Patra: Parallel Tree-reweighted Message Passing Architecture

Wenlai Zhao*†‡, Haohuan Fu*†‡, Guangwen Yang*†‡ and Wayne Luk§
*Department of Computer Science and Technology, Tsinghua University
†Ministry of Education Key Laboratory for Earth System Modeling, and Center for Earth System Science,Tsinghua University
‡Tsinghua National Laboratory for Information Science and Technology (TNList)
§Department of Computing, Imperial College London
Email: zhaowl-11@mails.tsinghua.edu.cn, {haohuan,ygw}@tsinghua.edu.cn, w.luk@imperial.ac.uk

***Abstract*—Maximum a posteriori probability inference algorithms for Markov Random Field are widely used in many applications, such as computer vision and machine learning. Sequential tree-reweighted message passing (TRW-S) is an inference algorithm which shows good quality in finding optimal solutions. However, the performance of TRW-S in software cannot meet the requirements of many real-time applications, due to the sequential scheme and the high memory, bandwidth and computational costs. This paper proposes Patra, a novel parallel tree-reweighted message passing architecture, which involves a fully pipelined design targeting FPGA technology. We build a hybrid CPU/FPGA system to test the performance of Patra for stereo matching. Experimental results show that Patra provides about 100 times faster than a software implementation of TRW-S, and 12 times faster than a GPU-based message passing algorithm. Compared with an existing design in four FPGAs, we can achieve 2 times speedup in a single FPGA. Moreover, Patra can work at video rate in many cases, such as a rate of 167 frame/sec for a standard stereo matching test case, which makes it promising for many real-time applications.**

***Keywords*—Markov Random Field, Maximum a posteriori probability, Parallel Tree-Reweighted Message Passing, FPGA**

I. Introduction

MRF (Markov Random Field) has been a very popular and powerful tool in computer vision. Many computer vision applications, such as stereo matching [1], photomontage [2], and background segmentation [3], described as label assignment problems, can be formulated in the framework of MRF and solved using MAP (Maximum a posteriori probability) inference algorithms [4]. The MRF MAP inference algorithms often involve a heavy computation load, making it difficult to meet the real-time processing requirements in many applications. Hardware-based acceleration is one of the most practical solutions to improve the performance of the algorithms. Moreover, as the MRF MAP inference methods can also be used in quite a few other application domains, it is important to design a general and highly efficient framework for various hardware-based MRF MAP methods. This paper introduces our starting efforts towards this goal.

Finding the optimal solution of the MRF MAP problem is NP-hard [6]. Most related algorithms have focused on improving the performance and efficiency of the solving process, such as graph cuts [7], belief propagation (BP) [8], and sequential tree-reweighted message passing (TRW-S) [9]. Among these different options, BP and TRW-S are based on the message passing pattern and both can find an approximation to the global optimal solution.

BP can be easily parallelized, but the accuracy and convergence of BP is unguaranteed. Compared with BP, TRW-S has better convergence properties and shows better accuracy of the results in most cases. However, TRW-S leads to a significantly higher cost in memory, bandwidth and computation and the sequential message passing pattern makes it difficult to parallelize TRW-S.

In this paper, we propose a parallel tree-reweighted message passing architecture (Patra) based on an FPGA. Our major contributions are as follows:

- We propose a parallel message passing pattern for tree-reweighted message passing algorithm, which eliminates the sequential data dependencies and can be parallelized.
- Based on the parallelized algorithm, we propose Patra, which is a fully pipelined design on an FPGA and achieves significant improvements in performance.
- We build a hybrid CPU/FPGA system to test the performance of Patra using the stereo matching application.

The experimental results show that Patra provides about 100 times faster than a software implementation of TRW-S [4], and 12 times faster than a GPU implementation of BP [5]. Compared with an existing design in four FPGAs [10], we can achieve 2 times speedup in a single FPGA on the overall performance with less resource consumption: one-quarter of LUTs and FFs, and three-quarters of Block RAMs. Moreover, Patra can work at video rate in many cases, such as a rate of 167 frame/sec for a standard stereo matching test case, which makes it promising for many real-time applications.

