A Parallel Architecture for High Frame Rate Stereo using Semi-Global Matching

Akshay Jain
akshayj@iiitd.ac.in
Alexander Fell
alex@iiitd.ac.in
Saket Anand
anands@iiitd.ac.in

Indraprastha Institute of
Information Technology
Delhi, India

Abstract

Semi-Global Matching (SGM) is a popular algorithm to calculate depth maps in stereo images offering the best trade-off among accuracy, computational costs and high frame rates. This paper presents two architectural improvements in FPGA implementations of SGM to achieve high frame rates. First, a highly parallel, pipelined and scalable architecture is implemented which stores the intermediate values internally in the Block-RAMs of the FPGA, rendering external, off-chip memory obsolete. The architecture facilitates the parallelization of the cost computations, over the complete disparity range, in every clock cycle. Secondly, a novel SGM architecture based on multi-clock systems is introduced, which allows the integration of both, disparity-level and row-level parallelism and thus obtain even higher FPS rates. Results show that the FPS obtained are higher than any other FPGA based SGM implementation available in literature. On a Virtex-7 FPGA device, for VGA images (640×480 pixels) and a disparity range of 128, a rate of 475 FPS is achieved. A rate as high as 70 FPS is realized for Full-HD images (1920×1080 pixels).

1 Introduction

Various autonomous and semi-autonomous systems like Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs) and Advanced Driver Assistant Systems (ADAS) rely on depth perception for effectively sensing their environments by detecting, localizing and identifying objects around them. Often sensors like LIDARs or RADARs are employed for detection and localization of objects. While these sensors provide accurate depth perception, detection and localization, in many applications these sensors prove to be inadequate. RADARs and 2D LIDARs can only provide range and azimuth information, but not elevation. 3D LIDARs can yield range, azimuth and elevation, however, they are prohibitively expensive and have poor spatial resolution. Moreover, both LIDARs and RADARs fail to capture appearance, making identification of objects very difficult.

Stereo cameras serve as an attractive alternative to LIDARs and RADARs due to their cost effectiveness, small form factor, and high spatial resolution. However, the depth estimates obtained from stereo cameras are much poorer compared to LIDARs and RADARs, particularly at larger depths. This problem can be mitigated by applying depth super-resolution techniques, which rely on fusing multiple depth estimates [22]. This fusion is reliable only if the environment is changing slowly, or equivalently, if the depth estimation can operate at a high frame rate (frames per second or FPS). In addition, high FPS depth maps are constructive for other applications like real-time projection mapping [14] and camera tracking [6]. Unfortunately, high resolution images increase the computational complexity of stereo based depth estimation and therefore achieving higher FPS is very challenging. This challenge is addressed in this paper and a novel, highly parallel and scalable implementation of a popular technique called Semi-Global Matching (SGM) [8], allowing to achieve significantly higher FPS, is proposed.

Stereo depth maps are generated using dense point correspondences across the image pairs. Several approaches exist for dense correspondence search and can be categorized into local, global and semi-global approaches [20]. In local approaches like Sum of Absolute Differences (SAD) and Census based rank transform [24], a small part of the image, which is a window centered around a pixel, is used to calculate the disparity independently. The advantage of the local algorithms is a high throughput at the cost of reduced accuracy. On the other hand, global approaches such as Graph Cuts [15, 20] and Belief Propagation [4], utilize the complete image, resulting in a higher accuracy at a lower throughput. The semi-global approach provides a trade-off between speed and accuracy by using a local method as an initial input followed by the minimization of the global cost function, e.g. the Semi-Global Matching (SGM) algorithm [8].

Software implementations of these algorithms executed on General Purpose Processors (GPP), are too slow to be used for applications requiring high frame rates. This limitation of GPP implementations is due to the sequential nature of the execution of instructions in these architectures. Moreover, the limited number of cores and small cache memory further restrict GPPs processing power. Alternatively GPUs offer a high throughput at the cost of high power consumption (hundreds of Watt) rendering them infeasible in small, portable applications like UAVs. FPGAs overcome these drawbacks of GPPs and GPUs and offer a compact, light weight, fast and low power solution to the problem. However, in order to develop an FPGA implementation, the algorithm needs to be redesigned to fully exploit the parallel nature of the target architecture.

