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

CE-MS-2012-29

M.Sc. Thesis

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

Hsiu-Chi Yeh

Abstract

In recent 3D TV market, the technologies addressing to the applications such as stereoscopic depth scaling, glass-free 3D display, and Free Viewpoint TV are getting more attentions. A low-cost solution that can synthesize intermediate views from stereoscopic input contents(left and right camera views) is strongly needed. To render the interpolated views, the depth information of the left and right views are commonly-used in view synthesis algorithm. Therefore, this thesis researches stereo matching algorithms, which generate disparity maps. We implement a state-of-the-art semi-global stereo matching algorithm(dynamic programming and cross based) with FPGA. Our solution also concerns several aspects include disparity mapsequences quality, hardware cost, and real-time performance. Afterward the stereo matching engine design is integrated into IMEC's 3D TV SoC prototype. Several peripheral components, include color space converters, video IO adaptors and a dedicated memory hierarchy, are developed for supporting both stereo matching and view synthesis engines. Finally, the SoC prototype is evaluated with EP3SL150 FPGA chip. So far it can process dual channel XGA video format (1024×768 @ 60 FPS) in real-time performance and render an acceptable synthesized view quality for depth scaling application. This design shows a promising solution for the 3D TV market.

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
Codenumber	:	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
April 20, 2012

Contents

List of Figures

List of Tables

Introduction 1

This thesis work is developed at IMEC Leuven, Belgium and supported by the Computer Engineer Group from Delft University of Technology at the Netherlands. The main research and development goal surrounds an IMEC proposed 3D HD TV Project, which is a SoC solution addressing to 3D TV market. This thesis work especially interests in the stereo matching engine part, which is used for extracting depth information from two stereoscopic camera sources. We designed the stereo matching algorithm and implemented with FPGAs. Then it has been furhter integrated with a view point synthesis engine. Another assignment in this thesis work is to develop peripheral components for supporting the IMEC 3D HD TV SoC prototype.

1.1 Motivation

Nowadays, 3D display related applications are getting more attentions in the consumer market. One obvious clue is the number of movies with 3D content is growing dramatically. It shows the trends to the entertainment tendency of audiences. It is foreseeable that 3D display applications will become prevalent in the coming future.

The fundamental principle of 3D display technologies is based on binocular vision. It is about how our brain perceives the depth of objects in real world. As Figure 1.1, our eyes receive horizontal deviated views, and our brain automatically analyzes the disparities of objects and fuses them to 3D perception. Many stereoscopic vision technologies, such as Free Viewpoint Television (FTV) [48][65], 3D Autostereoscopic Displays [17] and depth scaling, are based on this principle.

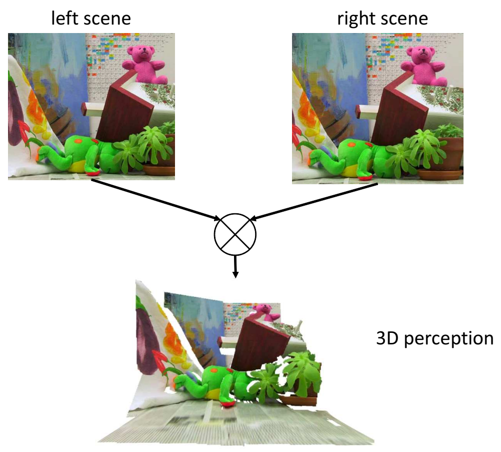


Figure 1.1: 3D Perception

It is frequently reported that audiences easily suffer from nausea, eye strain or headache after watching 3D contents. Figure 1.2 illustrates the personal depth comfort regions in a green zone. However, the ranges of comfort zone depends on different people. For example, the eye distance of adult is around 6.25 cm. However, the eye distance is shorter in the case of children, so the 3D contents might not applicable to them. Moreover, the comfort zone also relates to screen size, view distance and display technology. It leaves a great challenge to the 3D contents vendors.

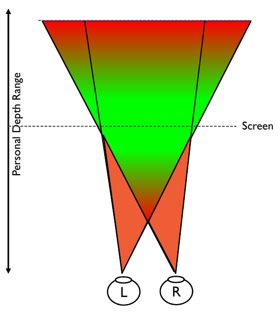


Figure 1.2: Depth perception according to visual comfort.

Therefore, IMEC proposes a low-cost depth scaling solution addressing to the visual comfort for 3D TV. The principle is synthesizing the virtual view that lies in-between stereoscopic video sources, then outputs the in-between virtual view associated with one raw stereoscopic video channel instead of original stereoscopic video sources to 3D TV. Through this method, audiences receives less 3D depth perception from the 3D TV. In order to synthesize the in-between virtual view, the depth information of the original stereoscopic video sources are needed to be extracted firstly. With the raw stereoscopic scenes and their depth information, the in-between virtual view can be synthesized through warping and occlusion holes filling processes. Figure 1.3 simply illustrates the general principle of the proposal. With this system, audience can choose an arbitrary depth intensity of 3D video contents based on their preferences. In this project, we aims at realizing the prototype with FPGAs.

1.2 Research Goals and Problem Definition

This thesis work aims at the SoC prototyping work for the IMEC proposed 3D TV system. First of all, we are assigned to design and implement an efficient stereo matching algorithm with FPGAs. The selection of stereo matching algorithm takes three aspects into consideration: real-time performance, disparity mapvideo quality, and hardware cost. The reasons are explained in the following:

Rendering quality disparity maps from a good stereo matching algorithm is one of the goals in this thesis. Although the stereo matching algorithms have been researched for decades, the solutions to occlusion, texture less, and repetitive pattern regions are not always perfect. Besides, generating quality disparity map is required in the IMEC proposed 3D TV system because it affects the accuracy of the synthesized virtual view. Previous study has shown that humans are sensitive to the edge regions of the image than

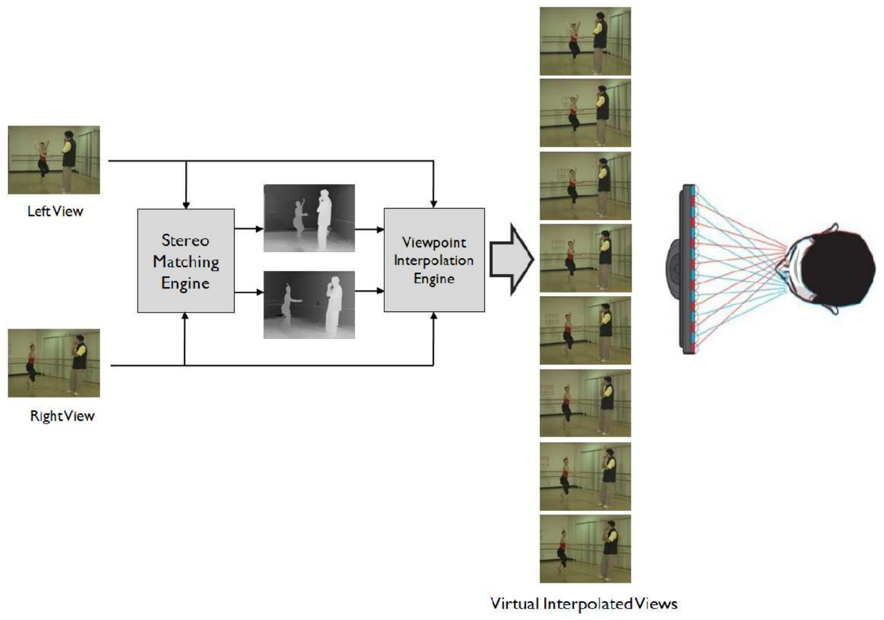


Figure 1.3: Depth adjustment processing flow with stereo matching and viewpoint synthesis engine.

the plain parts. The inaccurate disparity maps easily cause the so called ghost effect in the synthesized view. Figure 1.4 is an example that demonstrates the mentioned synthesis error.

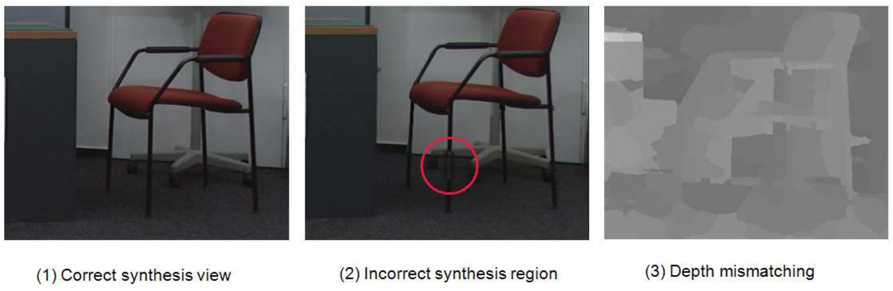


Figure 1.4: Ghost effect and inaccurate disparity map

Generally, the disparity map 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, which causes the inconsistent disparity map video. A previous study [64] 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 the temporal noise is even more obvious than spatial noise.

The hardware cost of SoC is concerned in the design. For example, we would like use dynamic programming algorithm to improve the quality of disparity maps. However, it

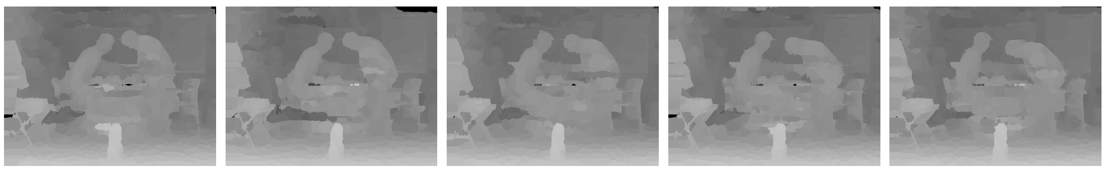


Figure 1.5: disparity map sequence without temporal consistency

will require $O(W \cdot D_{max}^2)$ computational complexity and $(W \cdot D_{max} \cdot D_{bit})$ memory space when real-time performance is expected. Where the term W represents the image width, and D_{max} represents the maximum disparity range. Obviously, a solution is needed to improve the tradeoff between the computational performance and hardware cost.

Finally, this thesis work also in charge of designing extra peripheral components such as off-chip memory controller, color space converters, and adaptors for the entired 3D TV SoC solution.

1.3 Solution and Contribution

This thesis aims at providing a totoal solution for the IMEC 3D TV SoC design addressing to video quality, real-time performance, and hardware cost. The contributions of thesis are listed in the following:

Stereo Matching Algorithm Improvement and Implementation

- Firstly, we survey the state-of-art stereo matching algorithms.
- We refer to the stereo matching engine design [63] that was contributed by Zhang and chose the scanline-based dynamic programming algorithm [23] for improving the quality of disparity maps. A 6.6% average pixel error rate is achieved according to Middlebury's benchmark [52].
- A simple pixel-based temporal consistency method is used to enhance the disparity map sequences by adjusting the matching raw cost based on last disparity maps and images. After introducing the temporal consistency method into our proposed stereo matching algorithm, the stability and quality of the disparity map sequences is improved.
- We compare the computational complexity and performance of different smoothness models for the global optimization function of our stereo matching engine. A smoothness model, Potts model, is chosen in our case. We proposed a method to reduce the computational complexity and memory consumption by rewriting the energy function. The result shows the computational complexity can be reduced from $O(W \cdot D_{max}^2)$ to $O(W \cdot D_{max})$, and the memory consumption can be 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 for compressing disparity maps. Our pre-experiments show the compression rate can reach above 12 times compression rate to the disparity map with almost no loss of quality. Therefore, we proposed a new memory architecture with run-length coding encoder/decoder. The new memory architecture only consumes 21% hardware cost comparing to conventional on-chip memory architecture. It proofs run-length coding is a promising solution for disparity map compression.

To construct the IMEC 3D TV SoC, we design and implement extra peripheral components to support our stereo matching engine and the view synthesis engine that is supported by NCTU(National Chiao Tung University). Those components are finally integrated in the 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. We compare two hardware efficient designs and evaluated it from quality, hardware consumption, and flexibility aspects.
- Three video signal adaptors are designed for input sequences synchronization, memory data extraction, and output display signal generation based on standard VGA signal timing.

Finally, the IMEC 3D TV prototype SoC is evaluated on EP3SL150 FPGA. The SoC integrates video adaptors, color space converters, stereo matching engine, view synthesis engine, and memory hierarchy. So far, the IMEC 3D TV prototype is capable of processing up to XGA format (1024x768@60FPS 65MHz Pixel Rate) video stream in real-time performance. Users can adjust the depth intensity of 3D contents through on-board buttons and watch anaglyph or synthesized stereoscopic video from 2D or 3D TV through our solution.

1.4 Overview of Chapters

In the next chapter, Chapter 2, it provides the background of stereo matching algorithms. In Chapter 3, we introduce a stereo matching algorithm proposal that was achieved by last thesis student. In addition, we evaluate its performance and hardware complexity. Then, several techniques and proposals are illustrated to optimize the disparity map sequences quality and hardware usage. The proposals of Chapter 3 are evaluated in Chapter 4. The designs are estimated addressing to image/video quality, hardware cost, and real-time performance. In Chapter 5, we integrate our stereo matching engine design into IMEC 3D TV SoC. The overview of the system architecture will first be introduced. Then we will mention about the designs of peripheral components, including memory hierarchy, color space converters, and video signal adaptors. In Chapter 6, we evaluate complete IMEC 3D TV SoC design with EP3SL150 FPGA. Firstly, the quality of interpolated virtual view sequences is measured by PSNR and TPSNR. Then the overall hardware cost and real-time performance information are also listed in this chapter for reference. The final chapter summarizes this thesis work. Several suggestions for future work and relative applications will be mentioned too.

Background on Stereo Matching and Related Works 2

This chapter makes a survey of depth extraction methods. We are especially interesting in stereo matching algorithms, which estimate the depth information by using stereoscopic image pairs. In Section 2.1, it reveals the principle of stereo vision and depth estimation. Then we try to summarize and compare the related works of stereo matching algorithms in Section 2.2. Furthermore, we survey the algorithms for reinforcing the consistency of disparity map sequences in Section 2.3. Finally, Section 2.4 concludes the frequently used platforms to implement stereo matching algorithms.

2.1 Stereo Matching Principle

The stereo matching principle is to mimick human's visual sensation from stereo input images. Figure 2.1 depicts a overlapped scene that is taken from two horizontally separated cameras. The nearby objects show larger displacement on the stereo images, whereas the distant objects only have trivial displacement on the stereo pair. We define the offset between corresponding points along scanline as the term D̈isparity.̈ Our goal is to apply this principle to computer algorithm, which helps to explore pixel correspondence on the stereo image pair and find their displacement. Since the disparity of each pixel is integer number, it can be encoded to grey scale range for output display. Figure 2.2 demonstrates a disparity map. Where the low gray pixel values represent low disparities and high grey pixel values correspond to high disparities.

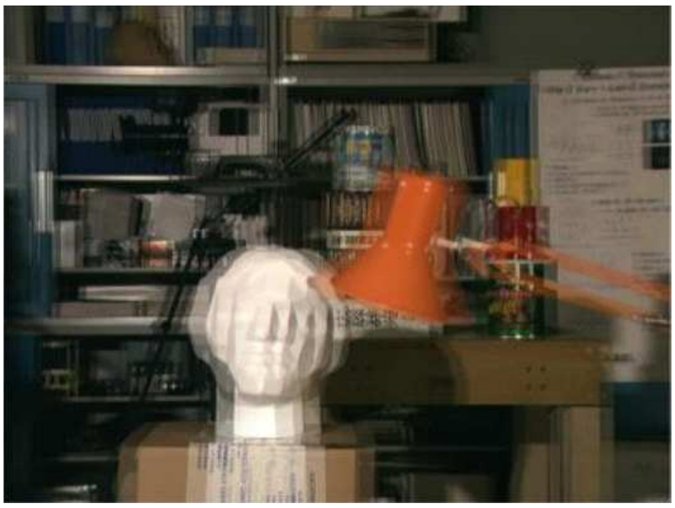


Figure 2.1: Overlapped stereo camera scene

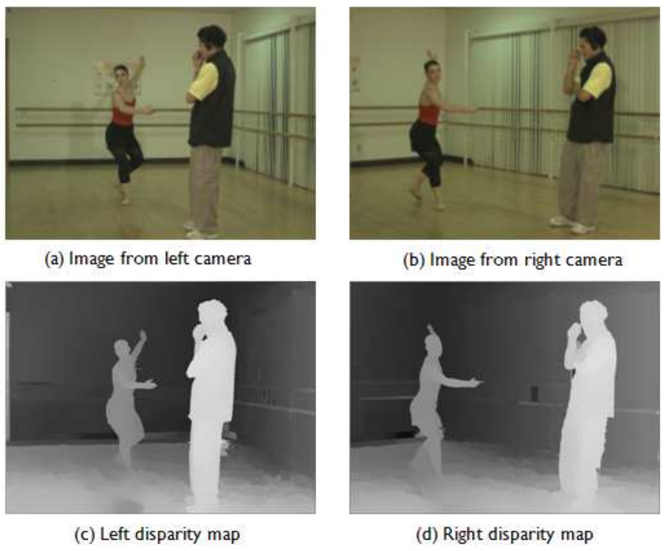


Figure 2.2: Example of disparity maps

Image rectification is an important step to simplify stereo matching searching space (pixel correspondence exploration) from two dimensions to one dimension searching space. Because the stereo scenes are not always taken from the well horizontally-aligned cameras, as Figure 2.3 shows, the mapping work is based on the concept of Epipolar line geometry [22][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 [55][41][8] uses the intrinsic and extrinsic camera parameters (camera calibration information) which includes camera rotation, translation, lenses distortion, rotation, etc to align the Epipolar line geometry of two projected points to new image plane. The stereo matching problem is therefore simplfied to one dimensional searching space (along the horizontal line).

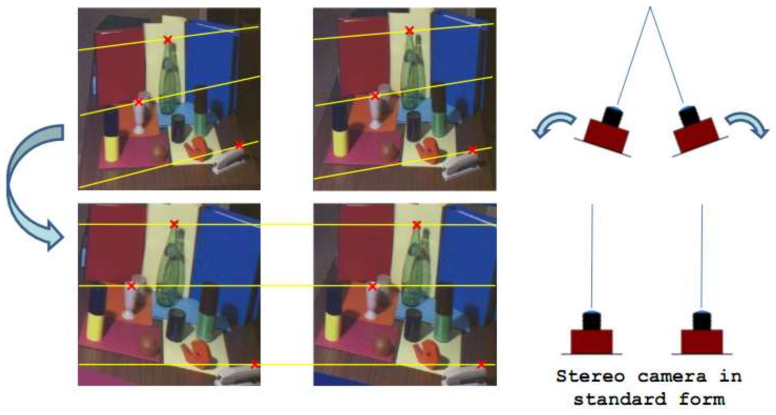


Figure 2.3: Stereo image pair rectification [38]

With the disparity map and stereo camera parameters, we can reconstruct the 3D model of view scenes by using triangulation principle. Figure 2.4 demonstrates the relations between stereo camera parameters and object P.

