorithm, the PSNR and Temporal PSNR of synthesized virtual sequences are summarized in Figure 4.6 based on 100 frames of Book Arrival test sequences.

4.3 Hardware Resource Estimation of Dynamic Programming Processor

In the last chapter, we proposed a hardware friendly dynamic programming architecture for Stereo Matching Engine. Hence, the hardware utilization of the Dynamic Programming Processor will be estimated in this section. The RTL circuit gate count is measured by a Cadence RTL compiler with OSU academic TSMC 0.25 library.

4.3.1 Hardware Resource Estimation

Figure 4.7 shows the cell area of RTL circuit. We measure the gate count in different disparity range scenarios. It shows that the cell area increases linearly when the disparity range rises.

The on-chip memory consumption (BRAM) is evaluated in Figure 4.8. We also explore the scalability under different disparity ranges. It shows the memory consumption increases linearly when the disparity range rises. We further compare the improvement of on-chip memory utilization before applying 2-Port RAM and the path simplification method. In disparity 64 scenario, the BRAM requirement of the Dynamic Programming

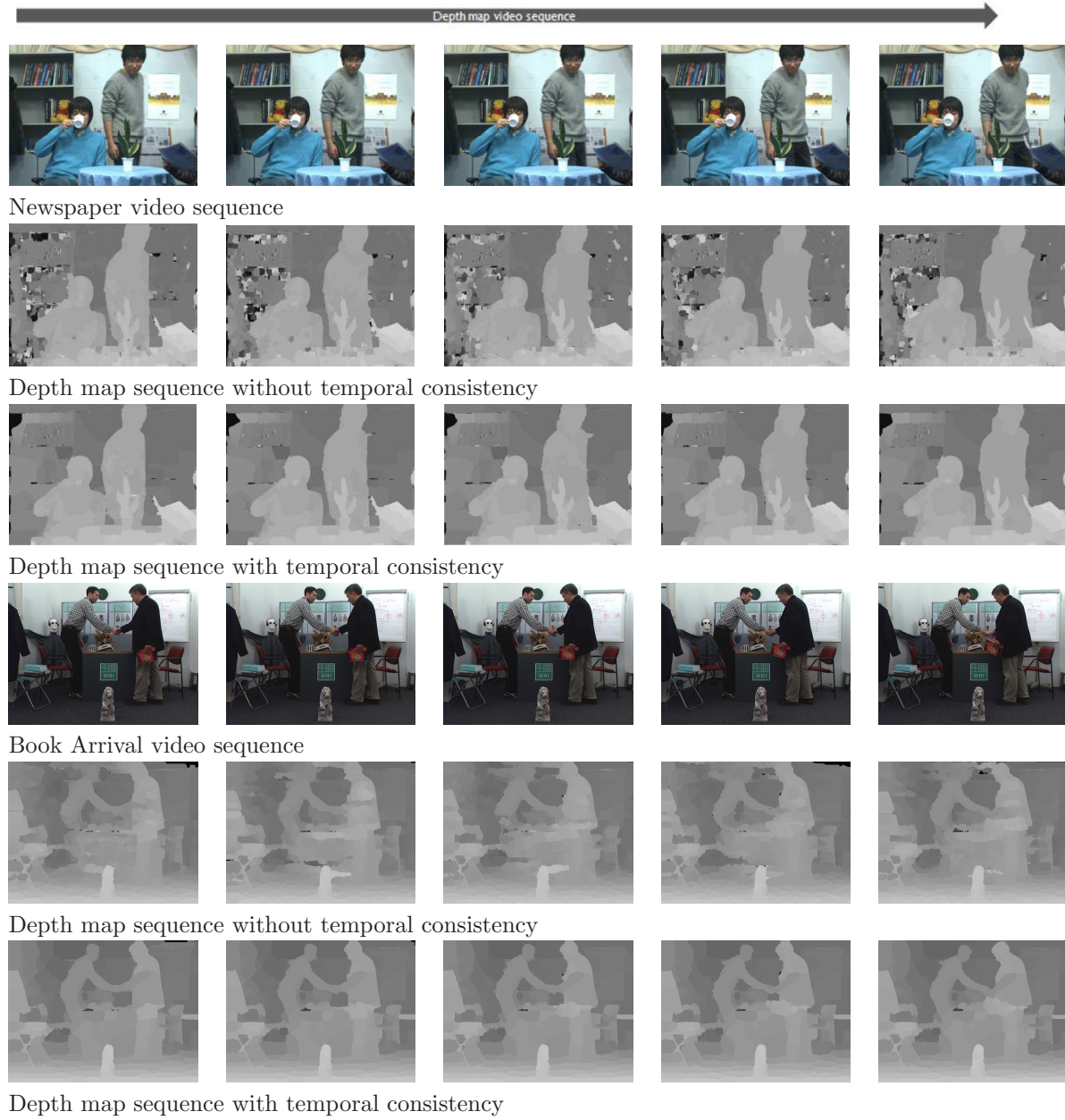


Figure 4.5: Temporal consistency empirical evaluation

Processor with path simplification design only takes one eleventh of 2-Port RAM.

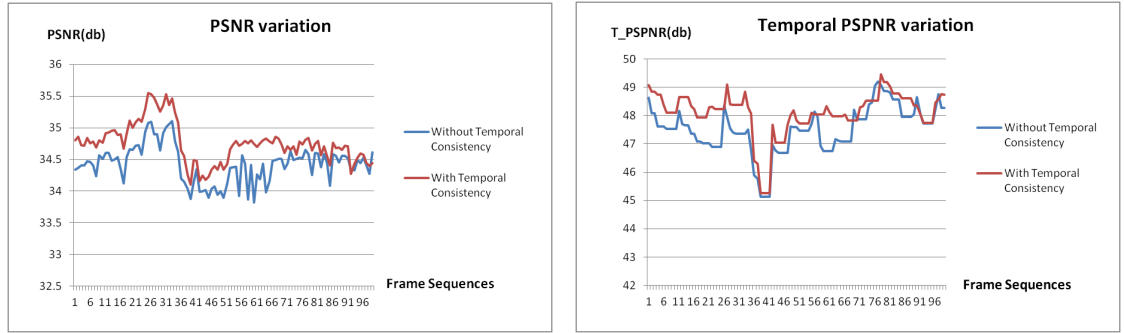


Figure 4.6: PSNR improvement of temporal consistency

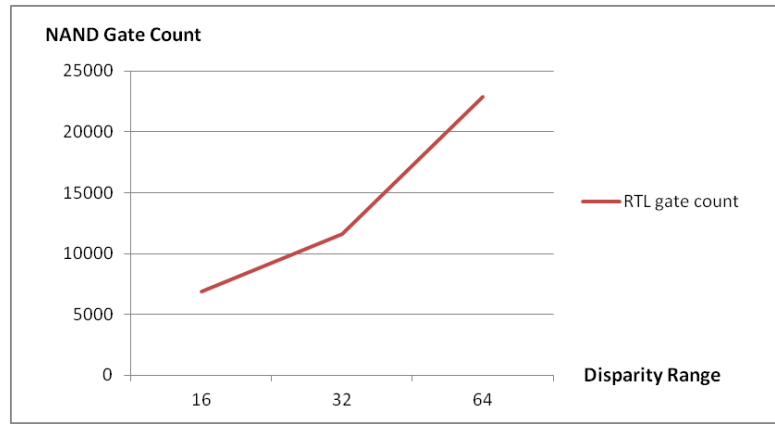


Figure 4.7: RTL circuit gate count synthesis for Dynamic Programming Processor

4.4 Evaluation of the Memory Architecture with Run-length Coding for Vertical Voting Processor

In last Chapter, we propose an on-chip memory architecture with run-length encoder/decoder to reduce the memory usage of the Vertical Voting Processor. However,

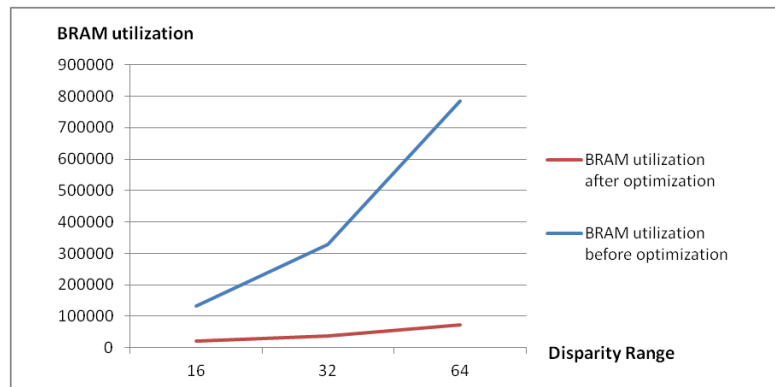


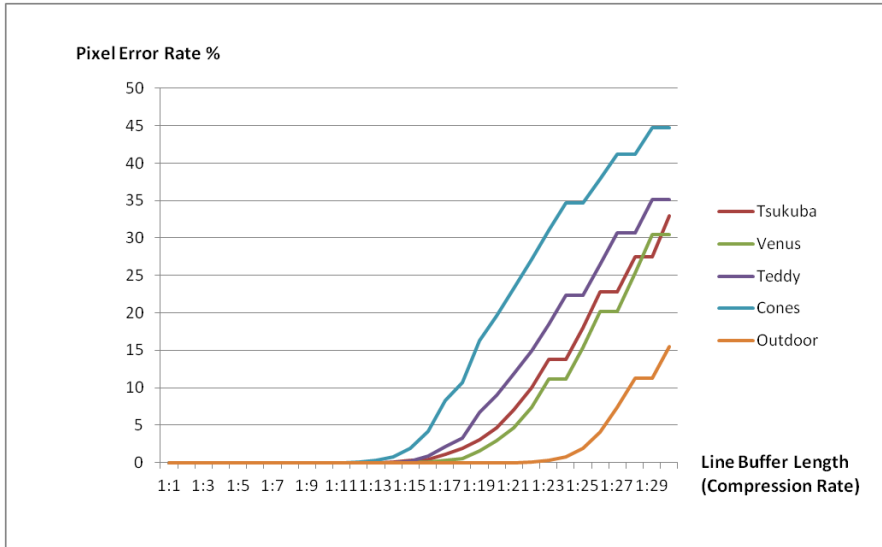
Figure 4.8: On-chip memory utilization estimation of Dynamic Programming Processor

run-length coding is a variable length coding algorithm because the compression rate varies depending on the complexity of disparity sequences. Since the proposed memory architecture uses fixed length line buffers (BRAM) to store the compressed disparity scanline pixels, the length of line buffer should be long enough; otherwise, part of the encoded disparity data will be discarded. Besides, we explore the range of run-length counter because it relates to the compression efficiency and the BRAM utilization. Thus, we propose a procedure to explore the trade-off between compression rate, bad pixel rate, range of run-length counter, and hardware efficiency for different complexities of disparity sequences.

To measure the error rate that is caused by the truncated length of line buffers and different complexities of disparity sequences, we evaluate the decoded disparity map with raw disparity map. Then, the percentage of bad pixels is computed from Equation 4.2.

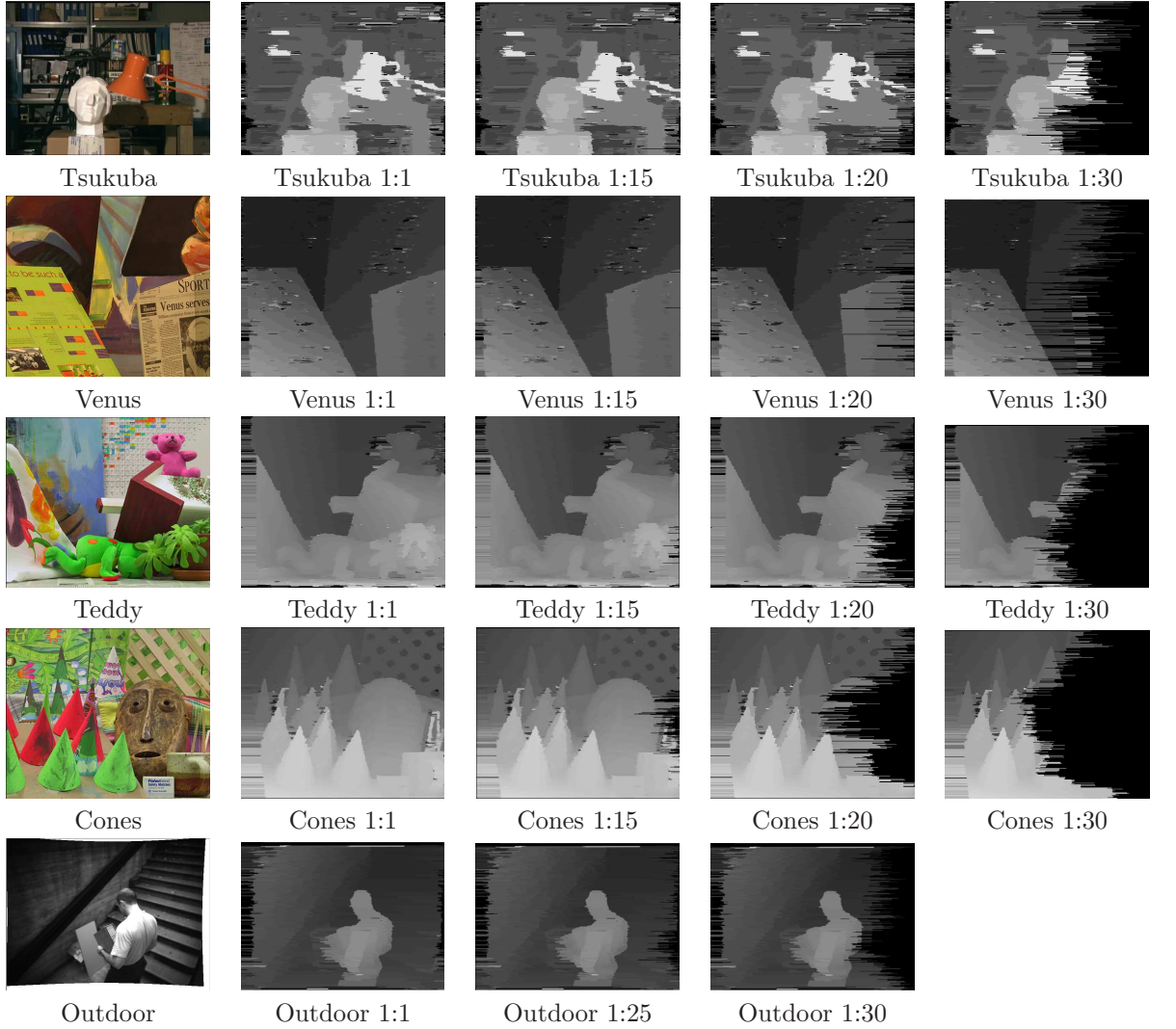
4.4.1 Parameter Exploration for the Memory Architecture with Run-length Coding

Figure 4.9 explores the bad pixel rate under different compression rates (truncated length line buffer). Several common used stereoscopic test sets including Tsukuba, Venus, Teddy, Cones [44], and Outdoor are chosen in the evaluation. First, we extract the disparity maps that are directly output from the proposed Dynamic Programming Processor to be the control group. These disparity maps will be compared with the disparity maps that are generated from the proposed memory architecture with run-length coding function in bad pixel error rate.



disparity map, Outdoor, is able to tolerate more than a twenty times of compression rate without losing quality. This is because of the high simplicity of the disparity map. In contrast, the disparity map, Cones, can only achieve a twelve times of compression rate without losing quality because of the high complexity of the Cones' disparity map. Since the compression rate varies in different disparity map complexities, it should be adjusted based on applications.