The organization of the paper is as follows. Section II reviews the MRF MAP inference algorithms and existing hardware implementations of them. Section III introduces the parallel tree-reweighted message passing algorithm. Section IV demonstrates our hybrid CPU/FPGA system, including details of the fully-pipelined FPGA design. Section V presents experimental results, and Section VI covers conclusions and future work.

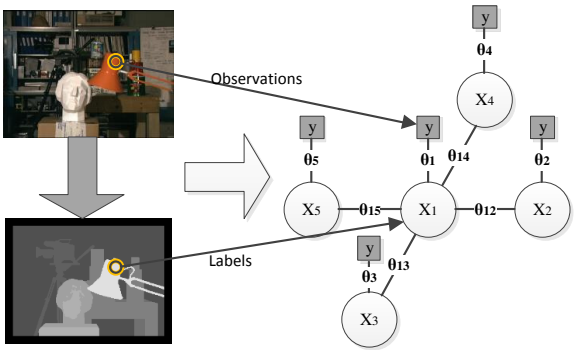

Fig. 1. Mapping a stereo matching problem to a 4-connected MRF

II. Background and Related Work

A. Basic model

A discrete-labeling problem can be mapped to a MRF MAP problem. Figure 1 gives an example for mapping a stereo matching problem to a 4-connected 2D MRF. Consider an MRF as an undirected graph $G = (E, V)$ with a set of nodes V and a set of edge E. For each node $s \in V$, an observation value y_s is given, while the assignment x_s is unknown. Let X and Y represent the vector of assignments and observations on G respectively. The MAP problem can be written as (1).

$$\arg\max_X P(X|Y) = \arg\max_X P(Y|X)P(X) \tag{1}$$

We can easily understand (1) according to the Bayes' Rule, where $P(X|Y)$ is the posterior probability; $P(Y|X)$ is the likelihood of Y on assignment X; $P(X)$ is the prior probability of X.

As we can see in Figure 1, the likelihood is defined by the local relationship (θ_s) between observations and assignments, while prior probability is defined by a pairwise relationship (θ_{st}) between assignments of neighboring nodes. Therefore, the posterior probability can be formulated as (2).

$$P(X|Y) \propto exp(-(\Sigma_{s\in V}\theta_s(x_s, y_s) + \Sigma_{(s,t)\in E}\theta_{st}(x_s, x_t))) \tag{2}$$

If we take the negative log in (2) as an energy function, the maximum a posteriori problem can be formulated as minimizing the *energy function* E over X and θ (3).

$$E(X|\theta) = \theta_{const} + \Sigma_{s\in V}\theta_s(x_s) + \Sigma_{(s,t)\in E}\theta_{st}(x_s, x_t) \tag{3}$$

In practice, we call the first term of the energy function *data energy* and the second term *smoothness energy*. Data energy is the sum of per-node data costs and smooth energy is the sum of smoothness costs of all neighboring node pairs. For different applications, we have different ways to define the costs. Therefore the energy functions are application-dependent. It has been shown in [7] that for general energy functions, finding the globally optimal solution is NP-hard, so many algorithms have been proposed to find approximate solutions.

B. Relevant algorithms

Belief propagation (BP) is an efficient algorithm for energy minimization problems and is widely used in many applications [8]. In BP, each node has a vector of *belief* which indicates the costs of different label assignments to the node. For node s and its neighbor t, belief of s is propagated to t as *messages* and t will re-decide its belief according to the messages from s and then propagates its new belief back. The BP algorithm iteratively processes the message passing and belief re-decision until reaching the global minimal energy.

BP is originally designed to find the exact solution for graphical models without loops. In graphs with loops, the message of one node can pass back to itself through the loops. In such cases, the global energy will not converge but fall into a loop with its value higher than the global minimum [4].