The main contribution of this paper is a novel, highly parallel and scalable implementation of the SGM algorithm. It utilizes internal BlockRAMs as dual-port RAMs and multi-clock based subsystems to obtain a high throughput resulting in high FPS rates. First a single clock based disparity parallel architecture is introduced which calculates disparity at every clock cycle. Secondly a multi-clock architecture is presented to process multiple rows in parallel and obtain even higher FPS. The key features of the architecture in comparison to other SGM implementations are:

- Stream based operation with no external memory
- Multi-clock based, highly parallel and scalable architecture
- Row-level parallelism integrated with disparity-level parallelism
- Higher frame rates obtained upto Full-HD images compared to state-of-the-art

This paper is organized as follows: In section 2 related work is discussed, while in section 3 a brief overview of the SGM algorithm is given. The proposed implementation of the SGM algorithm on an FPGA is presented in detail in section 4. In section 5 obtained results are compared and section 6 concludes the paper.

2 Related Work

To achieve high frame rates, hardware implementations of stereo correspondence algorithms have always attracted attention of researchers. SAD implementations with different modifications can be found in [7, 12, 16] and [17]. The window based approach in SAD suggests a parallel, stream based architecture to achieve high FPS rates. The authors in [13] implemented a Census based architecture and achieved 230 FPS for VGA images (640×480 pixels) and 64 distinct disparities.

With the advancement in FPGA technology, even complex computations required by global methods, can be mapped on its hardware resources. Dynamic Programming (DP) based designs were implemented by [11, 19]. In [19] 64 FPS were achieved for VGA images with 128 disparities. However maps generated by DP suffer from a streaking effect as the minimization function considers only one direction.

An SGM implementation was first introduced in [5]. It achieves 25 FPS for VGA images and 128 disparities. The design utilizes external memory located off-chip, to store the intermediate path costs increasing computation latencies. In addition scalability for higher throughput was limited by the lack of resources of the architecture. The most referred SGM implementation is given in [2]. The presented architecture is a technique based on a systolic array with two-dimensional parallelization concept. The design is able to achieve FPS rates as high as 103 and 167 for VGA images with 128 and 64 disparities respectively. These rates are achieved by computing multiple pixels in parallel (depending on the number of available image rows) for a single disparity value. However with every additional row added to the parallel computation, memory requirements and latency increases significantly.

In order to deal with the high memory requirements of SGM, [9] implemented an efficient SGM algorithm (eSGM) reducing the memory requirements at the expense of an overhead of 56% in compute time. This resulted in a low FPS rate of 33 FPS for VGA images with 64 disparities. In another implementation based on a combination of FPGA and mobile CPU, [10] implemented an SGM based design which aimed at low power and lower-end FPGAs. For 752×480 pixels image, a rate of 60 FPS was achieved at a disparity range of only 32 pixels. A recent SGM implementation with cross-based cost aggregation [23] claims to achieve the highest frame rates for high definition images. The proposed architecture is able to achieve better frame rates than [23].

In this paper an implementation is proposed that achieves high FPS rates with 128 disparities utilizing only internal Dual-Port BlockRAM (BRAM) integrated in the FPGA. The path cost computation has been successfully parallelized covering the complete disparity range in every clock cycle. Further a multi-clock design is used to integrate row level parallelism along with the disparity parallel design. Section 4 discusses the architecture and its implementation in detail.

3 SGM Overview

SGM [8] minimizes the global cost function given by (1) along different one dimensional paths.

$$\begin{aligned} L_r(p,d) = C(p,d) + \min(&L_r(p-r,d), L_r(p-r,d-1)+P_1, \\ &L_r(p-r,d+1)+P_1, \min_i(L_r(p-r,i)+P_2)) - \min_k(L_r(p-r,k)) \end{aligned} \tag{1}$$