Figure 4.10: Disparity results by using the proposed memory architecture with RLC



The range of the run length counter is another parameter that should be taken into consideration when applying run-length coding. Figure 4.11 and Figure 4.12 illustrate the evaluation results on Outdoor and Cones' disparity maps when applying different ranges of run length counter and compression rates. From the observation, the low range run-length counter can only achieve a low compression rate without quality degradation

because it requires longer length of line buffers to store repetitive disparity sequences. In contrast, the higher range of run-length counter could achieve higher compression rate without quality degradation.

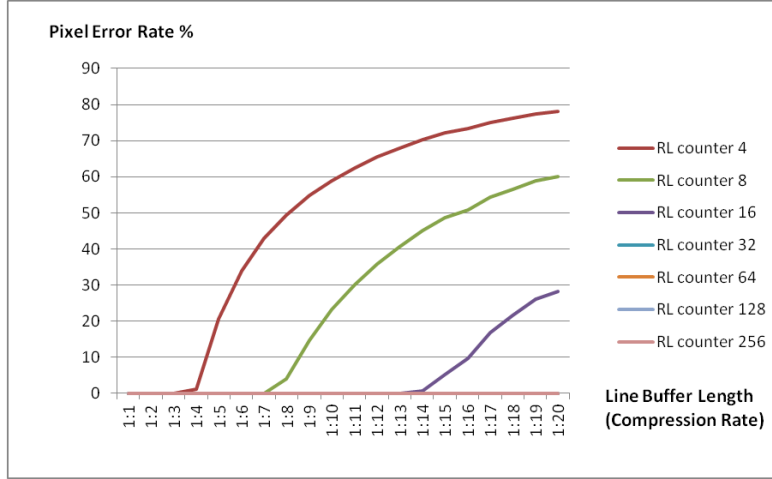


Figure 4.11: Explore the range of run length counter (Outdoor)

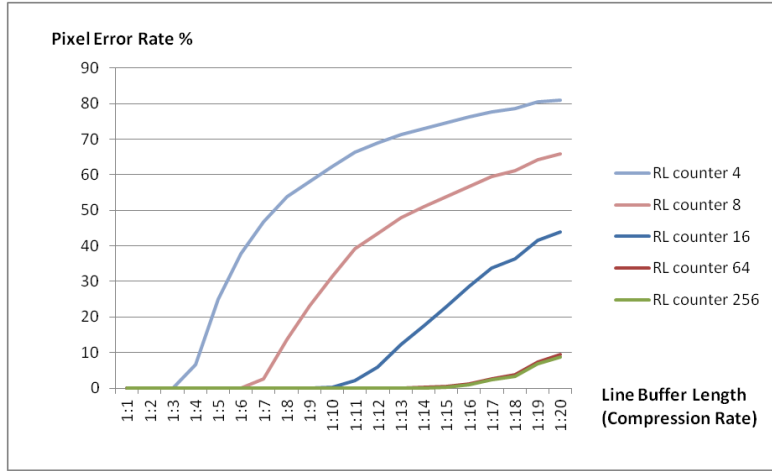


Figure 4.12: Explore the range of run length counter (Cones)

In the case of the Cones' disparity map, the compression rates, without losing quality, stop increasing after a certain range of run-length counter. In order to achieve better hardware efficiency, the range of run length counter should be chosen optimally. The optimal compression rate is explored based on pixel error rate criteria. Table 4.1 estimates the memory consumptions in different truncated lengths of line buffer (compression rate) and different ranges of run length counters. Equation 4.3 illustrates the memory consumption formula for the proposed memory architecture with run-length encoder/decoder. Taking the Cones' disparity map for example, a 32

run-length counter and 64 truncated length line buffer are the optimal configurations to achieve both optimal on-chip memory consumption and lossless pixel error rate.

$$\text{BRAM size} = \text{line buffer length} \times (\text{bit number of run length counter} + \text{bit number of disparity value}) \quad (4.3)$$

Table 4.1: Hardware resource comparison of run length counter and compression rate

Range of Run Length Counter	4 (2 bit)	8 (3 bit)	16 (4 bit)	32 (5 bit)	64 (6 bit)	128 (7 bit)	256 (8 bit)
512 Line Buffer Length(50%)	4096	4608	5120	5632	6144	6656	7168
256 Line Buffer Length(25%)	2048	2304	2560	2816	3072	3328	3584
128 Line Buffer Length(12.5%)	1024	1152	1280	1408	1536	1664	1792
64 Line Buffer Length(6.25%)	512	576	640	704	768	832	896
Assuming the disparity value range is 64 (6 bit) and the full scanline length is 1024							

4.4.2 Hardware Resource Estimation and Comparison

Finally, we evaluate the hardware usage of the proposed memory architecture for the Vertical Voting Processor. Although the BRAM (line buffer) requirements are reduced by using the proposed memory architecture with run-length coding, extra logic gate counts are burdened from extra circuits, including run-length encoder, run-length decoders, MUX array, and control circuit. Table 4.2 lists the resource usage of two memory architecture designs for Vertical Voting Processor. One of the memory architectures uses 30 full line buffers in a circular memory architecture, while the other memory architecture applies the run-length coding algorithm which contains 31 truncated line buffers with extra logic counts (1 run-length encoder, 31 runlength decoders, MUX array, and control logics). We assume the range of run length counter is 32(5bit) and the truncated length of line buffers is 128(1:8). The design is measured by Cadence RTL compiler with OSU TSMC 0.25um library. Comparing the proposed memory and circular memory architectures, the BRAM consumption is reduced by 4.75 times but with 10.5k extra gate overhead.

Table 4.2: On-chip memory architecture resource utilization analysis

Mem Architecture	NO. of Line Buffer	Line buffer Length	Total BRAM (bit)	RTL Gate Count(NAND)
Circular	30	1024	245760	0
Proposed	31	64(1:8)	51584	10.5 K

Assuming the scanline length of disparity map is 1024, and one disparity pixel is represented in 8 bits

4.5 Hardware Resource Estimation of Stereo Matching Engine on FPGA

We further estimate the improvement of the resource overhead problem in the stereo matching engine Table 4.3. On the one hand, the logic gate count of dynamic pro-

gramming function is reduced 1.72 times by rewriting the forward pass equation. On the other hand, the on-chip memory consumption of dynamic programming function is improved 11.4 times by simplifying the backward path information and using 2-Port RAM. Furthermore, the block RAM of vertical voting function is reduced 4.75 times when applying a run-length coding technique on disparity data compression. Inevitably, the run-length encoder, decoder and Mux circuit introduce an extra logic gate penalty to the vertical voting process unit 1.44 times. Finally, the hardware optimization proposals achieve 2.53 times of improvement to the overall on-chip memory consumption and remain almost the same logic gate count in this thesis work.

Table 4.3: Hardware resource analysis of optimized stereo matching engine

<i>Stereo Matching Engine</i>	<i>Yig [47]</i>			<i>Proposal</i>		
<i>Processor units (x2)</i>	<i>LC Comb.</i>	<i>LC Reg.</i>	<i>Block Mem (Bit)</i>	<i>LC Comb.</i>	<i>LC Reg.</i>	<i>Block Mem (Bit)</i>
CT + SRB	2916	1451	498704	2916	1451	498704
Bypass FIFO	24	24	131072	24	24	131072
Raw Cost Scatter	6065	3991	2672	6065	3991	2672
Dynamic Programming	16717	4452	1630208	9300	2212	143360
Disparity Output Logic	9	38	0	9	38	0
Consistency Check	622	1429	0	622	1429	0
Horizontal Voting	10782	9168	624	10782	9168	624
Vertical Voting	10602	8704	491520	15288	10036	103168
SR FIFO for Voting	0	0	294912	0	0	294912
Median Filter	448	389	32768	448	389	32768
Sum	49376	29703	3049712	45454	28738	1207280

4.6 Performance Analysis of Stereo Matching Engine on FPGA

From the report of Synplify Premier, the critical path is located on the forward passing function of Dynamic Programming Processor in Stereo Matcher Engine, in which 72.4MHz frequency is estimated. Therefore, the SoC is capable of handling up to the standard XGA (1024x768@ 60FPS 65MHz) video format. It is believed that new generations of FPGA or ASIC can easily handle higher resolutions of standard video based on our design.

To compare the computational performance with others' designs, MDE/s (Million Disparity Evaluation per second) is commonly used to measure the matching performance despite the implementation platform. The formula of the MDE is listed in Equation 4.4. Table 4.4 simply compares the computational performance of our design with others' works.

$$MDE/s = frame\ width \times frame\ height \times D_{max} \times FPS \quad (4.4)$$

Table 4.4: comparison of state-of-art stereo matching implementations

<i>Algorithm</i>	<i>Platform</i>	<i>Disparity</i>	<i>Frame Rate</i>	<i>MDE/s</i>
Jin et al. 2010[21]	FPGA Virtex-4	64	640X480 @ 230	4521
Our Method	FPGA Stratix III	64	1024X768 @ 60	3019
John el al. 2006 [45]	Tyzx DeepSea II	52	512x480 @ 200	2600
Jacobi et al. 2010 [20]	FPGA Virtex II	64	176x144 @ 52	1420
Masrani [28]	FPGAs Transmogri er-4	64	480 x 640 @ 30	330

IMEC 3D Depth Intensity Adjustable System with Stereo Matching on FPGA

5

In this chapter, we apply the proposed Stereo Matching Engine to IMEC 3D TV system to support depth intensity adjustment. The depth adjustment method is based on synthesizing the interpolated virtual view from the stereoscopic cameras. Watching from the original left view and the interpolated virtual view, the viewer can perceive less 3D effects. To generate an interpolated view, both left and right disparity maps and image sources are required for View Synthesis Engine. Hence, we apply the proposed Stereo Matching Engine to IMEC 3D TV system for depth extraction. To support the completely system design, peripheral components include Video Adaptors, Color Space Converters, and memory hierarchy are designed.

The overview of system architecture is presented in Section 5.1. Afterward, the individual function units will be introduced in the following sections. We firstly provide a brief background of View Synthesis Engine in Section 5.2. Then we presents the design of Color Space Converter in Section 5.3. In Section 5.5, we further propose a customized memory hierarchy to support frame buffering for the temporal consistency function.

5.1 System Architecture Overview

Figure 5.1 reveals IMEC 3D TV System. The input stereoscopic sequences are processed in a pipeline manner through stream processors. The throughput is expected to match the input pixel clock in order to achieve real-time performance. The system is composed of five parts: Video Adaptors, Color Space Converters, Stereo Matching Engine, memory hierarchy, and View Synthesis Engine. Anaglyph processor is an option to adapt conventional 2D display technology. In this thesis works, we design and implement most of components by ourselves, which includes Video Adaptors, Color Space Converters, Stereo Matching Engine, and DDR Scheduler. The View Synthesis Engine is provided by NCTU and IMEC-Taiwan. And we utilize DDR2 High Performance Controller [12] to speed up the development.

5.1.1 Function Definition

The function of each component are briefly summarized in Table 5.1. More detail information will be introduced in the following sections.

5.1.2 Clock Domain Design

As shown in Figure 5.1 and Table 5.1, the proposed system architecture contains three clock domains: pixel clock, DDR local interface clock, and DDR clock.

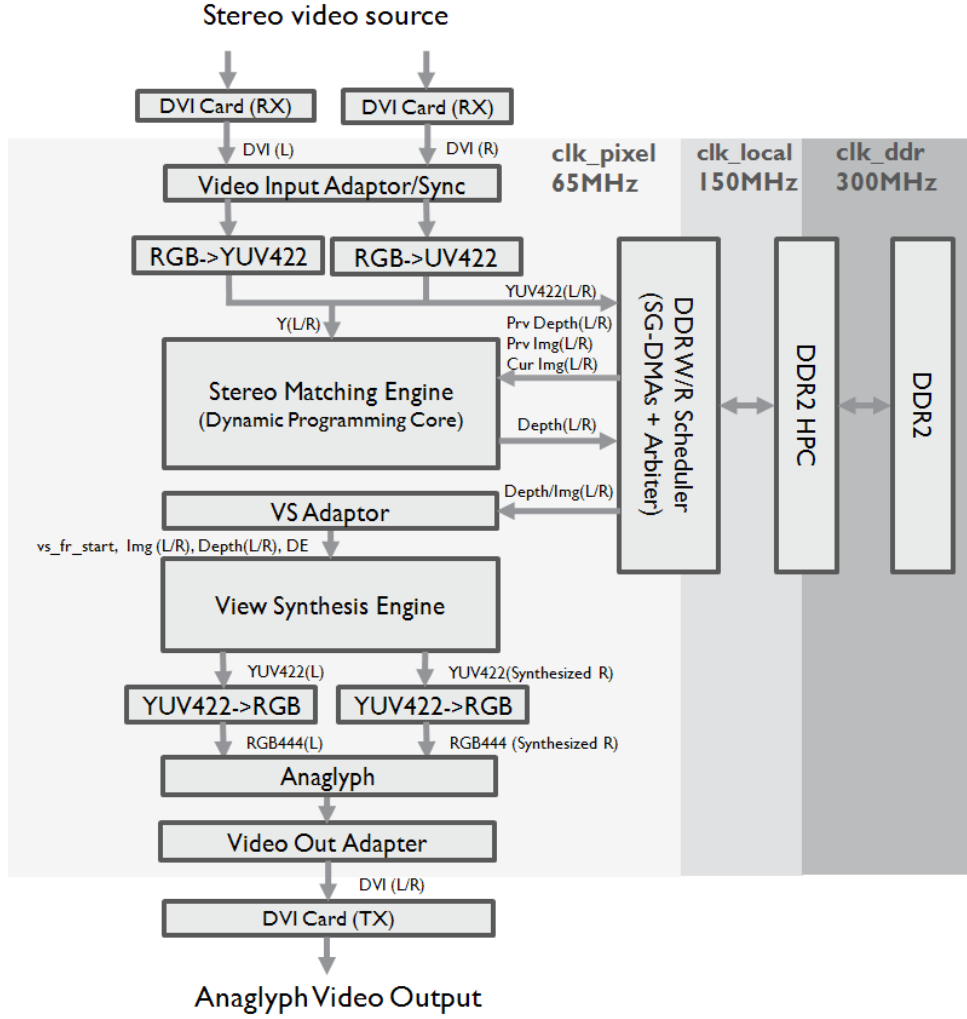


Figure 5.1: Dual DVI receivers scenario

In the Dual Port DVI inputs scenario, the pixel clock domain is synchronized with input pixel rate. Taking standard XGA video format (1024X768@60FPS 65MHz) for example, the pixel clock rate synchronizes with the input pixel clock rate in 65MHz. In order to achieve real-time processing, pipeline architecture is applied throughout complete system. However, the design of each processor unit must abide by the limitation of critical path.

The proposed Scatter-Gather(SG) DMA components of DDR Scheduler run under both pixel clock and DDR local interface clock domains. The YCbCr422 and disparity map streams are gathered in the Dual-Port RAM of SG-DMA device under pixel clock rate and will be delivered to DDR controller in DDR local interface clock rate. Vice versa for the data reading operations. Therefore, the Dual-Port RAM plays an

Table 5.1: Function definition in SoC

<i>Block</i>	<i>Function Description</i>	<i>Clock Domain</i>
Video Input Adaptor	Synchronizes stereo video sources by utilizing FIFO. The other task is to filter out the incomplete input pixels and output complete valid frame pixels	clk_pixel
RGB to YCbCr422	Converts RGB 444 to YCbCr 422 format	clk_pixel
Stereo Matching Engine (Dynamic Programming)	Extracts left and right disparity maps with the help of Dynamic Programming function	clk_pixel
DDR Scheduler	It includes multiple Scatter-Gather DMAs and an arbiter for frame buffering	clk_pixel/clk_local
DDR2 HPC	DDR2 high performance controller IP from Altera.	clk_local/clk_ddr
VS Adaptor	Fetches image and disparity map streams for View Synthesis Engine based on standard video timing	clk_pixel
View Synthesis	View Synthesis Engine generates the in-between virtual view from stereoscopic video sources (YCbCr 422 format) and disparity maps	clk_pixel
YCbCr422 to RGB	Converts YCbCr 422 back to RGB444 format	clk_pixel
Anaglyph	Generates anaglyph video for 2D TV	clk_pixel
Video Out Adapter	Generates video timing signals (hsync/vsync) and outputs 3D content	clk_pixel

important role as a clock domain bridge.