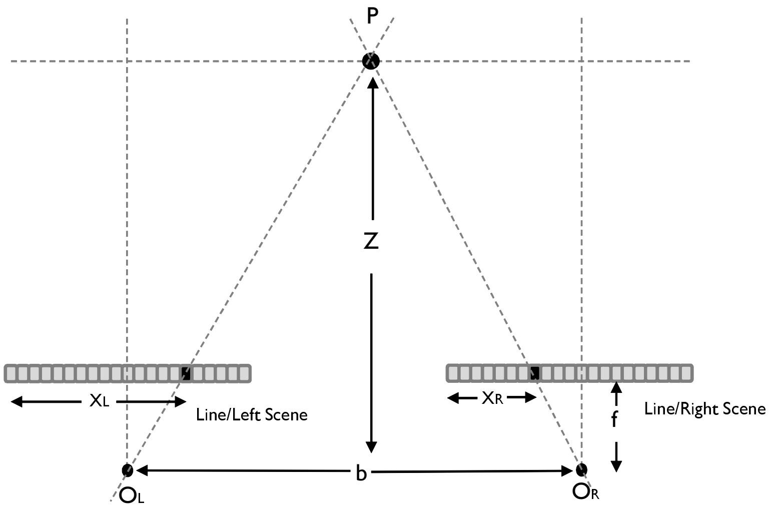


Figure 2.4: Example of triangulation principle. It depicts object P is captured from stereo cameras. Where f represents focal length, b represents the physical horizontal displacement between stereo cameras, and Z is the physical depth length from object P to stereo cameras.

Once the disparity of the object P between left and right scenes is known, its depth value Z can be calculated by refering to the camera focal length and baseline length as 2.1. Where the disparity d is generally constrained by a disparity range $[d_{min}, d_{max}]$. In addition, the X and Y position of object P can also be traced back by using 2.2.

$$Z = \frac{2 \times l \times f}{X_L - X_R} = \frac{b \times f}{d} \tag{2.1}$$

$$X = \frac{Z \times X_R}{f}, Y = \frac{Z \times Y_R}{f} \tag{2.2}$$

Based on the stereo matching principle, there are several common challenges to extract disparity map. Occlusion problem is the one should be concerned. It occurs when the lack of correspondence matching position in stereo image pairs. Figure 2.5 illustrates that the marked regions are unmatchable on the stereo image pairs because the foreground objects block the background information as Figure 2.6. The same scenario occurs to both stereo scenes. In general, the occlusion regions are nearby object boundaries.

Luminance difference between stereo scenes is also a common challenge for stereo matching. Different camera sensors might capture images by using different intensity gain. Consequently, the luminance bias between left and right scenes easily leads to mismatching.

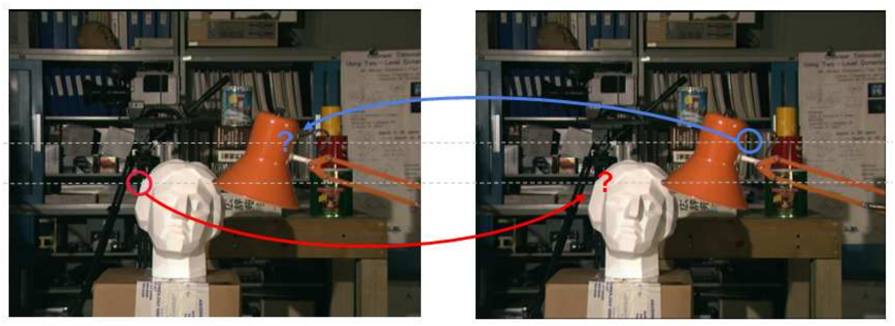


Figure 2.5: Example of occlusion problem

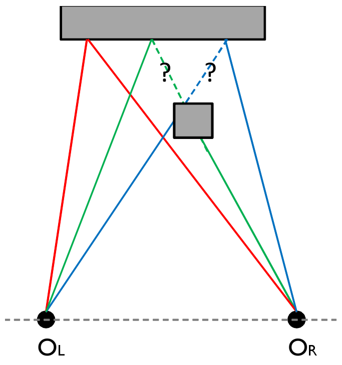


Figure 2.6: Example of occlusion problem

Another challenge for stereo matching principle is tthe textureless and repetitive regions. Figure 2.7 depicts the examples of texture less and repetitive regions. When trying to match the corresponding pixels for those regions in left and right scenes, we found the difficulty to make the right choice, which easily leads to mismatching.

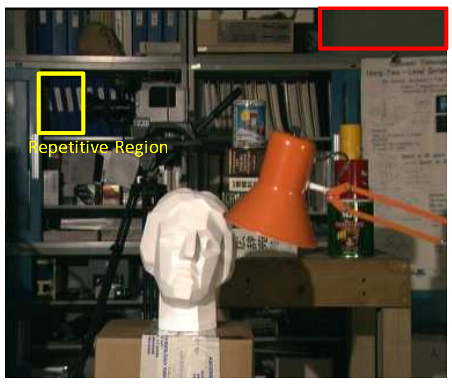


Figure 2.7: Example of textureless and repetitive regions

2.2 Stereo Matching Algorithms

There are two main classes of stereo 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 and object pattern). The other approach, correlation-based, refers to searching for the best correlation pixel on the target image by matching the homologous pixel within a disparity range. This thesis will focus on the correlation-based approach because of its robustness and simplicity for real-time hardware implementation.

According to the taxonomy summarized by Scharstein and Szeliski [16], the algorithms for correlation-based stereo matching can be categorized into two classes, the local-approach and the global-approach algorithms. The stereo matching computational flow can be summarized in four processing steps as Figure 2.8.

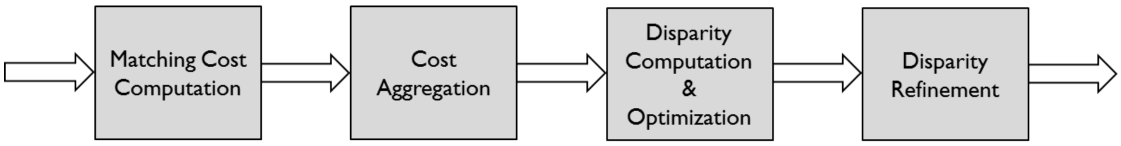

Matching Cost Computation
Cost Aggregation
Disparity Computation & Optimization
Disparity Refinement

Figure 2.8: Disparity map extraction flow

ps. Local stereo matching algorithm generally performs four processing steps but without global optimization. Global stereo matching algorithm sometimes ignore the step 2 (cost aggregation).

In the local-approach algorithms, the disparity is calculated by matching the corresponding pixel candidates within a disparity range. In order to increase the matching accuracy, the local approaches rely on cost aggregation step to cover more matching cost information from neighborhood pixels. Whereas the global-approach algorithms concentrate more on disparity computation and global optimization by introducing energy functions and smoothness assumptions. The energy function includes the matching cost information from the entire image. It reduces the chance of mismatching in some critical pixels/regions such as signal noise, occlusion, texture less, or repetitive pattern, etc. Although global-approaches improve the quality of disparity map, they require higher computational complexity demands and memory resources, compared to the local-approach methods, making a real-time system implementation challenging. All in all, there exists a trade-off between matching quality and computational complexity. In this section, we will introduce more detail about the stereo algorithm.

2.2.1 Matching Cost Computation

The variable, matching cost, is defined firstly between reference pixel and target matching candidates as Figure 2.9. Generally, the higher the raw matching cost represents the lower similarity between reference pixel and candidate matching pixel; vice versa.

The simplest method to define the matching cost is based on pixel color information. There are two common-used methods: absolute difference (AD) and squared difference

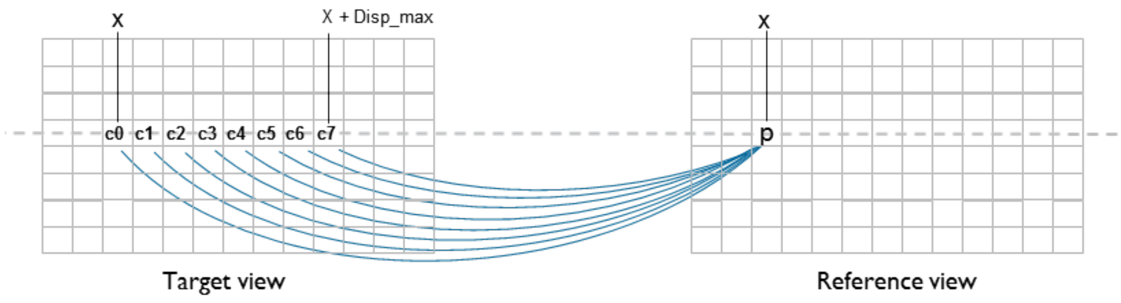


Figure 2.9: Fundamental pixel-to-pixel matching scenario

(SD) as Table 2.1 [51]. The raw matching cost value only takes pixel color information from stereo image pair. To enhance the pixel-based matching cost approach, area-based matching cost approaches [30] are proposed to improve the matching accuracy, which will be introduced in next sub-section.

Table 2.1: Definition of absolute difference and square difference

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

2.2.2 Matching Cost Aggregation

To improve the matching accuracy, cost aggregation is a solution to gather more neighbor pixels information. The pixel-based matching cost is extended to area-based approach. 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 as Table 2.2.

In the case of Census Transform matching cost function, the raw matching cost is aggregated by measuring the hamming distance between two census bit strings as 2.10. The hamming distance between 01001001 and 00001111 is 3 because 3 bits are unmatched.

In recent years, the Census Transform with Hamming Distance approach has become popular because of its robust performance. Chang [10], Heiko [21], Bleyer [36] have proofed Census Transform performs outstandingly against other approaches for stereo matching algorithm. The advantage of census-transform is it encodes the luminance relationship between the central pixel and neighbor pixels. Since no luminance

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)	$\sum_{(x,y)\in B(u,v)} \lvert I_{ref}(x,y) - I_{tar}(x+d,y)\rvert$
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.	$\frac{\sum_{(x,y)\in B(u,v)}(I_{ref}(x,y)-\mu_{tar}(x+d,y))\cdot(I_{tar}(x+d,y)-\mu_{tar}(u+d,v)))}{\sqrt[2]{\sum_{x,y\in B(u,v)}(I_{ref}(x,y)-\mu_{tar}(x+d,y))^2 \quad \cdot(I_tar(x+d,y)-\mu_tar(x+d,v))}}$
Rank	Rank transform calculates the number of neighbor pixels which have the value lager than the central pixel. The matching cost is calculated from the absolute difference of the two ranks.	$\sum_{(x,y)\in B(u,v)} \lvert Rank_{ref}(x,y) - Rank_{tar}(x+d,y)\rvert$ $Rank(u,v) = \sum_{(i,j)\in R(u,v)} L(i,j)$ $L(i,j) = \begin{cases} 0\ , & I(i,j) > I(u,v) \\ 1\ , & I(i,j) \geq I(u,v) \end{cases}$
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.	$f(u,v,d) = \sum_{(x,y)\in B(u,v)} Hamming(Census_{ref}(x,y) - Census_{tar}(x+d,y))$ $Census(u,v) = Bitstring_{(i,j)\in R(u,v)}(I(i,j) \geq I(u,v))$

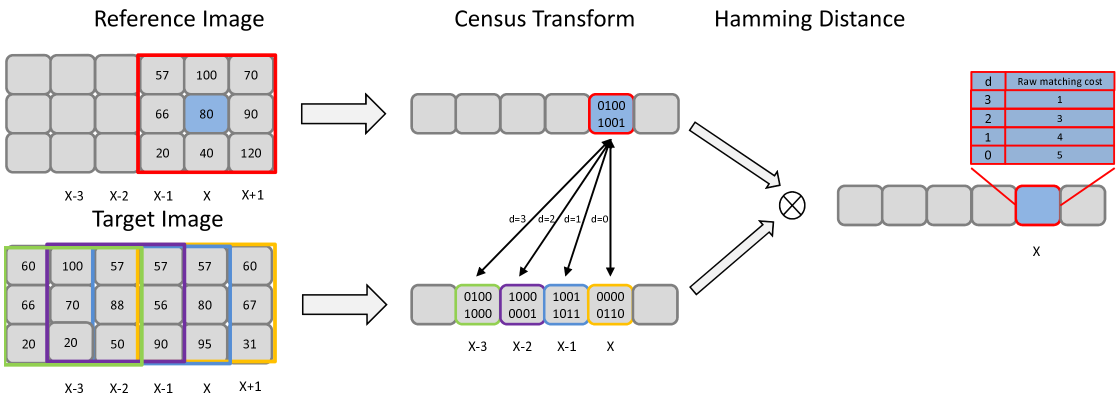


Figure 2.10: Census transform and hamming distance to generate raw matching cost

or chrominance value is involved in generating the matching cost, any luminance and gamma variations of stereo sources doesn't effect the final result.

Now the question is how to define the cost aggregsion region? So, we further investigate the commonly used aggregation region strategies from coarse to fine. Figure 2.11 shows different aggregation strategies [61] include fixed window, multiple window, adaptive shape, and adaptive weight. Fixed window is a efficient approach, which requires low computation complexity. However, it performs badly to the regions such as boundary, slant surface, and repetitive pattern. Therefore, multiple windows approach [38] is a advanced solution. 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 the image in regions with similar color intensity and aggregates costs within the similar segmentation. The last but the most accurate one is adaptive weight [29] [24]. 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 against to other methods. However, the computational complexity is higher than others.

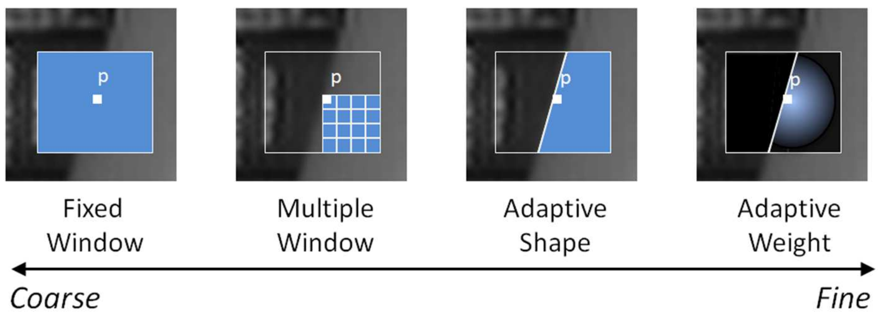


Figure 2.11: Categories matching cost aggregation approaches from coarse to fine.

2.2.3 Disparity Computation and Optimization

There are two categories of disparity computation approaches: local stereo matching and global optimization method. Both approaches have their pros and cons to computational resource and matching accuracy. Their details will be introduced in the following:

[Local Stereo Matching]

The local stereo matching method computes the disparity value by selecting the matching candidate which possesses the minimum matching cost value as Figure 2.12. This method is so-called winner-take-all (WTA) strategy [16]. Where the matching cost generation methods were mentioned in the previous sub-section. Then the displacement between the reference pixel and the maching pixel is regarded as the disparity result.

[Global Optimization]

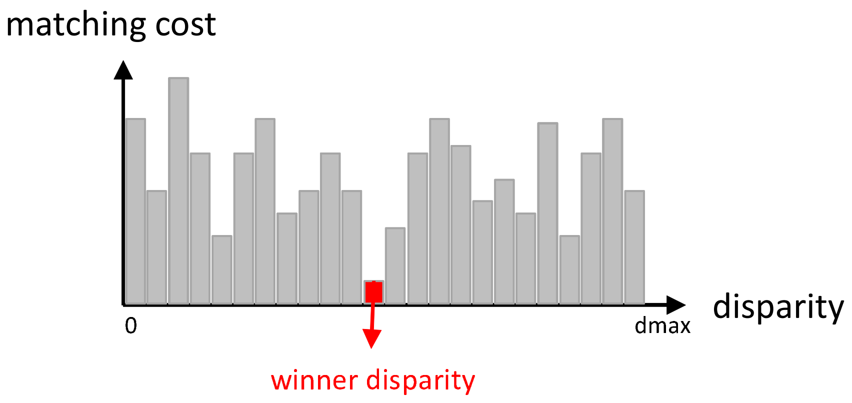


Figure 2.12: Example of winner take all strategy(WTA)

The global algorithm improve the WTA strategy by introducing the energy function with smoothness assumptions. The energy function is defined as 2.3. We can simply regard the matching strategy as an energy minimization problem.

$$E(d) = E_{data}(d) + E_{smooth}(d) \tag{2.3}$$

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

$$E_{data}(d) = \sum_{j \in N} C(j, d) \tag{2.4}$$

Where C( ) is matching cost function, and N represents global pixels.

$$E_{smooth}(d) = \sum_{j \in N, d' \in [0, D_{max}-1]} \lambda \cdot S(d, d') \tag{2.5}$$

where S( ) represents smoothness function. λ is a scaling coefficient, which adapts to the luminance variation of adjacent pixels. For example, it provides sharper disparity discontinuity when encountering edge regions; whereas it stabilize the disparity variation when processing the surface of objects.

The first term, 2.4, illustrates the sum of matching costs in a disparity range number of arrary. The second term, 2.5, is a smoothness function, which generates a penalty value for smoothness assumption based on the disparity distances of adjacent pixels. The idea of smoothness assumption 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 smoothness of disparity map.

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 [23], Graph Cut [7], and Belief Propagation [44]. In this thesis, we will focus on scanline dynamic programming algorithm because of its hardware-friendly trait.

In the following, we introduce two type of smoothness models: first order and second (higher) order smoothness functions. The most commonly used smoothness function models are summarized in Table 2.3.

Table 2.3: Common used smoothness function models

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

In the linear model, the higher distance between the disparity array of adjacent pixels d and d' intorduces the larger smoothness penalty. Figure 2.13 is an example shows the gradient penalty costs are imposed on heterogenous disparity positions. This leads the disparity result changes gradually only within small steps. Hence, the linear model performs outstanding in extracting the disparity information for slanted surfaces.

However, the linear model performs badly in the disparity continuity regions (edge). Figure 2.14 is an example shows the blurrd 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 2.16).

The Potts model can preserve the edge information of objects in disparity map by imposing a constant cost penalty to any heterogenous disparity position. Figure 2.17 illustrates the cost penalties are added on the disparity dissimilar positions in energy function. Figure 2.15 is an example of disparity map by using Potts model, which performs superior in disparity discontinuous regions against 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 level of constant cost penalty to adjacent disparities. The modification not only allows the Potts model to handle slight slanted surfaces but also permits it to reconstruct disparity discontinuous regions.