Sequential tree-reweighted message passing (TRW-S) algorithm is designed for loopy graphs. The main idea of TRW-S is to decompose the loopy graph to a set of trees. Then the original energy function on the graph can be transformed to a sum of energy functions on trees as (4),

$$E(X|\theta) = \Sigma_T \omega_T E(X_T|\theta_T) \tag{4}$$

where θ_T and ω_T satisfy

$$\theta = \Sigma_T \omega_T \theta_T \tag{5}$$

Then we obtain relationship between the minimal energy on the graph and the minimal energy on trees is as (6).

$$\min_X E(X|\theta) \geq \Sigma_T \omega_T \min_X E(X_T|\theta_T) \tag{6}$$

The minimal energy on trees determines the lower bound of the global energy. TRW-S seeks for the global minimal energy by maximizing the lower bound.

In TRW-S, messages update in a specific order called monotonic chains. Messages are passed forward and backward alternately by reversing the order after each iteration. It is proved in [9] that the lower bound is guaranteed not to decrease if the update follows the specific order.

C. Challenges for hardware designs

Because messages can be updated synchronously, BP is easy to parallelize and well suited for hardware implementations. In [5], [11]–[13], several GPU and FPGA based BP designs have been proposed for stereo matching application. However, all of them has limited performance due to the high consumption of memory and bandwidth. Moreover, accuracy is a major issue because of BP's disadvantage on loopy graph.

TRW-S always shows better accuracy than BP in comparison experiments on CPU platforms [4] [14], but shares the same challenges on memory and bandwidth consumption with BP. However, there are few works on hardware acceleration for TRW-S because of the sequential message passing pattern. Some efforts have been made in [10]. They propose a multi-FPGA implementation based on a diagonal ordering updating scheme. It has achieved a respectable performance but the sequential data dependency still exists. Consequently, it is still not the most suitable form of algorithm for pipelined designs, and it has a poor scalability for applications with high-resolution.

III. Algorithmic Optimizations

The following summarizes the shortcomings of TRW-S presented in II:

a) Sequential message passing pattern involves data dependency between neighbouring nodes. Therefore, TRW-S can not be parallelized and fully pipelined in hardware designs.
b) Reversing data after each iteration is inefficient for memory accessing, especially for streaming designs based on FPGAs.
c) Iterative computation causes high consumption of bandwidth, memory and computational costs, so that TRW-S can hardly meet the requirements of many real-time applications.

To overcome the shortcomings, we introduce a parallel message passing pattern for tree-reweighted algorithm. The description of the parallel tree-reweighted message passing algorithm is as follows:

Algorithm 1 Parallel Tree-reweighted Message Passing

1: Initialize $M_{us}^{(0)}$ to zero;
2: **for** Iteration i from 1 to $MAXITER$ **do**
3: **for** Each node $s \in V$ **do**
4: compute and normalize the belief as

$$\hat{\theta}_s^{(i)} = \overline{\theta}_s + \Sigma_{(u,s)\in E} M_{us}^{(i-1)} \quad (7)$$

5: **end for**
6: **for** Each edge $(s,t) \in E$ **do**
7: update and normalize the messages as

$$M_{st;k}^{(i)} := \min_j \{(\gamma_{st}\hat{\theta}_{s;j}^{(i)} - M_{ts;j}^{(i-1)}) + \overline{\theta}_{st;jk}\} \quad (8)$$

8: **end for**
9: Check the stop criterion
10: **end for**

In the algorithm, $M_{us}^{(i)}$ represents the message vector sent from node u to s in the ith iteration and $M_{us;j}^{(i)}$ is the jth element of $M_{us}^{(i)}$. $\hat{\theta}_s^{(i)}$ is the belief vector of node s in the ith iteration and $\hat{\theta}_{s;j}^{(i)}$ is the jth element. $\overline{\theta}_s$ and $\overline{\theta}_{st}$ are priori information, where $\overline{\theta}_s$ is the *data cost* vector of node s and $\overline{\theta}_{st;jk}$ is the *smooth cost* when node s and t take the label j and k respectively. γ_{st} is the parameter defined on trees which is always determined by tree decomposition [9].