The DDR2 HPC controller [12] provides two clock rate modes for external interface: full clock rate and half clock rate. Full data rate mode captures the signal in both clock edges, whereas half rate mode captures the signal at the positive edge of clock but requires double data width. The half-rate solution allows the bus works in double bandwidth for a given number of data pins when the external logic is frequency limited. Generally, the half clock rate mode is chosen because it provides lower clock rate limitation to user interface but remain the same throughput as full clock rate mode. In the proposed system Figure 5.1, the external interface of DDR2 controller is working under 150 MHz and DDR is working under 300 MHz.

5.1.3 System On-chip Interconnection

Avalon Stream Interface (Avalong ST) protocol [11] is applied on the video processing data path because it handles the stream computational flow properly. The interface is illustrated in Figure 5.2, which contains DATA, VALID, RREADY, SOP, and EOP signals. The pin width of data signal is user-defined so it provides great flexibility for designers. The READY signal generates back pressure to Data Source block when Data Sink block is unable to accept any incoming data. The back pressure stops the pipeline

output from Data Source block. As a result, the Data Source block will also generate pressure ($READY = 0$) back to its superior block. Finally, the SOP (start of packet) and EOP (end of packet) signals are reserved to indicate the start and end points of a frame.

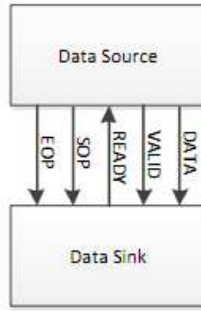


Figure 5.2: Avalon Interface

For the interface between stream processing units and DDR controller, a multi-port front end memory controller, DDR Scheduler, is proposed instead of using conventional standard bus protocol such as Wishbone, OCP, ARM, ARM and AHB. The dedicated DDR Scheduler is able to operate off-chip memory access stand alone without additional processor core. Multiple components include SG-DMA devices, Mux, and Arbiter are integrated in side of DDR Scheduler to enhance memory efficiency. Since it is mainly designed for stream processing application, the interface adapts Avalon ST protocol. Furthermore, the interface between the DDR Scheduler and DDR2 high performance controller is configured to DDR local interface [12].

5.2 Background of View Synthesis Engine

The view point synthesis engine is co-designed by NCTU and IMEC-Taiwan. It is capable of generating virtual interpolated views from stereoscopic image and depth map sequences. The view synthesis flow mainly contains three stages: depth maps forward wrapping, texture reverse wrapping, and blending/hole-filling stages. Figure 5.3 illustrates the high level architecture of View Synthesis Engine.

In the depth maps forward wrapping stage, two virtual depth views are generated separately from left and right depth maps. In texture reverse warping stage, the new virtual views are produced from mapping the texture of original image based on the virtual depth view information. In the last stage, the new generated virtual view is refined with blending and hole filling functions.

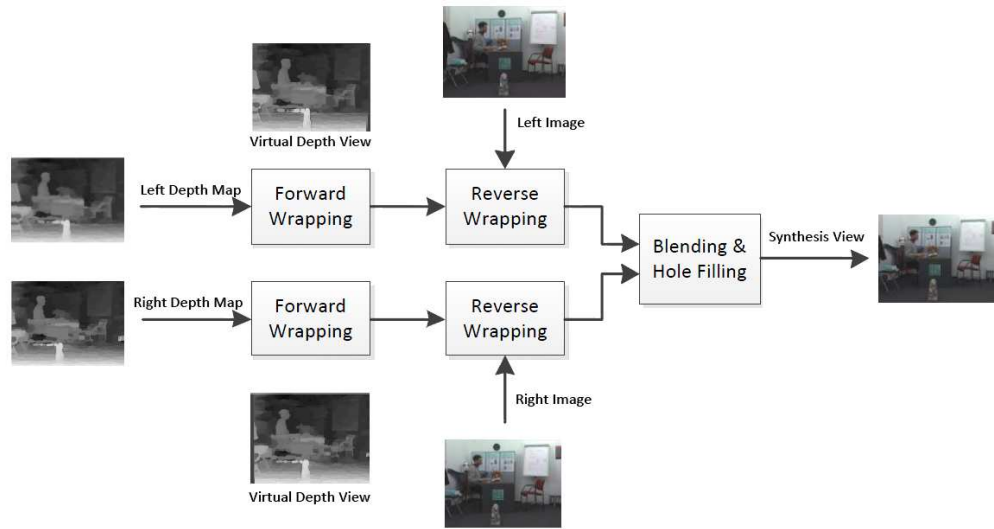


Figure 5.3: High level architecture of View Synthesis Engine

5.3 Video Adaptor Design and Implementation

Video Input Adaptor outputs synchronized left and right sequence streams, and it guarantees the output pixels start from the first pixel of a frame. Since the DVI sources are not synchronized all the time, the Video Input Adaptor is designed to coordinate left and right DVI input streams with FIFO. The other task of Video Input Adaptor is to filter the incomplete pixel signals of a frame in the initial stage. Figure 5.4 illustrates the standard VGA signal format. In the initial stage, the state machine of Video Input Adaptor waits for the positive edge of **v_sync** signal in order to confirm the beginning pixel of a frame. Afterward, the following valid pixel data are regarded as valid frame pixels for output.

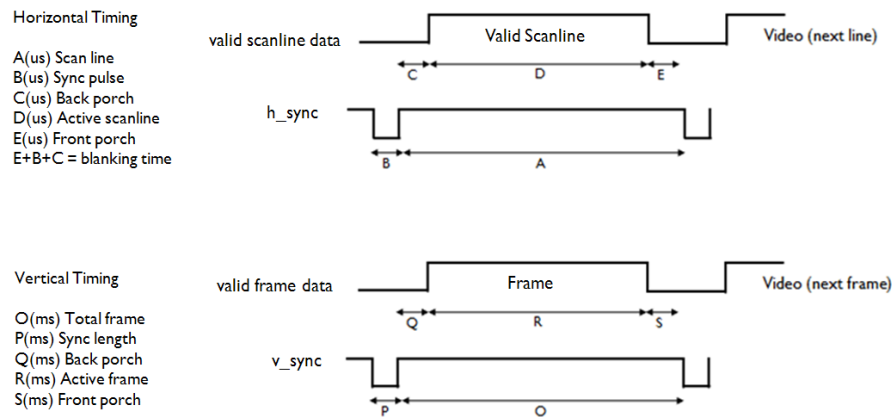


Figure 5.4: Standard VGA signal format

View Synthesis Adaptor (VS Adaptor) is designed to fetch out both stereo disparity map and image sequences (YCbCr422) from DDR Scheduler for View Synthesis Engine. The data fetching timing is generated based on the valid pixel signal of standard VGA format. Because the pre-fetch mechanism of DDR Scheduler, VS Adaptor can acquire the disparity and YCbCr422 pixel streams in the first cycle without delay.

Finally, the Video Output Adaptor exports the synthesized stereoscopic sequences in standard VGA timing to DVO port for display.

5.4 Color Space Converter Design and Implementation

YCbCr image format has been utilized in many popular video applications such as MPEG1-4, H.261-4, and JPEG etc. In the 3D depth intensity adjustable system, Stereo Matching Engine takes the luminance information of pixel for depth extraction, and Viewpoint Synthesis Engine also processes YCbCr formats stereoscopic sequences. Since the input sequences from DVI are RGB format, the Color Space Converters are required.

In Sub-section 5.4.1, we provides the background of color space conversion. Before proposing the hardware design, the background of floating point to integer mathematic approaches will be introduced in Sub-section 5.4.2. In Sub-section 5.4.3, the hardware designs of color space converter will be presented.

5.4.1 Background of Color Space Conversion

The YCbCr model defines a color space that contains one luminance (Y) and two chrominance (Cr and Cb) elements [39]. More specifically, Y represents perceptual brightness, and Cr and Cb represent blue-luminance and red-luminance differences. Figure 5.5 illustrates YCbCr information that are extracted from RGB format image. In reality, human eye is more sensitive to the luminance variation of an image, whereas it is poor to differentiate subtle color variation. Therefore, the chrominance information, Cr and Cb, could be down-sampled. Hence, YCbCr color format is broadly used in industry.

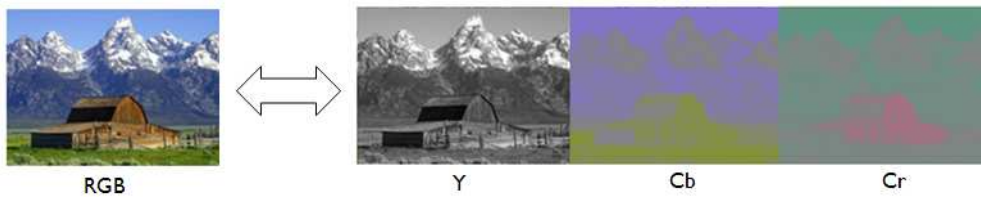


Figure 5.5: Example of RGB and YCbCr Format

YCbCr color space was defined in ITU-R 601 [18] and ITU-R 709 [19] standards for worldwide digital component video format. In the scaled YCbCr color space format, Y is in the range of 16 to 235, and Cb and Cr are in the range of 16 to 240.

YCbCr format can be converted from RGB source. There are two commonly used standards for color space conversion: ITU-R BT. 601 and ITU-R BT. 709. ITU-R BT.601 [18] defines its coefficient vector for Standard TV, whereas ITU-R BT.709 [19] possesses different coefficient vector for High-Definition TV. Equation 5.1 is an example that demonstrates RGB to ITU-R 601 full-range YCbCr format. Equation 5.2 is that inversion vector to convert the YCbCr signal back to RGB format.

1. RGB to Full Range YCbCr Format

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (5.1)$$

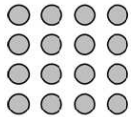
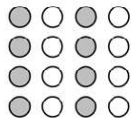
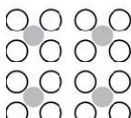
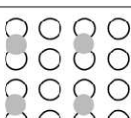
2. Full Range YCbCr to RGB Format

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0.114 \\ 1 & -0.343 & -0.711 \\ 1 & 1.765 & 0 \end{pmatrix} \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} \quad (5.2)$$

In YCbCr format, A:B:C notation is used to describe the sample factor of Cb and Cr (chrominance) components relative to Y (luminance) component. The first digit indicates the sub-sampling factor of luminance component on vertical domains. The second digit specifies the sub-sampling factor of Cb and Cr components on horizontal domains. The third digit represents the sub-sampling factor of Cb and Cr on vertical domains. Table 5.2 lists common used YCbCr formats in A:B:C notation.

Unfortunately, down-sampling and up-sampling chrominance components introduce the artificial colour information which doesn't exist in the original image; therefore the image quality will be slightly different after the conversion. In here, we compare three common used interpolation methods for chrominance up-sampling: nearest neighbor interpolation, bicubic interpolation, and fractal interpolation. Firstly, the nearest neighbor interpolation approach duplicates the information of adjacent pixels. This method is the most hardware efficiency option. However, this tends to make the chrominance channel looks blocky, and accentuates the jaggedness. Secondly, bicubic interpolation is more sophisticated and produces smoother edges than bilinear interpolation. It calculates the missing gap position by interpolating four adjacent pixels with weight. Finally, the fractal interpolation [34] is broadly used for enlarging object as retaining the shape. The result is more cleaner, sharper edges, less halos and blurring around the edges than bicubic interpolation would do. In this thesis work, we will only adopt the nearest neighbor interpolation method to implement the up-sampling function.

Table 5.2: Common used YCbCr formats based on A:B:C notation

Sample Position	A:B:C Notation	Description	Data Size
	YCbCr 4:4:4	No down sampling of both luminance and chrominance channels	Every pixel requires 24 bits
	YCbCr 4:2:2	Chrominance channels (Cb and Cr) 2:1 horizontal down sampling only.	Each pair of pixels requires 32 bits
	YCbCr 4:2:0 (MPEG-1 scheme)	Chrominance channels (Cb and Cr) 2:1 horizontal down sampling in the middle, with 2:1 vertical down sampling.	Each four-pixel block requires 48 bits
	YCbCr 4:2:0 (MPEG-2 scheme)	Chrominance channels (Cb and Cr) 2:1 horizontal down sampling, with 2:1 vertical down sampling.	Each four-pixel block requires 48 bits

5.4.2 Background of Floating Point to Integer Mathematic Approaches

Fixed-point approach is preferable than floating-point when implementing digital system because of computation simplicity. Fixed-point approximation is a common technique to operate floating point calculation in integer format. The floating-point variables are firstly scaled up and rounded to integers. So the following calculations can be fully operated under numerical mathematics. After the numerical calculations, the result will be rounded and scaled down.

Recalling the RGB-YCbCr conversion Equation 5.1 (ITU-R 601 Full Range Format), RGB signals are fixed point variable and the coefficient matrix are also represented in floating point. The floating point coefficients are good candidate to implement integer approximation. Thus, the RGB-YCbCr conversion equations can be rewritten as the following two equations where the original floating-point coefficients are scaled up to constant numbers by multiplying 256. After the numerical calculation, the final sum of each channel will be divided by 256. In digital system, the divide operand can be achieved by bit shifting technique.

RGB to YCbCr Conversion

Range

$$\begin{aligned}
 Y &= \text{clip}(\text{rounding}(77 * R + 150 * G + 29 * B) >> 8 + 0) & Y &= [0 - 255] \\
 Cb &= \text{clip}(\text{rounding}(-43 * R - 85 * G + 128 * B) >> 8 + 128) & Cb &= [0 - 255] \\
 Cr &= \text{clip}(\text{rounding}(128 * R - 107 * G - 21 * B) >> 8 + 128) & Cr &= [0 - 255]
 \end{aligned}$$

where the clip function returns 255 when the sum excesses 255; it returns 0 when the sum less than 0. In equation Y, 0 is for the base range; in equation Cb and Cr, 128 is to ensure the final

integer values are positive.

YCbCr to RGB Conversion	Range
$CB = Cb - 128$	
$CR = Cr - 128$	
$R = \text{rounding and clip}((128 * Y + 358 * CR))$	$R = [0 - 255]$
$G = \text{rounding and clip}((128 * Y - 88 * CB + 182 * CR))$	$G = [0 - 255]$
$B = \text{rounding and clip}((128 * Y + 452 * CB))$	$B = [0 - 255]$

When applying integer approximation, rounding is an important factor which affects the computational precision. The commonly used rounding methods are round towards zero and round half up. Round towards zero (truncate or round away from infinity) method directly truncates the fraction part and keeps the integer portion. In contrarily, round half up method rounds towards nearest neighbor unless both neighbors are equidistant, in which case round up. For example, 11.5 will be rounded to 12. This is the rounding mode that is typically taught in schools. In next chapter, we will further compare the above mentioned rounding methods to the quality losses during color space conversions.

5.4.3 Hardware Architecture Design and Implementations

Figure 5.6 illustrates the computation flow of the color space conversion in the 3D Depth Intensity Adjustable System. The incoming pixel sequences in RGB format will be converted into YCbCr color space in the beginning. The luminance component Y will support Stereo Matching Engine. Simultaneously, the YCbCr streams will be further down-sampling to YCbCr 4:2:2 format and stored in DDR2. In the last stage of the IMEC 3D TV System, the YCbCr 4:2:2 format streams will be converted back to RGB format for display.

