MEMORY EFFICIENT ARCHITECTURE FOR BELIEF PROPAGATION BASED DISPARITY ESTIMATION

Sih-Sian Wu[⋆]*, Hong-Hui Chen*[⋆]*, Chen-Han Tsai*[⋆]*, Liang-Gee Chen*[†]

DSP/IC Design Lab, National Taiwan University, Taiwan

[⋆]{benwu, sernc, phenom}@video.ee.ntu.edu.tw [†]lgchen@video.ee.ntu.edu.tw

ABSTRACT

This paper introduces a memory efficient architecture for belief propagation based disparity estimation. To find the bottleneck of the memory, a lifetime analysis of the exchanged message is presented. The analysis leads to architecture which can take advantage of the data characteristics resulting in memory reduction, computing resource and memory access. In the experimental result, the proposed architecture gains 20% speed up in software simulation compared to non-optimized counterpart. In hardware implementation, more than 42% area is reduced by this architecture design without affecting the performance. With further design effort 61.8% area reduction is achieved.

***Index Terms*—** BP-M, tile-based BP, memory efficient, stereo matching, disparity estimation

1. INTRODUCTION

While the requirement of the quality of depth map grows with higher resolution display, nowadays Full-HD 1920×1080 is the basic requirement and 4K2K is emerging in future display market. Moreover, 3D related application becomes more and more popular. How to get a higher video and disparity resolution with reasonable depth granularity is the emerging problem. Those stereo matching algorithms, loopy belief propagation (BP) [1], graph cut [2] and tree reweighted message passing [3] are discussed widely due to their promising results. Among them, BP algorithms is regular and hardware-friendly which is suitable for VLSI implementation. However, the serious memory problem is the critical bottleneck for implementation. Block-based BP[4] and tile-based BP[5] are proposed which can reduce memory significant. The comparison of the memory requirement is shown in Table 1. In this work, we propose a memory efficient architecture to further reduce the on-chip memory.

The rest of the paper is organized as follows: In Section 2, we address the problem we are facing and briefly introduce several previous works. The discussion of the proposed architecture is shown in Section 3. The experimental result and conclusion are presented in Section 4 and 5, respectively.

Table 1. The memory comparison between BP-M and tile-based design during different speculations. The size of tile is 32×32 with label number of disparity 32. The data cost value is 8 bit and the message is 10 bit in length.

Spec.	Memory	BP-M[1]	Tile-based BP[2]
720 HD (1280×7200)	Cost	56.25 MB	64 KB
	Message	281.25 MB	320 KB
Full HD (1920×1080	Cost	126.5625 MB	64 KB
	Message	281.25 MB	320 KB
4K2K (3840×2160)	Cost	506.25 MB	64 KB
	Message	2531.25 MB	320 KB

[1] The bit number for BP-M is derived from image size × number of labels × length of value.

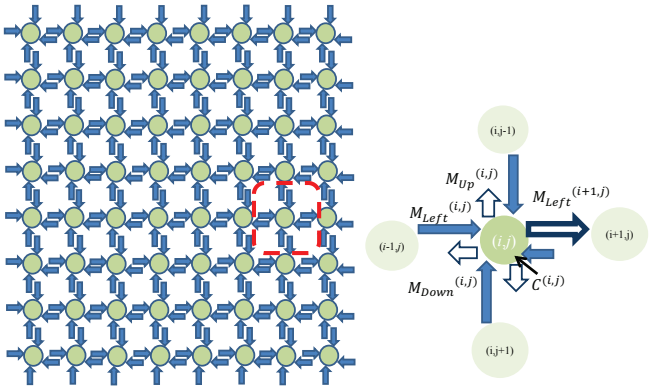
[2] The bit number for tile-based BP is derived from tile size × number of labels × length of value.

2. PREVIOUS WORK

BP-based disparity estimated algorithm, whose abbreviation is BP-M (BP media)[6], is used widely in depth estimation method which provides a solution of global energy minimization for stereo matching algorithm. The concept of BP-M is shown in Fig. 1, each node comes with four message from four directions and its data cost. Because of the data dependency, the requirement of storage is significant. Eq.(1) illustrates the message passing from node (i, j) to node (i + 1, j). Where $M_{Left}(i,j)$ represents the message from left for node (i,j) and $C(i,j)$ presents the data cost of the specific pixel. As Eq.(2), after message updating process the $M_{Left}(i+1,j)$ is the input for next node. Although each row can process simultaneously while horizontal processing and vice versa, there exits high data dependency between adjacent messages.

$$M^*_{Left}(i+1,j) = M_{Left}(i,j) + M_{Up}(i,j) + M_{Down}(i,j) + C(i,j) \quad (1)$$

$$M_{Left}(i+1,j) = MessageUpdate(M^*_{Left}(i+1,j)) \quad (2)$$



(a) (b)

Fig. 1. (a)Framework of BP-M. (b)Detailed message flow of a single node.

