Instruction Set Extension for High Throughput Disparity Estimation in Stereo Image Processing

Christian Banz, Carsten Dolar, Fabian Cholewa, Holger Blume
Institute of Microelectronic Systems
Leibniz Universität Hannover
Hannover, Germany
Email: {banz,dolar,cholewa,blume}@ims.uni-hannover.de

Abstract—This paper presents the implementation and evaluation of an application-specific instruction set for a customizable RISC-processor for very high throughput stereo image processing. Compared to the base processor the overall processing time is accelerated by a factor of over 130, while the processor silicon area requirement increases only by a factor of 2.9. The processor has been enhanced with algorithm-specific extensions (i.e. special functional units), as well as with extensions that are not restricted to a specific algorithm (e.g. single-instruction-multiple-data). Hereby, the special functional units account for 50 % of the speed-up, but less than 14 % of the processor silicon area requirement. The proposed processor extensions thereby sustain the full flexibility of a programmable processor while enabling disparity estimation of 640x480 stereo video sequences at 20 fps when running at a clock frequency of 373 MHz.

Keywords—ASIP; stereo video processing; semi-global matching; custom instructions

I. INTRODUCTION

One of the most demanding tasks in stereo image processing is the computation of the depth information from the images delivered by a stereo camera system. Ideally, after this processing step each pixel in the images will have a depth information attached to it, thus yielding a dense depth-map of the recorded scene. There are several algorithmic approaches that estimate the depth information by identifying corresponding pixels in both images and derive the depth from the pixel location difference (referred to as disparity). The challenge for disparity estimation algorithms is to find the correct correspondences.

The semi-global matching (SGM) by Hirschmüller [1] is among the top performing disparity estimation algorithms. SGM is a very promising candidate to be applied in various machine vision tasks, e.g. in automotive applications [2]. The combination of rank transformation and SGM has been shown to be insensitive to a number of interferences such as lighting, exposure, and gamma differences [3]. It has been shown in [2] and [4], that dense disparity estimation based on SGM can be performed in real-time using dedicated hardware. The disadvantage of a dedicated hardware solution is however the lack of flexibility concerning changes in the algorithm. Thus, mapping the algorithm on a programmable device is a preferable solution because algorithmic changes result in software modification, not hardware modification.

It is, however, not possible to meet real-time constraints with standard processors or even with digital signal processors for the SGM. To overcome this limitation, application specific instruction set processors (ASIP) offer a good compromise between flexibility and specialization. However, designing an ASIP from scratch involves high development effort and is very time-consuming, since it necessitates not only hardware design and verification but also the creation of supporting software tools, e.g. linker, compiler, scheduler. Adequate software development tools are one major requirement for successful deployment of new ASIPs [5]. Consequently, a number of hardware/software environments evolved, which provide customizable and extendable processor templates with accompanying supporting software tools. New features added to the basic instruction set architecture become available to the software developer e.g. via C intrinsics or new assembler mnemonics. With the help of such environments, time-to-market for ASIP design and application implementation can significantly be reduced.

This contribution presents the design and implementation of a custom instruction set for the efficient computation of semi-global matching disparity estimation using a commercially available ASIP development environment. This case-study design is based on the Xtensa environment by Tensilica [6]. A set of special functional units is proposed to accelerate the most computation intensive arithmetic equations of the algorithm. To maximize performance these units are integrated into a generic media processing instruction set extension.

Similar work has been presented in [7] using the the RAPANUI design space exploration environment, which includes a pipeline simulator, an instruction scheduler, and parameterized HDL description of a VLIW architecture template. It could be shown, that real-time conditions for dense depth map generation using SGM can be fulfilled. A detailed comparison of the SGM implementations on both ASIPs and dedicated hardware [4] is presented in Section IV.

Beucher et al. [8] presented similar work on real-time motion estimation and demonstrated a speed-up over 100 for full-search block matching on a Tensilica LX configurable processor. In [9] a multi-core system using the configurable LX cores is presented for a 3D target tracking algorithm, that achieves a speed-up of 22.