Based on the parallel algorithm, we propose an FPGA-based parallel tree-reweighted message passing architecture (Patra). The following shows the novel features of Patra and explains how Patra overcomes the shortcomings of TRW-S:

1) In each iteration i, nodes can be updated parallelly using messages from iteration $(i-1)$. There is no data dependency between neighbouring nodes. Therefore, Patra can be fully pipelined in design.
2) No data reversing operations because of the unordered updating. So memory accessing in Patra is very efficient.
3) Forward and backward message passing can be done in one iteration, which reduces the iteration number by half and introduces great improvement to the performance of Patra.

In Patra, we change the sequential update pattern. It has been proved in [9] that unordered update will influence the convergence of the algorithm and therefore cause a loss of accuracy. To evaluate the accuracy, we run a set of standard stereo matching test cases and compare the results with TRW-S. The maximal iteration number is set to 500. We compare the minimal energy obtained by TRW-S and Patra and find that the loss of accuracy in Patra is less than 1%. With such numerical difference, the disparity maps of two algorithms are almost the same in a subjective evaluation, which will be shown in section V-C.

Although TRW-S can converge in some cases, the number of iterations (more than 100 in our experiment) is an unacceptable cost for many real-time applications. In existing TRW-S designs, the algorithm only runs several iterations for each frame to meet the real-time (5 iter/frame with 38 frame/sec in [10], energy loss: 6.6%). In Patra, taking advantage of the speed, we can do more iterations for one frame and achieve lower energy at the same frame rate (21 iter/frame with 38 frame/sec, energy loss: 3.2%). Therefore, Patra can get a better accuracy in real-time tasks and shows potential in dealing with more complex applications.

IV. Architecture Design

We design a hybrid CPU/FPGA system to test Patra using stereo matching applications. In this section, we give an overview of the system and show the details of Patra in FPGA design.

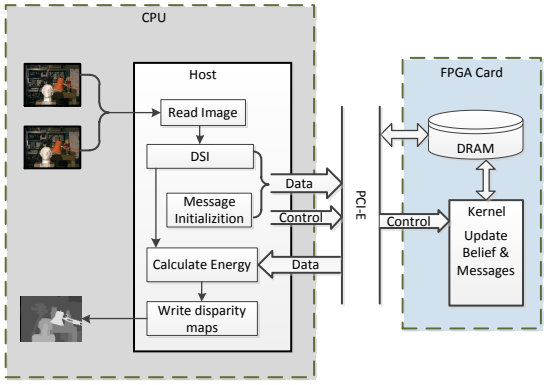

Fig. 2. Overview of the hybrid CPU/FPGA system

A. Hybrid CPU/FPGA system

Stereo Matching takes two or more images as inputs and finds the disparities for every pixel. The disparity map is the output which contains the depth information. We use the software provided on Middlebury website as a reference.

We decompose the stereo matching process into three parts: input part, calculation part and refinement part. Patra is the calculation kernel implemented on an FPGA. The input part

and refinement part run on CPUs in the host. Figure 2 gives an overview of the hybrid system.

The host reads two input images and calculates the disparity space image (DSI) [15], which is defined as the data costs in stereo matching [4]. Then the host initializes the messages and transfers all the data to the DRAM on the FPGA card via PCI-E. After that, the kernel will run for certain iterations which is controlled by the host. Finally, the host reads new messages from the DRAM, calculates the results and saves the disparity maps.