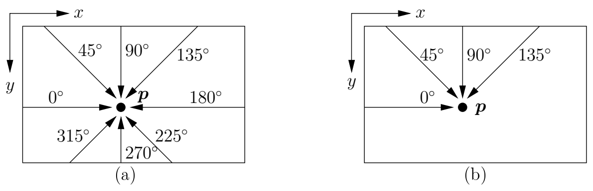

Figure 1: This figure illustrates (a) 8 path and (b) 4 path orientations

$C(p,d)$ represents the initial cost for pixel p at disparity level d, calculated by one of the local methods like SAD. Path cost, L_r, is calculated for a pixel p by adding $C(p,d)$ and the penalized minimum path cost of the previous pixel, $p-r$. There is no penalty for the path cost at the same disparity, d. To accommodate surfaces with smoothly varying depths, a small penalty P_1 is imposed on the adjacent disparities ($d \pm 1$), while a larger penalty P_2 is added to the minimum of the costs of all the remaining disparity levels. The subtrahend in (1) limits the maximum value resulting in a reduced bit-width of $L_r(p,d)$ lowering the overall memory requirements. The path costs are aggregated and the disparity level assigned to the pixel p is the one which minimizes its aggregated cost. For the detailed algorithm refer to [8].

A total of eight paths oriented at multiples of 45° start from the edge of the image and end at pixel $p(x,y)$ as shown in fig. 1. Since images are streamed in a raster scan method, it becomes impossible to calculate the costs of lower four paths ($180°, 225°, 270°$ and $315°$), unless the complete image is first stored and then processed in two passes (forward and backward pass). It has been observed that a depth map obtained using only the four paths shown in fig. 1 ($0°, 45°, 90°$ and $135°$), is similar to that obtained using eight paths [2]. As there is no need for separate forward and backward passes, the requirements for memory space and computation time are reduced at a marginal increase in error by 1-2%.

4 Architecture and Implementation

The complete design is divided into two subsystems as shown in fig. 2. The rectified left and right image pixels are streamed in a raster scan mode into the SAD Computation Unit. The initial costs $C(p,d)$, computed by the SAD Unit are sent to the Path Computation Unit via Row-Demux and FIFO synchronizers. Path costs generated by the Path Computation Unit are aggregated by the Path Aggregation Units and given to the Disparity Calculation Units. The disparity calculation follows a winner takes all (WTA) approach and outputs the minimum aggregated cost index as the final disparity. These disparities are then interleaved via FIFO synchronizers and Row-Mux to obtain final disparities in sequence. Section 4.1 explains in detail how disparity level parallelism is achieved. The disparity value is calculated at every clock cycle. Section 4.2 discusses the scheme and architecture of the overall multi-clock design. The multi-clock design allows to integrate disparity level and row-level parallelism. The implementation details are discussed in section 4.3.

4.1 Disparity-Level Parallelism

This section introduces the proposed architecture design that achieves disparity-level parallelism and therefore is able to compute the final disparity value in every clock cycle. For

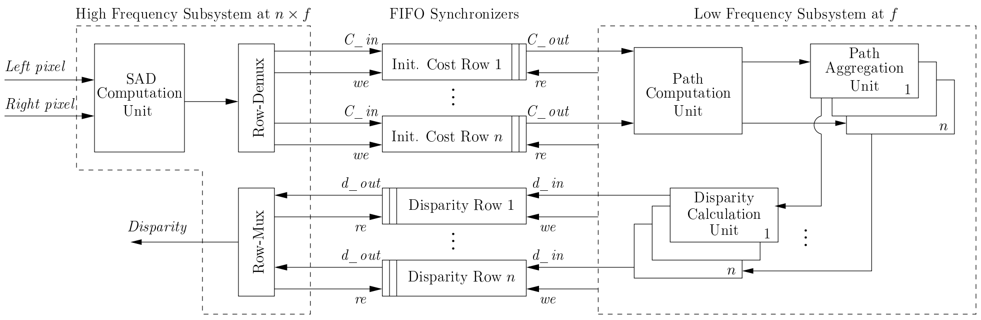

Figure 2: Overall Architecture: The Path Computation Unit is designed to process multiple rows along with disparity-level parallelism. The multi-clock design with two subsystems operating at different frequencies and communicating via FIFO synchronizers, support the integration of row-level parallelism.

simplicity the architecture for single row computation is explained in the following subsection.