The remainder of this paper is organized as follows: Section II provides algorithmic background on SGM-based disparity estimation for generating the depth maps. In Section III the
II. DISPARITY ESTIMATION USING SEMI-GLOBAL MATCHING

Disparity estimation is the task of identifying the projection point of the same 3-D real-world point in two or more images taken from distinct viewpoints. The disparity \( d \) is the location difference of both projection points in the stereo images (see Fig. 1). The main challenge for disparity estimation algorithms is to identify corresponding projection points, as a very high level of ambiguity exists. Once the disparity \( d \) has been obtained, the calculation of the distance \( z \) between the 3-D point and the baseline of the cameras is straightforward

\[
z = fT / (x_l - x_r) = fT / d
\]

where \( f \) and \( T \) are the rectified focal length and the baseline of the camera pair, respectively. Due to the underlying epipolar geometry [10] of a stereo-camera setup, the search space for corresponding pixels is oriented along the epipolar lines. A separate image transform, the rectification, is executed beforehand and aligns the epipolar lines in such a way that they are parallel to the scanlines of the image [11].

In most stereo matching methods, a similarity measure between two pixels in the base and match image (or left and right image, respectively) is calculated and those with highest accordance or correlation are assigned as corresponding. In this work the similarity measure is calculated in three steps: matching cost calculation, path cost aggregation, and cost summation. The matching costs \( C(p,d) \) between a pixel \( p = [x,y]^T \) in the base image and its potentially corresponding pixels in the match image are calculated as

\[
C(p,d) = |R_l(x,y) - R_r(x-d,y)|
\]

where \( R \) is the area-based non-parametric rank-transform [12]. It is defined as the number of pixels \( p' \) in a square \( M \times M \) neighborhood \( A(p) \) of the center pixel \( p \) with an intensity \( I(p') \) less than \( I(p) \).

\[
R(p) = \left\| \{ p' \in A(p) | I(p') < I(p) \} \right\|.
\]

In order to deal with non-unique or wrong correspondences due to low texture and ambiguity, the semi-global matching introduces consistency constraints by aggregating matching costs along several independent, one-dimensional paths across the image [1]. An example for four path orientations, as used in this work, is shown in Fig. 2. The path costs \( L_r(p,d) \) are aggregated along a path \( r \) according to

\[
L_r(p,d) = C(p,d) + \min \{ L_r(p-r,d) \},
L_r(p-r,d-1) + P_1,
L_r(p-r,d+1) + P_1,
\min_i L_r(p-r,i) + P_2 - \min_i L_r(p-r,i)
\]

The first term describes the primary matching costs. The second term adds the minimal path costs of the previous pixel \( p - r \) including a penalty \( P_1 \) for disparity changes and \( P_2 \) for disparity discontinuities, respectively. Discrimination of small changes \( |\Delta d| < 1 \) pixel (px) and discontinuities \( |\Delta d| > 1 \) px allows for slanted and curved surfaces on the one hand and preserves disparity discontinuities on the other hand. \( P_1 \) is an empirically determined constant. \( P_2 \) is adapted to the image content with \( P_2 \propto I(p) - I(p-r) \)−1. The last term prevents constantly increasing path costs. For a detailed discussion refer to [1].

\[ S(p,d) = \sum_r L_r(p,d). \]
A detailed algorithm analysis was performed. The most computation intensive single arithmetic function is the calculation of the path costs according to (4). Including the loads and stores of the intermediate results 1.0 billion cycles of a total of 2.1 billion cycles are required on the base processor. This is a consequence of being a hotspot in three domains: First, in the computational domain due to the number of operations. Second, in the data flow domain because the previous pixel’s path cost minimum must be known before continuing with the recursive computation. And third, in the memory access domain because of the enormous amounts of intermediate data generated that must be stored for later processing stages.

Since 50 % of the total computation time is spent on the path cost computation it is clearly necessary to investigate concepts to accelerate this calculation. To meet the high performance demands of the entire disparity estimation algorithm for such a complex case a single functional unit will not be sufficient. To achieve global acceleration three key concepts have been identified in [8]:

- **Perform parallel computations.** Perform as many computations in parallel as possible (e.g. using SIMD). This implies a more data-greedy algorithm implementation which requires higher bandwidth load/store operations and increased data reuse.
- **Maximize data reuse.** Reuse data (e.g. intermediate results or once-loaded pixel) as often as possible and as soon as possible so that it can be kept in local fast memory (e.g. register file). This eliminates the need to load the same data several times or store intermediate results into slow external memory.
- **Efficient cache usage.** Access data on the external memory in such a way that it can easily be cached. Except for boundary conditions always use all the data contained in a cache line. Thus, lengthy pipeline stalls due to cache misses are minimized.

To achieve the required overall acceleration, the four following distinct enhancements to the base processor have been implemented into which these concepts have been incorporated.