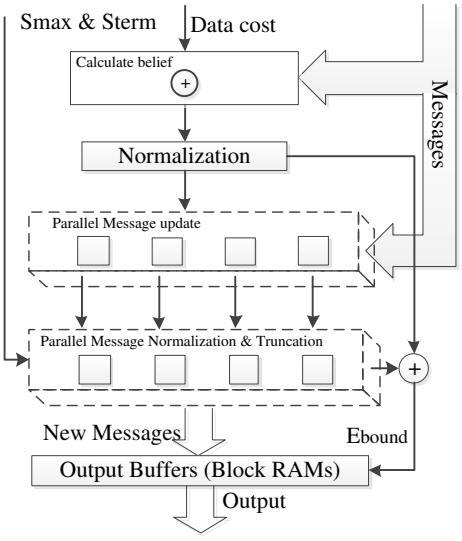

Fig. 3. Architecture and dataflow of Patra

B. Design of Patra

Figure 3 shows the pipelined design of Patra and the dataflow for message update. The operations in the algorithm can be divided into four modules. We give a detailed analysis for each module based on a design with 16 disparity levels (the vector size of belief and messages is 16).

The first module is belief calculation. According to (7), we add the data cost and four messages of one node to calculate the new belief. So there are $4 * 16 = 64$ additions.

The second is belief normalization. We first find the minimum value in the belief vector. Then subtract the min-value from each item in belief vector. So we need 15 comparisons and 15 subtractions in this module.

The third module calculates the new messages. We put four message update units in this module, working in parallel. The update unit is designed based on (8). One difference is that we do not have to add the smoothness cost. Instead, we use truncation operations in the next module which can avoid data overflow [4]. Because we decompose the graph into trees following rows and columns, we got $\gamma_{st} = 0.5$. So that we can use a bit shift operation to do the multiplication in (8). Thus, to implement one message update unit, we need 16 bit-shifts, 16 subtractions and 15 comparisons.

The last module does the message normalization and truncation. Also there are four units in this module. The normalization unit of message is the same with the second module. The truncation operations are based on two parameters [4] (notated as $Smax$ and $Sterm$), which are sent from the host as a control information in Patra. The difference between adjacent elements in messages will be truncated by $Sterm$, which involves 15 subtractions for calculating the differences, 15 comparisons and 15 subtractions for truncation. All values in the message are truncated by $Smax$, which needs 16 comparisons and 16 subtractions. Therefore, in one normalization and truncation unit, there are 46 comparisons and 61 subtractions.

The sum of min-values in normalization modules are stored to DRAM for calculating E_{bound}. The output of the kernel will be temporarily stored in the BRAM. The new messages will be written back to the memory as the input data of four neighbouring nodes for next iteration. Data of these nodes is nonadjacent in the DRAM. So we cache the new messages in BRAM and store data back to DRAM under a sequential memory access pattern.

As we can see, our design is fully pipelined by the four modules and parallel design in the last two modules can shorten the pipeline and improve the efficiency.

C. Hardware design issues

1) Data representation: To decide the data representation in our design, we analyze the range of values for data costs and messages, based on three standard test cases (Tsukuba, Venus, Teddy [4]) of stereo matching.

The data costs are usually described in two ways, *absolute difference* and *squared difference*. An integer truncation parameter (notated as D_{max}) restricts the maximum of data costs. The range of data cost is from 0 to D_{max} (absolute difference) or D^2_{max} (squared difference). In default, D_{max} is set to the color range in the image, for example 255 in test case Tsukuba under absolute different. However in most cases, D_{max} is set to a small number, such as 16 in Venus and Teddy under squared difference. The max value of messages is determined by the $Smax$ because of the truncation operations. $Smax$ is usually set empirically, such as 40 in Tsukuba, 350 in Venus and 10 in Teddy. So in general, the value of data costs and messages is nonnegative and less than 2^{10}, which means a 10-bit unsigned integer is suffice in data costs and messages.