Table 2. The area partition of tile-based BP with tile size 32 X 32, 64 disparity label, 8-bit cost value and 10-bit message.

	On-chip Memory	Non-memory
Gate count	1954966	128818
Percentage	93.8%	6.2%

In the BP-based disparity estimation, each pixel will receive four messages from up, down, left and right neighboring nodes respectively. In the traditional BP-M system, the four direction exchanged messages for every label of each node in the tile are stored in individual memory space which is the immense requirement.

Liang *et al.*[5] proposed tile-based BP which is derived from the concept of block-based BP[4] with boundary message sharing. Leverage the hardware-friendly of tile-based BP several teams targeted implementation the proposed system on GPU or FPGA platform[7]. VLSI implementation is also proposed[8]. Only the cost value and messages of the processing tile are stored into on-chip SRAM. With acceptable data size, the chip implementation is possible, however, memory size still dominates the size of design.

3. PROPOSED SYSTEM

As Table 2 shows, on-chip memory occupies more than 90% area of the non-optimized system. A new architecture is proposed to solve this problem. To thoroughly understand the data flow, we start from lifetime analysis of exchanged messages.

3.1. Lifetime analysis of message

According to Eq.(1), we know that cost value do not change during processing, which means exchanging message are our target to take care. We illustrate the 4×4 frame as example. The timing slot of message lifetime is shown as Fig.2. In the example, four BP processing units are involved and each direction processing of each pixel can be done in one cycle. That is, there are four updated messages being generated per cycle. The system requires four cycles to finish one direction passing and sixteen cycles to complete one iteration. We adopt the definition in tile-based BP[5], the BP processing follows the order as to right, left, down and up. In Fig.2, we use different colors to present message from different iterations.

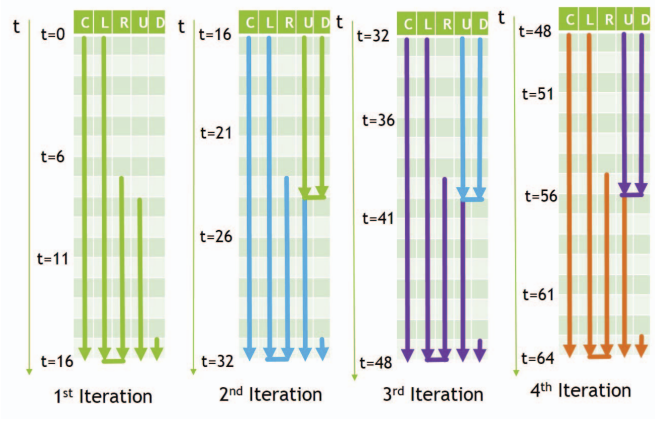


Fig. 2. Message lifetime of pixel (1,1).

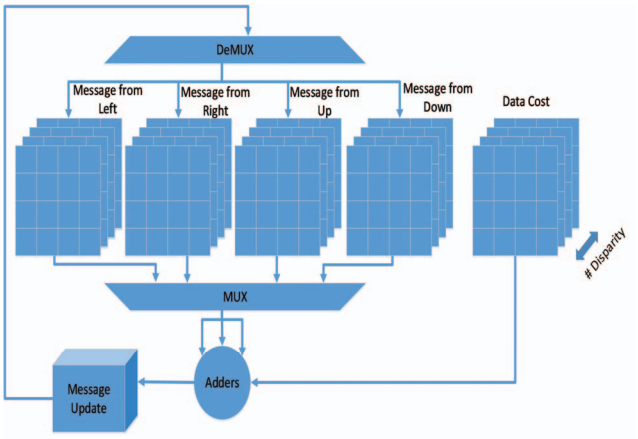


Fig. 3. Original Architecture for tile-based BP.

We summarize our observation as two aspects. First, the message still occupies the memory since the updated message of particular direction has not generated yet, even the message is never called. Both BP-M and tiled-based architecture store the message into its corresponding directly, only update the old message while new one is ready. Second, messages from vertical direction terminate in the same time while finishing horizontal processing and verse visa. We follow aforementioned observation to re-design the tile-based BP architecture.