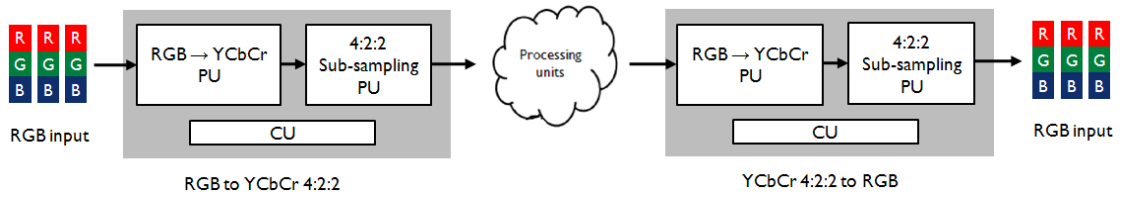


Figure 5.6: Color space conversion flow

Equation 5.4.2 is rewritten in Equation 5.4.3 which turns the round half up function into calculation. Then the hardware structure is showed in Figure 5.7.

Since the proposed RGB-YCbCr hardware structure requires nine multipliers, we further consider the design of constant multiplier in two approaches to reduce the hardware utilization and latency.

HW:RGB to YCbCr Conversion

Range

$$\begin{aligned}
 Y &= \text{clip}((77 * R + 150 * G + 29 * B + 128) \gg 8) & Y &= [0 - 255] \\
 Cb &= \text{clip}((-43 * R - 85 * G + 128 * B + 32768) \gg 8) & Cb &= [0 - 255] \\
 Cr &= \text{clip}((128 * R - 107 * G - 21 * B + 32768) \gg 8) & Cr &= [0 - 255]
 \end{aligned}$$

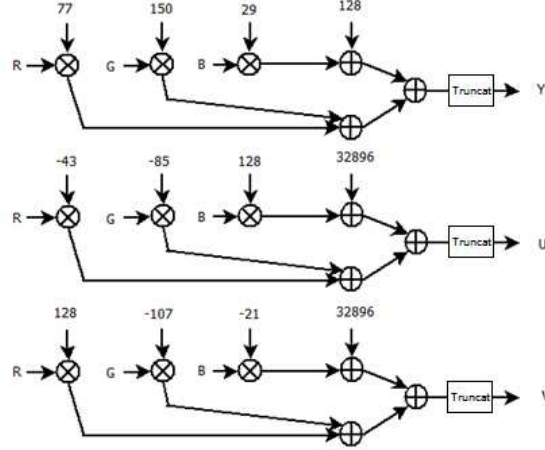


Figure 5.7: RGB to YCbCr Processor Unit

Firstly, constant multiplier can be simply illustrated as shift and add/sub scenario. Since the constant is fixed, it is possible to implement one constant multiplier by several shifts and add/sub operands in parallel. For example, constant 77 is the sum of 64, 8, 4, and 1. To multiply R and number 64, 8 and 4 are able to be computed with bit shift technique in digital system. Therefore, we can apply the shift and add/sub scenario on all constant multipliers in RGB-YCbCr processor unit as Figure 5.8. The main advantage of shift and add/sub architecture is the low hardware utilization.

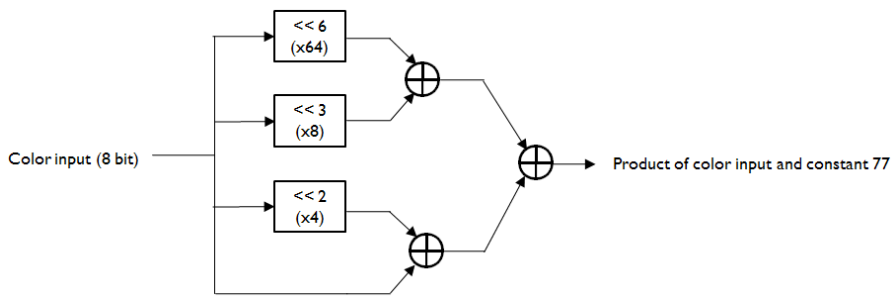


Figure 5.8: Constant multiplier implementation (77) with Shift and Add/Sub architecture

Unfortunately, the constant multiplier in the first proposed design is fixed and lack of flexibility. The conversion vector (coefficients) will be unchangeable once the standard is predefined. Therefore, we introduces look up table (LUT) approach

into the design of constant multiplier. One approach is utilized full look up table to store the complete product results of color value and fixed point coefficient but it takes tremendous on-chip memory. To reduce the on-chip memory requirement, the solution is to improve Shift-and-add Multiplication method [3] with two look up table [24]. Figure 5.9 shows the 8 bit multiplicand can be separated to upper 4 bit nibble and lower 4 bit nibble and perform multiple operations with 4x8 LUTs. Hence, each look up table only has 4 bit address which points to 16 entries. Figure 5.10 illustrates the LUT configuration for constant 77. Finally, we sum up the two products with a 12 bit adder. The sum of adder is assigned to upper 12 bits of final product result. The full hardware architecture of constant multiplier is implemented in Figure 5.11. Although this approach takes a extra hardware utilization than the first approach, it keeps the flexibility to cope with different standards in run-time configuration.

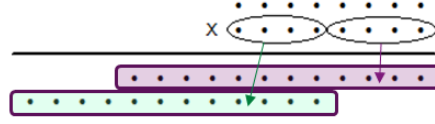


Figure 5.9: upper 4 bit nibble and lower 4 bit nibble multiplication

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0	77	154	231	308	385	462	539
8	616	693	770	847	924	1001	1078	1155

Figure 5.10: 16 Entries of Look Up Table for multiplying constant 77

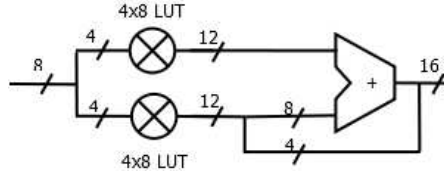


Figure 5.11: 8x8 Constant multiplier with 4x8 LUTs

5.5 Memory Hierarchy Design and Implementation

Although FPGA/ASIC is capable of achieving high computational power with the help of parallelism, the on-chip memory is still relatively limited because of cost concern. Off-chip memory is a solution for the case of great memory consumption. Therefore, this section proposes a memory hierarchy to support the 3D Depth Intensity Adjustable System implementation.

Sub-section 5.5.1 firstly analyzes the off-chip bandwidth and critical latency requirements for the 3D Depth Intensity Adjustment System. In Section 5.5.2, a memory hierarchy design which includes SG-DMA, Arbiter, and DDR controller is proposed to support stream processor units.

5.5.1 Memory Architecture Analysis for Stream Processing

In this thesis work, we utilize DDR2 SDRAM as off-chip memory. The DDR2 SDRAM bandwidth can be formulas as Equation 5.3 [1].

$$Bandwidth = SDRAM \text{ Bus Width} \times 2 \text{ Clock Edges} \times Frequency \text{ of Operation} \times Efficiency \quad (5.3)$$

When the DDR SDRAM component has 64 bit bus width working under 300 MHz, the maximum available bandwidth achieves $64 \times 2 \times 300MHz \times 100\% = 38.4Gbps$ in theory. However, the bus efficiency is alternative depending on the factors include command latency, refresh period, and access addresses. To increase the memory access efficiency, we summarize the suggestions from Altera[1] as following:

1. Accessing continuous addresses is preferable than random addresses. This related to data mapping.
2. Series of write or read commands is preferable than interlaced write/read operations.
3. Accessing different row introduces extra latencies because active command has to be executed again.
4. Well-controlled refresh timing contributes to better efficiency.

In the proposed memory hierarchy architecture Figure 5.12, seven Scatter-Gather DMA devices access to the DDR controller: two data writing SG-DMA and five data reading SG-DMA devices. The required throughput can be calculated based on Equation 5.4. Table 5.3 is an example shows the total throughput when processing XGA video format (1024x768 @60FPS). The sum of required throughput is 8.305G bps which consumes 22% of total bandwidth.

$$Throughput = Frame \text{ Rate} \times Frame \text{ Width} \times Frame \text{ Height} \times Pixel \text{ Format} \times Channel \text{ Number} \quad (5.4)$$

5.5.2 Memory Architecture Design for Stream Processing

In the IMEC 3D TV SoC, off-chip memory is required to implement frame buffering. Since temporal disparity consistency mechanism is presented to strengthen the consistency between disparity maps, the disparity value and luminance value of the pixel from previous and current frames are required the calculation. Furthermore, the disparity

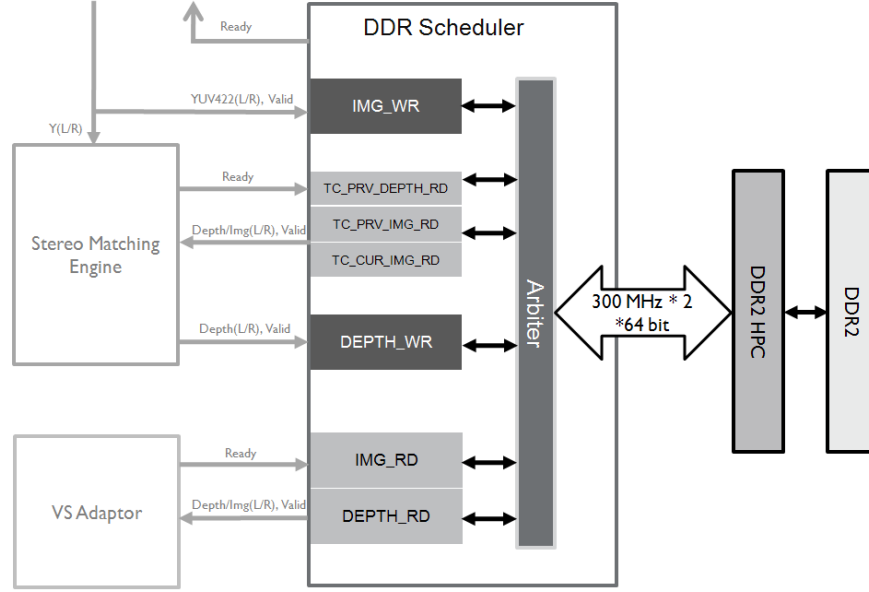


Figure 5.12: Proposed Memory Hierarchy

Table 5.3: System memory breakdown

<i>Memory Access Components</i>	<i>Throughput (G bps)</i>	<i>Critical Latency</i>
IMG_WR	$60 \times 1024 \times 768 \times 16 \times 2 = 1.51$	Internal buffer length $\times 8$
TC_PRV_DEPTH_RD	$60 \times 1024 \times 768 \times 8 \times 2 = 0.755$	Internal buffer length $\times 16$
TC_PRV_IMG_RD	$60 \times 1024 \times 768 \times 16 \times 2 = 1.51$	Internal buffer length $\times 8$
TC_CUR_IMG_RD	$60 \times 1024 \times 768 \times 16 \times 2 = 1.51$	Internal buffer length $\times 8$
DEPTH_WR	$60 \times 1024 \times 768 \times 8 \times 2 = 0.755$	Internal buffer length $\times 16$
IMG_RD	$60 \times 1024 \times 768 \times 16 \times 2 = 1.51$	Internal buffer length $\times 8$
DEPTH_RD	$60 \times 1024 \times 768 \times 8 \times 2 = 0.755$	Internal buffer length $\times 16$
Total	8.305	

map and image information are needed again in View Point Synthesis Processor unit. Therefore, we propose a memory hierarchy with customized Scatter-Gather DMAs, Arbiter, DDR controller, and off-chip SDRAM (DDR2) to support the SoC. The memory hierarchy is designed to access multiple frame buffers simultaneously without delay.

Data locality has to be analyzed first. The goal is to keep the frequently used data on chip in order to reduce the off-chip bus overhead. Slide window technique is a solution which is broadly used in our stream processor unit designs such as median filter, support region builder, and disparity voting units. The processing data includes horizontal and vertical direction of image pixels. The line buffers(2-Port BRAM) in the on-chip memory architecture are used to store and propagate stream data in a circular manner. This architecture is able to serve the pixels include vertical domain. Figure 5.13 is an example that shows the on-chip memory architecture for 5x5 slide

window application [31]. As the demonstration in Figure 5.14, each scanline data will be reused 4 times when the slide window shifts to next row.

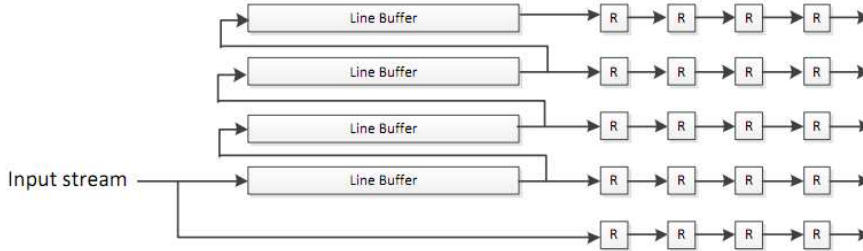


Figure 5.13: 5x5 slide window operation

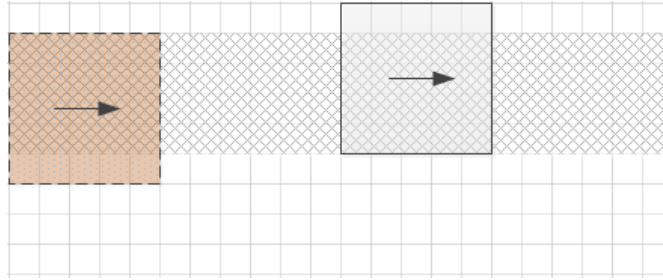


Figure 5.14: Example of data reuse: the weaved texture region represents the data reuse zone in slide window application

When large storage space is required and impossible to keep them locality, off-chip memory structure will be needed. Figure 5.15 shows the proposed memory hierarchy for frame buffering in this thesis work. This memory hierarchy contains four parts: Scatter-Gather DMAs (SG-DMA), Arbiter, memory controller, and off-chip memory.

Dual-Port block RAM plays an important role in the proposed memory hierarchy. On the one hand, the transfer latency between function units and off-chip memory can be hid by a series of burst operations with the help of Dual-Port RAM. On the other hand, the Dual-Port RAM is able to deal with the data crossing tasks. To achieve higher off-chip memory bandwidth, the off-chip memory controller usually works in a higher frequency. Memory can execute both write and read operations in different clock domains. In addition, the Dual-Port RAM can be regarded as buffer for data packing and reordering, which makes the off-chip memory bandwidth be efficient.

To construct the memory hierarchy for IMEC 3D TV system, latency constraints have to be explored. In order to achieve real-time performance, stream data are computed throughout processor units in pipeline architecture. Any pending will produce back pressure through the system. It means the frame buffer should be able to store input stream continuously. In the other hand, frame buffers are designed to serve

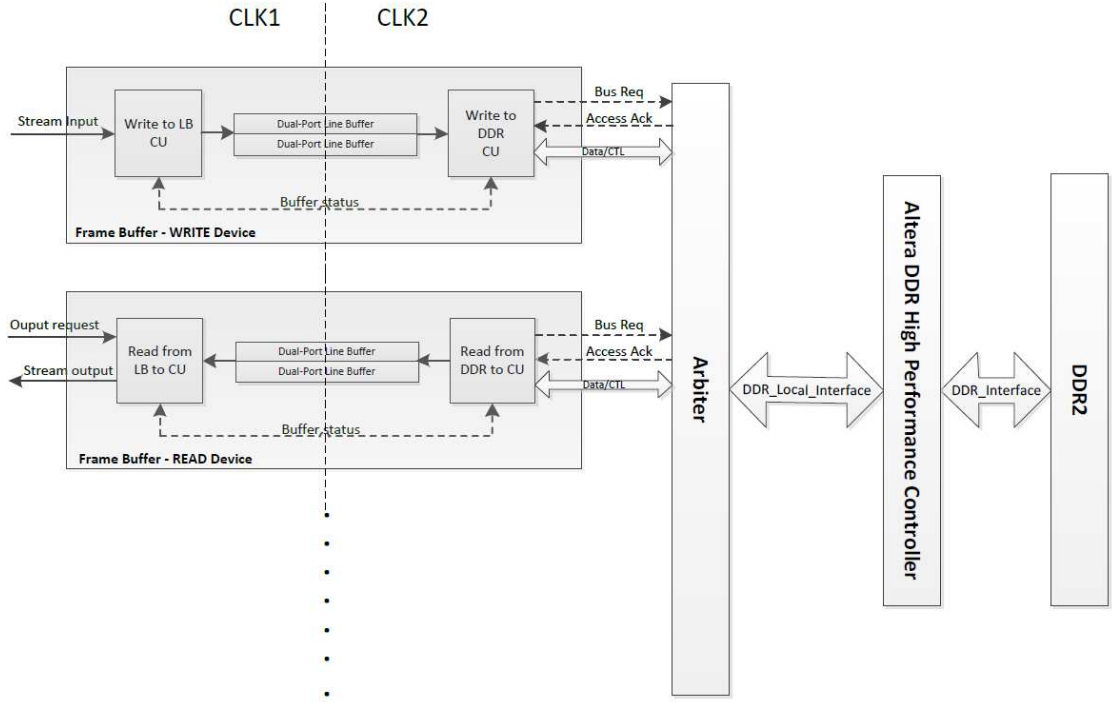


Figure 5.15: Proposed memory hierarchy for frame buffering

stream data in the first cycle with zero latency whenever the function unit asserts data request signal. In order to achieve the latency constraint, line buffers (Dual-Port RAM) and pre-fetch technique are used to hide the latency.