The Second (higher) order model is proposed by Woodford [57], which improves the

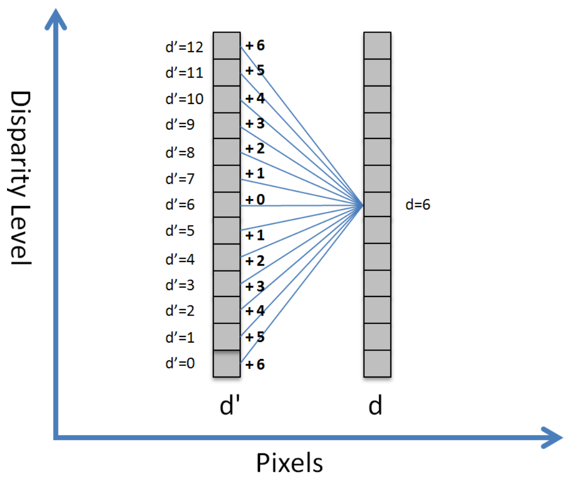


Figure 2.13: Example of smoothness cost penlties by linear model

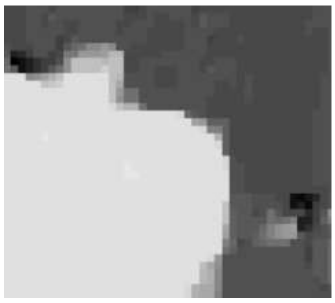


Figure 2.14: Linear Model

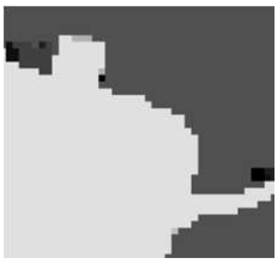


Figure 2.15: Potts Model

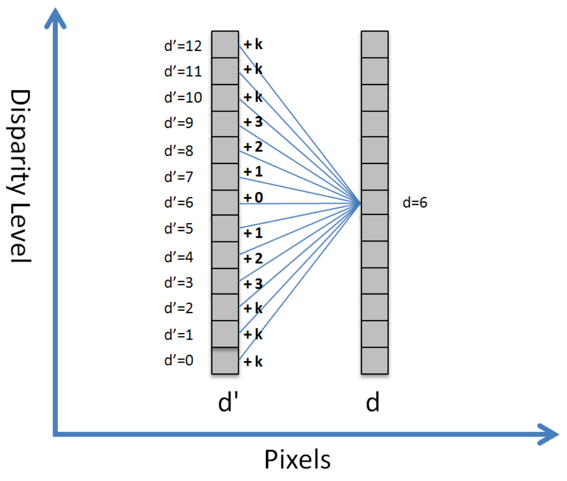


Figure 2.16: Example of smoothness cost penlties by truncated linear model

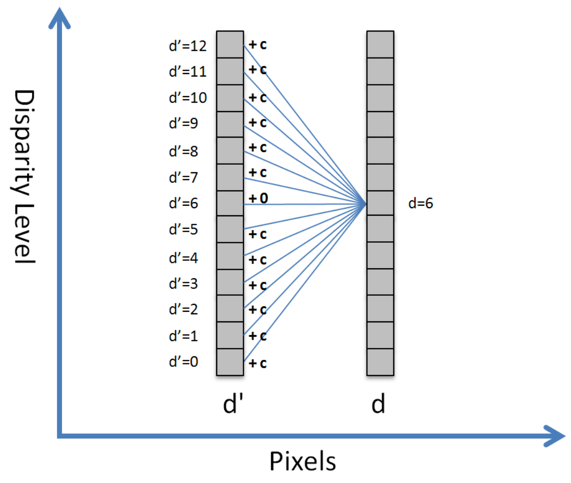


Figure 2.17: Example of smoothness cost penlty by Potts model

disparity map in curvature surface regions. The second order model detects slanted planes and assigns lower cost penalty to them. Besides, the truncated principle can also be applied to this model. In recent researches, this possesses the state-of-the-art quality against the above mentioned models. However, the computational complexity is relatively high.

The above-mentioned energy function principle is broadly apply to global stereo matching. Global stereo matching approach extends the local Winner-Take-All strategy to global energy minimization problem. The state-of-art global stereo matching algorithms such as graph cuts [7], belief propagation [44], and dynamic programming [23] are all using energy function as their background. In this thesis, we will take the scanline-based dynamic programming algorithm as an example to explain how energy minimization can be used for stereo matching.

Dynamic Programming

Dynamic programming (DP) is a technique to used to solve the complex problem by breaking them down into subproblems. It solves each subproblem only once, which greatly reduces the computational complexity. Thus, we can apply the dynamic programming principle to solve energy minimization problem (minimizing the energy function $E(D)$) in stereo matching. By using energy function, it is possible to achieve the nearly optimal disparity map solution. In the following, we demonstrates how dynamic programming principle can be applied to find the optimal matching disparity along a image scanline.

After the cost aggregation step, each pixel position has D_{max} (maximum disparity range) number of matching cost array $C(j, d)$, where the term j represents the pixel position along a scanline and d is defined as disparity range. The dynamic programming algorithm requires two processing steps, forward pass and backward pass, to find the optimal disparity solution for a scanline.

1. **Forward Pass**

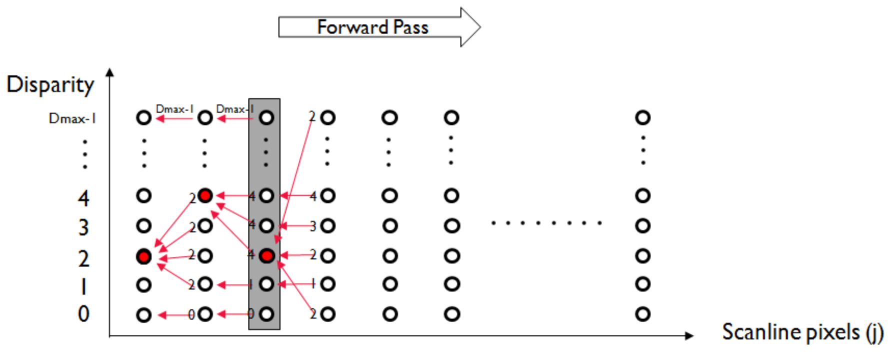


Figure 2.18: Example of forward pass function

The forward pass procedure seeks optimal backward disparity entries (path) along the scanline by selecting the minimum matching cost energy with smoothness assumption. Figure 2.18 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 forward pass formula, Equation 2.6, 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 forward pass iteration, the winner of minimum aggregation cost energy (second term) will be summed with matching cost (first term) as an aggregation cost again. When executing the forward pass step, the dynamic programming tree path information are stored in Backward_Path array Equation 2.7 for the later backward pass step.

$$C_{agg}(j,d) = C_{raw}(j,d) + min_{j \in scanline \quad W, d' \in [0, D_{max}-1]} \{C_{agg}(j-1,d') + S(d,d')\} \quad (2.6)$$

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, and S( ) is smoothness cost function.

$$Backward\_Path(j,d) = arg \quad min_{d' \in [0, D_{max}-1]} (C_{agg}(j-1,d') + S(d,d')) \quad (2.7)$$

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

2. **Backward Pass**

The backward pass step tracks the backward pass paths iteratively in order to obtain the optimal disparity map scanline solution. In the beginning of the backward pass step, the initial disparity entry of the last disparity pixel is decided by the minimum winner of disparity matching cost candidates as Equation 2.8. Afterward, the backward path step, Equation 2.9, starts to trace back along the backward paths from the last pixel of scanline back to first pixel. Finally, the final backward pass path is regarded as the optimal disparity solution over the scanline. Figure 2.19 shows the backward pass procedure as an example.

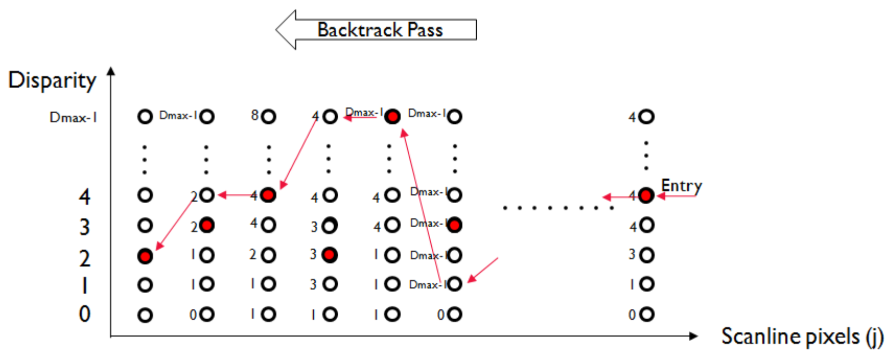


Figure 2.19: Example of backward pass procedure

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

$$d(W-1) = arg \;\; min_{d\in[0,D]} C_{agg}(W-1, d) \tag{2.8}$$

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:

$$d(j-1) = Backward\_Path(j, d(j)) \tag{2.9}$$

The above-mentioned scanline based dynamic programming approach can be further improved. Olga Veksler [54] extends the scanline structure to tree structure for dynamic programming based stereo matching, which has no restriction on accumulating the cost energies only within scanline. Another proposals [31][20], two pass dynamic programming, solves the inter-scanline inconsistency problem by executing dynamic programming on horizontal scanlines firstly then do vertical passes.

2.2.4 Disparity Map Refinement

Disparity map refinment is usually used in the post processor unit of stereo matching algorithm to improve the matching quality. This sub-section summarizes the solutions for the problems such as occlusion region, disparity mismatching, and sparkle noise.

Left-Right consistency check is a commonly used technique to detect the mismatched regions on disparity maps. After extracting the left and right disparity maps, Equation 2.10 can be used to check whether the disparity map pixels on left disparity map share the same disparity values with the corresponding pixels on the right disparity map. The same scenario can be applied to check the consistency of right disparity map by Equation 2.11. If the disparity value difference between a pixel and its corresponding pixel is beyond a threshold value, it is regarded as inconsistent pixel, which is not a reliable result.

Consistency check of left disparity map

$$\begin{cases} Good \;\; Disparity \;, & if \;\; |d_{right'}(x - d_{left}(x,y), y) - d_{left}(x,y)| \leq Threshold \\ Occlusion \;, & if \;\; |d_{right'}(x - d_{left}(x,y), y) - d_{left}(x,y)| > Threshold \end{cases} \tag{2.10}$$

Consistency check of right disparity map

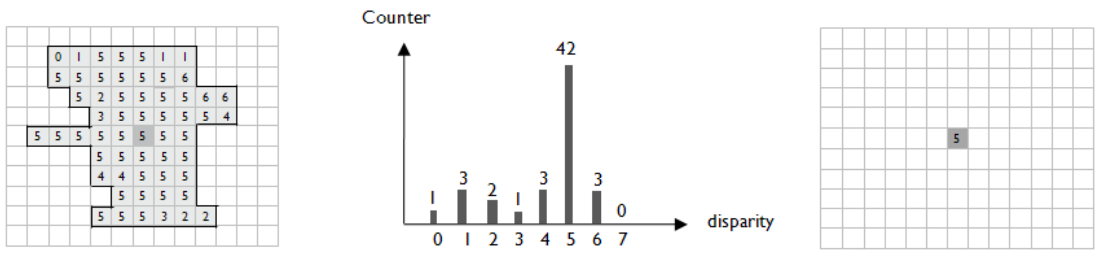
$$\begin{cases} Good\ \ Disparity\ , & if\ \ |d_{left'}(x+d_{right}(x,y),y)-d_{right}(x,y)| \leq Threshold \\ Occlusion\ , & if\ \ |d_{left'}(x+d_{right}(x,y),y)-d_{right}(x,y)| > Threshold \end{cases} \tag{2.11}$$

One simple solution to fix the mismatched pixels is replacing them by the nearest good disparity pixels. This method assumes the occlusion regions that tend to produce mismatching share the same disparity value with the nearest background objects. Based on this assumption, the mismatched pixels on the left disparity map can be replaced by the nearest good disparity pixel from its left side. On the contrary, the mismatching pixels of the right disparity map can be replaced by the nearest good disparity pixel from its right side. Figure 2.20 demonstrates that the mismatching pixels can be replaced by the nearest good disparity pixels from the nearest good disparity pixels. Since this method only performs on horizontal scanline (one dimension), it provides a simple solution for dispairty map refinement.

(a) Occlusion position in left image (b) Occlusion position in right image

Figure 2.20: Simple occlusion handling method

Another commonly used solution for disparity map refinment is segmentation based method [18][32][58][6][45][43]. Most proposals apply color segmentation strategy. The general idea assumes the segmentation with similar color intensity shares the same disparity value. Figure 2.21 demonstrates an example of 2D support region voting for disparity map refinement. Where the disparity voting stage counts the disparity values within the support region (segmentation) to the central pixel into a histogram and then chose the winner as its final disparity value.



(a) Disparities in support region (b) Disparities in support region (c) Voted disparity

Figure 2.21: 2D disparity voting

In this thesis we introduce cross-based algorithm [60] that is proposed by our group member Zhang. It's principle is based on segmentation to achieve disparity map refinement. The cross-based algorithm firstly searches the support region in four directions (upper, lower, left and right) for each pixel. Where the cross-based support region is obtained by simply extending the central pixel until two adjacent pixels are not consistent in color or reaches the maximum arm length. To find out the arm length, the method is according to Equation 2.12 and Equation 2.13.

$$r^* = max_{r\in[1,L]}(r\Pi_{i\in[1,r]}\delta(p,p_i)) \tag{2.12}$$

Where r^* indicates the largest span for the four direction arms. p_i represents $(x_p - i, y_p)$. and L corresponds to the predetermined maximum arm length.

$$\delta(p_1,p_2) = \begin{cases} 1, & max_{c\in[R,G,B]}(|I_c(p_1) - I_c(p_2)|) \le Threshold \\ 0, & otherwise \end{cases} \tag{2.13}$$

After having the cross arm information of each pixel, we can reconstruct their segmentations (support regions) through entired image. Figure 2.22 demonstrates the support region of the anchor pixel p. The four arms of pixel $p(h_p^-, h_p^+, v_p^-, v_p^+)$ are used to define the horizontal segment H(p) and vertical segment V(p). Then the full support region U(p) is the integration of all the horizontal segments of those pixels residing on the vertical segment.

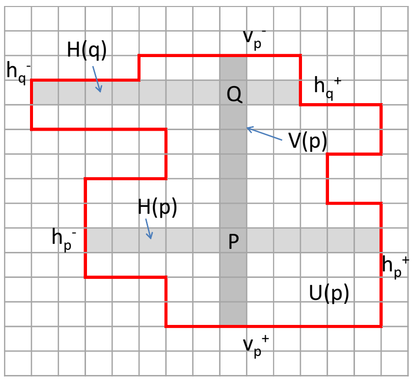


Figure 2.22: The support region of anchor pixel p

Once the support region has been determined, we can use disparity voting method to enhance the disparity value consistency within support region. The disparity value has maximum population within the support region is regarded as the final voting result. Figure 2.23 shows the example of disparity voting.

In Lu's proposed design [62], he simplified the 2D voting into two-pass 1D voting in order to reduce the computational complexity as Figure 2.24. 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.

Finally, we introduce median filter for disparity map refinement. It is wildly used in image processing applications to eliminate the so-called pepper and salt noises. Median filter selects the median value within the window to the anchor pixel as the final disparity value. It helps alleviate the speckle and impulsive noises.

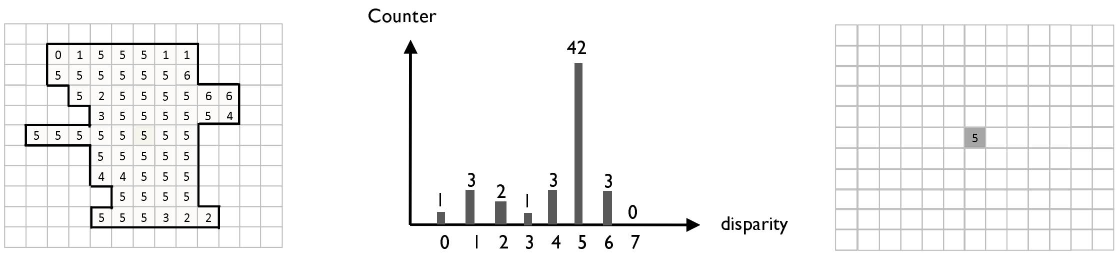


Figure 2.23: The example of disparity voting

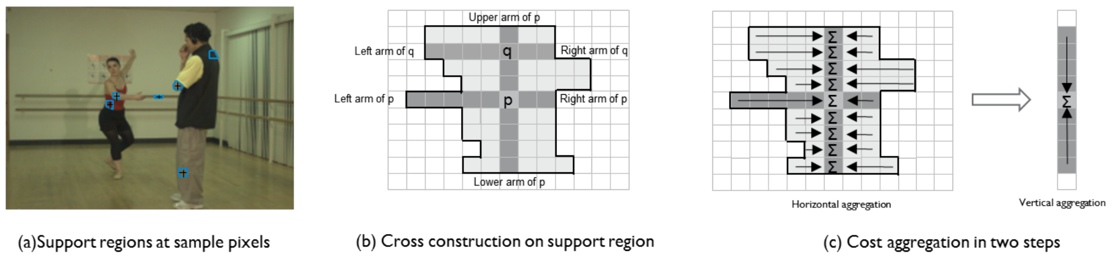


Figure 2.24: Two-pass 1D disparity voting

2.3 Temporally Consistent Disparity Map Sequences

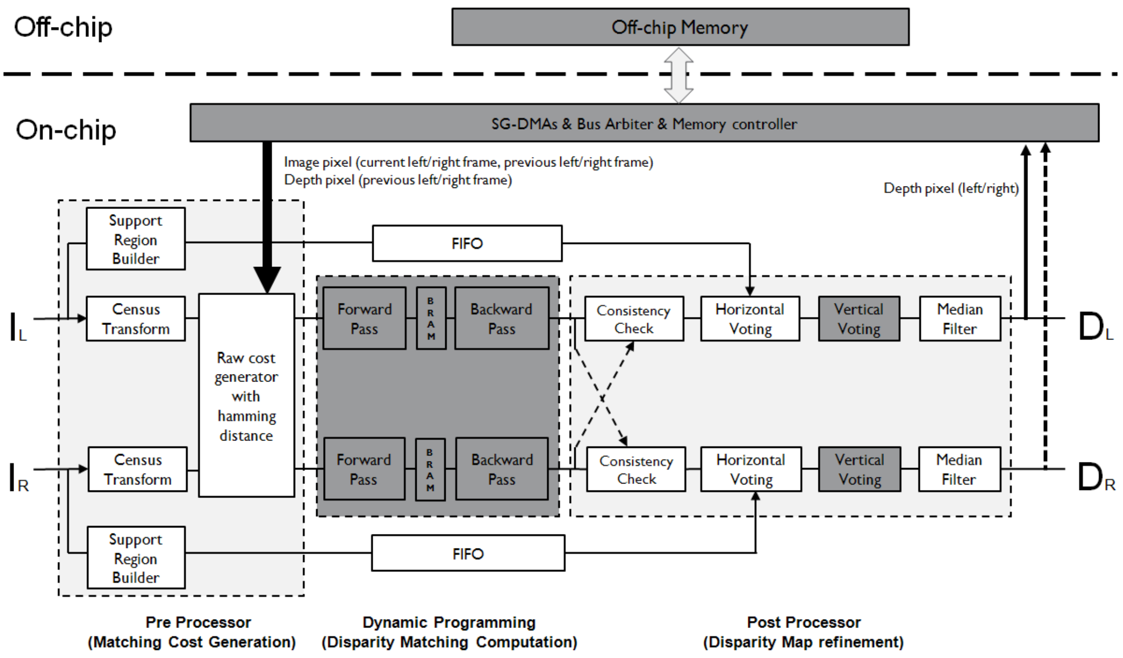
The static parts of disparity map sequences are expected to remain the same disparity value. Some researches [13][35][11] expand the disparity estimation from single frame to video sequences. The general idea is to update the matching cost according to the static regions. DERS, a disparity map generation reference software, is contributed from MPEG community can achieve 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). In the static regions, the raw cost corresponding to the disparity value of the previous frame is scaled down. This leads to the static regions keep the same disparity result again, and the stability of disparity map sequences is enforced.