4.1.1 Path Computation Unit

The architecture of the Path Computation Unit is shown in fig. 3. In every clock cycle the SAD values are taken as input and the corresponding four path costs (for the complete disparity range) are calculated. Identical Cost Units are used to calculate the path costs L_{0°, L_{45°, L_{90° and L_{135° parallelly. To calculate $L_r(p,d)$ the value for $L_r(p-r,d)$ is required.

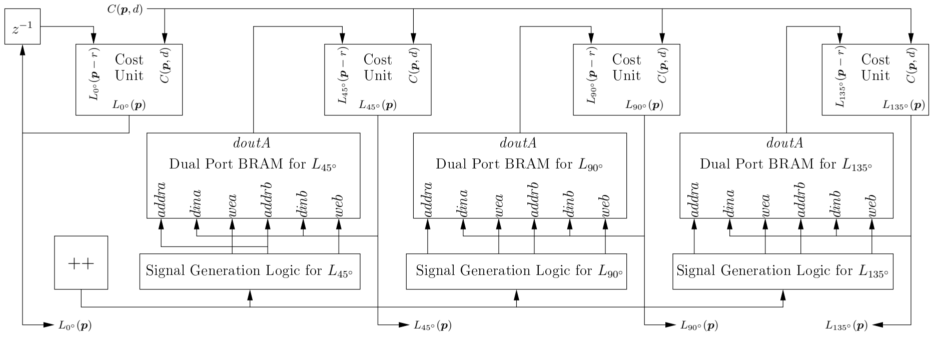
z^{-1}
$C(\mathbf{p},d)$
Cost Unit
$L_{0^\circ}(\mathbf{p}-r)$
$L_{0^\circ}(\mathbf{p})$
$L_{45^\circ}(\mathbf{p})$
$L_{90^\circ}(\mathbf{p})$
$L_{135^\circ}(\mathbf{p})$
doutA
Dual Port BRAM for L_{45°
Dual Port BRAM for L_{90°
Dual Port BRAM for L_{135°
addra
dina
wea
addrb
dinb
web
++
Signal Generation Logic for L_{45°
Signal Generation Logic for L_{90°
Signal Generation Logic for L_{135°

Figure 3: Architecture of the Path Computation Unit

For path L_{0°, $p-r$ is the same pixel for which new path costs were calculated in the previous clock cycle. They are fed back via a delay unit (refer to fig. 3). However for the paths L_{45°, L_{90° and L_{135° the corresponding $p-r$ pixels are different and the same optimization cannot be applied. Thus, they need to be stored and then fetched separately as per the demand. This irregular access pattern makes parallelization of SGM complex and increases the memory requirements. Therefore, three dual-port BRAMs are implemented to address this problem and thus eliminating the need of off-chip, external memory. An up-counter with a maximum count equal to the width of the image, is used for the address generation. Three different logic units generate the required signals (*addra, addrb, wea* and *web* in fig. 3) based on the counter value. More details about the signal generation are given in section 4.1.3.

4.1.2 Cost Unit

The Cost Unit is inspired by [18] which calculates the path costs for a fixed disparity range in parallel. The design has been extended for the whole disparity range to calculate the path costs in parallel. Although this extension results in a higher resource requirement, a better performance in terms of FPS is achieved. On obtaining the new path costs, a tree based approach is used to calculate the new minimum $min(L(p,d))$. This minimum value is concatenated with the calculated path costs and stored together in the BRAM.

4.1.3 Memory Mapping and Retrieval of Path Costs

Three dual port memories are used to store and retrieve the three path costs (refer to fig. 3). Every path cost consists of a total of d values plus one minimum, i.e. $d+1$ values. These values are concatenated and stored as one word in the BRAM. The total width of each memory is configured according to the width w of the image. A total memory capacity of $w+1$, $w+1$ and $w+2$ words is required for L_{45°, L_{90° and L_{135° respectively. The boundary conditions are considered for each path while allocating the memory.