However, there are some other constraints. First, the max value of the intermediate variables is about 5 times of the message value coming out when calculating belief. Second, there are subtractions in message update, so unsigned data type is not proper. Third, in our FPGA platform, data width of input and output is prefer to be multiples of 8-bits because of the truncation problems in data transfer. In order to make our kernel applicable in most cases, we choose 16-bit integer in our design, which can avoid the overflow in computation and is suitable for the FPGA platform.

2) Bandwidth and Disparity levels: For a stereo matching problem, the bandwidth requirement is mainly determined by two factors, the data width and the disparity level. In test case Tsukuba, the disparity level is 16. In each cycle, our kernel takes 1 data cost vector and 4 message vectors as inputs while

TABLE I
RESOURCE UTILIZATION OF PATRA AND F-STRW-S [10]

	Patra	F-sTRW-S
Platform	MaxWorkstation 1 Xilinx Virtex-6 FPGA	Convey HC-1 4 Xilinx Virtex-5 FPGAs
LUTs	73217 / 297600	(71281 / 207360)*4
FFs	110700 / 595200	(154111 / 414720)*4
BRAMs	380 / 1064	(121 / 288)*4

TABLE II
COMPARISON OF MINIMAL ENERGY OBTAINED BY PATRA AND TRW-S

	Tsukuba	Venus	Teddy
Image Size	$384 * 288$	$434 * 383$	$450 * 375$
Disparity Level	16	20	60
TRW-S	313543	3002038	1181766
Patra	313768	3007777	1193379
Relative Error	0.07%	0.19%	1.01%

4 message vectors and 1 integer for E_{bound} as outputs. We use 16-bits data representation, so the requirement of bandwidth will be 290 Byte/cycle. If the kernel works at 100MHz, the bandwidth will be 29 GByte/s.

Generally, we assume the data width is N bits, the disparity level is D_l and the working frequency is f Hz, then the requirement of bandwidth is $(9D_l + 1) \cdot N \cdot f$ (bit/s).

In our design, we use DRAM to store data. The bandwidth of DRAM is 38 GByte/s. We use 16-bits integer to represent the data. The default working frequency of our FPGA is 100MHz. According to the relationship above, the bandwidth of DRAM can meet the requirement of kernel if the disparity level is less than 22.5. Therefore, for test cases with disparity levels more than 22.5 (Venus and Teddy), bandwidth becomes the bottleneck of performance in Patra.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Platform information

We implement our design on a Maxeler FPGA platform [16]. The MaxWorkstation has two Intel i7 quad-core CPU and a Max3 acceleration card with a Xilinx Virtex-6 (SX475T) FPGA. The FPGA card connects to CPU via PCI Express 2.0 x8 which has a bandwidth of 8 GB/s. There are 24GB DDR3 DRAM on the FPGA card with a maximum bandwidth of 38 GB/s. In our experiment, the FPGA is running at 100MHz. We use Maxeler MaxCompiler development environment to program the design and map to the FPGA.

B. Resource Utilization

Table I shows the resource utilization when mapping our design to an FPGA card with 16-bits data costs, 16-bits messages and 16 disparity levels. We take the FPGA-based TRW-S implementation in [10] (F-sTRW-S) as a comparison. F-sTRW-S uses 15-bits data costs, 24-bits messages and 16 disparity levels and is implemented on a platform with 4 Xilinx Virtex-5 (V5LX330) FPGAs. The resources in Virtex-5 and Virtex-6, such as lookup tables(LUT), flip-flops(FF), and block RAMs(BRAM), have same specifications, so it is a relatively fair way to compare the amount of resources used