- **A 16x512-bit register file** with specialized load/store instructions.
- **Four special functional units** implement performance critical composite arithmetic functions required by various parts of the algorithm.
- **A generalized media instruction set** for highly parallel data processing using SIMD, which is not restricted to a particular algorithm.
- **Fast, local 64-kB data memory** directly connected to the processor for single cycle access.

![Fig. 3. Disparity estimation algorithm using rank transformation and semi-global matching.](image1)

![Fig. 4. (a) Left input image of a real-world scenario (taken from [13]) and (b) gray-scale representation of the disparity map.](image2)
enable massively parallel computations and maximum data reuse throughout the entire stereo image application. Since the maximum external bus width is 128 bit, special load/store operations have been implemented. A VR_RL128 (ring load) instruction first shifts left the register content by 128 bits and then loads a 128-bit word from memory into to the lower part. Four of these loads result in 512 consecutive bits from memory in the register. A VR_IL128 (insert load) simply replaces the lower 128 bits of the register file, which is interesting when data is consumed subword by subword by the algorithm from the high end of the register. Additionally, the combination of these access schemes allows to process data over the boundaries of word-alignment making stream-based data processing very efficient. This is used e.g. for the implementation of the rank transform and the median filter.

The functional unit for the VR_RANK_ROW instruction calculates the partial rank-transform of one kernel row for 16 pixels (128 bits) in parallel. A code snippet with equivalent code examples is given in Fig. 7a. Processing of the horizontal image borders is included in the functional unit (selectable via a mode switch) as it only requires very few additional 8-bit multiplexers and greatly simplifies software implementation. Not using the full 512 bits available in the register file for this operation has two reasons. First, it is a trade-off between acceleration and additional area-requirement. Secondly, it keeps the balance between the number of load/store operations and the number of data processing operations. Since the external bus width is limited to 128 bit, the number of load/store operations cannot be further reduced and processing more pixels in parallel results in an implementation that is overly dominated by the load/store operations. This would result in only minor further acceleration at the expense of much higher silicon area requirement.

The SGM_CALC_PC operation calculates the path costs of the SGM according to (4) for all disparity levels of one path of one pixel in parallel. The operation includes a total of 446 arithmetic basic operators and is scheduled over two execution stages. The architecture is shown in Fig. 6 and its usage is shown in Fig. 7b. The maximum disparity that can be computed with one instruction call is limited by the width of the register file (512 bits) and is therefore 64 pixels. It is the
biggest functional unit in terms of silicon area but accounts for the most significant single speed-up. The number of required processing cycles (including load/store of intermediate results) is reduced from 1.0 billion to 4.7 million which is a speed-up of factor 220.

The SGM_CALC_P2 operation calculates the adaptive penalty term \( P_2 \) for all four paths and stores the results in a dedicated state to make them available to the SGM_CALC_PC operation. Using a dedicated state is necessary since the maximum number of register inputs is reached for the SGM_CALC_PC operation. This additional instruction became necessary as the calculation with basic instructions consumed disproportionately many cycles compared the other parts of the accelerated implementation.

The MEDIAN_3x3 operation calculates the local median values in a 3-by-3 pixel window for four horizontally neighboring center pixels (see Fig. 7d). The functional unit implementation is based on the bit-slice approach of Danielsson [15] and its hardware implementation in [16]. To maintain a reasonable hardware cost vs. performance ratio only four median processing units with nine 8-bit input values each, are used in parallel. Thus, four consecutive calls are needed to process 16 neighboring pixels. An offset position within the input vector registers is passed to the functional unit via an immediate value. The horizontal border processing is hereby implemented similar to the rank transform. Using two additional 8-bit multiplexers per input register, the rightmost or leftmost pixel can be repeated to simplify the software implementation. Due to the restrictions imposed by the limited number of access ports of the vector register file, the result is written back to the upper 128 bits of one of the 512-bit input registers, which are no longer needed for further calculation.

To perform a fair comparison, the SW reference code without the median special functional unit was optimized applying massive data reuse. The pseudo code from Fig. 7d serves only as a functional description.

An additional uncached local 64 kB on-chip memory is directly connected to one of the four available data RAM ports of the base processor. It allows single-cycle access while re-using the same load/store operations described above. The memory is beneficial in cases where intermediate results have medium life-span and high data bandwidth is required. This is the case with the path costs of the semi-global matching which require 3-512 bit storage space per input pixel and are only required again at the same pixel position in the next row. By using the local memory rather than the external memory and the cache the processing time of the SGM algorithm is reduced from 52.6 million cycles to 16.4 million cycles (assuming a delay of 10 cycles for a read cache miss). For further functionality and other algorithms this general purpose memory could be extended to the functionality of a DMA unit.