Considering a pixel p at position (x,y) in the image, to calculate its path costs $L_r(p,d)$ the previous path costs $L_r(p-r,d)$ are read. In the next clock cycle $L_r(q,d)$ for pixel q with position $(x+1,y)$ are calculated, while the results of $L_r(p,d)$ are stored. The path costs are read and written from different addresses of the block RAMs. These addresses can be obtained in accordance with the $x-$coordinates of the pixel being processed. Fig. 4 summarizes this relation by giving an example for 2 pixels, p and q at $x=4$ and $x=5$ respectively.

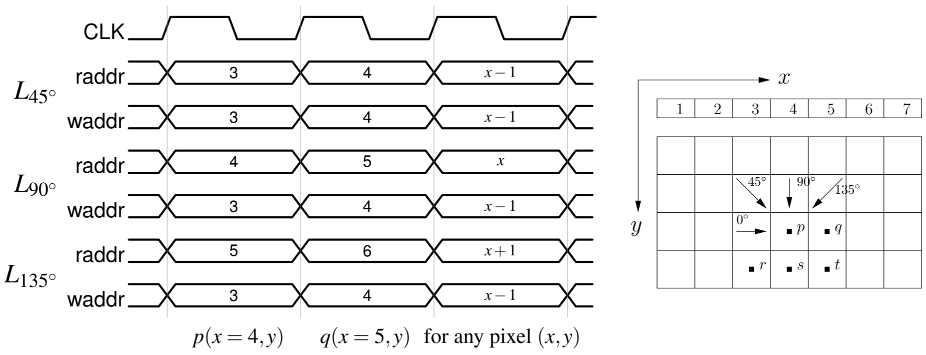

Figure 4: Access patterns for read and write operations of Path Costs

The access pattern for pixel p at the position $x=4$ is shown in fig. 4. Path costs L_{0° are obtained from the Delay Unit as explained in section 4.1.1. L_{45°, L_{90° and L_{135° are read from the address locations 3, 4 and 5 ($x-1$, x and $x+1$) of their corresponding BRAMs, respectively. Due to the parallel design of the Cost Units, the new path costs are available in one clock cycle. At $x=5$, these values need to be written at their required memory locations. L_{45° calculated at $x=4$ will be required for pixel t (fig. 4), thus it needs to be stored at address location 4 in the next clock cycle. Similarly, L_{90° and L_{135°, required for pixel s and r, are stored at address location 4 in their respective memories.

Since, there is a need to access different addresses for reading and writing in a single clock cycle, dual-port BRAMs are used. The memories are implemented in a read before write mode, to prevent any overwriting. Using these conditions the write enable signals and addresses for both port A and B of the dual-port memory are generated by the Signal Generation Logic Units (fig. 3). Port A is always used for reading out the output, while port B is used to only write the calculated path costs. This memory mapping allows to achieve high FPS rates without using an external memory.

4.2 Multi-Clock Based Integration of Row-Level Parallelism

With the above approach one pixel is processed every clock cycle and the disparity is calculated parallelly. However, the maximum operating frequency is limited by the computationally expensive Path Computation Unit. Therefore, in every clock cycle more pixels need to be processed to achieve higher FPS. Although the row-level parallelism is similar to [2], the proposed design calculates disparities in every clock cycle. To allow this integration the SAD Unit must provide the initial costs at higher throughputs (n times in order to process n pixels in one clock cycle). Due to an inherent parallel structure and lower complexity, SAD Unit is able to run at higher frequencies compared to the overall SGM design. This advantage of SAD can be leveraged in a multi-clock architecture.

Fig. 2 shows the top level architecture of the multi-clock based row-level parallelism integrated with disparity-level parallelism. The design operates on two clocks ($n \times f$ and f). The high frequency subsystem operates on a clock frequency of $n \times f$, where n is the number of rows which are used for parallel processing. The FIFOs select their read and write clock frequencies equal to that of the subsystem from which the read enable (re) and write enable (we) signals are fed into them, respectively.

As shown in fig. 2, SAD costs are calculated in the high frequency subsystem. These initial costs have to cross over into the low frequency subsystem utilizing n FIFO synchronizers. Similarly, the n disparities calculated parallelly by the low frequency subsystem are read sequentially by the high frequency subsystem via FIFO synchronizers. Row-Demux and Row-Mux units are used to deinterleave and interleave the initial costs and the final disparity values, respectively. The final disparities are read out from the high frequency subsystem.