2.4 Implementation Platforms

CPU, GPU, DSP, and FPGA/ASIC are commonly used platforms to realize stereo matching algorithms. Several aspects such as matching accuracy, robustness, real-time performance, computational complexity and scalability should be taken into consideration into the design based on the application. Zhang[60][61] achieve a real-time design with high matching accuracy by CUDA. Banz[3] also realize the Semi-Global algorithm[23] to a GPU platform. Unfortunately, the GPU based design is not always flexible to computational logic, instructions and data paths. Besides, high clock frequency and memory bandwidth are required. FPGA and ASIC platforms are the alternative options which allow parallelism exploration and a pipeline architecture to achieve a higher throughput at moderate clock speeds. Chang[9] implement a high performance stereo matching algorithm with mini-Census Transform and Adaptive Support Weight on UMC 90 nm ASIC, achieving 352288@42FPS with 64 disparity ranges. Jin[28] design a pipelines hardware architecture with the Census Transform and sum of hamming distances, achieving 640480@30FPS at a 64 disparity range under 20MHz. Banz[4] realize the Semi-global matching algorithm on a hybrid FPGA/RISC architecture, achieving 640480@30FPS at a 64 disparity range under 12M 208MHz. Stefan[19] also utilize Semi-global matching algorithm to implement a stereo matching engine, which achieves 340200@27FPS with a disparity range of 64 levels. Zhang[63] implement a local algorithm with a Cross-Based support region in the cost aggregation and refinement stages, achieving over a range of 64 disparity levels with a resolution of 1024768@60FPS under 65MHz.

Stereo Matching Algorithm Implementation and Optimization on Hardware 3

This chapter implements our stereo matching algorithm to SoC architecture as Figure 3.1. We improve the design that was contributed by Zhang's work [62][63] by replacing the local winner-take-all algorithm to semi-global scanline-based dynamic programming.



Off-chip
Off-chip Memory
On-chip
SG-DMAs & Bus Arbiter & Memory controller
Image pixel (current left/right frame, previous left/right frame)
Depth pixel (previous left/right frame)
Depth pixel (left/right)
Support Region Builder
Census Transform
Raw cost generator with hamming distance
FIFO
Forward Pass
BRAM
Backward Pass
Consistency Check
Horizontal Voting
Vertical Voting
Median Filter
I_L
I_R
D_L
D_R
Pre Processor (Matching Cost Generation)
Dynamic Programming (Disparity Matching Computation)
Post Processor (Disparity Map refinement)

Figure 3.1: System architecture of stereo matching engine

It performs the following steps:

1. Pre Processor: support region builder, raw cost generator with Census Transform and Hamming distance
2. Dynamic programming based stereo matching
3. Post Processor: L-R consistency check, disparity voting, and median filter

The arechitecture includes three parts: pre-processor, dynamic programming, and post-processor. In the design of pre-processor, the raw matching costs are generated from census transformhamming distance and vertical aggregation functions. In the disparity matching processor, dynamic programming algorithm is chosen for disparity map computation and optimization. Finally, the post-processor refines the disparity maps by using L-R consistency check technique, cross-based disparity voting, and median filter. Beyond the stereo matching engine, a external memory hierarchy is designed to support frame bufferring function for enhancing the temporal consistency of disparity map sequences. In this thesis, we will propose a solution for the gray regions which include Dynamic Programming and Vertical Voting functions. The peripheral components, video adaptors, color space converters, and memory hierarchy, will be proposed in Chapter 5 to support IMEC's 3D TV SoC.

Two proposals for Dynamic Programming and Vertical Voting functions are designed 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 proposes in Section 3.2 for reducing the on-chip memory consumption of the Vertical Voting Processor.

Temporal inconsistency is a common issue to disparity map sequences. It is because of 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 inconsist. 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, which takes 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.

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

where the smoothness function $S(d,d') = \begin{cases} 0\,, & if \;\; d = d' \\ C\,, & if \;\; otherwise \end{cases}$ 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.

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

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.

$$Backward\_Path(j,d) = \begin{cases} 1\,, & if \;\; C_{minAssum} \leq C_{agg}(j-1,d) \\ 0\,, & if \;\; C_{minAssum} > C_{agg}(j-1,d) \end{cases} \tag{3.4}$$
$$Backward\_MinC\_Path(j) = arg \;\; min_{d\in[0,D-1]}\{C_{agg}(j-1,d)\}$$

where $j \in [0, \text{Image Width} - 1]$, and D represents maximum disparity range.

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

where W represents scanline width
D_{max} is maximum disparity range
D_{bit} is the bit numberof 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.

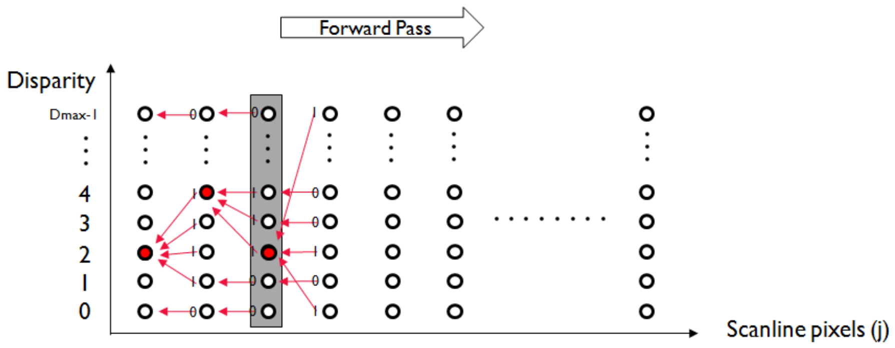


Figure 3.2: Forward pass with backward path encoding

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.

$$d(j-1) = \begin{cases} Backward\_MinC\_Path(j)\,, & if \;\; Backward\_Path(j,d) = 1 \\ d(j)\,, & if \;\; Backward\_Path(j,d) = 0 \end{cases} \tag{3.6}$$

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 allows that writing and reading commands can operate concurrently. As Figure 3.4

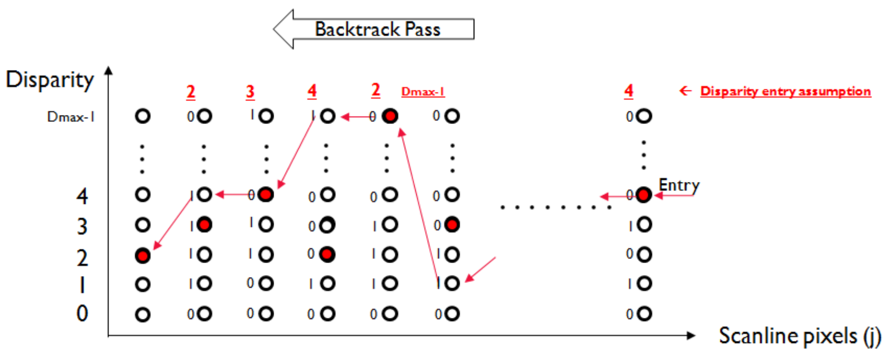
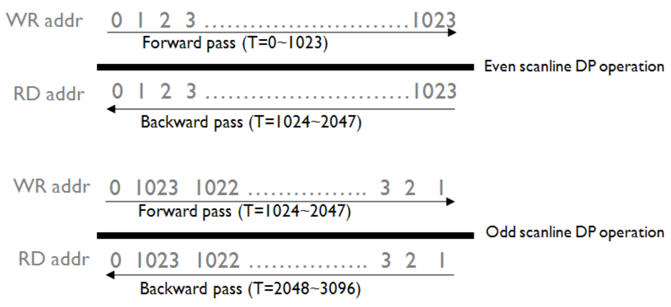


Figure 3.3: Example of backward pass with decoded path data

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.



WR addr 0 1 2 3 1023
Forward pass (T=0~1023)
Even scanline DP operation
RD addr 0 1 2 3 1023
Backward pass (T=1024~2047)
WR addr 0 1023 1022 .... 3 2 1
Forward pass (T=1024~2047)
Odd scanline DP operation
RD addr 0 1023 1022 .... 3 2 1
Backward pass (T=2048~3096)

Figure 3.4: DP operation sequence

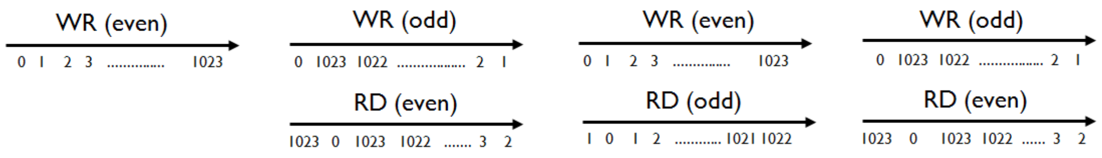


Figure 3.5: Example of 2-Port RAM access patterns

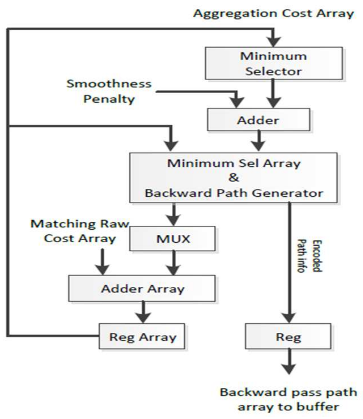


Figure 3.6: Forward Pass HW design

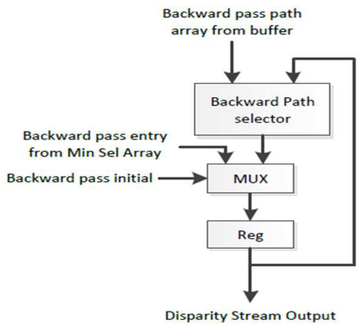

Backward pass path array from buffer
Backward Path selector
Backward pass entry from Min Sel Array
Backward pass initial
MUX
Reg
Disparity Stream Output

Figure 3.7: Backward Pass HW design

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 pass computations should be completed within 1 clock cycle. The forward pass function processes one set of matching cost array 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 pixel concurrently.

3.2 Run-Length Coding Algorithm and Disparity Map Sequence

In this section, we propose using a run-length coding algorithm to compress disparity map data. Then we apply this idea to reduce the on-chip memory utilization of Vertical Voting processor.

From our observation, the output sequence of a disparity map tends to show a long repetitive disparity value. 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)[39]. 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 [62] architecture to reduce the on-chip memory resource. 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-

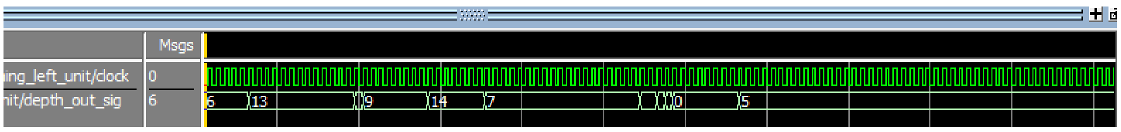


Figure 3.8: Disparity output from dynamic programming function. The depth/disparity value can be represented in run-length coding format [Repeat count, Disparity value]: [12, 13], [0, 5], [7, 9], [6, 14], [18, 7], [1, 2], [0, 3], [0, 2], [7, 0], [31, 5], [15, 5].

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
```

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 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 does't generate fixed-length ouputs. The length of encoded sequence changes depending on the complexity of the disparity

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
```

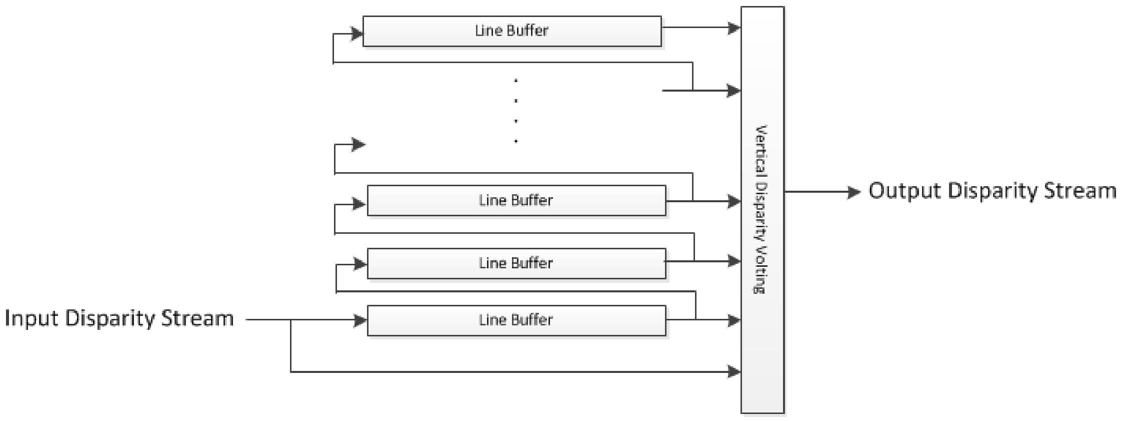


Figure 3.9: Circular memory architecture for Vertical Diaprity Voting processor

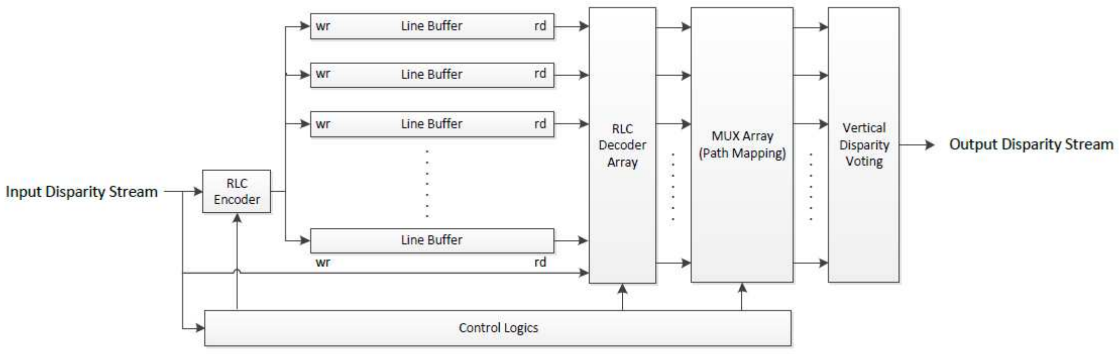


Figure 3.10: Proposed memory reduction architecture with run-length coding

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 luminance of the pixels in current frame remains the same as previous frame, it assumes their disparity value are unchanged. 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.

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

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.

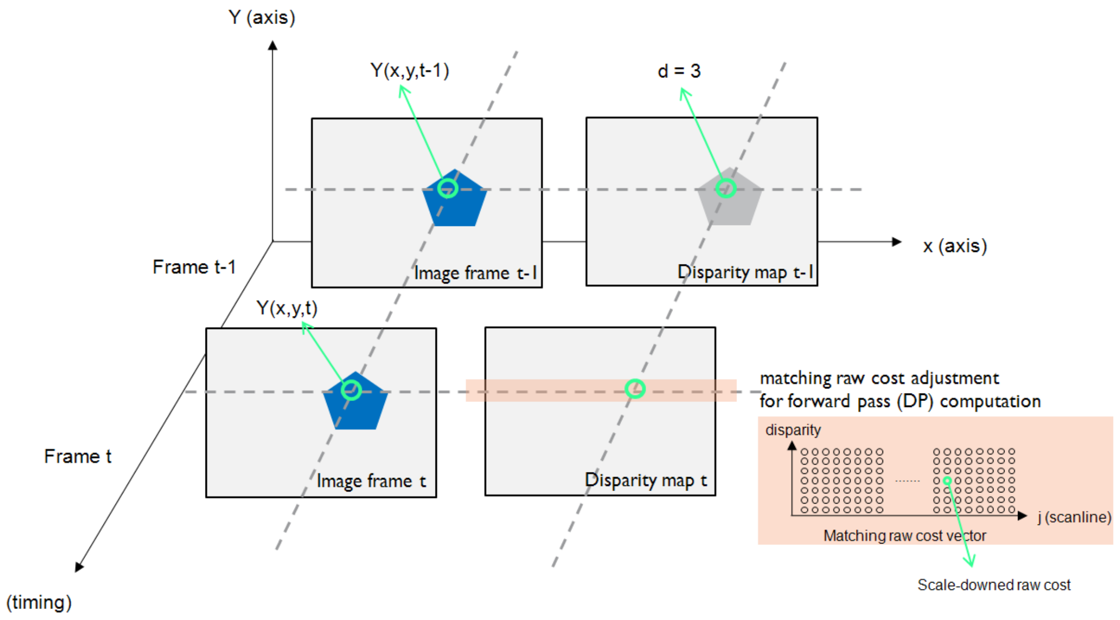


Figure 3.11: Example of temporal consistency algorithm in dynamic programming

The on-chip memory is not a cost efficient solution to store complete frame information(previous luminance map and disparity map). Hence, we will propose a dedicated memory hierarchy to support the temporal consistency function in Chapter 5.

Evaluation of the proposed Stereo Matching Hardware 4

This chapter summaries the experimental results of our stereo matching engine design. The proposed algorithms and hardware architectures are co-evaluated from three points of view:

- Quality of disparity map and disparity sequences
- Hardware utilization and scalibility
- 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 generate the disparity map or video sequences by using RTL simulation tools such as Modelsim and Candence Simulator.

Two evaluation models [16], Root-Mean-Squared Error(RMS) Equation 4.1 and Bad Matching Pixel(B) Equation 4.2, are usually chosen to measure the disparity map quality. The generated disparity map is compared with the ground truth disparity map. We use an academic evaluation benchmark tool, Middlebury Stereo Evaluation Benchmark [52], which is provided by Middlebury University to execute the measurement. It applies Bad Matching Pixel model to evaluate disparity map quality. The Middlebury Stereo Evaluation website provides three error metrics: unconcluded, complete image and disparity discontinuity regions. This provides designer with more flexibility to focus on improving specific regions.

1. Root-Mean-Squared Error

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

   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

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

   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 [53]) are tested with our algorithm.

4.1.1 Parameter Exploration for Dynamic Programming Processor

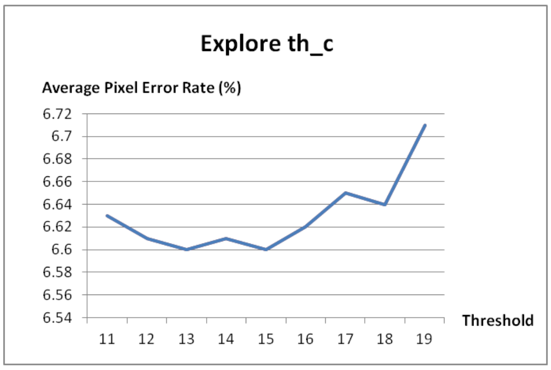
There are two parameters have great effects on the performance of Dynamic Programming Processor. One is the threshold for the absolute difference of adjacent pixel luminances. If the absolute difference is larger than the threshold, the continuous boundary is assumed on the disparity map. The other parameter 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 for 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 shown in Figure 4.3.

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 [63], 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%.