These enhancements to the base processor allow a very efficient stream-based implementation of the disparity estimation algorithm despite the non-scan aligned directions of the path costs. The input images are processed row-wise starting from the top-left corner and every pixel is passed only once. All four path directions are processed within that single pass. The local on-chip memory is used as a ring buffer for the path cost values in a similar fashion to the intermediate buffer in [4]. All operations, except control operations, are performed using subword parallelism on the vector register file. The final computation of the minimum from the path costs (including the uniqueness check) does not require an additional functional unit. The example code is shown in Fig. 7c. The uniqueness check is performed implicitly. The HMIN8U operation sets a dedicated single-bit state indicating the uniqueness of the found minimum. This state (selected via a bit-mask) is checked by the PACK8_COND operation. Due to the round-to-zero functionality of the ADD8_SR operation, the uniqueness threshold is 6. The threshold of 6 is an empirically
The general purpose 90nm CMOS process technology from TSMC is presented in Table II. The total gate count for the base processor and the enhanced processor have been obtained after synthesis with Synopsis Design Compiler v2009. Due to the presented architectural extensions a total area increase from 128,539 gates to 375,294 gates is observed, which is a factor of 2.9. The detailed breakdown of the gate count relies on pre-synthesis estimates provided by the Tensilica tools. It shows that the most significant part is accounted for by the general purpose vector register file, which requires approximately half of the additional gates. The algorithm-unspecific general media processing instructions account for approximately 11 %, and the algorithm-specific functional units for only 14 % of the total area requirement. The extended instruction set does not impair the maximum clock frequency of the processor. The difference between the pre-synthesis estimates (using the Tensilica Tools) and the post-synthesis results (using Synopsis Design Compiler) is constant and is between 10 % and 13 %.

Table III presents the cycle counts required by the implementation of the algorithm on the base processor and the enhanced processor. Utilizing all the mechanisms presented in Section III the obtained speed-up is 188 for the rank transform, 135 for the SGM and 74 for the median filter. The overall speed-up for the complete stereo matching is 137. Further, the influence of a slower system has been modeled by increasing the read latency of the memory. Read latency is used since in firstly HW-cost vs. performance and secondly power consumption. However, despite the fact that the GPU and CPU implementation compute eight and 16 paths, an essential difference in firstly HW-cost vs. performance and secondly power consumption.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Base Processor</th>
<th>Enhanced Processor</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>100,235,587</td>
<td>1,027,984</td>
<td>98</td>
</tr>
<tr>
<td>SGM</td>
<td>3,000,768,000</td>
<td>17,180,000</td>
<td>175</td>
</tr>
<tr>
<td>Median</td>
<td>17,407,991</td>
<td>744,641</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>3,218,847,165</td>
<td>19,980,609</td>
<td>161</td>
</tr>
</tbody>
</table>

The equivalent gate count (without memories) for using the general purpose 90nm CMOS process technology from TSMC (TSMC 90nm-G) is presented in Table II. The total gate count for the base processor and the enhanced processor have been obtained after synthesis with Synopsis Design Compiler v2009. Due to the presented architectural extensions a total area increase from 128,539 gates to 375,294 gates is observed, which is a factor of 2.9. The detailed breakdown of the gate count relies on pre-synthesis estimates provided by the Tensilica tools. It shows that the most significant part is accounted for by the general purpose vector register file, which requires approximately half of the additional gates. The algorithm-unspecific general media processing instructions account for approximately 11 %, and the algorithm-specific functional units for only 14 % of the total area requirement. The extended instruction set does not impair the maximum clock frequency of the processor. The difference between the pre-synthesis estimates (using the Tensilica Tools) and the post-synthesis results (using Synopsis Design Compiler) is constant and is between 10 % and 13 %.

Table III presents the cycle counts required by the implementation of the algorithm on the base processor and the enhanced processor. Utilizing all the mechanisms presented in Section III the obtained speed-up is 188 for the rank transform, 135 for the SGM and 74 for the median filter. The overall speed-up for the complete stereo matching is 137. Further, the influence of a slower system has been modeled by increasing the read latency of the memory. Read latency is used since in firstly HW-cost vs. performance and secondly power consumption. However, despite the fact that the GPU and CPU implementation compute eight and 16 paths, an essential difference in firstly HW-cost vs. performance and secondly power consumption.