SG-DMA Design

Figure 5.16 shows a SG-DMA design to handle data writing task for frame buffer function. It contains three parts: Write_to_lb control unit, Dual_port_line_buffer, and Write_to_DDR control unit. This structure passes input data crossing two different clock domains. The Write_to_lb control unit concatenates consecutive input pixels to the data width that fits for burst and stores it in the Dual_port_line_buffer in pixel clock domain (CLK1). The data width of the Dual_port_line_buffer is set to the product of DDR burst length and DDR data width. Figure 5.17 shows how the image pixels(YCbCr 4:2:2) and disparity map pixels be concatenated. In the YCbCr 422 scenario, each address space of Dual_port_line_buffer contains 8 sets of YCbCr 422 stereoscopic pixels. In the disparity map pixel scenario, each address space of Dual_port_line_buffer contains 16 sets of disparity stereoscopic pixels. When any one of Dual_port_line_buffers is full, Write_to_DDR control unit launches bus request signal to Arbiter. The Write_to_DDR control unit works in the way as a dependent DMA controller. After the writing authority is confirmed by Arbiter, the Write_to_DDR control unit will start to transfer the data from Dual_port_line_buffer to DDR controller in a higher clock rate (CLK2) to gain more off-chip memory bandwidth. All in all, the proposed SG-DMA writing mechanism guarantees that the input data can be stored

into off-chip memory continuously without pending.

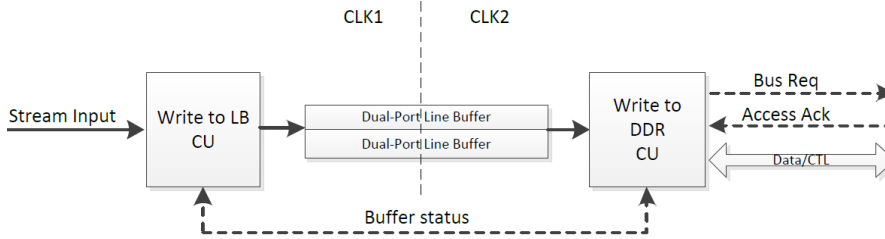


Figure 5.16: SG-DMA architecture for write function

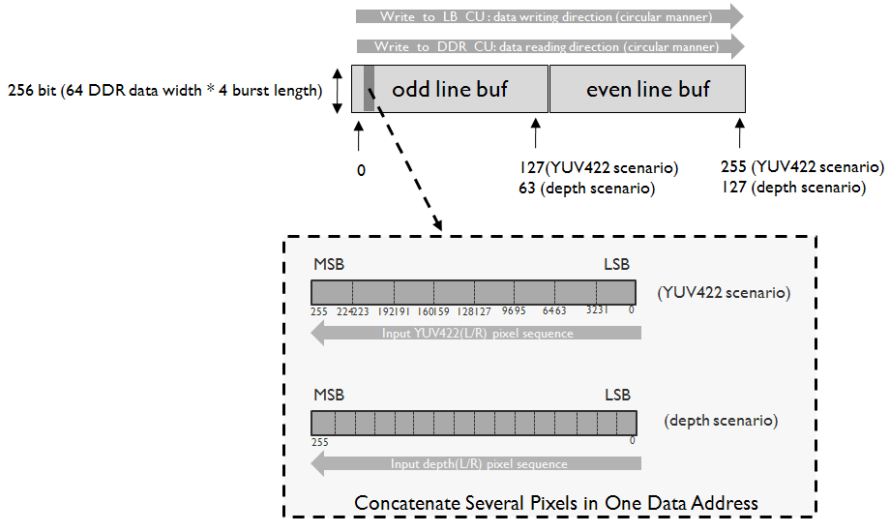


Figure 5.17: Example of data concatenation

Figure 5.18 shows a SG-DMA design to handle data pre-fetch task for frame buffer function. It contains three parts: Read.Line.Buffer control unit, Dual.port.line.buffer, and Read.from.DDR control unit. This structure pre-fetches frame data from DDR then outputs in stream whenever processing unit requests. Firstly, Read.from.DDR control unit will preload one scan line of concatenated stream data from DDR to the Dual.port.line.buffer after the first scan line data have been bursted into off-chip memory. When external function unit requires the stream data from frame buffer, the Read.from_lb control unit will unpack the concatenated data from Dual.port.line.buffer and outputs the data stream pixel by pixel. In the meaning while, the Read.from.DDR control unit will monitor the condition of Dual.port.line.buffer devices. If any one of Dual.port.line.buffer devices is empty, the Read.from.DDR control unit will send a pre-fetch request to the Arbiter in order to get the access authority of off-chip memory. The proposed SG-DMA reading mechanism guarantees that the requested external function units will never suffer from data starving.

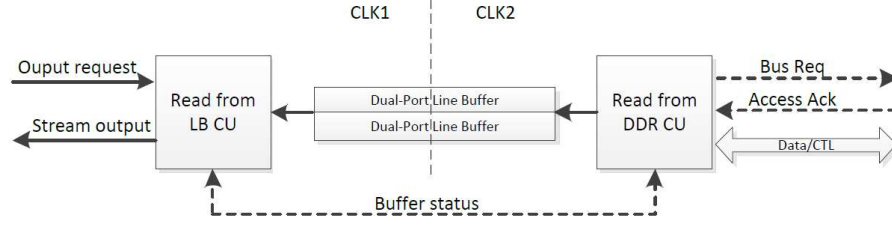


Figure 5.18: SG-DMA architecture for read function

The access addresses to off-chip memory are generated from the address generator which is placed in both Write_to_DDR control unit Figure 5.16 and Read_from_DDR control unit Figure 5.18. The address generator cooperates with a state machine. It makes the SG-DMA components capable of working independently without the control of CPU. In the proposed SoC, we choose DDR2 to implement the off-chip memory. The address of DDR2 includes row address, bank address, column address and chip select signals. Each row is divided into banks, and each bank is composed of columns. Currently one dimension linear address and two dimensions block-based address are the most commonly-used address generation patterns in video processing applications [25]. In one dimensional address generator scenario, the 1-D address generator automatically generates addresses linearly by giving the offset address and access length. The 2-D address generator is designed for block based processing units such as MPEG 2 motion estimation and DCT. Group writing or reading in consecutive addresses is preferable because the operations such as accessing different row or interlaced commands introduces additional latencies. Fortunately, 1-D address generator has already fulfilled the requirement of frame buffer function in this thesis work. In order to make multiple frame buffers working simultaneously, each frame buffer is assigned to an offset address (index address) and an address range (data length) in off-chip memory. Therefore, the address generator in each SG-DMA component (WR/RD) can generate the physical address by summing up the offset address and a counter. Of course the range of the counter is defined based on the frame size and data width. The detail of memory allocation will be further introduced in the next section.

Arbiter

Arbiter is designed to manage the access schedule of SG-DMA devices to off-chip memory. More specifically, the Arbiter decides which SG-DMA device be served first. The most commonly used scheduling policies are round-robin, first in first serve and fixed priority mechanisms. Round-robin policy repeatedly checks all memory bus requirements and provides even opportunity to all request devices. However, there is an uncertainty to predict and guarantee the waiting latency for the latency sensitive device. In contrarily, fixed priority policy assigns priority to each off-chip memory bus request device based on its latency and bandwidth constraints. However, the lower priority devices are easily suffered from data starving when the devices with higher priority occupy the channel all the time. [25] In the proposed Arbiter design, both priority and

round-robin policies are implemented in RTL code.

Memory Hierarchy Interconnection

1. Interface between SG-DMA and Arbiter

In order to reduce the system complexity, SG-DMA devices and DDR controller are connected directly through an Arbiter. We abandon the conventional standard bus design such as AMBA, Avalon, Wishbone, etc. Instead, the Arbiter not only responses of managing access schedule but also in charge of switching the channel (DDR_Local_Interface) between multiple SG-DMA devices and DDR controller 5.19 with MUX.

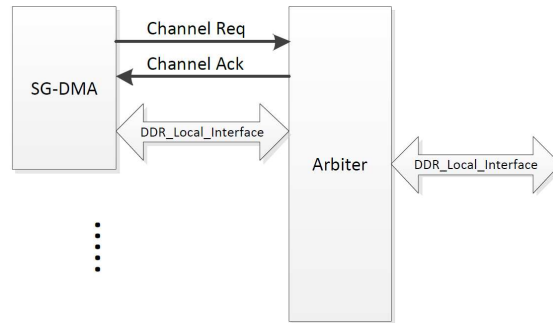


Figure 5.19: The interface between SG-DMA and Arbiter

The interface between SG-DMA devices and Arbiter abides by Virtual Component Interface (VCI) protocol. The Virtual Component Interface is an interface rather than a bus. The design follows request-response protocol, contents and coding for the transfer of requests and responses. Flexibility and adaptive are the main advantages of using VCI protocol. To deal with the interface between SG-DMA devices and Arbiter, a handshaking procedure is illustrated in Figure 5.20.

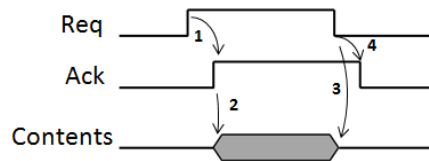


Figure 5.20: Handshaking protocol between SG-DMA and Arbiter

- (a) SG-DMA asserts request signal for the access authority of DDR_local_Interface channel
- (b) When the channel is available, arbiter assigns memory access acknowledgment to the SG-DMA request device. After the SG-DMA possesses the chan-

nel authority, it starts to transmit memory read/write commands to DDR controller.

- (c) After SG-DMA finishes memory read/write operations, the request signal will be retracted.
- (d) Arbiter releases the acknowledgement and ready to cope with next request.

2. Interface between stream processors and SG-DMA

The interface between stream processors and SG-DMA devices abides by Avalon ST protocol [11] in Figure 5.21. The main advantage of using this structure is the Ready signal indicates back pressure to the source unit. When the Sink unit is readied to accept data, the Ready signal is asserted. Otherwise the Ready signal will be de-asserted to pause the data sending from its source port. When the SG-DMA is configured for writing off-chip memory, it is defined as a data sink device and the external stream processor is defined as a data source device. In contrarily, SG-DMA is defined as a data source device and the external stream processor is defined as data sink device when the SG-DMA is configured for reading off-chip memory .

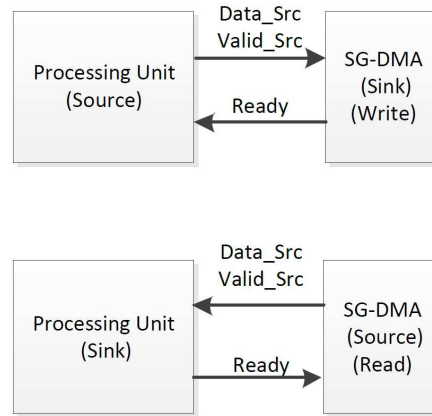


Figure 5.21: Avalon stream interface between processing unit and SG-DMA

Memory Allocation

The memory allocation in the off-chip memory (DDR2 SDRAM) contains two parts: stereo image frame buffer and disparity map buffer. The memory spaces for these two frame buffers are calculated in the following:

1. Image (L/R) frame buffer

The memory space for image frame buffer can be calculated in Equation 5.5. Taking standard XGA video for example, it requires $2 \times 2 \times 1024 \times 768 \times 16 = 50331648$ bits memory space.

$$\begin{aligned} \text{Image(L/R) Frame Buffer Space} = & 2(\text{current and previous frame}) \times \\ & 2(\text{stereo left and right channels}) \times \text{Frame Width} \times \text{Frame Height} \\ & \times 16\text{bit(YCbCr422 Pixel Format)} \end{aligned} \quad (5.5)$$

Assuming that the data width of SDRAM is 64 bits, we can concatenate two sets of left and right YCbCr 422 pixel pairs into one memory address. Therefore, the required memory space is calculated in Equation 5.6. Taking standard XGA video for example, a range of $2 \times 2 \times 1024 \times 768 \times 16/64 = 786432$ physical memory addresses are needed for buffering two frames.

$$\begin{aligned} \text{Image Frame Buffer Address Range} = & 2(\text{current and previous frame}) \\ & \times 2(\text{stereo left and right channels}) \times \text{Frame Width} \times \text{Frame Height} \\ & \times 16\text{bit(YCbCr422 Pixel Format)} / 64(\text{DDR2 data width}) \end{aligned} \quad (5.6)$$

2. Disparity map (L/R) frame buffer

The memory space for disparity map frame buffer can be calculated in 5.7. Taking standard XGA video for example, it requires $2 \times 2 \times 1024 \times 768 \times 8 = 25165824$ bits memory space.

$$\begin{aligned} \text{Disparity Map(L/R) Frame Buffer Space} = & 2(\text{current and previous frame}) \\ & \times 2(\text{stereo left and right channels}) \times \text{Frame Width} \times \text{Frame Height} \\ & \times 8\text{bits}(0 - 255 \text{ disparity range}) \end{aligned} \quad (5.7)$$

Assuming that the data width of SDRAM is 64 bit, we concatenate four sets of disparity pixel pairs that from left and right channels into one memory address. Therefore, the required memory address space can be calculated as Equation 5.8. Taking standard XGA video for example, a range of $2 \times 2 \times 1024 \times 768 \times 8/64 = 393216$ physical memory addresses are needed for buffering two frames.

$$\begin{aligned} \text{Disparity Frame Buffer Address Range} = & 2(\text{current and previous frame}) \\ & \times 2(\text{stereo left and right channels}) \times \text{Frame Width} \times \text{Frame Height} \\ & \times 8\text{bits}(0 - 255 \text{ disparity range}) / 64(\text{DDR2 data width}) \end{aligned} \quad (5.8)$$

Figure 5.22 is an example that shows the address generating patterns.

To generate the physical access address for a frame buffer, the offset address is accumulated with the burst size to gain the efficiency. If the burst size is 4 and the data width is 64 bit, DDR controller will bursts 256 bits into 4 addresses from/to SDRAM in one single reading/writing command. Then the accumulated result can be further mapped on SDRAM address. In general, SDRAM address includes chip select, row, bank, and column addresses. It is prefferable to perform group writing or

reading commands consecutively but avoiding row changes in order to achieve higher off-chip memory bus efficiency. On the one hand, row switching should be avoid, which introduces extra latencies. On the other hand, it is possible to refresh other banks when accessing one bank in the same row.