4.3 Implementation Details

The design is implemented using a combination of Bluespec System Verilog (BSV) [1] and the Xilinx System Generator (Sysgen) tool. Verilog codes obtained from BSV are integrated in the Sysgen design. Using Xilinx ISE/Vivado, the Sysgen design is used for logic synthesis and implementation onto a target FPGA. Practically the image resolutions for different cameras vary, but the disparity range for depth maps is fixed. The BSV codes and Sysgen design can be easily scaled for any image resolution by changing the input parameters for a fixed disparity value. The design implements the same SGM algorithm as that by OpenCV Semi Global Block Matching (OCV-SGBM) [3].

5 Results

Different measures are used to compare the results of real-time implementations of stereo correspondence algorithms. As stated earlier, currently SGM provides the best trade-off

Table 1: Error rates for all pixels, including the occluded pixels, at a threshold of 1 pixel

Work	Method	Tsukuba	Venus	Cones	Teddy	Average
[13]	Census	11.56%	5.27%	17.58%	21.5%	14%
[19]	DP-ML	13.58%	5.33%	-	-	>13%
[8][1]	SGM + Rank transform (4-paths)	6.8%	4.1%	9.5%	13.3%	8.425%
[23]	SGM + cross-based cost aggregation	3.27%	0.89%	7.74%	12.1%	6%
Proposed	SGM + SAD (4-paths)	6.34%	2.95%	11.46%	11.71%	8.12%

[1] results available for non-occluded regions.

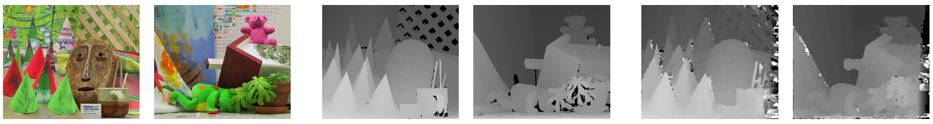

Figure 5: *cones* and *teddy* dataset (a) Original (b) Ground Truths (c) Obtained maps.

between accuracy and speed. The accuracy is measured by calculating the percentage of pixels, for which the calculated disparity differs from the ground truth by a given threshold. Section 5.1 discusses the accuracy results obtained for a threshold of 1 pixel of the Middlebury [20, 21] dataset.

The comparison of the speed for real-time implementations is not straight forward. This is due to the variations in the different evaluation platforms used and modifications in the algorithm implemented. One method is to observe the minimum clock frequency for which real-time results (30 FPS) are achieved. Section 5.2 compares the results obtained using this method and clearly shows that the proposed approach outperforms current implementations available. The other method is to evaluate the maximum performance, i.e. million disparity estimates per second (MDE/s) which reflects the FPS rates. Results for the MDE/s and maximum FPS obtained are discussed in section 5.3.

5.1 Accuracy

All pixels, including the occluded pixels have been considered while calculating the error rates. Table 1 shows the error rates on the Middlebury [20, 21] dataset, for a threshold of 1 pixel. The average error rates for [13] and [19] are on a higher side (around 14%) as compared to that of [8, 23] and proposed work (8.425, 6 and 8.12%, resp.). A slight increase in error of the proposed implementation as compared to [23] is justified, as SGM with cross-based aggregation is slightly more accurate than with SAD. However, SAD costs can be generated at higher speeds in comparison to cross-based aggregated costs. Fig. 5 shows disparity maps generated, corresponding to the right image of the *cones* and *teddy* dataset.

5.2 Minimum Clock Frequency for 30 FPS

The minimum clock frequency required for 30 FPS is considered a fair estimate to compare real-time implementations. Since this is moderately independent of the evaluation platform and accounts mostly for the algorithm and the architecture employed. Table 2 shows the comparison for VGA images. The SGM implementation of [2] obtained the 30 FPS requirement for VGA images at a minimum frequency of 38 MHz. The proposed work is able to achieve the same FPS at a frequency as low as 9.3 MHz.