A performance comparison of different implementations of the semi-global matching algorithm is presented in Table IV. The chosen references present their results with different frame sizes, number of paths, and employed matching cost algorithms. Thus, performing an accurate quantitative performance comparison (e.g. based on normalized throughput) is a complex problem. Even though basic operation count scales linearly, the throughput does not necessarily scale linearly because of the dependency on the type of employed parallelization.

### Table II

<table>
<thead>
<tr>
<th>Hardware extension</th>
<th>Area [Gates]</th>
<th>Area [mm²]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced ASIP core</td>
<td>375,294</td>
<td>0.95</td>
</tr>
<tr>
<td>LX2 base core</td>
<td>128,539</td>
<td>0.32</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Base Processor</th>
<th>Enhanced Processor</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>95,020,000</td>
<td>504,680</td>
<td>188</td>
</tr>
<tr>
<td>SGM</td>
<td>2,079,716,000</td>
<td>15,404,000</td>
<td>135</td>
</tr>
<tr>
<td>Median</td>
<td>16,502,196</td>
<td>222,338</td>
<td>74</td>
</tr>
<tr>
<td>Total</td>
<td>2,286,258,196</td>
<td>16,635,698</td>
<td>137</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Base Processor</th>
<th>Enhanced Processor</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>98,016,446</td>
<td>835,372</td>
<td>117</td>
</tr>
<tr>
<td>SGM</td>
<td>2,593,820,000</td>
<td>16,428,000</td>
<td>158</td>
</tr>
<tr>
<td>Median</td>
<td>17,022,591</td>
<td>550,521</td>
<td>31</td>
</tr>
<tr>
<td>Total</td>
<td>2,806,875,482</td>
<td>18,649,265</td>
<td>151</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Base Processor</th>
<th>Enhanced Processor</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>100,235,587</td>
<td>1,027,984</td>
<td>98</td>
</tr>
<tr>
<td>SGM</td>
<td>3,000,768,000</td>
<td>17,180,000</td>
<td>175</td>
</tr>
<tr>
<td>Median</td>
<td>17,407,991</td>
<td>744,641</td>
<td>23</td>
</tr>
<tr>
<td>Total</td>
<td>3,218,847,165</td>
<td>19,980,609</td>
<td>161</td>
</tr>
</tbody>
</table>
TABLE IV
PERFORMANCE COMPARISON OF SGM IMPLEMENTATIONS ON DIFFERENT HARDWARE PLATFORMS. THE DIFFERENT MATCHING COST FUNCTIONS ARE RANK TRANSFORM (RT) AND BIRCHFIELD-TOMASI (BT). (MF) DENOTES THE MEDIAN FILTER.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Frame Size (Disp. range)</th>
<th>Performance fps @ MHz</th>
<th>Algorithm (Paths) Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced Tensilica LX2</td>
<td>640×480 (64)</td>
<td>20 fps 373 MHz</td>
<td>RT+SGM+MF (4) This work</td>
</tr>
<tr>
<td>Parameterized Media VLIW</td>
<td>640×480 (64)</td>
<td>30 fps 400 MHz</td>
<td>RT+SGM (4)</td>
</tr>
<tr>
<td>Nvidia Quadro FX5600</td>
<td>450×375 (64)</td>
<td>5.85 fps 2.8 GHz</td>
<td>BT+SGM (8)</td>
</tr>
<tr>
<td>AMD Opteron</td>
<td>450×375 (64)</td>
<td>0.56 fps 2.8 GHz</td>
<td>HMI+SGM+MF (16)</td>
</tr>
<tr>
<td>Virtex-5 FPGA</td>
<td>640×480 (64)</td>
<td>66 – 167 fps 133 MHz</td>
<td>RT+SGM+MF (4)</td>
</tr>
</tbody>
</table>

The high performance for SGM computation could be achieved while maintaining the flexibility of the enhanced processor for general media processing by implementing general hardware extensions, such as basic arithmetic SIMD-operations and a wide vector register file, as well as a small, but highly specialized set of algorithm-specific instructions. In total, the silicon area requirement increases only by factor 2.9, resulting in an equivalent gate count of 375,294 for the general purpose TSMC 90nm-G CMOS process technology. Stereo processing under real-time conditions, such as 640x480-pixel images at 20 frames per second, can be achieved at a clock frequency as low as 373 MHz, which can easily be achieved using the high performance TSMC 90nm-GT process technology.

ACKNOWLEDGMENTS

The authors wish to acknowledge the contributions of coworkers at Pro Design Electronic GmbH. This work has been financially supported by the BMBF (Bundesministerium für Bildung und Forschung), Germany.

REFERENCES