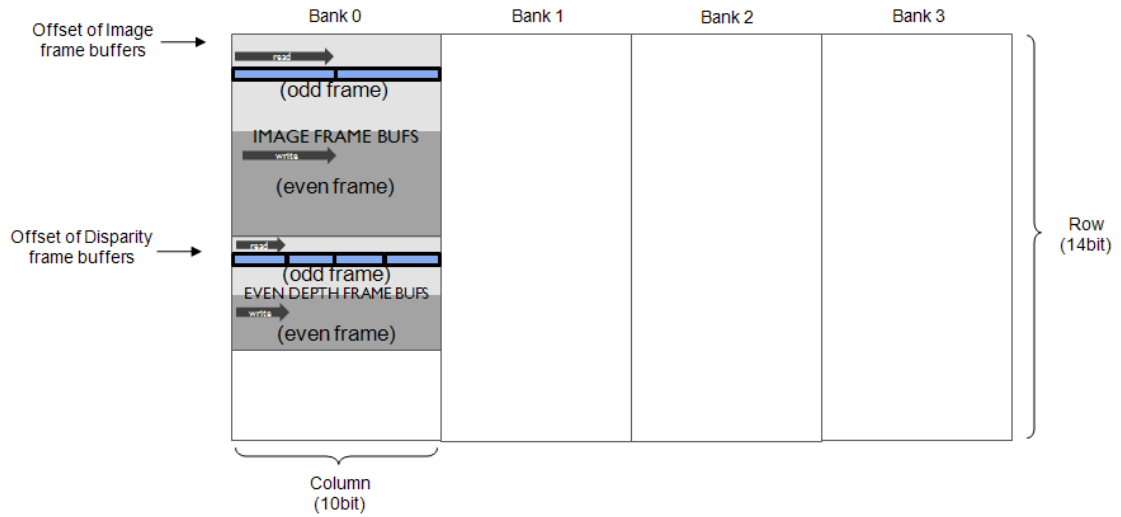


Figure 5.22: Example of address generation pattern

IMEC 3D TV SoC Evaluation and Experimental Result

6

In Section 6.1, the design of color space converter is firstly explored and evaluated individually. Then we will evaluate the IMEC 3D TV system SoC architecture at the system level. Section 6.2 will measure the quality of interpolated virtual view with true interpolated camera source in PSNR. In addition, we further analyze the temporal quality of the interpolated video sequences in TPSNR. Then the hardware usage of individual components are estimated in Section 6.3. In Section 6.4, the real-time performance of the system is assessed based on the critical path of the entire SoC system and the bandwidth of off-chip memory.

6.1 Color Space Converter Design Evaluation

This part evaluates the Color Space Converter designs in their quality, hardware resource, and latency aspects.

6.1.1 Quality Evaluation

This sub-section evaluates the quality degradation during color space conversions. After the RGB-YCbCr-RGB color space conversions, some color information will lose because of the round-off errors of floating point to integer. If the integer approach adopts a lower scaling resolution to implement floating math calculation, the result will be far from the floating point math approach. In addition, video quality degrades when applying down-sampling and up-sampling conversions. Although the luminance component is kept, parts of the chroma (Cb and Cr) information are discarded during down-sampling.

Table 6.2 illustrates the evaluation statistic of different RGB-YCbCr-RGB approaches in PSNR. The PSNR is measured by comparing original input image and inverted output image. The higher PSNR means the less loss of color information. In general, the PSNR is required to be more than 30db-40db so that the human eye will not easily notice the degradation of image quality. This table exams eight approaches based on ITU-R 601 and ITU-R 709. As well, five test images are chosen and shown on Table 6.1. We chose the frequently-used test picture, Lena, for image processing evaluation. The rest of four images (Tsukuba, Venus, Teddy, and Cones [44]) are the test sets from stereo matching society. Of course theose pictures are not the only candidates for PSNR measurement. The test set can be expanded to video so that the PSNR can be computed in average value.

The result of Table 6.2 shows that the accuracy of all kind of YCbCr 4:4:4 approaches without chroma sampling. The difference between the original image and the image

Table 6.1: Test sets



after conversion is hardly noticeable. It also shows that the wider range of pixel (0-255) achieves slightly higher accuracy than the lower range of pixel (16-235-240). Finally, it is found that chroma sampling (YCbCr 4:2:2 and YCbCr 4:2:0) conversion causes tremendous quality degradation to test images. The more chroma samples that are excluded during sub sampling, the more color information is lost, and the lower the quality of the final converted image.

Table 6.2: PSNR evaluation for different color space conversion standards

Color Format/Sample Rate/Data Range / Computation Approach	PSNR Lena (512x512)	PSNR Tsukuba (512x512)	PSNR Venus (434x383)	PSNR Teddy (450x375)	PSNR Cones (450x375)
YCbCr 444 ITU601 8bit 0-255 floating maths	52.994	52.891	52.633	52.833	52.783
YCbCr 422 ITU601 8bit 0-255 floating maths	47.016	40.783	35.689	34.511	32.906
YCbCr 420 ITU601 8bit 0-255 floating maths	46.204	38.076	33.541	32.205	31.184
YCbCr 444 ITU601 8bit 0-255 integer maths (8bit)	52.000	52.793	52.393	52.650	53.569
YCbCr 422 ITU601 8bit 0-255 integer maths (8bit)	46.739	40.777	35.682	34.508	32.906
YCbCr 420 ITU601 8bit 0-255 integer maths (8bit)	45.988	38.073	33.538	32.205	31.183
YCbCr 444 ITU601 8bit 16-235-240 floating maths	50.002	49.918	49.950	49.884	49.875
YCbCr 422 ITU601 8bit 16-235-240 floating maths	46.044	40.550	35.623	34.446	32.854
YCbCr 420 ITU601 8bit 16-235-240 floating maths	45.370	37.960	33.503	32.164	31.146
YCbCr 444 ITU601 8bit 16-235-240 integer (8bit)	51.735	52.014	51.429	51.727	51.599
YCbCr 422 ITU601 8bit 16-235-240 integer (8bit)	46.665	40.727	35.656	34.493	32.893
YCbCr 420 ITU601 8bit 16-235-240 integer (8bit)	45.913	38.053	33.518	32.194	31.176
YCbCr 444 ITU709 8bit 0-255 floating maths	53.042	53.170	53.083	53.040	53.055
YCbCr 422 ITU709 8bit 0-255 floating maths	46.679	40.442	35.303	34.039	32.362
YCbCr 420 ITU709 8bit 0-255 floating maths	45.872	37.753	33.122	31.719	30.657
YCbCr 444 ITU709 8bit 0-255 floating maths	52.064	52.965	51.685	52.473	52.279
YCbCr 422 ITU709 8bit 0-255 floating maths	46.442	40.419	35.289	34.035	32.353
YCbCr 420 ITU709 8bit 0-255 floating maths	45.652	37.744	33.114	31.718	30.653
YCbCr 444 ITU709 8bit 16-235-240 floating maths	51.740	51.933	51.803	51.701	51.724
YCbCr 422 ITU709 8bit 16-235-240 floating maths	46.348	40.374	35.281	34.026	32.348
YCbCr 420 ITU709 8bit 16-235-240 floating maths	45.582	37.728	33.106	31.715	30.649
YCbCr 444 ITU709 8bit 16-235-240 integer (8bit)	49.372	50.775	50.485	49.307	49.776
YCbCr 422 ITU709 8bit 16-235-240 integer (8bit)	45.546	40.325	35.253	33.995	32.338
YCbCr 420 ITU709 8bit 16-235-240 integer (8bit)	44.900	37.707	33.091	31.705	30.638

Since floating point calculation is more difficult and complicated to implement on hardware design, integer computation is preferred by scaling the floating point coefficients to integers in color space transform. Table 6.3 evaluates several scaling resolutions from four bit (256) to ten bit (1024). As a result of the lower scaling resolution, it delivers a lower quality of output image. From the observations, the result of PSNR in 6 bit (128) scaling up resolution approximates to the result of PSNR in the floating point approach. Thus, from our experiment, it is possible to use less hardware resources to achieve a quality result with the integer approach.

Two commonly used rounding approaches were mentioned: rounding towards zero and rounding to the nearest neighbor. Therefore we measure those two rounding methods in the software. Table 6.4 illustrates that the rounding to nearest method

Table 6.3: scale resolutions of integer approximation from four bit (256) to ten bit (1024)

Color Format/Sample Rate/Data Range / Computation Approach	PSNR Lena (512x512)	PSNR Tsukuba (512x512)	PSNR Venus (434x383)	PSNR Teddy (450x375)	PSNR Cones (450x375)
YCbCr 444 ITU601 8bit 0-255 integer maths(4bit)	34.603	35.176	36.043	34.943	35.530
YCbCr 422 ITU601 8bit 0-255 integer maths(4bit)	34.418	34.146	32.893	31.764	31.099
YCbCr 420 ITU601 8bit 0-255 integer maths(4bit)	34.376	33.407	31.630	30.403	29.919
YCbCr 444 ITU601 8bit 0-255 integer maths(6bit)	51.360	52.849	51.241	50.489	50.083
YCbCr 422 ITU601 8bit 0-255 integer maths(6bit)	46.547	40.758	35.635	34.434	32.835
YCbCr 420 ITU601 8bit 0-255 integer maths(6bit)	45.781	38.059	33.495	32.145	31.129
YCbCr 444 ITU601 8bit 0-255 integer maths(8bit)	52.000	52.793	52.393	52.650	53.569
YCbCr 422 ITU601 8bit 0-255 integer maths(8bit)	46.739	40.777	35.682	34.508	32.906
YCbCr 420 ITU601 8bit 0-255 integer maths(8bit)	45.988	38.073	33.538	32.205	31.183
YCbCr 444 ITU601 8bit 0-255 integer maths(10bit)	53.015	52.880	52.668	52.842	52.787
YCbCr 422 ITU601 8bit 0-255 integer maths(10bit)	47.021	40.782	35.691	34.512	32.907
YCbCr 420 ITU601 8bit 0-255 integer maths(10bit)	46.208	38.075	33.543	32.206	31.185
YCbCr 444 ITU601 8bit 0-255 floating maths	52.994	52.891	52.633	52.833	52.783
YCbCr 422 ITU601 8bit 0-255 floating maths	47.016	40.783	35.689	34.511	32.906
YCbCr 420 ITU601 8bit 0-255 floating maths	46.204	38.076	33.541	32.205	31.184
YCbCr 444 ITU709 8bit 0-255 integer maths(4bit)	29.805	34.856	32.200	30.914	31.159
YCbCr 422 ITU709 8bit 0-255 integer maths(4bit)	29.750	33.775	30.593	29.223	28.722
YCbCr 420 ITU709 8bit 0-255 integer maths(4bit)	29.724	33.072	29.771	28.309	27.913
YCbCr 444 ITU709 8bit 0-255 integer maths(6bit)	39.324	44.407	42.774	41.366	41.844
YCbCr 422 ITU709 8bit 0-255 integer maths(6bit)	38.744	39.033	34.576	33.327	31.914
YCbCr 420 ITU709 8bit 0-255 integer maths(6bit)	38.625	37.005	32.638	31.276	30.340
YCbCr 444 ITU709 8bit 0-255 integer maths(8bit)	52.064	52.965	51.685	52.473	52.279
YCbCr 422 ITU709 8bit 0-255 integer maths(8bit)	46.442	40.419	35.289	34.035	32.353
YCbCr 420 ITU709 8bit 0-255 integer maths(8bit)	45.652	37.744	33.114	31.718	30.653
YCbCr 444 ITU709 8bit 0-255 integer maths(10bit)	52.929	53.140	53.048	53.033	53.042
YCbCr 422 ITU709 8bit 0-255 integer maths(10bit)	46.654	40.441	35.304	34.042	32.364
YCbCr 420 ITU709 8bit 0-255 integer maths(10bit)	45.855	37.753	33.123	31.722	30.661
YCbCr 444 ITU709 8bit 0-255 floating maths	53.042	53.170	53.083	53.040	53.055
YCbCr 422 ITU709 8bit 0-255 floating maths	46.679	40.442	35.303	34.039	32.362
YCbCr 420 ITU709 8bit 0-255 floating maths	45.872	37.753	33.122	31.719	30.657

keeps a higher PSNR than rounding to nearest method. From our observation, rounding to the nearest method is strongly recommended to be implemented in the color space converter, in order to keep the quality.

Table 6.4: Rounding approach evaluation

Color Format/Sample Rate/Data Range / Computation Approach	PSNR Lena (512x512)	PSNR Tsukuba (512x512)	PSNR Venus (434x383)	PSNR Teddy (450x375)	PSNR Cones (450x375)
YCbCr 444 ITU709 0-255 8bit floating maths with rounding to nearest	53.042	53.170	53.083	53.040	53.055
YCbCr 444 ITU709 0-255 8bit floating maths with round towards zero	43.589	43.612	43.695	43.665	43.619
YCbCr 444 ITU709 0-255 8bit integer maths (8bit) with rounding to nearest	52.064	52.965	51.685	52.473	52.279
YCbCr 444 ITU709 0-255 8bit integer maths (8bit) with round towards zero	42.487	43.238	42.407	43.244	42.714

6.1.2 Hardware Utilization Evaluation

In Table 6.5, we further calculate the requirements of look up table (LUT) for implementing a 8x8 constant multiplier in different scale resolutions. In this table, constant multipliers are implemented in the Full LUT and 2 LUT+ Adder structure that was introduced in Chapter 4. From the observations, the size of LUT increases in linear when the coefficient resolution increases. This table also shows that the 2 LUT Adder structure takes much less hardware resource.

Finally, Table 6.6 evaluates the resource consumption of RGB-YCbCr color space converters by using three different common constant multipliers (8x8) on Altera Quartus

Table 6.5: Hardware resource comparison of Full LUT Size approach and 2 LUT Size + Adder approach

<i>Adder approach</i>	<i>Full LUT Size</i>	<i>2 LUT+Adder</i>	<i>LUT Size Improvement</i>
Gate Count 4bit coefficient	256 entries $\times(8+4)=3072$	2x16 entries $\times(4+4)=256$	12
Gate Count 6bit coefficient	256 entries $\times(8+6)=3584$	2x16 entries $\times(4+6)=320$	11.2
Gate Count 8bit coefficient	256 entries $\times(8+8)=4096$	2x16 entries $\times(4+8)=384$	10.67
Gate Count 10bit coefficient	256 entries $\times(8+10)=4608$	2x16 entries $\times(4+10)=448$	10.29
Gate Count 12bit coefficient	256 entries $\times(8+12)=5120$	2x16 entries $\times(4+12)=512$	10

II9.1. A RGB-YCbCr444 converter requires 9 constant multipliers and 9 adders. From the observation, Full LUT constant multiplier consumes a great amount of Block ROMs size. 2LUT+Adder constant multiplier solution takes a little bit more combinational resource than Full LUT approach but only requires one-tenth of the block ROM. Shift and addsub approach takes the minimum hardware resource because it is composed of wire-shift and adders. Obviously, the shift and addsub approach provides the lowest cost solution for implementing the constant multiplier of color space converter. And the 2 LUT Adder approach not only consumes low Block ROM resources but also keeps the flexibility to different conversion vectors.

Table 6.6: Hardware utilization analysis of RGB to YCbCr converter

<i>Constant Multiplier approach</i>	<i>Full LUT approach</i>	<i>2 LUT + Adder</i>	<i>Shift and Add/sub approach</i>
LC Combinational	157(9 adder)	256(18 adder)	354 (28 adder)
LC Registers	10	9	33
Block ROMs (bits)	36864	3456	0

The worst case propagation delay of RGB to YCbCr converter for both 2 LUT Adder and ShiftAddsub approaches is further evaluated by the synthesizing tool, Synplify Premier, with stratix III library. The result shows the RGB-YCbCr converter with 2LUT Adder constant multiplier has a 0.66 ns execution propagation delay because of the look up table structure. This means the clock rate can achieve up to 979.8MHz. In the case of RGB to YCbCr converter with Shift and Addsub constant multiplier, the propagation delay is around 3.4ns. The clock rate can achieve 297 MHz.

6.2 Quality Evaluation

To assess the video quality for 3D TV system, the synthesized virtual video is calculated with the true interpolated video in PSNR [38] [9]. The interpolated position is defined to be located in the center of two stereo cameras as shown in Figure 6.1. Then the quality of the synthesized virtual view from our system will be monitored in PSNR value.