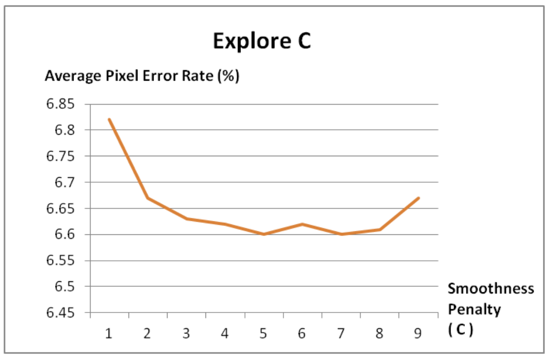
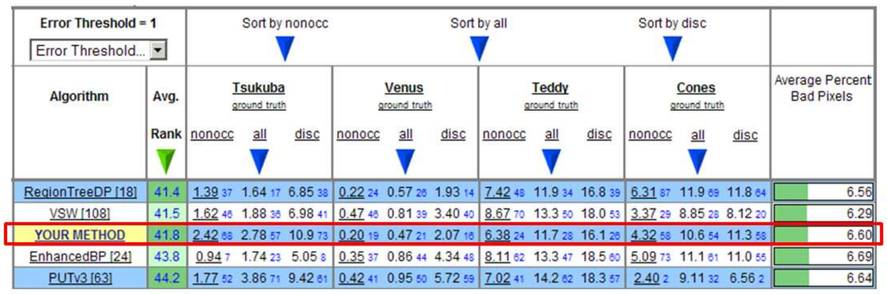


Figure 4.1: Parameter exploration for Dynamic Programming


Error Threshold = 1
Error Threshold...
Sort by nonocc
Sort by all
Sort by disc

Algorithm	Avg. Rank	Tsukuba nonocc	Tsukuba all	Tsukuba disc	Venus nonocc	Venus all	Venus disc	Teddy nonocc	Teddy all	Teddy disc	Cones nonocc	Cones all	Cones disc	Average Percent Bad Pixels
RegionTreeDP [18]	41.4	1.39 37	1.64 17	6.85 38	0.22 24	0.57 26	1.93 14	7.42 48	11.9 34	16.8 39	6.31 87	11.9 69	11.8 64	6.56
VSW [108]	41.5	1.62 46	1.88 36	6.98 41	0.47 46	0.81 39	3.40 40	8.67 70	13.3 50	18.0 53	3.37 29	8.85 28	8.12 20	6.29
YOUR METHOD	41.8	2.42 68	2.78 57	10.9 73	0.20 19	0.47 21	2.07 16	6.38 24	11.7 28	16.1 26	4.32 58	10.6 54	11.3 58	6.60
EnhancedBP [24]	43.8	0.94 7	1.74 23	5.05 8	0.35 37	0.86 44	4.34 48	8.11 62	13.3 47	18.5 60	5.09 73	11.1 61	11.0 55	6.69
PUTv3 [63]	44.2	1.77 52	3.86 71	9.42 61	0.42 41	0.95 50	5.72 59	7.02 41	14.2 62	18.3 57	2.40 2	9.11 32	6.56 2	6.64

Figure 4.2: Evaluation results from Middleburry's benchmark

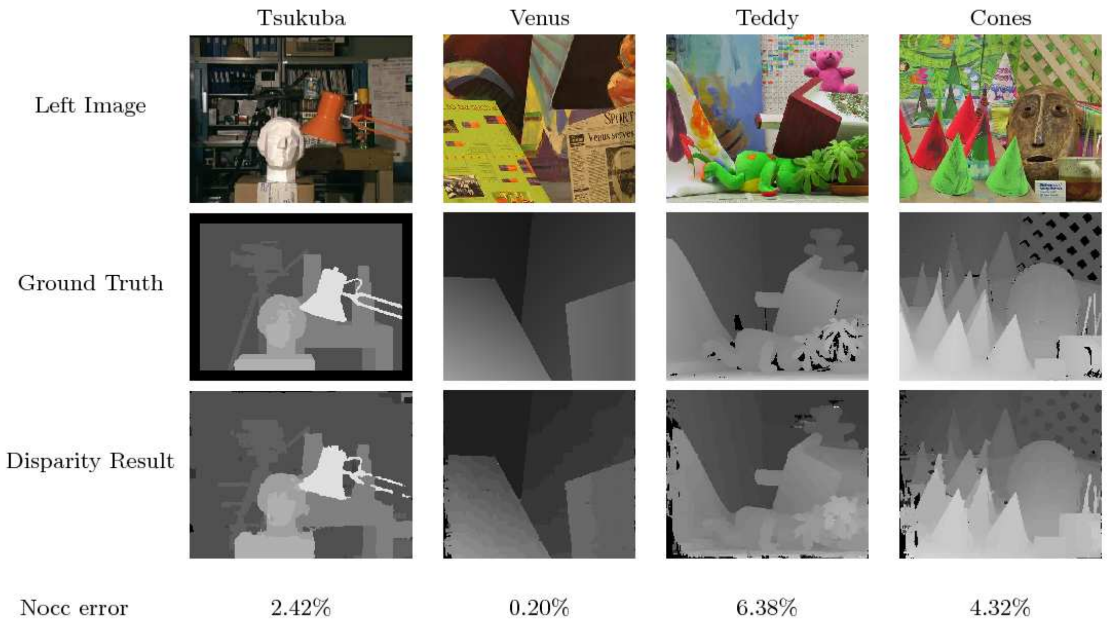


Figure 4.3: Ground truth disparity maps and test disparity maps

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 by using View Synthesis Reference Software (VSRS) [46], 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 [64], to measure the temporal quality of the synthesized view. The 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 [64] 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.

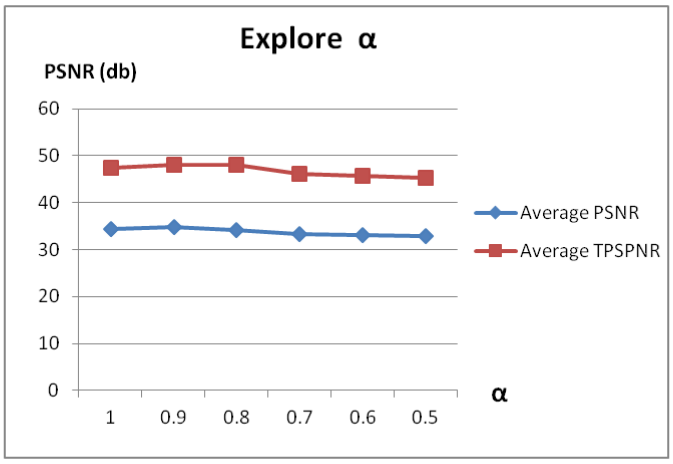


Figure 4.4: Exploration of matching cost scaling parameter α with Book Arrival test set

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

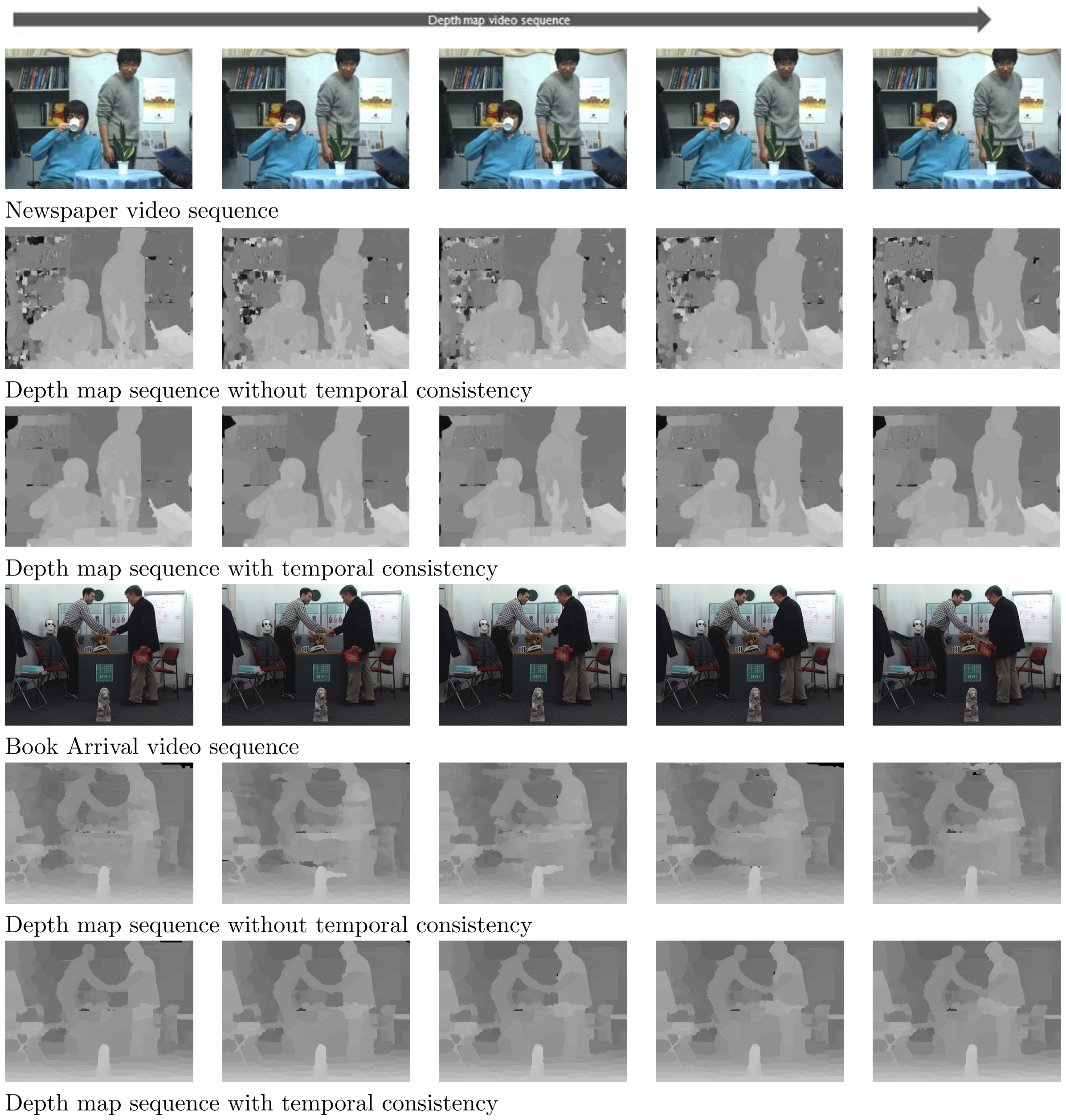
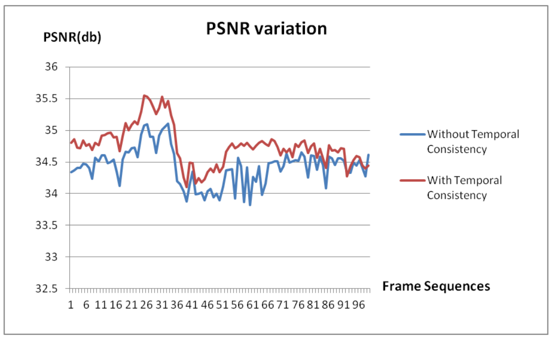

Depth map video sequence
Newspaper video sequence
Depth map sequence without temporal consistency
Depth map sequence with temporal consistency
Book Arrival video sequence
Depth map sequence without temporal consistency
Depth map sequence with temporal consistency

Figure 4.5: Temporal consistency empirical evaluation

Currently there is no standard ground truth video source available for measuring the consistency of disparity map sequences. One solution is measuring the quality of the



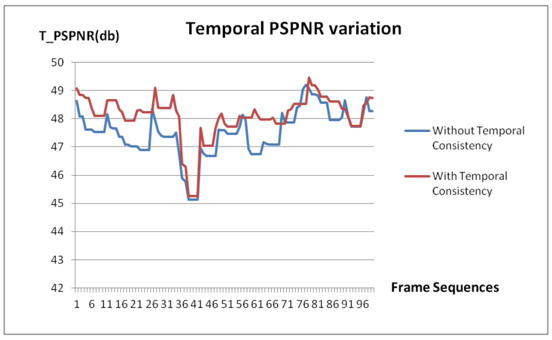


Figure 4.6: PSNR improvement of temporal consistency

synthesized virtual sequences. To measure the performance of temporal consistency function, 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 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.

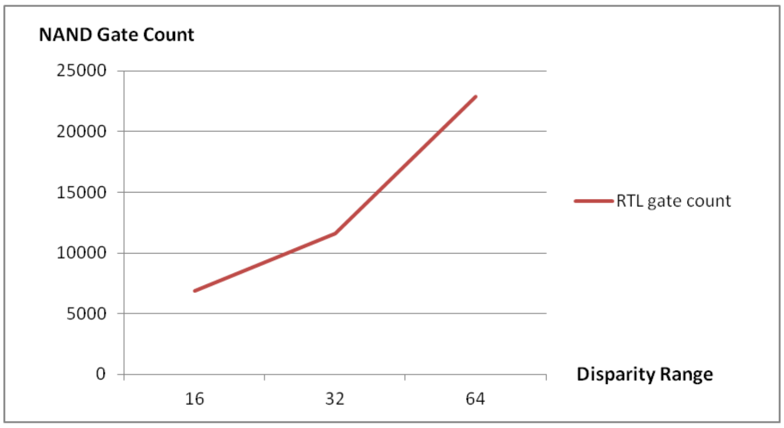


Figure 4.7: RTL circuit gate count synthesis for Dynamic Programming Processor

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. Under the disparity 64 scenario, the BRAM requirement of the Dynamic Programming Processor with path simplification design only takes one eleventh of 2-Port RAM.

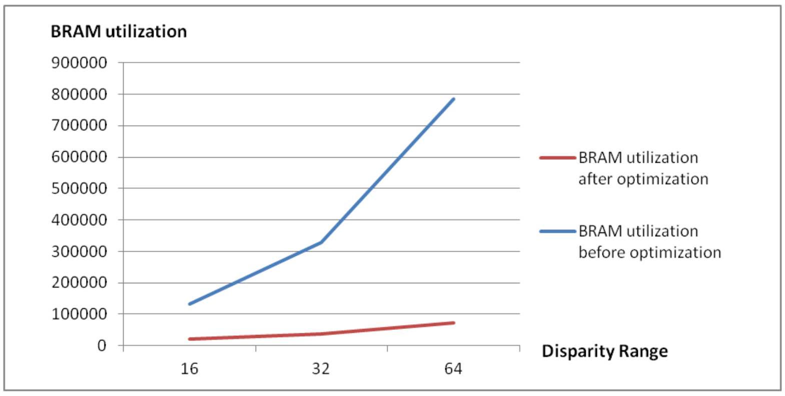


Figure 4.8: On-chip memory utilization estimation of Dynamic Programming Processor

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, run-length coding is a variable length coding algorithm because the compression rate various 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 of 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 by using 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 [53], and Outdoor are chosen. First, we extrat 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.

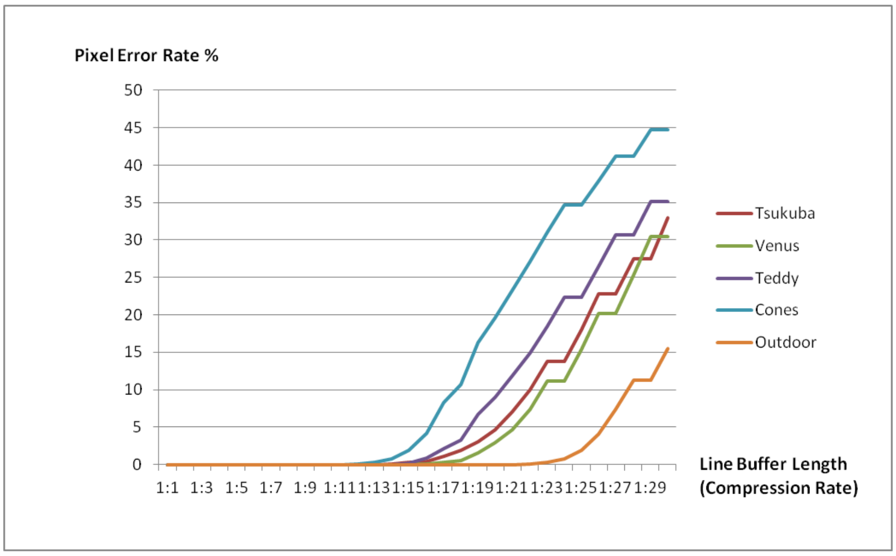


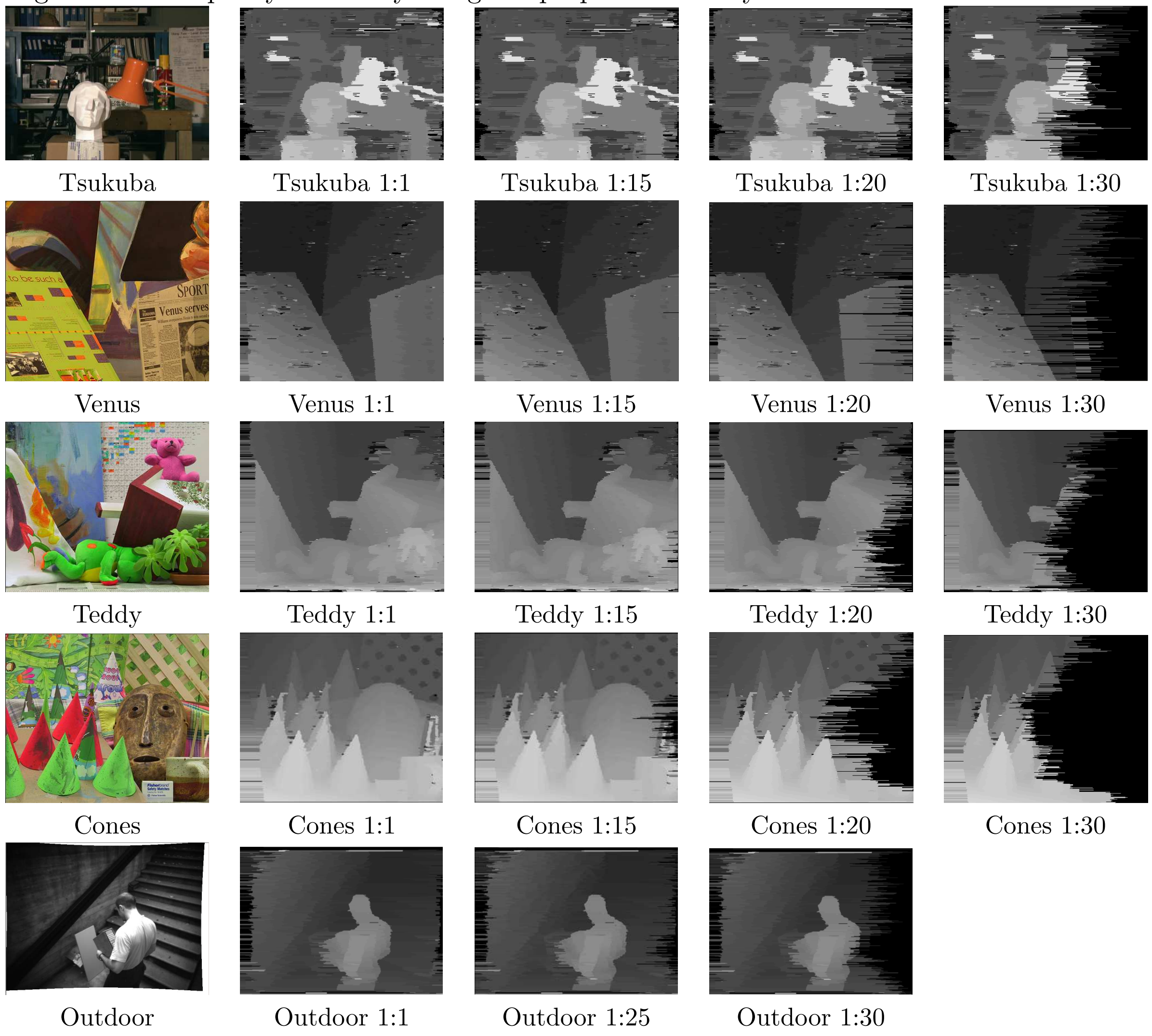
Figure 4.9: Exploration of truncated line buffer length and pixel error rate

Figure 4.10 shows the disparity maps that are derived from different compression rates by using the proposed memory architecture with run-length encoder/decoder. The 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 achive 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.