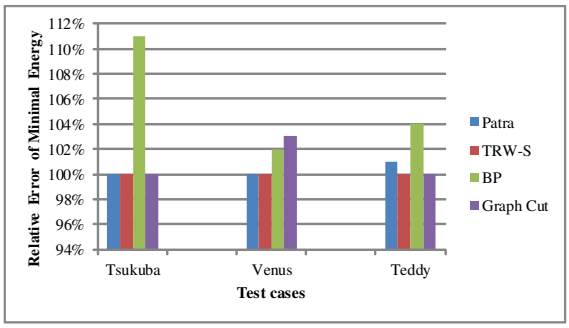

Fig. 4. Comparison of minimal energy for different algorithms

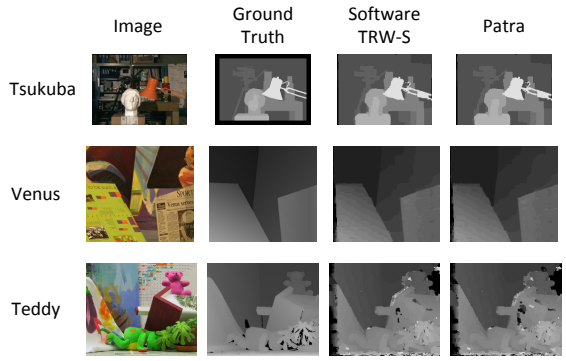

Fig. 5. Disparity maps of Patra and software TRW-S

in two designs. Patra uses more Block RAMs on each FPGA because of the optimization of output messages, which have been discussed in section IV-B. However, in terms of total resource utilization, Patra consumes about one-quarter of LUTs and FFs and three-quarters of Block RAMs.

C. Accuracy

Table II shows the minimal energy obtained by the TRW-S reference software [4] and our design in three standard test cases: Tsukuba, Venus and Teddy. As we have discussed in section III, the loss of accuracy for Patra is less than 0.2% in Tsukuba and Venus, and a little higher (1.01%) in Teddy due to its high disparity levels.

We show the comparison of Patra and TRW-S with two other widely used algorithms, BP and Graph Cut, in Figure 4. In all cases, Patra can find a approximate solution to the global optimum, which is nearly the same with TRW-S and better than BP. Graph Cut is considered to be the most accurate minimization algorithm, but it only works for a specific class of energy functions [9]. Therefore, Patra is more competitive, though Graph Cut shows equivalent accuracy.

The disparity maps of three stereo matching test cases obtained by Patra and TRW-S are shown in Figure 5. As we can see, the results of Patra are almost the same with TRW-S and intuitively acceptable for stereo matching application.

D. Speed

Patra achieves great improvement in speed than TRW-S. In table III, we show the execution time of one iteration and

TABLE III
COMPARISON OF EXECUTION TIME FOR ONE ITERATION (N/A: NO RELATED RESULT IN [5])

		Software Impl. [4]	Tile-based BP [5]	F-sTRW-S [10]	Patra
Platform		Intel Core i5 @2.3GHz	NV GeForce 8800GTS	Convey HC-1 4 FPGAs	MaxWorkstation 1 FPGA
Tsukuba 384*288*16L	Time(msec)	261.5	19.5	3.2	1.2
	Speedup	1×	13.4×	81.7×	**217.9×**
Venus 434*383*20L	Time(msec)	447.1	N/A	9.6	4.8
	Speedup	1×	N/A	46.6×	**93.1×**
Teddy 450*375*60L	Time(msec)	1371.2	112.9	19.4	9.4
	Speedup	1×	12.1×	70.7×	**145.9×**

compare it with the reference software and the F-sTRW-S in [10]. We run the benchmark provided by [4], which includes Tsukuba (384*288*16L), Venus (434*383*20L) and Teddy (450*375*60L). To show the best performance of our design, we configure the kernel with different specifications so that we can run the test cases with different disparity levels. As we have mentioned in section II-C, GPUs have also been widely used for accelerating message passing algorithms but there are few work on accelerating TRW-S with GPUs, so we compare our design with a GPU-based BP (Tile-based BP [5]) to make our results more complete.