Table 2: Minimum clock frequency required to achieve 30 FPS for 640×480 images

Work	[13]	[19]	[5]	[2]	**Proposed**
Disparity Range	64	128	64	128	128
Minimum Clock Frequency	12MHz	100MHz	133MHz	38MHz	**9.3MHz**

Table 3: Maximum performance results for 640×480 images on Virtex-5 LX220T

	$d = 64$				$d = 128$			
	Max. freq.	FPS	LUTs	Memory (BRAM)	Max. freq.	FPS	LUTs	Memory (BRAM)
[2]	133MHz	167	35.3%	2380kb	133MHz	107	**35.3%**	2380kb
Proposed	**74MHz**	**240**	**34%**	2808kb	**54MHz**	**175**	68.7%	5076kb

5.3 Maximum FPS and Corresponding Operating Frequency

The maximum FPS obtained and the corresponding operating frequency are mainly dependent on the platform for which the design is targeted. The proposed design is first compared with [2], which implemented SGM on the same FPGA (Xilinx Virtex-5 LX220T). Table 3 shows the results obtained for the single row design (due to limited resources of Virtex-5 it does not support multiple row design). For the same specifications the proposed work (single row) achieves 40 to 60% higher FPS at 0.4 to 0.55 times of the maximum frequency. This is obtained at the expense of using extra LUTs and inbuilt BRAMs.

Table 4 compares the result for two architectures, both for single row (only disparity-level parallelism) and double row (multi-clock with $n = 2$). It shows the results obtained for Xilinx Virtex-7 FPGAs for different image resolutions. Table 4 confirms that moving from single row to double row architecture, the FPGA resource utilization, i.e. LUT (look-up-table) and BRAM, increases significantly. For similar increment in the FPGA utilization, extending the architecture for $n \geq 3$ improves the FPS only marginally by 2-3%. Based on preliminary results the SAD Computation Unit poses a bottleneck due to the maximum operating frequency limit.

For VGA images a maximum FPS rate of 475 and for full-HD rate of 70 FPS is achieved. The comparison of million disparities estimated per second (MDE/s) with previous works is given in table 5. As it can be observed, the proposed work gives highest MDE/s and is thus able to compute the highest FPS over all image resolutions.

Table 4: Summary of results and resource utilization for different image resolutions on Virtex-7 690t FPGA with $d = 128$. Columns marked 'Single' and 'Double' respectively refer to implementations with only disparity-level parallelism and integrated row-level parallelism with two rows.

Image Size	**LUTs**		**BRAMs (kb)**		**Max. Frequency**[1]		**FPS**	
	Single	**Double**	**Single**	**Double**	**Single**	**Double**	**Single**	**Double**
640×480	90123 (21%)	168471 (39%)	5076 (10%)	7452 (14%)	79 MHz	73 MHz	257	475
960×540	90532 (21%)	168381 (39%)	5076 (10%)	8208 (16%)	79 MHz	73 MHz	152	277
1280×720	90069 (21%)	169080 (39%)	10098 (21%)	16290 (31%)	77 MHz	70 MHz	83	152
1920×1080	90871 (21%)	170161 (39%)	10098 (21%)	16290 (31%)	79 MHz	73 MHz	38	70

[1] Maximum operating frequency of the lower frequency subsystem in multi-clock design, i.e. f.

Table 5: Comparison of FPGA based SGM Implementations

Work	**Image Size**	**Disparity Levels**	**FPS**	**MDE/s**
[5]	680×400	64	25	435.2
[2]	640×480	128	103	4050
[23]	1600×1200	128	42.61	10472
Proposed	**1920 × 1080**	**128**	**70**	**18580**

6 Conclusion

In this paper a new architecture for implementing the SGM algorithm is introduced. By utilizing dual-port Block RAMs (no external memory) and parallel path cost computation a faster scan-aligned SGM implementation is presented. The multi-clock based design allows the integration of disparity-level and row-level parallelism. Results show that the proposed design is faster than any other previous work reported in literature. An FPS rate of 475 for VGA images is achieved. The design is scaled upto Full-HD images for which high frame rates of 70 FPS are obtained. Thus, a method to achieve high frame rates and accurate depth maps for higher resolution stereo images has been presented.