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.

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 (com-

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



pression 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.

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

4.4.2 Hardware Resource Estimation and Comparison

Finally, we evaluate the hardware cost of the proposed memory architecture for the Vertical Voting Processor. Although the BRAM (line buffer) requirements are reduced by

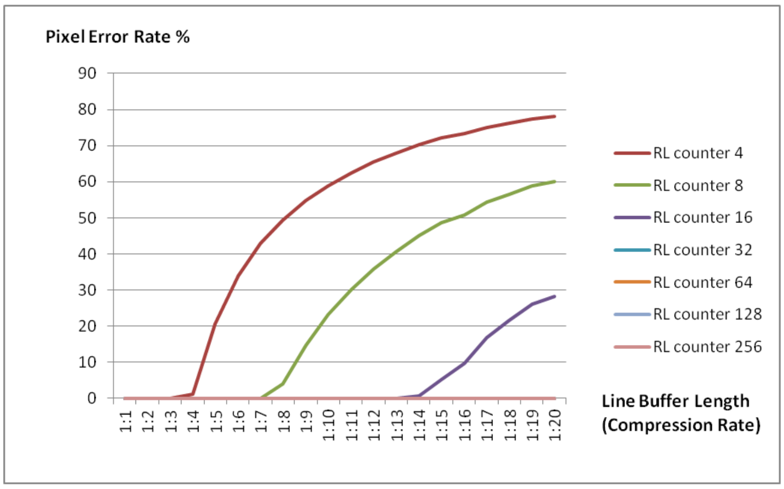


Figure 4.11: Explore the range of run length counter (Outdoor)

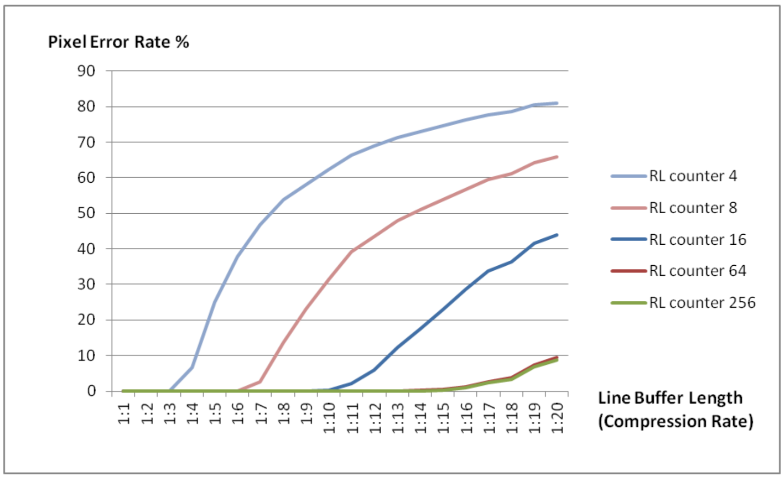


Figure 4.12: Explore the range of run length counter (Cones)

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. The design is measured by Cadence RTL compiler with OSU TSMC 0.25um library. One of the memory architectures uses 30

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							

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). 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 by 8 bits

4.5 Hardware Resource Estimation of Stereo Matching Engine on FPGA

We further estimate the improvement of the resource overhead for the stereo matching engine Table 4.3. On the one hand, the logic gate count of dynamic programming 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

Stereo Matching Engine	*Yig [59]*			*Proposal*		
Processor units (x2)	*LC Comb.*	LC Reg.	*Block Mem (Bit)*	*LC Comb.*	*LC Reg.*	*Block Mem (Bit)*
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 in the forward passing function of Dynamic Programming Processor, in which 72.4MHz frequency is estimated. Therefore, the SoC is capable of handling up to standard XGA (1024x768@ 60FPS 65MHz) video format. It is believed that the new generations of FPGA or ASIC can easily handle higher pixel rate 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. It depicts our design performs a quite good performance in existing SoC solutions.

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

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

Algorithm	*Platform*	*Disparity*	*Frame Rate*	*MDE/s*
Jin et al. 2010[28]	FPGA Virtex-4	64	640X480 @ 230	4521
Our Method	**FPGA Stratix III**	**64**	**1024X768 @ 60**	**3019**
John el al. 2006 [56]	Tyzx DeepSea II	52	512x480 @ 200	2600
Jacobi et al. 2010 [27]	FPGA Virtex II	64	176x144 @ 52	1420
Masrani [37]	FPGAs Transmogri er-4	64	480 x 640 @ 30	330

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

In this chapter, we apply our stereo matching engine to IMEC's 3D TV SoC to achieve the depth intensity adjustment function. The depth adjustment function is based on synthesizing the interpolated virtual view from the stereoscopic cameras/video sources. 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. Hence, we apply the proposed stereo matching engine to IMEC's 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. Then, 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 the IMEC's 3D TV SoC architecture. 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(National Chiao Tung University) and IMEC-Taiwan. And we use a DDR2 High Performance Controller [15] to speed up the development of memory hierarchy.

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

Figure 5.1 and Table 5.1 depicts that the proposed system architecture contains three clock domains: pixel clock, DDR local interface clock, and DDR clock.

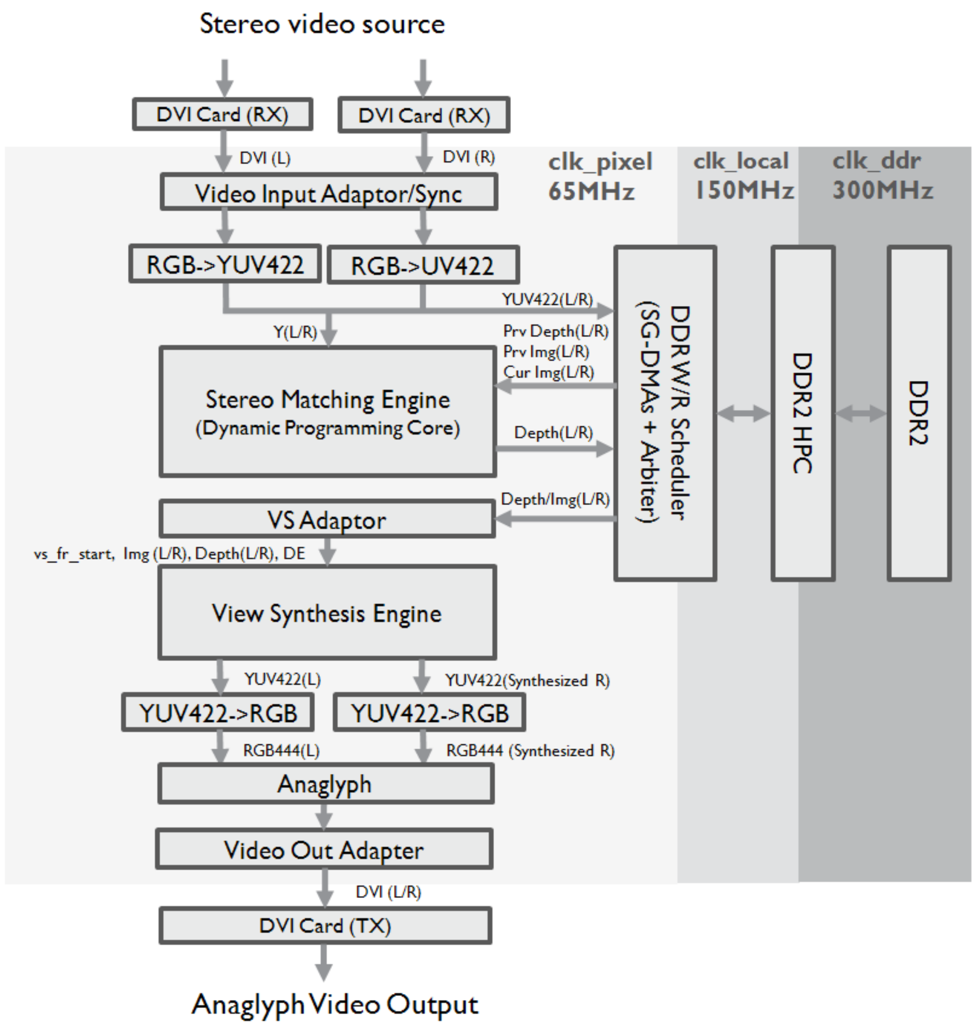


Figure 5.1: Dual DVI receivers scenario

Under 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 under 65MHz. In order to achieve real-time processing, pipeline architecture is applied throughout entired system. 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 important role as a clock domain bridge.

The DDR2 HPC controller [15] 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

Table 5.1: Function definition in SoC

Block	*Function Description*	*Clock Domain*
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 dispairty 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

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 works under 150 MHz and DDR wirks under 300 MHz.

5.1.3 System On-chip Interconnection

Avalon Stream Interface (Avalong ST) protocol [14] is applied to the video processing data path because it can handle 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.

For the interface between stream processing units and DDR controller, a multi-port

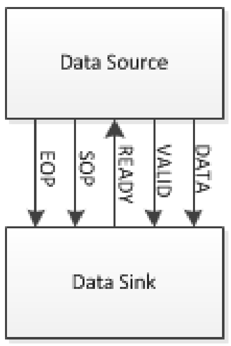


Figure 5.2: Avalon Interface

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 [15].

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 images and depth maps. The view synthesis engine design has 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.

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 synchronize left and right DVI input streams by using FIFOs. The other task of Video Input Adaptor is to filter the incomplete pixel signals of a frame in the intial 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

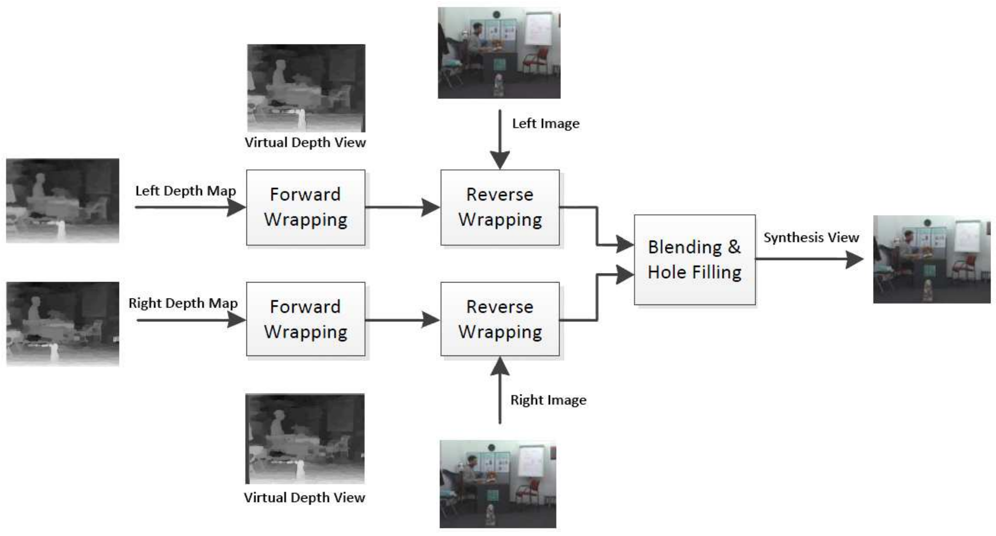


Figure 5.3: High level architecture of View Synthesis Engine

pixel of a frame. Afterward, the following valid pixel data are regarded as valid frame pixels for output.

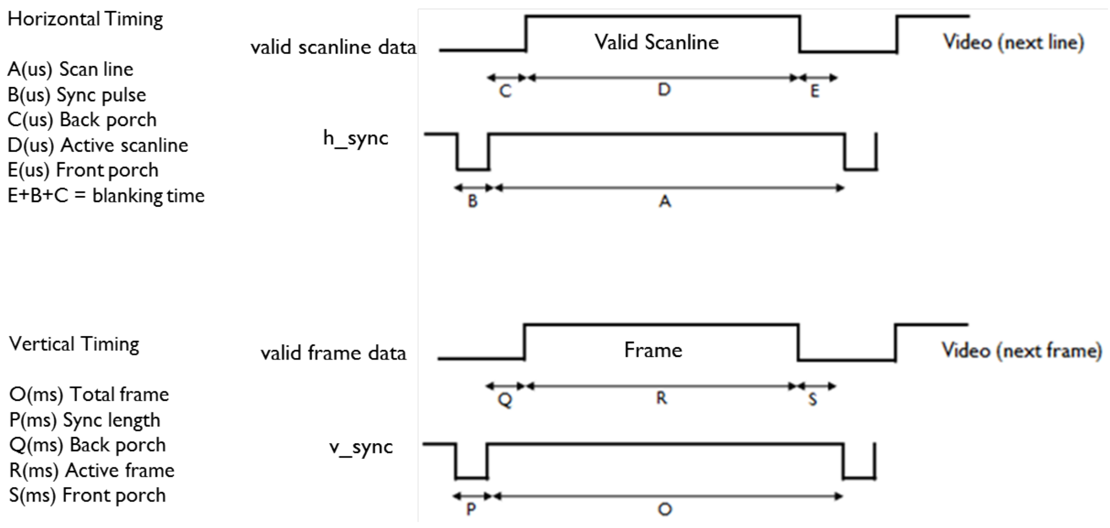


Figure 5.4: Standard VGA signal format

View Synthesis Adaptor (VS Adaptor) is designed to fetch out both stereo disparity map and image sequences (YCbCr422) from DDR Scheduler to 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 sterescopic sequences in standard VGA timing to DVO port for display.

5.4 Color Space Convertor Design and Implementation

YCbCr image format has been used in many popular video applications such as MPEG1-4, H.261-4, and JPEG etc. In the IMEC's 3D TV SoC design, the stereo matching engine takes the luminance information of pixel for depth extraction, and viewpoint synthesis engine also receives the YCbCr format of 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. Then the hardware designs of color space converter will be presented in Sub-section 5.4.3.

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 [50]. 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 variasion. Therefore, the chrominance information, Cr and Cb, could be down-sampled. Hence, YCbCr color format is broadly used in industry.

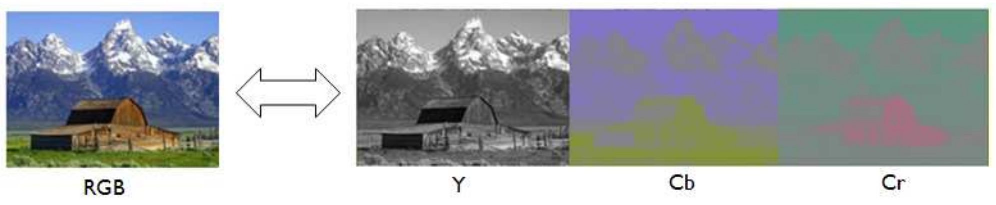


Figure 5.5: Example of RGB and YCbCr Format

YCbCr color space was defined in ITU-R 601 [25] and ITU-R 709 [26] 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 [25] defines its coefficient vector for Standard TV, whereas ITU-R BT.709 [26] 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

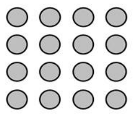
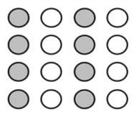
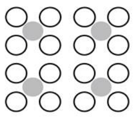
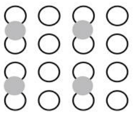
$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{5.1}$$

2. Full Range YCbCr to RGB Format

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

In YCbCr format, A:B:C notation is used to describe the sample factor of Cb and Cr (chrominance) components that 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 the commonly used YCbCr formats in A:B:C notation.

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

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 [42] 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.

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**
$Y = clip(rounding(77 * R + 150 * G + 29 * B) >> 8 + 0)$	$Y = [0 - 255]$
$Cb = clip(rounding(-43 * R - 85 * G + 128 * B) >> 8 + 128)$	$Cb = [0 - 255]$
$Cr = clip(rounding(128 * R - 107 * G - 21 * B) >> 8 + 128)$	$Cr = [0 - 255]$

where the clip function returns 255 when the sum excesses 255; it returns 0 when the sum is negative number. In equation Y, +0 is for the base range; in equation Cb and Cr, +128 is to ensure the final integer values is positive number.

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

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 ïiearest neighborünless 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 422 format and stored in DDR2. In the last stage of the IMEC 3D TV System, the YCbCr 422 format streams will be converted back to RGB format for display.

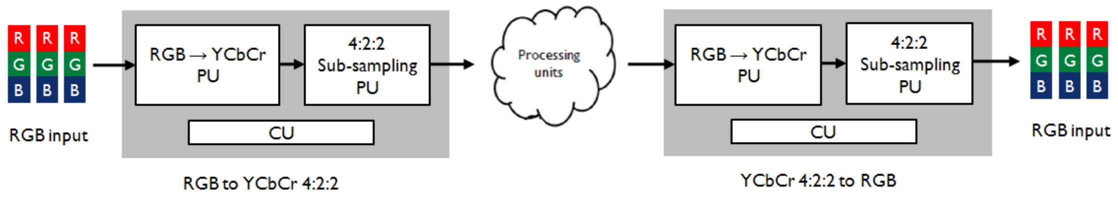


Figure 5.6: Color space conversion flow

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.

HW:RGB to YCbCr Conversion	**Range**
$Y = clip((77 * R + 150 * G + 29 * B + 128) >> 8)$	$Y = [0 - 255]$
$Cb = clip((-43 * R - 85 * G + 128 * B + 32768) >> 8)$	$Cb = [0 - 255]$
$Cr = clip((128 * R - 107 * G - 21 * B + 32768) >> 8)$	$Cr = [0 - 255]$

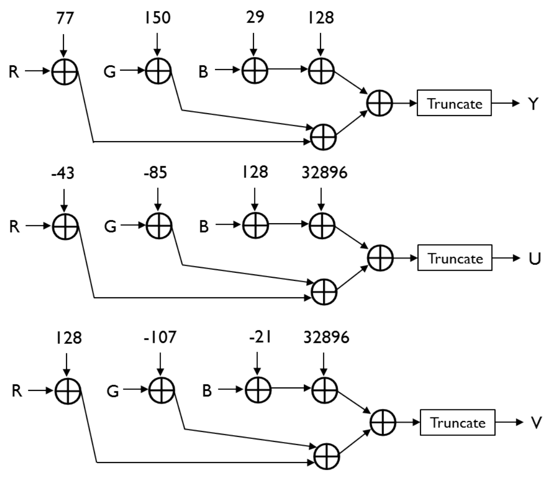


Figure 5.7: RGB to YCbCr Processor Unit

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.