We compare the synthesis sequences from IMEC 3D TV system design with the synthesis sequences from Depth Estimation Reference Software (DERS) [37] and View Synthesis Reference Software (VSRS) [36]. Both DERS and VSRS were contributed

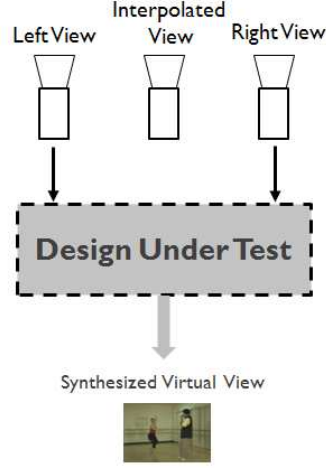


Figure 6.1: Interpolated video evaluation structure for our system

from the MPEG community. It is worth noting that DERS requires three camera views to generate each disparity map. In our design, only two views are required. DERS applies a Graph Cut stereo matching algorithm, a global optimization method. Compared to our stereo matching algorithm, Dynamic Programming is chosen to implement the global optimization function. In the disparity map refinement stage, DERS uses image segmentation and plane fitting; whereas, we use cross-based support region disparity voting. Besides, DERS also equips with a cost adjustment mechanism for temporal consistency enhancement; the cost adjustment condition is based on block-based motion detection. The motion detection mechanism applies Mean of Absolute Different (MAD) algorithm on a 16x16 block. Contrarily, in our case, we apply Absolute Different (AD) on single pixels in order to reduce the computation complexity. Figure 6.3 shows the virtual view evaluation results for 100 frames of Book Arrival stereo sequences. Obviously, the DERS solution achieves a higher video quality because the computational complexity of the algorithm itself is much higher than ours.

Finally, several anaglyph outputs with different depth intensity from our proposed system are captured and demonstrated in Figure 6.4.

6.3 Hardware Utilization Estimation

To estimate the hardware utilization of the 3D TV SoC design, we use Synplify Premier and Stratix III library. Table 6.7 concludes the hardware utilization of each processor unit on EP3SL150 FPGA. The default length of line buffers is set to 1024 in order to handle up to XGA format VGA video. It shows that the stereo matching engine consumes a large portion of the logic gates and block memory resources in the system.

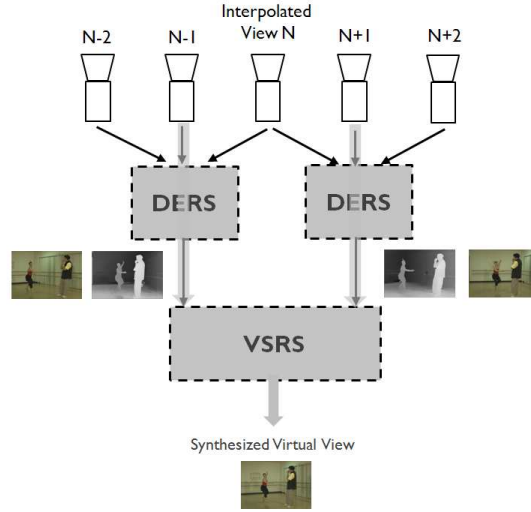


Figure 6.2: Interpolated video evaluation structure for DERS+VSRS

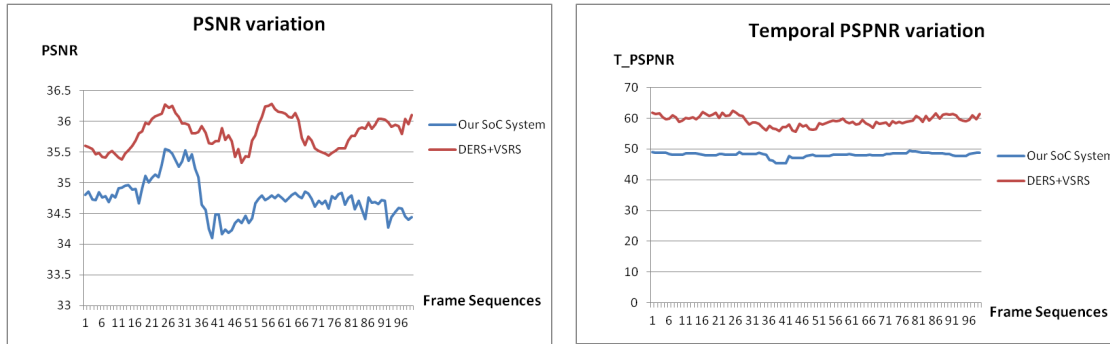


Figure 6.3: Quality evaluation for Book Arrival

Table 6.7: Hardware resource utilization summary on EP3SL150 FPGA

<i>Processor Units</i>	<i>LC Comb.</i>	<i>LC Reg.</i>	<i>Block Mem Bits</i>
Video Input Adaptor	27	52	0
RGB \rightarrow YUV422 (X 2)	708	66	0
Stereo Matching Engine-D64 (X 2)	46970	27921	1207280
DDR W/R Scheduler	2745	1163	393216
DDR2 HPC controller	2312	2560	5120
VS Adaptor	142	148	0
View Synthesis Engine	4854	16989	598196
YUV422 \rightarrow RGB (X 2)	524	176	0
Anaglyph	11	7	192
Video Out Adapter	126	87	0
Sum of Hardware Utilization	58419	49169	2204004
Available Hardware Resource	113600	113600	5630976
Hardware Utilization Rate	51.4%	43.3%	39.1%

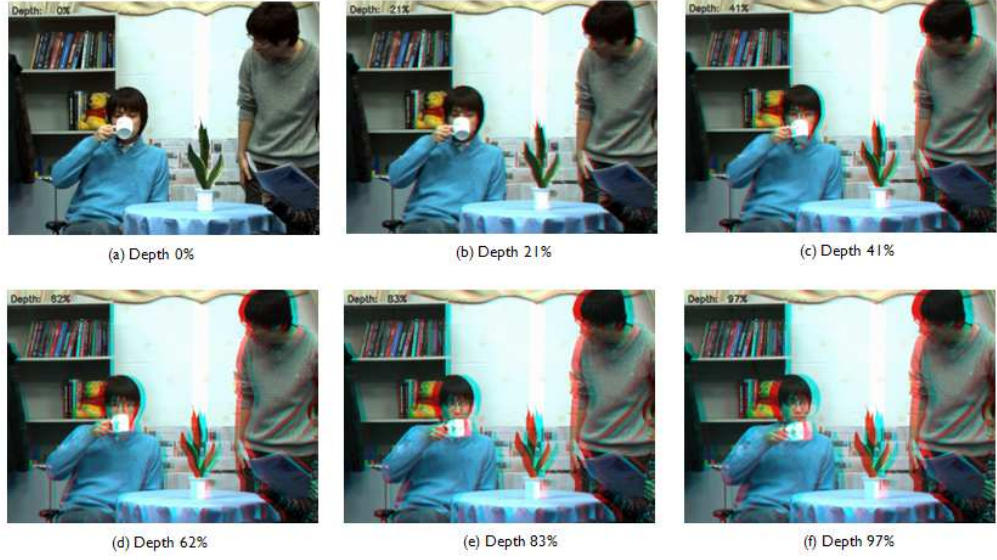


Figure 6.4: Anaglyph outputs for different depth intensity

6.4 Evaluation of Real-Time Performance

We evaluate the real-time performance of the proposed IMEC 3DTV SoC architecture on FPGA. There are three clock domains in the proposed system: pixel clock, DDR controller local clock, and DDR clock. Based on the clock frequency, the timing constraints of our designs are checked because it definitely shows much real-time performance this system can achieve.

In the pixel clock domain, the clock frequency is synchronized with the input pixel clock. Since our processing units process the stream data in pipeline throughout the system, the critical path of the design has to follow the timing constraint of the pixel clock. From the report of Synplify Premier, the critical path lies to the Stereo Matcher Engine, in which 72.4MHz frequency is estimated. Therefore, the SoC can process up to standard XGA (1024x768@60FPS 65MHz under 65 MHz Pixel Rate) video format on EP3SL150FPGA.

The memory hierarchy crosses both pixel and DDR controller local clock domains. The report of Synplify Premier shows that the circuit works on pixel clock domain can achieve a maximum of 275.4MHz, and the other part of the circuit that works on DDR clock domain can achieve a maximum of 311.7MHz. Since the pixel clock rate is far lower than the pixel clock rate in the proposed SoC, and the DDR controller local clock domain is 150MHz in our default setting, this proves that the worst case latency is still in the range of critical latency.

In what follows, we further evaluate the real-time performance of the customized memory hierarchy design. We first evaluate the bandwidth requirement of stream

processors based on different standard video input sources in Table 6.8. The interface between DDR Scheduler and DDR controller are working at half the DDR clock rate, which is 150MHz, and the DDR is working at 300MHz. The bandwidth usage shows it is sufficient to support a stream processor in all kinds of standard VGA formats.

Table 6.8: Throughput estimation based on standard VGA video source

Format	VGA	SVGA	XGA	HD-720p	Sbs Full HD-1080p
Frame Size	640X480	800X600	1024X768	1280x720	Side-by-side 1920x1080
Pixel Rate (MHz)	25.175	40	65	74.15	148.5 (half rate 74.25)
Frame per Second	60	60	60	60	60
Throughput (G bit/s)	3.26	5.07	8.3	9.73	10.95
Bandwidth Utilization (%)	10.5	16.3	26.7	31.3	35.2

However, the command efficiency is not guaranteed to support the critical latency requirement of each SG-DMA device. Stream processors access frame buffers continuously. If the critical latency of the SG-DMA is not fulfilled, stream processor will suffer from data starvation or data congestion. Extra latencies might be contributed from DDR refresh period, pre-charge time, command activate time, DDR controller latency, or Arbiter latency. Figure 6.5 shows an example of the long latencies which includes regions (1), (2), (3) and (4).

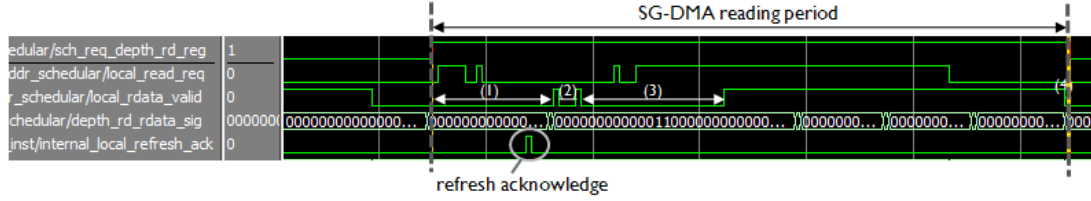


Figure 6.5: Example of latencies during burst reading

To ensure no critical latency constraints of SG-DMA devices are violated, a simulation environment is built as Figure 6.6. The memory and controller models [12] are generated from VHDL from Altera Quartus II 9.1. The device under test (DUT) includes multiple SG-DMA components, Arbiter, DDR controller, and DDR model. To trigger the operation of SG-DMA component, the data access patterns of stream processors are generated based on standard video timing with pipeline delays. Finally, the worst case latencies of SG-DMA devices are monitored on Simulation tool (Modelsim).

Since the design of SG-DMA devices uses long data burst technique, we try to estimate the impacts of different burst lengths to the efficiency. Table 6.9 reveals different burst length configurations to bus efficiency. In this experiment, the data access pattern is based on the signal timing of XGA (1024x768@60FPS under 65 MHz Pixel Rate). The DDR SDRAM model works at 300 MHz, and the local interface of DDR controller works at 150 MHz with burst size 4. The result shows long burst

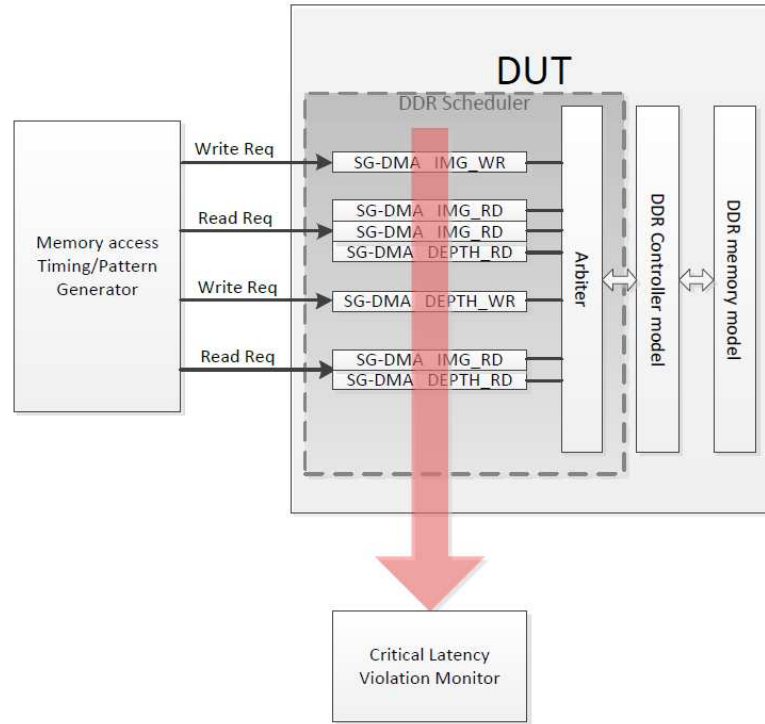


Figure 6.6: Memory hierarchy evaluation environment

length facilitate the command efficiency because extra latencies are shared in one long burst. In contrarily, short burst length dampens the command efficiency because extra latencies are frequently introduced from command switching. The final result shows the top three burst length configurations can pass the critical latency constrain in worst case scenario.

Table 6.9: Burst length settings and critical latency analysis

<i>Burst Length (Line Buffer Length)</i>		<i>Command number(cycle)</i>	<i>Commands + Worst case latencies(cycle)</i>	<i>Command Efficiency(%)</i>	<i>Critical Latency(cycle)</i>	<i>Latency Violation</i>
<i>Image</i>	<i>Depth</i>					
256	128	1408	1573	89.5	2048	Pass
128	64	704	869	81	1024	Pass
64	32	352	507	69.4	512	Pass
32	16	176	341	51.6	256	Fail
16	8	88	249	35.3	128	Fail

Conclusion and Future Works

This chapter summarizes the achievements of this thesis. In Section 7.1, the proposals and experimental results are concluded. Then we summarise each chapter and present the contributions in Section 7.2. Finally, future works and applications are provided as suggestion in Section 7.3.

7.1 Conclusion

Two stereo matching algorithm improvements have been introduced and implemented in this thesis. In order to generate high quality disparity map, dynamic programming algorithm is introduced into the stereo matching computational flow. We select Potts model as the smoothness function because it performs outstandingly in the disparity discontinuous regions and low computational complexity. From the Middlebury's benchmark, the stereo matching algorithm achieves a 6.6% average pixel error rate. In order to improve the temporal quality of disparity sequences, we adjust the matching cost based on the history disparity map result and pixel-based motion information. The final disparity map sequences is improved especially in the flickering issue.

In this thesis we provide two ideas to improve the hardware utilization of the Stereo Matching Engine. By taking the advantage of Potts model smoothness function, we proposed a hardware efficient architecture. We simplify the back track path data by only recording the path decision information. Furthermore, we apply 2-Port BRAM with a sophisticate memory address generation pattern to reduce the on-chip memory requirement to half, compared to ping-pong memory architecture. The on-chip memory requirement of Dynamic Programming Processor is reduced 11 times without losing the quality in the case of 64 disparity range. As well, we creatively introduce Run-length Coding (RLC) algorithm into the post-processor of stereo matching engine in order to reduce memory consumption. This thesis found the disparity sequence is a perfect candidate to implement RLC algorithm. The experiment shows that it can achieve above 4.75 times of compression rate with nearly lossless quality. So we further propose an on-chip memory architecture with RLC on Vertical Voting Processor. Finally, the resource consumption of the Stereo Matching Engine is reduced dramatically after applying memory compression techniques on dynamic programming and post processor functions. The on-chip memory of the stereo matching engine is optimized by 2.53 times. So far, the stereo matching engine only takes 40% of combinational logic, 25.3% of register and 21% of bit on-chip Block RAM on EP3SL150 FPGA.