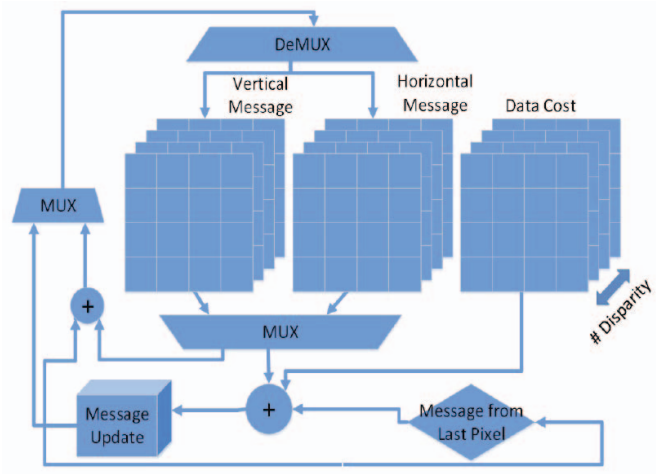


Fig. 4. Proposed 1 architecture for tile-based BP.

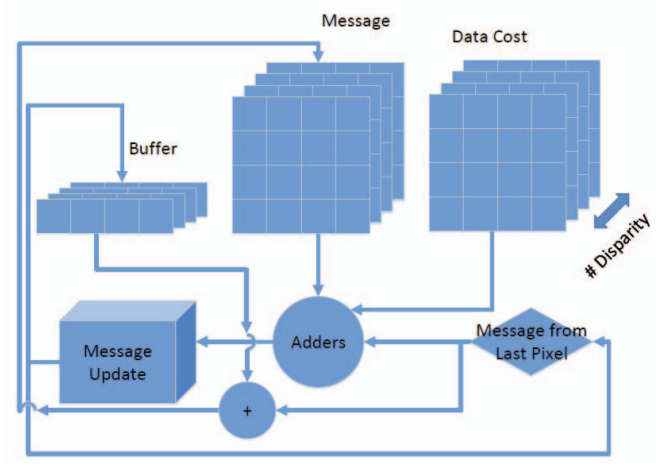


Fig. 5. Proposed 2 architecture for tile-based BP.

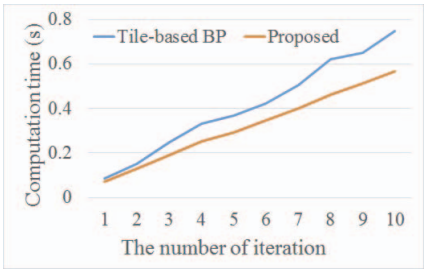


Fig. 6. The performance comparison between traditional BP and the proposed one.

3.2. Proposed architecture

The original tile-based BP is shown in Fig.3. First of all, we group messages into vertical and horizontal messages as Eq.(3).

$$\begin{cases} \text{vertical}, & \text{message from up and down} \\ \text{horizontal}, & \text{message from left and right} \end{cases} \tag{3}$$

Each message in the same group is ended in the same time. Memory requirement can be reduce into half via this design. However, we face two problems while implementing above architecture. Those messages in the same group can not be added up directly because they are generated in the different timing. Through the timing analysis, we also notice that messages within the same group are called in the same time in orthogonal direction process but not parallel one. In other words, those message can be added up only after parallel processing. Before backward processing, we have to store another message into the buffer. Here, we make some definition. In horizontal direction processing, we call processing to **right** as *forward* passing and to **left** as *backward* passing. Same definition in vertical processing, to **down** as *forward* passing and to **top** as *backward* passing. While combining two messages into one group the requirement is reduce to half. We can gain the benefit from doing nothing but summing two messages up and storing into memory. The above-mentioned architecture is presented in Fig.4.