Firstly, constant multiplier can be simply illustrated as Wireshifter-Add/Sub scenario. Since the constant is fixed, it is possible to implement one constant multiplier by multiple Wireshifter-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 Wireshifter-Add/Sub scenario to all constant multipliers in RGB-YCbCr processor unit as Figure 5.8. The main advantage of Wireshift-Add/Sub architecture is the low hardware utilization.

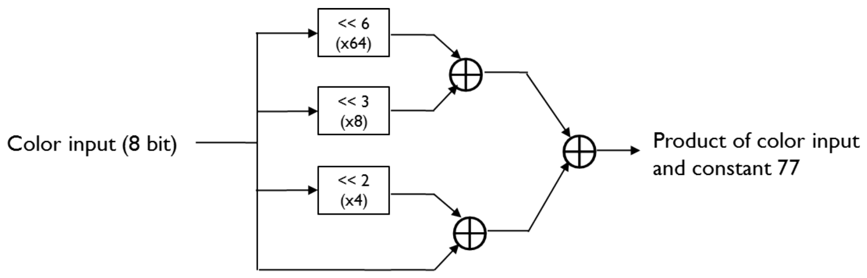


Figure 5.8: Constant multiplier implementation (77) with Wireshifter-Add/Sub architecture

The major problem of the constant multiplier in the first proposed design is lack of flexibility. The convertion 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 the shift-and-add Multiplication method [5] with two look up table [33]. 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.

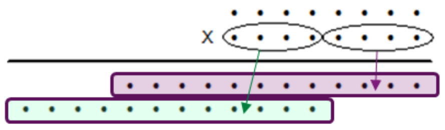
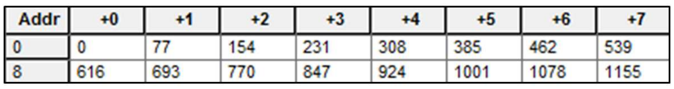


Figure 5.9: upper 4 bit nibble and lower 4 bit nibble multiplication



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

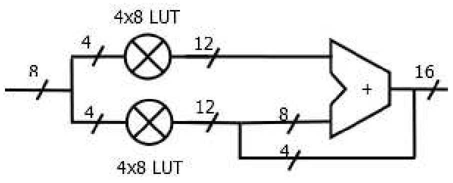


Figure 5.11: 8x8 Constant multiplier with two 4x8 LUTs and Adder

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].

$$Bandwidth = SDRAM\ \ Bus\ \ Width \times 2\ \ Clock\ \ Edges \times Frequency\ \ of\ \ Operation \times Efficiency \tag{5.3}$$

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.

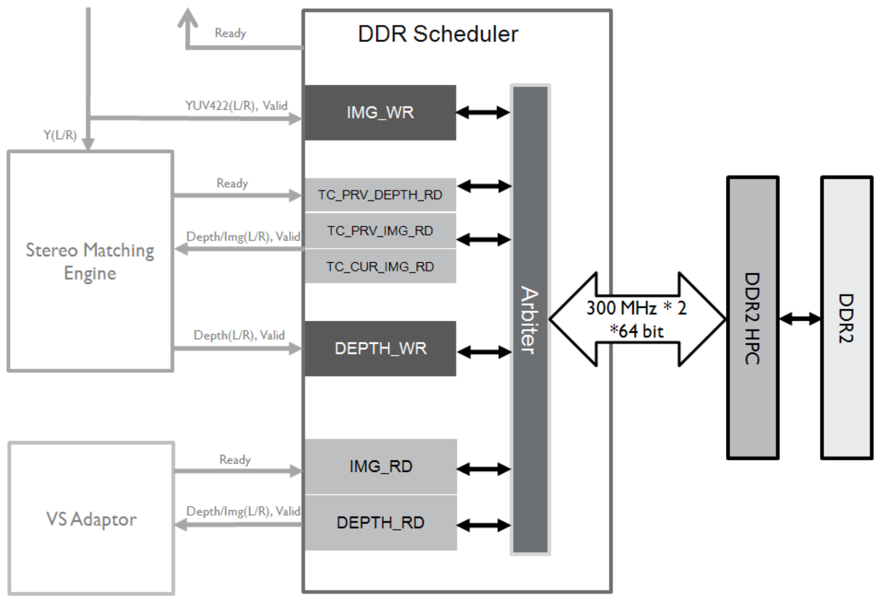


Figure 5.12: Proposed Memory Hierarchy

$$Throughput = Frame\ \ Rate \times Frame\ \ Width \times Frame\ \ Height \times Pixel\ \ Format \times Channel\ \ Number \tag{5.4}$$

Table 5.3: System memory breakdown

Memory Access Components	*Throughput (G bps)*	*Critical Latency*
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	

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

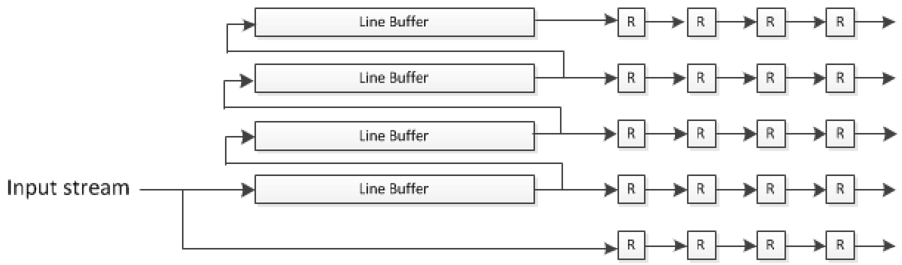


Figure 5.13: 5x5 slide window operation

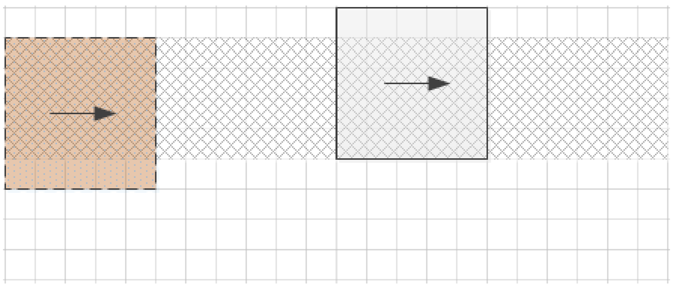


Figure 5.14: Example of data reuse: the weaved texture region represents the data reuse zone in slide window application

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.

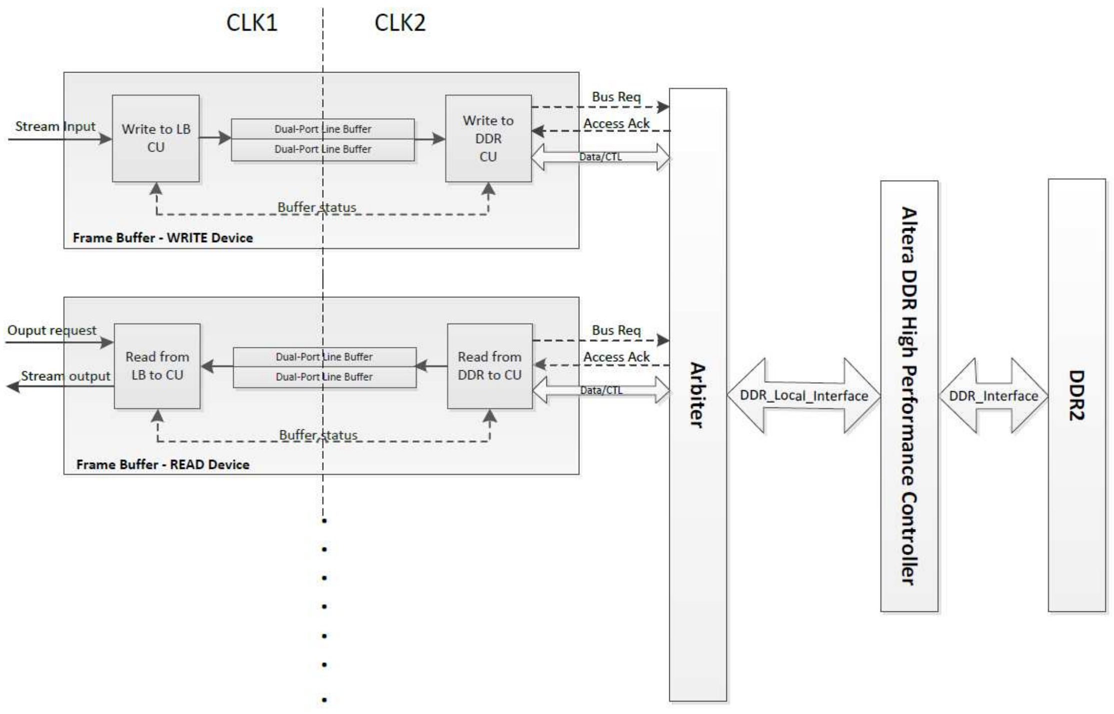


Figure 5.15: Proposed memory hierarchy for frame buffering

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 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.

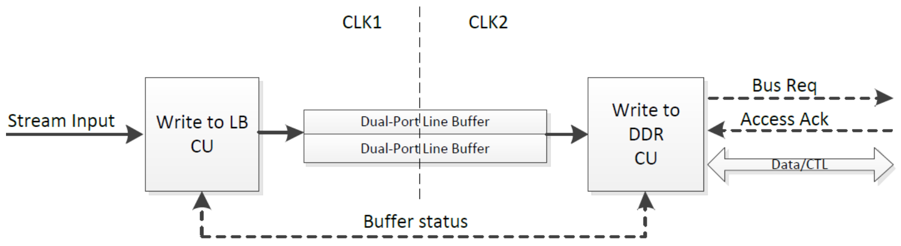


Figure 5.16: SG-DMA architecture for write function

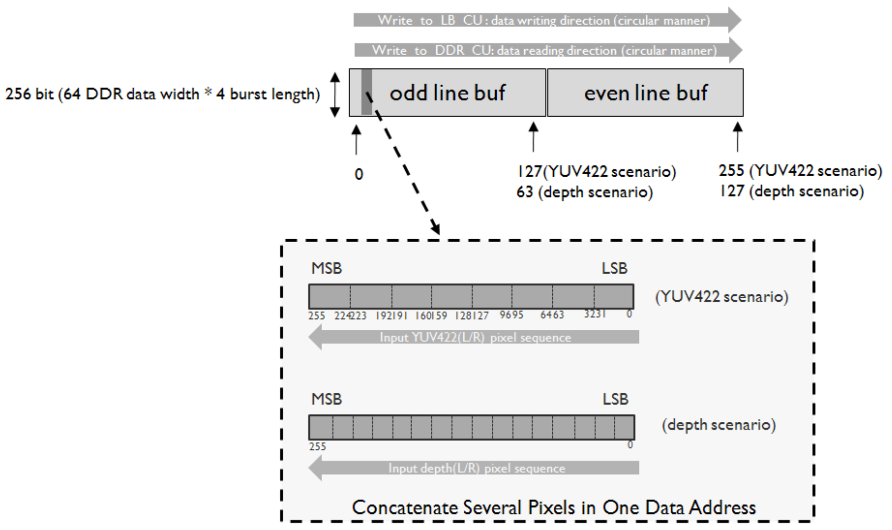

Write to LB CU : data writing direction (circular manner)
Write to DDR CU: data reading direction (circular manner)
256 bit (64 DDR data width * 4 burst length)
odd line buf
even line buf
0
127(YUV422 scenario)
63 (depth scenario)
255 (YUV422 scenario)
127 (depth scenario)
MSB
LSB
(YUV422 scenario)
255 224 223 192 191 160 159 128 127 96 95 64 63 32 31 0
Input YUV422(L/R) pixel sequence
MSB
LSB
(depth scenario)
255
0
Input depth(L/R) pixel sequence
Concatenate Several Pixels in One Data Address

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.

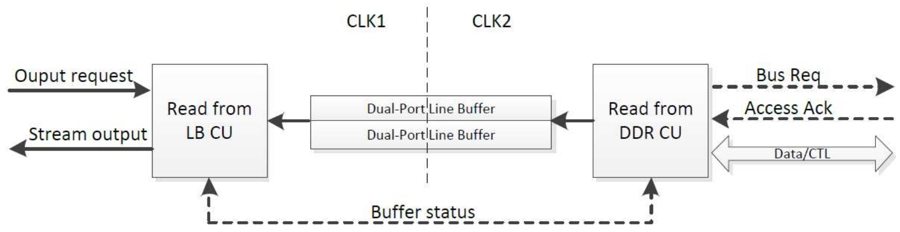


Figure 5.18: SG-DMA architecture for read function

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 [34]. 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 prefferable 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. [34] In the proposed Arbiter design, both priority and round-robin policies are implemented in RTL code.

[**Memory Hierarchy Interconnection**]

1. Interface between SG-DMAs 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.

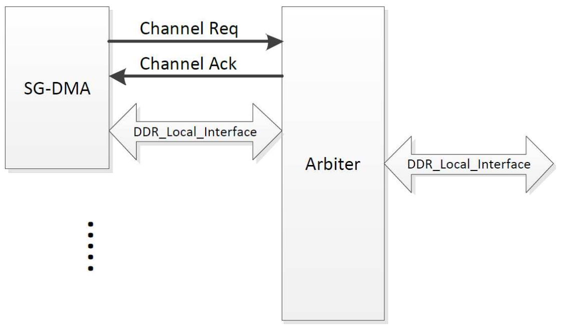


Figure 5.19: The interface between SG-DMA and Arbiter

   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.

   (a) SG-DMA asserts request signal for the access authority of DDR_local_Interface channel

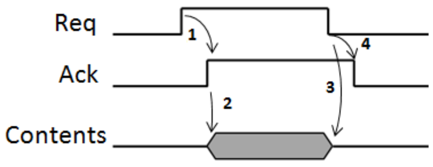


Figure 5.20: Handshaking protocol between SG-DMA and Arbiter

(b) When the channel is available, arbiter assigns memory access acknowledgment to the SG-DMA request device. After the SG-DMA possesses the channel 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 [14] 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.

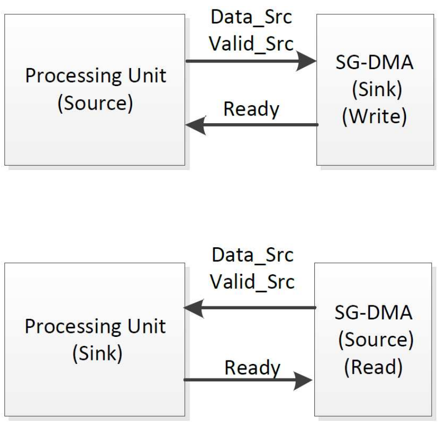


Figure 5.21: Avalon stream interface between processing unit and SG-DMA

[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 theose 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.

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

   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.

$$\begin{aligned} &Image\ \ Frame\ \ Buffer\ \ Address\ \ Range = 2(current\ \ and\ \ previous\ \ frame) \\ &\times 2(stereo\ \ left\ \ and\ \ right\ \ channels) \times Frame\ \ WidthxFrame \\ &Heightx16bit(YCbCr422\ \ Pixel\ \ Format)/64(DDR2\ \ data\ \ width) \end{aligned} \tag{5.6}$$

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.

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

   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.

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

Figure 5.22 is an example shows the address generating patterns according to the storage data allocation. 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.

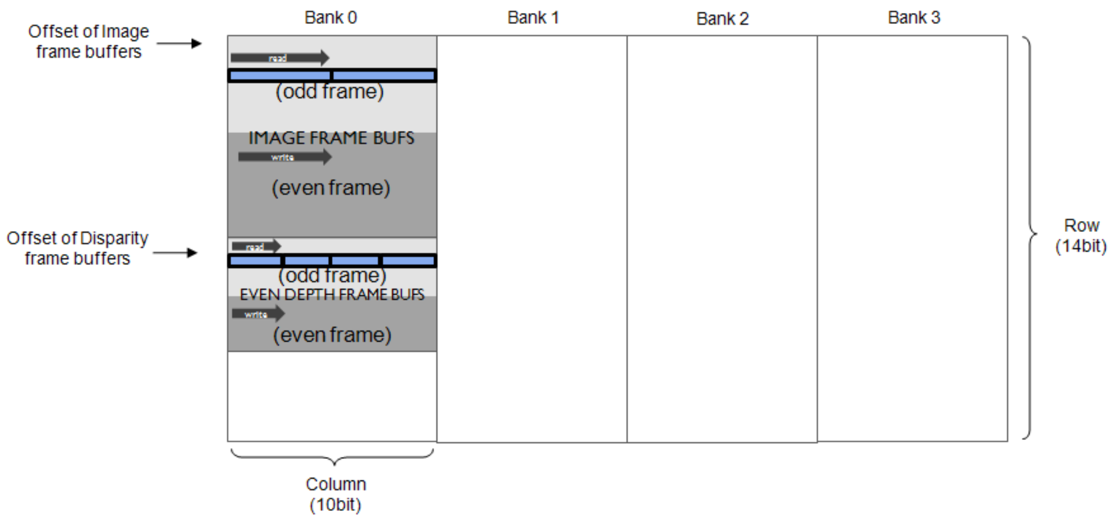


Figure 5.22: Example of address generation pattern

IMEC 3D TV SoC Evaluation and Experimental Result 6

In Section 6.1, we firstly evaluate the color space converter designs. Then we will test the IMEC 3D TV system SoC architecture at the system level. Section 6.2 will measure the quality of interpolated virtual view by using PSNR model. In addition, we analyze the temporal quality of the interpolated video sequences by TPSNR model. Then the component hardware costs are estimated in Section 6.3. Finally, the real-time performance of the system is estimated in Section 6.4. This work ensure the critical path of the entire SoC system and the bandwidth of off-chip memory achieve the real-time criterias.

6.1 Color Space Converter Design Evaluation

This part evaluates the color space converter designs according to their quality, hardware resource, and latency.

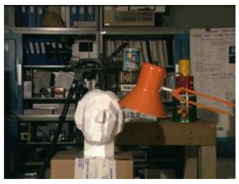
6.1.1 Quality Evaluation

This sub-section evaluates the quality degradation from 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 by using PSNR model. The PSNR model is measured from comparing original input image and inverted output image. The higher PSNR represents the less loss of color information during conversions. In general, the PSNR result 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 [53]) are the test sets from stereo matching society. Of course theose pictures are not the only test candidates. 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 after conversion is hardly noticeable. It also shows that the wider range of pixel resolution (0-255) achieves slightly higher accuracy than the lower range of pixel (16-235-240). It is found that chroma sampling (YCbCr 4:2:2 and YCbCr 4:2:0) conversion causes

Table 6.1: Test sets

				
Lena	Tsukuba	Venus	Teddy	Cones

tremendous quality degradation to test images. The more chroma samples that are excluded during sub sampling, the more color information is lost.