In the system implementation work, we successfully implement the IMEC 3D TV SoC on EP3SL150 FPGA and make the 3D intensity of stereoscopic contents be adjustable.

The proposed system architecture is mainly composed of five parts: Video Adaptors, Color Space Converters, Stereo Matching Engine, memory hierarchy, and View Synthesis Engine. The Video Adaptor designs successfully synchronize input streams and output complete video frame pixels in a standard timing sequence. Then we design the Color Space Converters and provide two low cost constant multiplier configurations, based on the requirement of flexibility. Furthermore, the proposed memory hierarchy successfully supports the temporal consistency function of stereo matching and view point synthesis blocks for frame buffering. The memory hierarchy hides the DDR access latency by long burst with the help of 2-Port RAM. The total throughput achieves the required 8.035G bits with a 81% command efficiency. Finally, we integrate the above mentioned works with Stereo Matching and View Synthesis Engines. The stereo video sources and disparity streams are properly calculated to generate an accurate virtual view from the Viewpoint Synthesis Engine. According to the report of quality evaluation, the interpolated videos achieve average 34.7db PSNR and 50db TPSPNR in Book Arrival test sequences which show an acceptable quality for 3D TV application. To display 3D video on screen, we insert an anaglyph IP into the proposed system to generate two dimension video for conventional 3D monitor. Through the proposed system, an audience is able to adjust the depth intensity of stereoscopic 3D contents by the on-board buttons and watch the 3D video in true real-time.

7.2 Summary of Chapters and Contributions

Chapter 1 points out that 3D visual comfort is required by viewers. Due to the depth comfort region, variously dependent on the viewer and display technology, adaptive 3D contents is a challenge for the existing 3D content standards. Therefore, this thesis aims to provide a SoC solution to support depth intensity adjustment. In this chapter, we point out the spatial and temporal quality issues in generating synthesized 3D contents. We also point out that the hardware overhead is introduced by applying dynamic programming algorithm which performs global optimization in a stereo matching computational flow.

Chapter 2 provides a background overview of the stereo matching algorithm. This chapter first sum up the common stereo matching flows in both local and global stereo matching approaches. Then the relevant researches for individual steps, including matching cost generation, stereo matching computation, global optimization, and refinements, are concluded. Since we are interesting in global optimization approach, the background of dynamic programming approach is explained explicitly.

In Chapter 3, we introduce the dynamic programming algorithm into the stereo matching computational flow. This thesis further propose a hardware efficient dynamic programming architecture by taking the advantage of Potts model smoothness function. A backtrack path data simplification method and a sophisticated 2-Port BRAM access pattern were presented to reduce the on-chip memory requirement. However, the on-chip memory consumptions of the preprocessor and the postprocessor in the stereo matching engine are still large. Therefore, we propose an on-chip memory

architecture with run-length coding algorithm to reduce the memory consumption of the Vertical Voting Processor in disparity voting function. Finally, we insert matching cost updating technique into the stereo matching algorithm in order to enhance the temporal consistency of disparity sequences.

In Chapter 4, several proposals from Chapter 3 are evaluated. We first measure the quality of test disparity maps. It achieves a 6.6% average pixel error rate in Middlebury's benchmark. Then we evaluate the hardware utilization and scalability of the Dynamic Programming Processor. The hardware utilization of the proposed hardware efficient Dynamic Programming Processor was estimated. The on-chip memory shows an 11 times of improvement without quality being lost in 64 disparity range scenario. Another memory optimization proposal is in the Vertical Voting Processor. The proposed memory architecture with run-length coding encoderdecoder is explored based on the tradeoff of compression rate and pixel error rate. We developed an evaluation flow to search for the optimal parameter settings. The proposed memory architecture shows above a 12 times of improvement without quality being lost in 5 test sets. Finally, the total on-chip memory utilization of the stereo matching engine is optimized by 2.53 times in the above-mentioned designs.

In Chapter 5, the 3D TV System SoC architecture is designed and implemented on EP3SL150 FPGA. The SoC architecture includes Video Adaptors, Color Space Converter, Stereo Matching Engine, memory hierarchy, and View Synthesis Engine. We designed most of the components, except the viewpoint synthesis engine (support by NCTU and IMEC-Taiwan) and DDR controller (IP from Altera). In the beginning, we provided a system architecture overview from three aspects: function definition, clock domain, and system on-chip interconnection. Then the individual component designs are presented. First, the background of view synthesis algorithm is given. Then the Video InputOutput Adaptors are proposed to synchronize and control the incoming and display sequences based on standard VGA signal. Afterward, we present the RGB-YCbCr Color Space Converter designs. Two hardware-friendly constant multiplier configurations for the color space converter, Shift Addsub and 2LUT+Adder approaches, are proposed. Finally, a customized memory hierarchy is constructed to support frame buffering for stream processors. The memory architecture contains SG-DMA, Arbiter, memory controller, and off-chip memory. The SG-DMA and Arbiter are integrated into DDR Scheduler block, which can work alone without extra processor kernel and standard on-chip bus to increase the efficiency. The proposed memory hierarchy is designed to hide the data read-and-write latency by long data burst and data pre-fetch scheduling.

Chapter 6 evaluates the entire 3D TV SoC design from three aspects: quality, hardware utilization, and real-time performance. With the support from memory hierarchy, the temporal consistency function works on the stereo matching engine and generates stable and accurate depth map video for viewpoint synthesis engine. The experiment shows the interpolated view achieves an average 34.7db PSNR and 50 db TPSPNR in Book Arrival test sequences. We analyzed the entire SoC design from hardware utilization aspect. To analyse the real-time performance, we start from evaluating the memory

hierarchy design in a worst case latency and detected the command efficiency under different burst lengths, in the case of the SoC. The critical latency of each SG-DMA device is monitored and verified on both simulation tool and FPGA. Finally, the critical paths of stream processors are estimated by synthesis tool. The analysis shows that it can process up to XGA standard video format (1024x768@60FPS 65MHz under 65 MHz Pixel Rate) on EP3SL150 FPGA.

7.3 Future Work and Application Development

Regarding the current processing unit designs, there are several aspects can be improved. The proposed idea will be introduced as follows.

1. Rendering better disparity mapvideo is a motivation to improve the Stereo Matching Engine. The matching cost generation algorithm can be further improved in order to achieve high quality disparity map. The global optimization algorithm can also be improved since the scanline-based dynamic programming algorithm is applied. In the scanline-based dynamic programming algorithm, the global optimization is only restricted in single scanline and lack connection between scanlines. In addition, the temporal consistency of disparity video can be improved since the current system is only using pixel-based motion detection mechanism. The pixel-based motion detection mechanism is still relatively simple and can be improved in advance.

In the resource utilization aspect, dynamic programming algorithm requires a large memory space for storing the backward pass information. We have introduced a specific compression method by using Potts model smoothness function in Chapter 4. However, it is unable to cover other smoothness function models such as linear and second ordered models. Besides, the pre-processor and post-processor still take a large among of onchip memory because of the cross-based disparity voting algorithm. A hardware-friendly stereo matching algorithm with both hardware efficiency and accuracy is expected.

2. The proposed memory hierarchy will integrate with extra processing units. For example, H.264 codec can be extended into HD3DTV processing unit, and it will consume a large amount of off-chip bandwidth. Thus, a critical latency-aware arbiter and high efficient off-chip memory controller will be needed to achieve quality of service (QoS).
3. Standard network on chip bus such as AHB, OCB, Wishbone etc. can be introduced into the SoC system when the number of processing units are expended. For example, a standard bus structure could be built to deal with the interconnections between processing units and off-chip memory controllers.

All in all, the technologies in the proposed IMEC 3D TV system SoC could be applied to many applications such as object tracking, 3D reconstruction, navigation system, gesture detection, 3D digital camera and eye-gazing view point interpolation, as potential examples. There is no doubt that 3D-related technologies will influence our lives profoundly in the future soon to come.

Bibliography

- [1] Altera, *The efficiency of the ddr & ddr2 sdram controller compiler*, May 2011.
- [2] Pedram Azad, Tilo Gockel, and Rdiger Dillmann, *Computer vision: Principles and practice*, elektor, 2008.
- [3] Z. F. Baruch, *Structure of computer systems, page 62, isbn 973-8335-44-2*, U. T. PRES, Cluj-Napoca, 2002.
- [4] Y. Boykov, O.Veksler, and R. Zabih., *Efficient approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2001), 11.
- [5] Yuri Boykov, Olga Veksler, and Ramin Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2001).
- [6] G. Bradsky and A. Kaehler, *Learning opencv*, OReilly, 2008.
- [7] Y.C. Chang, Y.C. Tseng, and T.S. Chang, *Analysis of color space and similarity measure impact on stereo block matching*, IEEE Asia Pacific Conference, In Circuits and Systems APCCAS (2008).
- [8] A.Y.C. Chen and J.J. Corso, *Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm*, IEEE Workshop on Applications of Computer Vision (WACV) (2011).
- [9] Wei Chen, Jerome Fournier, Marcus Barkowsky, and Patrick Le Callet, *New requirements of subjective video quality assessment methodologies for 3dtv*, VPQM (2010).
- [10] C. Cigla and A.A. Alatan, *Temporally consistent dense depth map estimation via belief propagation*, 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (2009).
- [11] Altera Coperation, *Avalon interface specifications*, May 2005.
- [12] ———, *Ddr and ddr2 sdram high-performance controller user guide*, March 2009.
- [13] Scharstein D., Szeliski R., and Zabih R., *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, IEEE Workshop, Stereo and Multi-Baseline Vision (2001).
- [14] Hirschmuller H. and Scharstein D., *Evaluation of stereo matching costs on images with radiometric differences*, Pattern Analysis and Machine Intelligence, IEEE Transactions (2009).
- [15] Richard Hartley and Andrew Zisserman, *Multiple view geometry*, CVPR, June 1999.

- [16] Heiko Hirschmuller, *Accurate and efficient stereo processing by semi-global matching and mutual information*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2005).
- [17] Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann, *Local stereo matching using geodesic support*, IEEE, ICIP (2009).
- [18] ITU, *Recommendation bt.601*, 2011.
- [19] ———, *Recommendation bt.709*, 2011.
- [20] Ricardo P. Jacobi, Renato B. Cardoso, and Geovany A. Borges, *Voc: A reconfigurable matrix for stereo vision processing*, 20th International Parallel and Distributed Processing Symposium IPDPS (2006).
- [21] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon, *Fpga design and implementation of a real-time stereo vision system*, IEEE Transactions on Circuits and Systems for Video Technology (2010).
- [22] Kuk jin Yoon Student Member In So Kweon, *Adaptive support-weight approach for correspondence search*, IEEE Trans. PAMI (2006).
- [23] T. Kanade and M. Okutomi, *A stereo matching algorithm with an adaptive window: Theory and experiments*, IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994), 9.
- [24] Steve Knapp, *Constant-coefficient multipliers save fpga space and time*, EDA (1998).
- [25] Kun-Bin Lee, Tzu-Chieh Lin, and Chein-Wei Jen, *An efficient quality-aware memory controller for multimedia platform soc*, IEEE Circuits and Systems for Video Technology (2005).
- [26] Carlos Leung and Brian C. Lovell, *An energy minimisation approach to stereo-temporal dense reconstruction*, Proc. International Conference on Pattern Recognition (2004).
- [27] Humenberger M., Engelke T., and Kubinger W., *A census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching quality*, IEEE Computer Society Conference , Computer Vision and Pattern Recognition Workshops (CVPRW) (2010).
- [28] D.K. Masrani and W.J. MacLean, *A real-time large disparity range stereo-system using fpgas*, IEEE International Conference on Computer Vision Systems (2006).
- [29] Stefano Mattoccia, *Stereo vision:algorithms and applications*, \www.vision.deis.unibo.it/smatt, May 2011.
- [30] Mark Nelson, *The data compression book*, M&T Books, 1992.

- [31] Daggu Venkateshwar Rao, Shruti Patil, Naveen Anne Babu, and V Muthukumar, *Implementation and evaluation of image processing algorithms on reconfigurable architecture using c-based hardware descriptive languages*, 2006.
- [32] R.I.Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2000.
- [33] Fridtjof Stein, *Efficient computation of optical flow using the census transform*, 2004.
- [34] Gary Sullivan and Stephen Estrop, *Video rendering with 8-bit yuv formats*, Microsoft Digital Media Division (2002).
- [35] Jian Sun, Heung yeung Shum, and Nan ning Zheng, *Stereo matching using belief propagation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2003).
- [36] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, *Reference softwares for depth estimation and view synthesis*, ISO/IEC JTC1/SC29/WG11, Archamps, France, Tech. Rep. **M15377** (2008).
- [37] M. Tanimoto, T. Fujii, M. P. Tehrani, and M. Wildeboer, *Depth estimation reference software(ders) 4.0*, ISO/IEC JTC1/SC29/WG11, MPEG 2008/M16605, London, UK (2009).
- [38] A. Tikanmaki, A. Gotchev, A. Smolic, and K. Miller, *Quality assessment of 3d video in rate allocation experiments*, ISCE (2008).
- [39] M. Tkalcic and J. Tasic., *Color spaces perceptual, historical and applications background*, EUROCON, IEEE **1** (2003), 304–308.
- [40] T.Kanade, *Development of a video-rate stereo machine. in image understanding workshop, pages 549-557*, Morgan Kaufman, 1994.
- [41] Carlo Tomasi, *Global stereo in polynomial time*, 2007.
- [42] E. Trucco and A. Verri, *Introductory techniques for 3d computer vision*, Prentice Hall, 1998.
- [43] Middlebury university, *Middlebury stereo evaluation tool*, <http://vision.middlebury.edu/stereo/eval/>.
- [44] ———, *Middlebury stereo datasets*, <http://vision.middlebury.edu/stereo/data/>, 2001.
- [45] John Iselin Woodfill, Gaile Gordon, Dave Jurasek, Terrance Brown, and Inc. Ron Buck Tyzx, *The tyzx deepsea g2 vision system, a taskable, embedded stereo camera*, IEEE Computer Society Workshop on Embedded Computer Vision, Conference on Computer Vision and Pattern Recognition, (New York, NY) (2006).
- [46] O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon, *Global stereo reconstruction under second order smoothness priors*, CVPR (2008).

- [47] Guanyu Yi, *High-quality real-time hd video stereo matching on fpga*, Master's thesis, TU Delft, 2011.
- [48] Chang Yuan, Hao Pan, and Scott Daly, *Stereoscopic 3d content depth tuning guided by human visual models*, Sharp Laboratories of America, 5750 NW Pacific Rim Blvd, Camas, WA 98607, USA (2011).
- [49] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit, *Cross-based local stereo matching using orthogonal integral images*, IEEE transactions on circuit and systems for video technology (2009).
- [50] Ke Zhang, Jiangbo Lu, Gauthier Lafruit, Rudy Lauwereins, and Luc Van Gool, *Accurate and efficient stereo matching with robust piecewise voting*, IEEE international conference on Multimedia and Expo (2009).
- [51] Lu Zhang, *Design and implementation of real-time high-definition stereo matching on soc on fpga*, Master's thesis, TU Delft, 2010.
- [52] Lu Zhang, Ke Zhang, Tian Sheuan Chang, Gauthier Lafruit, Georgi Krasimirov Kuzmanov, and Diederik Verkest, *Real-time high-definition stereo matching on fpga*, 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (2011), 55–64.
- [53] Yin Zhao and Lu Yu, *Pspnr tool 2.0*, MPEG2009/M16890 (2009).