Furthermore, we also observe that vertical and horizontal messages overlap in the timing slot which causes the proposed system have to allocate two memory spaces to each message. We start to consider that, can we further reduce the memory via eliminating the timing overlap. If we can postpone the saving time one cycle after finishing backward passing, another message space is not necessary any more. By applying the above scheme, two messages can be stored in the same memory without any conflict. In forward processing, the message from last pixel is summed with it data cost and messages from orthogonal directions then pass to the next pixel. The message from the last pixel is stored into buffer at the same cycle. In the backward type, everything is the same as forward but the message from the last pixel is added with buffer then replacing the message instead of storing into the buffer. At the last, the computation for the proposed BP becomes as Eq.(4). And the architecture is shown in Fig.5

$$\begin{cases} \text{Forward:} & \\ B(j) & = M_{last_{pixel}} \\ M_{last} & = M(i,j) + M_{last} + C(i,j) \\ \text{Backward:} & \\ Temp & = M_{last} + B(j) \\ M_{last} & = M(i,j) + M_{last} + C(i,j) \\ M(i,j) & = Temp \end{cases} \tag{4}$$

4. EXPERIMENTAL RESULT

The proposed system runs software simulation on a Intel i7-3612QM CPU. We implement proposed architectures and counterpart design with TSMC 40nm technology. Fig.6 illustrates the software comparison between traditional BP and the proposed architecture. The proposed architecture gains nearly 20% speed up in software simulation because of the reduction of computation in the proposed architecture.

Computing resource and memory access comparison between the proposed architecture and the counterpart one is

Table 3. The computing resource and memory access comparison.

Architecture	Tile-based	Proposed
Adders	3	3(Backward) 2(Forward)
SRAM	4 + 1	1 + 1
Mem. read	4	3(Backward) 2(Forward)
Mem. write	1	1

Table 4. Memory requirement comparison between tile-based bipartite and proposed two architectures.

	Tile-based	Proposed 1	Proposed 2
Cost(KB)	64	64	64
Message(KB)	320	176	90.75
Total(KB)	384	240	154.75
Reduction (%)	0	37.5	59.7

shown in Table 3. The number of adders is reduce nearly 20% which is the reason that causes our architecture has nearly 20% improvement in software simulation.

The memory requirement of on-chip memory in different structures is shown in Table 4. Follow the same conditions with Table 1, proposed architectures combine two messages together and each combined message becomes 11-bit in length. Proposed 1 architecture is the one with vertical and horizontal message SRAM and proposed 2 is the one with one message SRAM only. By adopting proposed architecture, nearly 40% of memory storage is required.

In Table 5, gate count analysis for each architecture is presented. All architectures are under same synthesis condition. It is worthy mention that, both proposed architectures utilize same combinational circuit and only with different SRAM allocation. Proposed 1 and Proposed 2 architectures reduce more than 40% and 60% gate count respectively.

5. CONCLUSION

A memory efficient BP-based architecture is proposed. Compare to the traditional BP-M, the proposed one requires less computation resource, memory and memory access. There is 20% speed up in software simulation because of computing resource reduction. The memory storage of proposed architecture is nearly 40% compared to non-optimized one. With further design effort, 60% memory storage and its gate count are saved without any lose in performance. We have demonstrated the memory efficient of proposed architectures compared to non-optimized tile-based BP.

Table 5. Gate counts comparison between tile-based bipartite and proposed two architectures.

	Tile-based	Proposed 1	Proposed 2
Memory	1954966	1140397	733112
Reduction (%)	0	41.67	62.5
Non-memory	128818	62785	62785
Reduction (%)	0	51.26	51.26
Total	2083783	1203182	795897
Reduction (%)	0	42.26	61.81

6. REFERENCES

[1] J. Sun, N.N. Zheng, and H.Y. Shum, "Stereo matching using belief propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 7, pp. 787–800, July 2003.

[2] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, Nov 2001.

[3] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1568–1583, Oct 2006.

[4] Y.C. Tseng and T.S. Chang, "Architecture design of belief propagation for real-time disparity estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 11, pp. 1555–1564, Nov 2010.

[5] C.K. Liang, C.C. Cheng, Y.C. Lai, L.G. Chen, and H.H. Chen, "Hardware-efficient belief propagation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 5, pp. 525–537, May 2011.

[6] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, Aseem Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 6, pp. 1068–1080, June 2008.

[7] A. Ahmadzadeh, H. Madani, K. Jafari, F. S. Jazi, S. Daneshpajouh, and S. Gorgin, "Fast and adaptive bp-based multi-core implementation for stereo matching," in *Formal Methods and Models for Codesign, 2013 Eleventh IEEE/ACM International Conference on*. IEEE, 2013, pp. 135–138.

[8] C.C. Cheng, C.T. Li, C.K. Liang, Y.C. Lai, and L.G. Chen, "Architecture design of stereo matching using belief propagation," in *Circuits and Systems, Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 4109–4112.