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 approach is much complicated for hardware implementation, fixed point approach is preferred option. It scales up the floating point coefficients to integers in color space transform. Table 6.3 evaluates several scaling up resolution from four bit (256) to ten bit (1024). As a result, the lower scaling up resolution delivers a lower quality of output image. From the observations, the result of PSNR by using 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 fixed point 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 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 6.3: 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

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 Evalution

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 2LUTs-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 2LUTs-Adder structure takes much less hardware resource.

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

Adder approach	*Full LUT Size*	*2 LUT+Adder*	*LUT Size* Improvement
Gate Count 4bit coefficient	256 entries x(8+4)=3072	2x16 entries x(4+4)=256	12
Gate Count 6bit coefficient	256 entries x(8+6)=3584	2x16 entries x(4+6)=320	11.2
Gate Count 8bit coefficient	256 entries x(8+8)=4096	2x16 entries x(4+8)=384	10.67
Gate Count 10bit coefficient	256 entries x(8+10)=4608	2x16 entries x(4+10)=448	10.29
Gate Count 12bit coefficient	256 entries x(8+12)=5120	2x16 entries x(4+12)=512	10

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 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. 2LUTs-Adder constant multiplier solution takes a little bit more combinational resource than Full LUT approach but only requires one-tenth of the block ROM. WireShift-AddSub approach takes the minimum hardware resource because it is composed of wire-shift and adders. Obviously, the wireshift-AddSub approach provides the lowest cost solution for implementing the constant multiplier of color space converter. And the 2 LUTs-Adder approach not only consumes low Block ROM resources but also keeps the flexibility to different convertion vectors.

Table 6.6: Hardware utilization analysis of RGB to YCbCr converter

Constant Multiplier approach	*Full LUT approach*	*2 LUTs-Adder*	*Wireshift-Add/Sub approach*
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 2LUTs-Adder and Wireshift-AddSub approaches is further evaluated by the synthesizing tool, Sinplify Premier, with stratix III library. The result shows the RGB-YCbCr converter with 2LUTs-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 Wireshift-AddSub constant multiplier, the propagation delay is around 3.4ns. The clock rate can achieve 297 MHz.

6.2 Quality Evaluation

To assess the output video quality of our 3D TV system, the synthesized virtual video is compared with the true interpolated video by using PSNR model [49] [12]. 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) [47] and View Synthesis Reference Software (VSRS) [46]. Both DERS and VSRS were contributed 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)

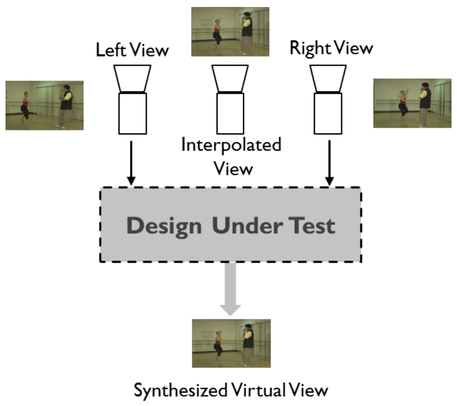


Figure 6.1: Interpolated video evaluation structure for our system

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.

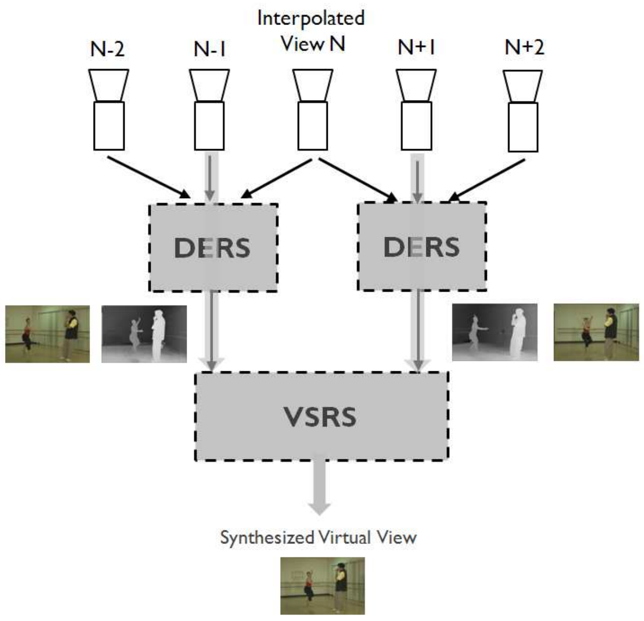


Figure 6.2: Interpolated video evaluation structure for DERS+VSRS

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

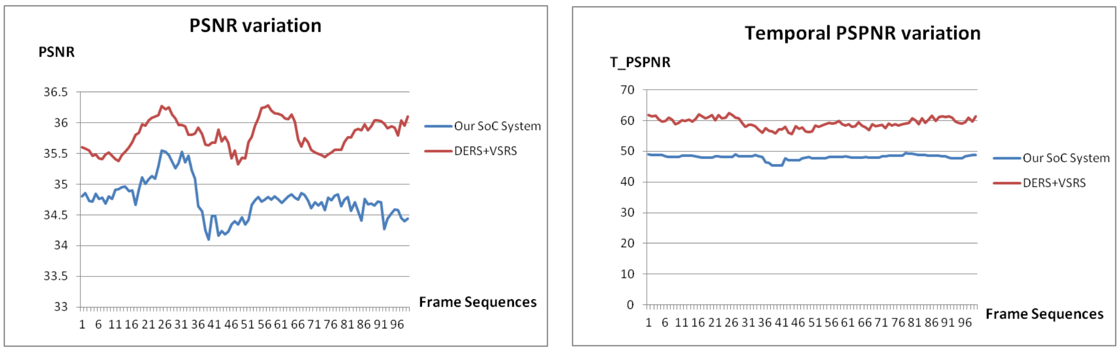


Figure 6.3: Quality evaluation for Book Arrival

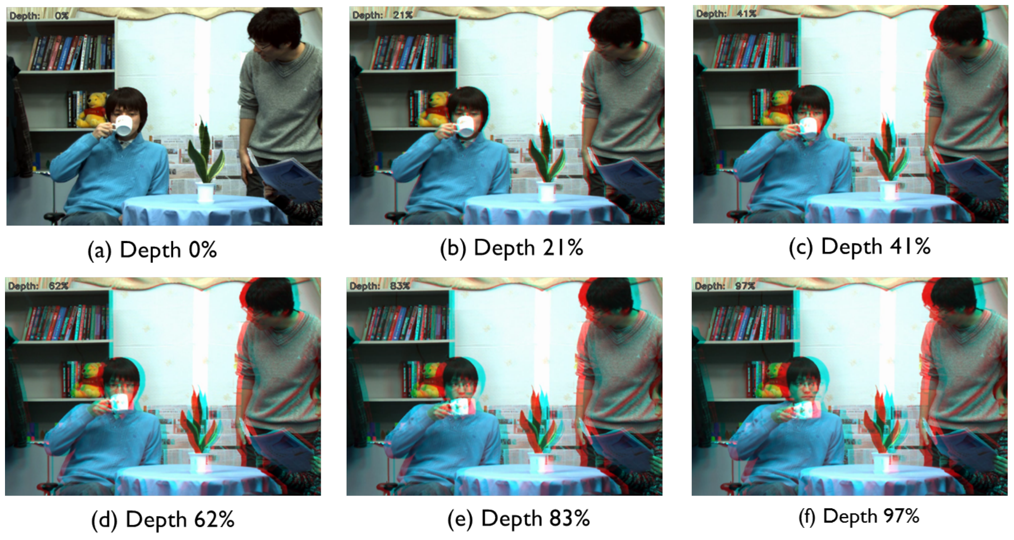


Figure 6.4: Anaglyph outputs for different depth intensity

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.

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.

Table 6.7: Hardware resource utilization summary on EP3SL150 FPGA

Processor Units	*LC Comb.*	*LC Reg.*	*Block Mem Bits*
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 Enginse	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%

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 is an example that depicts the long latencies which includes regions (1), (2), (3) and (4).

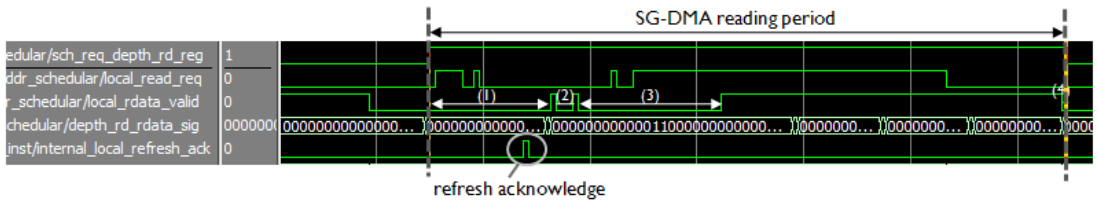


Figure 6.5: Example of latencies during burst reading

To ensure no critical latency constrains of SG-DMA devices are violated, a simulation environment is built as Figure 6.6. The memory and controller models [15] 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 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

Burst Length (Line Buffer Length)		*Command number(cycle)*	*Commands + Worst case latencies(cycle)*	*Command Efficiency(%)*	*Critical Latency(cycle)*	*Latency Violation*
Image	*Depth*					
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

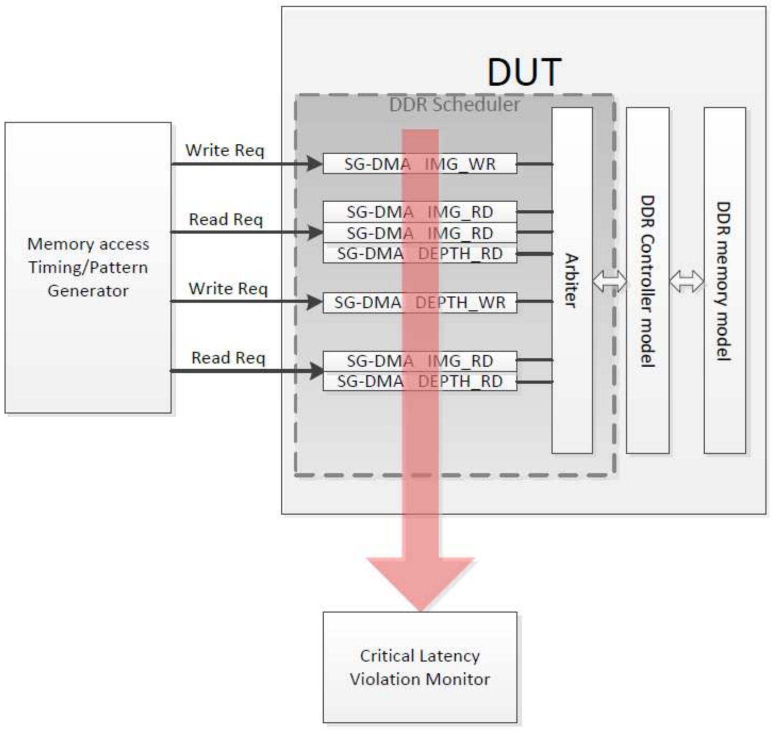


Figure 6.6: Memory hierarchy evaluation environment

Conclusion and Future Works 7

This chapter summarizes entire thesis work. In Section 7.1, the proposals and experimental results are concluded. Then we review each chapter and present the contributions in Section 7.2. Finally, the future works and relative applications are suggested in Section 7.3.

7.1 Conclusion

Two improvements for stereo matching algorithm have been introduced and proofed in this thesis. In order to generate high quality disparity map, the dynamic programming algorithm is used. We select Potts model as the smoothness function because it performs outstandingly in the disparity discontinuous regions and requires less computational complexity. From the test result of Middlebury's benchmark, our stereo matching algorithm achieves a 6.6% average pixel error rate. In order reinforce the temporal consistency of disparity sequences, we adjust the raw matching cost based on the history disparity map result and pixel-based motion information. The final disparity map sequences is improved addressing to the flickering problem.

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. The back track path data is simplified by only recording the encoded path information(decision). Furthermore, we apply 2-Port BRAM with a sophisticate memory address generation pattern to reduce the on-chip memory requirement to half, compared to the conventional ping-pong memory architecture. The on-chip memory requirement of dynamic programming processor is reduced by 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 coder/decoder for the vertical voting processor. Finally, the resource consumption of our 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 IMEC's 3D TV SoC with EP3SL150 FPGA, which allows the 3D intensity of stereoscopic contents become 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 solutions, 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 our stereo matching engine and NCTU's view synthesis engine. 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 video 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 (Altera's IP). 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, Wireshift-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 entired 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 to single scanline. There is a lack of 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 relativelly 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, both critical latency-aware arbiter and high efficient off-chip memory controller will be strongly required to achieve the 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's 3D TV SoC can 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 near future soon to come.

Bibliography

[1] Altera, *The efficiency of the ddr & ddr2 sdram controller compiler*, May 2011.

[2] Pedram Azad, Tilo Gockel, and R'diger Dillmann, *Computer vision: Principles and practice*, elektor, 2008.

[3] C. Banz, H. Blume, and P. Pirsch, *Real-time semi-global matching disparity estimation on the gpu*, ICCV Workshops, IEEE (2011).

[4] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, *Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation*, Embedded Computer Systems (SAMOS) (2010).

[5] Z. F. Baruch, *Structure of computer systems, page 62, isbn 973-8335-44-2*, U. T. PRES, Cluj-Napoca, 2002.

[6] Michael Bleyer, Margrit Gelautz, Carsten Rother, and Christoph Rhemann, *A stereo approach that handles the matting problem via image warping.*

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2001).

[8] G. Bradsky and A. Kaehler, *Learning opencv*, O'Reilly, 2008.

[9] N.Y.-C. Chang, Tsung-Hsien Tsai, Bo-Hsiung Hsu, Yi-Chun Chen, and Tian-Sheuan Chang, *Algorithm and architecture of disparity estimation with mini-census adaptive support weight*, Circuits and Systems for Video Technology (2010).

[10] 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).

[11] 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).

[12] Wei Chen, Jerome Fournier, Marcus Barkowsky, and Patrick Le Callet, *New requirements of subjective video quality assessment methodologies for 3dtv*, VPQM (2010).

[13] 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).

[14] Altera Coperation, *Avalon interface specifications*, May 2005.

[15] Altera Corp., *Ddr and ddr2 sdram high-performance controller user guide*, March 2009.

[16] 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).

[17] N. A. Dodgson, *Autostereo displays: 3d without glasses*, EID '97.

[18] Zeng fu Wang and Zhi gang Zheng, *A region based stereo matching algorithm using cooperative optimization*, CVPR 2008.

[19] Stefan Gehrig, Felix Eberli, and Thomas Meyer, *A real-time low-power stereo vision engine using semi-global matching*, ICVS (2009).

[20] Minglun Gong and Yee-Hong Yang, *Real-time stereo matching using orthogonal reliability-based dynamic programming*, Image Processing, IEEE (2007).

[21] Hirschmuller H. and Scharstein D., *Evaluation of stereo matching costs on images with radiometric differences*, Pattern Analysis and Machine Intelligence, IEEE Transactions (2009).

[22] Richard Hartley and Andrew Zisserman, *Multiple view geometry*, CVPR, June 1999.

[23] 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).

[24] Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann, *Local stereo matching using geodesic support*, IEEE, ICIP (2009).

[25] ITU, *Recommendation bt.601*, 2011.

[26] ______, *Recommendation bt.709*, 2011.

[27] Ricardo P. Jacobi, Renato B. Cardoso, and Geovany A. Borges, *A reconfigurable matrix for stereo vision processing*, 20th International Parallel and Distributed Processing Symposium IPDPS (2006).

[28] 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).

[29] Kuk jin Yoon Student Member In So Kweon, *Adaptive support-weight approach for correspondence search*, IEEE Trans. PAMI (2006).

[30] 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.

[31] Jae Chul Kim, Kyoung Mu Lee, Byoung Tae Choi, and Sang Uk Lee, *A dense stereo matching using two-pass dynamic programming with generalized ground control points*, CVPR 2005. IEEE.

[32] A. Klaus, M. Sormann, and K. Karner, *Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure*, ICPR 2006.

[33] Steve Knapp, *Constant-coefficient multipliers save fpga space and time*, EDA (1998).

[34] 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).

[35] Carlos Leung and Brian C. Lovell, *An energy minimisation approach to stereo-temporal dense reconstruction*, Proc. International Conference on Pattern Recognition (2004).

[36] 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).

[37] 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).

[38] Stefano Mattoccia, *Stereo vision:algorithms and applications*, `\www.vision.deis.unibo.it/smatt`, May 2011.

[39] Mark Nelson, *The data compression book*, M&T Books, 1992.

[40] 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.

[41] R.I.Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2000.

[42] Gary Sullivan and Stephen Estrop, *Video rendering with 8-bit yuv formats*, Microsoft Digital Media Division (2002).

[43] Jian Sun, Yin Li, Sing Bing, and Kang Heung yeung Shum, *Symmetric stereo matching for occlusion handling*, CVPR 2005.

[44] Jian Sun, Heung yeung Shum, and Nan ning Zheng, *Stereo matching using belief propagation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2003).

[45] Yuichi Taguchi, Bennett Wilburn, and C. Lawrence Zitnick, *Stereo reconstruction with mixed pixels using adaptive over-segmentation*, CVPR 2008.

[46] 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).

[47] 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).

[48] M. Tanimoto, M.P. Tehrani, T. Fujii, and T. Yendo, *Free-viewpoint tv*, Springer.

[49] A. Tikanmaki, A. Gotchev, A. Smolic, and K. Miller, *Quality assessment of 3d video in rate allocation experiments*, ISCE (2008).

[50] M. Tkalcic and J. Tasic., *Color spaces perceptual, historical and applications background*, EUROCON, IEEE **1** (2003), 304–308.

[51] T.Kanade, *Development of a video-rate stereo machine. in image understanding workshop, pages 549-557*, Morgan Kaufman, 1994.

[52] Middlebury university, *Middlebury stereo evaluation tool*, `http://vision.middlebury.edu/stereo/eval/`.

[53] ______, *Middlebury stereo datasets*, `http://vision.middlebury.edu/stereo/data/`, 2001.

[54] O Veksler, *Stereo correspondence by dynamic programming on a tree*, CVPR 2005. IEEE.

[55] E. Trucco ; A. Verri, *Introductory techniques for 3d computer vision*, Prentice Hall, 1998.

[56] 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).

[57] O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon, *Global stereo reconstruction under second order smoothness priors*, CVPR (2008).

[58] Qingxiong Yang, Ruigang Yang, J. Davis, and D. Nister, *Spatial-depth super resolution for range images*, CVPR 2007.

[59] Guanyu Yi, *High-quality real-time hd video stereo matching on fpga*, Master's thesis, TU Delft, 2011.

[60] 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).

[61] 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).

[62] Lu Zhang, *Design and implementation of real-time high-definition stereo matching on soc on fpga*, Master's thesis, TU Delft, 2010.

[63] 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.

[64] Yin Zhao and Lu Yu, *Pspnr tool 2.0*, MPEG2009/M16890 (2009).

[65] Svitlana Zinger, Luat Do, Daniel Ruijters, and Peter H. N., *iglance: Interactive free viewpoint for 3d tv*, 17th International Conference on Computer Graphic, Visualization and Computer Vision.