The results show that, our design is at least 93 times faster than the software implementation of TRW-S, 12 times faster than the BP implementation on GPUs and about 2 times faster on the overall performance than F-sTRW-S. With the outstanding performance showed in one iteration, Patra can meet the requirements of many real-time applications. For Tsukuba, if we do 5 iterations for each frame, we can run at 167 frame/sec. Even for more difficult cases like Teddy, our design can also runs near video-rate (21 frame/sec for 5 iter/frame).

VI. CONCLUSIONS

This paper presents a novel parallel tree-reweighted message passing architecture (Patra), which adopts a fully-pipelined design targeting FPGA technology. We build a hybrid CPU/FPGA system to evaluate the performance of Patra for applications in stereo matching. With the proposed optimizations on both algorithm and hardware design, our architecture achieves about 100 times speedup over a software implementation of TRW-S. Compared with an existing design of TRW-S in four FPGAs, we achieve 2 times speedup in a single FPGA while consuming much less resources. Moreover, our design can work at video-rate in many cases, which makes it promising for many real-time applications. Current and future work includes extending Patra to cover designs with multiple FPGAs, and to support automated implementations targeting a variety of applications.

ACKNOWLEDGEMENT

This work was supported in part by National Natural Science Foundation of China (Grant No. 61303003,41374113), by National High-tech R&D (863) Program of China (Grant No. 2013AA01A208), by UK EPSRC, by the European Union Seventh Framework Programme (Grant No. 257906, 287804 and 318521), by the HiPEAC NoE, Maxeler University Program, and Altera.

REFERENCES

[1] Y. Boykov, O. Veksler, and R. Zabih, "Markov Random Fields with Efficient Approximations", *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, pp. 648-655, 1998.

[2] A. Agarwala, et al. "Interactive Digital Photomontage", *ACM Trans. Graphics*, vol. 23, no. 3, pp. 294-302, 2004.

[3] Y. Zhou, et al. "Background segmentation using spatial-temporal multi-resolution MRF", *Application of Computer Vision, 2005 Seventh IEEE Workshops on.* pp. 8-13, 2005.

[4] R. Szeliski, O. Veksler and M. Tappen, "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness Based Priors", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068-1080, Jun. 2008.

[5] C. Liang, C. Cheng and Y. Lai, "Hardware-Efficient Belief Propagation", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 21, no. 5, pp. 525-537, May 2011.

[6] P. Dagum, M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP-hard", *Artificial intelligence*, vol. 60, no. 1, pp. 141-153, Mar. 1993.

[7] Y. Boykov, O. Veksler and R. Zabih, "Fast Approximate Energy Minimization Via Graph Cuts", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.

[8] J. Sun, H. Shun and N. Zheng, "Stereo Matching Using Belief Propagation", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 787-800, Jul. 2003.

[9] V. Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568-1583 Oct. 2006.

[10] J. Choi and R. Rutenbar, "Hardware Implementation of MRF MAP Inference on an FPGA Platform", *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on.*, pp. 209-216, 2012.

[11] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming", *in Proc. IEEE 3D Data Processing, Visualization, and Transmission (3DPVT)*, pp. 798-805, 2006.

[12] J. Park, el al., "A 30fps Stereo Matching Processor Based on Belief Propagation with Disparity -Parallel PE Array Architecture", *in Proc. IEEE Int. Symp. Circuits and Systems*, pp. 453-456, 2010.

[13] J. Prez, P. Sanchez and M. Martinez, "High memory throughput FPGA architecture for high-definition Belief-Propagation stereo matching", *IEEE Signals, Circuits and Systems (SCS), 2009 3rd International Conference on.* , pp. 1-6, 2009.

[14] J. Kappes, et al. , "A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problems", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[15] A. Bobik and S. Intille., "Large occlusion stereo", *Intern. Journal on computer vision 33*, pp.181-200, 1999.

[16] O. Pell and V. Averbukh, "Maximum Performance Computing with Dataflow Engines", *Computing in Science & Engineering*, pp. 98C103, 2012.