References

[1] Arvind. Bluespec: A Language for Hardware Design, Simulation, Synthesis and Verification Invited Talk. In *Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, MEMOCODE '03, page 249, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1923-7. URL http://dl.acm.org/citation.cfm?id=823453.823860.

[2] 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. In *Embedded Computer Systems (SAMOS), 2010 International Conference on*, pages 93–101, July 2010. doi: 10.1109/ICSAMOS.2010.5642077.

[3] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, November 2000.

[4] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Belief Propagation for Early Vision. *Int. J. Comput. Vision*, 70(1):41–54, October 2006. ISSN 0920-5691. doi: 10.1007/s11263-006-7899-4. URL http://dx.doi.org/10.1007/s11263-006-7899-4.

[5] Stefan K Gehrig, Felix Eberli, and Thomas Meyer. A real-time low-power stereo vision engine using semi-global matching. In *International Conference on Computer Vision Systems*, pages 134–143. Springer, 2009.

[6] Ankur Handa, Richard Newcombe, Adrien Angeli, and Andrew Davison. Real-time camera tracking: When is high frame-rate best? *Computer Vision–ECCV 2012*, pages 222–235, 2012.

[7] Masanori Hariyama, Yasuhiro Kobayashi, Haruka Sasaki, and Michitaka Kameyama. FPGA implementation of a stereo matching processor based on window-parallel-and-

pixel-parallel architecture. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(12):3516–3522, 2005.

[8] Heiko Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008.

[9] Heiko Hirschmüller, Maximilian Buder, and Ines Ernst. Memory efficient semi-global matching. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:371–376, 2012.

[10] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4930–4935, Sept 2014. doi: 10.1109/IROS.2014.6943263.

[11] Hong Jeong et al. Real-time stereo vision FPGA chip with low error rate. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 751–756. IEEE, 2007.

[12] Yunde Jia, Xiaoxun Zhang, Mingxiang Li, and Luping An. A miniature stereo vision machine (MSVM-III) for dense disparity mapping. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 728–731. IEEE, 2004.

[13] 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*, 20(1): 15–26, 2010.

[14] CHEN Jun, Takashi Yamamoto, Tadayoshi Aoyama, Takeshi Takaki, and Idaku Ishii. Real-Time Projection Mapping Using High-Frame-Rate Structured Light 3D Vision. *SICE Journal of Control, Measurement, and System Integration*, 8(4):265–272, 2015.

[15] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001.

[16] SungHwan Lee, Jongsu Yi, and JunSeong Kim. Real-time stereo vision on a reconfigurable system. In *International Workshop on Embedded Computer Systems*, pages 299–307. Springer, 2005.

[17] Stefania Perri, Daniela Colonna, Paolo Zicari, and Pasquale Corsonello. SAD-based stereo matching circuit for FPGAs. In *Electronics, Circuits and Systems, 2006. ICECS'06. 13th IEEE International Conference on*, pages 846–849. IEEE, 2006.

[18] Mikołaj Roszkowski and Grzegorz Pastuszak. FPGA design of the computation unit for the semi-global stereo matching algorithm. In *Design and Diagnostics of Electronic Circuits & Systems, 17th International Symposium on*, pages 230–233. IEEE, April 2014.

[19] Siraj Sabihuddin, Jamin Islam, and W James MacLean. Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array. In *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, pages 001461–001466. IEEE, 2008.

[20] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, April 2002. ISSN 0920-5691. doi: 10.1023/A:1014573219977. URL http://dx.doi.org/10.1023/A:1014573219977.

[21] Daniel Scharstein and Richard Szeliski. High-accuracy Stereo Depth Maps Using Structured Light. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, pages 195–202, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1900-8, 978-0-7695-1900-5. URL http://dl.acm.org/citation.cfm?id=1965841.1965865.

[22] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. LidarBoost: Depth Superresolution for ToF 3D shape scanning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 343–350. IEEE, 2009.

[23] W. Wang, J. Yan, N. Xu, Y. Wang, and F. H. Hsu. Real-Time High-Quality Stereo Vision System in FPGA. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(10):1696–1708, Oct 2015. ISSN 1051-8215. doi: 10.1109/TCSVT.2015.2397196.

[24] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *European conference on computer vision*, pages 151–158. Springer, 1994.